ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
«ҒҰМАРБЕК ДАУКЕЕВ АТЫНДАҒЫ АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
коммерциялық емес акционерлік қоғамы
IT-инжиниринг кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
Кафедра меңгерушісі
PhD, доцент А.А. Досжнова
_____ «___» _____ 2020 ж.

ДИПЛОМДЫҚ ЖОБА

Тақырыбы: Жасанды нейрон желілерін пайдалана отырып, спорттық жарыстардың нәтижелерін болжауға арналған ақпараттық жүйені құру
Мамандығы: 5В070300 – «Ақпараттық жүйелер»
Орындаған: Төлебай Қ.Қ. Тобы: ИСк-16-1
(Аты-жөні)
Ғылыми жетекші: PhD, доцент Кожамкулова Ж.Ж.
(ғылыми дәрежесі, атағы, Т.А.Ж)

Кеңесшілер:
Экономикалық бөлім:
э.ғ.к., асс. профессор Габелашвили К.Р. _____ «___»_____2020 ж.
(ғылыми дәрежесі, атағы, Т.А.Ж) (қолы)

Өміртіршілік қауіпсіздігі:
аға оқытушы Бегимбетова А.С. _____ «___»_____2020 ж.
(ғылыми дәрежесі, атағы, Т.А.Ж) (қолы)

Есептеу техникасын қолдану:
аға оқытушы Айткулов Ж.С. _____ «___»_____2020 ж.
(ғылыми дәрежесі, атағы, Т.А.Ж) (қолы)

Норма бақылаушы:
аға оқытушы Абсатарова Б.Р. _____ «___»_____2020 ж
(ғылыми дәрежесі, атағы, Т.А.Ж) (қолы)

Сын-пікір беруші:
_____ _____ «___»_____2020 ж.
(ғылыми дәрежесі, атағы, Т.А.Ж) (қолы)

Алматы 2020

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ БІЛІМ ЖӘНЕ ҒЫЛЫМ МИНИСТРЛІГІ
Коммерциялық емес акционерлік қоғамы
«ҒҰМАРБЕК ДӘУКЕЕВ атындағы АЛМАТЫ ЭНЕРГЕТИКА ЖӘНЕ БАЙЛАНЫС УНИВЕРСИТЕТІ»
Басқару жүйелері және ақпараттық технологиялар институті
Ақпараттық жүйелер мамандығы
«IT – инжиниринг» кафедрасы

Дипломдық жобаны орындауға берілген

ТАПСЫРМА

Студент: Төлебай Қайырхан Қанатұлы
Жоба тақырыбы: Жасанды нейрон желілерін пайдалана отырып, спорттық жарыстардың нәтижелерін болжауға арналған ақпараттық жүйені құру.

«__» _____ 20____ ж. №_____ университет бұйрығы бойынша бекітілген.

Аяқталған жұмысты тапсыру мерзімі: «_____» _____2020 ж.

Жобаға бастапқы деректер (талап етілетін жоба нәтежелерінің параметрлері және нысанның бастапқы деректері): Бұл дипломдық жоба Python тілін пайдаланып, спорттық ойындардың нәтижелерін болжамдайтын ақпараттық жүйені құру.

Диплом жобасындағы әзірленуі тиіс сұрақтар тізімі немесе диплом жобасының қысқаша мазмұны:

а) Пәндік саланы талдау;
б) Жобалау бөлімі;
в) Программалық қамтаманы құру;
г) Экономикалық бөлім;
д) Өміртіршілік қауіпсіздігі;
е) А қосымшасы. Программа мәтіні;

Негізгі ұсынылатын әдебиеттер:

1 Маркин, А.В. PHP web-бағдарламалау негіздері: учеб. пособие / А.В. Маркин. – М.: Диалог-МИФИ, 2012 жыл.

2 Когтзолл Д. PHP5. Толық жетекшілік. – М.: Вильямс, 2010.

3 CSS туралы оқулық – Электронды оқулық: http://www.wisdomweb.ru/CSS.

4 HTML туралы оқулық – Электронды оқулық: https://html5book.ru/

Жоба бойынша бөлімшелерге қатысты белгіленетін кеңесшілер

Бөлімшелер	Кеңесшілер	Мерзімі	Қолы
Экономикалық бөлім	Габелашвили К.Р.	15.04.2020 – 30.04.2020	
Өміртіршілігі қауіпсіздігі	Мусаева Ж.К.	15.04.2020 – 30.04.2020	
Бағдарламалық қамтама	Айткулов Ж.С.	13.05.2020 – 18.05.2020	
Норма бақылау	Абсатарова Б.Р.	13.05.2020 – 18.05.2020	

Диплом жобасын дайындау
КЕСТЕСІ

№ р/с	Тарау аттары, әзірленетін сұрақтардың тізімі	Жетекшіге ұсыну мерзімдері	Ескерту
1	Теориялық бөлім	17.02.2020 - 16.03.2020	
2	Бағдарламалық қосымшаны жобалау	17.03.2020 - 05.04.2020	
3	Қосымша әзірлеу бөлімі	06.04.2020 - 09.05.2020	

Тапсырманың берілген уақыты «___» _____ 20____ж.

Кафедра меңгерушісі _____ Досжанова А.А.

Жоба жетекшісі _____ Кожамкулова Ж.Ж.

Орындалатын тапсырманы
қабылдаған студент _____Төлебай Қ. Қ.

АҢДАТПА

Бұл дипломдық жоба нейрондық желілерді қолдана отырып, спорт ойындарының нәтижелерін болжауға арналған.

Программа толығымен Python программалау тілінде жазылған. Көптеген қосымша модульдер қолданылды.

Нәтижені қолайлы түрде алу үшін Телеграм мессенджерінде бот жасалды. Телеграммен жұмыс істеу үшін telegram модулі қолданылды.

Нейрондық желіні жасап, жаттықтыру үшін sklearn модулі пайдаланды.

АННОТАЦИЯ

Данный дипломный проект предназначена для прогнозирования исходов спортивных состязаний с использование нейронных сетей.

Программа полностью написана на языке программирования Python. Было использовано множество модулей.

Чтобы пользователю было удобно работать с моделью, был написан телеграм-бот. Для его работы был использован модуль telegram.

Для того чтобы создать и тренировать нейронную сеть был использован модуль sklearn.

ANNOTATION

This diploma project was written to predict sports using neural networks

Program is fully written in Python programming language. Many of modules were used.

For the comfort of user was written a telegram-bot. For its operation was used telegram module.

To create and train our neural network was used sklearn module.

Мазмұны

Кіріспе

Компьютерлік нейрондық желілер идеясы бұрыннан бар, тіпті өткен ғасырда компьютерлік нейрондық желілер жұмысының мысалдары болды. Бұл технология әдеттегі мидың құрылысына негізделген, ол оны симуляциялайды. Біздің миымызда нейрондар бар және олар өзара байланысты.

Нейрондық желі - жасанды интеллекті жүзеге асыру тәсілдерінің бірі.

Нейрондық желі адамзат жүйке жүйесінің жұмысын моделдейді, оның ерекшелігі алдыңғы тәжірибені ескере отырып, өзін-өзі оқыту қабілеті болып табылады. Осылайша, жүйе әр жолы аз қателерді жасайды.

2013 жылы OxfordMartin School жұмысында барлық жұмыс орындарының 47% келесі 20 жыл ішінде автоматтандырылуы мүмкін деп айтылды. Бұл процестің негізгі драйвері адамға тиімді ауыстыру ретінде үлкен деректермен жұмыс істейтін жасанды интеллектті қолдану болып табылады.

Біздің жүйке жүйесі сияқты нейрондық желі бөлек есептеу элементтерінен тұрады – Нейрон. Олар бірнеше қабатта орналасқан. Нейрожеліге берілетін деректер оның әр қабатында дәйекті өңделеді. Сонымен қатар, әрбір нейронның белгілі бір параметрлері бар, олар алынған нәтижелерге байланысты өзгеруі мүмкін – бұл желіні оқыту болып табылады.

Нейрожелінің міндеті - мысықтарды иттерден ажырату болсын. Нейрондық желіні реттеу үшін мысықтар мен иттердің суреттерінің үлкен массиві беріледі, ол массивте әр суретте ит немесе мысықтың суреті екенін көрсететін белгі болады. Нейросеть осы суреттердегі белгілерді (оның ішінде сызықтарды, пішіндерді, олардың өлшемі мен түсін) талдайды және эталондық нәтижелерге қатысты қателер пайызын азайтатын тану моделін жасайды.

Нейрожелілер жақында ғана жалпыға ортақ назар орталығына түскеніне қарамастан, бұл машиналық оқытудың көне алгоритмдерінің бірі. Алғаш 1943 жылы Уоррен Маккаллок және Уолтер Питтс ұсынған.

Ал 1958 жылы Фрэнк Розенблатт бірінші нейрондық желіні әзірледі. Өзінің қарапайымдылығына қарамастан, ол екі өлшемді кеңістіктегі нысандарды ажырата алды.

1 Нейрондық желілер тарихы мен қолданылуы

1.1 Нейрондық желілер тақырыбының өзектілігі

Жалпы нейрондық желілер технологиясы өзекті. Менің жұмысым нейрондық желілерді пайдаланудың мүмкін тәсілдерінің бірін ғана көрсетеді. Оларды кез келген мақсатта қолдануға болады.

Нейрожелілерді пайдалану кез келген қызметтің тиімділігін арттырады. Қазіргі таңда барлық ірі компаниялар оларды өз жұмысында қолданады.

Прогрестің жаңа кезеңі жиі қорқытады. Алайда, нейрондық желі негізінде жұмыс істейтін программалар адамдардын орнын толықтай баса алмайды, керісінше, нейрондық желі өндеген ақпаратпен жұмыс істеу әлдеқайда тиімді. Компьютерлер даму кезеңінде де адамдарда өз жұмыс орны үшін қорқыныш болды, бірақ қазіргі уақытта компьютерсіз жұмысты елестету мүкін емес. Бүгінгі күні алдын ала анықталған нәтижемен қандай да бір операцияларды жасайтын және оны ұмытатын бағдарламалар мен машиналардан біз бірте-бірте тәжірибе жинақтап, адам сияқты тәжірибе жинақтап, үйренетін бағдарламаларға көшудеміз.

Нейрожелілердің қысқаша тарихы

Нейрондық желілер идеясы мидағы нейрондар "коннекционизм" деп аталатын және интеллектуалды мінез-құлықты модельдеу үшін байланысты тізбектерді қолданатын модель ретінде пайда болғаны таңқаларлық емес .1943 жылы нейрофизиолог Уоррен Маккалох және математик Уолтер Питтс оны қарапайым электр схемасының көмегімен сипаттады. Дональд Хебб бұл идеяны өзінің "мінез-құлықты ұйымдастыру" (1949) кітабында одан әрі дамытып, нейрондық жолдар әрбір келесі пайдалану кезінде, әсіресе бір мезгілде жұмыс істеу үрдісі бар нейрондар арасында күшейіп, осылайша мидың күрделі процестерін сандық бағалауға ұзақ жолды бастап.

Нейрондық желілердің ізашары болып табылатын екі негізгі ұғымдар:

- шекті логика - үздіксіз кіріс сигналын дискретті Шығыс сигналына түрлендіру;

- Хеббиандық оқыту-нейрондық икемділікке негізделген, Дональд Хебб оның "мінез-құлықты ұйымдастыру" кітабында ұсынған оқыту моделі".

1950-ші жылдары зерттеушілер бұл желілерді есептеу жүйелеріне көшіруге тырысты. Бірінші Геббиан желісі 1954 жылы Массачусетск технологиялық институтында сәтті іске асырылды.

Шамамен сол уақытта Корнель университетінің психологы Фрэнк Розенблатт шыбын көзіндегі шешімдерді қабылдаудың салыстырмалы қарапайым жүйелерін түсінумен жұмыс істеді. Бұл процесті түсіну және сандық бағалау талпынысында ол 1958 жылы персептрон идеясын ұсынды. Маккаллок-Питтс нейроны кіріс деректерін қабылдайды, өлшенген соманы қабылдайды және егер нәтиже шекті мәннен төмен болса '0" және керісінше жағдайда " 1 " қайтарады.

Mark I персептронының керемет оның салмақ коэффициенттерін қалаған және нақты шығудың арасындағы айырмашылықты азайтып, дәйекті түрде берілетін кіріс деректерінің көмегімен "үйренуге" болатын.

Осы идеяның басты кемшілігі, бұл персептрон қарапайым, бірақ сызықсыз эксклюзив-немесе еңсерілмейтін кедергі тізбегін жасай отырып, тек сызықты бөлінген сыныптарды бөлуді үйренуі мүмкін.

Мидан басқа шешім қабылдау жүйелерін сандық бағалау үшін машиналық оқытуды пайдаланудың ретсіз және біршама қанағаттанғысыз пайда болғанына қарамастан, заманауи жасанды нейрондық желілер осы перцептрондардың бірнеше қабатынан ғана тұрады.

Шамамен сол уақытта нейрондық желілердің жылдам дамуы басталды, және 1959 жылы Бернард Видроу Стэнфорд және марсиан Хофф нақты проблемаға табысты қолданылған алғашқы нейрондық желіні әзірледі. Бұл жүйелер бірнеше адаптивті желілік элементтерді пайдалану құрметіне ADALINE және MADALINE аталды, оның соңғысы телефон желілерінде шуды жою үшін арнайы әзірленген және әлі күнге дейін пайдаланылады. Бұл жасанды нейрондар, алайда, персептрондардан бұл жағдайда өлшенген кіріс сигналы болған шығу сигналы ретінде қайтарғанымен ерекшеленді.

Нейрондық желілерді инженерлік пайдалану

Инженерия-бұл нейрондық желілерді қолдану саласы, әсіресе ұшуларды басқаруды, химиялық машина жасауды, Энергетикалық қондырғыларды, автомобиль басқаруды, медициналық жүйелерді және автономияны талап ететін басқа да жүйелерді қоса алғанда, түрлі салаларда пайда болған "жоғары сенімділік жүйелерінде".

Нейрондық жүйе сарапшылардың көзқарасымен

Төменде инженерлік сектордағы екі сарапшының олардың қосымшалары бөлшек сауданы, өндірісті, мұнай мен газды, навигацияны және офистік ортадағы ақпаратты іздеуді қалай жақсартатыны туралы пікірі келтірілген.

Адамдар сымсыз технологияны пайдаланады, ол құрылғыларға Интернетке қосылуға немесе белгілі бір аймақ шегінде бір-бірімен қарым-қатынас жасауға мүмкіндік береді, шығындарды азайту және тиімділікті арттыру үшін әр түрлі салаларда. Хью Риз, Kodak Cloud сату және маркетинг жөніндегі вице-президенті, Wi - Fi өнімділігін оңтайландыруға арналған қосымшалар тек кейбір пайдалану түрлерін сипаттайды.

Риз Wi-Fi қолдануының бірнеше күнделікті мысалдарын ұсынады: "супермаркеттер желілері Wi-Fi сканерлерін өз тарату орталықтарында және жеке нарықтарда өнімді сканерлеу үшін пайдаланады. Егер Wi-Fi жақсы жұмыс істемесе, бүкіл кәсіпорындар бұзылады. Өндірістік және мұнай-газ концерндері Wi - Fi маңызды миссия болып табылатын кәсіпорындардың жақсы мысалдары болып табылады, өйткені сенімділік пен оңтайландыруды қамтамасыз ету абсолютті талап болып табылады", - дейді ол [1].

Wi-Fi-бұл керемет, бірақ ол өз жұмысын жасау үшін үлкен қадағалауды талап етеді. "Көптеген корпоративтік немесе ірі масштабты сымсыз жергілікті

желілік шешімдер Wi-Fi бойынша жоғары білікті мамандардың тұрақты мониторингін және баптауын талап етеді, бұл желінің оңтайлы жұмысын қамтамасыз етудің қымбат тәсілі болып табылады", - деп атап өтті Риз. "Kodak Cloud бұл проблеманы алгоритмдер мен өзін - өзі жетілдіру циклін жасайтын бейімделген оқытуды пайдаланатын зияткерлік жүйе арқылы шешеді", - деп толықтырды ол.

Риз Kodak бұлттық технологиясы нейрондық желілердің артықшылықтарын үнемі жетілдіру үшін пайдаланатындығымен бөліседі: "желі жаһандық және жергілікті оқыту негізінде оқиды және өзін-өзі емдейді. Мұнда жаһандық мысал: жүйе жаңа Android операциялық жүйесі кеңейтілгенін және оңтайлы жұмыс үшін конфигурация мен табалдырықты қосымша өзгертуді талап ететінін біледі. Жүйе осы жаңарту үшін қажетті түзетулер мен өлшеуіш жақсартулар енгізгеннен кейін, ол осы білімді KodaCloud барлық басқа клиенттеріне бірден қолданады, сондықтан жүйе кез келген осындай құрылғыны бірден танып, проблемаларды шешеді. Жергілікті мысал үшін, жүйе әрбір кіру нүктесі үшін жергілікті радиожиілік ортасын зерттейді. Содан кейін әрбір құрылғы әрбір кіру нүктесіне қосылады, бұл жергілікті құрылғының радиопараметрлерінің шекті өзгеруіне әкеледі. Бұл процесс жаһандық және жергілікті әрбір құрылғы үшін Wi-Fi сапасын оңтайландырудың үздіксіз циклі болып табылады."

Тез дамып келе жатқан технология, ұшқышсыз ұшу аппараттары дүлей зілзалалардың салдарын жою, мұнай, газ және пайдалы қазбаларды барлау, әуе бақылауы, жылжымайтын мүлік және құрылыс, сондай-ақ кино өндірісі кезінде пайдаланылады. Нил Макарон-Кэмпбелл Aeiou бас директоры.көптеген түрлі секторларда пайдалану үшін ұшқышсыз ұшу аппараттарының озық технологияларын әзірлейтін технология. "Біздің Dawn платформасы датчиктердің борттық сериясын және кез келген ұшқышсыз ұшу жүйесіне навигация және кедергілерді болдырмау сияқты бортмеханиктен Инфрақұрылым инспекциясы немесе пакеттерді жеткізу сияқты қызметтерге дейін ұсынатын артықшылықтардың кең спектрін қолдануға мүмкіндік беретін ілеспе компьютер болып табылады",-дейді Макоран-Кэмпбелл.

Макарон-Кэмпбелл DAWN биологияның екі деңгейі негізінде қалай жұмыс істейтінін түсіндіреді: "бірінші деңгейде Энн шикі ақпаратты өңдеу үшін пайдаланамыз. Біз пайдаланатын желілердің үш түрлі түрі бар: болашақ туралы болжамдарды алу үшін өткенді пайдаланатын рекуррентік нейрондық желілер; нейрондардың "жылжымалы" шоғырын пайдаланатын ұюды нейрондық желілер (біз әдетте суреттерді өңдеу үшін осы түрін пайдаланамыз); және қарапайым нейрондық желілер, яғни нақты нейрондық желілер. Қарапайым нейрондық желілер навигация сияқты тапсырмалар үшін өте пайдалы,әсіресе олар рекуррентік элементтермен біріктірілген.

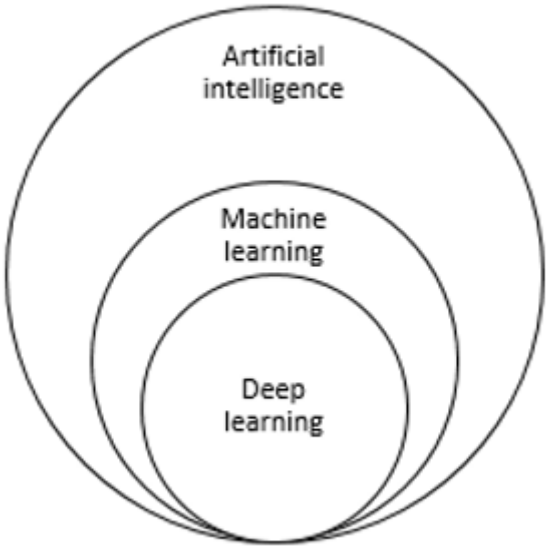
Неғұрлым күрделі, екінші деңгейде Dawn құрылымы ақпаратты өңдеу үшін бар ең жақсы архитектураны имитациялайды: адам миы. Бұл бізге автономияның өте күрделі проблемасын биология сияқты жасауға мүмкіндік береді: бөлінген "кортекспен", олардың әрқайсысының өзінің нейрондық

желілері бар және әрқайсысы өзінің коммуникациялық жолдары мен иерархиялық командалық құрылымдары бар. Нәтижесінде толқындармен ақпарат мидың қабығы арқылы бас миы сияқты өтеді.

Төменде әр түрлі салаларда қазіргі уақытта пайдаланылатын нейрондық желілерді дамыту үшін басқа қолданбалар тізімі:

- *Аэроғарыш саласы:* әуе кемелері компоненттерінің ақауларының детекторлары және оларды моделдеу, әуе кемелерін басқару жүйелері, жоғары тиімді автопилотирлеу және ұшу траекториясын модельдеу;
- *Автоматтандыру өнеркәсібі*: жетілдірілген бағыттау жүйесі, қуатты агрегаттарды, виртуалды датчиктер мен кепілдік белсенділікті талдағыштарды әзірлеу;
- *Электроника*: микросхемалардың істен шығуын талдау, микросхемалардың орналасу сұлбасы, машинамен көру, сызықты емес модельдеу, кодтық бірізділікті болжау, технологиялық процесті басқару және дауыс синтезі;
- *Өндіріс*: химиялық өнімнің дизайнын талдау, химиялық процестер жүйесін динамикалық модельдеу, технологиялық процестерді басқару, процестер мен машиналардың диагностикасы, өнімді жобалау және талдау, Қағаз сапасын болжау, жобалық сауда-саттық, жоспарлау және басқару, компьютерлік чиптердің сапасын талдау, сапаны бақылау және дәнекерлеу сапасын талдау;
- *Механика*: жай-күйінің мониторингі, жүйелерді моделдеу және оларды басқару;
- *Робототехника:* шанышқы Роботтар, манипуляторлардың контроллерлері, траекторияны басқару және визуалды бақылау жүйелері;
- *Телекоммуникациялар:* банкоматтар желісін басқару, автоматтандырылған ақпараттық қызметтер, клиенттердің төлемдерін өңдеу жүйелері, деректерді сығу, эквалайзерлер, істен шығуды басқару, қолжазба мәтінін тану, желіні жобалау, басқару, бағыттау және бақылау, желі мониторингі, нақты уақытта сөйлесу сөзін аудару және бейнелерді тану (тұлғалар, объектілер, саусақ іздері, семантикалық талдау, орфографияны тексеру, сигналдарды өңдеу және сөйлеуді тану).

Нейрондық желілер, терең оқыту, машиналық оқыту және жасанды интеллект арасындағы айырмашылықтар туралы



1.1 сурет - Жасанды интелекттің компоненттерін сипаттайтын Венн диаграмасы.

Жасанды интеллект, машиналық оқыту және терең оқыту қазір көптеген кәсіпорындардың ажырамас бөлігі болып табылады. Жиі бұл терминдер синонимдер ретінде қолданылады [12].

Жасанды интеллект үлкен қадамдармен қозғалуда — пилотсыз көлік құралдары саласындағы жетістіктер мен покер және Мо сияқты ойындарда адамды жеңу қабілеттерінен, клиенттерге автоматты қызмет көрсету. Жасанды интеллект-бұл бизнесте революция жасауға дайын озық технология.

Жиі терминдер жасанды интеллект, машина оқыту және терең оқыту өзара алмасатын ретінде жүйесіз пайдаланылады, бірақ шын мәнінде, олардың арасында айырмашылықтар бар. Бұл терминдердің айырмашылығы төменде қарастырылады.

Жасанды интеллект (ЖИ)

Жасанды интеллект-озық машина интеллектіне қатысты кең түсінік. 1956 жылы Дартмутта жасанды интеллект бойынша конференцияда бұл технология былайша сипатталған: "оқытудың әрбір аспектісі немесе ақыл-ойдың кез-келген басқа ерекшелігі машина оларды біріктіре алатындай етіп сипатталуы мүмкін.»

Жасанды интеллект кез келген нәрсеге қатысты болуы мүмкін — шахмат ойнауға арналған компьютерлік бағдарламалардан сөйлеуді тану жүйесіне дейін, мысалы, сөйлеуді қабылдауға және сұрақтарға жауап беруге қабілетті Amazon Alexa дауыстық көмекшісі сияқты. Жасанды интеллект

жүйесін жалпы үш топқа бөлуге болады: шектеулі жасанды интеллект (Narrow AI), жалпы жасанды интеллект (AGI) және өте ақылды жасанды интеллект.

1996 жылы Гарри Каспаровты шахматқа жеңіп шыққан IBM компаниясының Deep Blue бағдарламасы немесе 2016 жылы Го Ли Седольден әлем чемпионын жеңіп алған Google DeepMind компаниясының AlphaGo бағдарламасы бір нақты міндетті шешуге қабілетті шектеулі жасанды интеллектінің үлгісі болып табылады. Бұл адам интеллектімен бір деңгейде тұрған және әртүрлі тапсырмаларды орындай алатын жалпы жасанды интеллект (AGI) оның басты айырмашылығы.

Өте ақылды жасанды интеллект адамнан жоғары сатыға тұрады. Бостром Ник оны былай сипаттайды: "ақыл-ой, ол ең жақсы адам миына қарағанда әлдеқайда ақылды, барлық салаларда, соның ішінде ғылыми шығармашылықта, жалпы даналықта және әлеуметтік дағдыларда."Басқаша айтқанда, бұл машиналар бізден әлдеқайда ақылды болады [2].

Машиналық оқыту

Машиналық оқыту жасанды интеллект бағыттарының бірі болып табылады. Негізгі принцип машиналар деректерді алады және сол деректерге сүйеніп "оқиды". Қазіргі уақытта бұл жасанды интеллектке негізделген бизнес үшін ең перспективалы құрал. Машиналық оқыту жүйесі үлкен деректер жинағында оқыған кезде алған білімдерін тез қолдануға мүмкіндік береді, бұл оларға адамдарды тану, сөйлеуді тану, объектілерді тану, аударма және т.б. сияқты міндеттерді табысты атқаруға мүмкіндік береді. Нақты міндеттерді орындау үшін қолмен кодталған нұсқаулықтары бар бағдарламаларға қарағанда, машиналық оқыту жүйеге үлгілерді өз бетінше танып білуге және болжам жасауға мүмкіндік береді.

Екі бағдарлама да — Deep Blue және DeepMind жасанды интеллект пайдалану мысалдары болып табылады, Deep Blue алдын ала бағдарламаланған ережелер жиынтығында салынған, сондықтан ол машиналық оқытумен байланысты емес. Екінші жағынан, Deep Mind машина оқыту үлгісі болып табылады: бағдарлама тәжірибелі ойыншылардың жасаған осы қадамдардың үлкен жиынтығында өздерін үйрете отырып-ақ бойынша әлем чемпионын жеңді.

Терең оқыту

Терең оқыту машиналық оқытудың ішкі жиыны болып табылады. Ол адамның шешім қабылдауын имитациялайтын нейрондық желілерді пайдалана отырып, нақты тапсырмаларды шешу үшін машиналық оқытудың кейбір әдістерін қолданады. Терең оқыту қымбат болуы мүмкін және оқыту үшін үлкен деректерді талап етеді. Бұл жалған жұмыс істемеуді болдырмау үшін оқыту алгоритмдері үшін теңшеу қажет көптеген параметрлер бар екенін түсіндіреді. Мысалы, терең оқыту алгоритмдеріне мысықтың түр-сипатын "білу" деген нұсқау берілуі мүмкін. Оқыту үшін мысықты гепард, пантера немесе түлкіден ажыратуға мүмкіндік беретін ұсақ бөлшектерді ажыратуды үйрену үшін көптеген суреттер қажет болады.

Жоғарыда айтылғандай, 2016 жылдың наурыз айында жасанды интеллект AlphaGo DeepMind бағдарламасы терең оқыту арқылы 5 ойынның 4-де Го Ли Седольден әлем чемпионы жеңген кезде ірі жеңіске жетті. Google-да түсіндіргендей, терең оқыту жүйесі " терең Нейронды желілермен ағаштан іздеу үшін Монте-Карло әдісін біріктіру арқылы жұмыс істеді.

Терең оқытудың бизнес-қосымшалар да бар. Көптеген деректерді алуға болады-миллиондаған суреттер, және олардың көмегімен белгілі бір сипаттамаларды анықтау. Мәтіндік іздеу, алаяқтықты анықтау, спамды анықтау, қолжазба енгізуді тану, суреттерді іздеу, сөйлеуді тану, аудармабарлық осы тапсырмалар терең оқыту арқылы орындалуы мүмкін. Мысалы, Google-да терең оқыту желілері "ережелерге негізделген және қолмен жұмыс істеуді талап ететін жүйелер" көптеген процедураларды алмастырды.

Айта кету керек, терең оқыту өте "жалған" болуы мүмкін. Мысалы, бастапқыда Google бет тану жүйесі ашылған кезде, ол көптеген қара нәсілді тұлғаларды горилла деп санады. "Бұл сіздің оқу жиынында афроамерикандық тұлғалар болмаса, не болады деген мысал", деді any Tewary, Mint at Intuit деректерімен жұмыс жөніндегі бас маманы. "Егер сізде жүйемен жұмыс істейтін афроамерикандықтар болмаса, егер сізде жүйені тестілейтін афроамерикандықтар болмаса, онда сіздің жүйеңіз афроамерикандық тұлғаларға тап болғанда, ол өзін қалай ұстау керектігін білмейді. »

Бірақ, терең оқытуды асыра мақтап жіберді деген пікірлер де жетеді. Мысалы, Sundown AI жүйесі машиналық оқыту комбинациясын және терең оқытуды пайдаланбай-ақ policy graph алгоритмдерді пайдалана отырып, клиенттермен автоматтандырылған өзара әрекеттесуді ұсынады. [21]

1.2 Нейрондық жежілерді қолдану мақсаты

Дипломдық жобаның басты мақсаты жасанды нейрон желілерін пайдалана отырып, спорттық жарыстардың нәтижелерін болжау.

Қазіргі таңда нейрондық желілер (NN) бізді жасанды интеллект саласында жаңа деңгейге шығара отырып, бизнес пен күнделікті өмірді революциялайды. Мидың өзара байланысты жасушалары, NN қолдайтын машиналар қалай жұмыс істейтінін еліктей отырып, енді гуманоидты мәнерде паттерналарды зерттеу, тану және болжам жасау, сондай-ақ бизнестің әрбір секторында проблемаларды шешу үйренеді.

Нейрондық желілерді не себепті қолданамыз?

Нейрондық желілердің адами қасиеттері және олардың шексіз орын ауыстырулар мен комбинацияларда тапсырмаларды орындау қабілеті оларды үлкен деректерге негізделген заманауи қосымшалар үшін бірегей етеді. Нейрондық желілер де бірдей емес, қарама-қайшы немесе толық емес деректерді ұғынуға бірегей қабілетке ие болғандықтан, олар нақты модельдер жоқ кезде басқарылатын процестерді пайдалана алады.

Statista жариялаған есепке сәйкес, 2017 жылы жаһандық деректер көлемі айына 100 000 петабайтқа (яғни бір миллион гигабайт) жетті; болжам

бойынша, 2021 жылға қарай олар 232,655 петабайтқа жетеді. Компаниялар, жеке тұлғалар мен құрылғылар ақпараттың үлкен санын жасайды кезде, барлық осы BIG DATA бағасы, және нейрондық желілер осы деректермен пайдалы көп нәрсе жасай алады.

Нейрондық желілер орындайтын міндеттер

Олар мәліметтерді түсіну бойынша тапсырмаларды орындай алады, сонымен қатар өзінің барлық басқа атрибуттарын сақтай отырып, нейрондық желілер өте құнды. Төменде нейрондық желілерді орындайтын негізгі міндеттері сипатталған:

- *Жіктеу:* нейрондық желі алдын ала анықталған класстарға шаблондарды немесе деректер жинағын ұйымдастырады;
- *Болжамдау*: олар берілген кіріс деректерінен күтілетін нәтиже шығарады;
- *Кластерлеу:* олар деректердің бірегей ерекшелігін сәйкестендіреді және оны алдыңғы деректер туралы қандай да бір біліксіз жіктейді;
- *Ассоциация*: сіз нейрондық желілерді "есте сақтау" паттерналарын үйрете аласыз. Сіз үлгінің бейтаныс нұсқасын көрсеткенде, желі оны есте ең салыстырмалы нұсқасымен байланыстырады және соңғысына қайтарылады.

Нейрондық желілер терең оқыту, нейрондық желі әдістерінің сенімді жиынтығы үшін іргелі мәнге ие, ол биоинформатика, дәрі-дәрмектерді әзірлеу, әлеуметтік желілерді сүзу және табиғи тілге аудару сияқты дерексіз міндеттерді шешуге мүмкіндік береді. Терең оқыту-бұл ғылым мен техникадағы ең күрделі мәселелерді, соның ішінде озық робототехниканы шешетін боламыз. Нейрондық желілер ақылды және жылдам болғандықтан, біз күнделікті негізде табыстар жасаймыз.

Нейрондық желілерді нақты және салалық қолданыс мысалдары

2018 жылғы тамыздағы New York Times мақаласында көрсетілгендей, " автоматтандыру үшін бағдарламалық жасақтаманы тарта бастаған компаниялар мен үкіметтік мекемелер гамманы басқарады. Олар General Motors, BMW, General Electric, Unilever, MasterCard, "рабсилу", FedEx, Cisco, Google, қорғаныс және НАСА министрлігін қамтиды."Біз әлемнің қалай жұмыс істейтінін өзгертетін нейрондық желілер мен жасанды интеллектті қолданудың басталуын байқаймыз.

1.3 Іске асыру ортасының компоненттерін таңдау

Дипломдық жобаны жазу кезінде келесі құралдар қолданылды:

- Python-әзірлеушінің өнімділігін және кодтың оқылуын арттыруға бағытталған жалпы мақсаттағы жоғары деңгейлі бағдарламалау тілі. Python ядросының синтаксисі минималды, яғни үлкен код қамтымайды. Сонымен қатар стандартты кітапхана пайдалы функциялардың үлкен көлемін қамтиды. Көбінесе оны нейрондық желілермен жұмыс істеу кезінде қолданады, өйткені олармен жұмыс істеу үшін көптеген кітапханалар бар;
- PyCharm – Python бағдарламалау тілі үшін интеграцияланған даму ортасы. Кодты талдау құралы, графикалық реттеуші, юнит-тестерді іске қосу құралы және Django веб-әзірлемесін қолдайды. PyCharm IntelliJ IDEA негізінде JetBrains компаниясымен жасалған;
- TensorFlow – адам қабылдауының деңгейіне жету арқылы бейнелерді автоматты табу және жіктеу мақсатында нейрондық желіні құру және жаттықтыру мәселелерін шешу үшін Google компаниясы әзірлеген машинамен оқытуға арналған ашық бағдарламалық кітапхана. Зерттеу үшін де, Google өз өнімдерін әзірлеу үшін де қолданылады. Кітапханамен жұмыс істеу үшін негізгі API Python үшін іске асырылды, сондай-ақ C Sharp, C++, Haskell, Java, Go және Swift үшін орнықталған варианттары бар.

1.4 Іске асыруға арналған технологиялар мен құралдар

1.4.1 Tensorflow кітапханасы

TensorFlow-Машиналық оқыту үшін ашық бастапқы коды бар өтпелі платформа. Ол зерттеушілерге ML қазіргі заманғы күйін ілгерілетуге мүмкіндік беретін, ал әзірлеушілерге ML базасында қосымшаларды оңай құруға және өрістетуге мүмкіндік беретін қоғамдастық құралдарының, кітапханалары мен ресурстарының тұтас, икемді экожүйесіне ие.

TensorFlow бастапқыда Google machine Intelligence Research зерттеу ұйымы шеңберінде Google Brain тобында жұмыс істейтін зерттеушілер мен инженерлер жасаған. Бұл жүйе әртүрлі басқа салаларда қолдануға жеткілікті жалпы болып табылады.

Google Brain командасы жасаған TensorFlow-сандық есептеулер мен ірі көлемді машиналық оқыту үшін ашық бастапқы коды бар кітапхана. TensorFlow машиналық оқыту және терең оқыту (нейрондық желі) көптеген модельдері мен алгоритмдерін біріктіреді және оларды ортақ метафора ретінде пайдалы етеді. Ол фреймворк көмегімен қосымшаларды құру үшін ыңғайлы интерфейсті API қамтамасыз ету үшін Python пайдаланады, бұл ретте осы қосымшаларды жоғары өнімді C++ - де орындайды.

TensorFlow бірге сіз қолжазба сандар, жіктеу, бейнелерді тану, сөз тіркемелер, рекурренттік нейрондық желілер, машиналық аударма модельдерінің тізбектілігі, табиғи тілді өңдеу, және негізінде модельдеу

теңдеулері (Дифференциалдық теңдеу) үшін терең нейрондық желілермен жаттығып, жұмыс істей аласыз. Ең жақсы, TensorFlow оқыту үшін пайдаланылатын үлгілермен масштабта өндірісті болжауды қолдайды.

Tensoreflow қалай жұмыс істейді

TensorFlow әзірлеушілерге деректер ағындарының графиктерін жасауға мүмкіндік береді. Ол графиктерде деректер баған немесе өңдеу тораптарының қатары бойынша қалай жылжитынын сипаттайтын құрылымдар көрсетіледі. Әрбір баған торабы математикалық операция болып табылады, ал тораптар арасындағы әрбір қосылым немесе қабырға көп өлшемді деректер массиві немесе тензор болып табылады.

TensorFlow Python тілінің көмегімен барлық осы мүмкіндіктерді бағдарламашыға ұсынады. Python игеруге және жұмыс істеуге оңай програмамалау тілі, сондай-ақ жоғары деңгейлі абстракциялар бірге қосылуы мүмкін ыңғайлы өрнектерді ұсынады. TensorFlow Python нысандары және TensorFlow бағдарламалар өздері Python бағдарламалары болып табылады.

Алайда, нақты математикалық операциялар Python-да орындалмайды. TensorFlow арқылы қол жетімді түрлендіру кітапханалары C++ жоғары өнімді екілік файлдар ретінде жазылады. Python жай ғана бөліктер арасында трафик бағыттайды және оларды бірге байланыстыру үшін жоғары деңгейлі бағдарламалық абстракцияларды ұсынады.

TensorFlow қосымшаларын кез келген ыңғайлы нысанда іске қосуға болады: дербес компьютер, бұлттық (облачный) кластері, iOS және Android құрылғылары, процессорлар немесе графикалық процессорлар. Егер сіз Google бұлтын пайдалансаңыз, TensorFlow Processing Unit (TPU) silicon пайдаланушы процессорында одан әрі жеделдету үшін іске қосуға болады. Алайда, TensorFlow жасаған нәтижелік модельдер болжамдарға қызмет көрсету үшін пайдаланылатын кез келген құрылғыда іс жүзінде өрістетілуі мүмкін.

2019 жылы қазан айында шығарылған TensorFlow 2.0 фреймворк қолданушылардың пікірлері негізінде жұмыс істеу үшін қарапайым (мысалы, модельдерді оқыту үшін қатысты қарапайым Keras API көмегімен) және одан да көп өнімді жасау үшін жаңартады. Таратылған оқытуды жаңа API арқасында оңай орындауға болады, ал TensorFlow Lite қолдауы модельдерді әртүрлі платформаларда өрістетуге мүмкіндік береді. Алайда, TensorFlow ерте нұсқалары үшін жазылған код қайта жазылуға тиіс, себебі TensorFlow 2.0 жаңа мүмкіндіктерін барынша пайдалану мүмкіндігі жоғалады.

TensorFlow-дын орындалу моделі.

Диаграммалар.

Машиналық оқыту тез қиындап кетуі мүмкін және терең оқыту модельдері тым үлкен болып кетуі мүмкін. Көптеген үлгілер бағандары үшін сіз ақылға қонымды уақыт кезеңі ішінде итерацияны орындау мүмкіндігі болуы үшін бөлінген оқыту қажет. Және, әдетте, сіз әзірлейтін модельдер бірнеше платформаларда ашылғанын қалайсыз.

TensorFlow-дың қазіргі нұсқасында сіз есептеу бағанын құру үшін кодты жазасыз, содан кейін оны орындайсыз. Кесте сіз орындауғыңыз келетін есептеулерді толық сипаттайтын деректер құрылымы болып табылады. Бұның көптеген артықшылықтары бар:

- Кесте дереу орындалуы немесе кейінірек пайдалану үшін сақталуы мүмкін, өйткені ол бірнеше платформаларда жұмыс істей алады: процессорлар, графикалық процессорлар, TPU, мобильді, кіріктірілген видеокарталарда. Бұдан басқа, ол бағандарды салған қандай да бір кодқа, оны орындау үшін қажетті орындау уақытына байланысты қажетсіз өндіріске өрістетілуі мүмкін;
- Оны түрлендіруге және оңтайландыруға болады, себебі кесте осы платформа үшін оңтайлы нұсқасын алу үшін түрлендірілуі мүмкін. Сонымен қатар, жадты немесе есептеулерді оңтайландыру және олардың арасындағы ымыраласу орындалуы мүмкін. Бұл, мысалы, үлкен машиналарда оқыған кейін жылдам мобильді шығаруды қолдау үшін пайдалы;
- Бөлінген орындауды қолдау.

Оқытуды тасымалдау

Көптеген TensorFlow модельдері сіздің жеке сыныптауыңызды зерттеу үшін, мысалы, тасымалдауға оқыту үшін оларды қалай пайдалануға болатынын көрсететін оқыту салмақтары мен мысалдарды қамтиды. Әдетте, сіз пайдалы абстракцияларды кодтайды—содан кейін оны өз сыныптарыңызды болжау үшін сіздің әлдеқайда аз нейрондық желісін оқыту үшін кіріс деректер ретінде пайдаланасыз. Меңгерілген абстракциялардың күшінен қосымша оқыту әдетте үлкен деректер жиынтығын талап етпейді.

Көптеген TensorFlow модельдері сіздің жеке сыныптауыңызды зерттеу үшін, мысалы, тасымалдауға оқыту үшін оларды қалай пайдалануға болатынын көрсететін оқыту салмақтары мен мысалдарды қамтиды. Әдетте, сіз пайдалы абстракцияларды кодтайсыз, содан кейін оны өз сыныптарыңызды болжау үшін сіздің әлдеқайда аз нейрондық желісін оқыту үшін кіріс деректер ретінде пайдаланасыз. Меңгерілген абстракциялардың арқасында қосымша оқыту әдетте үлкен деректер жиынтығын талап етпейді.

Мысалы, сіз бастапқы кескіндерді жіктеу үлгісімен трансферлік оқытуды пайдалана аласыз, ол сіздің арнайы бейнелер деректерін пайдаланады.

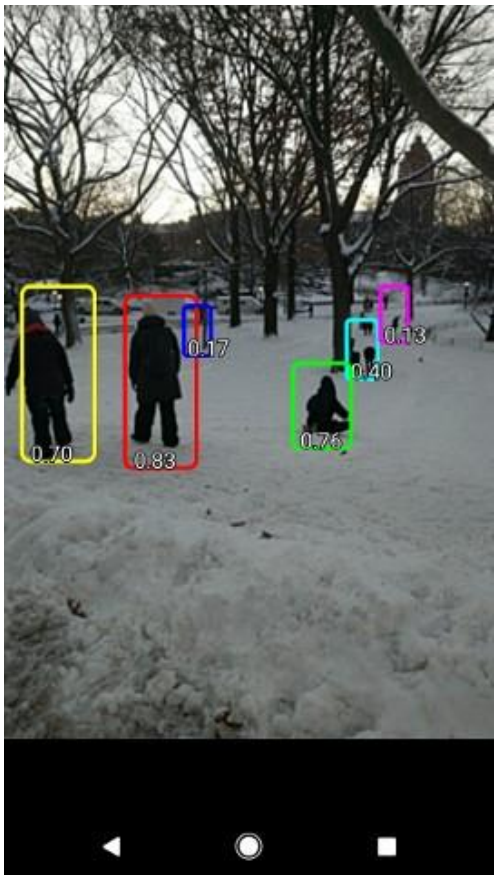
Мобильді құрылғыларда TensorFlow-ды пайдалану

Ұялы телефон – TensorFlow пайдалану үшін тамаша мысал. Ұялы телефон желі байланысы нашар, тіпті жоқ болған кезде немесе серверге үздіксіз деректерді жіберу тым қымбат болған кезде қолданудың мағынасы бар. Бірақ сіз өз моделін үйретіп, оны пайдалануға дайын болған соң, сіз құрылғыда алынған моделдің тым үлкен болғанын қаламайсыз.

TensorFlow, MobileNet жобаларының бірінде мобильді құрылғыларда немесе кіріктірілген қосымшаларда ескерілуі қажет жылдамдық пен дәлдік арасындағы ымыраларды жою үшін арнайы әзірленген компьютерлік көру

үлгілерінің жиынтығы әзірленеді. MobileNet модельдерін TensorFlow модельдерінің репозиториясында табуға болады.

Android, TF Detector жаңа әзірлемелерінің бірі TensorFlow объектілерін анықтау API көмегімен оқытылған мобильді желілік модельді пайдаланады(1-суретті қараңыз). [24]



1.2 сурет - TensorFlow объектілерін анықтау API көмегімен оқытылған мобильді қосымша.

TensorFlow-дың артықшылықтары

TensorFlow машиналық оқытуды дамыту үшін ұсынатын жалғыз ең үлкен артықшылығы - абстракция. Алгоритмдерді жүзеге асырудың ұсақ бөлшектерімен айналысудың немесе бір функцияны екінші функцияның кіруіне байланыстырудың дұрыс тәсілдерін анықтаудың орнына әзірлеуші қосымшаның жалпы логикасына шоғырлануы мүмкін.

TensorFlow қосымшаларында өзін-өзі тексеру және алу қажет әзірлеушілерге қосымша қолайлы. Орындау режимі барлық кестені бір мөлдір емес объект ретінде салудың және оның барлығын бірден бағалаудың орнына әрбір операцияны жеке және мөлдір бағалауға және өзгертуге мүмкіндік береді. TensorBoard визуализация жинағы интерактивті мониторинг веб-панелінің көмегімен графиктердің жұмысын көруге және профильдеуге мүмкіндік береді. TensorFlow TensorFlow қосымшаларында өзін-өзі тексеру және алу қажет әзірлеушілерге қосымша қолайлы. Орындау режимі барлық

кестені бір мөлдір емес объект ретінде салудың және оның барлығын бірден бағалаудың орнына әрбір операцияны жеке және мөлдір бағалауға және өзгертуге мүмкіндік береді. TensorBoard визуализация жинағы интерактивті мониторинг веб-панелінің көмегімен графиктердің жұмысын көруге және профильдеуге мүмкіндік береді.

TensorFlow сондай-ақ Google a-list коммерциялық жабдығын қолдаудан көптеген артықшылықтарды алады. Google жобаның жылдам даму қарқынын күшейтіп қана қоймай, сонымен қатар TensorFlow айналасында көптеген маңызды ұсыныстар жасайды, бұл оның өрістеуі мен қолданылуын жеңілдетеді: жоғарыда аталған Google бұлт өнімділігін жеделдету үшін TPU silicon; фреймворк арқылы жасалған модельдерді алмасу үшін онлайн-хаб; фреймворк арқылы орнатылған браузерлік және мобильді фреймворк; және тағы басқа ұсыныстар.

Бір ескерту: Tensorflow жүзеге асырудың кейбір бөлшектері кейбір оқу тапсырмалары үшін толық детерминирленген оқыту нәтижелерін алуды қиындатады. Кейде бір жүйеде оқытылған модель, олар бір деректерді алса да, басқа жүйеде оқытылған үлгіден сәл өзгеше болады. Бұл тайғақ себептері-мысалы, кездейсоқ сандар немесе графикалық процессорларды пайдалану кезінде мінез-құлықтың кейбір детерминацияланған модельдері қалай және қайда себіледі). Дегенмен, осы мәселелерді айналып өтуге болады, және TensorFlow командасы жұмыс процесінде детерминизм әсер ету үшін қосымша басқару элементтерін қарастырады.

TensorFlow бәсекелестермен салыстырмалы түрде

- TensorFlow көптеген басқа машина оқыту жүйелерімен бәсекелеседі. PyTorch, CNTK және MXNet көптеген бірдей қажеттіліктерге жауап беретін үш негізгі платформа болып табылады. Төменде олардың айырмашылықтары мен ұқсастықтары келтірілген;

- PyTorch Python-да құрастырудан басқа, TensorFlow-мен көптеген басқа ұқсастықтары бар: аппараттық үдеткіш компоненттер, әзірлеу шамасына қарай жұмыс істеуге мүмкіндік беретін әзірлеудің жоғары зияткерлік моделі және көптеген пайдалы компоненттер қазірдің өзінде қосылған. PyTorch, әдетте, қысқа мерзімде іске қосылуы тиіс жобаларды жылдам әзірлеу үшін ең жақсы таңдау болып табылады, бірақ TensorFlow ірі жобалар мен одан да күрделі жұмыс процестері үшін жеңеді;

- CNTK, Microsoft Cognitive Toolkit Tensorflow сияқты деректер ағынын сипаттау үшін баған құрылымын пайдаланады, бірақ терең оқытудың нейрондық желілерін құруға көп көңіл бөледі. CNTK нейрон желісінің көптеген тапсырмаларын жылдам орындайды және API (Python, C ++, C #, Java) кең жиынтығы бар. Бірақ CNTK қазіргі уақытта TensorFlow сияқты зерттеу немесе ашу оңай емес;

- Amazon AWS терең оқытудың негізгі ортасы ретінде қабылдаған Apache MXNet бірнеше графикалық процессорлар мен бірнеше компьютерлерде іс жүзінде желілік масштабталуы мүмкін. Ол сондай-ақ Python, C ++, Scala, R, JavaScript, Julia, Perl, Go тілдік API - интерфейстердің

кең спектрін қолдайды, бірақ оның нативті API-интерфейстерімен жұмыс істеу TensorFlow сияқты жақсы емес;

- бірегей бөлшектерді тереңірек зерттеуге лайық;
- негізгі кітапхана тек терең оқыту үшін емес, Машиналық оқыту техникаларының кең тобына арналған;
- сызықтық алгебра және басқа да ішкі заттар сыртынан жақсы көрінеді;
- машина оқытудың негізгі функционалдылығына қосымша, TensorFlow, сондай-ақ өз логика жүйесін, жеке интерактивті визуализатор логов және тіпті қуатты деректерді жеткізу сәулетін қамтиды;
- TensorFlow орындау моделі scikit-learn, Python тілі және көптеген құралдар ерекшеленеді.

1.4.2 PYTHON программалау тілі

PYTHON деген не?

Python-жалпы мақсаттағы бағдарламалау үшін қолданылатын динамикалық семантикасы бар кең қолданылатын, түсіндірілетін, объектілі-бағытталған және жоғары деңгейлі бағдарламалау тілі. Ол Гвидо ван Россуммен құрылды және 1991 жылдың 20 ақпанында алғаш рет шығарылды.

Сіз питонды үлкен жылан ретінде білуіңіз мүмкін, ал, Python бағдарламалау тілінің атауы "Монти Пайтон ұшатын циркі" деп аталатын BBC комедия сериалының ескі телехикаяның атауынан шыққан.

Python таңғажайып ерекшеліктерінің бірі - шын мәнінде бұл бір адамның жұмысы болып табылады. Әдетте жаңа бағдарламалау тілдері көптеген кәсіпқойлар жұмыс істейтін ірі компаниялармен әзірленеді және жарияланады, және авторлық құқық ережелеріне байланысты жобаға қатысушылардың бірін атау өте қиын.

Әрине, ван Россум Python барлық компоненттерін жалғыз өзі әзірлеп, дамытқан жоқ. Python бүкіл әлем бойынша жылдам тарады. Мыңдаған үздіксіз жұмыс нәтижесі болып табылады, өте жиі анонимді бағдарламашылар, тестілеушілер, пайдаланушылар мен энтузиасттар, бірақ бірінші идея Гвидоның басына келді деп айтуға болады.

Python бағдарламалық жасақтаманы әзірлеушілер үшін швейцариялық пышақпен жасайтын кейбір ерекшеліктері бар. Ол объектілі-бағытталған және түсіндіріледі. Нысандар нақты әлемнің физикалық объектілеріне ұқсайды. Олардың өзіндік қасиеттері (деректер ішінде сақталады) және мінез-құлықтары (олар қабылдай алатын әрекеттер) бар.

Нысандар бір-бірімен өзара әрекеттесуі мүмкін. Python сондай-ақ, бағдарлама компиляциясыз тікелей орындалуы мүмкін дегенді білдіреді. Python аудармашы бағдарламаны басқа тілдерге аудара алады.

Ашық бастапқы коды бар Технология және кез келген корпорацияға қарамастан әзірленді. Бұл Python қауымдастығының рухын ерекше етеді ерекшелігі.

Python Кремний алқабын нәрлендіреді

Әлемдегі ең табысты өнімдердің көпшілігі Python пайдаланып жасалды.

Facebook Instagram, Netflix, Spotify, Reddit, Facebook және Google осы тілді қолданатын ең маңызды сандық жобалардың бірі болып табылады.

Google негізін қалаушылар өз іздеу жүйесін құру кезінде қарапайым даму ережесін ойлап тапты: "біз мүмкін жерде Python-ды пайдаланамыз, оның мүмкіндігі жетпейтін жерде C++ тілін қолданамыз". Бұл олар үшін жақсы жұмыс істеді.

Python сондай-ақ Силикон алқабының келесі ірі табысқа тарихы қатысты, өйткені ол Facebook жасау үшін пайдаланылатын үшінші тіл болды (C ++ және PHP кейін бірден).

2016-да Instagram инженерлері командасы Python-да толық жазылған Django веб-фреймворкын іске қосты.

Reddit бастапқыда Лиспада жазылған болса да, 2005 жылы олар барлық платформаны Python-ға көшіруді шешті. Шешім код кітапханаларының кең спектрімен және әзірлеудің икемділігімен шартталған.

Сервер жағындағы деректерді талдау үшін Python пайдалану да күшті дәлел болды. Spotify инженерлері негізінен деректерді және серверлік қызметтерді талдау үшін Python пайдаланады, ал Netflix сервер жағындағы деректерді талдау үшін Python пайдаланады.

Python ең маңызды артықшылықтары

Python оны пайдалану кезінде алатын өте күшті артықшылықтардың тізімі арқылы әзірлеушілер мен өнім иелері арасында өте танымал.

1) Қолданбаларды жылдам жасау

Python-да көптеген кітапханалар бар, сондай-ақ сапалы кодты қайта өңдеу және қайта пайдаланумен айналысады.

Сонымен қатар, ол динамикалық үлгілеу және динамикалық байланыстыру - оны қосымшаларды жылдам әзірлеу үшін тамаша технологиямен (RAD) жасайтын функциялар, бағдарламалық жасақтаманы икемді әзірлеу үшін пайдаланылатын тәсіл, онда мақсаты мүмкіндігінше тезірек өміршеңдігі төмен (MVP) функционалдық өнімді іске қосу болып табылады, содан кейін пайдаланушылар мен нарықтың нақты пікірлері негізінде қажетті жақсартулар мен өзгерістерді енгізу.

2) Қарапайым интеграция

Python сондай-ақ "желім тілі" деп аталады, өйткені оны басқа компоненттермен біріктіру оңай. Бұл сипаттама, сондай - ақ бағдарламалық қамтамасыз етуді әзірлеудің жаңа үрдістеріне-модульдік бағдарламалауға сәйкес келеді, онда сіз барлық үздік бағдарламалау тілдерінен, қаңқалар мен сыртқы сервистерден өз өніміңізді "түзетесіз".

3) Көптеген модульдер мен пакеттер пайдалануға дайын

Python модульдері мен пакеттері кез келген конфигурацияда пайдаланылуы мүмкін - оларды қосуға, жоюға, ауыстыруға немесе өзгертуге болады. Эксперимент үшін көп орын бар. Уақыт өте келе стандартты кітапхана қауымдастық жасаған пакеттерге айналды.

Python-да кез келген тапсырма үшін пакет бар деп саналады. Бұл ақпаратты өтірік деп айту қиын, себебі пакеттер қоймасында 147 000 астам пакет бар.

4) Тегіс бағдарламалау циклі

Python интерпретаторы, яғни командаларды орындайтын бағдарлама осының бәрін әлдеқайда ыңғайлы етеді. Интерпретатор стандартты бағдарламалау циклін (өңдеу, тестілеу және жөндеу) өте тегіс етеді ешқандай компиляция қажет емес.

Жылдам Итерация қосымшаларды жылдам жасау үшін (RAD - rapid application development) шешуші мәнге ие. Сонымен қатар, қате немесе зақымдалған енгізу сегменттеу қатесін тудырмайды. Олар табылған жағдайда интерпретатор ерекшелік жасайды.

5) Тамаша өнімділік

Жалпы алғанда, Python жақсы өнімділікті қамтамасыз етеді, бұл тіл тез танымал неге негізгі себеп болуы мүмкін. Инженерлер заттарды тез жасауды жақсы көреді.

Python оңай жазуға, оқуға және оқуға мүмкіндік беретін қарапайым синтаксис бар. Кейбір Python құжаттаманы қажет емес деп айтады, Егер сіз жақсы жазылған кодты оқып және кез келген қосымша түсіндірмелерсіз оны түсінуге болады.

Python Сіздің бизнесіңіз үшін не істей алады?

Үздік инженер-бағдарламашылар жақсы көретін нәрсе тамаша бизнес әлеуеті болуы тиіс. Python стартаптар, сондай-ақ корпоративтік бағдарламалық қамтамасыз етуді әзірлеу, деректер туралы ғылым жобалары, Машиналық оқыту (ml тағы бірнеше мысалдары) кеңінен қолданылады. Ол, бәлкім, оның үлкен деректерді өңдеу, сондай-ақ блоктық байланысты кітапханалардың кең спектрі үшін қаржы компаниялары арасында әсіресе танымал болды.

Төменде Python бәсекелестерін жеңетін кейбір салалар келтірілген:

1) Бағдарламалық қамтамасыз етуді әзірлеу

Жоғарыда айтылғандай, ол өте әмбебап. Сіз Python-да толық қолданбаларды немесе басқа технологиялармен оңай байлануға болатын кейбір модульдерді жасай аласыз.

Құрастыруды бақылау, автоматтандырылған және үздіксіз компиляция және тестілеу, жобаларды басқару және қателерді бақылау - осының барлығы бағдарламалық жасақтаманы әзірлеушілер үшін өте пайдалы.

Django оны жылдам және тұрақты веб-әзірлеу үшін қуатты құралы етеді.

2) Жасанды Интеллект және машиналық оқыту

Python бағдарламалау тілі қазіргі уақытта ғылыми бағдарламалауды нәрлендіруде. Барлығы NumPy және SciPy сияқты Python есептеу машиналарын шығарудан басталды. Баяу, бірақ дұрыс Python компьютерлік ғылымдар саласындағы зерттеулер үшін қолайлы тіл бола бастады.

3) Blockchain

Python жылдам қосылу үшін қол жетімді кітапханалардың үлкен жинағы бар.

4) Корпоративтік бағдарламалық қамтамасыз етуді әзірлеу

Кәсіпорындар үшін бағдарламалық қамтамасыз ету қолданыстағы деректер базасы және веб-қосымшалар сияқты ескірген жүйелермен интеграциялауды жиі қажет етеді. Python басқа тілдерді өте оңай біріктіре алады және әрқашан жоғары қауіпсіздік стандарттарын қолдайды.

Python қарапайым; бұл сіздің өнімнің нарыққа шығу уақытын қысқартады.

Ол қашықтағы бірлескен жұмысты жеңілдететін қарапайым синтаксис бар. Қаржы секторы үшін құнды шешімдерді қамтитын ашық бастапқы коды бар кітапханалар бизнес үшін одан да көп мүмкіндіктер жасайды.

Ақырында, Python табиғатта қауіпсіз технология болып табылады, бұл құпия жеке деректермен жұмыс істейтін қаржылық қосымшалар немесе қосымшалар үшін тамаша таңдау жасайды.

Python бағдарламалау тілінің ерекшеліктері:

1) Python кодын оқу өте ыңғайлы

2) Үйрену оңай: python үйрену оңай, өйткені бұл мәнерлі және жоғары деңгейлі бағдарламалау тілі, бұл оны түсіну оңай және, демек, үйрену оңай.

3) Кросс-платформалық: Python-бұл қол жетімді және Mac, Windows және Linux, Unix және басқалар сияқты түрлі операциялық жүйелерде жұмыс істей алады. Бұл оның кросс-платформалық және портативті тіл етеді.

4) Ашық бастапқы коды: Python-ашық бастапқы коды бар бағдарламалау тілі.

5) Үлкен стандартты кітапхана: Python үлкен стандартты кітапханамен бірге келеді, ол Python кодын жазу кезінде пайдалана алатын кейбір ыңғайлы кодтар мен функцияларға ие.

6) Тегін: Python тегін жүктеу және пайдалануға болады. Бұл дегеніміз, сіз оны тегін жүктеп, қолданбада пайдалануға болады. Қараңыз: Ашық Бастапқы Коды Бар Python Лицензиясы. Python FLOSS үлгісі болып табылады (Free/Libre Open Source Software), бұл сіз осы бағдарламалық қамтамасыз ету көшірмелерін еркін таратуға болады дегенді білдіреді, оның бастапқы кодын оқып және оны өзгертуге.

7) Ерекшеліктер өңдеу қолдау: Егер сіз жаңа болса, сіз ерекшелік не сұрай аласыз? Ерекшелік-бұл бағдарламаны жою кезінде орын алуы мүмкін және бағдарламаның қалыпты ағынын бұзуы мүмкін оқиға. Python ерекшеліктерді өңдеуді қолдайды, бұл дегеніміз, біз аз қате кодты жаза аламыз және кейінірек ерекшелік тудыруы мүмкін әр түрлі сценарийлерді сынауға болады.

8) Кеңейтілген мүмкіндіктер: генераторларды қолдау және тізімді түсіну. Біз кейінірек осы ерекшеліктерді қарастырамыз.

9) Жадыны автоматты басқару: Python жадты автоматты түрде басқаруды қолдайды, бұл жады автоматты түрде тазартылатынын және босатылатыны білдіреді.

Сіз Python колдана отырып не істей аласыз?

Мүмкін, сіз барлық Python қосымшалары туралы білгіңіз келеді, бірақ олар тым көп. Төменде кейбіреулері мысал ретінде келтірілген:

- веб-әзірлеу – django және flask сияқты веб-фреймворктары python негізінде жасалған. олар сізге деректер базасын басқаруға, бэкенд бағдарламалау логикасын жазуға, url мекенжайын салыстыруға көмектесетін сервер жағында кодты жазуға көмектеседі;
- машина оқыту – python тілінде жазылған көптеген машина оқыту бағдарламалары бар. машиналық оқыту - бұл машина өз бетімен оқып, белгілі бір мәселені шеше алатын логика жазу тәсілі. мысалы, amazon, flipkart, ebay және т. б. сияқты сайттардағы өнімдер бойынша ұсыныстар. телефоныңызда бет пен дауысты тану – бұл машиналық оқытудың тағы бір мысалы;
- деректерді талдау-деректерді талдау және деректерді диаграмма түрінде визуализациялау сондай-ақ python көмегімен жасалуы мүмкін;
- скриптинг – бұл қарапайым тапсырмаларды автоматтандыру үшін шағын бағдарламаларды жазу;
- ойындарды жасау – сіз python пайдаланып ойындар жасай аласыз;
- сіз python ішіне енгізілген қолданбаларды жасай аласыз;
- үстелдік қосымшалар-сіз tkinter немесе qt сияқты кітапханаларды пайдалана отырып, python үстелдік қосымшаларды әзірлей аласыз.

1.4.3 PyCharm

PyCharm деген не?

Қазіргі уақытта көптеген бағдарламашылар кодтың ықшам, таза және ыңғайлы базасы бар бағдарламалық қосымшаларды құру үшін Python таңдайды. Олар тіпті Python үшін интеграцияланған даму ортасының (IDE) артықшылықтарын пайдалана отырып, пайдаланушылық бағдарламалық қосымшаларды әзірлеуді тездетуі мүмкін. PyCharm-Python бағдарламалау тілі үшін ең кең қолданылатын IDE бірі. Қазіргі уақытта Python IDE Twitter, Pinterest, HP, Symantec және Groupon сияқты ірі кәсіпорындар қолданады.

JetBrains PyCharm-ны Python үшін IDE кроссплатформалы етіп әзірледі. 2 нұсқаларын қолдаудан басқа.x және 3.x Python, PyCharm Windows, Linux және macOS үйлесімді. Сонымен қатар, PyCharm ұсынатын құралдар мен функциялар программистерге Python әр түрлі бағдарламалық қосымшаларды тез және тиімді жазуға көмектеседі. Әзірлеушілер тіпті олардың нақты қажеттіліктері мен артықшылықтарына сәйкес PyCharm пайдаланушы интерфейсін теңшей алады. Сонымен қатар, олар жобаның күрделі талаптарын қанағаттандыру үшін 50-ден астам плагиндерді таңдап, IDE кеңейте алады.

PyCharm ұсынатын маңызды функциялар мен құралдарды шолу.

Код редакторы.

PyCharm ұсынатын зияткерлік код редакторы бағдарламашыларға жоғары сапалы Python кодын жазуға мүмкіндік береді. Редактор

программистерге түсті схемалар арқылы кодты оңай оқуға, автоматты түрде жаңа жолдарға шегіністерді енгізуге, сәйкес келетін кодтау стилін таңдауға және контексті ескере отырып, кодты аяқтау үшін кеңестерді пайдалануға мүмкіндік береді. Сонымен қатар, программистер, сондай-ақ редактор код блогын өрнекке немесе логикалық блоктарға дейін кеңейту үшін пайдалана алады, код фрагменттерін пайдалана алады, код базасын пішімдей, жазудағы қателер мен қателерді анықтай алады, қайталанатын кодты тауып, кодты автоматты түрде жасай алады. Сонымен қатар, редактор әзірлеушілерге кодты талдауды және кодты жазу кезінде қателерді анықтауды жеңілдетеді.

Код бойынша навигация.

PyCharm ұсынатын зияткерлік кодты шарлау опциялары бағдарламашылар қосымша уақыт пен күш жұмсамай, кодты өңдеуге және жақсартуға көмектеседі. IDE ортасы goto хабарламаларымен бірге класына, файлына және символдарға ауысуды жеңілдетеді. Пайдаланушы бастапқы кодта, код фрагменті, пайдаланушы интерфейсінің элементін немесе пайдаланушы әрекетін бірден таба алады. Олар әр түрлі таңбаларды пайдалану орнын қосымша анықтай алады және кодта бетбелгілерді орната алады. Сонымен қатар, әзірлеушілер тіпті кодты объектив режимінде мұқият зерттеу үшін код бойынша навигация функциясын пайдалана алады.

Рефакторин.г

PyCharm әзірлеушілерге жергілікті және жаһандық өзгерістерді тез және тиімді енгізуді жеңілдетеді. Python қарапайым кодын жазу және Python платформаларымен жұмыс істеу кезінде әзірлеушілер тіпті ide ұсынған рефакторинг мүмкіндіктерін пайдалана алады. Олар файлдарға, сыныптарға, функцияларға, әдістерге, қасиеттерге, параметрлерге және жергілікті / жаһандық айнымалыларға арналған ауыстыруларды қайта атау мен рефакторингті пайдалана алады. Сонымен қатар, олар код сапасын жақсарта алады, айнымалыларды, өрістерді, константаларды және параметрлерді алады. Сонымен қатар, PyCharm программистерге extract әдісі арқылы ұзын сыныптар мен әдістерді бөлуге мүмкіндік береді.

Танымал веб-технологияларды қолдау.

PyCharm бағдарламашыларға HTML, CSS, JavaScript, TypeScript және CoffeeScript сияқты кең қолданылатын веб-технологияларды қолдайтын Python түрлі веб-қосымшаларды жазуды жеңілдетеді. Веб-әзірлеушілер өңдеу және браузерде бір веб-бетті бір уақытта көру үшін IDE ұсынатын нақты уақыт режимінде алдын ала қарау функциясын пайдалана алады. Сонымен қатар, ide ұсынатын тікелей өңдеу функциясы бағдарламашыларға кодқа енгізілген өзгерістерді веб-браузерде бірден көруге мүмкіндік береді. Сонымен қатар, PyCharm әзірлеушілерге JavaScript реттеуші, сондай-ақ CoffeeScript және TypeScript редакторларын пайдалануға мүмкіндік береді. Ол тіпті AngularJS және NodeJS қолдау арқылы изоморфты веб-қосымшаларды әзірлеуді жеңілдетеді.

Танымал Python веб фреймворктерін қолдау.

Жиі қолданылатын веб-технологияларды қолдаудан басқа, PyCharm да Django сияқты сенімді Python веб-инфрақұрылымына жоғары сапалы қолдау көрсетеді. Әзірлеушілер тегтерге, сүзгілерге, параметрлерге және Django айнымалы үлгілеріне арналған кодты толықтыру бойынша ұсыныстар алу үшін интеграцияланған даму ортасын пайдалана алады. Сонымен қатар, олар тез құжаттамаға қарап, тегтер мен сүзгілер туралы қосымша ақпарат жинай алады. IDLE Python тіпті веб-әзірлеушілерге Django үлгілерін реттеуге, кодты пішімдеуге, кодты тексеруге және консольдерді басқаруға көмектеседі .py. Сонымен қатар, PyCharm Pyramid және Web2Py сияқты кең қолданылатын Python веб-фреймворкаларын да қолдайды. Ол кодты аяқтауды және Pyramid тән навигация параметрлерін қамтамасыз етеді. Сонымен қатар, ол веб-әзірлеушілерге web2py-мен жұмыс істеу кезінде кодты автоау және шарлау мүмкіндігін пайдалануға мүмкіндік береді [11].

Python ғылыми кітапханаларын қолдау.

PyCharm сондай-ақ программистерге үлкен жобалар мен ғылымды қажетсінетін жобаларға Python-ды неғұрлым тиімді пайдалануға көмектеседі. Ол Python - NumPy, Anaconda және Matplotlib үшін кеңінен пайдаланылатын ғылыми кітапханалардың кейбір қолдайды. Әзірлеушілер интерактивті графиктерді, кодты терең түсінуді және Ide ұсынатын массивтерді көру құралдарын пайдалана отырып, осы ғылыми кітапханалармен тиімді жұмыс істей алады. Олар тіпті PyCharm ұсынатын REPL Python консолін іске қосуы мүмкін, мысалы, flam синтаксисті тексеру және кодты тексеру сияқты сенімді функцияларды пайдалану. Сонымен қатар, бағдарламашылар қосымша уақыт пен күш жұмсамай инновациялық шешімдерді жасау үшін IDE-ді IPython Notebook-мен оңай біріктіре алады.

Деректер қорының құралдары.

PyCharm Python-ның әртүрлі кітапханалар мен құрылымдарды қолдаудан басқа, әзірлеушілерге Oracle, SQL Server, MySQL және PostgreSQL сияқты реляциялық деректер базаларымен жұмыс істеуге мүмкіндік береді. Әзірлеушілер сұраныстарды орындау, SQL кодын өңдеу, деректерді қарау, кесте деректерін өзгерту және схемаларды өзгерту / талдау үшін қосымша IDE қолдана алады. PyCharm сондай-ақ sqlalchemy кітапханасын қолдайды және SQL кодын әртүрлі бағдарламалау тілдерінде жазылған кодқа енгізеді. IDE кәсіби нұсқасы әзірлеушілерге деректер торының көмегімен деректердің үлкен көлемін тиімді өңдеуді жеңілдетеді.

Визуальный отладчик.

IDE ұсынатын визуалды реттеуші программистерге Python, JavaScript және Django кодын реттеуге көмектеседі. Әзірлеушілер тікелей өңдегіште нақты уақытта жөндеу деректерін қарау үшін кірістірілген реттегішті пайдалана алады. Сонымен қатар, олар бір уақытта бірнеше Python процестерін реттей алады және кітапхананы айналып өтіп, кодты айналып өтуі мүмкін. PyCharm, сондай-ақ, әрбір тестілеу сценарийі немесе реттеуші үшін қайта пайдаланылатын және теңшелетін конфигурацияны жасайды.

Пайдаланушыларда тіпті қашықтағы интерпретаторлармен визуалды түзеткішті біріктіріп, қашықтағы түзетуді жеңілдету мүмкіндігі бар.

Кірістірілген терминал.

PyCharm Windows, Linux және mac OS үшін жергілікті терминалдармен жеткізіледі. Кірістірілген терминал программистерге IDE-ден шықпай кодтауды және тестілеуді жалғастыруға мүмкіндік береді. Сонымен қатар, бағдарламашылар Ide-ді Python файлдарын іске қосу және жобаның нақты талаптарына сәйкес Python пайдаланушы ортасын реттеу үшін пайдалана алады. Сонымен қатар, олар ide-де Python немесе Django интерактивті консольдерін іске қосуға болады. Консоль кодты аяқтау, фигуралық жақшаларды автоматты түрде салыстыру және синтаксисті динамикалық өзгерту сияқты пайдалы функцияларды ұсынады. Бағдарламашылар тіпті консольді жергілікті және алыстағы аудармашылармен біріктіре алады.

Негізгі нұсқаларды бақылау жүйелерін қолдау.

PyCharm әзірлеушілерге Git, Mercurial, Perforce және SVN сияқты кең қолданылатын нұсқаларды бақылау жүйелерімен жұмыс істеуге мүмкіндік береді. Ол тіпті автоматты қосу, файлдарды жою және жою сияқты күрделі тапсырмаларды орындайды. Әзірлеушілер тіпті нұсқаларды басқару жүйесін таңдауына қарамастан, IDE ұсынатын бірқатар функцияларды пайдалану мүмкіндігіне ие - кейбір өзгерістер тізіміне жеке өзгерістерді топтастыру, қалпына келтіруге жататын өзгерістерді кейінге қалдыру, әр түрлі пайдаланушылар код қоймасына енгізілетін өзгерістерді мониторингілеу. кодқа енгізілген өзгерістерді жергілікті көшірмеге кірмес бұрын тексеріңіз.

Бағдарламалық қамтамасыз етуді тестілеу.

Басқа IDE сияқты, PyCharm да Python қосымшаларын тестілеуді жеңілдету үшін функциялар мен құралдармен жеткізіледі. Бұл әзірлеушілерге nose, Attest және Doctests сияқты танымал Python тестілеу ортасы арқылы модульдік тестілеуді орындауға мүмкіндік береді. Тестерде кейбір немесе бірнеше тест файлдарын және тест сыныптарын іске қосу мүмкіндігі бар. Олар қосымша ide-мен біріктіре алады Coverage.py қосымшаларды тестілеу кезінде кодты жабуды өлшеу үшін. Көп нүктелі қосымшаларды тестілеу кезінде тестілеушілер қосымшаны толық және тиімді басқару үшін IDE ұсынатын ағын параллелизмінің визуализация опциясын пайдалана алады. Сонымен қатар, PyCharm қолданушыларға мінез-құлық негізінде (BDD) әзірлемелерді енгізе отырып, жоғары сапалы бағдарламалық жасақтаманы ұсынуға мүмкіндік береді.

Қашықтан (онлайн) өңдеу мүмкіндіктері.

PyCharm әзірлеушілерге түрлі машиналарға қосылуға және қашықтан бағдарламалық бағдарламаларды жасауға мүмкіндік береді. Программистер ide ұсынатын кірістірілген SSH консолін компьютерлерге қосылу және SSH арқылы әртүрлі өңдеу міндеттерін орындау үшін пайдалана алады. Олар тіпті жергілікті интерпретаторды қашықтағы интерпретатормен ауыстыра отырып, қашықтан ортада Python қолданбаларын іске қосуға, реттеуге және профильдеуге болады. Сонымен қатар, PyCharm бағдарламашылар vagrant

сияқты сенімді құралдың көмегімен өңделетін даму ортасын құруға және Docker көмегімен таратылған қосымшаларды әзірлеуді жеңілдетуге мүмкіндік береді. Пайдаланушылардың PyCharm-ды проблемаларды бақылау жүйелерімен кедергісіз біріктіруге мүмкіндігі бар.

Жалпы PyCharm-Python үшін ең танымал IDE бірі. Python программистері PyCharm лицензиялық бағдарламалық қамтамасыз ету ретінде пайдалана алады. Дегенмен, JetBrains әзірлеушілерге IDE - қауымдастықтың, кәсіби және білім берудің үш түрлі нұсқасынан таңдауға мүмкіндік береді. Әзірлеушілер әрдайым қауымдастық үшін PyCharm нұсқасын таңдау арқылы бағдарламалық жасақтаманы әзірлеу шығындарын қысқартуы мүмкін. Жалпы шығарылымда кәсіби шығарылымдармен ұсынылатын кейбір кеңейтілген функциялар әлі де жоқ.

JetBrains бағдарламашыларды анағұрлым өнімді жасау және бағдарламалық жасақтаманы әзірлеуді жеңілдету үшін Python IDE екі нұсқасын жаңартады. Мысалы, PyCharm 2017.3 кәсіби нұсқасы API функционалдығын тестілеуді әзірлеушілерге кіріктірілген rest клиентін ұсыну арқылы жеңілдетеді. Сонымен қатар, ол бағдарламашылар Django 2.0 ұсынатын жаңа функцияларды пайдалануға мүмкіндік береді. Сонымен қатар, ол әзірлеушілерге деректерді өңдеу режимі арқылы деректерді тиімді визуализациялауға және талдауға мүмкіндік береді.

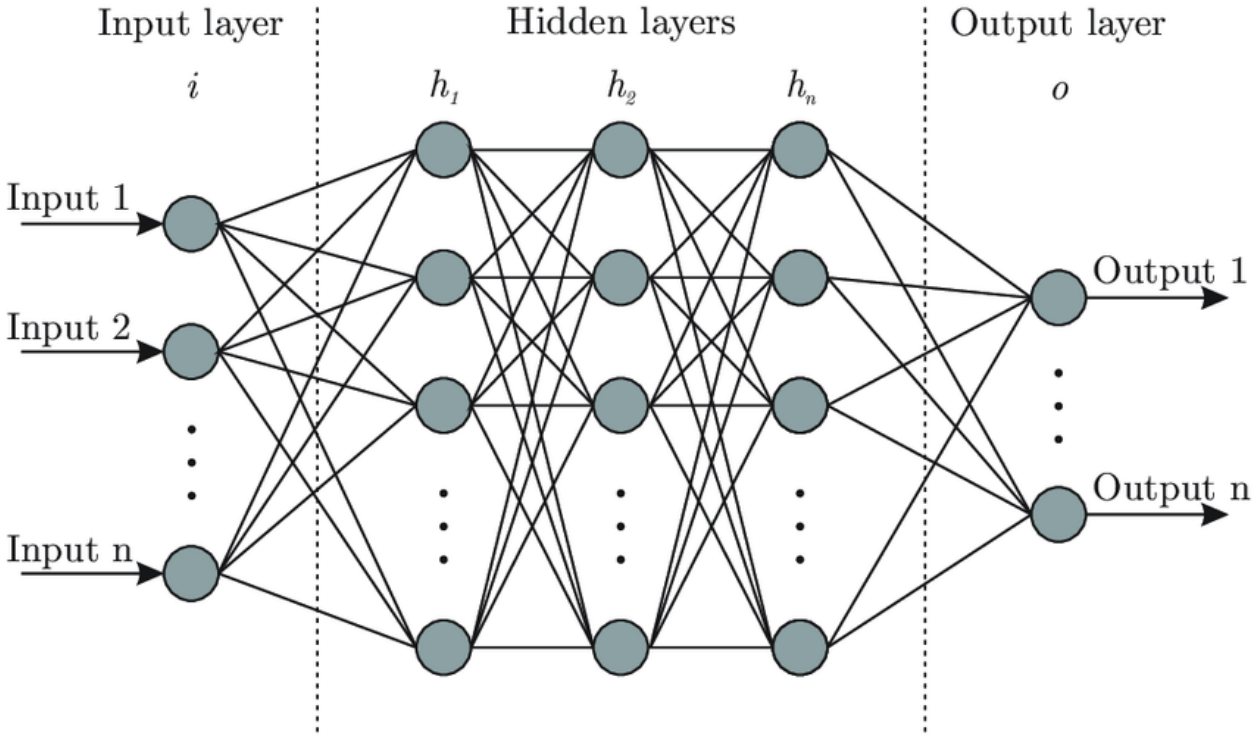
2 Нейрондық желі моделін жобалау

Нейрондық желілерді оқыту қатты шатасып кетуі мүмкін, сол себепті оны алдын ала жоспарлап, әзірлеу керек.

Жобалау барысында келесі сұрақтарға жауап табу керек: Жақсы оқу жылдамдығы неге тең? Қанша жасырын қабат болуы тиіс сіздің желі? Отсев шынымен пайдалы ма? Градиенттер неліктен жоғалады?

Жобалау барысында біз нейрондық желілердің кейбір өзекті аспектілерін анықтаймыз және нейрондық желінің қандай архитектурасын пайдаланатынымызды шешеміз.

2.1 Нейрондық желінің базалық құрылымы



2.1 сурет - Нейрондық желінің базалық құрылымы.

Кіріс нейрондар:

- бұл біздің нейрондық желі өз болжамдар үшін пайдаланатын функциялар саны;

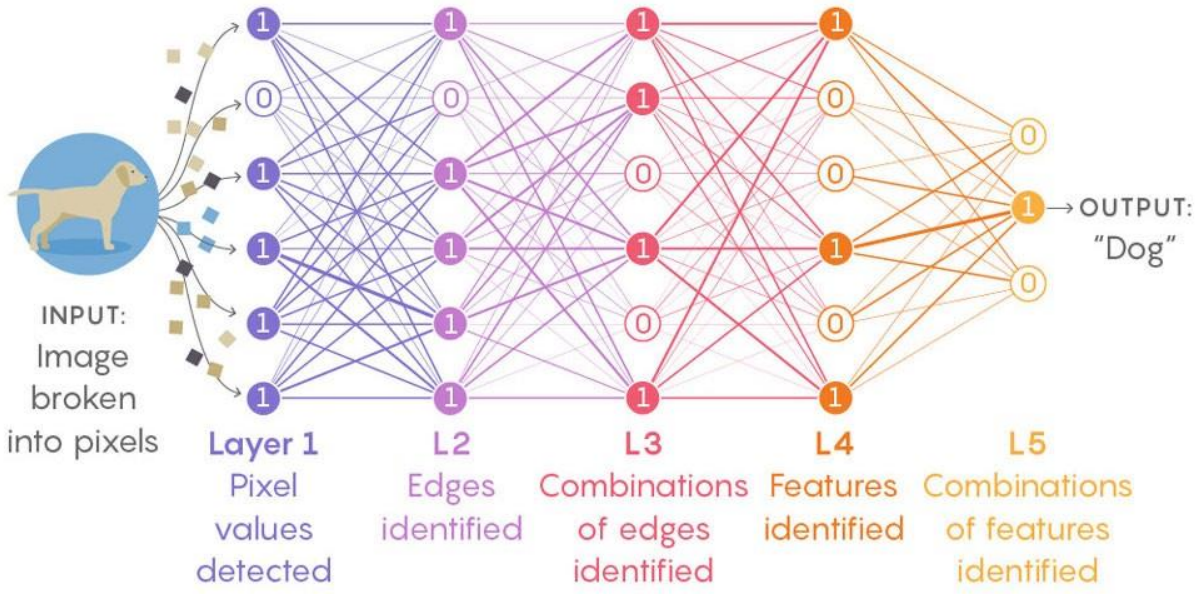
- кіріс векторы әрбір объект үшін бір шығу нейронын қажет етеді. кестелік деректер үшін бұл деректер жиынтығындағы тиісті объектілер саны. сіз осы мүмкіндіктерді мұқият таңдау және жаттығу жиынтығынан тыс сөйлесу мүмкін емес үлгілер болуы мүмкін барлық жою керек (және шамадан тыс құрылғыны тудыруы).

Шығыс нейрондар

- бұл біздің жасағымыз келетін болжамдар саны;
- регрессия: регрессия міндеттері үшін бұл бір мән болуы мүмкін (мысалы, тұрғын үй бағасы). бізде демалыс нейрон бір: команда қандай жеңеді? көп өлшемді регрессия үшін бұл болжамды мәнге бір нейрон (мысалы, шекаралық блоктар үшін бұл 4 нейрон болуы мүмкін — биіктігі, ені, х координаттары, у координаттары үшін бір-бірден);
- жіктеу: екілік жіктеу үшін (спам-спам емес) біз оң класына бір шығу нейронын қолданамыз, және шығу оң класының ықтималдығын білдіреді. көп классты жіктеу үшін (мысалы, данасы автокөлік, ит, үй және т.б. ретінде жіктелуі мүмкін объектілер табылған кезде), біз сыныпқа бір шығу нейронымыз бар және 1-ге дейінгі ықтималдықтардың түпкілікті сомасын қамтамасыз ету үшін шығу қабатындағы softmax активтендіру функциясын пайдаланамыз.

Жасырын қабаттар және ондағы нейрондар саны

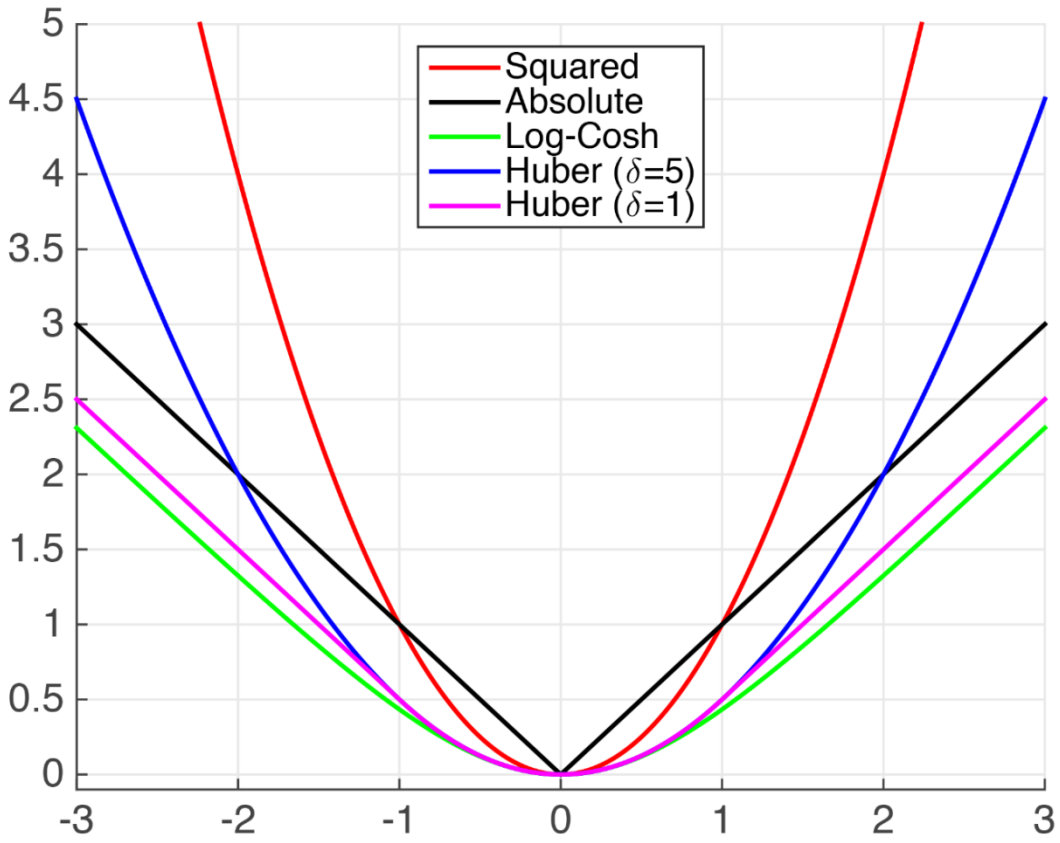
- жасырын қабаттардың саны нейрондық желінің міндеті мен архитектурасына қатты байланысты. біз нейрондық желінің мінсіз архитектурасына өз жолымызды табуға тырысамыз – тым үлкен емес, тым кішкентай емес, ең оңтайлы вариантты табуға тырысамыз;
- әдетте, 1-5 жасырын қабаттар көптеген мәселелер үшін жақсы қызмет етеді. суретпен немесе сөйлеу мәліметтерімен жұмыс істеу кезінде біздің желі ондаған-жүздеген қабаттар болуы қажет болуы мүмкін, олардың барлығы толығымен қосылуы мүмкін. осы жағдайларда пайдалану үшін алдын ала оқытылған модельдер (yolo, reset, vga) бар, олар өз желілерінің үлкен бөліктерін пайдалануға және өз моделін осы желілердің жоғарғы жағынан ғана жоғары ретті функцияларды зерттеуге үйретуге мүмкіндік береді. бұл жағдайда модель әлі де оқу үшін бірнеше қабаттар болады;
- жалпы, барлық жасырын қабаттар үшін бірдей нейрондарды пайдалану жеткілікті. кейбір деректер жиынтықтары үшін үлкен бірінші қабаттың болуы және одан кейінгі қабаттарда жоғары тәртіптегі бірнеше объектілермен қоректенуі мүмкін төмен деңгейдегі көптеген объектілерді зерттеуге болады [13].



2.2 сурет - Тану моделінің жұмыс істеу мысалы.

- әдетте, біз әрбір қабатта нейрондардың санын үлкейткеннен емес, қабаттардың санын үлкейткенде өнімділікті арттырамыз;
- біз қайта құру бастағанға дейін 1-5 қабаттар мен 1-100 нейрондарды бастау және баяу жаңа қабаттар мен нейрондарды қосу ұсынылады. "жасырын қабаттар + жасырын нейрондар" қандай комбинациясын көру үшін таразылар мен жылжулар тақтасындағы жоғалтулар мен дәлдігін қадағалауға болады;
- қабаттардың / нейрондардың аз санын таңдағанда, егер бұл сан тым аз болса, сіздің желі сіздің деректеріңізде негізгі паттерналарды зерттей алмайды және, демек, пайдасыз болады. бұл әрекет тәсілі жасырын қабаттар + жасырын нейрондардың үлкен санынан бастау, содан кейін сіз үшін нейрондық желі өлшемін азайту үшін бөліктер мен ерте тоқтауды пайдалану болып табылады. тағы да, мәселе үшін мінсіз желі өлшемін анықтау үшін бірнеше комбинацияларды көріңіз және салмақ және жылжу панелінде өнімділікті бақылау ұсынылады;
- андрей карпати сондай - ақ overfit then regularize тәсілін ұсынады - " алдымен ол қайта құрылуы мүмкін жеткілікті үлкен моделін алу (яғни, оқуды жоғалтуға назар), содан кейін тиісті түрде оны реттеу (валидация жоғалтуды жақсарту үшін кейбір оқу жоғалтудан бас тарту).”

Шығын функциясы



2.3сурет - Функциялардың жұмысы көрсеткіші.

- регрессия: орташа квадраттық қате, егер шығарындылардың елеулі саны болмаса, оңтайландыру үшін шығындардың ең көп таралған функциясы болып табылады. бұл жағдайда орташа абсолютті қатені немесе хубердің жоғалуын пайдаланыңыз;
- жіктеу: кросс-энтропия көп жағдайларда жақсы қызмет етеді;

Пакет өлшемі.

1) Пакеттердің үлкен өлшемдері үлкен опция болуы мүмкін, себебі олар бір рет оқыту даналарының көп санын өңдеу үшін графикалық процессорлардың қуатын пайдалана алады. open air пакеттің үлкен өлшемі (кескіндерді жіктеу және тілдік модельдеу үшін ондаған мың, сондай-ақ rl-агенттер жағдайында миллиондаған) масштабтау және параллельдеу үшін жақсы екенін анықтады.

2) Дегенмен, партияның аз мөлшерін жасау керек жағдайлар болады. шебер және ласки мақаласына сәйкес, үлкен пакеттерді орындау кезінде параллельді көбейтуден алынған артықшылық өнімділіктің жоғары жинақталуымен және аз пакеттерге жеткен аз жады көлемімен өтеледі. олар пакеттердің үлкейтілген өлшемдері тұрақты конвергенцияны қамтамасыз ететін оқу жылдамдығының рұқсат етілген ауқымын азайтатынын көрсетеді.

олардың шығарылуы - аз мөлшері, шын мәнінде, жақсы; ең жақсы өнімділікке шағын партия есебінен 2-ден 32-ге дейін жетеді.

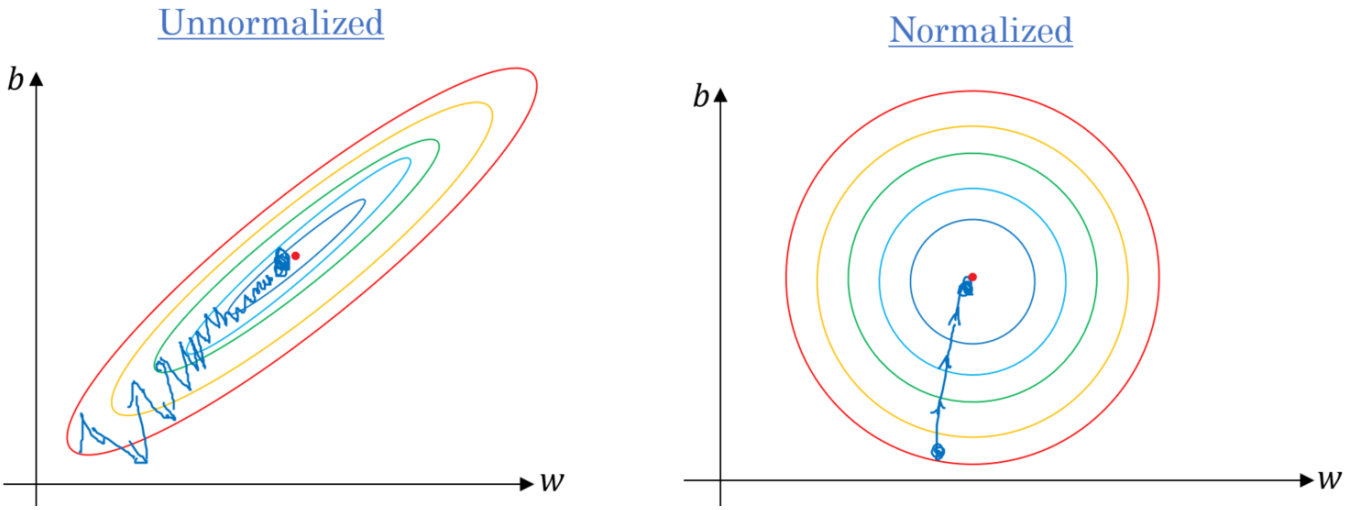
3) Партияның аз өлшемдерінен бастау және біртіндеп өлшемін арттыру және ең жақсы қиыстырып анықтау үшін таразылар мен жылжулар панеліндегі өнімділікті бақылау ұсынылады.

Дәуірдің саны.

Дәуір - оқыту процесінде бір итерация, ол оқу жиынынан барлық мысалдарды көрсету және бақылау жиынында оқыту сапасын тексеру мүмкіндігін қамтиды.

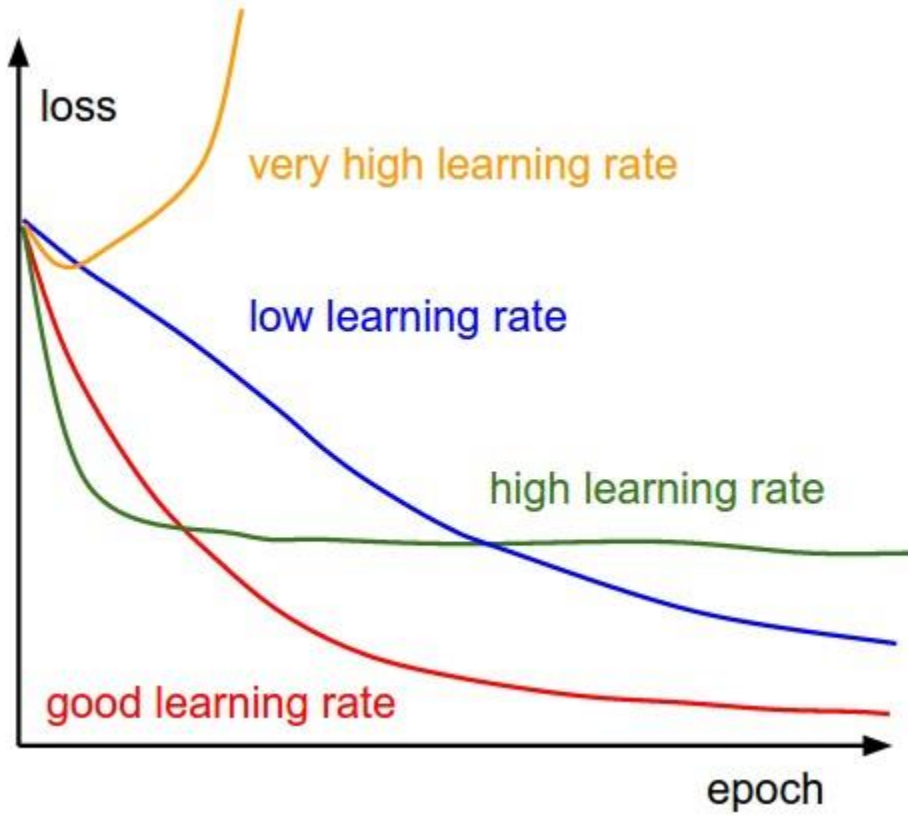
Өнімділік артуын тоқтатқан кезде оқуды тоқтату үшін дәуірдің көп мөлшерде бастау және ерте тоқтату пайдалану ұсынылады.

Сипаттардың масштабталуы



2.4 сурет - Нейрондық желінің нормализациясы

Қысқаша ескерту: біздің нейрондық желі үшін кіріс деректер ретінде пайдаланбас бұрын біздің барлық функцияларымыз бірдей ауқымға ие екеніне көз жеткізу керек. Бұл тез конвергенцияны қамтамасыз етеді. Біздің функцияларымыз әр түрлі ауқымға ие болған кезде (мысалы, жалақы мың және ондаған жылдық тәжірибе), шығындар функциясы сол жақтағы сурет сияқты көрінеді. Бұл сіздің оңтайландыру алгоритмі нормаланған объектілерді пайдалану салыстырғанда алқапты кесіп көп уақыт алады дегенді білдіреді (оң жақта). [14]



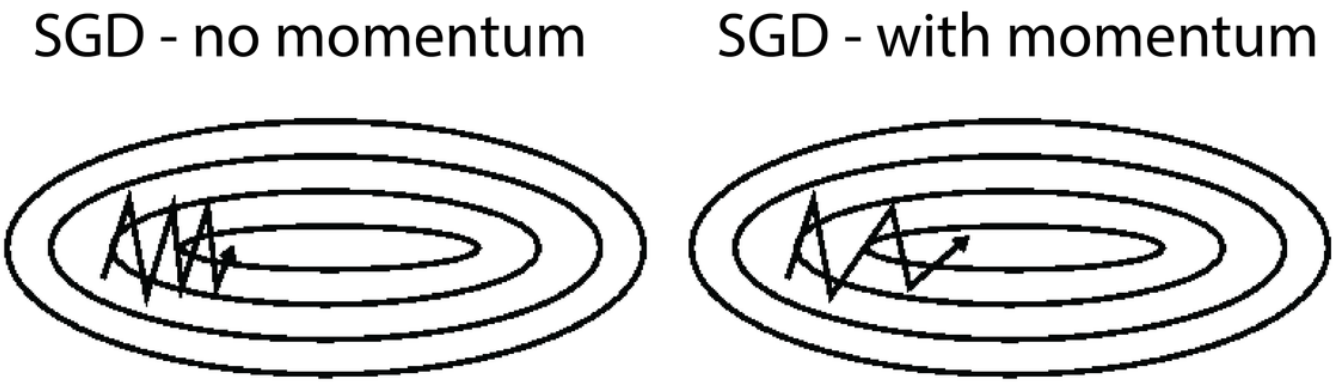
2.5 сурет - Оқыту жылдамадығы мысалдары.

- егер біз оқытудың өте жақсы өтетініне көз жеткізгіміз келсе, оқу жылдамдығын таңдау өте маңызды;

- ең жақсы оқу жылдамдығын табу үшін өте төмен мәннен (10^-6) бастау керек және оны өте жоғары мәнге (мысалы, 10) жеткенше константаға баяу көбейту керек. таразылар мен жылжулар тақтасындағы модельдің өнімділігін ұдайы өлшеп тұру керек (оқу жылдамдығы журналымен салыстырғанда). содан кейін осы оңтайлы оқу жылдамдығын пайдалана отырып, үлгіні қайта оқытуға болады;

- оқытудың ең ыңғайлы жылдамдығы әдетте оқыту жылдамдығының жартысын құрайды, бұл модельдің алшақтығына әкеледі. жиі ілеспе кодында learn_rate үшін әртүрлі мәндерді орнату және оқыту жылдамдығына қатысты өз түйсігін дамыту үшін модель өнімділігіне қалай әсер етеді көру керек;

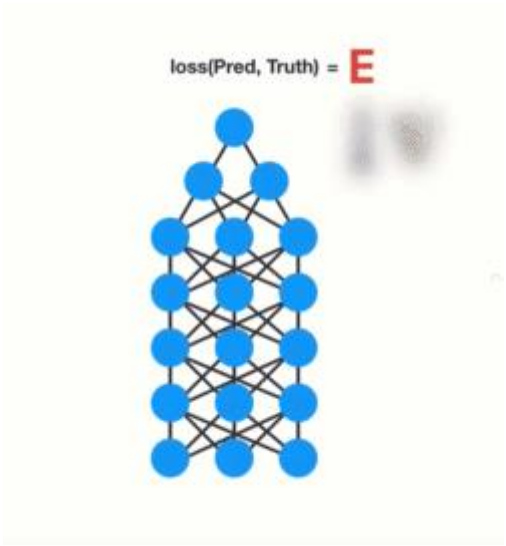
- лесли смит ұсынған оқу жылдамдығын іздеу әдісін қолдануға болады. бұл көптеген градиентті оңтайландырғыштар үшін жақсы оқу жылдамдығын табудың тамаша тәсілі (көптеген sgd нұсқалары) және көптеген желілік сәулетпен жұмыс істей алады.



2.6 сурет - Градиентті түсу мысалдары.

- градиент түсіру жергілікті минимумдарға кішкентай дәйекті қадамдар жасайды, және градиенттер кішкентай болған кезде, концвертацияға көп уақыт кетуі мүмкін. екінші жағынан, импульс алдыңғы градиенттерді ескереді және конвергенцияны жылдамдатады, алаптар бойынша қозғалысты жылдамдатады және жергілікті минимумдарды болдырмайды [3];
- жалпы, импульстің мәні бір мәніне өте жақын болуы керек. 0.9-бұл шағын деректер жиынтығы үшін бастау үшін жақсы орын, және біз бірге (0.999) жақындаған сайын, біздің деректер жиынтығымыз да өседі.

Жоғалатын және жарылатын импульстер



2.7 сурет - Жарылатын және жоғалатын градиенттер

- тура адамдар сияқты, нейрондық желінің барлық қабаттары бірдей жылдамдықпен үйренбейді. осылайша, backprop алгоритмі бірінші қабаттарға шығу қабатынан қате градиентін тарататын кезде, градиенттер бірінші қабаттарға жеткен кезде дерлік аз болады. бұл бірінші қабаттардың салмағы әр қадамда айтарлықтай жаңартылмайды дегенді білдіреді;
- бұл жоғалып бара жатқан градиенттер мәселесі. белгілі бір қабаттар үшін градиенттер көп және одан да көп болған кезде жарылатын

градиенттерге ұқсас проблема пайда болады, басқалардан айырмашылығы, бұл кейбір қабаттар үшін салмақ жаппай жаңартуға әкеледі;

- жоғалып бара жатқан градиентке қарсы тұрудың бірнеше жолы бар [15].

2.2 Активация функциялары

Активация функциялары - бұл нейронның қосылатының немесе қосылмайтыны туралы шешім қабылданатын, кіріс мәліметтері мен шығындардың өлшенген сомасын есептеу үшін нейрондық желілерде қолданылатын функциялар. Әдетте ол берілген градиентті өңдеудің кейбір градиентінің көмегімен берілген мәліметтерді өңдейді және деректерде параметрлері бар нейрондық желі үшін келесі нәтижені алады. Кейбір әдеби дереккөздердегі бұл функциялар көбінесе беру функциясы деп аталады.

Активация функциясы ол ұсынатын функцияға байланысты сызықтық немесе сызықты емес болуы мүмкін және объектілерді тану мен жіктеуден бастап, сөйлеуді тануға, сегментацияға, көріністерді түсінуге және сипаттауға, машинаны аударуға тестілеуге, әртүрлі бағыттардағы нейрондық желілерден шығуды бақылау үшін қолданылады. ерте зерттеу нәтижелері бар сөйлеу жүйелері, қатерлі ісік ауруын анықтау жүйелері, саусақ іздерін анықтау, ауа-райын болжау, өздігінен жүретін автомобильдер және басқа да салалар активтендіру функцияның дұрыс таңдау нейрондық желінің есептеу нәтижелерін жақсартады деп тұжырымдайды.

Активтендіру функциясының желілік құрылымдағы орны оның желідегі функциясына байланысты, сондықтан активтендіру функциясы жасырын қабаттардан кейін орналастырылған кезде, зерттелген сызықтық картографияларды таратудың сызықтық емес формаларына түрлендіреді және шығыс қабатында болжауды орындайды.

Терең архитектуралар бірнеше өңдеу қабаттарынан тұрады, олардың әрқайсысы сызықты және сызықты емес операцияларды қамтиды, олар осы мәселені шешу үшін бірге зерттеледі. Бұл терең желілер жақсы сипаттамаларға ие, дегенмен, жалпы проблемалар, мысалы, түсіп кететін градиенттер және жарылғыш градиенттер, әдетте 1-ден төмен туынды терминдерден туындайды, егер осы туынды терминдерді дәйекті көбейткенде, шамасы кішірейіп, нөлге айналады. градиент түседі. Сондықтан, егер мәндер 1-ден үлкен болса, онда жүйелі көбейту мәндерді арттырады және градиент шексіздікке ұмтылады, осылайша градиент жарылып кетеді. Осылайша, активтендіру функциялары белгілі бір шектерде осы градиенттердің мәндерін қолдайды. Оларға әр түрлі математикалық функцияларды қолдану арқылы қол жеткізіледі, ал нейрондық желілерді есептеу үшін пайдаланылатын активтендірудің кейбір алғашқы ұсыныстарын Эллиот 1993 жылы нейрондық желіде активтендіру функцияларын қолдануды зерттеген кезде зерттеген.

Ең көп тараған активация функциялары:

1) Сигма Функциясы. Кейбір әдеби көздерде сигмоид белсендіру функциясы кейде логистикалық функция немесе жаншу функциясы деп аталады. Сигма тәрізді функцияны зерттеу нәтижелері терең оқыту қолданылатын сигма тәрізді активациялық функцияның үш нұсқасын алуға мүмкіндік берді. Сигмоида-бұл тікелей байланыстың нейрондық желілерінде қолданылатын сызықты емес активациялық функция. Бұл барлық жерде оң туынды және тегістіктің кейбір дәрежесі бар нақты кіріс мәндері үшін анықталған шектеулі сараланатын заттай функция.

Сигналдық функция DL архитектурасының Шығыс қабаттарында пайда болады және олар қорытынды негізінде ықтималдылықты болжау үшін пайдаланылады және бинарлық жіктеу, логистикалық регрессия есептерін модельдеу, сондай-ақ басқа да есептерде сәтті қолданылды. Сонымен қатар, ол сигналдық функцияның негізгі артықшылықтарын көрсетеді, ол оңай түсінікті және негізінен таяз желілерде қолданылады. Сонымен қатар, Gloros және Bengio, 2010 сигмоидты активтендіру функциясын нейрондық желіні шағын кездейсоқ таразылардан инициализациялау кезінде болдырмау керек деп болжайды.

Дегенмен, Сигма белсендіру функциясы күрделі кемшіліктерден зардап шегеді, олар күрт ылғалды градиенттерді қамтитын терең жасырын қабаттардан кіріс қабаттарына кері тарату кезінде, градиенттің қанығуы, баяу конвергенция және нөлдік емес орталықтанған шығу, осылайша жаңарту градиентін шақыра отырып, әр түрлі бағытта тарайды. Активтендіру функциясының басқа да нысандары ұсынылды, оның ішінде белсендіру сигма тәрізді функциясына тән кейбір кемшіліктерді жою үшін қызмет көрсететін гиперболалық жанама функция [4].

A. Қатты Сигма функциясы (SiLU): қатты сигма тәрізді белсендіру - сигма тәрізді белсендіру функциясының тағы бір нұсқасы.

Қатты сигмоиды жұмсақ сигмоидамен салыстыру қатты сигмоида жоғары көрсетілгендей мамандандырылған аппараттық және бағдарламалық нысанда іске асырылған кезде аз есептеу құнын қамтамасыз ететінін көрсетеді және авторлар терең желілер негізінде бинарлық жіктеудің міндеттері бойынша кейбір перспективалық нәтижелерді көрсетті.

B. Сигмоид-өлшенген сызықтық бірлік (dSiLU): бұл оқуды күшейтуге негізделген аппроксимациялық функция. SiLU функциясы тек қана терең нейрондық желілердің жасырын қабаттарында және тек базалық жүйе оқытуды нығайту үшін пайдаланылуы мүмкін.

C. Сигма тәрізді-өлшенген сызықтық бірліктерден (dSiLU) туынды: сигма тәрізді-өлшенген сызықтық бірліктерден туынды SiLU функциясының градиенті болып табылады және dSiLU деп аталады.

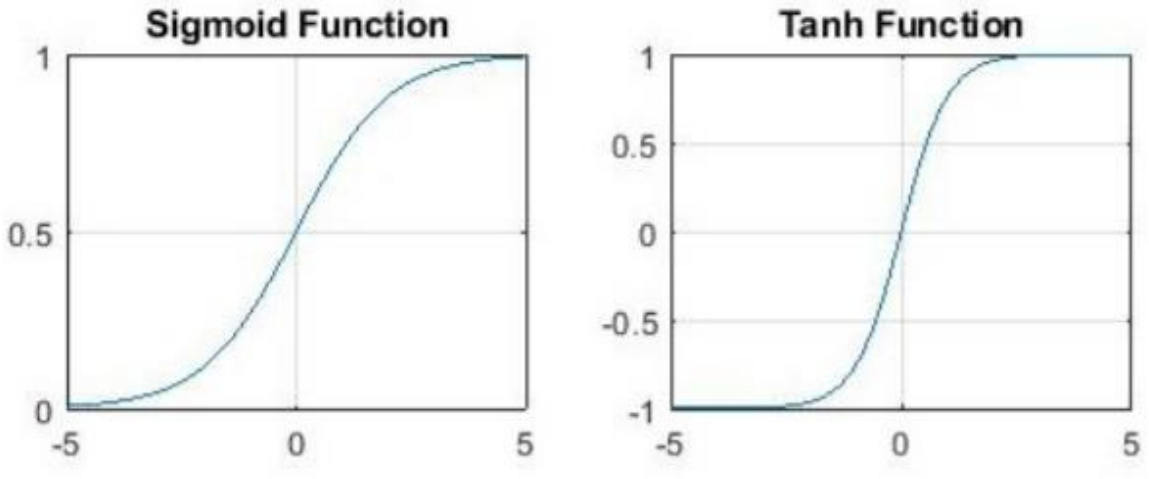
2) Гиперболалық Тангенс Функциясы (Tanh). Гиперболалық жанама функция-бұл терең оқытуда қолданылатын белсендіру функциясының тағы бір түрі және ол терең оқыту қосымшаларында қолданылатын кейбір нұсқалары бар. Tanh функциясы ретінде белгілі гиперболалық жанама

функция-1-ден 1-ге дейінгі диапазоны бар тегіс нөлдік орталандырылған функция.

Tanh функциясы ол көп қабатты нейрондық желілер үшін жақсы жаттығу өнімділігін береді мағынада Сигма функциясымен салыстырғанда қолайлы функция болды. Алайда, tanh функциясы жоғалып бара жатқан градиент мәселесін, сондай-ақ сигмоидты функцияларды шеше алмады. Бұл функциямен қамтамасыз етілетін басты артықшылық, ол кері тарату процесіне көмектесе отырып, нөлдік орталықтанған шығуды өндіреді.

Tanh функциясының қасиеті ол тек 1 градиентке қол жеткізе алады, тек кіріс сигналының мәні 0-ге тең болғанда, яғни x нөлге тең болғанда. Бұл есептеу кезінде tanh кейбір өлі нейрондарды өндіруге мүмкіндік береді. Өлі нейрон-бұл белсенді масса, нөлдік градиенттің нәтижесінде сирек қолданылатын жағдай. Бұл tanh функциясының шектелуі проблеманы шешу үшін активтендіру функциялары саласындағы әрі қарай зерттеулер жүргізді және ол түзетілген сызықтық бірлі-жарым активтендіру функциясын (ReLU) тудырды.

Tanh функциялары негізінен табиғи тілді және сөйлеуді тану есептерін өңдеу үшін рекуррентті нейрондық желілерде қолданылды



2.8 сурет - Сигмоид және тангенс функцияларының көрсеткіші.

Қатты гиперболалық Функция: Hardtune функциясы ретінде белгілі қатты гиперболалық функция терең оқыту қосымшаларында қолданылатын tanh белсендіру функциясының тағы бір нұсқасы болып табылады. Hardtanh-бұл oftheh арзан және одан да көп есептеу тиімді нұсқасы [17].

Hardtanh функциясы табиғи тілді өңдеуде сәтті қолданылды, және авторлар оның жылдамдығы мен дәлдігін жақсартады деп хабарлайды.

3) Softmax Функциясы. Softmax функциясы-нейрондық есептеулерде қолданылатын белсендіру функциясының тағы бір түрі. Ол нақты сандардың векторы бойынша ықтималдық үлестіруін есептеу үшін қолданылады. Softmax

функциясы 0-ден 1-ге дейінгі мәндердің диапазонын білдіретін шығыс сигналын береді.

Softmax функциясы көп класты модельдерде қолданылады, онда ол әр сыныптың ықтималдығын қайтарады, сонымен қатар мақсатты сыныптың ең үлкен ықтималдығы бар. Softmax функциясы негізінен олар пайдаланылатын терең оқыту сәулетінің барлық шығыс қабаттарында пайда болады.

Сигмоид пен softmax активтендіру функциясы арасындағы негізгі айырмашылық сигмоид екілік жіктеуде қолданылады, ал Softmax көп өлшемді жіктеме есептерін шешу үшін қолданылады.

4) Түзетілген Сызықты Блоктың Функциясы (ReLU). Түзетілген желілік блокты (ReLU) іске қосу функциясы 2010 жылы Наир және Хинтон ұсынған және содан бері ең тиімді болып табылады. бүгінгі таңда ең заманауи нәтижелермен терең оқыту қосымшалары үшін белсендіру функциясы кеңінен қолданылады. Rely-бұл ең табысты және кең қолданылатын функция болып табылатын АФФ оқыту жылдамырақ. Ол Сигма және Tanh-белсендіру функцияларымен салыстырғанда терең оқытудағы үздік өнімділік пен қорытуды ұсынады. Rally шамамен сызықтық функция болып табылады және сондықтан сызықтық модельдердің қасиеттерін сақтайды, бұл оларды оңай оңтайландырады, градиентті түсіру әдістері.

Real белсендіру функциясы нөлден аз мәндер орнатылған әрбір кіріс элементі үшін шекті әрекетті орындайды.

Бұл функция мәні нөлден аз кіріс сигналдарының мәндерін түзетеді, осылайша оларды нөлге мәжбүрлейді және функцияны активтендірудің ерте түрлерінде байқалатын жойылып бара жатқан градиентті жою. Relay функциясы басқа АФ-мен терең нейрондық желілердің жасырын блоктарында қолданылған, ол жүйенің Шығыс қабаттарында қолданылатын, объектілер жіктеуінде және сөйлеуді тану үшін қосымшаларда табылған типтік мысалдармен.

Relu-ді есептеудің басты артықшылығы-олар тез есептеуге кепілдік береді, себебі бұл экспоненттер мен бөліктер шынымен есептелмейді, сонымен бірге есептеудің жалпы жылдамдығы артады. Rally тағы бір қасиеті-бұл жасырын өлшем бірліктеріне ыдырауды енгізеді, өйткені ол нөлден максимумға дейін мәндерді қысады. Дегенмен, ReLU Сигма функциясымен салыстырғанда кулакты оңай ілінеді, бірақ сору әдісі релаксациялық және түзетілген желілерді қайта жабдықтау әсерін азайту үшін қабылданған болса да, терең нейрондық желілердің жұмысын жақсарту [6].

ReLU және оның опциялары шектеулі Больцман машина және нейрондық желілердің тесу архитектурасын қамтитын түрлі терең оқыту сәулетшілерінде қолданылған, дегенмен (Nair and Hinton, 2010) ReLU оның қарапайымдылығы мен сенімділігіне байланысты көптеген сәулетшілерде қолданылған.

Riu-да елеулі шектеу бар, ол кейде жаттығу кезінде нәзік болып табылады, бұл кейбір градиенттердің өліміне алып келеді. Бұл кейбір

нейрондар өлі болып табылады, өйткені өлі нейрондар нөлдік белсендіруді береді. Өлі нейрондармен проблемаларды шешу үшін "тесік Rely"ұсынылды.

A."тесік LReLU": 2013 жылы АФ ретінде ұсынылған, ол бүкіл көбею процесі кезінде тірі салмақтың жаңартылуын қолдау және қолдау үшін қалжыңдауға сәл теріс көлбеу енгізеді. Альфа-параметр градиенттер жаттығу кезінде кез келген уақытта нөлге тең болмайды, сондықтан өлі нейрондарды релаксация мәселесін шешу ретінде енгізілді. LReLU теріс градиент α үшін өте аз тұрақты мәні бар градиентті 0,01 диапазонында есептейді.

LReLU стандартты жауаппен салыстырғанда бірдей нәтиже береді, ол осылайша нөлдік емес градиенттер бар қоспағанда, барлық уақыт кезеңі ішінде relu және tanh стандартты функцияларымен салыстырғанда сиретілген және дисперсияны қоспағанда, нәтижені айтарлықтай жақсарту жоқ деп болжауға болады. LReLU сөзді автоматты тану деректер жиынтығында сыналды.

B. Параметрлік түзетілген сызықтық бірліктер (PReLU): prelu ретінде белгілі параметрлік ReLU, оларға ұсынылған ReLU AF тағы бір нұсқасы болып табылады, 2015, және PReLU адаптивті зерттелген болып табылатын функцияның теріс бөлігі бар, оң бөлігі сызықтық болып табылады.

Авторлар PReLU өнімділігі relu-ге қарағанда үлкен масштабты бейнелерді тану жақсы екенін хабарлады, және PReLU-ден осы нәтижелер көрнекі тану бойынша адам деңгейінің өнімділігін асып кеткен бірінші болды [].

C. Рандомизацияланған "тесік ReLU"(RReLU): рандомизацияланған "тесік ReLU" - бұл тесілген ReLU динамикалық нұсқасы, мұнда кездейсоқ сан U (l, u) біркелкі таралудан таңдалған, желіні оқыту үшін пайдаланылады. RReLU келесі формула бойынша беріледі:

2.3 Терең оқыту архитектурасында қолданылатын активациялық функциялардың даму үрдістерін салыстыру

Активациялық функцияларды іздеу ImageNet Image Large Scale Visual recognition Challenge (ILSVRC) конкурсының жеңімпаздарының жарияланған зерттеу нәтижелеріне, сондай-ақ әдебиетте табылған активациялық функцияларды зерттеу нәтижесінде алынған зерттеулердің кейбір дәйексөзделген нәтижелеріне негізделген. Imagine байқауы үрдістерді салыстыру үшін таңдалып алынды,өйткені дәл осы байқау терең оқытудағы алғашқы жетістікті әкелді. Олар осы мақалада сипатталған функцияларды іздеу үшін, сондай-ақ белсендіру функцияларын пайдалану үрдістерін зерттеу үшін негізгі негізді қалыптастырады, бұл үрдістердің аумағы суреттерді тану шеңберінен шығып кетсе де, терең оқытуды белсендірун жаңа функцияларын тудырған басқа да қосымшалар табиғи тілді өңдеу сияқты енгізілген.

ImageNet ретінде белгілі (ILSVRC) бейнелерді визуалды тану үлкен масштабты міндеті-визуалды тану бойынша жарыстарды өткізу үшін пайдаланылатын бейнелердің деректер базасы. Imagine байқауы – бұл жыл

сайынғы жарыс, онда зерттеушілер мен олардың командалары визуалды тану міндеттерінде қол жеткізген дәлдікті жақсартуды қарастыру үшін нақты деректер жинағында әзірленген алгоритмдерді бағалайды [9].

Терең оқыту сәулеті – бұл көп қабатты персептрон деп аталатын бірнеше жасырын қабаты бар архитектура. Бұл архитектуралар көп және өзіне терең тікелей нейрондық желілер, орайтын нейрондық желілер, ұзақ мерзімді қысқа мерзімді жады, Рекуррентік нейрондық желілер және Больцманның терең машиналары, сенімдің терең желілері, генеративті жарыс желілері және т.б. сияқты терең генеративтік модельдер кіреді. Терең оқыту архитектурасын пайдалану деректердегі заңдылықтарды зерделеуді, кейбір кіріс функцияларын шығу мүмкіндіктерімен салыстыруды және т. б. қамтиды.

Терең оқыту сәулеттерінде пайдаланылатын белсендіру функциялары нейрондық желілерді есептеу үшін неғұрлым терең сәулеттерді енгізу алдында шығады. 2012 жылға дейін суреттерді тану міндеттерінде терең емес архитектуралар пайдаланылды, және оларда іске қосу функциясы аз болды. Нейронды есептеулерге арналған неғұрлым терең архитектураны енгізу алғаш рет 2012 жылы Крижевский бейнелерді визуалды танудың ауқымды міндеті шеңберінде зерттелді, ол 2010 жылы басталған жыл сайынғы іс-шара болып табылады. Содан бері терең сәулетті пайдалану терең сәулетті оқыту үшін оңтайландыру әдістері саласындағы үлкен ғылыми жетістіктердің куәсі болды, себебі желі тереңірек болған сайын, оқыту және оңтайландыру қиын.

Активация функциясы - терең үйрету архитектурасының әр түрлі деңгейлерінде жүзеге асырылатын нейрондық желілерді оқыту және оңтайландыру үшін негізгі компонент және барлық салаларда, соның ішінде табиғи тілді өңдеу, объектілерді анықтау, жіктеу және сегментация және т.б. қолданылады [10].

1-кестеде осы архитектураларда пайдаланылатын активтендіру функциялары туралы позицияларды көрсететін бейнелерді (ILSVRC) танудың ірі ауқымды міндеті үшін пайдаланылатын қолданыстағы архитектураларды едәуір жақсарту нәтижесінде пайда болған терең нейрондық желілердің кейбір қазіргі заманғы архитектуралары келтірілген.

TABLE I

TYPES AND POSITIONS OF AFS USED IN DL ARCHITECTURES.

Architecture	Hidden Layers	Output Layer	Cross Reference
AlexNet	ReLU	Softmax	Krizhevsky et al., 2012
NiN	No activation	Softmax	Lin et al., 2013
ZFNet	ReLU	Softmax	Zeiler & Fergus, 2013
VGGNet	ReLU	Softmax	Simonyan & Zisserman, 2015
SegNet	ReLU	Softmax	Badrinarayanan et al., 2015
GoogleNet	ReLU	Softmax	Szegedy et al., 2015
SqueezeNet	ReLU	Softmax	Golkov et al., 2016
ResNet	ReLU	Softmax	He et al., 2016
ResNeXt	ReLU	Softmax	Xie et al., 2017
MobileNets	ReLU	Softmax	Howard et al., 2017
SeNet	ReLU	Sigmoid	Fu et al., 2017

2.9 сурет - Активация функцияларының қазіргі таңда қолдану жиілігі

Осы кестеде ReLU және Softmax белсендіру функциялары терең оқытудың практикалық қосымшаларында қолданылатын белсенді емес функциялар болып табылады. Сонымен қатар, Softmax функциясы терең оқытудың ең көп таралған практикалық Қосымшаларының Шығыс қабатында қолданылады, алайда терең оқытудың ең соңғы архитектурасы шығу қабатындағы оның болжамына жету үшін сигмоидты функцияны пайдаланды, ал ReLU блоктары жасырын қабаттарда қолданылады [17].

Қазіргі заманғы архитектураны таңдау кезінде тану міндеттері үшін 1-кестеден ағымдағы және ең соңғы архитектураны таңдау және желінің архитектуралық құрамын түсіну ыңғайлы.

Жасырын Қабатты Белсендіру

Жалпы алғанда, әр түрлі активтендіру функцияларын пайдалану төмендегі тәртіпте жақсарады (ең төменгі → ең жоғары көрсеткіштен): logistic → tanh → ReLU → Leaky ReLU → ELU → SELU.

ReLU – ең танымал белсендіру функциясы, және егер біз іске қосу функциясын орнатқымыз келмесе, онда Rely-бастау үшін тамаша вариант. Сондай-ақ, келесі нұсқаларды көруге болады:

- нейрондық желілерді асқын кернеумен күресу үшін: RreLU;
- орындау кезінде кідірісті азайтыңыз: leaky ReLU;
- жаппай оқыту торлары үшін: PReLU;
- нәтижені жылдамдату үшін: leaky ReLU;
- Егер желі қалыпты болмаса: ELU;
- жалпы сенімді іске қосу функциясы үшін: SELU.

Шығыс қабатын белсендіру

Регрессия: регрессия мәселелері олардың шығу нейрондары үшін іске қосу функцияларын талап етпейді, себебі біз шығу деректері кез келген мәнді қабылдауды қалайды. Мәндер белгілі бір диапазонмен шектелгенін қаласаңыз, біз tanh 1 → 1 мәндеріне және 0 → 1 мәндеріне арналған логистикалық

функцияны пайдалана аламыз. Біз тек оң нәтиже іздеген жағдайда, біз softplus белсендіруді пайдалана аламыз.

Жіктеу: шығу мәні 0-ден 1-ге дейінгі ауқымда екеніне көз жеткізу үшін екілік жіктеу үшін сигмоид іске қосу функциясын пайдаланыңыз. Softmax қолданыңыз мультиклассикалық жіктеу үшін шығу мәнінің ықтималдығы 1-ді құрайды.

3 Программалық қамтаманы жүзеге асыру және тестілеу

3.1 Бағдарлама құрылымы

Бағдарлама бірнеше өзара әрекеттесетін бөліктерден тұрады. Әрбір бағдарлама .py кеңейтімі бар мәтіндік файлда сақталады, бұл кеңейту бұл бағдарлама python-да орындалатынын жүйеге түсінуге мүмкіндік береді. Әрбір жеке файл (код бөлігі) нақты функцияны орындайды. Төменде файлдар тізімі:

- availableStats;
- configureCWD;
- createModel;
- customHeaders;
- getDailyMatchups;
- getStats;
- makePastPredictions;
- nbaPredict;
- standardizeStats;
- teamIds.

Одан әрі біз әрбір файл бойынша өтіп, оның қалай жұмыс істейтінін, жұмыста қандай құралдарды пайдаланатынын, не үшін қажет екенін, қандай функцияны орындайтынын анықтаймыз.

3.2 AvailableStats

Файл атауынан анық, бұл бағдарлама статистикамен байланысты және бізге ойындар бойынша қолжетімді статистиканы сақтауға мүмкіндік береді. Бұл келесідей жасайды:

```
availableStats = {'W_PCT':'Base',
  'REB':'Base',
  'TOV':'Base',
  'PLUS_MINUS':'Base',
  'OFF_RATING':'Advanced',
  'DEF_RATING':'Advanced',
  'TS_PCT':'Advanced'}
```

Бұл файлда барлық қалғандары сияқты үлкен код жоқ. Мұнда бір ғана деректер түрі бар-сөздік.

Python сөздіктер-кілтпен қол жетімді нысандардың ретсіз коллекциясы. Олар кейде ассоциативті массив немесе хеш-кесте деп аталады.

Сөздікпен жұмыс істеу үшін оны құру керек. Оны зерттеу әдісімен жасауға болады. Біріншіден, Литер көмегімен:

```
>>> d = {}
>>> d
{}
```

```
>>> d = {'dict': 1, 'dictionary': 2}
>>> d
{'dict': 1, 'dictionary': 2}
```

Екіншіден, dict функциясы арқылы:

```
>>> d = dict(short='dict', long='dictionary')
>>> d
{'short': 'dict', 'long': 'dictionary'}
>>> d = dict([(1, 1), (2, 4)])
>>> d
{1:1, 2:4}
```

Үшіншіден, кілттер элементінің көмегімен:

```
>>> d = dict.fromkeys(['a', 'b'])
>>> d
{'a': None, 'b': None}
>>> d = dict.fromkeys(['a', 'b'], 100)
>>> d
{'a': 100, 'b': 100}
```

Төртіншіден, тізім генераторларына өте ұқсас сөздіктер генераторларының көмегімен.

```
>>> d = {a: a ** 2 for a in range(7)}
>>> d
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}
```

Сөздіктер әдістері

dict.clear () - сөздікті тазартады.

dict.copy () - сөздіктің көшірмесін қайтарады.

classmethod dict.fromkeys (seq [, мән]) - seq кілттері мен мәндер сөздігін жасайды (әдепкі бойынша).

dict.get (key [, default]) - кілт мәнін қайтарады, бірақ жоқ болса, ерекшелік бермейді, әдепкі мәндерді қайтарады (None әдепкі).

dict.items () - жұптарды (кілтті, мәнді) қайтарады.

dict.keys () - сөздікте кілттерді қайтарады.

dict.pop (key [, default]) - кілтті жояды және мәнді қайтарады. Егер кілт жоқ болса, әдепкі мәнді қайтарады (әдепкі ерекшелік тастайды).

dict.popitem () - жұпты жояды және қайтарады (кілт, мән). Егер сөздік пуст болса, KeyError-ді тастайды. Есіңізде болсын, сөздіктер реттелмеген.

dict.setdefault (кілт [, әдепкі]) - кілт мәнін қайтарады, бірақ мән таңдалмаған болса, әдепкі кілт жасалады (әдепкі).

dict.update ([басқа]) - басқа жұптарды (кілтті, мәнді) қосу арқылы сөздік жаңартады. Бар кілттер қайта жазылады. Кері жоқ (жаңа сөздік емес!).

dict.values () - сөздік мәнін қайтарады.

3.3 ConfigureCWD

```
import os

def setCurrentWorkingDirectory(directoryName):

    programDirectory = os.path.dirname(os.path.abspath(__file__))
    newCurrentWorkingDirectory = os.path.join(programDirectory, directoryName)
    os.chdir(newCurrentWorkingDirectory)
```

Бұл бағдарламаның функциясы-бағдарлама қалтасының қай жерде орналасқанына қатысты жұмыс каталогын орнату.

Os модулі пайдаланылады, ол файлдық жүйемен, қоршаумен жұмыс істеуге, процестерді басқаруға мүмкіндік береді.

Біздің жағдайда os әдісі арқылы.path біз ағымдағы қалтаға көшеміз.

3.4 StandardizeStats

```
from nba_api.stats.endpoints import leaguedashteamstats
import statistics
from getStats import getStatsForTeam
import time
from customHeaders import customHeaders


def basicOrAdvancedStatMean(startDate, endDate, stat,statType = 'Base', season='2018-19'):

    time.sleep(.2)

    allTeamsInfo = leaguedashteamstats.LeagueDashTeamStats(per_mode_detailed='Per100Possessions',
                                        measure_type_detailed_defense=statType,
                                        date_from_nullable=startDate,
                                        date_to_nullable=endDate,
                                        season=season,
                                        headers=customHeaders,
                                        timeout=120)
    allTeamsDict = allTeamsInfo.get_normalized_dict()
    allTeamsList = allTeamsDict['LeagueDashTeamStats']

    specificStatAllTeams = []
    for i in range(len(allTeamsList)):
        specificStatAllTeams.append(allTeamsList[i][stat])

    mean = statistics.mean(specificStatAllTeams)
    return mean



def basicOrAdvancedStatStandardDeviation(startDate, endDate, stat,statType = 'Base', season='2018-19'):

    time.sleep(.2)

    allTeamsInfo = leaguedashteamstats.LeagueDashTeamStats(per_mode_detailed='Per100Possessions',
                                        measure_type_detailed_defense=statType,
                                        date_from_nullable=startDate,
                                        date_to_nullable=endDate,
                                        season=season,
```

```
                                    headers=customHeaders,
                                    timeout=120
                                    )
    allTeamsDict = allTeamsInfo.get_normalized_dict()
    allTeamsList = allTeamsDict['LeagueDashTeamStats']

    specificStatAllTeams = []
    for i in range(len(allTeamsList)):
        specificStatAllTeams.append(allTeamsList[i][stat])

    standardDeviation = statistics.stdev(specificStatAllTeams)
    return standardDeviation


def basicOrAdvancedStatZScore(observedStat, mean, standardDeviation):

    zScore = (observedStat-mean)/standardDeviation

    return(zScore)
```

Бағдарламаның осы бөлімінде бірнеше кітапхана пайдаланылуда. Nba_api модулі бізге NBA матчтары бойынша әртүрлі статистиканы ала алатын сайтпен байланысуға мүмкіндік беру үшін қажет.

API (Application programming interface) - бұл бағдарлама ұсынатын келісімшарт. "Маған мына параметрлер бойынша хабарласуға болады" деген сияқты талаптар қойылады.

Келісім-шартт келесі пункттерді қамтиды:

- біз орындай алатын операцияның өзі;
- кіріске түсетін деректер;
- шығатын деректер.

Statistics модулі математикалық статистика үшін жауап береді. Бұл модуль сандық деректердің математикалық статистикасын есептеу үшін функцияларды ұсынады.

Time модулі-бұл уақытпен байланысты түрлі функцияларды ұсынатын модуль.

Time әдісі арқылы.sleep шақырушы ағынның орындалуын көрсетілген секундқа тоқтата аламыз. Дәлел ұйқының нақты уақытын көрсету үшін өзгермелі үтірмен сан болуы мүмкін. Кідірістің нақты уақыты сұралғаннан аз болуы мүмкін, өйткені кез келген кідіртілген сигнал sleep() режимі осы сигналды ұстап қалу процедурасын орындағаннан кейін. Сонымен қатар, кідірту уақыты жүйедегі басқа іс-қимылдарды жоспарлаудан ерікті шамаға сұралғаннан артық болуы мүмкін.

Біз қажетті модульдерді импорттайтын блокта python коды бар көрші файлдардан функцияларды алатын жолдар бар. Мысалы, getStats және customHeaders. Олардың мазмұны мен функцияларын біз төменде қарастырамыз.

3.5 CustomHeaders және getDailyMatchUps

```
customHeaders = {
    'Host': 'stats.nba.com',
    'Connection': 'keep-alive',
    'Cache-Control': 'max-age=0',
    'Upgrade-Insecure-Requests': '1',
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_5) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/75.0.3770.100 Safari/537.36',
    'Accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-
exchange;v=b3',
    'Accept-Encoding': 'gzip, deflate, br',
    'Accept-Language': 'en-US,en;q=0.9',
    'Referer': 'https://stats.nba.com/',
    'x-nba-stats-origin': 'stats',
    'x-nba-stats-token': 'true',
}
```

Сұраулар браузерден келіп сияқты көріну үшін пайдаланылады. Сондай-ақ сөздікке жазылады.

Негізгі функция - күнделікті NBA ойындарын табу.

```
from nba_api.stats.endpoints import leaguegamelog, scoreboard
from teamIds import teams
from customHeaders import customHeaders


def dailyMatchupsPast(date, season):

    dailyMatchups = leaguegamelog.LeagueGameLog(season=season, league_id='00', season_type_all_star='Regular
Season', date_from_nullable=date,date_to_nullable=date, headers=customHeaders,timeout=60)
    dailyMatchupsDict = dailyMatchups.get_normalized_dict()
    listOfTeams = dailyMatchupsDict['LeagueGameLog']

    winLossList = []
    homeAwayDict = {}
    for i in range(0,len(listOfTeams),2):
        if '@' in listOfTeams[i]['MATCHUP']:
            awayTeam = listOfTeams[i]['TEAM_NAME']
            homeTeam = listOfTeams[i+1]['TEAM_NAME']

            winLossList.append(listOfTeams[i+1]['WL'])

        else:
            awayTeam = listOfTeams[i+1]['TEAM_NAME']
            homeTeam = listOfTeams[i]['TEAM_NAME']

            winLossList.append(listOfTeams[i]['WL'])

        homeAwayDict.update({homeTeam:awayTeam})

    matchupsResultCombined = [homeAwayDict, winLossList]
    return(matchupsResultCombined)



def dailyMatchupsPresent(date):

    dailyMatchups = scoreboard.Scoreboard(league_id='00', game_date=date, headers=customHeaders, timeout=120)
```

```
    dailyMatchupsDict = dailyMatchups.get_normalized_dict()
    listOfGames = dailyMatchupsDict['GameHeader']

    homeAwayDict = {}

    for game in listOfGames:

        homeTeamID = game['HOME_TEAM_ID']

        for team, teamID in teams.items():
            if teamID == homeTeamID:
                homeTeamName = team

        awayTeamID = game['VISITOR_TEAM_ID']

        for team, teamID in teams.items():
            if teamID == awayTeamID:
                awayTeamName = team

        homeAwayDict.update({homeTeamName:awayTeamName})

    return homeAwayDict
```

Бұл кодта екі функцияны жазылған:

- dailyMatchupsPast;
- dailyMatchupsPresent.

Атаулардан белгілі болғандай, осы функцияларды қолдану арқылы біз көрсетілген күндерге арналған ойындар тізімін аламыз. Олардың тек біріншісінде: біріншісінде өткен ойындардың тізімдері, екіншісінде - қазіргі ойындар. Екі жағдайда да функция тізімді қайтарады, онда 0 индекс - сөздік, ал 1 индекс - ойын нәтижесі. Күнді mm / dd / yyyy форматында енгізу керек, ал мезгіл - yyyy / yy.

3.6 QetStats – статистиканы алу

НБА-да кез-келген команда үшін статистиканы топтастыру негізгі функция болып табылады.

```
        from teamIds import teams
from nba_api.stats.endpoints import teamdashboardbygeneralsplits, leaguedashteamstats
import time
from customHeaders import customHeaders


def getStatsForTeam(team, startDate, endDate, season='2019-20'):

    time.sleep(1)

    generalTeamInfo = teamdashboardbygeneralsplits.TeamDashboardByGeneralSplits(team_id=teams[team],
per_mode_detailed='Per100Possessions', date_from_nullable=startDate, date_to_nullable=endDate, season=season,
headers=customHeaders, timeout=120)
    generalTeamDict = generalTeamInfo.get_normalized_dict()
    generalTeamDashboard = generalTeamDict['OverallTeamDashboard'][0]
```

```
    winPercentage = generalTeamDashboard['W_PCT']
    rebounds = generalTeamDashboard['REB']
    turnovers = generalTeamDashboard['TOV']
    plusMinus = generalTeamDashboard['PLUS_MINUS']


    advancedTeamInfo = teamdashboardbygeneralsplits.TeamDashboardByGeneralSplits(team_id=teams[team],
measure_type_detailed_defense='Advanced', date_from_nullable=startDate, date_to_nullable=endDate,
season=season, headers=customHeaders, timeout=120)
    advancedTeamDict  = advancedTeamInfo.get_normalized_dict()
    advancedTeamDashboard = advancedTeamDict['OverallTeamDashboard'][0]


    offensiveRating = advancedTeamDashboard['OFF_RATING']
    defensiveRating = advancedTeamDashboard['DEF_RATING']
    trueShootingPercentage = advancedTeamDashboard['TS_PCT']


    allStats = {
        'W_PCT':winPercentage,
        'REB':rebounds,
        'TOV':turnovers,
        'PLUS_MINUS':plusMinus,
        'OFF_RATING':offensiveRating,
        'DEF_RATING': defensiveRating,
        'TS_PCT':trueShootingPercentage,
    }

    return allStats
```

GetStatsForTeam функциясы жазылған. Оның міндеті: енгізілген команда үшін әр түрлі статистиканы сөздікте қайтару. Барлығы дұрыс жұмыс істеуі үшін келесі шарттар орындалуы керек:

- команда TeamIds.py файлында топ атауына сәйкес келуі кере;
- startDate (басталу күні) және endDate (аяқталу күні) dd / mm / yyyy форматында жазылуы керек.

Осы кодты орындау нәтижесінде белгілі бір топтың статистикасы бар allStats сөздігін аламыз.

3.7 teamIds және createModel

Бұл кодта stats.nba.com ресурсында әр NBA командасының жеке ID жазылған сөздік бар.

```
        teams = {
    "Atlanta Hawks": 1610612737,
    "Boston Celtics":1610612738,
    "Brooklyn Nets": 1610612751,
    "Charlotte Hornets": 1610612766,
    "Charlotte Bobcats": 1610612766,
    "Chicago Bulls": 1610612741,
    "Cleveland Cavaliers": 1610612739,
    "Dallas Mavericks": 1610612742,
    "Denver Nuggets": 1610612743,
    "Detroit Pistons": 1610612765,
    "Golden State Warriors": 1610612744,
    "Houston Rockets": 1610612745,
    "Indiana Pacers": 1610612754,
    "LA Clippers": 1610612746,
    "Los Angeles Clippers": 1610612746,
    "Los Angeles Lakers": 1610612747,
    "Memphis Grizzlies": 1610612763,
    "Miami Heat": 1610612748,
    "Milwaukee Bucks": 1610612749,
    "Minnesota Timberwolves": 1610612750,
    "New Jersey Nets": 1610612751,
    "New Orleans Hornets": 1610612740,
    "New Orleans Pelicans": 1610612740,
    "New York Knicks": 1610612752,
    "Oklahoma City Thunder": 1610612760,
    "Orlando Magic": 1610612753,
    "Philadelphia 76ers": 1610612755,
    "Phoenix Suns": 1610612756,
    "Portland Trail Blazers": 1610612757,
    "Sacramento Kings": 1610612758,
    "San Antonio Spurs": 1610612759,
    "Toronto Raptors": 1610612761,
    "Utah Jazz": 1610612762,
    "Washington Wizards": 1610612764,
}
```

Осы кодта біз НБА матчтарының нәтижесін болжау үшін үлгіні жасаймыз, дайындаймыз және сынаймыз.

```
        from standardizeStats import basicOrAdvancedStatZScore, basicOrAdvancedStatStandardDeviation,
basicOrAdvancedStatMean
from getDailyMatchups import dailyMatchupsPast
from getStats import getStatsForTeam
from availableStats import availableStats
from configureCWD import setCurrentWorkingDirectory

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import pandas as pd
import pickle

from datetime import timedelta, date


def zScoreDifferential(observedStatHome, observedStatAway, mean, standardDeviation):

    homeTeamZScore = basicOrAdvancedStatZScore(observedStatHome, mean, standardDeviation)
```

```
    awayTeamZScore = basicOrAdvancedStatZScore(observedStatAway, mean, standardDeviation)

    differenceInZScore = homeTeamZScore - awayTeamZScore
    return differenceInZScore


def infoToDataFrame(dailyGames, meanDict, standardDeviationDict, startDate, endDate, season):

    fullDataFrame = []
    gameNumber = 0
    dailyResults = dailyGames[1]

    for homeTeam,awayTeam in dailyGames[0].items():

        homeTeamStats = getStatsForTeam(homeTeam, startDate, endDate, season)
        awayTeamStats = getStatsForTeam(awayTeam, startDate, endDate, season)

        currentGame = [homeTeam,awayTeam]

        for stat,statType in availableStats.items():
            zScoreDif = zScoreDifferential(homeTeamStats[stat], awayTeamStats[stat], meanDict[stat],
standardDeviationDict[stat])
            currentGame.append(zScoreDif)

        if dailyResults[gameNumber] == 'W':
            result = 1
        else:
            result = 0

        currentGame.append(result)
        gameNumber += 1

        print(currentGame)
        fullDataFrame.append(currentGame)

    return(fullDataFrame)


def daterange(startDate, endDate):

    for n in range(int ((endDate - startDate).days)):
        yield startDate + timedelta(n)


def createMeanStandardDeviationDicts(startDate, endDate, season):

    meanDict = {}
    standardDeviationDict = {}

    for stat, statType in availableStats.items():
        statMean = basicOrAdvancedStatMean(startDate, endDate, stat, statType, season)
        meanDict.update({stat: statMean})

        statStandardDeviation = basicOrAdvancedStatStandardDeviation(startDate, endDate, stat, statType, season)
        standardDeviationDict.update({stat: statStandardDeviation})

    bothDicts = []
    bothDicts.append(meanDict)
    bothDicts.append(standardDeviationDict)

    return bothDicts
```

```
def getTrainingSet(startYear, startMonth, startDay, endYear, endMonth, endDay, season, startOfSeason):

    startDate = date(startYear, startMonth, startDay)
    endDate = date(endYear, endMonth, endDay)

    startDateFormatted = startDate.strftime("%m/%d/%Y")
    allGames = []

    for singleDate in daterange(startDate, endDate):
        currentDate = singleDate.strftime("%m/%d/%Y")
        print(currentDate)

        previousDay = singleDate - timedelta(days=1)
        previousDayFormatted = previousDay.strftime("%m/%d/%Y")

        meanAndStandardDeviationDicts = createMeanStandardDeviationDicts(startOfSeason, previousDayFormatted,
season)
        meanDict = meanAndStandardDeviationDicts[0]
        standardDeviationDict = meanAndStandardDeviationDicts[1]

        currentDayGames = dailyMatchupsPast(currentDate, season)
        currentDayGamesAndStatsList = infoToDataFrame(currentDayGames, meanDict, standardDeviationDict,
startOfSeason, previousDayFormatted, season)  # Formats Z Score difs for games on current date in loop

        for game in currentDayGamesAndStatsList:
            game.append(currentDate)
            allGames.append(game)

    print(allGames)
    return(allGames)


def createDataFrame(listOfGames):

    games = pd.DataFrame(
        listOfGames,
        columns=['Home', 'Away', 'W_PCT', 'REB', 'TOV', 'PLUS_MINUS', 'OFF_RATING', 'DEF_RATING',
'TS_PCT', 'Result', 'Date']
    )

    print(games)
    return(games)


def performLogReg(dataframe):

    featureColumns = ['W_PCT', 'REB', 'TOV', 'PLUS_MINUS', 'OFF_RATING', 'DEF_RATING', 'TS_PCT']

    X = dataframe[featureColumns]
    Y = dataframe.Result

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, shuffle=True)
    logreg = LogisticRegression()

    logreg.fit(X_train, Y_train)

    Y_pred = logreg.predict(X_test)

    confusionMatrix = metrics.confusion_matrix(Y_test, Y_pred)

    print('Coefficient Information:')
```

```
    for i in range(len(featureColumns)):

        logregCoefficients = logreg.coef_

        currentFeature = featureColumns[i]
        currentCoefficient = logregCoefficients[0][i]

        print(currentFeature + ': ' + str(currentCoefficient))

    print('---------------------------------')

    print("Accuracy:", metrics.accuracy_score(Y_test, Y_pred))
    print("Precision:", metrics.precision_score(Y_test, Y_pred))
    print("Recall:", metrics.recall_score(Y_test, Y_pred))

    print('---------------------------------')

    print('Confusion Matrix:')
    print(confusionMatrix)

    return logreg


def saveModel(model, filename):

    setCurrentWorkingDirectory('SavedModels')

    with open(filename, 'wb') as file:
        pickle.dump(model, file)


def createModel(startYear=None, startMonth=None, startDay=None, endYear=None, endMonth=None,
endDay=None, season='2018-19', startOfSeason = '10/16/2018', filename='model.pkl'):



    setCurrentWorkingDirectory('Data')
    allGamesDataframe = pd.read_csv('COMBINEDgamesWithInfo2016-19.csv')

    logRegModel = performLogReg(allGamesDataframe)

    saveModel(logRegModel, filename)
```

Бұл кодта біз бұрын пайдаланылмаған питон кітапханаларын импорттаймыз, олар:

- Sklearn;
- pandas;
- pickle.

Scikit-learn - бұл Python үшін машиналарды тегін оқытатын кітапхана. Ол векторлық машина, кездейсоқ ормандар және k-көршілер сияқты әртүрлі алгоритмдерге ие, сонымен қатар NumPy және SciPy сияқты Python сандық және ғылыми кітапханаларын қолдайды.

Pandas - бұл деректерді талдауға арналған жоғары деңгейдегі Python кітапханасы. Неліктен мен оны жоғары деңгей деп атаймын, себебі ол төменгі

деңгейдегі NumPy кітапханасының үстіне салынған (C түрінде жазылған), бұл үлкен плюс. Python экожүйесінде pandas - деректерді өңдеуге және талдауға арналған ең дамыған және жылдам дамитын кітапхана. Мен өз жұмысымда оны күнделікті дерлік пайдалануым керек, сондықтан болашақта бірдеңені ұмытып қалсам, оны қысқаша жазуға тырысамын.

Pickle модулі Python нысандарының құрылымын сериализациялау және қайта реализациялау үшін қолданылады. Python кез келген нысан дискіде сақтау үшін таңдалуы мүмкін. Бұл pickle жасайды, сондықтан ол алдымен "сериализовать" файлды жазу алдында объект. Pickling-бұл Python нысанын (тізім, dict және т.б.) таңбалар ағынына түрлендіру тәсілі. Бұл идея басқа Python сценарийінде нысанды қалпына келтіру үшін қажетті барлық ақпаратты қамтиды.

Кодта келесі функциялар жазылған:

- zScoreDifferential;
- infoToDataFrame;
- daterange;
- createMeanStandardDeviationDicts;
- getTrainingSet;
- createDataFrame;
- performLogReg;
- savemodel;
- createMode.l

zScoreDifferential көрсетілген статистика үшін екі команда арасындағы «zScore дифференциалын» есептейді.

infoToDataFrame panda датафрейміна орналастырылатын барлық деректерді біріктіру және пішімдеу үшін қолданылады.

Daterange - берілген басталу күнінен бастап (startDate) аяқталу күніне дейін (endDate) қайталануға мүмкіндік беретін функция.

createMeanStandardDeviationDicts функциясы 0 индексі әрбір көрсеткіштің орташа мәні болып табылатын тізімді қайтару болып табылады. 1-индекс - бұл әр көрсеткіштің стандартты ауытқуын қамтитын сөздік

getTrainingSet (сөзбе-сөз - жаттығулар жиынтығын алыңыз) - басталу мен аяқталу арасындағы әр күн ішінде қайталанады және әр ойынды қайтару үшін бір тізімге қосады.

createDataFrame «zScore дифференциалдары» бар ойындар тізімінен датафрейм қайтарады.

PerformLogReg функциясы логистикалық регрессия моделін жасайды және модельдің нақтылығын тексереді.

Savemodel модельді болашақта пайдалану үшін бумаға сақтайды

create Model логистикалық регрессияның жаңа үлгілерін генерациялау үшін қолданылады. CSV файлынан әрбір ойын үшін статистика мен болжамдарды импорттай алады немесе дербес жасалуы мүмкін.

3.8 nbaPredict

Бағдарламаның бұл бөлігі аталған күнгі NBA ойындарының нәтижелерін болжайды. Көбінесе болжам үшін ол іске қосылады. Екі Python кітапханалары қолданылады, біз импорттайтын басқа функциялар.

```
        import pickle
import pandas as pd

from getDailyMatchups import dailyMatchupsPresent
from createModel import createMeanStandardDeviationDicts, zScoreDifferential
from availableStats import availableStats
from getStats import getStatsForTeam
from configureCWD import setCurrentWorkingDirectory



def dailyGamesDataFrame(dailyGames, meanDict, standardDeviationDict, startDate, endDate, season):

    fullDataFrame = []

    for homeTeam,awayTeam in dailyGames.items():

        homeTeamStats = getStatsForTeam(homeTeam, startDate, endDate, season)
        awayTeamStats = getStatsForTeam(awayTeam, startDate, endDate, season)

        currentGame = [homeTeam,awayTeam]

        for stat,statType in availableStats.items():
            zScoreDif = zScoreDifferential(homeTeamStats[stat], awayTeamStats[stat], meanDict[stat],
standardDeviationDict[stat])
            currentGame.append(zScoreDif)

        fullDataFrame.append(currentGame)

    return(fullDataFrame)


def predictDailyGames(currentDate, season, startOfSeason):

    dailyGames = dailyMatchupsPresent(currentDate)
    meanDict, standardDeviationDict = createMeanStandardDeviationDicts(startOfSeason, currentDate, season)
    dailyGamesList = dailyGamesDataFrame(dailyGames, meanDict, standardDeviationDict, startOfSeason,
currentDate, season)


    gamesWithZScoreDifs = pd.DataFrame(
        dailyGamesList,
        columns=['Home', 'Away', 'W_PCT', 'REB', 'TOV', 'PLUS_MINUS', 'OFF_RATING', 'DEF_RATING',
'TS_PCT']
    )

    justZScoreDifs = gamesWithZScoreDifs.loc[:,'W_PCT':'TS_PCT']

    with open('finalized_model.pkl', 'rb') as file:
        pickleModel = pickle.load(file)

    predictions = pickleModel.predict_proba(justZScoreDifs)

    gamesWithPredictions = [dailyGames, predictions]
    return gamesWithPredictions
```

```
def interpretPredictions(gamesWithPredictions):

    dailyGames = gamesWithPredictions[0]
    probabilityPredictions = gamesWithPredictions[1]

    for gameNum in range(len(probabilityPredictions)):
        winProb = probabilityPredictions[gameNum][1]
        winProbRounded = round(winProb,4)
        winProbPercent = "{:.2%}".format(winProbRounded)

        homeTeam = list(dailyGames.keys())[gameNum]
        awayTeam = list(dailyGames.values())[gameNum]

        print('There is a ' + winProbPercent + ' chance that the ' + homeTeam + ' will defeat the ' + awayTeam + '.')


def makeInterpretPredictions(currentDate, season, startOfSeason):

    setCurrentWorkingDirectory('SavedModels')

    print('Predictions for ' + currentDate + ':')
    predictions = predictDailyGames(currentDate, season, startOfSeason)
    interpretPredictions(predictions)


makeInterpretPredictions('12/12/2019', '2019-20', '10/22/2019')
```

Келесі функциялар жазылған:

- dailyGamesDataFrame;
- predictDailyGames;
- interpretPredictions;
- makeinterpretPredictions.

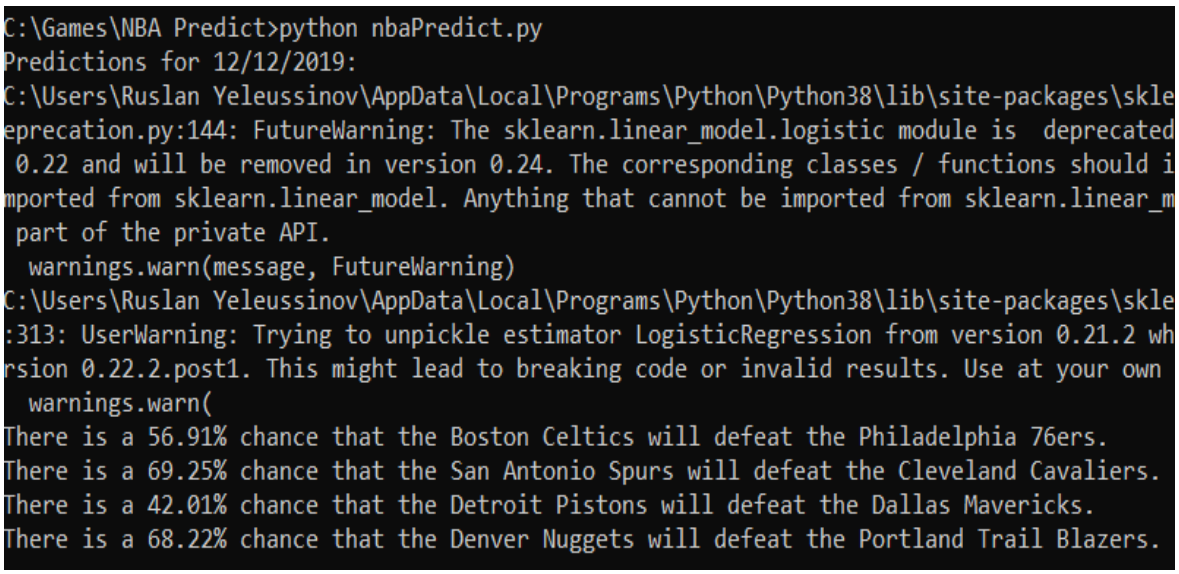
daily Games Data Frame-Z Score дифференциалды ойындар тізімін pandas датафрейміне орналастыратын командалар арасында қайтарады.

predictDailyGames 0 – dailyGames индексі {Home : Away} пішіміндегі сөздік түрінде тізімді қайтарады. 1 индексі-әрбір ойын үшін Болжамдар тізімі

interpretPredictions әр ойында қонақтар командасын жеңуге мүмкіндік береді.

makeinterpretPredictions берілген күнге ойын шығарып, әрбір ойын үшін болжамдарды қайтарады.

Осы кодты орындау нәтижесінде біз келесі қорытынды аламыз:



3.1 сурет – nbaPredict программасының нәтижесі

3.9 MakePastPredictions

NBA өткен ойындарының белгілі бір ауқымын болжау үшін пайдаланылады. NbaPredict файлына өте ұқсас, айырмашылық тек нәтижесінде біз нақты күнге емес, өткен уақыт аралығы үшін болжам нәтижелерін аламыз.

```
        import pickle
import pandas as pd

from createModel import getTrainingSet, createDataFrame
from configureCWD import setCurrentWorkingDirectory



def getTrainingSetCSV(startYear, startMonth, startDay, endYear, endMonth, endDay, season, startDateOfSeason,
filename='gamesWithInfo.csv'):


    rangeOfGames = getTrainingSet(startYear, startMonth, startDay, endYear, endMonth, endDay, season,
startDateOfSeason)
    rangeOfGamesDataframe = createDataFrame(rangeOfGames)

    setCurrentWorkingDirectory('Data')

    rangeOfGamesDataframe.to_csv(filename)



def getPredictionsCSV(gameDataFilename, outputFilename):

    setCurrentWorkingDirectory('Data')

    gamesWithZScoreDifs = pd.read_csv(gameDataFilename)

    withoutNums = gamesWithZScoreDifs.loc[:, 'Home':'Date']
    justZScoreDifs = gamesWithZScoreDifs.loc[:, 'W_PCT':'TS_PCT']
```

```
    setCurrentWorkingDirectory('SavedModels')
    with open('finalized_model.pkl', 'rb') as file:
        pickleModel = pickle.load(file)

    predictions = pickleModel.predict(justZScoreDifs)
    probPredictions = pickleModel.predict_proba(justZScoreDifs)

    numCorrect = 0
    numWrong = 0
    allGames = []

    for i in range(len(probPredictions)):

        winProbability = probPredictions[i][1]
        homeTeam = withoutNums.iloc[i, 0]
        awayTeam = withoutNums.iloc[i, 1]
        date = withoutNums.iloc[i, 10]

        currentGameWithPred = [date, homeTeam, awayTeam, winProbability]

        allGames.append(currentGameWithPred)


        predictionsDF = pd.DataFrame(
            allGames,
            columns=['Date', 'Home', 'Away', 'Home Team Win Probability']
        )

        setCurrentWorkingDirectory('Data')
        predictionsDF.to_csv(outputFilename)

        value = withoutNums.iloc[i,9]
        if value == predictions[i]:
            numCorrect += 1
        else :
            numWrong += 1

    print('Accuracy:')
    print((numCorrect)/(numCorrect+numWrong))



def makePastPredictions(startYear, startMonth, startDay, endYear, endMonth, endDay, season, startDateOfSeason,
                        gameDataFilename='gamesWithInfo.csv', outputFilename='predictions.csv'):


    getTrainingSetCSV(startYear, startMonth, startDay, endYear, endMonth, endDay, season, startDateOfSeason,
                      gameDataFilename)

    getPredictionsCSV(gameDataFilename, outputFilename)



makePastPredictions(2018, 12, 28, 2019, 1, 1, '2018-19', '10/16/2018',
                    'gamesWithInfo.csv', 'predictions.csv')
```

Келесі функциялар жазылған:

- getTrainingSetCSV;
- getPredictionsCSV;
- makePastPredictions.

getTrainingSetCSV көрсетілген уақыт кезеңі арасындағы барлық ойындар үшін ақпаратты жоба шеңберіндегі "деректер" қалтасына экспорттайды.

getPredictionsCSV бірнеше ойындар үшін болжам беретін CSV файлын жасайды. Берілген уақыт аралығында ойындарды болжауда модельдің дәлдігін көрсету.

makePastPredictions аталған уақыт ауқымында ойын болжамды болжамдарын жасайды, ойын туралы ақпаратпен CSV оларды экспорттайды.

Орындау нәтижесінде біз келесі нәтиже аламыз;

	Home	Away	W_PCT	...	TS_PCT	Result	Date
0	Los Angeles Lakers	LA Clippers	-0.121520	...	-0.878637	0	12/28/2018
1	Phoenix Suns	Oklahoma City Thunder	-2.709190	...	0.439318	0	12/28/2018
2	Orlando Magic	Toronto Raptors	-2.130181	...	-2.573151	1	12/28/2018
3	Miami Heat	Cleveland Cavaliers	1.829954	...	0.125520	1	12/28/2018
4	Indiana Pacers	Detroit Pistons	1.122277	...	1.820034	1	12/28/2018
5	Denver Nuggets	San Antonio Spurs	0.807753	...	-1.066916	1	12/28/2018
6	Minnesota Timberwolves	Atlanta Hawks	1.415355	...	0.188279	0	12/28/2018
7	Charlotte Hornets	Brooklyn Nets	0.092927	...	0.000000	1	12/28/2018
8	Washington Wizards	Chicago Bulls	0.814901	...	1.882793	0	12/28/2018
9	New Orleans Pelicans	Dallas Mavericks	-0.400302	...	0.313799	1	12/28/2018
10	Phoenix Suns	Denver Nuggets	-3.006606	...	-0.320140	0	12/29/2018
11	LA Clippers	San Antonio Spurs	0.519126	...	0.768337	0	12/29/2018
12	Portland Trail Blazers	Golden State Warriors	-0.490286	...	-1.920843	0	12/29/2018
13	Utah Jazz	New York Knicks	1.600639	...	1.920843	1	12/29/2018
14	Atlanta Hawks	Cleveland Cavaliers	0.519126	...	1.088478	1	12/29/2018
15	Milwaukee Bucks	Brooklyn Nets	1.780891	...	1.408618	1	12/29/2018
16	Washington Wizards	Charlotte Hornets	-1.002202	...	0.192084	1	12/29/2018
17	Memphis Grizzlies	Boston Celtics	-0.425395	...	-0.384169	0	12/29/2018
18	New Orleans Pelicans	Houston Rockets	-0.829160	...	-0.192084	0	12/29/2018
19	Los Angeles Lakers	Sacramento Kings	0.093002	...	0.190086	1	12/30/2018
20	Orlando Magic	Detroit Pistons	-0.314778	...	0.506896	1	12/30/2018
21	Miami Heat	Minnesota Timberwolves	0.307624	...	-0.887069	0	12/30/2018
22	Portland Trail Blazers	Philadelphia 76ers	-0.593785	...	-1.267241	1	12/30/2018
23	Dallas Mavericks	Oklahoma City Thunder	-1.259111	...	1.457327	1	12/30/2018
24	Toronto Raptors	Chicago Bulls	3.040466	...	2.597844	1	12/30/2018
25	San Antonio Spurs	Boston Celtics	-0.423240	...	0.577096	1	12/31/2018
26	Phoenix Suns	Golden State Warriors	-2.912463	...	-2.436629	0	12/31/2018
27	Houston Rockets	Memphis Grizzlies	0.408893	...	1.090071	1	12/31/2018
28	Indiana Pacers	Atlanta Hawks	2.532265	...	1.090071	1	12/31/2018
29	Charlotte Hornets	Orlando Magic	0.208033	...	1.282437	1	12/31/2018
30	Oklahoma City Thunder	Dallas Mavericks	1.025818	...	-1.538924	1	12/31/2018
31	New Orleans Pelicans	Minnesota Timberwolves	-0.286942	...	1.154193	1	12/31/2018

3.2 сурет – makePastPredictions программасының нәтижесі

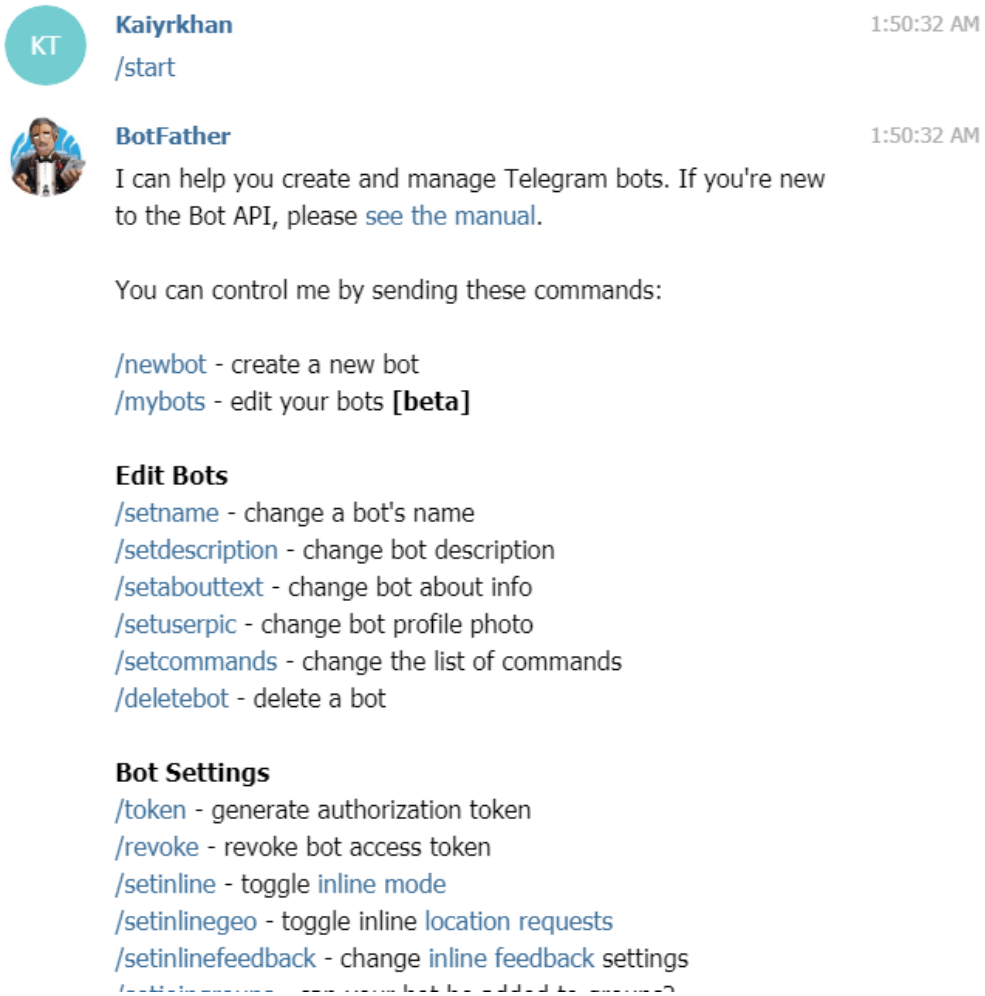
Мұнда біз алты негізгі бағана, үй командасы, команда қонақта, ойын нәтижесі мен күні бар.

Сондай-ақ, Data қалтасында predictions файлы жасалады.csv, төрт баған бар: күн, үй командасы, қонақ командасы және үй командасының Жеңіс ықтималдығы.

3.10 Телеграмға арналған бот жасау

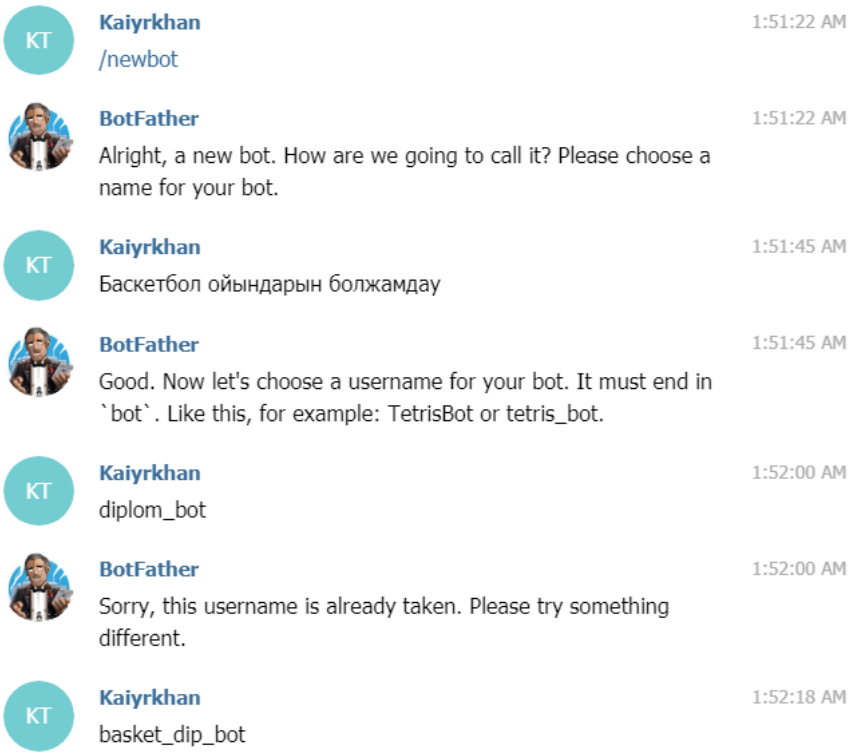
Бастау үшін біз Telegram-да біздің болашақ ботты тіркеу қажет. Бұл келесідей жасалады:

- алдымен telegram қосымшасын телефонға немесе компьютерге орнату керек;
- бізге жаңа ботты тіркеу үшін botfather ботын тауып алу керек;
- start батырмасын басып, ботпен чатты бастаймыз. ары қарай бот бізге төмендегідей командалар жиынтығын көрсетеді.



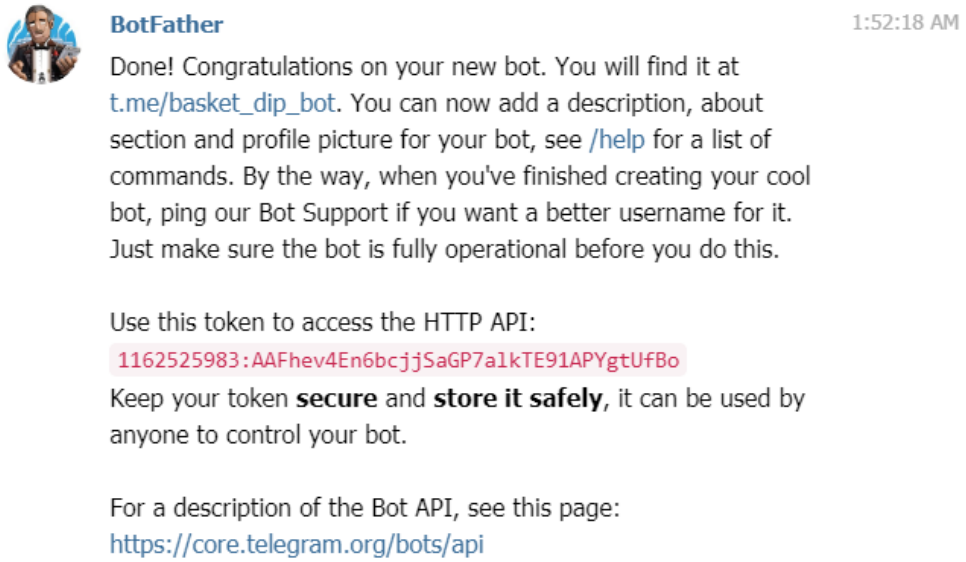
3.3 сурет – Телеграм бот жасау барысы

- жаңа ботты жасау үшін /newbot командасын орындау қажет. ары қарай инструкция бойынша жүреміз;



3.4 сурет – Телеграм ботқа ат тағайындау

- ботқа дұрыс ат бергеннен кейін келесідей нәтиже көреміз;



3.5 сурет – Телеграм ботқа арналған токен алу

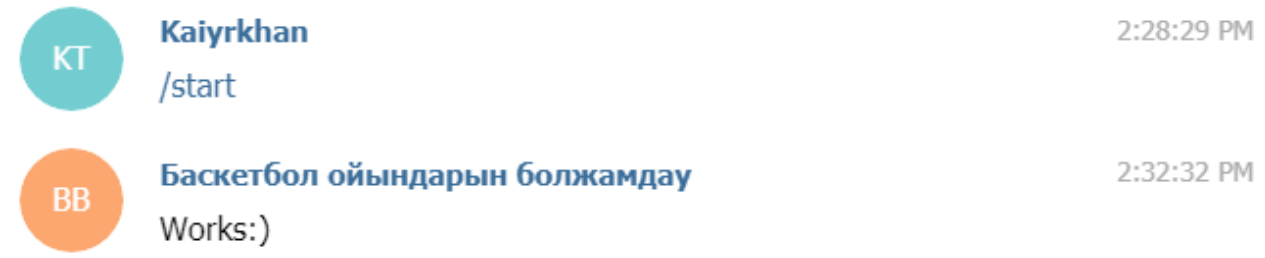
- бұл нәтижеден бізге token параметрі керек, ол бойынша біз ботты басқара аламыз және telegramapi-мен жұмыс жасауға мүмкіндік аламыз.

Негізгі программа сақталған папкада bot.py жаңа файлын жасаймыз. Файлда келесідей кодты орындаймыз:

```
        import logging
import telegram
import telebot
from nbaPredict import makeInterpretPredictions

class Request(object):
 def __init__(self):
  self.date = None
  self.season = None
  self.start_season = None

STATUS = NON_ASKING, DATE, SEASON, START_SEASON = range(4)
TOKEN = "1162525983:AAFhev4En6bcjjSaGP7alkTE91APYgtUfBo"
ASK_DATE = "Please enter the date in the following format: mm/dd/yyyy"
ASK_SEASON = "Please enter the season in the following format: yyyy-yy"
ASK_START_SEASON = "Please enter the start date of season in the following format: mm/dd/yyyy"
ASK_IN_PROCESS = "Please wait 5-6 minutes, data are taken and program will reply as it will finish"

request_params = Request()
cur_status = NON_ASKING
bot = telebot.TeleBot(TOKEN)

@bot.message_handler(commands=['start'])
def start_message(message):
    bot.send_message(message.chat.id, 'Works:)')

@bot.message_handler(commands=['predict'])
def start_message(message):
 global cur_status
 cur_status = DATE
 bot.send_message(message.chat.id, ASK_DATE)

def send_some_message(chat_id, text):
 global bot
 bot.send_message(chat_id, text)

def calc_and_send_result(chat_id, request_params):
 results = makeInterpretPredictions(request_params.date, request_params.season, request_params.start_season)
 print (results)
 send_some_message(chat_id, "Results are following: \n" + results)


@bot.message_handler(content_types=['text'])
def start_message(message):
 global cur_status
 print ("OK")
 if cur_status == NON_ASKING: return
 elif (cur_status == DATE):
  request_params.date = message.text
  cur_status = SEASON
  send_some_message(message.chat.id, ASK_SEASON)
 elif (cur_status == SEASON):
  request_params.season = message.text
  cur_status = START_SEASON
  send_some_message(message.chat.id, ASK_START_SEASON)
 elif (cur_status == START_SEASON):
  request_params.start_season = message.text
```

```
    cur_status = NON_ASKING
    print (request_params)
    send_some_message(message.chat.id, ASK_IN_PROCESS)
    calc_and_send_result(message.chat.id, request_params)

bot.polling()
```

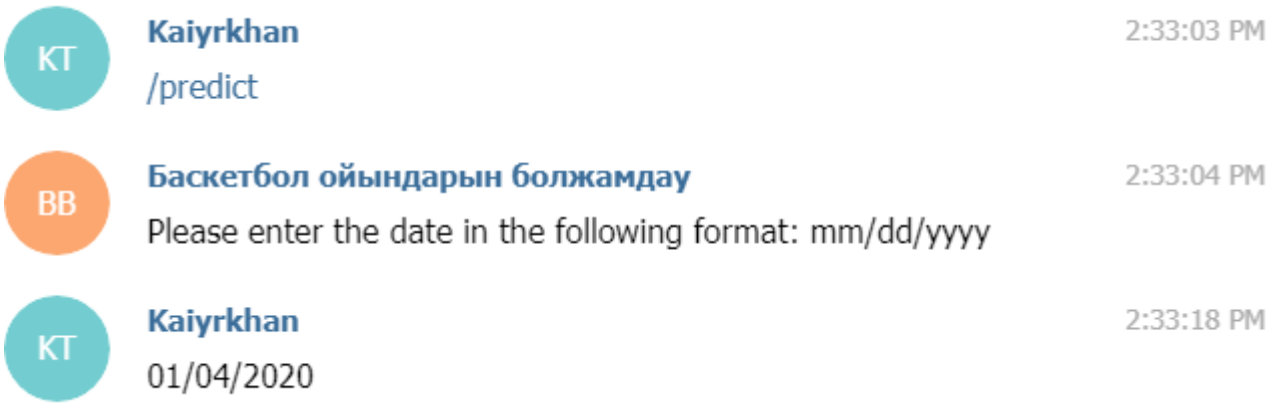
Программаны орындауға жіберіп, телеграм қосышасына кіреміз. Start батырмасын басқанда келесі нәтиже көреміз.



3.6 сурет – start командасының нәтижесі

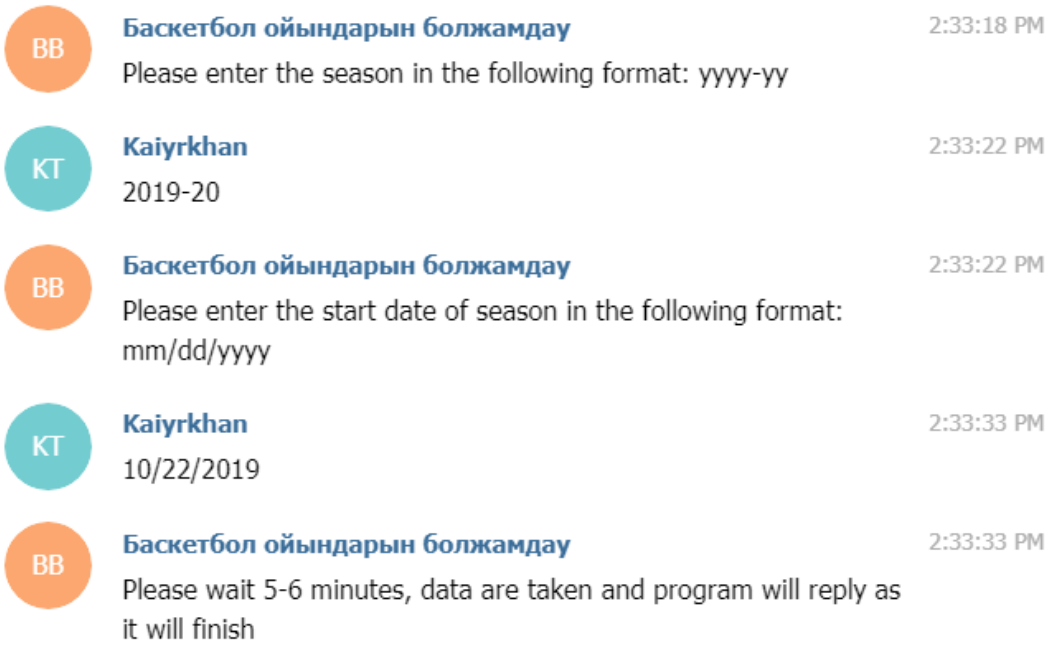
Бот бізге жұмыс істеп тұрмын деген жауап қайтарады.

/predict командасының жауабы:



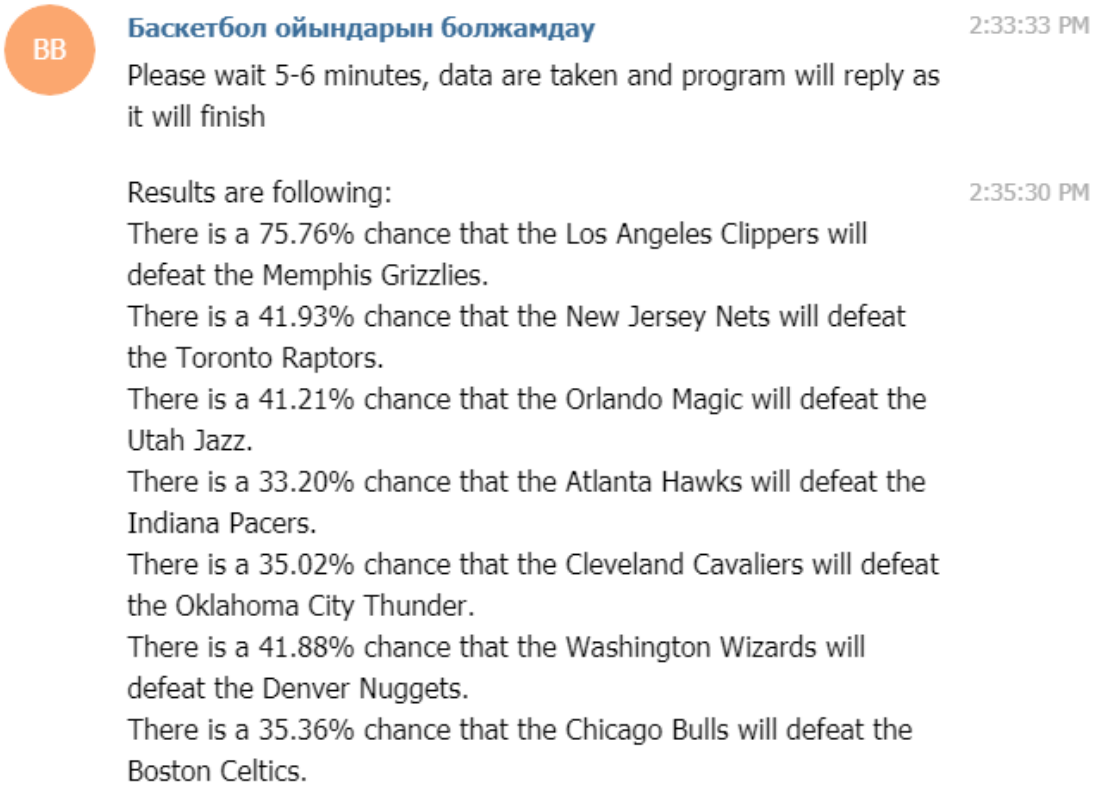
3.7 сурет – Ойын күнін енгізу

Бот бізге нақты ойынның күнін анықтау үшін келксі сұрақтар қояды:



3.8 сурет – Сезон мен сезонның басталу мерзімін енгізу

Керекті мәліметтерді жинап алғаннан кейін бот 5-6 минут ішінде болжамды экранға шығарады:



3.9 сурет – бот жұмысының нәтижесі

4 Техникалық-экономикалық көрсеткіштер

4.1 Бағдарламаның өзіндік құнын есептеу

Бұл бөлімде «Жасанды нейрон желілерін пайдалана отырып, спорттық жарыстардың нәтижелерін болжауға арналған ақпараттық жүйені құру» әзірленген бағдарламалық өнімнің техникалық-экономикалық көрсеткіштері есептелген. Бұл жоба негізінен нейрондық желілердің қолдану мүмкіндіктерін көрсету үшін арналған. Осыған ұқсас программалық қамталарды алып компаниялар алғаншында эксперимент ретінде көрсеткен, алайда қазіргі таңда көптеген букмекерлік кеңселерде пайдаланылады.

Жобаның экономикалық тиімділігін есептеу үшін оның өзіндік құнын есептеп, бағасын, маржасын есептеп, таза пайданы анықтау қажет.

Өндірістің өзіндік құны – тауар өндіруге жұмсалған шығын. Өнімнің өзіндік құнын келесі экономикалық элементтерді қосу арқылы аламыз:

- негізгі материалдар, сатып алынатын өнімдер (m);
- технологиялық мақсаттар үшін электр энергиясы (эл.);
- еңбекақы төлеу (z);
- әлеуметтік қажеттіліктерге аударымдар (соц.);
- амортизациялық аударымдар (а.);
- басқа шығындар (пр.).

«Материалдық шығындар» сәтінде материалдық ресурстардың құны көрсетілген, олардың жиынтығы олар сатып алынған бағаларға негізделеді және ешқандай қосымша төлемдерсіз, үстеме ақыларсыз, әртүрлі үшінші тараптардың қызметтерінің құны (кеден, қойма, көлік ұйымдары және т.б.) көрсетіледі. Сіз материалдық ресурстарға қайтарылатын қалдықтардың құнын алып тастай аласыз. Қайта өңделетін қалдық деп шикізат, материалдар, жартылай фабрикаттар және басқа да материалдық құндылықтардың қалдықтары оны өндіруге толық немесе ішінара пайдалануға болады. Мұндай қайта өңделетін қалдықтар әлеуетті пайдалану деңгейіне қарай бағаланады.

Технологиялық мақсаттар үшін электр энергиясына шығын бұл өнім өндіру процессінде кеткен электр энергия құны.

«Еңбекке ақы төлеу» элементі еңбек шығындарын, оның ішінде өндірістік персоналдың өндірістік шығындарын, өндіріс нәтижелерін, ынталандыру мен өтемақыны, сондай-ақ олардың негізгі қызметімен айналыспайтын жұмысшылардың еңбек ақыларын қамтиды.

«Әлеуметтік қажеттіліктерді алып тастау» элементтері әлеуметтік сақтандыру мекемелері, Зейнетақы қорлары және міндетті дәрі-дәрмек туралы заңнамаға сәйкес белгіленген барлық салықтарға шегеріледі. Олар жұмысшылардың жалақысына және еңбек ақыға (қызметтерге) қосылады («еңбек шығындары» үшін). Сақтандыру сыйлықақылары жалақының белгілі бір түрлері үшін төленбейді (мұндай төлемдер нормативтік құжатқа енгізілген).

Негізгі құралдардың амортизациясы – бұл құралдардың тозу дәрежесін көрсетеді.

Басқа шығындарға жиынтықтағы салықтар, төлемдер, сақтандыру қорларына салымдар, несие төлемдері, жол жүру, оқу, байланыс қызметтері, банктер, ақпараттық қызметтер және т.б. кіреді.

Бағдарламалық өнімді әзірлеудің еңбек салымы

Еңбек салымын анықтау үшін тапсырманы бірнеше кезеңге бөлу, осы кезеңде жүргізілетін жұмыстарды анықтау және жүктемені анықтау қажет, осылайша ПҚ әзірлеудің еңбек салымы алынады [8].

4.1 кесте – жұмыс кезеңдері және олардың күрделілігі келтірілген.

№	ПҚ-ны әзірлеу кезеңі	Кезеңде жасалатын жұмыс	Еңбек салымы	
			Адам х сағ.	Сағ. х күн
1	Талаптарды талдау	Пәндік саланы талдау, мақсаттар мен міндеттерді белгілеу.	1 х 16	8 х 2
2	Нарықты талдау	Букмекерлік кеңселердің осындай шешімдерді пайдалану тәжірибесін талдау, функционалды егжей-тегжейлі қарау, артықшылықтар мен кемшіліктерді анықтау. Бағдарламалық өнімді әзірлеу қажеттілігін анықтау және бәсекелестік шешімдер аясында ерекшеліктерді анықтау.	1 х 16	8 х 2
3	Жобалау	Техникалық тапсырманы әзірлеу. Әзірлеу үшін қаражат жинағын анықтау.	1 х 40	8 х 5
4	Жүзеге асыру	Нейрондық желіні әзірлеу және жаттықтыру	1 х 120	8 х 15
5	Тестілеу	Нейрожеліні болжау дәлдігін зерттеу, ең жақсы нәтижелермен салыстыру	1 х 24	8 х 3
6	Енгізу және қолдау	Бағдарламалық өнімді пайдалануға енгізу, персоналды өніммен таныстыру және оны сүйемелдеу.	1 х 16	8 х 2

Технологиялық мақсаттар үшін электр энергиясына кеткен шығынды есептеу

Технологиялық мақсаттар үшін электр энергиясының шығыны мына формула бойынша анықталады [7]:

$$Эл. = К*Т*Ц, \quad (4.1)$$

Мұнда К – сағатына компьютер қуат тұтынуы, кВт;
Т – қамтаманың құрылу уақыты, сағ.;
Ц - кВт/сағ құны, тг.

4.2 кесте – Технологиялық мақсаттар үшін электр энергиясының шығындары

Компьютердің қуатын тұпривтыну, кВт / сағ	1 кВт құны, тг.	Бағдарламаны әзірлеу уақыты, сағ.	Бағдарламаны әзірлеу уақыты, күн	Электр энергиясының жалпы шығыны, тг.
0,135	19,17	232	29	600,4

Бағдарламалық өнімді әзірлеуге кеткен шығындарды есептеу

Жобалық шешімді әзірлеуге толық шығындарды есептеу мынадай формула бойынша жүзеге асырылады:

$$C_{ni} = З_{тр} + З_{сзi} + М_i + Р_{сi} + П_{зi} + Р_{нi}, \quad (4.2)$$

Мұнда, $З_{тр}$ – еңбекақыға кеткен шығындар, теңге;
$З_{сзi}$ – әлеуметтік салық бойынша аударымдар, теңге;
$М_i$ – материалдарға кеткен шығындар, теңге;
$Р_{сi}$ – әзірлеу үшін қажетті арнайы бағдарламалық құралдарға кеткен шығындар, теңге;
$П_{зi}$ – басқа шығындар, теңге;
$Р_{нi}$ – үстеме шығыстар, теңге.

Жұмысшының еңбекақы мөлшері келесі формула бойынша есептеледі:

$$З_{фот} = \sum_{i=1}^{n} ЧС_i \times T_i, \quad (4.3)$$

Мұнда T_i – ПҚ-ны әзірлеуге жұмсалған уақыт;
$ЧС_i$ – сағаттық мөлшерлеме, теңге/сағ.

4.3 кесте – еңбек ақы төлеу шығындары

Орындаушы	Жұмсалатын уақыт, адам х сағ.	сағаттық мөлшерлеме, теңге/сағ	Сомасы, теңге
Әзірлеуші - бағдарламашы	1 х 232	700	162400
Еңбекақының жалпы құны			162400

Әлеуметтік салық қызметкердің еңбекақысынан 9,5% құрайды және мына формула бойынша есептеледі:

$$З_{сзі} = (З_{тр} - ПО - ВОСМС) \times 9{,}5\% - З_{соі}, \quad (4.4)$$

Мұнда, ПО – еңбекақының 10% - ын құрайтын зейнетақы аударымдары.

$$ПО = З_{тр} \times 10\%, \quad (4.5)$$

Әлеуметтік аударымдар әзірлеушінің табысының 3,5% құрайды, мынадай формула бойынша есептеледі:

$$З_{соі} = (З_{тр} - ПО) \times 3{,}5\%, \quad (4.6)$$

Міндетті әлеуметтік медициналық сақтандыру бойынша жарналарға аударымдар (ӘМСЖА) қызметкердің еңбекақысының 2% - ын құрайды және мынадай формула бойынша есептеледі:

$$ӘМСЖА = З_{тр} \times 2\%, \quad (4.7)$$

Сонымен,

$$ПО = 162400 \times 10\% = 16240 \text{ тенге};$$
$$ӘМСЖА = 162400 \times 2\% = 3248 \text{ тенге};$$
$$З_{соі} = (162400 - 16240) \times 3{,}5\% = 5116 \text{ тенге};$$
$$З_{сзі} = (162400 - 16240 - 1624) \times 9{,}5\% - 5116 = 8616 \text{ тенге}.$$

Барлық салықтың сомасы жалақының 10,46% құрайды. Осы жерден:

$$З_{сзі} = 162400 \times 10{,}46\% = 16987 \text{ тенге}$$

Материалдарға кеткен шығындар келесі формуламен анықталады:

$$M_i = \frac{З_{тр} \times Н_{мз}}{100\%}, \quad (4.8)$$

Мұндағы $Н_{мз}$ - еңбекақыдан түскен материалдар шығысының нормасы (3-5%).

Сондықтан материалдарға кеткен шығындар келеседей болады,

$$M_i = \frac{162400 \times 5\%}{100\%} = 8120 \text{ тенге}.$$

4.4 кесте – Құралдыр кестесі

Атауы	Сипаттамасы	Саны, дана	1 дананың бағасы, тг	Сомасы, тг
Acer nitro 5, ноутбук	Intel Core i3 7020U 2300 MHz / 15.6" / 1920x1080 / 4Gb DDR4 / 256Gb SSD / Intel HD Graphics 620 / Wi-Fi / Bluetooth / Windows 10	1	432000	432000
PyCharm	Python программалық кодын орындауға арналған әзірлеу ортасы	1	8 500	8 500
Жалпы сома				440500

Жылдық амортизациялық аударымдар сомасы мынадай формула бойынша азайтылатын қалдық әдісімен анықталады:

$$A = \frac{Ф \times Н_а}{100}, \quad (4.9)$$

Мұнда Ф – негізгі өндірістік қорлардың бастапқы құны;

$Н_а$ – амортизация нормасы.

Азайтылатын қалдық әдісін пайдалану кезінде амортизацияның екі еселенген ставкасы қолданылады.

Негізгі қорлар амортизациясының жылдық нормалары ҚР Салық кодексі бойынша қабылданады немесе негізгі қорларды пайдалы пайдалану мерзімінің негізінде мына формула бойынша анықталады:

$$Н_а = \frac{100}{Т}, \quad (4.10)$$

Мұнда Т - пайдаланудың ықтимал мерзімі, жыл.

Бағдарламалық өнім үшін пайдалы пайдалану мерзімі – 4 жыл.

$$Н_а = \frac{100}{4} \times 2 = 50\%$$

4.5 кесте – амортизациялық салымдар

Период	Амортизация нормасы	Кезең ішіндегі Амортизация	Жинақталған амортизация	Жыл соңындағы баланстық құны
				440500
2020 год	50%	220250	220250	220250
2021 год	50%	110125	330375	110125
2022 год	50%	55062.5	385437.5	55062.5
2023 год	50%	27531.25	440500	0

Соңғы кезеңдегі Амортизация алдыңғы кезеңнің соңындағы баланстық құнына тең.

Бағдарламалық өнімді әзірлеуге жалпы шығындарды есептеу үшін әзірлеу кезеңі үшін амортизациялық аударымдарды есептеу қажет:

$$З_{ам} = \frac{Ф \times Н_а \times N}{100\% \times 12 \times t}, \quad (4.11)$$

Мұнда N – программалық қамтаманы қолдану мерзімі, күн.

t – 1 айдын ішіндегі жұмыс күндерінің саны.

Негізгі өндірістік қорлар өнімді әзірлеу және тестілеу кезеңінде ғана пайдаланылады, бұл пайдалану уақыты 232 сағатқа немесе 29 күнге тең.

Демек,

$$З_{ам} = \frac{440500 \times 50\% \times 29}{100\% \times 12 \times 21} = 25346.2 \text{ тенге.}$$

"Өзге шығындар" бабы басқару аппаратын, қосалқы шаруашылықтарды және тәжірибелік (эксперименталдық) өндірістерді ұстауға арналған шығындарды, сондай-ақ жалпы шаруашылық мұқтаждықтарға арналған шығыстарды қамтиды, орындаушылардың еңбекақысына пайыздық қатынаста нормативтік бойынша нақты шығыстарға жатады. Норматив тұтастай ұйым белгілейді:

$$P_{нi} = З_{тр} \times \frac{Н_{нр}}{100}, \quad (4.12)$$

мұнда $P_{нi}$ – нақты бойынша үстеме шығындар (мың тенге);

$Н_{нр}$ – жалпы ұйым бойынша үстеме шығыстар нормативі 70% - ға тең.

Демек,

$$P_{нi} = 162400 \times \frac{70}{100} = 113680 \text{ тенге.}$$

"Өзге шығындар" бабы бойынша нақты БҚ-ға арналған шығыстар арнайы ғылыми-техникалық ақпарат пен арнайы әдебиетті сатып алуға және дайындауға арналған шығындарды қамтиды. Жалпы ұйым бойынша әзірленетін норматив бойынша, жалақыға пайызбен анықталады:

$$П_{зi} = З_{тр} \times \frac{Н_{рнк}}{100} \quad (4.13)$$

мұнда $Н_{рнк}$ – ұйым бойынша жалпы өзге шығындар нормативі, 20% - ға тең.

Демек,

$$П_{зi} = З_{тр} \times \frac{20}{100} = 32480 \text{ тенге.}$$

4.6 кесте – орындалған есептеулердің нәтижелері

Шығындар	Шартты белгіленуі	Мәні, теңге	Жалпы сомадан пайызы
Еңбекақы	$З_{тр}$	162400	20,6%
Еңбекақыдан алынатын салықтар	$Зн$	16987	2,2%
Материалдар	M_i	8120	1,03%
Арнайы жабдықтар	$Р_{сі}$	440500	55.9%
Электр энергиясына кеткен шығындар	$З_Э$	600	0,1%
Негізгі қорлардың амортизациясы	$З_{ам}$	13142	1,67%
Басқа шығындар	$П_{зі}$	32480	4,12%
Үстеме шығыстар	$Р_{нi}$	113680	14,43%
Барлығы	$С_{пі}$	787909	100%

Өнімнің бастапқы бағасын мына формула бойынша есептеуге болады:

$$С_0 = С_{ni} + p,$$

Мұнда $С_0$ – табыс

$С_{ni}$ – өнімнің өзіндік бағасы

р – күтілетін пайда (шамамен өнімнің өзіндік бағасының 20% - 40% құрайды)

$$p = С_{ni} * 0.2 = 787909 * 0,4 = 315163,6$$

$$С_0 = С_{ni} + p = 787909 + 315163,6 = 1103072,6$$

12% тең ҚҚС есебімен дайын өнімнің бағасы мынадай формула бойынша есептеледі:

$$С_р = С_0 + НДС, \quad (4.14)$$

Осыдан, бағдарламалық өнімнің қорытынды бағасы төменгі мәнге тең:

$$С_р = 1103072,6 + (1103072,6 \times 0.12) = 1235441,3 \text{ теңге.}$$

4.2 Бағдарламалық өнімнің салыстырмалы экономикалық тиімділігін есептеу

Матчтың қорытындысын болжай алатын қарапайым нейрожелі спорт талдаушыларының құрамын толығымен алмастыра алмайды, өйткені спорттық ойынның нәтижесіне әсер ететін факторлар тым көп. Және нейрожеліге жаңа факторларды енгізіп, оны жаттықтыруға уақыт жетпейді.

ПҚ-ны енгізуден күтілетін жылдық әсердің шамасы мына формула бойынша есептеледі

$$Э_г = Э_{уг} - К \times Е_н , \quad (4.15)$$

мұндағы, $Э_г$ – күтілетін жылдық экономикалық тиімділік, теңге;

$Э_{уг}$ – күтілетін шартты-жылдық үнем, теңге;

К – капиталды салымдар, теңге;

$Е_н$ – капиталды салымдар тиімділігінің нормативтік коэффициенті.

$Е_н$ келесі формула бойынша анықталады:

$$Е_н = \frac{1}{Т_н} = 0.25$$

Мұнда $Т_н$ – капиталды салымдардың өтелімділігінің нормативтік мерзімі, жыл.

Бағдарламалық өнімдер үшін өтелімділік мерзімі 4 жылға тең.

Спорт аналитигінің алатын орташа төлемақысы 150 000 тг-ге тең. Букмекерлік кеңседе спорт аналиктерінін қанша адам болатынын айту қиын, шағын кеңсеге 1 адам да жетуі мүмкін. Бір адамның жылдық жалақысы 1800000 теңгеге тең болады.

Сонда,

$$Э_г = 1800000 - 1235441,3 * 0.25 = 1491139,7 \text{ тг}$$

Капиталдық салымдардың экономикалық тиімділігінің есептік коэффициенті:

$$Е_р = \frac{Э_{уг}}{К}, \quad (4.16)$$

Капиталдық салымдардың өтелімділігінің есептік мерзімі мына формула бойынша жүргізіледі:

$$Т_р = \frac{1}{Е_р}, \quad (4.17)$$

Осы жерден,

$$Е_р = \frac{1491139,7}{1235441,3} = 1,21;$$

$$Т_р = \frac{1}{1,21} = 0,82 \text{ жыл.}$$

Өтімділік мерзімі 0,82 жылды құрайды немесе 9 айдан астам.

4.7 кесте – есептеулер нәтижесі

Көрсеткіштердің атауы	Мәні
Шығындарды шартты жылдық үнемдеу, теңге	1800000
Капиталды салымдардың экономикалық тиімділігінің коэффициенті	1,21
Капиталды салымдардың өтелу мерзімі, жыл	0,82

4.3 Экономикалық бөлім бойынша қорытынды

Спорт нәтижелерін болжау өте күрделі мәселе. Мен жасаған нейрожелі статистика бойынша 51-53% дәлдікпен болжам жасайды, орташа алғанда

букмекерлік кеңселер 56-57 % деген көрсеткіштерге жетеді. Программалық қамтама адамның жұмысын толықтай жасай алмайды, бірақ аналитик программа берген нәтижеге сүйеніп қорытынды дәлдікті тағы бірнеше пайызға арттыра алады.

5 Тіршілік қауіпсіздігі бөлімі

5.1 Өрт дабылдарының типін, санын және орналасуын анықтау

Дүние жүзінде 1 жылдың ішінде 5 милионнан көп өрт болады. Он мың адам оттың салдарынан қаза табады. Өрт үлкен материалды шығындарға алып келеді. Өндірісте өрттін алдын алу үшін арнайы системалар орнатылады. Адамдарға әсер ететін өрттің қауіпті факторлары мыналар, ашық от және от ұшқыны, ауа температурасының жоғарлауы, газ қышқылының концентрациясының көбеюі, қондырғылардың, ғмиараттырдың зақымдануы және қирауы, түтін. Өндірістерде өрт және жарылыстар технологиялық режимдердін бұзылуынан, электр қондырғылардың дұрыс қолданбаудан және т.б. себептерден болады. Қарапайым кеңседе өрт көзі ақаулары бар компьютердегі электрондық схемалары, техникалық қызмет көрсетуге арналған құралдар, қуаттандыру құрылғылары болуы мүмкін. Өрт қауіпсіздігін қамтамасыз ету кезінде электр сымдарына ерекше көңіл бөлу керек. Өрт қауіпсіздігі талаптарын сақтаудың бұл талабы статистикалық мәліметтерге сәйкес барлық өрттердің 85% - ы сапасыз орындалған сымдардың себебінен орын алады.

Өрттің алдын алу шаралары:

- құрылыстық-жобалау;
- техникалық;,
- ұйымдастырушылық;

Құрылыстық-жобалау шаралары - ғимараттар мен құрылыстардың отқа төзімділігімен анықталады (конструкция материалдары жанғыш, қиын жанатын, жанбайтын болып бөлінеді). Отқа төзімділік шегі дегеніміз – бұл оттың әсерінен құрылыс конструкцияларының бірінші сызат пайда болғанға дейінгі шыдайтын уақыт интервалы. Барлық құрылыс конструкциялары отқа төзімділік шегі бойынша 8 деңгейге бөлінеді. Ғимараттардың отқа төзімділік деңгейіне байланысты өрт кезінде эвакуациялау үшін шығатын жерлерге дейінгі қашықтықтар белгіленеді.

Техникалық шаралары**:**

- өмірге қажетті жүйелерді (жылу, жарықтандыру, вентиляция т.б.) орнатқан кездерде өрт қауіпсіздігі нормаларын сақтау;
- құрал-жабдықтар жұмысының тәртібі мен технологиялық процестер парамертлерін сақтау;
- әртүрлі қорғану жүйелерін пайдалану.

Ұйымдастырушылық шаралар - құрамына өрт қауіпсіздігі бойынша оқу өткізу, өрт қауіпсіздігі шараларының сақталуын тексеру кіреді.

Өрт сигнализациясын тікелей миссиясы – өртті дер кезінде анықтау, оның пайда болған жері туралы хабарлау, шығу жолдары бойынша адамдарды эвакуациялауға команда беру және автоматты өрт сөндіруді бастау.

Өрт сигнализациясының автоматты қондырғылары (АБҚ) бірнеше элементтен тұратын, әрқайсысының өз функционалы және тар мақсаты бар күрделі техникалық құралдар кешені болып табылады:

Қуаттандыру көздері

Өрт дабылы жүйелерінде электрмен жабдықтау сенімділігін қамтамасыз ету дәрежесі I-санатқа жатқызылуы тиіс.

Тәулік бойы адамдар болатын ғимараттарда — ауруханаларда, интернаттарда, қонақ үйлерде — электрмен жабдықтау міндетті түрде үш тәуелсіз қоректену көздерінен қамтамасыз етіледі, олардың біреуі жалпы қоректену жүйесі ажыратылған жағдайда пайдаланылатын автономды электр генераторы болуы тиіс. Басқа объектілер үшін екі қорек көзі жеткілікті — олар айнымалы токтың екі көзі немесе бір айнымалы және бір тұрақты токтың көзі болуы мүмкін. Олардың арасында ауыстырып қосу негізгі блок істен шыққан кезде автоматты түрде жүзеге асырылуы тиіс. Резервтік көз ретінде аккумуляторлық батареяларды пайдалануға болады, бұл ретте міндетті түрде зарядтау мүмкіндігі болуы тиіс.

Хабарлағыштар – датчиктер, өрт факторларының детекторлары. Жұмыс принципі бойынша хабарлағыштар АБҚ - да қол және автоматты болуы мүмкін. Қол өрт хабарлағыштары өрт белгілері анықталған кезде тікелей адам іске қосылады. Мұндай сенсорды іске қосу үшін түймені басу немесе рычагты бұру қажет-осыдан кейін дабыл сигналы қосылады. Кезекші режимге оралу үшін арнайы кілт пайдаланылады. Әдетте қол датчиктерін адамдар көп жиналатын жерлерде — оқу орындарында, бизнес-орталықтарда, сауда кешендерінде және т. б. қосымша орнатады.

Сонымен қатар, барлық хабарлағыштар әртүрлі жану факторларына сезімталдығына байланысты топтарға бөлінеді.

Түтін хабарлағыштар — ең танымал. Олар оптикалық және ионизациялық болуы мүмкін. Бастапқы кезеңде жану көбінесе (бірақ әрдайым емес) түтіннің бөлінуімен қатар жүретіндіктен, түтін датчиктерін қолдану өте тиімді болып саналады. Әдетте оларды санитарлық тораптар мен баспалдақ алаңдарынан басқа жабық үй-жайларда орнатады. Жұмыс істеу принципі түтін камерасында орналасқан жарық диод пен фотоқабылдағыштың өзара іс-қимылына негізделген оптикалық датчиктер ең көп таралған. Қалыпты режимде бұл екі элемент бір-бірімен өзара әрекеттеспейді, бірақ камераға түтін түскен жағдайда жарық диодынан жарықты үзіп, фотоқабылдағышқа түседі. Хабарлаушы өрт дабылы жүйесін басқару панеліне (блогына) дабыл сигналын іске қосады және береді.

Оптикалық датчиктердің плюстеріне жоғары емес баға және өртті ерте сатыда анықтау қабілеті жатады, алайда бөлмедегі шаңданудан немесе сезімтал элементтерден мұндай датчиктер жалған дабыл бере алады. Сонымен қатар, кейбір модельдер, мысалы, резеңке жану кезінде бөлінетін қара түтінге жауап бермейді.

Жылу хабарлағыштар. Әрбір өрт факторын анықтау принципі мен температураның белгілі бір деңгейге дейін өзгеруін және осы өзгерістердің

жылдамдығын ұстап тұратын жылу сезгіш элементтер қолданылатын бірнеше түрі бар. Олардағы жалпы жылу хабарлағышы егер үй-жайда температура күрт жоғарыласа іске қосылады. Көптеген датчиктерде іске қосылу шегі 70-72°С деңгейінде болады, кейде ол жеке теңшеледі. Мұндай датчиктерді барлық жерде, соның ішінде түтін датчиктерін қолдану мүмкін емес жерлерде орнатуға болады. Егер өрттің ең ықтимал белгісі температураның тез көтерілуі болған жағдайда ғана жылу хабарлағыштарын қолдану орынды. Олар төбелері биік үй-жайларда және "ыстық" цехтарда іс жүзінде пайдасыз.

Жалын хабарлағыштар-ашық жалынның немесе тұтану ошағының пайда болуына әсер ететін құрылғылар. Әдетте оларды өнеркәсіптік үй-жайларда, ашық алаңдарда, яғни жылу және түтін датчиктерін қолдану орынсыз жерлерде қолданады — мысалы, мұнай немесе газ өнеркәсібінде, онда жану өнімдерінің көпшілігі түтін бөлмейтін немесе аздаған мөлшерін бөлетін болады. Датчиктердің сезімтал элементтері бір немесе бірнеше оптикалық диапазонда жалынның электромагниттік сәулеленуіне әсер ете алады. Жалын хабарлағыштары-датчиктердің ең қымбат түрлерінің бірі. Олардың жалған іске қосылуы қыздыру шамдарын, күн сәулесі, доғалық дәнекерлеу және найзағайдың разрядтарын тудыруы мүмкін.

Сонымен қатар , өрт сигнализациясы жүйелерінде бір уақытта өрттің бірнеше белгілеріне, мысалы, түтіннің пайда болуына және үй-жайда температураның жоғарылауына ден қоятын құрамдастырылған хабарлағыштар орнатылуы мүмкін. Бұл өрт ошағының орналасуын нақты анықтауға және дабыл белгісін беруге мүмкіндік береді.

Байланыс желілері. Өрт дабылы жүйесінің барлық элементтері арасында ақпарат алмасуды және АБҚ қоректендіруді қамтамасыз етеді. Ол үшін сымды және сымсыз немесе аралас байланыс арналары қолданылуы мүмкін.

Қабылдау құрылғылары немесе қабылдау-бақылау және басқару аспаптары (ПКУ), өрт сөндіру (ПКП). Бұл барлық өрт сигнализациясы жүйесінің "миы", ол хабарлағыштардан өрт туралы сигналдарды қабылдауды жүзеге асырады, шлейфтер мен датчиктердің жұмысын бақылайды және диагностикалайды, деректерді Орталық пультке немесе жергілікті диспетчерге береді.

Өрт сигнализациясы жүйелері мынадай талаптарға сай болуы тиіс:

1) Ең ерте сатыларда жануды анықтау үшін объектіде 24/7 режимінде бақылауды қамтамасыз ету.

2) Өртті оқшаулауды барынша дәл анықтау.

3) Өрт жүйесі жалған жұмыс істемеуі тиіс.

Бірақ бұл міндеттерді орындау үшін нақты объект үшін қандай АБҚ қолайлы екенін анықтау қажет.

Өрт сигнализациясының түрін таңдау кезінде өрт қауіпсіздігі жүйесінің құрамында АБҚ орындауға тиіс міндеттерді, сондай-ақ қорғау объектісіндегі оның жұмысының техникалық шарттарын басшылыққа алу қажет.

Өрт дабылы жүйесінің үш түрі белгілі: шектік, адрестік-сауалнамалық және адрестік-аналогтық.

Шекті немесе дәстүрлі АБҚ. Оның жұмыс істеу принципі шлейфтегі кедергінің өзгеруіне негізделген, ал датчиктер тек екі негізгі жағдайда ғана болуы мүмкін: "норма" және "өрт". Бастапқы АБҚ құрылымы (топологиясы) шлейфтердің радиалды орналасуымен ерекшеленеді, яғни бақылау пультінен хабарлағыштар қосылған бірден бірнеше шлейф жұмсалады. Егер датчиктердің біреуі жануды тіркеген жағдайда, ол өзінің ішкі кедергісін өзгертеді, ал папканы осы хабарлаушы жататын шлейф бойынша дабыл сигналы береді. Әрдайым өрттің көзін анықтау мүмкін емес, себебі бір шлейфке 10-20 шақты датчик қосылып тұратын жағдай болады. Бастапқы жүйенің артықшылығы төмен құны мен орнату қарапайымдылығын жатқызуға болады. Ең маңызды минус-жалған іске қосылудың үлкен ықтималдығы, салыстырмалы түрде кеш сатыларда өртті табу және қызмет көрсетудің ыңғайсыздығы.

Адрестік-сауалнамалық (АБҚ). Ол автоматты режимде датчиктердің күйін мезгіл-мезгіл тексеруге қабілетті. Бұл жүйенің жұмыс істеу принципі хабарлағыштарды сұраудың өзге алгоритмінде болады. Құрылғы хабарлаушы режимін ауыстыру сигналын күтпейді, ал уақыт өте келе оның ағымдағы жай-күйіне жауап алуды жүзеге асырады. Әрбір датчикке өз нөмірі берілді, бұл қабылдау-бақылау аспабына жану ошағының қайда екенін дәл анықтауға мүмкіндік береді. Осындай АБҚ датчиктерінің бірнеше физикалық жағдайы болуы мүмкін — "норма", "өрт", "ақаулық", "үзілу" және т.б., олар өз бетімен басқа жағдайға көшуге қабілетті. Адрестік-сауалнамалық АБПС — дағы шлейф түрі-көбінесе сақиналы. Мұндай жүйелерді жалпы мақсаттағы бір типті үй-жайларда: офистерде, дүкендерде, оқу және медициналық мекемелерде пайдаланған жөн. Адрестік-сауалнамалық АБҚ жоғары ақпараттылығымен ерекшеленеді.

Адрестік-Аналогты АБҚ. Адрестік-сауалнамалық жүйе сияқты функционалға ие бола отырып, ол хабарлағыштардан сигналдарды өңдеу тәсілімен айтарлықтай ерекшеленеді. Бұл жағдайда ақпаратты талдау және басқа күйге ауысу туралы шешімді хабарлағыштың өзі емес, бақылау панелі (ПБК) қабылдайды. Бұл барлық АБҚ сыртқы факторларға теңшеуге мүмкіндік береді, ал басқару панелінің ішіндегі орталық контроллер бірқатар күрделі функцияларды жүзеге асыруға қабілетті: датчиктерді сұрау, ақпаратты талдау және түсіндіру, деректерді шекаралық мәндермен салыстыру, бірнеше типті хабарлағыштармен біріктірілген деректер негізінде түпкілікті шешім қабылдау және т. б. Мұндай жүйелердің артықшылықтарының бірі — кез келген түрдегі топологияны қолдану мүмкіндігі: "сақина", "жұлдыз" немесе "шина", бұл жүйе элементтері арасындағы байланыс желілері үзілген жағдайда жүйенің жұмысқа қабілеттілігін сақтау үшін шлейфтер төсемесінің ең қолайлы нұсқасын таңдауға мүмкіндік береді.

Жоғарыда айтылғандай, офистік бөлмеге адрестік-сауалнамалық жүйені қойған дұрыс болады. Ал хабарлағыш ретінде түтінді бақылайтын шешімді алайық.

Офистік бөлменің параметрлері келесідей: ұзындығы – 9 м, ені – 8 м, биіктігі – 4 м.

Өрт хабарлағыштар арасындағы ең үлкен қашықтықты анықтайық

$$x_1 = \sqrt{\frac{(2S_{\text{бп}})}{\pi}}, \quad (5.1)$$

Мұндағы $S_{\text{бп}}$ – бір өрт хабарлағышының бақылай алатын ауданы, норма бойынша түтін хабарлағыштың көрсеткіші 20 м^2-қа тең.

$$x_1 = \sqrt{\frac{(2 * 20)}{3.14}} = 3.57$$

Хабарлағыштың қабырғаға дейінгі ең үлкен қашықтық

$$y_1 = \frac{x_1}{2} = \frac{3.57}{2} = 1.79$$

Бұрыштық өрт хабарландырушыларының арасындағы қашықтық

$$L_1 = A - 2y_1 = 8 - 3.57 = 4.43$$
$$L_1 = B - 2y_1 = 9 - 3.57 = 5.43$$

Өрт хабарландырушыларының қатарлары арасындағы аралық санын анықтаймыз

$$M = \frac{L_1}{x_1} = \frac{4.43}{3.57} = 1.2$$

Және қатарда

$$N = \frac{L_2}{x_1} = \frac{5.43}{3.57} = 1.52$$

M және N мәндерін ең жақын үлкен көрсеткішке келтіреміз

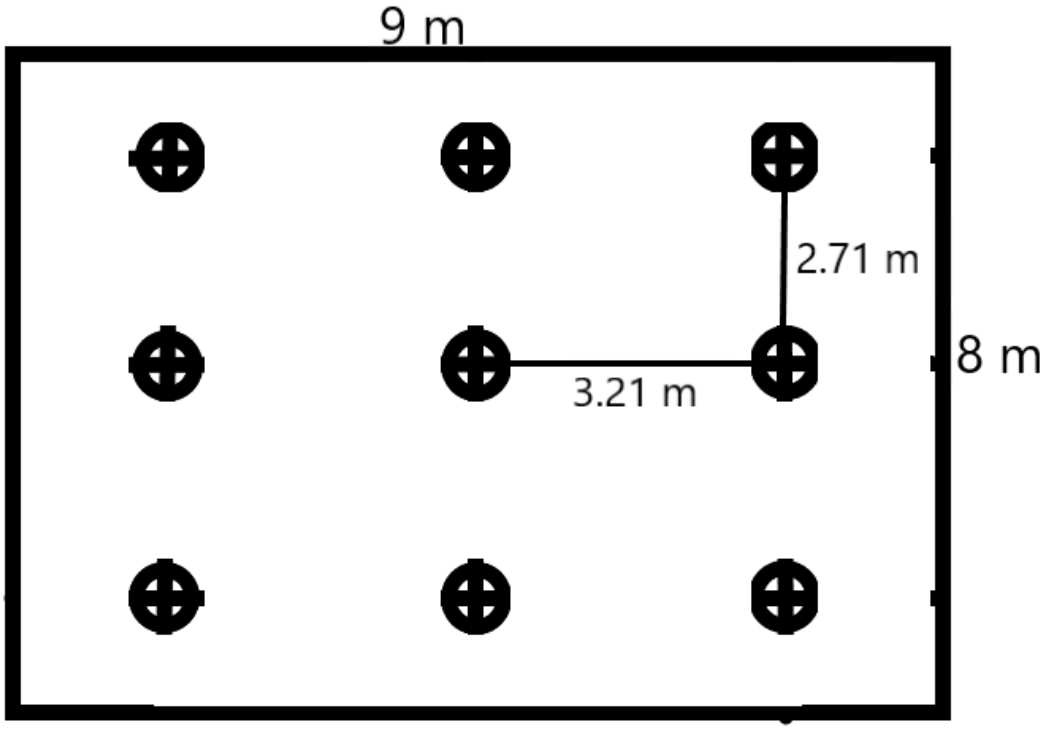
$$M = 2 \text{ дана}$$
$$N = 2 \text{ дана}$$

Қатардағы хабарлағыштар арасындағы қашықтықты анықтаймыз

$$n = \frac{L_2}{N} = \frac{5.43}{2} = 2.71 \text{ м}$$

Және катар арасында

$$m = \frac{L_1}{M} = \frac{6.43}{2} = 3.21 \text{ м}$$



5.1 сурет – от хабарлағыштарының орналасу схемасы

Осылайша біздің бөлмде 9 өрт хабарлағыш орналасады.

5.2 ПЭВМ операторының қол жеткізу аймағының өлшемдерін, ауданын, нысанын анықтау. Қол жеткізу шарттарынан пернетақта мен тінтуірдің орналасуын анықтау

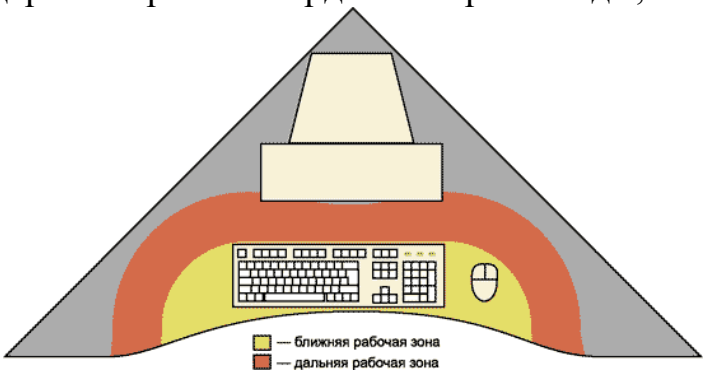
Жұмыс орнының дұрыс ұйымдастырылуы сіздің өнімділігіңізге айтарлықтай әсер етуі мүмкін. Бұл сіздің уақытыңызды үнемдейді, шаршаудың алдын алады және тапсырмаларды жоспарланғаннан тезірек орындайды.

Жұмыс орнын ұйымдастыру кезінде келесі міндеттерді шешу қажет: жұмыс орнын берілген технологиялық процестерге сәйкес еңбек құралдарымен және еңбек заттарымен жабдықтау; жұмыс орнын ұтымды жоспарлау; жұмыс қауіпсіздігін, қалыпты еңбек жағдайларын қамтамасыз ету.

Алғаш ойланатын нәрсе сатып алынатын үстел мен орындық. Кез келген үстелдің ыңғайлылығын бағалау үшін әмбебап критерийлердің бірі — жұмыс аймағының өлшемдері. Үстел басында отырған адам денесіне қысылған шынтақпен жұмыс істей алатын столшаның учаскесі жақын жұмыс аймағы деп аталады, ал ол қолын толығымен түзетіп, созылуы мүмкін аймақ — алыс жұмыс аймағы деп аталады. Жұмыс аймағының көлемі неғұрлым көп болса, үстелде жұмыс істеу ыңғайлы. 1-суретте көбіне қолданылатын үстел формасы көрсетілген. Бұл жағдайда жұмыс аймағының ауданы жеткілікті деп саналады.

5.2 сурет - Жұмыс аумақтары

Қазіргі таңда әлдеқайда ыңғайлы үстелдер бар, олардың формасы мен жұмыс аймағы стандартты көрсеткіштерден жоғары болады, мысалы



5.2 сурет - Ыңғайлы үстел варианты

5.4 сурет - Өте ыңғайлы үстел варианты

Суреттерде көрсетілгендей, пернетақта мен тінтуір әрдайым жақын жұмыс аймағында болу керек.

Пернетақтаны таңдағанда, оның сыртқы түріне ғана емес, сондай-ақ пернелердің жүрісіне және қатаңдығына да назар аудару қажет. Ұзақ уақыт бойы тым жеңіл немесе тым қиын пернелер елеулі жайсыздық тудыруы мүмкін. Сонымен қатар, пернетақтаның әртүрлі модельдерінде жиі қолданылатын пернелердің (Shift, Enter, Backspace) пішіні мен өлшемдері

әртүрлі және көптеген пайдаланушылар үшін бұл маңызды фактор болып табылады.

Тінтуір корпусының пішіні мен өлшемдері манипуляторды оңай жылжытуға және тиісті қол үстелдің бетіне тірелмейтін жағдайда басқару органдарымен операция жасауға мүмкіндік бере отырып, жұмыс процесінде қолдың табиғи жағдайын қамтамасыз етуі тиіс. Өте кішкентай немесе керісінше, тым үлкен тінтуірмен ұзақ жұмыс ыңғайсыздық, шаршау және тіпті кейбір аурулардың пайда болуына себеп болуы мүмкін.

Қорытынды

Бұл дипломдық жоба спорт жарыстарының нәтижесін болжауға арналған нейрондық желінің моделін құруға арналған. Бұл тақырыпты зерттеу барысында бұл мақсаттар үшін Python бағдарламалау тілі жақсы екені белгілі болды. Python-да деректерді жинау, статистика және нейрондық желіні құру үшін барлық қажетті құралдар бар.

Нейрондық желіні құрып, жаттықтыру үшін Scikit-learn кітапханасы қолданылды. Себебі ол классикалық машинамен оқыту міндеттерін шешу үшін ең көп таралған таңдау. Ол мұғаліммен және мұғалімсіз оқыту алгоритмдерінің кең таңдауын ұсынады. Мұғаліммен оқыту мақсатты белгінің мәні Белгілі белгіленген датасеттің болуын көздейді. Мұғалімсіз оқыту датасетте белгі қоюды көздемейді-еркін деректерден пайдалы ақпаратты алуды үйрену қажет. Кітапхананың негізгі артықшылықтарының бірі ол бірнеше кең таралған математикалық кітапханалар негізінде жұмыс істейді және оларды бір-бірімен оңай біріктіреді. Тағы бір артықшылығы-кең қоғамдастық және толық құжаттама. Scikit-learn өнеркәсіптік жүйелер үшін кеңінен қолданылады, онда классикалық Машиналық оқыту алгоритмдері қолданылады, зерттеу үшін, сондай-ақ Машиналық оқыту саласында алғашқы қадамдар жасап жатқан адамдар үшін ең дұрыс таңдау.

Пайдаланушының ыңғайлылығы үшін телеграм-бот жасалды. Боттар-бұл Telegram ішінде жұмыс істейтін бөгде қосымшалар. Пайдаланушылар боттармен өзара әрекеттесе алады, оларға хабарламалар, командалар және кірістірілген сұраныстар жібере алады. Ботты басқару екі тәселмен жүзеге асырылады: HTTPS-сұрауларды пайдалана отырып немесе Bot API арқылы.

Әдебиеттер тізімі

1. А.Т. Купарова. Методическое указания к выполнению дипломных работ (проектов) для студентов специальности 5В070300 – Информационные системы. – Алматы: АУЭС, 2012 – 39 с.
2. С. Белоусова, И. Бессонова, Руджеро Гиляревский. Введение в программные системы и их разработку. НИУ ВШЭ (қаралған күн 14.04.20).
3. Байт-код // URL: https://ru.wikipedia.org/wiki/Байт-код (қаралған күн 21.03.20).
4. Блок-схема // URL: https://ru.wikipedia.org/wiki/Блок-схема
5. Алан Джок. Компиляторы, интерпретаторы и байт-код. «Computerworld Россия», № 06, 2001. (қаралған күн 15.04.20).
6. TCP/IP // URL: https://professorweb.ru/my/csharp/web/level4/4_1.php (қаралған күн 20.04.20).
7. Экономика и организация производства. Конспект лекций. - А.: АУЭС, 2012.2. Боканова Г. Ш., Еркешева З.Д. Экономика и организация производства: Методические указания к выполнению расчетно-графических работ –А.:АУЭС. (қаралған күн 11.04.20).
8. Суша, Г.З. Экономика предприятия: учебное пособие / Г. З. Суша. – 3-е издание., испр. и доп. – Москва: Новое знание, 2010. – 512 (қаралған күн 18.04.20).
9. Сөздік // URL: sozdik.kz (қаралған күн 10.05.20).
10. ПЛК // Э. Парр. Программируемые контроллеры: руководство для инженера. — М.: БИНОМ. Лаборатория знаний, 2007. — 516 с
11. Минаев И.Г. Свободно программируемые устройства в автоматизированных системах управления / И.Г. Минаев, В.В. Самойленко, Д.Г. Ушкур, И.В. Федоренко - Ставрополь: АГРУС. 2016. - 168 с.
12. Венн диаграммасы // URL: https://www.researchgate.net/figure/Venn-diagram-of-the-components-of-artificial-intelligence_fig1_328826136
13. Нейрондық желінің структурасы // URL: https://mc.ai/neural-network-fundamentals/
14. Нейронды желіні жаттықтару туралы // URL: https://econet.kz/articles/180490-pervostepennaya-zadacha-kvantovyh-kompyuterov-usilenie-iskusstvennogo-intellekta
15. Tensorflow кітапханасымен танысу // URL: https://www.tensorflow.org/learn
16. Оқыту туралы // URL: https://www.mdpi.com/1996-1073/12/4/608/htm
17. Активация функциялары жайында // URL: https://arxiv.org/