МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

ДОСЛІДЖЕННЯ BIG DATA МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Дипломна робота

на здобуття ступеня вищої освіти магістр

Виконав: студент 2 курсу 261М групи
Спеціальності: 161 Інформаційні системи і технології
Веренич Костянтин Олександрович
Керівник: кандидат фізико-математичних наук, доцент
Вейцбліт Олександр Йосипович
Рецензент: кандидат фізико-математичних наук, доцент
Котова Ольга Володимирівна

Херсон - 2019

ЗМІСТ

ВСТУП

Технології машинного навчання задали дуже швидкий темп поширення у сучасній IT-спільноті. З’являється величезна кількість стартапів, які беруть за основу ідею, яку можна втілити у життя тільки методами машинного навчання. Одним із основних програмних засобів для розробки таких інструментів — є мова програмування Python, яка має одне з найбільших спільнот серед інших мов програмування, але при цьому досить складно знайти приклад з визначеними етапами розробки повноцінного проекту по дослідженню big data з використанням методів машинного навчання. Перед проектом поставлена задача – NLP, тобто обробка звичайної мови використовуючи big data у промислових масштабах, актуальність чого тільки росте з кожним днем.

Мета курсової роботи: ознайомитися с машинним навчанням та його застосуванням на мові Python та розробити додаток обробки big data використовуючи аккаунти соціальної мережі Instagram.

Завдання:

1. Дослідити поняття та призначення машинного навчання.
2. Визначити основні типи та підходи машинного навчанная.
3. Ознайомитися с технологіями (бібліотеками, фреймворками).
4. Дослідити джерело даних для додатку. (Облікові записи користувачів соціальних мереж)
5. Розробити нейронну мережу
6. Розробити браузерний додаток та підключити до API классифікації аккаунтів соціальних мереж
7. Навчити нейронну мережу та корегувати її роботу

Об'єктом дослідження даної теми є машинне навчання та задачі, які воно вирішує, алгоритми, які воно використовує. Big data - дані з якими воно працює.

Предметом дослідження є бібліотеки по машинному навчанню на мові програмуваня Python, фреймворки для будування веб-додатку та ресурси-джерела big data.

РОЗДІЛ 1

МАШИННЕ НАВЧАННЯ ЯК ОБ’ЄКТ ДОСЛІДЖЕННЯ

1.1 Історія та прогресс технології машинного навчання

Ще не так давно машинне навчання було суто вузкою “галузю”, представників якої навряд налічувало кілька сотень вчених, розкиданих по обраним університетам і лабораторіям. Але до 2012 року в силу ряду причин, машинне навчання вийшло на передній план, але досі не отримало достатнього визнання і популяризаціїї . Дивно і показово - машинному навчання була віддана перша смуга The New Times, йому були присвячені статті в Forbes і The New Yorker. На жаль, у нас важко собі уявити щось подібне. А через три роки машинне навчання доходить до комп'ютерних і околокомпьютерих мас і стає актуальною для підприємництва темою в Кремнієвій долині. Як і чому змогла відбутися ця стрімка метаморфоза?

Машинне навчання (Machine Learning, ML) - це напрямок в науці, а з недавніх пір і в технологіях, яке вирішує завдання навчання комп'ютерів. Під цим розуміють передачу апаратно-програмних комплексів якогось суто обмеженого набору знань з можливістю їх подальшого накопичення. В даному контексті не передбачається повноцінне навчання, порівнянне з людським. Машинне знання не дозволяє приймати по-справжньому інтелектуальні рішення, які можна порівняти з можливостями людини.

Нинішня практична потреба в машинному навчанні виникла в зв'язку з тим, що сьогодні різноманітність вхідних даних і можливих рішень стає занадто велике для традиційних заздалегідь запрограмованих систем.

Машине навчання реалізується на обчислювальних і статистичних принципах, об'єднуючи найрізноманітніші підходи, включаючи теорію ймовірностей, статистику, логіку, обчислювальну оптимізацію, пошукові

методи, навчання з підкріпленням, теорію управління.

Область застосування машинного навчаня на даний момент поширюється на широке коло додатків, среди яких обробка всіх можливих видів даних (текст, відео, аудіо), прогнозування, розпізнавання образів, видобуток даних (data mining), експертні системи, робототехніка і навіть ігри.

Історія машинного навчання, як і багато іншого в штучному інтелекті, почалася з багатообіцяючих робіт в 1950-х - 1960-х роках, а потім був тривалий період накопичення знань, відомий як «зима штучного інтелекту». В останні роки спостерігається вибуховий інтерес головним чином до одного з напрямків - глибинного, або глибокого навчання (deep learning).

Піонерами технології були Артур Семюель, Джозеф Вейцбаум і Френк Розенблат. Перший отримав широку популярність створенням в 1952 році самонавчальної програми Checkers-playing, яка вміла грати в шашки. Можливо, більш значущим для нащадків виявилося його участь разом з Дональдом Кнутом в проекті TeX, результатом якого стала система комп'ютерної верстки, ось уже майже 40 років не має собі рівних для підготовки математичних текстів. Другий в 1966 році написав віртуального співрозмовника ELIZA, здатного імітувати (а скоріше, пародіювати) діалог з психотерапевтом. А далі всіх пішов Розенблатт, він в кінці 50-х в Корнельському університеті побудував систему Mark I Perceptron, яку можна визнати першим нейрокомп'ютером.

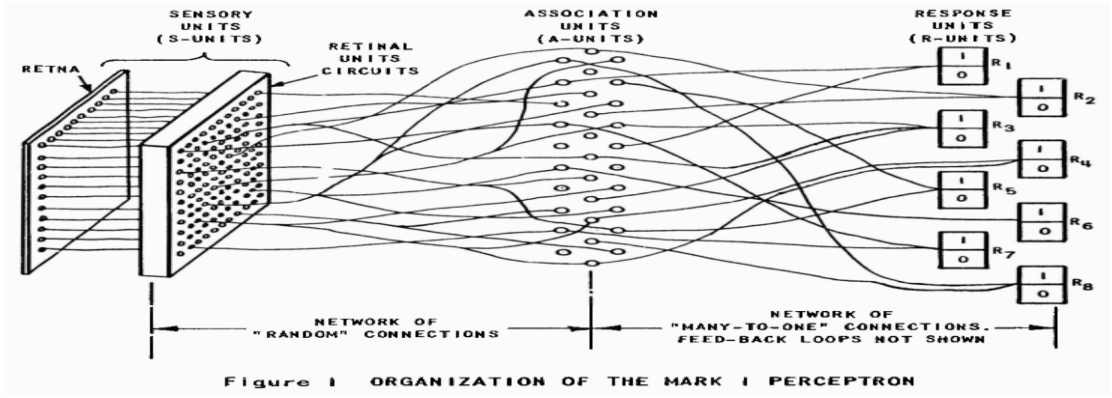

Рис 1.1. Перцептрон Розенблатта

Mark I призначався для класифікації візуальних образів (символів алфавіту) і представляв собою електронно-механічну систему, центром якої були 400 керованих фото-сенсорів, вони служили моделлю сітківки.

За наступні 30-40 років зусиллями академічно орієнтованих вчених машинне навчання було перетворено в самостійну математичну дисципліну.

Початок першого десятиліття XXI століття виявилося поворотною точкою в історії машинного навчання, і пояснюється це трьома синхронними тенденціями, які дали в сукупності помітний синергетичний ефект. Перша - Великі Дані (Big data). Даних стало так багато, що нові підходи були винайдені, а старі відкопані не через допитливість вчених, а через практичну необхідність. Друга - зниження вартості паралельних обчислень і пам'яті. Ця тенденція виявилася в 2004 році, коли компанія Google розкрила свою технологію MapReduce, за якою послідував її відкритий аналог Hadoop (2006), і всі разом вони дали можливість розподілити обробку величезних обсягів даних між простими процесорами. Тоді ж Nvidia зробила прорив на ринку GPU: якщо раніше в ігровому сегменті їй могла скласти конкуренцію AMD / ATI, то в сегменті графічних процесорів, які можна використовувати для цілей машинного навчання, вона виявилася монополістом. І в той же час помітно зменшилася вартість оперативної пам'яті, що відкрило можливість для роботи з великими обсягами даних в пам'яті і, як наслідок,

з'явилися численні нові типи баз даних, в тому числі NoSQL.

Третя - нові алгоритми глибинного машинного навчання, успадковуючі і розвиваючі ідею перцептрону в поєднанні з вдалою науковою PR-кампанією. Своєю критикою Марвін Мінський і Сеймур Паперть зіграли позитивну роль, вони розкрили слабкості перцептрону в тому вигляді, як його придумав Розенблатт, і одночасно стимулювали подальші роботи по нейронних мережах, які до 2006 року залишалися теоретичними. Ймовірно, першим, хто вирішив «розгвинтити» (поглибити) перцептрон, був радянський математик А.Г. Івахненко, який опублікував починаючи з 1965 року ряд статей і книг, в яких, зокрема, описана моделює система «Альфа». У 1980 році Куніхіко Фукусіма запропонував ієрархічну багатошарову сверточное нейронну мережу, відому як неокогнітрон.

Наступні роки позначені інтенсивною роботою багатьох вчених в області глибинних нейронних мереж (Deep Neural Network, DNN), однак детальний і бажано об'єктивний аналіз подій цього періоду ще чекає свого дослідника. Вважається, що власне термін deep learning був запропонований в 1986 році Ріной Дехтерев, хоча історія його появи, мабуть, складніше.

До середини минулого десятиліття була накопичена критична маса знань в області DNN, і в хедлайнером стає Джефрі Хінтон, британський вчений, що продовжив свою кар'єру в Канаді. С 2006 року він сам і разом з колегами почав публікувати численні статті, присвячені DNN, в тому числі і в науково-популярному журналі Nature, чим заслужив собі прижиттєву славу класика. Навколо нього утворилося сильне і згуртоване співтовариство, яке кілька років працювало, як тепер горять, «в невидимому режимі.

1.2 Застосування машинного навчання

У бізнесі можна розглядати три сфери застосування машинного навчання:

- описова
- прогнозуюча
- нормативна

Описове застосування відноситься до запису та аналізу статистичних даних для розширення можливостей бізнес-аналітики. Керівники отримують опис і максимально інформативний аналіз результатів і наслідків минулих дій і рішень. Цей процес в даний час звичайний для більшості великих компаній по всьому світу - наприклад, аналіз продаж і рекламних проектів для визначення їх результатів і рентабельності.

Друге застосування машинного навчання - прогнозування. Збір даних і їх використання для прогнозування конкретного результату дозволяє підвищити швидкість реакції і швидше приймати вірні рішення. Сьогодні цей процес застосовується в більшості великих компаній.

Третє і найбільш просунуте застосування машинного навчання впроваджується вже існуючими компаніями і вдосконалюється зусиллями недавно створених. Простого прогнозування результатів або поведінки вже недостатньо для ефективного ведення бізнесу. Розуміння причин, мотивів і навколишньої ситуації - ось необхідна умова для прийняття оптимального рішення. Цей метод найбільш ефективний, якщо людина і машина об'єднують зусилля. Машинне навчання використовується для пошуку значущих залежностей і прогнозування результатів, а фахівці з даними інтерпретують результат, щоб зрозуміти, чому такий зв'язок існує. В результаті стає можливим приймати більш точні і правильні рішення.

Простий приклад:

Вибираються вхідні дані і задаються умови введення (наприклад, банківські операції з використанням карт).

Будується алгоритм машинного навчання і налаштовується на конкретну задачу (наприклад, виявляти шахрайські транзакції).

Використовувані в ході навчання дані доповнюються бажаної вихідною інформацією (наприклад, ці транзакції - шахрайські, а ці ні).

1.3 Процес роботи машинного навчання

Штучний інтелект - це наука і технологія по розробці заходів і методів, що дозволяють комп'ютерам успішно виконувати завдання, які вимагають інтелектуального осмислення людини. Машинне навчання - частина цього процесу: це методи і технології, за допомогою яких можна навчить комп'ютер виконувати поставлені завдання.

Методи машинного навчання все ще в розвитку. Деякі вже вивчені і використовуються, але очікується, що з часом їх кількість буде тільки зростати. Ідея в тому, що абсолютно різні методи використовуються для абсолютно різних комп'ютерів, а різні бізнес-завдання вимагають різних методів машинного навчання.

Для вирішення комп'ютером завдань із застосуванням штучного інтелекту потрібні практика і автоматичне налаштування. Модель машинного навчання потребує тренування з використанням бази даних і в більшості ситуацій - в підказці людини.

Штучний інтелект потребує надання досвіду - іншими словами, йому необхідні дані. Чим більше в систему штучного інтелекту надходить даних, тим точніше комп'ютер взаємодіє з ними, а також з тими даними, що отримує в подальшому. Чим вище точність взаємодії, тим успішніше буде виконання поставленого завдання, і вище ступінь прогностичної точності.

Машинне навчання часто називають чарівним або чорним ящиком:

Введення даних → «чарівний чорний ящик» → виведення виконана.

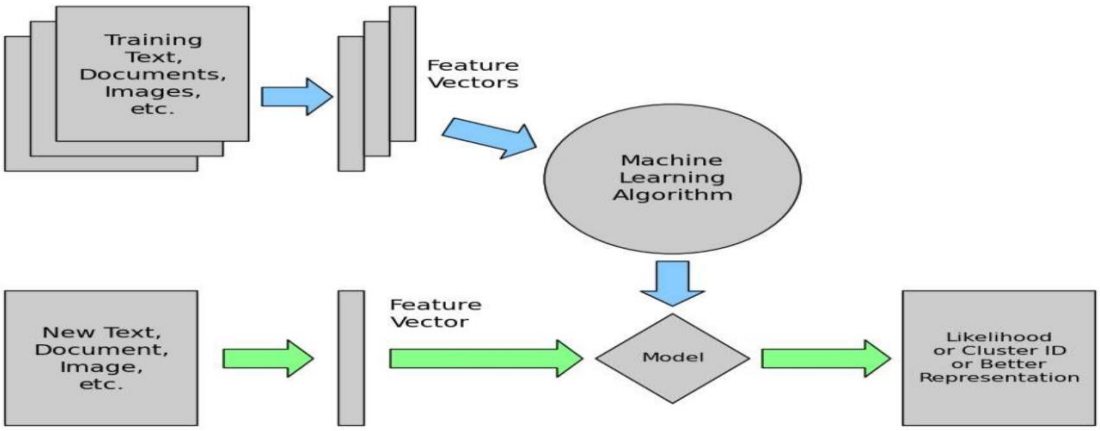

Рис 1.2. Повний процес машинного навчання

Процес навчання:

Збір

Машинне навчання ґрунтується на даних. Перший крок - переконатися, що наявні дані вірні і відносяться саме до тієї задачі, яку ви намагаєтеся вирішити. Слід оцінити свої можливості для збору даних, обміркувати їх джерело, необхідний формат і т. Д.

Очищення

Дані часто формуються з різних джерел, відображаються в різних форматах і мовах. Відповідно, серед них можуть опинитися нерелевантні або непотрібні значення, які потрібно видалити. І навпаки, якихось даних може не вистачати, і буде потрібно їх додати. Від правильної підготовки бази даних прямим чином залежить і придатність до використання, і достовірність результатів.

Поділ

Залежно від розміру набору даних в деяких випадках може знадобитися тільки невелика їх частина. Зазвичай це називається вибіркою. З обраної частини дані треба розділити на дві групи: одна для використання алгоритмом, а інша для оцінки його дій.

Навчання

Цей етап фактично спрямований на пошук математичної функції, яка точно виконає зазначену задачу. Навчання різниться в залежності від типу використовуваної моделі. Побудова ліній в простій лінійній моделі - це навчання; генерація дерева прийняття рішень для алгоритму випадкового лісу - це також навчання. Зміна відповідей при побудові дерева рішень допоможе скорегувати алгоритм.

Алгоритм використовує частину даних, обробляє їх, заміряє ефективність обробки і автоматично регулює свої параметри (метод зворотного поширення помилки) до тих пір, поки не зможе послідовно проводити бажаний результат з достатньою достовірністю.

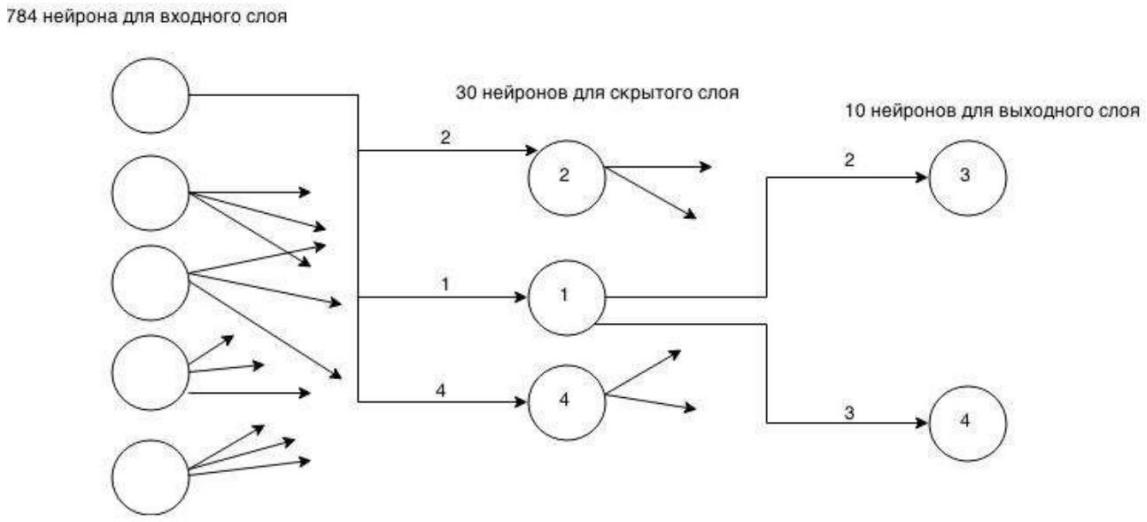

Рис 1.3. Навчання нейронної мережі

Оцінка

Після того як алгоритм добре показав себе на навчальних даних, його ефективність оцінюється на даних, з якими він ще не стикався. Додаткове корегування здійснюється при необхідності. Цей процес дозволяє запобігти перенавчання - явище, при якому алгоритм добре працює тільки на навчальних даних.

Оптимізація

Модель оптимізується, щоб при інтеграції в додаток важити як можна менше і якомога швидше працювати.

1.4 Класифікація машинного навчання

Існує безліч моделей для машинного навчання, але вони, як правило, відносяться до одного з трьох типів:

- навчання з учителем (supervised learning);
- навчання без учителя, або самонавчання (unsupervised learning);
- навчання з підкріпленням (reinforcement learning).

Залежно від виконуваної завдання, одні моделі можуть бути більш придатними і більш ефективними, ніж інші.

Навчання з учителем (supervised learning).

У цьому типі коректний результат при навчанні моделі явно позначається для кожного ідентифікованого елемента в наборі даних. Це означає, що при зчитуванні даних у алгоритму вже є правильна відповідь. Тому замість пошуків відповіді він прагне знайти зв'язку, щоб в подальшому, при введенні непозначених даних, виходили правильні класифікація або прогноз.

В контексті класифікації алгоритм навчання може, наприклад, забезпечуватися історією транзакцій по кредитних картах, кожна з яких позначена як безпечна або підозріла. Він повинен вивчити відносини між цими двома класифікаціями, щоб потім зуміти відповідним чином маркувати нові операції в залежності від параметрів класифікації (наприклад, місце покупки, час між операціями і т. Д.).

У разі коли дані безперервно пов'язані один з одним, як, наприклад, зміна курсу акцій у часі, регресійний алгоритм навчання може використовуватися для прогнозування наступного значення в наборі.

Навчання без вчителя (unsupervised learning).

В цьому випадку у алгоритму в процесі навчання немає заздалегідь встановлених відповідей. Його мета - знайти смислові зв'язки між окремими даними, виявити шаблони і закономірності. Наприклад, кластеризація - це використання неконтрольованого навчання в

рекомендаційних системах.

Навчання з підкріпленням (reinforcement learning).

Цей тип навчання є сумішшю перших двох. Зазвичай він використовується для вирішення більш складних завдань і вимагає взаємодії з навколишнім середовищем. Дані надаються середовищем і дозволяють алгоритму реагувати і вчитися.

Область застосування такого методу обширна: від контролю роботизованих рук і пошуку найбільш ефективної комбінації рухів, до розробки систем навігації роботів, де поведінковий алгоритм «уникнути зіткнення» навчається досвідченим шляхом, отримуючи зворотний зв'язок при зіткненні з перешкодою.

Основні типи задач машинного навчання

Класифікація

Вибірки даних належать до двох або більше класів і ми хочемо навчитися на вже розмічених даних передбачати клас нерозміченої вибірки. Прикладом завдання класифікації може стати розпізнавання рукописних чисел, мета якого - дати кожному вхідному набору даних одну з кінцевого числа дискретних категорій. Інший спосіб розуміння класифікації - це розуміння її як дискретної (як протилежність безперервної) форми керованого навчання, де у нас є обмежена кількість категорій, наданих для N вибірок; і ми намагаємося їх помітити правильної категорією або класом.

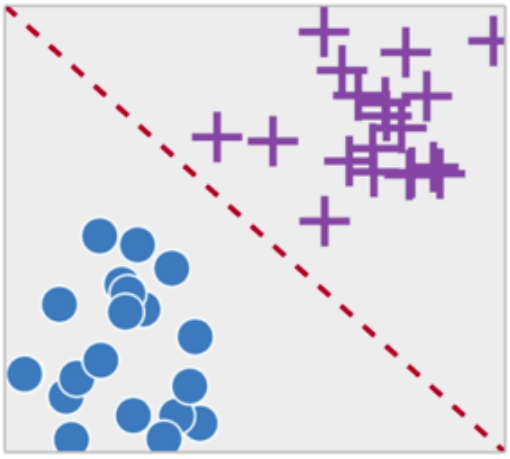

Рис 1.4. Класифікація

Кластеризація

В даному випадку навчальна вибірка складається з набору вхідних даних X без будь-яких відповідних їм значень. Метою подібних завдань може бути визначення груп схожих елементів всередині даних.

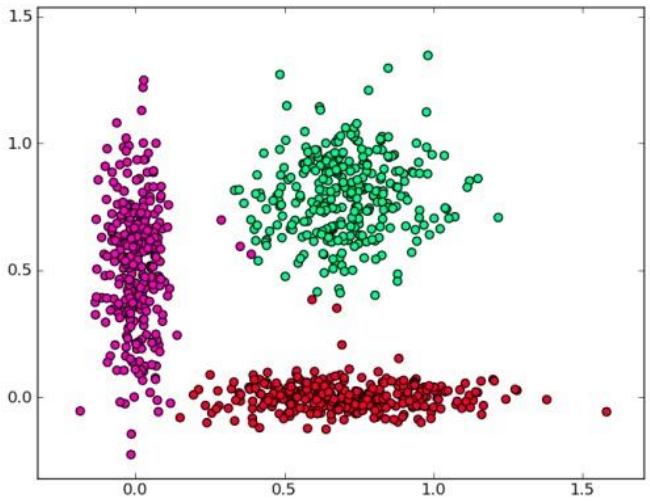

Рис 1.5. Кластеризація

Регресійний аналіз

Якщо бажаний вихідний результат складається з одного або більше безперервних змінних, тоді ми зіткаємося з регресійним аналізом. Прикладом вирішення такого завдання може служити передбачення довжини лосося як результату функції від його віку і ваги.

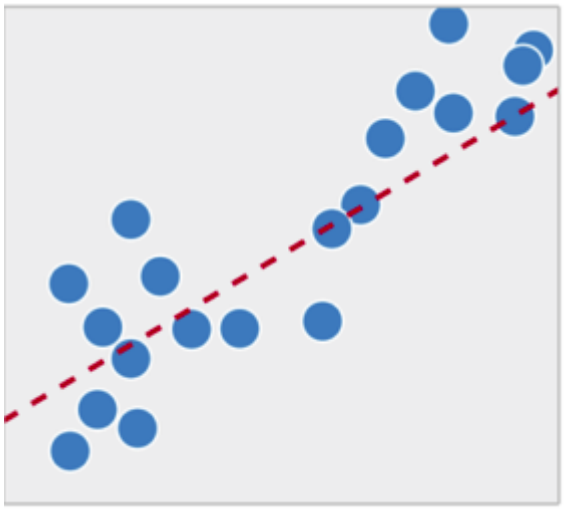

Рис 1.6. Регрессія

РОЗДІЛ 2

МАШИННЕ НАВЧАННЯ НА МОВІ ПРОГРАМУВАННЯ PYTHON

Python став загальноприйнятою мовою для багатьох сфер застосування науки про дані (data science). Він поєднує в собі міць мов програмування з простотою використання предметно орієнтованих скриптових мов типу MATLAB або R. В Python є бібліотеки для завантаження даних, візуалізації, статистичних обчислень, обробки природної мови, обробки зображень і багато чого іншого. Цей великий набір інструментів пропонує фахівцям з роботи з даними (data scientists) великий набір інструментів загального і спеціального призначення. Одним з основних переваг використання Python є можливість безпосередньо працювати з програмним кодом за допомогою терміналу або інших інструментів типу Jupyter Notebook, який ми розглянемо нижче. Машинне навчання та аналіз даних - це в основному ітераційні процеси, в яких дані задають хід аналізу. Вкрай важливо для цих процесів мати інструменти, які дозволяють оперативно і легко працювати. В якості мови програмування загального призначення Python дозволяє створювати складні графічні інтерфейси (GUI) і веб-сервіси, а також легко інтегруватися в уже існуючі системи.

2.1 Бібліотеки та інструменти на мові Python

scikit-learn

Проект з відкритим вихідним кодом, це означає, що його можна вільно використовувати і поширювати, і будь-яка людина може легко отримати вихідний код, щоб побачити, що відбувається «за лаштунками ». Проект scikit-learn постійно розвивається і вдосконалюється, і у нього

дуже активна спільнота користувачів. Він містить ряд сучасних алгоритмів машинного навчання, а також повну документацію по кожному алгоритму. scikit-learn — дуже популярний інструмент і найвідоміша пітонівска бібліотека для машинного навчання.

Вона широко використовується в промисловості і науці, а в інтернеті є багатий вибір навчальних матеріалів і прикладів програмного коду. scikit-learn прекрасно працює з низкою інших наукових інструментів Python.

Anaconda

Дистрибутив Python, призначений для великомасштабної обробки даних, прогнозної аналітики і наукових обчислень. Anaconda вже включає NumPy, SciPy, matplotlib, pandas, Ipython, Jupyter Notebook і scikit-learn. Є версії для Mac OS, Windows і Linux. Це дуже зручне рішення для користувачів, у яких ще не встановлені пакети Python для наукових обчислень. Крім того, зараз Anaconda включає в себе комерційну бібліотеку Intel MKL, якою можна користуватися безкоштовно. Використання MKL (це відбувається автоматично при установці Anaconda) може дати значний приріст швидкості при виконанні різних алгоритмів в scikitlearn.

Jupyter Notebook

Інтерактивна середа для запуску програмного коду в браузері. Це відмінний інструмент для розвідувального аналізу даних і широко використовується фахівцями з аналізу даних. Незважаючи на те що Jupyter Notebook підтримує безліч мов програмування, нам потрібна лише підтримка Python. Jupyter Notebook дозволяє легко інтегрувати програмний код, текст і зображенняю

NumPy

Один з основних пакетів для наукових обчислень в Python. Він містить функціональні можливості для роботи з багатовимірними масивами, високорівневими математичними функціями (операції лінійної

алгебри, перетворення Фур'є, генератор псевдовипадкових чисел). У scikit-learn масив NumPy - це основна структура даних. scikit-learn приймає дані у вигляді масивів NumPy. Будь-які дані, які ви використовуєте, повинні бути перетворені в масив NumPy. Базовий функціонал NumPy - це клас ndarray, багатовимірний (n-мірний) масив. Всі елементи масиву повинні бути одного і того ж типу. Масив NumPy виглядає наступним чином:

```
In[2]:
import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6]])
print("x:\n{}".format(x))

Out[2]:
x:
[[1 2 3]
 [4 5 6]]
```

Рис 2.1. Формування вибірки в numpy

SciPy

Це набір функцій для наукових обчислень в Python. Крім усього іншого він пропонує процедури лінійної алгебри, математичну оптимізацію функцій, обробку сигналів, спеціальні математичні функції і статистичні функції. scikit-learn використовує набір функцій SciPy для реалізації своїх алгоритмів. Найбільш важливою частиною SciPy є пакет scipy.sparse: за допомогою нього можна отримати розріджені матриці(Sparse matrices), які представляють собою ще один формат даних, який використовується в scikit-learn. Розріджені матриці використовуються всякий раз, коли нам потрібно зберегти 2D масив, який містить в основному нулі:

```
In[3]:
from scipy import sparse

# Создаем 2D массив NumPy с единицами по главной диагонали и нулями в остальных ячейках
eye = np.eye(4)
print("массив NumPy:\n{}".format(eye))

Out[3]:
массив NumPy:
[[ 1. 0. 0. 0.]
 [ 0. 1. 0. 0.]
 [ 0. 0. 1. 0.]
 [ 0. 0. 0. 1.]]
```

Рис 2.2. SciPy матриця

pandas

Бібліотека Python для обробки і аналізу даних. Вона побудована на основі структури даних, званої DataFrame і змодельованої за принципом датафреймов середовища статистичного програмування R. Простіше кажучи, DataFrame бібліотеки pandas являє собою таблицю, схожу на електронну таблицю Excel. Бібліотека pandas пропонує великий спектр методів по роботі з цією таблицею, зокрема, вона дозволяє виконувати SQL-подібні запити і приєднання таблиць. На відміну від NumPy, який вимагає, щоб всі записи в масиві були одного і того ж типу, в pandas кожен стовпець може мати окремий тип (наприклад, цілі числа, дати, числа з плаваючою точкою і рядки). Ще однією перевагою бібліотеки pandas є її здатність працювати з різними форматами файлів і баз даних, наприклад, з файлами SQL, Excel і CSV.

keras

Відкрита нейромережева бібліотека, написана мовою Python. Вона здатна працювати поверх Deeplearning4j, TensorFlow та Theano. Спроектовану для уможливлення швидких експериментів з мережами глибинного навчання, її зосереджено на тому, щоби вона була мінімальною, модульною та розширюваною. Її було створено як частину дослідницьких зусиль проекту ONEIROS (англ. Open-ended Neuro-Electronic Intelligent Robot Operating System), а її основним автором та підтримувачем є Франсуа Шоллє (фр. François Chollet), інженер Google.

І хоч команда TensorFlow Google й вирішила підтримувати Keras в основній бібліотеці TensorFlow, Шоллє сказав, що Keras було замислено радше як інтерфейс, аніж як наскрізну систему машинного навчання. Вона представляє високорівневий, інтуїтивніший набір абстракцій, який робить простим формування нейронних мереж незалежно від тилової бібліотеки наукових обчислень. Microsoft працює над доданням до Keras і тилу CNTK.

TensorFlow

Відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди. Її наразі застосовують як для досліджень, так і для розробки продуктів Google, часто замінюючи на його ролі її закритого попередника, DistBelief. TensorFlow було початково розроблено командою Google Brain для внутрішнього використання в Google, поки її не було випущено під відкритою ліцензією Apache 2.0 9 листопада 2015 року.

Theano

Бібліотека чисельного обчислення для Python. Обчислення в Theano виражаються NumPy-ським синтаксисом і компілюються для ефективного виконання на архітектурі або ЦП, або ГП.

Theano є відкритим проектом, основним розробником якого є група машинного навчання в Монреальському університеті.

2.2 Застосування фреймворків для машинного навчання

У цьому розділі ми розглянемо простий приклад застосування машинного навчання і побудуємо нашу першу модель.

Припустимо, що ботанік-любитель хоче класифікувати сорти ірисів, які він зібрав. Він виміряв в сантиметрах деякі характеристики ірисів: довжину і ширину пелюсток, а також довжину і ширину чашолистків

Крім того, у нього є вимірювання цих же характеристик ірисів, які раніше дозволили досвідченому експерту віднести їх до сортам setosa, versicolor і virginica. Щодо цих ірисів ботанік-любитель впевнено може сказати, до якого сорту належить кожен ірис. Давайте припустимо, що

перераховані сорти є єдиними сортами, які ботанік-любитель може зустріти в дикій природі.

Наша мета полягає в побудові моделі машинного навчання, яка зможе навчитися на основі характеристик ірисів, вже класифікованих за сортами, і потім передбачить сорт для нової квітки ірису. Оскільки у нас є приклади, за якими ми вже знаємо правильні сорту ірису, можна вирішити завдання є завданням навчання з учителем. В цьому завданні нам потрібно спрогнозувати один із сортів ірису. Це приклад завдання класифікації (classification). Можливі відповіді (Різні сорти ірису) називаються класами (classes). Кожен ірис в наборі даних належить до одного з трьох класів, таким чином завдання є задачею трикласової класифікації.

Відповіддю для окремої точки даних (ірису) є той чи інший сорт цієї квітки. Сорт, до якого належить квітка (конкретна точка даних), називається міткою (label).

Дані, які ми будемо використовувати для цього прикладу, - це набір даних Iris, класичний набір даних в машинному навчанні та статистикою. Він уже включений в модуль datasets бібліотеки scikit-learn. Ми можемо завантажити його, викликавши функцію load_iris.

```
In[10]:
from sklearn.datasets import load_iris
iris_dataset = load_iris()
```

Рис 2.3. Завантаження даних для прикладу

Об'єкт iris, що повертається load_iris, є об'єктом Bunch, який дуже схожий на словник. Він містить ключі і значення:

```
In[11]:
print("Ключи iris_dataset: \n{}".format(iris_dataset.keys()))

Out[11]:
Ключи iris_dataset:
dict_keys(['target_names', 'feature_names', 'DESCR', 'data', 'target'])
```

Рис 2.4. Ключі набору даних

Значення ключа DESCR - це короткий опис набору даних. Тут ми покажемо початок опису:

```
In[12]:
print(iris_dataset['DESCR'][:193] + "\n...")

Out[12]:
Iris Plants Database
====================
Notes
----
Data Set Characteristics:
:Number of Instances: 150 (50 in each of three classes)
:Number of Attributes: 4 numeric, predictive att
```

Рис 2.5. Короткий опис набору даних

Значення ключа target_names – це масив рядків, що містить сорти квітів, які ми хочемо передбачити:

```
In[13]:
print("Названия ответов: {}".format(iris_dataset['target_names']))

Out[13]:
Названия ответов: ['setosa' 'versicolor' 'virginica']
```

Рис.2.6. Назви класів

Значення feature_names - це список ланцюжків з описом кожної ознаки:

```
In[14]:
print("Названия признаков: \n{}".format(iris_dataset['feature_names']))

Out[14]:
Названия признаков:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
 'petal width (cm)']
```

Рис 2.7. Назви ознак

Самі дані записані в масивах target і data. data — масив NumPy, який містить кількісні вимірювання довжини чашолистків, ширини чашолистків, довжини пелюсток і ширини пелюсток:

```
In[15]:
print("Тип массива data: {}".format(type(iris_dataset['data'])))

Out[15]:
Тип массива data: <class 'numpy.ndarray'>
```

Рис 2.8. Тип масиву

Рядки в масиві data відповідають квітам ірису, а стовпціявляють собою чотири ознаки, які були виміряні для кожної квітки:

```
In[16]:
print("Форма массива data: {}".format(iris_dataset['data'].shape))

Out[16]:
Форма массива data: (150, 4)
```

Рис 2.9. Форма масиву

Ми бачимо, що масив містить вимірювання для 150 різних квітів по 4 ознаками. Згадаймо, що в машинному навчанні окремі елементи називаються прикладами (samples), а їх властивості — 30 характеристиками або ознаками (feature). Форма (shape) масиву даних визначається кількістю прикладів, помноженим на кількість ознак. Це є загальноприйнятою угодою в scikit-learn, і ваші дані завжди будуть представлені в цій формі. Нижче наведені значення ознак для перших п'яти прикладів:

```
In[17]:
print("Первые пять строк массива data:\n{}".format(iris_dataset['data'][:5]))

Out[17]:
Первые пять строк массива data:
[[ 5.1  3.5  1.4  0.2]
 [ 4.9  3.   1.4  0.2]
 [ 4.7  3.2  1.3  0.2]
 [ 4.6  3.1  1.5  0.2]
 [ 5.   3.6  1.4  0.2]]
```

Рис 2.10. Голова масиву

Поглянувши на ці дані, ми бачимо, що всі п'ять квіток мають ширину пелюстки 0.2 см і перша квітка має найбільшу довжину чашолистка, 5.1 см. Масив target містить сорти вже виміряних квітів, теж записані у вигляді масиву NumPy:

```
In[18]:
print("Тип массива target: {}".format(type(iris_dataset['target'])))

Out[18]:
Тип массива target: <class 'numpy.ndarray'>
```

Рис 2.11. Тип масиву класів

target це одновимірний масив, по одному елементу для кожної квітки:

```
In[19]:
print("Форма массива target: {}".format(iris_dataset['target'].shape))

Out[19]:
Форма массива target: (150,)
```

Рис 2.12. Форма масиву класів

Сорти кодуються як цілі числа від 0 до 2

```
In[20]:
print("Ответы:\n{}".format(iris_dataset['target']))

Out[20]:

Ответы:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Рис 2.13. Класи масиву відповідей

Значення чисел задаються масивом iris ['target_names']: 0 — setosa, 1 - versicolor, а 2 — virginica.

Вимірення ефективності: навчаючий і тестовий набори.

На основі цих даних нам потрібно побудувати модель машинного навчання, яка передбачить сорти ірису для нового набору вимірювань. Але перш ніж застосувати нашу модель до нового набору, ми повинні переконатися в тому, що модель насправді працює і її прогнозами можна довіряти. На жаль, для оцінки якості моделі ми не можемо використовувати дані, які були взяті нами для побудови моделі. Це обумовлено тим, що наша модель просто запам'ятає всю навчальну вибірку і тому вона завжди буде передбачати правильну мітку для будь-якої точки даних в навчальному наборі. Це «запам'ятовування» нічого не говорить нам про узагальнюючої здатності моделі (іншими словами, ми не знаємо, чи буде ця модель так само добре працювати на нових даних). Для оцінки ефективності моделі, ми пред'являємо їй нові розмічені дані (розмічені дані, які вона не бачила раніше). Зазвичай це робиться шляхом розбиття зібраних розмічених даних (в даному випадку 150 квітів) на дві частини. Одна частина даних використовується для побудови нашої моделі машинного навчання і називається навчальними даними (training data) або навчальним набором (training set). Інші дані будуть використані для оцінки якості моделі, їх називають тестовими даними (test data), тестовим набором (test set) або контрольним набором (hold-out set). У бібліотеці scikit-learn є функція train_test_split, яка перемішує набір даних

і розбиває його на дві частини. ця функція відбирає в навчальний набір 75% рядків даних з відповідними мітками. Решта 25% даних з мітками оголошуються тестовим набором. Питання про те, скільки даних відбирати в навчальний і тестовий набори, є дискусійним, проте використання тестового набору, що містить 25% даних, є хорошим правилом.

У scikit-learn дані, як правило, позначаються заголовною X, тоді як мітки позначаються маленькою y. Це навіяно стандартною математичною формулою f (x) = y, де x є аргументом функції, а y - висновком. Відповідно до деяких математичних угод ми використовуємо заголовну X, тому що дані являють собою двовимірний масив (матрицю) і малу y, тому що цільова змінна - це одновимірний масив (вектор). Давайте викличемо функцію train_test_split для наших даних і задамо навчальні дані, навчальні мітки, тестові дані, тестові мітки, використовуючи вищезгадані літери:

```
In[21]:
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    iris_dataset['data'], iris_dataset['target'], random_state=0)
```

Рис 2.14. Розподіл даних на навчальні і тестові вибірки

Перед розбиттям функція train_test_split перемішує набір даних за допомогою генератора псевдовипадкових чисел. Якщо ми просто візьмемо останні 25% спостережень в якості тестового набору, всі точки даних будуть мати позначку 2, оскільки всі точки данихвідсортовані по мітках. Використовуючи тестовий набір, який містить лише один з трьох класів, ви не зможете об'єктивно судити про узагальнюючої здатності моделі, таким чином, ми перемішуємо наші дані, щоб тестові дані містили всі три класи.

Щоб в точності повторно відтворити отриманий результат, ми скористаємося генератором псевдовипадкових чисел з фіксованим стартовим значенням, яке задається за допомогою параметра random_state. Це дозволить зробити результат відтворюємо, тому

вищенаведений програмний код буде генерувати один і той же результат. Ми завжди будемо ставити random_state при використанні рандомізованих процедур в цій книзі.

Висновком функції train_test_split є X_train, X_test, y_train і y_test, які всі є масивами Numpy. X_train містить 75% рядків набору даних, а X_test містить решта 25%:

```
In[22]:
print("форма массива X_train: {}".format(X_train.shape))
print("форма массива y_train: {}".format(y_train.shape))

Out[22]:
форма массива X_train: (112, 4)
форма массива y_train: (112,)

In[23]:
print("форма массива X_test: {}".format(X_test.shape))
print("форма массива y_test: {}".format(y_test.shape))

Out[23]:
форма массива X_test: (38, 4)
форма массива y_test: (38,)
```

Рис 2.15. Форма навчальних та тестових виборок

Перед тим як будувати модель машинного навчання, непогано було б досліджувати дані, щоб зрозуміти, чи можна легко вирішити поставлену завдання без машинного навчання або міститься потрібна інформація в даних. Крім того, дослідження даних - це хороший спосіб виявити аномалії і особливості. Наприклад, цілком можливо, що деякі з ваших ірисів виміряні в дюймах, а не в сантиметрах. У реальному світі нестиковки в даних і несподіванки дуже поширені.

Один з кращих способів досліджувати дані - візуалізувати їх.
Це можна зробити, використовуючи діаграму розсіювання (scatter plot). В діаграмі розсіювання одна ознака відкладається по осі x, а інший ознака - по осі y, кожному спостереженню відповідає точка. На жаль, екран комп'ютера мають тільки два виміри, що дозволяє розмістити на графіку тільки два (або, можливо, три) ознаки одночасно. Таким чином, важко розмістити на графіку набори даних з більш ніж трьома ознаками. Один із способів вирішення цієї проблеми - побудувати матрицю діаграм

розсіювання (Scatterplot matrix) або парні діаграми розсіювання (pair plots), на яких будуть зображені всі можливі пари ознак.

```
In[24]:
# создаем dataframe из данных в массиве X_train
# маркируем столбцы, используя строки в iris_dataset.feature_names
iris_dataframe = pd.DataFrame(X_train, columns=iris_dataset.feature_names)
# создаем матрицу рассеяния из dataframe, цвет точек задаем с помощью y_train
grr = pd.scatter_matrix(iris_dataframe, c=y_train, figsize=(15, 15), marker='o',
                        hist_kwds={'bins': 20}, s=60, alpha=.8, cmap=mglearn.cm3)
```

Рис 2.16. Будування матриці розсіювання з DataFrame

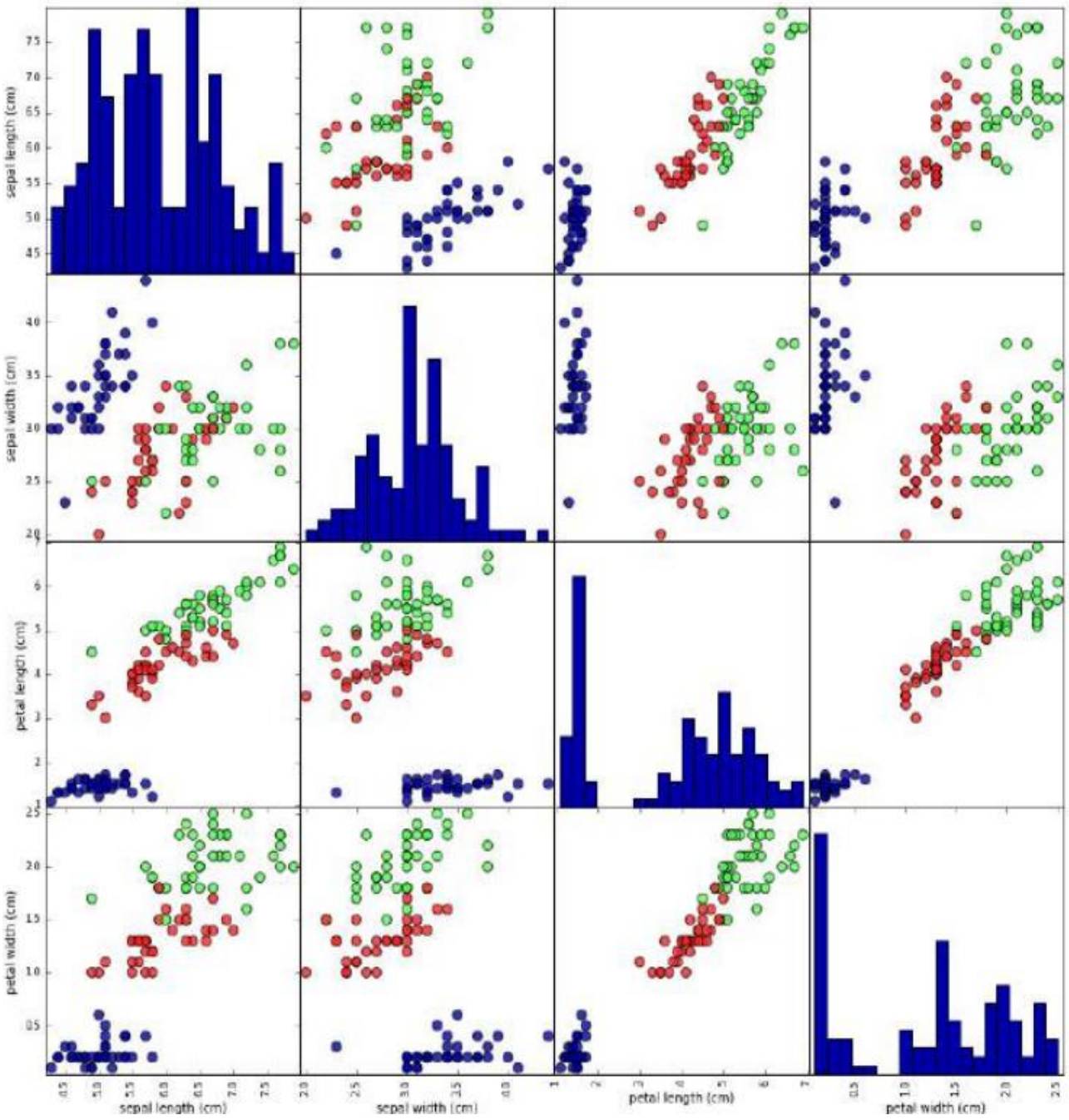

Рис 2.17. Матриці діаграм розсіювання

Рис. 2.17 представляє собою матрицю діаграм розсіювання для ознак навчального набору. Точки даних пофарбовані відповідно з сортами ірису, до яких вони належать. Щоб побудувати діаграми, ми спочатку перетворюємо масив NumPy в DataFrame (основний тип даних в бібліотеці pandas). У pandas є функція для створення парних діаграм розсіювання під назвою scatter_matrix. За діагоналі цієї матриці розташовуються гістограми кожної ознаки:

Поглянувши на графік, ми можемо побачити, що, схоже, вимірювання чашолистків і пелюсток дозволяють відносно добре розділити три класи. Це означає, що модель машинного навчання, ймовірно, зможе навчитися розділяти їх

Будування моделі: метод k ближчих сусідів.

Тепер ми можемо почати будувати реальну модель машинного навчання. У бібліотеці scikit-learn є досить багато алгоритмів класифікації, які ми могли б використовувати для побудови моделі. В даному прикладі ми будемо використовувати класифікатор на основі методу k найближчих сусідів, який легко інтерпретувати. Побудова цієї моделі полягає лише в запам'ятовуванні навчального набору. Для того, щоб зробити прогноз для нової точки даних, алгоритм знаходить точку в навчальному наборі, яка знаходиться ближче всього до нової точки. Потім він привласнює мітку, що належить цій точці навчального набору, нової точці даних.

k в методі k найближчих сусідів означає, що замість того, щоб використовувати лише найближчого сусіда нової точки даних, ми в ході навчання можемо розглянути будь-яку фіксовану кількість (k) сусідів (Наприклад, розглянути найближчі три або п'ять сусідів). Тоді ми можемо зробити прогноз для точки даних, використовуючи клас, якому належить більшість її сусідів. Детальніше ми поговоримо про це в розділі 2, а в даний момент ми будемо використовувати тільки одного сусіда.У scikit-learn всі моделі машинного навчання реалізовані в власних класах, які називаються класами Estimator. Алгоритм класифікації на основі методу k найближчих сусідів реалізований в класифікаторі KNeighborsClassifier модуля neighbors. Перш ніж використовувати цю модель, потрібно створити об'єкт-екземпляр класу. Це станеться, коли ми поставимо параметри моделі. Найважливішим параметром KNeighborsClassifier є кількість сусідів, які становитимуть 1:

```
In[25]:
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```

Рис 2.18. Будування моделі К-ближчих сусідів

Об'єкт knn включає в себе алгоритм, який буде використовуватися для побудови моделі на навчальних даних, а також алгоритм, який згенерує прогнози для нових точок даних. Він також буде містити інформацію, яку алгоритм витягнув з навчальних даних. У випадку з KNeighborsClassifier він буде просто зберігати навчальний набір.

Для побудови моделі на навчальному наборі, ми викликаємо метод fit об'єкта knn, який приймає в якості аргументів масив NumPy X_train, що містить навчальні дані, і масив NumPy y_train, відповідний навчальним міткам:

```
In[26]:
knn.fit(X_train, y_train)

Out[26]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

Рис 2.19. Навчання моделі К-ближчих сусідів

Метод fit повертає сам об'єкт knn (і змінює його), таким чином, ми отримуємо строкове представлення нашого класифікатора. Воно показує нам, які параметри були використані при створенні моделі. Майже всі параметри мають значення за замовчуванням, але ви також можете виявити параметр n_neighbor = 1, заданий нами. Більшість моделей в scikit-learn мають масу параметрів, але велика частина з них пов'язана з оптимізацією швидкодії обчислень або призначена для особливих випадків використання.

Отримання прогнозів.

Тепер ми можемо отримати прогнози, застосувавши цю модель до нових даними, за якими ми ще не знаємо правильні мітки. Досить уявіти, знахідку в дикій природі - ірис з довжиною чашолистка 5 см, шириною чашолистка 2.9 см, довжиною пелюстки 1 см і шириною пелюстки 0.2 см. До якого сорту ірису потрібно віднести цю квітку? Ми можемо помістити

ці дані в масив NumPy, знову обчислюючи форму масиву, тобто кількість прикладів (1), помножене на кількість ознак (4).

```
In[27]:
X_new = np.array([[5, 2.9, 1, 0.2]])
print("форма массива X_new: {}".format(X_new.shape))

Out[27]:
форма массива X_new: (1, 4)
```

Рис 2.20. Будування масива ознак для прогнозування

Слід зауважити, що вимірювання були записані по одній квітці в двовимірний масив NumPy, оскільки scikit-learn працює з двовимірними масивами даних. Щоб зробити прогноз, викликається метод predict об'єкта knn

```
In[28]:
prediction = knn.predict(X_new)
print("Прогноз: {}".format(prediction))
print("Спрогнозированная метка: {}".format(
       iris_dataset['target_names'][prediction]))

Out[28]:
Прогноз: [0]
Спрогнозированная метка: ['setosa']
```

Рис 2.21. Прогнозування за ознаками

Наша модель передбачає, що цей новий квітка ірису належить до класу 0, що означає сорт setosa. Але не факт що це є правильною відповіддю, тому-що невідома точність моделі.

Оцінка якості моделі

Це той самий момент, коли нам знадобиться створений раніше тестовий набір. Ці дані не використовувалися для побудови моделі, але ми знаємо правильні сорти для кожного ірису в тестовому наборі. Таким чином, ми можемо зробити прогноз для кожного ірису в тестовому наборі і порівняти його з фактичної міткою (вже відомим сортом). Ми можемо оцінити якість моделі, обчисливши правильність (Accuracy) - відсоток квітів, для яких модель правильно спрогнозувала сорти:

```
In[29]:
y_pred = knn.predict(X_test)
print("Прогнозы для тестового набора:\n {}".format(y_pred))

Out[29]:
Прогнозы для тестового набора:
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0 2]

In[30]:
print("Правильность на тестовом наборе: {:.2f}".format(np.mean(y_pred == y_test)))

Out[30]:
Правильность на тестовом наборе: 0.97
```

Рис 2.22. Перевірка точності моделі через прогнозування тестової вибірки

Правильність цієї моделі для тестового набору становить близько 0.97, що означає, що було дано правильний прогноз для 97% ірисів в тестовому наборі. При деяких математичних припущеннях, це означає, що ми можемо очікувати, що наша модель в 97% випадків дасть правильний прогноз для нових ірисів. Для нашого ботаніка-любителя цей високий рівень правильності означає, що наша модель може бути досить надійною у використанні.

РОЗДІЛ 3

ЗАСОБИ БУДУВАННЯ ДОДАТКІВ НА МОВІ ПРОГРАМУВАННЯ PYTHON

3.1 Фреймворк

Фреймворки полегшують життя розробників, пропонуючи їм структуру для розробки додатків. Вони автоматизують реалізацію загальних рішень, скорочуючи час розробки та дозволяючи розробникам зосередитись на логіці програми а не на рутинних елементах.

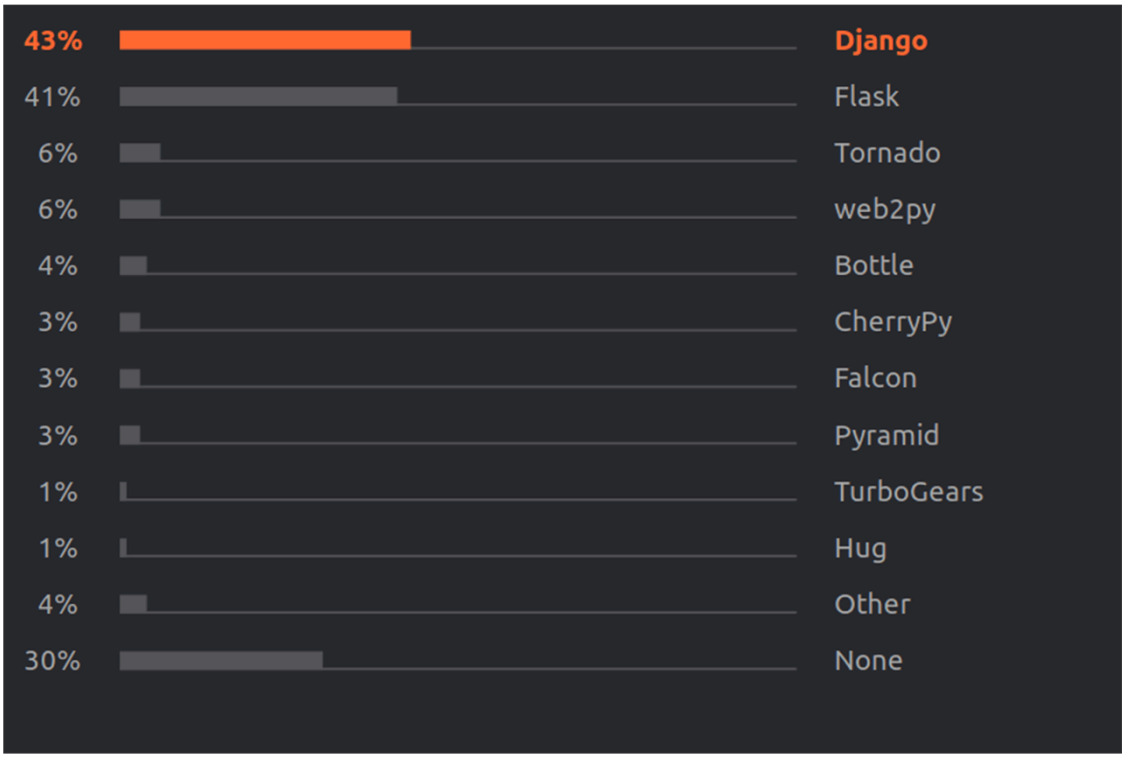

Рис 3.1. Рейтинг веб-фреймворків

Деякі речі, які слід врахувати:

- По-перше, вирішуючи, який фреймворк використовувати, треба звернути увагу на розмір та складність проекту. Якщо те, що ви хочете розробити - це велика система, в якій є конкретні функції та вимоги, fullstack фреймворк може бути правильним вибором. Якщо ваш додаток має менші та простіші перспективи, то краще розглядати мікрофреймворк.

- По-друге, потрібно перевірити, чи може фреймворк, масштабуватися вертикально і горизонтально. Це необхідно для проектів, які мають працювати на декількох серверах, обробляти величезну кількість трафіку та підтримувати додавання нових функцій для підвищення функціональності.

Однак фреймворки також можуть стати на шляху розвитку. Вибираючи fullstack фреймворк, залишається тільки згодитись з набором обмежень. Звичайно, можна знайти спосіб подолати їх, але іноді краще не витрачати більше часу на боротьбу за власну свободу, ніж писати додаток на чистому Python, або більш низкорівневому фреймворці.

Fullstack фреймворки

Fullstack фреймворки або enterprise фреймворки - це всебічні рішення з бібліотеками, налаштованими на безперебійну роботу разом. Вони підтримують розвиток сервісів, що надаються, фронт-енд інтерфейсу та баз даних. Fullstack фреймворк забезпечує все, що розробнику потрібно для створення програми. Python пропонує більше, ніж один fullstack фреймворк.

Django

Django - це безкоштовна система з відкритим кодом з fullstack фреймворк на Python. Вона намагається включити всі необхідні функції «з коробки», а не пропонувати їх як окремі бібліотеки.

Деякі з прикладних особливостей Django - це аутентифікація, маршрутизація URL-адресів, шаблонізатор, об'єктно-реляційний маппер (ORM) та міграція схеми бази даних (Django v.1.7 +).

Ці функції роблять Django легко масштабуючою, швидкою у розробці та надзвичайно універсальною системою.

Django використовує ORM для відображення об'єктів у таблиці баз даних. Один і той же код працює з різними базами даних і зовсім не важко перенести дані з однієї бази даних в іншу. Основні бази даних, з якими

працює Django, - це PostgreSQL, MySQL, SQLite та Oracle, але сторонні драйвери дозволяють також використовувати й інші.

Завдяки Django можна створювати будь-які веб-програми від невеликих проектів до складних веб-сайтів. Завдяки своїй гнучкості Django також використовується для створення MVP, дозволяючи стартапам оптимізувати свій час та бюджет.

Очікується, що користувальницька база Django зростатиме, оскільки багато розробників вважають цю основу найкращим вибором для нових технологій, таких як машинне навчання. Також у 2020 році громада Джанго планує випустити більше бібліотек для задоволення зростаючого попиту.

Pyramid

Піраміда другий по важливості Python фреймворк з відкритим кодом. Його мета - зробити якомога більше з мінімальною складністю. Працюючи на Python 3, Pyramid не відстає від технологічних удосконалень. Піраміда 1.10 - це поточна версія фреймворку і є десятим оновленням з 2010 року.

Найяскравішою особливістю Піраміди є її здатність добре працювати як з малими, так і з великими програмами. Деякі з найкращих функцій Pyramid включають:

- Однофайлові програми
- Генерація URL-адресів
- Розширювані конфігурації
- Різноманітні специфікації шаблонів та “ассетів”
- Гнучка аутентифікація та авторизація
- Тестування, підтримка та достатня документація
- Декоратори
- Предикати
- Рендери

За допомогою Pyramid розробник може сам вибрати щаблонізатор, загальні бібліотеки та інструменти для баз даних.

TurboGears

TurboGears - це орієнотований на дані fullstack фреймворк з відкритим кодом,. Він побудований на безлічі проміжного програмного забезпечення та багатьох бібліотек і спочатку був створений для поєднання найкращих компонентів інших фреймворків Python.

TurboGears дозволяє швидко розробляти розширювані веб-програми, орієнтовані на дані. Він оснащений зручним для користувача шаблонізатором і потужним і гнучким ORM. На додаток до чудових шаблонізаторів, що полегшують життя дизайнерів, TurboGears пропонує багато гнучкості, сильну підтримку агрегацій, потужний ORM та фрагменти багаторазового використання.

Деякі особливості TurboGears включають:

- Підтримка багатьох баз даних
- Архітектура в стилі MVC
- Підтримка SQLObject і SQLAlchemy
- Валідація за допомогою FormEncode
- Pylons як веб-сервер
- ToscaWidgets - бібліотека додатків, яка спрощує взаємодію фронт-енду та бек-енду.
- Шаблони PasteScript
- Інструменти терміналу
- Інтеграція MochiKit - JavaScript бібліотеки
- Усі особливості, реалізовані як декоратори функцій

Web2py

Web2py - це масштабований Python fullstack фреймворк з відкритим кодом. Він дуже потужний, коли справа стосується обробки даних.

Web2py спочатку пропонувався як навчальний інструмент з акцентом на простоті використання. Це пояснює, чому не існує файлів конфігурації на рівні проекту.

Хоча в Web2py чудово те, що він оснащений власним веб-IDE, який, серед іншого, включає редактор коду, налагоджувач та розгортання одним кліком.

Інші цінні функції Web2py включають:

- Немає вимог до встановлення та налаштування
- Можливість роботи на Windows, Mac, Linux / Unix, Google App Engine, Amazon EC2 та будь-який веб-хостинг, який підтримує або Python 2.5–2.7, або Java + Python
- Захист даних, що запобігає таких уразливостей, як міжсайтове скриптування, ін'єкції в базу даних та виконання шкідливих файлів
- Успішне використання методів інженерії програмного забезпечення, що робить код легким для читання та обслуговування
- Відстеження помилок, через журнал помилок та «квитки»
- Доступ оснований на ролях
- Підтримка інтернаціоналізації
- Зворотна сумісність, що забезпечує оновлення, орієнтоване на користувачів, без необхідності втрачати зв’язок з попередніми версіями

Система квитків - головна особливість Web2py; він видає квитки, коли виникають помилки. Таким чином, користувачі можуть відстежувати помилки та їхні статуси.

Незважаючи на всі ці переваги, спільнота Web2py є меншою, ніж Піраміда чи Джанго. Не багато розробників використовують його. Це означає, що шанси отримати підтримку нижчі.

Мікрофреймворки

Мікрофреймворкам або мінімалістичним фреймворкам веб-додатків не вистачає більшості функціональних можливостей повноцінних

фреймворків, таких як шаблонізатор, функція аутентифікації, акаунти, авторизація, перевірка введення даних та валідація вводу. Мікрофреймворк намагається надати лише набір компонентів, необхідний для створення програми. Це також може допомогти зосередитись на забезпеченні функціоналу для однієї конкретної сфери**.**

Flask

Flask - це Python фреймворк, доступний за ліцензією BSD. Основна ідея Flask - допомогти створити міцну основу веб-додатків. Тут можна використовувати будь-які розширення, які можуть знадобитися. Flask вибирається для будь-яких проектів. Насправді це стандартний вибір для будь-якого веб-проекту, для якого не відповідає Django.

Легкий і модульний дизайн Flask дозволяє легко адаптуватися до потреб розробників. Він включає ряд корисних функцій, які не випускаються з коробки:

- Вбудований сервер розробки та швидкий налагоджувач
- Комплексна підтримка unit-тестів
- RESTful архітектура запитів
- Шаблонізатор Jinja2
- Підтримка безпечних cookie файлів (сеанси на стороні клієнта)
- Підтримка WSGI 1.0
- Оснований на Unicode
- Можливість підключення будь-якого ORM
- Обробка HTTP запитів

З моменту запуску в 2010 році Flask оновлювали 27 разів. На сьогоднішній день він залишається найбільш популярною основою Python. Однак багато розширень Flask більше не підтримуються: документація застаріла і більше не розробляється.

Bottle

Bottle - це мікрофреймворк. Спочатку призначений для створення API, весь в одному відкритому файлі. Він не має залежностей, крім стандартної бібліотеки Python. Програмування за допомогою Bottle більше зближує вас з внутрощами комп'ютера, ніж програмування з будь-якими fullstack фреймвораками. Однак, Bottle підходить лише в тому випадку, якщо ви створюєте дуже невелику програму, не більше 500 рядків коду та ніяких особливих вимог.

Його типовими функціями є маршрутизація, шаблонування, утиліти та основна абстракція над стандартом WSGI.

З 2009 року «Bottle» оновлювався 73 рази, і тепер це ідеальне рішення для складання прототипів, вивчення організації веб-фреймворків та створення простих особистих додатків.

CherryPy

CherryPy - це мінімалістичний веб-фреймворк з відкритим кодом. Він робить створення веб-додатків Python таким самим як створення будь-якої іншої об'єктно-орієнтованої програми.

CherryPy створений для постійного розширення. Фреймворк навіть пропонує механізми для підключення точок входження і розширень.

Насправді веб-додаток CherryPy працює на автономному додатку Python, який вбудовує власний багатопотоковий веб-сервер. Додатки CherryPy запускаються в будь-якій операційній системі, яка підтримує Python (Windows, macOS, Linux тощо). Їх можна розгорнути де завгодно, де можна запустити звичайну програму Python. Вам не потрібен Apache сервер для CherryPy, але ви можете запустити додаток CherryPy з під Apache так само добре, як ви можете запустити його з під Lighttpd або IIS.

CherryPy не обмежений фреймворк, оскільки дозволяє використовувати будь-який тип технології для шаблонування, доступу до даних тощо. Однак він сам все ще сможе обробляти сеанси, статику, файли cookie, завантаження файлів та все інше, що зазвичай може веб-

фреймворк.

Деякі функції CherryPy за замовчуванням включають:

- Веб-сервер WSGI, сумісний з HTTP / 1.1
- Простота запуску декількох серверів HTTP одночасно
- Потужна система конфігурації
- Гнучка система плагінів
- Нестандартні інструменти для кешування, кодування, сеансів, аутентифікації, статичного вмісту тощо
- Вбудована підтримка профілювання, покриття та тестування
- Можливість роботи на Python 2.7+, Python 3.1+, PyPy, Jython та Android

Незважаючи на всі його особливості та переваги, багато розробників вважають, що документації CherryPy необхідні певні вдосконалення. Також, CherryPy заскладний у використанні.

Falcon

Falcon - це Python API фреймворк для створення дуже швидких програм. Логотип рамки - сокіл, який символізує, наскільки швидко Falcon працює.

Цей фреймворк дозволяє розробникам розробляти більш чисті конструкції та обробляти більшість запитів. Falcon ніколи не обмежує розробників у виборі бібліотек для баз даних та авторизації.

Інші цінні функції Falcon:

- 100% покриття коду за допомогою всебічного тестового набору
- Високо оптимізована база коду
- Попередня обробка виключень
- Базовані на REST класи представлень
- Шаблони URI для інтуїтивно зрозумілих процедур
- Легкий доступ через класи запитів та відповідей

- DRY обробка запиту через компоненти програмного забезпечення
- Ідіоматичні відповіді на помилки HTTP
- Тестування блоку за допомогою помічників WSGI

Falcon ігнорує марні операції під час розробки веб-рамки. Крім того, це дає свободу вибору: розробники можуть приймати рішення, які допоможуть їм налаштовувати реалізацію вільно.

Ще дві переваги, які варто згадати, - це наявність великої документації та активне співтовариство.

Hug

Hug - це фреймворк розробки API на Python 3. Він створений, щоб дозволити інженерам програмного забезпечення розробити API один раз, а потім повторно використовувати його, коли це потрібно. Фреймворк Hug спрощує розробку API, пропонуючи різні інтерфейси. Це основна причина, по якій цей фреймворк називають одним із найшвидших на Python 3.

Основні особливості Hug:

- Підтримка автоматичної документації
- Валідація на основі анотацій
- Вбудоване управління версіями
- Сумісність з Cython
- Може використовуватися як бібліотека Python

Фреймворк Hug містить якомога менше інтеграцій та якомога менше коду, залишаючись функціональним та забезпечуючи велику продуктивність. Плюс, спільнота Hug зростає. Це ознака попиту на фреймворк та доказ того, що кожен раз, коли виникає проблема, розробник може знайти допомогу.

FastAPI

FastAPI - це сучасний фреймворк на основі Python. Спочатку він був розроблений так, щоб його було легко використовувати для розробки. На сьогоднішній день FastAPI - це одна з найшвидших рамок для створення API з Python 3.6+.

Фреймворк взяв від Flask декілька характеристик, включаючи його простоту. Весь фреймворк побудований на Starlette і включає більшість його функцій (шаблони, WebSockets та підтримка GraphQL).

Рамка FastAPI пропонує:

- Збільшення швидкості розробки на 200-300%
- Зменшення помилок на 40%
- Менше часу налагодження
- Мінімізація дублювання коду
- Автоматична інтерактивна документація

FastAPI не пов'язаний з жодною базою даних. Однак він підтримує необхідні бази даних. За допомогою FastAPI ви можете структурувати проект так, як вам потрібно, і навіть використовувати більшість однакових файлів з одного з ваших проектів на Bottle.

Єдиним недоліком є його невелика спільнота. Але це тому, що FastAPI є відносно новим фреймворком.

Асинхронні фреймворки

Асинхронний фреймворк є відносно недавнім типом рамки Python. Це мікрофреймворк, який дозволяє розробникам обробляти великий набір одночасних з'єднань. Асинхронні фреймворки використовують неблокуючі сокети та бібліотеку Python asyncio.

Sanic

Sanic - це веб-фреймворк на Python, побудований на uvloop та створений спеціально для швидких відповідей HTTP за допомогою асинхронної обробки запитів.

Він працює на Python 3.5+. Sanic підтримує асинхронні обробники запитів, що робить його сумісним з функціями асинхронізації/очікування Python. Це підвищує його швидкість, і не блокує додаток.

Sonic вважається кращим асинхронним фреймворком у світі Python завдяки маршрутизації, проміжному програмному забезпеченню, файлам cookie, версіонуванні, статичним файлам, схемам на основі класів та сокетам. Однак він не пропонує підтримку шаблонізаторів та баз даних.

Під час тесту на навантаження із одним процесом та 100 підключеннями, Sanic зміг обробити 33 342 запити в секунду із середньою затримкою 2,96 мс.

Торнадо

Tornado - це веб-фреймворк та асинхронна мережева бібліотека Python, яка фокусується на швидкості та здатності обробляти великі обсяги трафіку. Він використовує незахисну мережу вводу / виводу і вирішує проблему C10k (тобто, якщо правильно налаштовано, він може обробляти 10 000+ одночасних з'єднань).

Це робить його чудовим інструментом для створення додатків, що вимагають високої продуктивності та десятки тисяч одночасних користувачів.

Основні особливості Торнадо:

- Вбудована підтримка аутентифікації користувачів
- Сервіси в режимі реального часу
- Висока ефективність
- Шаблонізатор на основі Python
- Неблокуючий клієнт HTTP
- Впровадження сторонніх схем аутентифікації та авторизації (Google OpenID / OAuth, Facebook Login, Yahoo BBAuth, FriendFeed OpenID / OAuth, Twitter OAuth)
- Підтримка перекладу та локалізації

За рівнем популярності Торнадо стоїть десь між Джанго та Фласком. Якщо ви хочете щось написати на Flask чи на Django через їхні особливості та інструменти, але вам потрібна набагато краща продуктивність, тоді вам слід вибрати Торнадо.

Інше

Dash

Dash - це програма з відкритим кодом Python, що використовується для створення аналітичних веб-додатків. Це особливо добре для науковців з даних на Python, які не дуже знайомі з веб-розробкою.

Є дві основні переваги Dash:

- Це дозволяє створювати інтерактивні програми, використовуючи лише код Python.
- Це дозволяє просто використовувати потужність інструментів Python для маніпулювання даними.

Програми Dash - це веб-сервери, які запускають Flask і спілкуються з пакетами JSON через HTTP-запити. Їх інтерфейс надає компоненти з React.js.

Програми, розроблені за допомогою Dash, відображаються у веб-браузері та можуть бути розгорнуті на сервери. Це також означає, що програми Dash за своєю суттю є платформними та готовими для мобільних пристроїв.

Розробники Dash мають доступ до базового екземпляра Flask та всіх його настроюваних властивостей. Щоб розширити можливості додатків Dash, розробники також можуть використовувати багатий набір плагінів.

3.2 Допоміжне програмне забезпечення

Будь який непримітивний додаток крім безпосередньо самого себе зазвичай вимагає допоміжне програмне забезпечення, без якого додаток

не зможе працювати повноцінно або взагалі. Зазвичай, це забезпечення для зберігання даних або для спрощення розробки.

Системи управління базами даних

Незважаючи на те, що всі системи управління базами даних виконують одну і ту ж основну задачу (тобто дають можливість користувачам створювати, редагувати і отримувати доступ до інформації, що зберігається в базах даних), сам процес виконання цього завдання варіюється в широких межах. Крім того, функції і можливості кожної СУБД можуть істотно відрізнятися. Різні СУБД документовані по-різному: більш-менш ретельно. По-різному надається і технічна підтримка.

При порівнянні різних популярних баз даних, слід враховувати, чи зручна для користувача і масштабована дана конкретна СУБД, а також переконатися, що вона буде добре інтегруватися з іншими продуктами, які вже використовуються. Крім того, під час вибору слід взяти до уваги вартість системи і підтримки, що надається розробником.

Якщо мова йде про вибір СУБД для підприємства, то слід взяти до уваги можливість СУБД «рости» разом з розвитком організації. Малому бізнесу можуть знадобитися тільки базові функції і можливості, а також невелика кількість інформації, що розміщується в БД. Але вимоги можуть істотно зростати з плином часу, а перехід на іншу СУБД може стати проблемою.

Існує кілька популярних СУБД, як платних, так і безкоштовних, які можна рекомендовати для застосування в організації.

Таблиця 3.1

Порівняльна характеристика СУБД

№ п/п	СУБД	Тип	Розробник	Операційна система	Ліцензія	Початковий код	Стабільність	Популярність	Підтримка
1	Oracle Database	Мульти-модельна	Oracle Corporation	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX	Комерц.	Замкн.	+	+	Платна
2	MySQL	Реляційна	Oracle Corporation	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	GNU GPL и комерц.	Відкр.	+	+	Платна
3	SQL Server	Реляційна	Microsoft	Linux, Microsoft Windows	Комерц.	Замкн.	+	+	Безкошт.
4	PostgreSQL	Об'єктно-реляційна	PostgreSQL Global Development Group	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, macOS, HP-UX, QNX	Дозволяюча ліцензція	Відкр.	+	+	Платна
5	MongoDB	Документо-орієнтована	MongoDB Inc.	Linux, Microsoft Windows, Oracle Solaris,FreeBSD,macOS	GNU AGPL (СУБД) и Apache License (драйверы)	Відкр.	+	+	Платна
6	MariaDB	Реляційна	MariaDB Corporation Ab, MariaDB Foundation	Linux, Microsoft Windows, Oracle Solaris,FreeBSD,macOS	GNU GPL	Відкр.	-	-	Платна
7	DB2	Об'єктно-реляційна	IBM	Linux, Microsoft Windows, Oracle Solaris,FreeBSD,macOS	Пропріетарна EULA	Замкн.	+	-	Безкошт.
8	SAP HANA	Реляційна	SAP SE	Linux	Комерц.	Замкн.	+	-	Платна.
9	ЛИНТЕР	Реляційна	РЕЛЭКС	Linux,Microsoft Windows, Oracle Solaris,FreeBSD,macOS, QNX, ІНТРОС	Комерц.	Замкн.	+	-	Безкошт.
10	Ред База Данных	Реляційна	Ред Софт	Linux, Microsoft Windows, Oracle Solaris,FreeBSD,HP-UX	GNU GPL	Відкр.	+	-	Платна

Розглянемо найпопулярніші СУБД детальніше

MySQL

MySQL - одна з найпопулярніших баз даних для веб-додатків. Фактично, є стандартом для веб-серверів, які працюють під управлінням

операційної Linux. MySQL - це безкоштовний пакет програмного забезпечення, при цьому постійно випускаються нові версі, в яких розширюють функціональність та покращують безпеку. Існують спеціальні платформи, передбачені для комерційного використання. Безкоштовна версія більш сфокусована на швидкості і надійності, а не на повноті функціоналу, який може стати перевагою і недоліком - в залежності від сфери роботи.

Розробка та підтримка MySQL здійснює корпорація Oracle, отримавши права на торговельну марку разом із поглощеною Sun Microsystems, яка раніше приєднала шведську компанію MySQL AB. Продукт розповсюджується як під загальною ліцензією GNU, так і під власною комерційною ліцензією. Крім цього, розробники створюють функціональність за запитами ліцензованих користувачів. Саме завдяки таким запитам в самих ранніх версіях, з'явився механізм реплікації.

Ця СУБД дозволяє вибирати різні рушії для системи зберігання, які дозволяють змінювати функціональні інструменти і виконувати обробку даних, зберігаючись у різних типах таблиць. Гнучкість СУБД MySQL забезпечує підтримку більшої кількості типових таблиць: користувачі можуть вибрати як таблиці MyISAM, підтримуючи полнотекстовий пошук, так і таблицю InnoDB, підтримуючи транзакції на рівні окремих записів. Крім того, СУБД MySQL розміщується з спеціальною таблицею пам'яті EXAMPLE, демонструючи принципи створення нових типових таблиць. Завдяки відкритій архітектурі та GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць. Вона також має можливість у використанні інтерфейсів і пакетних команд, які підтримують зручні форми, що містять величезні обсяги даних. Система неймовірно надійна і не прагне загарбити собі всі доступні апаратні ресурси.

Переваги:

- Розширюється безкоштовно
- Прекрасно документована
- Пропонує багато функцій, навіть у безкоштовній версії
- Пакет MySQL включає в стандартні репозиторії найбільш розповсюджені дистрибутивні операційні системи Linux, що дозволяє встановити всі елементи
- Підтримує набір користувальницьких інтерфейсів
- Можна працювати з іншими базами даних, включаючи DB2 та Oracle.

Недоліки:

- Доступно використовувати багато часу та використовувати, щоб залишити MySQL, виконуючи нескладні завдання, а інші системи роблять це автоматично, наприклад: створюють резервні копії.
- Існує вбудована підтримка XML або OLAP.
- Для безкоштовної версії доступна лише платформа підтримки.

Ідеально підходить для: організацій, яким потрібен надійний інструмент управління базами даних, але безкоштовний.

PostgreSQL

PostgreSQL є одним із безлічі безкоштовних популярних варіантів СУБД, часто використовується для баз даних веб-сайтів. Це була одна з перших розроблених систем управління базами даних, тому в даний час вона добре розвинена і здатна керувати як створеними, так і неструктурованими даними. Може бути використаний на більшості основних платформ, включаючи Linux. Прекрасно справляється із завданням імпортної інформації з інших типових баз даних із підтримкою власного інструментарію.

Рушій БД може бути розміщений у ряді середовищ, у тому числу

віртуальних, фізичних та хмарних. Остання версія PostgreSQL 11.5, пропонує обробляти більші об'єкти даних і збільшити кількість одночасно працюючих користувачів. Безпека була покращена завдяки підтримці DBMS_SESSION.

Переваги:

- Масштабований і здатний обробляти терабайти даних.
- Підтримує формат json.
- Існує багато наперед визначених функцій.
- Доступний ряд інтерфейсів.

Недоліки

- Документація туманна, тому, можливо, відповідь на деякі питання прийдеться шукати в інтернеті.
- Конфігурація може збентежити непідготовленого користувача.
- Швидкість роботи може падати під час проведення пакетних операцій або виконання заявок читання.

Ідеально підходить для організацій з обмеженим бюджетом, без кваліфікованих спеціалістів, коли потрібна можливість вибрати свій інтерфейс і використовувати json.

MongoDB

Ще одна безкоштовна база даних, яка має комерційну версію, вона призначена для додатків, які використовують як структуровані, так і неструктуровані дані. Ядро є дуже гнучким і працює при підключенні бази даних до додатків через драйвери MongoDB. Існує широкий вибір доступних драйверів, тому легко знайти драйвер, який буде працювати з необхідним мовою програмування.

Оскільки спочатку система MongoDB не була розроблена для обробки моделей реляційних даних (хоча може це виконувати), можуть виникнути проблеми продуктивності, якщо ви спробуєте використовувати її таким чином. Однак, рушій призначений для обробки різних даних, які не можна віднести до реляційних, і може добре

справлятися там, де інші движки працюють повільно або безсилі.

MongoDB 4.0.4 - це остання версія, і вона має нову підключається систему движків зберігання. Документи можуть бути перевірені в процесі оновлення або виконання вставок, а функції текстового пошуку були покращені. Нова здатність часткового індексування може привести до більш високої продуктивності, зменшуючи розмір індексів.

Переваги:

- Швидкість і простота у використанні
- Движок підтримує json і інші традиційні документи NoSQL.
- Дані будь-якої структури можуть бути збережені / прочитані швидко і легко.

Недоліки:

- SQL не використовується в якості мови запитів.
- Інструменти для перекладу SQL-запитів в MongoDB доступні, але їх слід розглядати саме як доповнення.
- Програма установки може зайняти багато часу.

Підходить для організацій, що працюють з різнорідними даними, які важко піддаються класифікації. Для впровадження потрібні висококласні фахівці.

MariaDB

Ця СУБД є безкоштовною, але як і багато інших безкоштовних програм, пропонує платні версії. Є безліч доступних плагінів розширень, мабуть, це сама швидко-розвиваюча СУБД на даний момент.

MariaDB фактично - це відгалуження від СУБД MySQL, що розробляється спільнотою під ліцензією GNU GPL. Розробка та підтримка сайта MariaDB здійснює компанія MariaDB Corporation Ab і фонд MariaDB Foundation. Поштовхом до створення стала необхідність забезпечення вільного статусу СУБД, на противагу політиці ліцензування MySQL компанією Oracle. Система ліцензування MariaDB зобов'язує

учасників, бажаючих додати свій код в основну гілку СУБД, обмінюватися своїми авторськими правами з MariaDB Foundation для охорони ліцензії і можливості створювати критичні виправлення для MySQL.

Ядро бази даних дозволяє робити вибір з кількох систем зберігання, і це робить використання ресурсів більш оптимізованим, що підвищує продуктивність запитів і обробки. До складу MariaDB включена підсистеми зберігання даних XtraDB для можливості заміни InnoDB, як основний підсистеми зберігання. Також включені підсистеми Aria, PBXT і FederateX. Вона повністю сумісна з MySQL, і прекрасно підходить в якості заміни, тому що повністю відповідає як набір команд, так і API. Багато розробники MySQL були залучені в процес розробки, а зараз беруть участь в розвитку.

Переваги:

- Система працює швидко
- Індикатори дадуть вам знати, як обробляється запит.
- Розширювана архітектура і плагіни дозволяють налаштовувати інструмент відповідно до ваших потреб.
- Шифрування є в мережі, сервері і рівні додатку.

Недоліки:

- На даний момент стабільність нижче, ніж у MySQL, тому навіть на нових проектах можна рекомендувати встановлювати mysql.
- Рушій досить новий, тому поки немає ніяких гарантій подальших оновлень.
- Як і в багатьох інших безкоштовних базах даних, вам доведеться платити за підтримку.

Ідеальна як альтернатива MySQL, якщо MySQL не влаштовує з якихось причин.

Програмне забезпечення середовища розробки

Pyenv

Pyenv - це простий, потужний і багатоплатформовий інструмент для управління декількома версіями Python в Linux-системах, він використовується для:

- перемикання глобальної версії Python для кожного користувача;
- установки локальної версії Python для кожного проекту;
- управління віртуальними середовищами, створеними anaconda або virtualenv;
- перевизначення версії Python зі змінною оточення;
- пошуку команд з кількох версій Python і для багато чого іншого.

Як правило, версія Python за замовчуванням використовується для запуску всіх ваших додатків, якщо ви явно не вкажете версію, яку хочете використовувати в додатку. Але pyenv реалізує просту концепцію використання прокладок (легкі виконувані файли), щоб передати вашу команду правильної версії Python, яку ви хочете використовувати, коли у вас встановлено кілька версій.

Ці прокладки вставлені pyenv в каталоги перед вашим PATH. Тому, коли ви запускаєте команду Python, вона перехоплюється відповідної прокладкою і передається в pyenv, який потім задає версію Python, зазначену вашим додатком, і передає ваші команди правильної версії Python.

Docker

Це ПО з відкритим кодом, принцип роботи якого найпростіше порівняти з транспортними контейнерами

Коли розробляється додаток, буває необхідність надати код разом з усіма його складовими, такими як бібліотеки, сервер, бази даних і т. Д. Ви можете опинитися в ситуації, коли додаток працює на вашому

комп'ютері, але відмовляється включатися на пристрої іншого користувача.

Ця проблема вирішується через створення незалежності ПО від системи. Докер розділяє ядро ОС на всі контейнери (Docker container), що працюють як окремі процеси. Це не єдина подібна платформа, але, безперечно, одна з найпопулярніших і запитаних.

До його переваг відносяться:

- Прискорений процес розробки. Немає необхідності встановлювати допоміжні інструменти на зразок PostgreSQL, Redis, Elasticsearch: їх можна запускати в контейнерах.
- Зручна інкапсуляція додатків.
- Зрозумілий моніторинг.
- Просте масштабування.
- Підтримувані платформи

Докер працює не тільки на його рідній ОС, Linux, але також підтримується Windows і macOS. Єдина відмінність від взаємодії з Linux в тому, що на macOS і Windows платформа інкапсулюється в крихітну віртуальну машину. На даний момент Докер для macOS і Windows досяг значного рівня зручності у використанні.

РОЗДІЛ 4

РОЗРОБКА ДОДАТКУ КЛАССИФІКАЦІЇ АККАУНТІВ СОЦІАЛЬНИХ МЕРЕЖ

4.1 Будування архітектури додатку классифікації аккаунтів соціальних мереж

Основною задачею цього додатку – є визначення направленості окремих користувачів соціальної мережі Instagram за описом(біографією) їх аккаунтів автоматично, в промислових масштабах. Першим кроком у розробці веб-додатку є побудова архітектури системи (Рис 4.1).

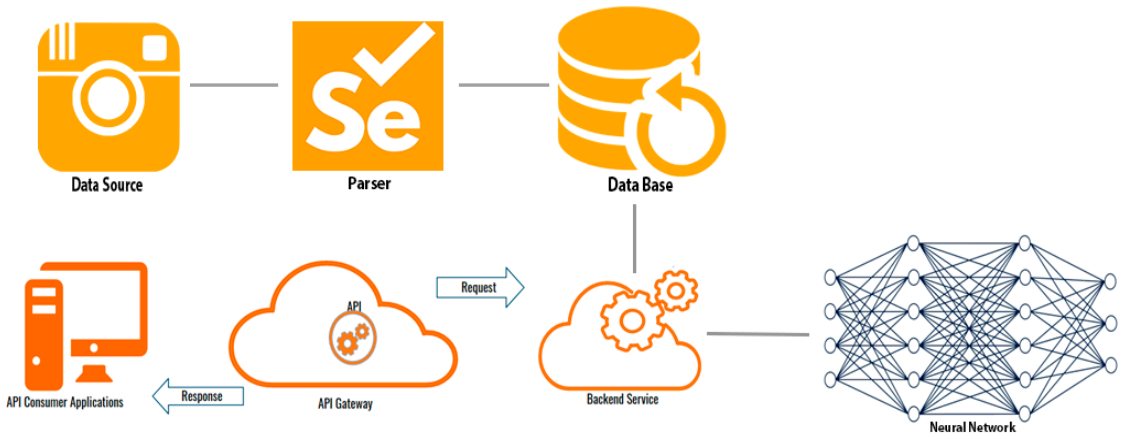

Рис 4.1. **Архітектура веб-додатку**

Парсер. Для навчання нейронної мережі потрібні дані. Ці дані ми спасрили використовуючи Selenium WebDriver. Він був обраний, оскільки для отримання даних на сторінці в соціальній мережі Instagram потрібно імітувати реального користувача. Саме тому ця технологія нам підходить, на відміну від BeautifulSoup або інших аналогів. Дані, які ми отримуємо таким чином, ми використовуємо виключно у дослідницьких цілях і зберігали спочатку в json файлі, з якого дублювали записи в базу даних PostgreSQL. Для нашого проекту потрібні дві обов'язкові таблиці: Аккаунт, Сегмент(клас). (Рис 4.2.)

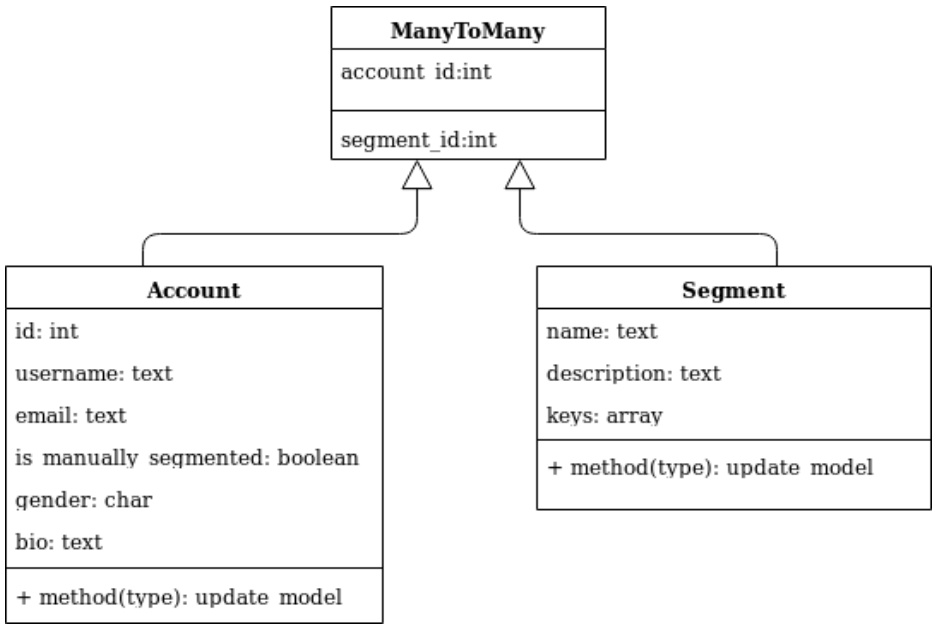

Рис 4.2. **Архітектура таблиць бази даних**

Навчати нейронну мережу нам потрібно за допомогою навчання з вчителем, тобто ми маємо отримати вибірку даних зі заздалегідь визначеними класами. Встановивши зв’язки між описами аккаунтів та відповідними класами всередині моделі нейронної мережі, вона зможе класифікувати тексти і визначати ймовірність відношення користувача(аккаунту) до відповідних класів.

Так як ми маємо нестандартне джерело, не використовуючи заздалегідь класифіковану вибірку даних, визначити класи доводиться мануально. Для цього ми розробили браузерне розширення, яке в свою чергу також виконує функцію візуального представлення всього додатку. Воно допомагає знайти відповідності у належності користувача до класів, знаходячись, безпосередньо на його сторінці, зберегти ці дані в базу, мати до них доступ, прогнозувати класи аккаунтів та пропонувати найбільш сприятливі для мануальної сегментації аккаунти, вони як правило мають опис не менше 100 символів, та хоча б якісь сегменти визначені пошуком на знаходження в описі ключових слів. Воно і служить головним візуальним інтерфейсом додатку. (Рис 4.3).

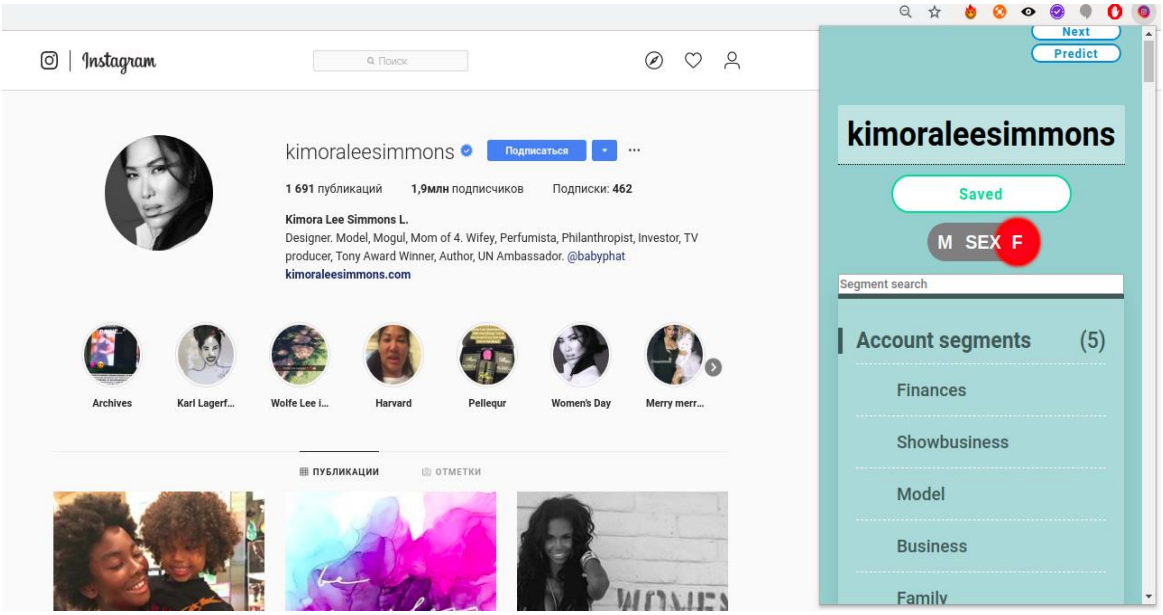

Рис 4.3. **Браузерний додаток класифікації аккаунтів**

Для роботи браузерного розширення нам потрібно API(Application Personal Interface), яке з'єднує розширення з сервером. Для імплементації API використано "Django REST framework".

Очевидно, що для того, щоб визначити належність аккаунтів до класів, ми маємо мати набір класів. Для цього було проведено аналіз популярних аккаунтів та медіа агенцій і виділено основні сегменти та їх підгрупи. Також значну кількість сегментів було додано у процесі. В результаті ми отримали 138 сегментів. Зроблено це було через стандартну адміністративну панель Django (Рис 4.4).

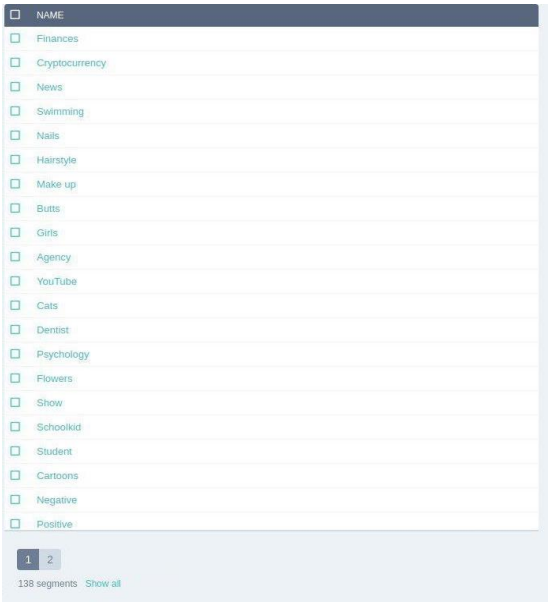

Рис 4.4. **Адміністративна панель з прикладами сегментів.**

Головним елементом у цій системі, звісно, є нейронна мережа, яку ми будемо будувати у наступному розділі.

Всі дані які були отримані шляхом мануальної сегментації через браузерне розширення були збережені з властивістю is_manually_segmented=True, тільки ці аккаунти використовуються при навчанні нейронної мережі. Описи акаунтів и сегменти векторизуються .

Для будування моделі ми використовуємо фреймворк Keras, та scikit-learn, як допоміжний інструмент для векторизації та інших прикладних задач. Основною складністю при будування моделі було підбір останнього шару моделі, та вибрати правильні функція активації та втрати, тому що на виході з мережі ми мали отримувати 138 ймовірностей відношення до класів (Рис 4.5).

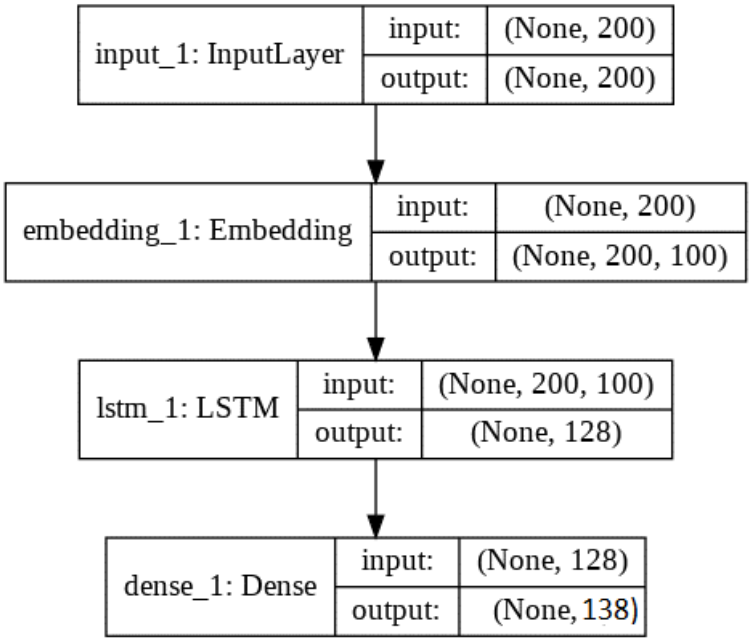

Рис 4.5. **Модель нейронної мережі**

Останнє, на що потрібно звернути увагу – це зберігання та актуалізація моделі нейронної мережі. Для того щоб мати можливість прогнозувати аккаунти, використовуючи найбільш актуальну модель, ми маємо періодично генерувати її заново, щоб містити в собі дані про нових мануально-сегментованих користувачів, але для того щоб

отримати відповідь від моделі швидко, ми маємо заздалегідь згенерувати її, та зберегти у будь якому вигляді. Це може бути вся модель переведена в байти, та збережена файлом, або таблиця ваг моделі, збережена файлом або у кеші, але це має виконуватись постійно та незалежно від веб-додатку, тому ми інтегрували бібліотеку celery, за допомогою якої, ми конфігурували періодичне перенавчання моделі нейронної та збереження таблиці ваг у кеш, звідки береться модель при запиті на прогнозування належності аккаунту Instagram, або при масовій сегментації всієїбази користувачів.

4.2 Реалізація додатку классифікації аккаунтів соціальних мереж та його використовування

Першим етапом розробки є ініціалізація і будування структури проекту. Ініціалізувавши проект бек-енду додатку ми створюємо віртуальне оточення для цього проекту. Встановивши всі залежності потрібно створити необхідні модулі, в яких буде розроблюватися функціонал проекту за допомогою команди python manage.py startapp <app_name>. Для комфортної розробки у команді і запобіження помилок різних оточень ми використовуємо Docker. Це програмне забезпечення яке допомагає контейнерузувати допоміжне програмне забезпечення і зафіксувати проект у певному стані. Ми конфігурували в ньому базу даних Postgres, та швидкі сховища Redis – для сбереження результатів celery, та RabbitMQ – менеджер черг задач. (Рис 4.6).

```yaml
version: "3.7"
services:
  db:
    image: postgres:9.6
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: socialrecognition_passwd
      POSTGRES_DB: socialrecognition_db
    ports:
      - "5437:5432"
  rabbitmq:
    image: rabbitmq:3.6
    ports:
      - "5679:5672"
  redis:
    image: redis:3.2
    ports:
      - "6382:6379"
```

Рис 4.6. **Конфігураційний файл docker-compose.yml**

Для того щоб використовувати додаток у мережі, треба його розгорнути. Ми вибрали для цього безкоштовну платформу Heroku. Вона досить легка у користуванні і забезпечує безперервну розробку та безперервну інтеграцію (CD/CI). Для цього потрібно створити порожній додаток на платформі, прописати Procfile з активними процесами на хості та release.sh з командами які мають викликатися при розгортці. В результаті ми маємо наступну структуру додатку (Рис 4.7.).

Опис структури:

- sr_server – головний модуль в якому знаходяться основні конфігурації та сервер додатку
- sr_core – модуль з елементами, які використовуються у всьому проекті
- sr_accounts – модуль з моделями бази даних та API додатку (розроблено іншим членом команди)
- sr_recognition – модуль створення моделі нейронної мережі

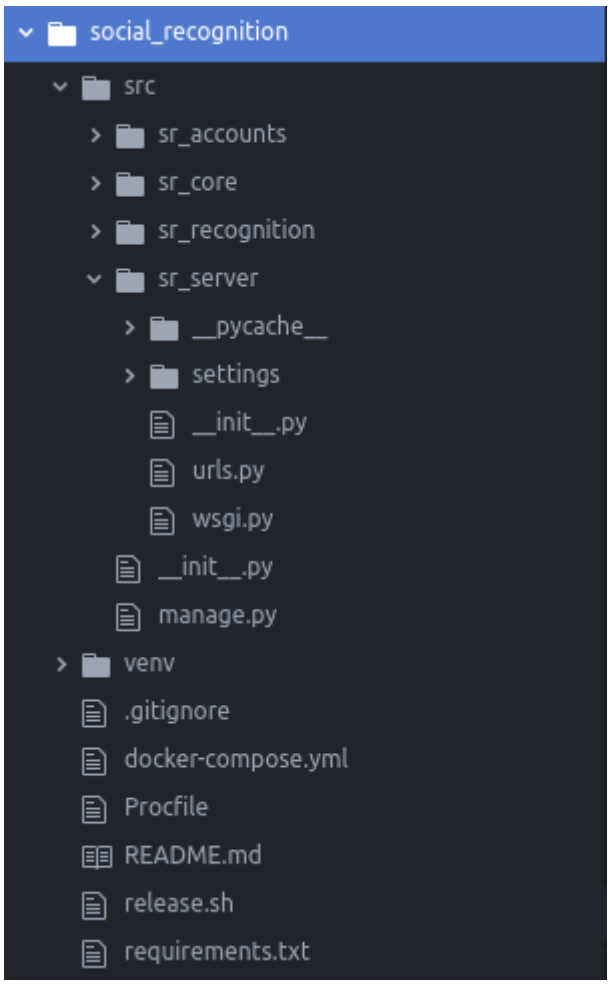

Рис 4.7. **Структура додатку**

Щоб розробити гнучкий модуль створення моделі нейронної мережі дані потрібно провести наступні етапи роботи з даними:

- Прочитати дані аккаунтів та сегментів с бази даних.
- Профільтрувати аккаунти на актуальність (біографія аккаунта має достатню кількість символів і має ключові слова на основі яких можливо робити висновки).
- Профільтрувати біографії та видалити пунктуаційні знаки, зайві пробіли та стоп-слова (артиклі, сполучники, тощо)
- Розділити вхідні дані на тренувальну та тестову вибірки.
- Обрізати всі вхідні дані до фіксованого розміру.
- Векторизувати або токенізувати вхідні дані.
- Встановити вхідний шар (Input).
- Побудувати і встановити шар векторного представлення слів

(Embedding) за допомогою файлів пре-тренерованих векторів від GloVe.

- Встановити шар довгої короткострокової пам'яті (LSTM)
- Встановити вихідний щільний шар(Dense) з активатором sigmoid
- Зкомпілювати шари у модель з встановленними функцією втрати binary_crossentropy та оптимізатором adam.
- Натренувати зкомпільовану модель тренувальними даними у 5 епох, валідаціонна вибірка – 20% від тренувальної (Рис 4.8.).

```
Train on 4 samples, validate on 2 samples
Epoch 1/5
4/4 [==============================] - 1s 232ms/step - loss: 0.6997 - acc: 0.5255 - val_loss: 0.6879 - val_acc: 0.5510
Epoch 2/5
4/4 [==============================] - 0s 7ms/step - loss: 0.6825 - acc: 0.6020 - val_loss: 0.6819 - val_acc: 0.5816
Epoch 3/5
4/4 [==============================] - 0s 7ms/step - loss: 0.6657 - acc: 0.6888 - val_loss: 0.6757 - val_acc: 0.6327
Epoch 4/5
4/4 [==============================] - 0s 6ms/step - loss: 0.6488 - acc: 0.7857 - val_loss: 0.6688 - val_acc: 0.6531
Epoch 5/5
4/4 [==============================] - 0s 7ms/step - loss: 0.6312 - acc: 0.8571 - val_loss: 0.6609 - val_acc: 0.6837
2/2 [==============================] - 0s 2ms/step
Test Score: 0.6851018667221069
Test Accuracy: 0.7551020383834839
```

Рис 4.8. **Процес тренуваня**

- Оцінити точність моделі за допомогою тестової вибірки.
- Відобразити результати навчання за допомогою mathplotlib (Рис 4.9.)

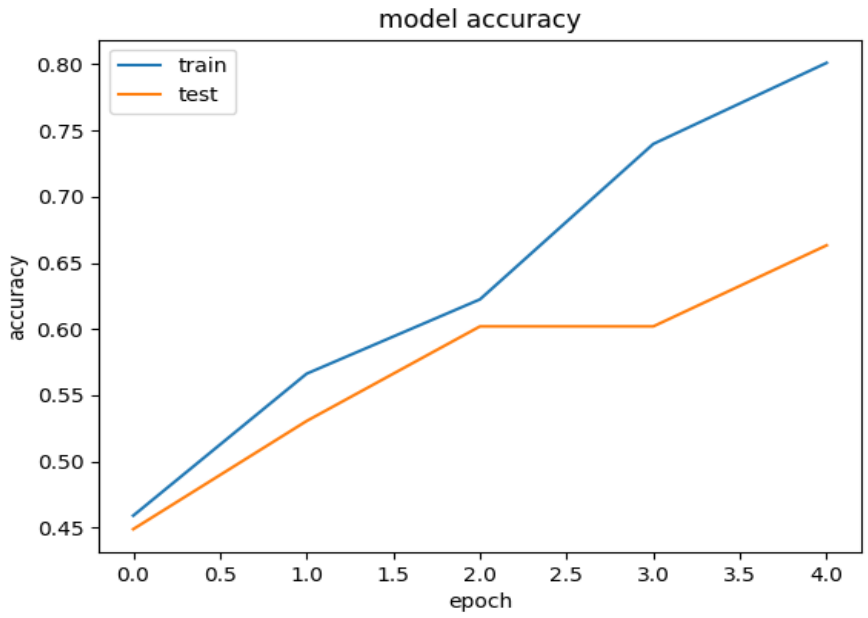

Рис 4.9. **Точність моделі нейронної мережі на тестовій вибірці**

Таким чином ми маємо працюючу нейронну мережу, але спочатку ми не мали тренувальних даних. Дані зібрані парсером не мали визначених класів, які є цілями класифікації. Показники зображені на рисунку отримані вже після заповнення бази даних тренувальними дані за допомогою веб-розширення, через яке ми мануально визначали класи до яких належать користувачі. Розробка веб-розширення складалася з таких етапів:

- Розробка візуальної частини
- Підключення розширення до API на сервері heroku.

В результаті ми маємо візуальний інтерфейс нашого додатку за допомогою якого ми можемо читати дані з бази, записувати їх, отримувати найбільш актуальні записи користувачів для мануальної класифікації та передбачувати їх класи, перебуваючи на їх сторінці у соціальній мережі Instagram (Рис 4.10.).

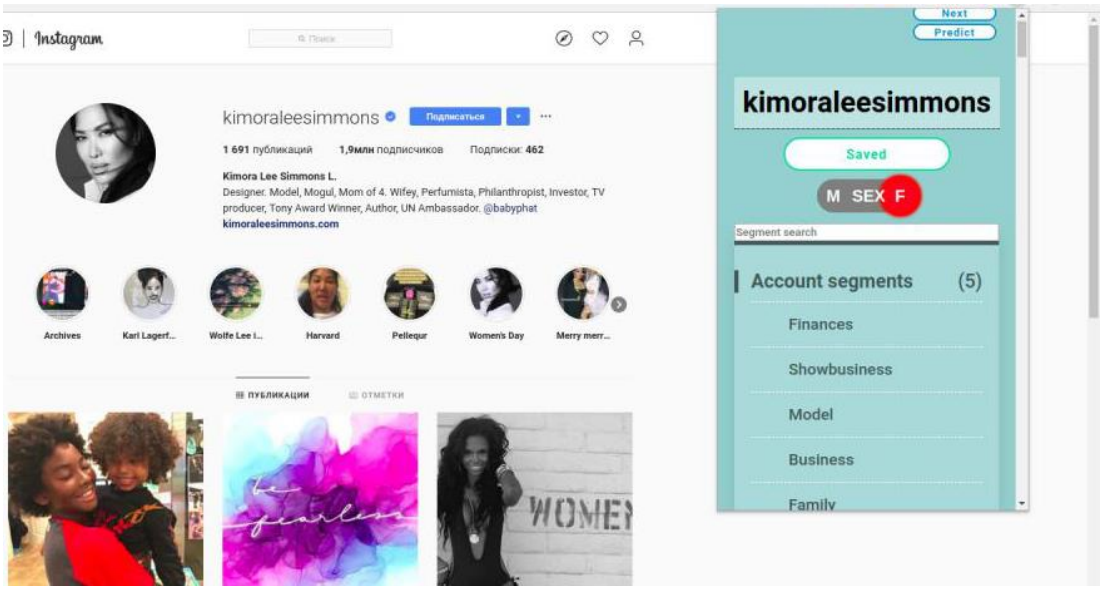

Рис 4.10. **Візуальне представлення додатку класифікації аккаунтів соціальних мереж.**

ВИСНОВКИ

- Було досліджено поняття та призначення машинного навчання.
- Визначено основні типи та підходи машинного навчанная.
- Ознайомлено с технологіями (бібліотеками, фреймворками).
- Досліджено джерело даних для додатку. (Облікові записи користувачів соціальних мереж)
- Розроблено нейронну мережу
- Розроблено браузерний додаток та підключити до API классифікації аккаунтів соціальних мереж
- Навчено нейронну мережу та використовуючі дані для навчання скореговано и оптимізовано роботу нейронної мережі.

СПИСОК ДЖЕРЕЛ

1. История глубинного машинного обучения [Електронний ресурс] / Леонід Черняк // computerworld – 2016. – Режим доступу до ресурсу: https://www.computerworld.ru/articles/Istoriya-glubinnogo-mashinnogo-obucheniya

2. Машинное обучение — это легко [Електронний ресурс] // Хабрахабр – 2017. – Режим доступу до ресурсу: https://habrahabr.ru/post/319288/

3. Введение в машинное обучение с помощью Python и Scikit-Learn [Електронний ресурс] // Хабрахабр – 2015. – Режим доступу до ресурсу: https://habrahabr.ru/company/mlclass/blog/247751/

4. Введение в машинное обучение с помощью Scikit-Learn (перевод документации)[Електронний ресурс] // Хабрахабр – 2015. – Режим доступу до ресурсу: https://habrahabr.ru/post/264241/

5. Machine learning with Python / Олег Шидловский / Doist [Python Meetup 27.03.15][Електронний ресурс] // YouTube – 2015. – Режим доступу до ресурсу: https://www.youtube.com/watch?v=me8ijsHw2c

6. Гид по структуре машинного обучения[Електронний ресурс] / Мария Пушикова // нетология – 2017. – Режим доступу до ресурсу: https://netology.ru/blog/machine-learning-guide

7. Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных.[Електронний ресурс] // MachineLearning.ru – 2015. – Режим доступу до ресурсу: http://www.machinelearning.ru

8. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными / А. Мюллер, Сара Гвидо [Електронний ресурс] // owlweb — 2016 — Режим доступу до ресурсу: https://owlweb.ru/wp-content/uploads/2017/06/a.myuller-s.gvido-

vvedenie-v-mashinnoe-obuchenie-s-pomoshhyu-python.-rukovodstvo-dlya-specialistov-po-rabote-s-dannymi-2017.compressed-1.pdf

9. Нейт Сильвер. Сигнал и Шум. Почему одни прогнозы сбываются, а другие — нет // Азбука-Аттикус, КоЛибри, 2015

10. William M. Bolstad. Introduction to Bayesian Statistics, 2nd Edition // Wiley-Interscience; 2nd edition

11. Хей Дж. Введение в методы байесовского статистического вывода. — М.: Финансы и статистика, 1987

12. Коэльо Л.П., Ричарт В. Построение систем машинного обучения на языке Python. 2016. 302 с.

13. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. Springer, 2014. — 739 c.

14. An introduction to ROC analysis. — URL: http://www.sciencedirect.com/science/article/pii/S016786550500303X

15. scikit-learn. — URL: http://scikit-learn.org/stable/

16. scikit-learn documentation. — URL: http://scikit-learn.org/stable/user_guide.html

17. MachineLearning.ru. Кривая ошибок. — URL: http://www.machinelearning.ru/wiki/index.php?title=ROC-%D0%BA%D1%80%D0%B8%D0%B2%D0%B0%D1%8F

18. Wikipedia. Random forest // Википедия, свободная энциклопедия. — URL: https://en.wikipedia.org/wiki/Random_forest

19. Wikipedia. Keras // Википедия, свободная энциклопе- дия. — URL: https://uk.wikipedia.org/wiki/Keras

20. Wikipedia. TensorFlow // Википедия, свободная энциклопедия. — URL: https://uk.wikipedia.org/wiki/TensorFlow

21. Wikipedia. Theano // Википедия, свободная энциклопедия. — URL: https://uk.wikipedia.org/wiki/Theano

22. Python для сложных задач: наука о данных и машинное обучение. – СПб.: Питер, 2018 – 576 с.: ил. – (Серия “Бестселлеры O’Reilly”)

23. Башмаков А.И. Интеллектуальные информационные технологи [Текст]. – М.: Изд. МГТУ им. Н.Э. Баумана, 2005. – 302 с.

24. Брюхомицкий Ю.А. Нейросетевые модели для систем информационной безопасности [Текст]. – Таганрог, 2005. – 159 с.

25. Комашинский В.И. Нейронные сети и их применение в системах управления и связи [Текст] / В.И. Комашинский, Д.А. Смирнов. – М.: Горячая линия – Телеком, 2003. – 94 с

26. Alpaydın Ethem. M. Introduction to Machine Learning/ Ethem Alpaydın.– London:. The MIT Press, 2010. - 579 с

27. Круглов В.В. Дли М.И. Голунов Р.Ю. Нечеткая логика и искусственные нейронные сети. ФИЗМАТЛИТ, 2001 г.

28. Лукашин, Ю. П. Адаптивные методы краткосрочного прогнозирования временных рядов / Ю.П. Лукашин. - М.: Финансы и статистика, 2015. - 416 с.

29. Галушкин, А.И. Нейронные сети: основы теории / А.И. Галушкин. - М.: ГЛТ, 2012. - 496 с.

30. Садовникова, Н.А. Анализ временных рядов и прогнозирование / Н.А. Садовникова, Р.А. Шмойлова. - М.: МФПУ Синергия, 2016. - 152 с.

31. Steel kiwi. Top 10 python web frameworks to learn// Steel kiwi — URL: https://steelkiwi.com/blog/top-10-python-web-frameworks-to-learn/

32. Drach. Сравнение современных СУБД // Drach. — URL: http://drach.pro/blog/hi-tech/item/145-db-comparison