Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет**

**при Правительстве Российской Федерации»**

**(Финансовый университет)**

Департамент анализа данных и машинного обучения

Пояснительная записка к курсовой работе по дисциплине

«Технологии анализа данных и машинного обучения»

на тему:

**«Машинное обучение в задачах классификации текстов»**

Выполнил:

Студент группы ПИ19-1

Содиков Фарход Фирдавсович

Научный руководитель:

Доцент, Кандидат педагогических наук

Никитин Пётр Владимирович

**Москва**

**2022**

Содержание

[ВВЕДЕНИЕ 3](#_Toc101907073)

[Постановка задачи 4](#_Toc101907074)

[Реализация 6](#_Toc101907075)

[Описание датасета 6](#_Toc101907076)

[Векторизация текста 12](#_Toc101907077)

[Построение моделей 12](#_Toc101907078)

[Логистическая регрессия (LogisticRegression) 14](#_Toc101907079)

[Метод опорных векторов с линейным ядром (SVC) 15](#_Toc101907080)

[Метод опорных векторов с гауссовым ядром (SVC) 16](#_Toc101907081)

[Метод k ближайших соседей (KNeighborsClassifier) 17](#_Toc101907082)

[Многослойный перцептрон (MLPClassifier) 18](#_Toc101907083)

[Логистическая регрессия (LogisticRegression) 19](#_Toc101907084)

[Классификатор дерева решений (DecisionTreeClassifier) 20](#_Toc101907085)

[Классификатор «Случайный лес» (RandomForestClassifier) 21](#_Toc101907086)

[Наивный байесовский метод (MultinominalNB) 22](#_Toc101907087)

[Вывод о моделях 22](#_Toc101907088)

[Инструменты реализации 24](#_Toc101907089)

[Среда разработки Jupyter Notebook 24](#_Toc101907090)

[Язык программирования Python 24](#_Toc101907091)

[Библиотека NumPy 25](#_Toc101907092)

[Библиотека Pandas 25](#_Toc101907093)

[Библиотека Scikit Learn 26](#_Toc101907094)

[Библиотека Matplotlib 30](#_Toc101907095)

[ЗАКЛЮЧЕНИЕ 31](#_Toc101907096)

[ИСТОЧНИКИ 32](#_Toc101907097)

# ВВЕДЕНИЕ

Основная идея курсового проекта, в целом машинного обучения, состоит в том, чтобы машина (компьютерная) не просто использовала заранее написанный алгоритмический код, а сама обучалась решению поставленной задачи. Главной и единственной средой разработки в создании машинного обучения является Jupyter Notebook, с помощью которого сразу можно видеть результат выполнения кода или одного из его фрагментов. В данном проекте единственным языком программирования является Python, потому что он является одним из тех языков, который легко и просто решает проблемы, связанные с машинным обучением и анализом больших данных. Также этот язык имеет множество различных библиотек, некоторые из которых используется в поставленной задаче.

В разработку машинного обучения входит:

* Классификация данных
* Кластеризация данных
* Выявление аномалий (в нашей задаче не является обязательным пунктом)
* Снижение размерности (в нашей задаче не является обязательным пунктом)
* Предсказание событий (регрессия)

В наше время искусственный интеллект значительно развивается и всё больше IT-компаний нуждаются в создании машин, которые обучают самих себя для автоматизации работ не только в IT-сфере, но и в других отраслях.

К концу работы должен быть готовый машинный код, который проанализирует небольшой набор данных (датасет) и обучит себя эмпирическим методом с использованием различных библиотек.

# Постановка задачи

Перед стартом реализации нашей программной части работы нужно понять, какая задача стоит перед нами.

Три этапа реализации:

1. Получение разновидных текстов из различных источников. Можно брать готовые датасеты.
2. Загрузка всех собранных данных или датасета.
3. Получение результата с максимально высшей точностью.

Давайте определимся с такими терминами как: машинное обучение, глубокое обучение, нейронные сети и искусственный интеллект. Как они связаны? Чем отличаются друг от друга?

Машинное обучение, глубокое обучение и нейронные сети — всё это области искусственного интеллекта. Однако глубокое обучение на самом деле является подразделом машинного обучения, а нейронные сети - подразделом глубокого обучения. Для понимания и освоения вышеперечисленных терминов давайте дадим им определения простыми словами:

* *Нейронные сети (искусственные)* – это математические модели. Их работоспособность схожа с нервными клетками животных. Нейронные сети являются подмножеством машинного обучения и лежат в основе алгоритмов глубокого обучения. Искусственные нейронные сети состоят из двух слоев, содержащих входной слой, один или несколько скрытых слоев и выходной слой. Каждый узел, или искусственный нейрон, соединяется с другим и имеет соответствующий вес и пороговое значение. Если выходные данные любого отдельного узла превышают указанное пороговое значение, этот узел активируется, отправляя данные на следующий уровень сети. В противном случае данные не передаются на следующий уровень сети.
* *Машинное обучение* — это отрасль искусственного интеллекта (ИИ) и информатики, которая фокусируется на использовании данных и алгоритмов для имитации способа обучения людей, постепенно повышая его точность.

После того как мы поняли основную идею курсового проекта, этапы реализации поставленной задачи и основные термины мы можем перейти к описанию датасета.

# Реализация

## Описание датасета

В нашей задаче мы будем использовать готовый датасет «The 20 newsgroups text dataset», в котором, как видно по названию, содержится 20 наборов данных, которые в свою очередь содержат более 18000 новостных постов по 20 различным темам. Данные разделены на два подмножества: одна для обучения, а другая для тестирования или для оценки производительности.

Рисунок с кодом загрузки датасета:

Изображение выглядит как текст

Автоматически созданное описание

Среда разработки не может показать нам все наши новости и предупреждает о превышении скорости ввода данных, так как датасет имеет большой объём данных. Поэтому сервер среды временно прекращает отправку выходных данных.

Рисунок с предупреждением о большом объёме данных в датасете:

Изображение выглядит как текст

Автоматически созданное описание

Чтобы увидеть хоть какое-то описание воспользуемся функцией .DESCR(), которая выводит базовую описательную статистику, включая метки переменных.

Рисунок с описательной статистики:

Изображение выглядит как текст

Автоматически созданное описание

Также можно посмотреть один из примеров постов. Например, возьмем случайную группу под нулевым индексом и пост под нулевым индексом, то есть первую в группе. Рисунок с примером поста:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок с ещё одним примером поста:

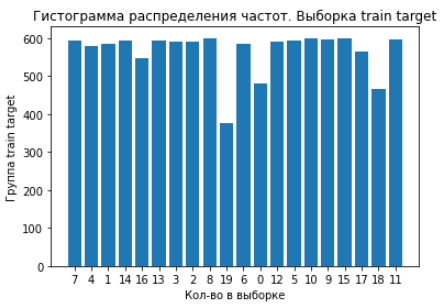
Изображение выглядит как текст

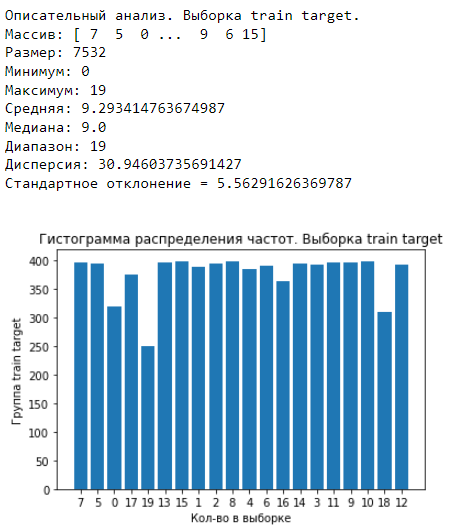
Автоматически созданное описание

Проведём описательный анализ датасета:

Изображение выглядит как текст

Автоматически созданное описание



Как можно заметить размер выборки равен 11314. Остальные из 18000 данных находятся в подмножестве newsgroup\_test, то есть в тестовом. Проверим это изменив первую строчку на «arr=newsgroups\_test.target»: 

## Векторизация текста

Для векторизации текста используем функцию CountVectorizer:

Изображение выглядит как текст

Автоматически созданное описание

*CountVectorizer* нужен нам для того, чтоб машина могла понимать символы и слова. Но так как мы имеем дело с текстовыми данными, нам нужно предоставлять их в цифрах, чтоб они были понятны машине. С помощью этого метода мы преобразуем текст в числовые данные.

## Построение моделей

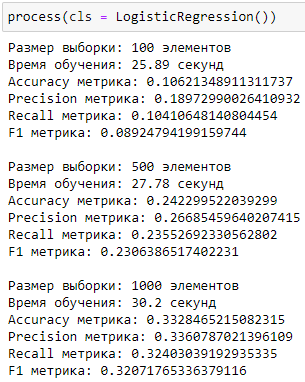
Для начала создадим общую функцию для создания моделей. Входной данной является классификатор. Функция будет выводить:

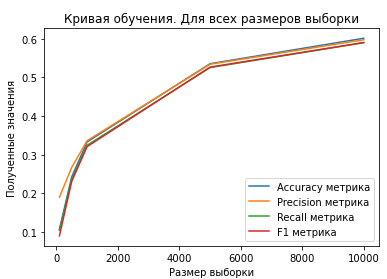
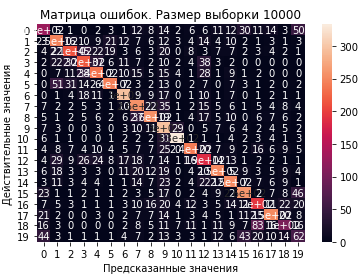
1. Размер выборки
2. Время обучения
3. Accuracy
4. Precision
5. Recall
6. F1 метрика
7. График кривой обучения
8. Матрица ошибок

Рисунок с функцией process():

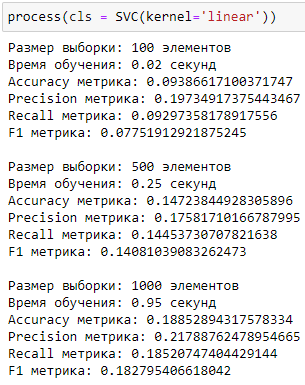


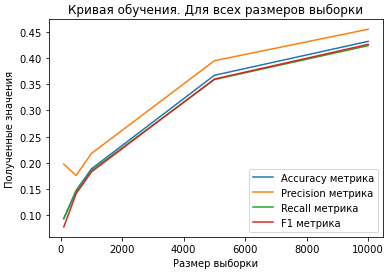
### **Логистическая регрессия (LogisticRegression)**

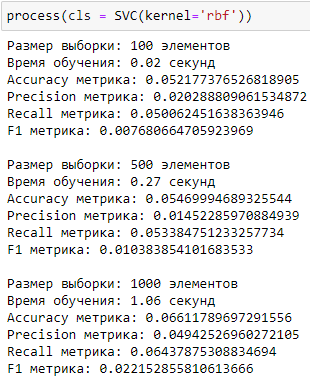
 

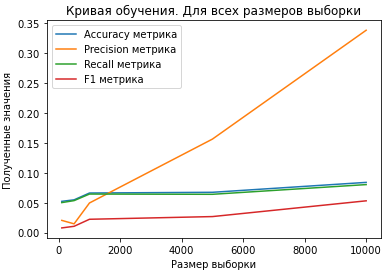
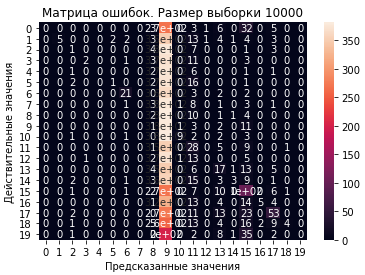
### Метод опорных векторов с линейным ядром (SVC)

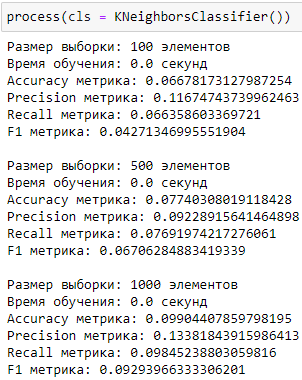
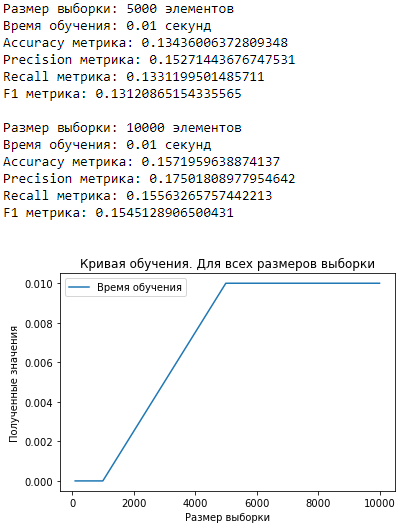
 

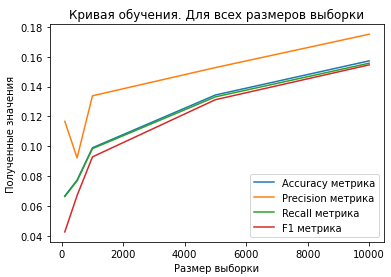
### Метод опорных векторов с гауссовым ядром (SVC)

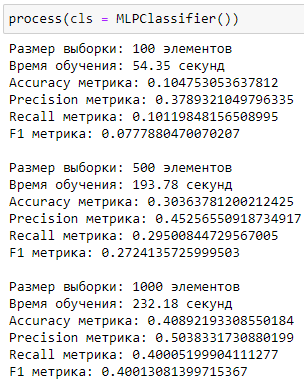
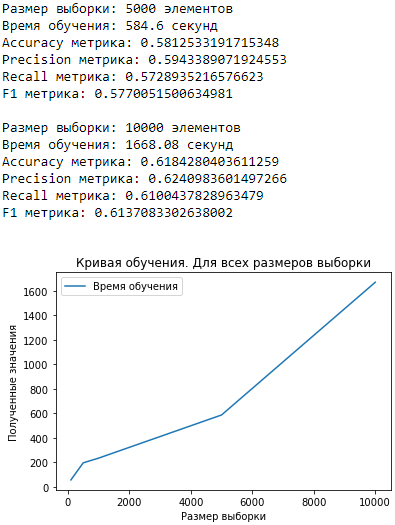
 

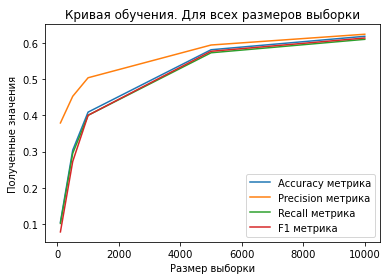
### Метод k ближайших соседей (KNeighborsClassifier)

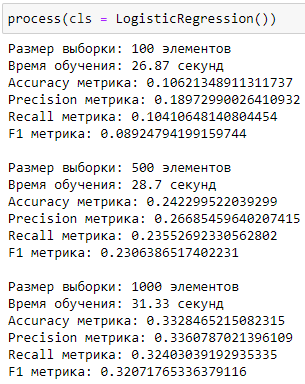
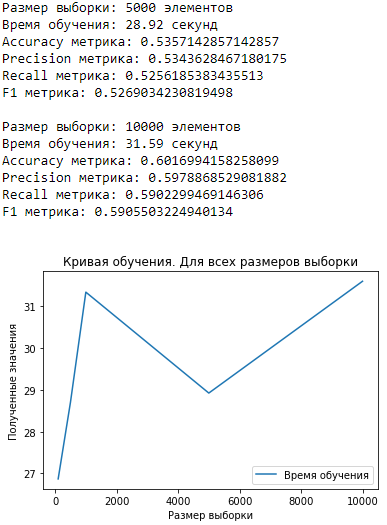
 

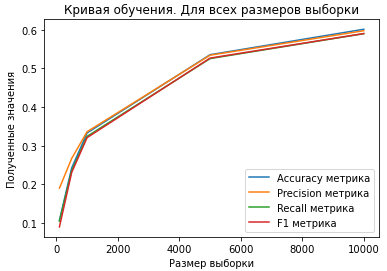
### Многослойный перцептрон (MLPClassifier)

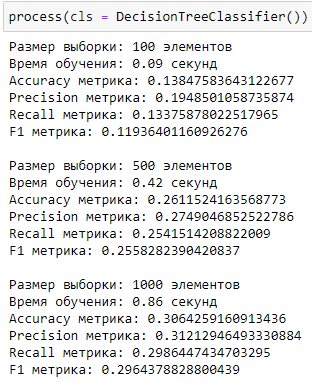
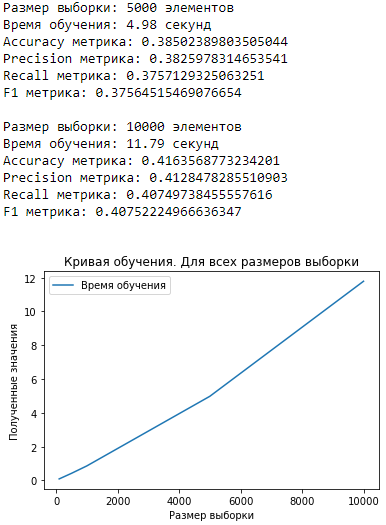
 

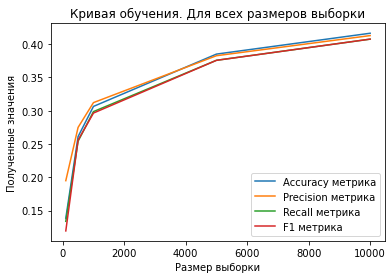
### Логистическая регрессия (LogisticRegression)

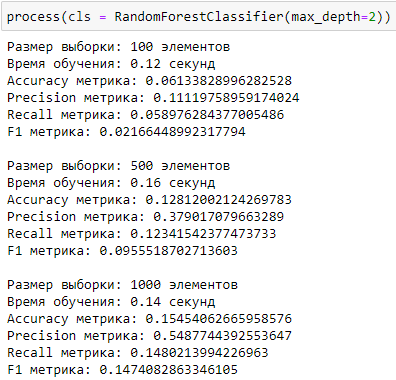
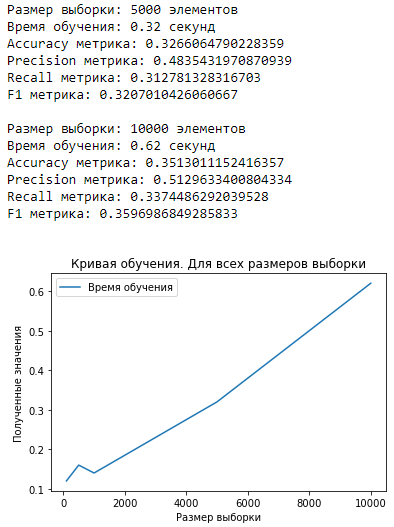
 

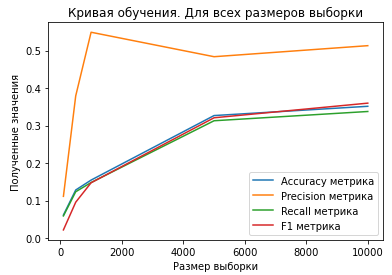
### Классификатор дерева решений (DecisionTreeClassifier)

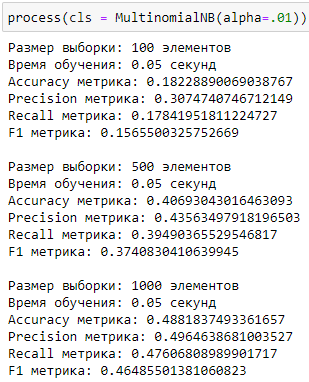
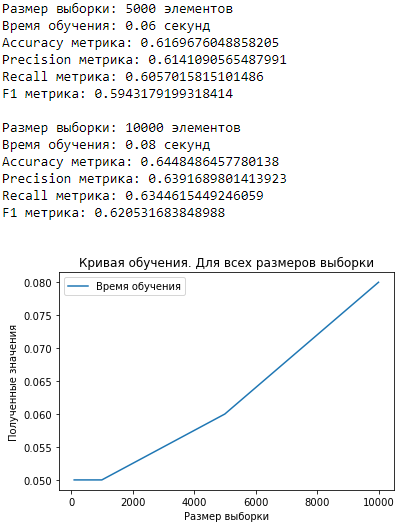
 

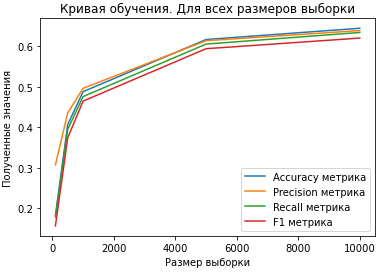
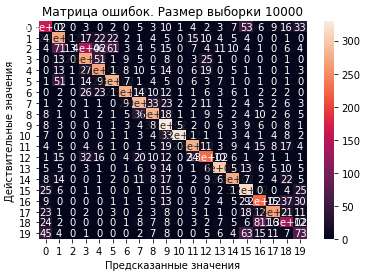
### Классификатор «Случайный лес» (RandomForestClassifier)

### Наивный байесовский метод (MultinominalNB)

## Вывод о моделях

Наиболее хорошо себя показали модели:

1. Наивный байесовский метод (MultinomialNB)
2. Логистическая регрессия (LogisticRegression)
3. Многослойный перцептрон (MLPClassifier)

Самая быстрая и точная модель:

1. Наивный байесовский метод (MultinomialNB)

Модель с самым долгим обучением:

1. Многослойный перцептрон (MLPClassifier)

# Инструменты реализации

## Среда разработки Jupyter Notebook

Jupyter Notebook – это среда разработки, где пользователь сразу может видеть выполнение кода или его отдельные фрагменты. Одной из главных отличий этой среды от других является возможность разбиения программного кода на куски с любым порядком выполнения. То есть вы можете создать класс или функцию и сразу же её проверить без запуска и выполнения всего программного кода. Также можно отдельно смотреть содержимое загруженных файлов и отдельно обработать данные файла. Этот функционал помогает достичь поставленных целей, так как, если мне захочется построить график прямо в середине кода я смогу просто запустить функцию и увидеть результат. Команда jupyter notebook создала свой ноутбук в облаке. Теперь пользователи могут пользоваться им через браузеры.

## Язык программирования Python

Язык программирования Python – высокоуровневый ЯП общего назначения, направленный на повышение производительности разработчика и читаемости кода. Этот язык имеет множество парадигм программирования, такие как объектно-ориентированное (ООП), императивное, структурное, функциональное и другие. У языка есть свободная лицензия, позволяющая использовать ЯП без ограничений в любых разработках.

Python хорошо подходит под выполнения математических задач, научных расчётов, а также для машинного обучения и анализа больших объёмов данных. Следующей причиной, по которой мы используем этот язык, является наличие большого количества различных библиотек для разработки приложений, а также анализа, обработки, визуализации данных, но основными пакетами в разработке машинного обучения и анализа данных являются следующие библиотеки: NumPy, Pandas, Sklearn, Matplotlib, Seaborn, а также time, collections, itertools, warnings.

## Библиотека NumPy

NumPy – это одна из популярных библиотек с открытым исходным кодом языка Python, которая помогает добавить поддержку огромных многомерных массивов и матриц, а также высокоуровневых и математических функций. Одним из основных объектов NumPy является одномерный массив.

Если реализовывать математические алгоритмы на чистом Python, то зачастую ваш программный код работает медленнее, чем на других компилируемых языках, поэтому NumPy помогает оптимизировать работу с многомерными массивами. NumPy написан на языке C.

## Библиотека Pandas

Pandas – это высокоуровневая (построена поверх более низкоуровневой библиотеки NumPy) библиотека языка Python для анализа и обработки данных. Данная библиотека является самой продвинутой и перспективной в работе с данными. Самыми основными структурами данного пакета: DataFrame и Series. Эти специальные структуры данных и операции для манипулирования временными рядами и числовыми таблицами. Название Pandas происходит от эконометрического термина «панельные данные». Pandas находится под лицензией BSD, которая позволяет свободно пользоваться данной библиотекой.

Возможности библиотеки:

* Манипуляция индексированными массивами двумерных массивов.
* Совмещение данных и обработка информации
* Объединение и слияние наборов данных

## Библиотека Scikit Learn

Scikit-learn – это также один из популярнейших и широко использующих пакетов для машинного обучения и Data Science. С помощью этого ПО можно выполнять большое количество операций. Scikit-learn имеет большое количество встроенных готовых алгоритмов. Эта библиотека имеет одну из лучших документаций о своих встроенных алгоритмах классах, методах и функциях.

Scikit-learn поддерживает:

* Выбор моделей
* Классификации
* Регрессии
* Предварительную обработку данных
* Кластерный анализ
* Уменьшение размерности

Scikit-learn не поддерживает:

* Нейронные сети
* Обучение с ассоциативными правилами
* Самоорганизующиеся карты (Кохонена сети)
* Обучения с подкреплением (Reinforcement learning)

У пакета есть свои наборы данных, с помощью которых можно тестировать свои модели. Так же как и вышеперечисленные библиотеки, Scikit-learn имеет открытый исходный код. Он бесплатный и лицензирован под BSD, как и Pandas-пакет.

Алгоритмы классификации:

* *LogisticRegression* (LogReg) – это один из главных алгоритмов данной библиотеки. Используется для отнесения исследований к дискретному набору классов. Эта регрессия преобразует свой вывод с помощью Сигмоида для возращения значения вероятности, которое в следствие может округлиться в сторону одного из дискретных классов.
* *KNeighborsClassifier* – это одна из контролируемых моделей машинного обучения. Модель учится на наборе помоченных данных. Сначала модель получает набор входных объектов и выходных значений. После приёма данных модель обучается на них, чтобы узнать, как сопоставить выходные данные с желаемыми выходными данными. Это, в свою очередь, нужно для того, чтоб модель могла делать прогнозы на невидимых данных. Работает она просто. В первую очередь модель берет ближайшие помеченные точки, глядя на «k». Затем происходит присвоение меток для большинства точек рядом с «k». Например, если k = 20, а пятнадцать из точек жёлтые, а пять остальных голубые, то рассматриваемая точка превратится в жёлтый, так как желтый цвет является большинством.
* *DecisionTreeClassifier* – это древовидная структура, похожая на блок-схему, в которой внутренний узел представляет функцию (или атрибут), ветвь представляет правило принятия решения, а каждый конечный узел представляет результат. Самый верхний узел в дереве решений известен как корневой узел. Он учится разделять на основе значения атрибута. Он рекурсивно разбивает дерево, вызывая рекурсивное разбиение. Эта структура, похожая на блок-схему, поможет вам в принятии решений. Это визуализация, похожая на блок-схему, которая легко имитирует мышление на человеческом уровне. Вот почему деревья решений легко понять и интерпретировать.
* *SVM* (машина опорных векторов) – это один из алгоритмов классификации с контролируемым машинным обучением. Её результаты хороши, поэтому этот алгоритм является одним из популярных. SVM отличается от других алгоритмов классификации тем, что он выбирает границу принятия решения, которая максимизирует расстояние от ближайших точек данных всех классов. SVM не просто находит границу принятия решения; он находит наиболее оптимальную границу принятия решения. Наиболее оптимальной границей принятия решения является та, которая имеет максимальный запас от ближайших точек всех классов. Ближайшие точки от границы принятия решения, которые максимизируют расстояние между границей принятия решения и точками, называются опорными векторами.
* *MLP* (Многослойный персептронный классификатор) – это модель искусственной нейронной сети с прямой связью, которая сопоставляет входные наборы данных с набором соответствующих выходных данных. MLP состоит из нескольких слоев, и каждый слой полностью соединен со следующим. Узлами слоев являются нейроны с нелинейными функциями активации, за исключением узлов входного слоя. Между входным и выходным слоями может быть один или несколько нелинейных скрытых слоев.
* *Наивные байесовские оценки* — это вероятностные оценки, основанные на теореме Байеса с предположениями о том, что между объектами существует сильная независимость. Теорема Байеса помогает нам определить вероятность наступления событий на основе некоторого предварительного знания условий, которые могут быть связаны с событием. Наивные байесовские классификаторы довольно хорошо работают для приложений классификации документов и фильтрации нежелательной почты. Он требует небольшого объема обучающих данных для настройки с учетом вероятностей для теоремы Байеса и поэтому работает довольно быстро. Scikit-Learn предоставляет список из 4 наивных байесовских оценок: 1) BernoulliNB - представляет собой классификатор, основанный на данных, представляющих собой многомерные распределения Бернулли. Распределение Бернулли подразумевает, что данные могут иметь несколько функций, но предполагается, что каждая из них является двоичной переменной.  
  2) GaussianNB - представляет собой классификатор, основанный на предположении, что вероятность признаков является гауссовым распределением.  
  3) ComplementNB - представляет собой классификатор, который использует дополнение каждого класса для вычисления весов модели. Это стандартный вариант многочленного наивного Байеса, который хорошо подходит для несбалансированных задач классификации классов.  
  4) MultinomialNB - представляет собой классификатор, который подходит для многомерно распределенных данных.
* *RandomForestClassifier* – это один из алгоритмов работы с учителем. Зачастую его используют для регрессии и для классификации. Этот алгоритм является одним из простых в использовании. Random Forest создаёт деревья решений для случайных sample-data. Затем делает прогноз от каждого дерева и в итоге выбирает лучший с помощью голосования. Бонусом является то, что он предоставляет мерку важности признаков. Случайный лес основан на алгоритме Борута, который определяет наиболее важные и значимые признаки датасета.

Этапы работы случайного леса:

1. Создание случайной выборки из датасета.
2. Построение деревьев для каждой выборки и получение результатов.
3. Голосование за каждый полученный прогноз.
4. Окончательным результатом будет являться ветвь с наибольшим количеством голосов.

## Библиотека Matplotlib

Matplotlib – это обширная библиотека для создания статических, анимированных и интерактивных визуализаций на Python. Matplotlib делает простые вещи простыми, а сложные возможными.

Возможности Matplotlib:

* Создание [интерактивных фигур](https://mybinder.org/v2/gh/matplotlib/mpl-brochure-binder/main?labpath=MatplotlibExample.ipynb), которые можно масштабировать, панорамировать и обновлять.
* Экспорт во [многие форматы файлов](https://matplotlib.org/stable/api/figure_api.html#matplotlib.figure.Figure.savefig).
* Богатый набор [сторонних пакетов](https://matplotlib.org/mpl-third-party/), созданных на основе Matplotlib.
* Создание [графиков качества публикации](https://ieeexplore.ieee.org/document/4160265/citations?tabFilter=papers).
* Настройка [визуального стиля](https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html) и [макета](https://matplotlib.org/stable/tutorials/provisional/mosaic.html).
* Встраивание в [JupyterLab и графические пользовательские интерфейсы](https://matplotlib.org/stable/gallery/#embedding-matplotlib-in-graphical-user-interfaces).

# ЗАКЛЮЧЕНИЕ

Поставленные задачи были реализованы в выполненной работе. Найден датасет с различными постами, содержащие текст, разбитый на различные темы и виды. Загрузка и исследование данных из датасета (приведены примеры текстов). Проведена векторизация текста, которая помогает машине конвертировать буквы в цифры для чтения и понимая слов в тексте. Создано девять различных моделей начиная с самых простых и быстрых заканчивая с самых сложных и долгих. Для моделей брались разные значения размера выборки (100, 500, 1000, 5000, 10000 элементов), но строились они с размером 10000. Время обучения у классификаторов были разные. Некоторые обучались за считанные секунды и даже доли секунд, но эффективность в некоторых моделях была минимальна. Другие более сложные модели обучались очень долго по несколько десяток минут, но гораздо эффективнее. Конечно, среди всех моделей есть те, что имеют среднюю эффективность и среднее время обучения.

Для улучшения качества обучения нашей машины стоит увеличить объём выборки и иметь большое количество объёмов данных.

Я думаю, направление классификации текстов нужно в наше время и его стоит развивать, так как существует много новостных сервисов, социальных сетей и СМИ, которые выпускают огромное количество текста, который можно исследовать и обучать машину для автоматизации сервисов.

# ИСТОЧНИКИ

1. L.P. Coelho, W. Richert. Building machine learning system with Python – Packt Publishing, 2013. – 290 с.

2. J. Grus, Data Science from Scratch: First Principles with Python – O’Reilly Media, Inc, 2015. – 330 с.

3. C. Albon, Machine Learning with Python Cookbook – O’Reilly Media, Inc, 2018 – 366 с.

4. Aurelien Geron, Hands-on Machine Learning with Sckit-Learn, Keras and Tensorflow, 2nd Edition – O’Reilly Media, Inc, 2019. – 600 с.

5. The EMNIST Dataset [Электронный ресурс] // nist.gov. 28.03.2019. URL: https://www.nist.gov/itl/products-and-services/emnist-dataset

6. Getting started with EasyOCR for Optical Character Recognition [Электронный ресурс] // pyimagesearch.com. 14.09.2020. URL: https://pyimagesearch.com/2020/09/14/getting-started-with-easyocr-for-optical-character-recognition/

7. How to OCR with Tesseract, OpenCV and Python [Электронный ресурс] // nanonets.com. 10.02.2022. URL: https:/nanonets.com/blog/ocr-with-tesseract/ (https://nanonets.com/blog/ocr-with-tesseract/)

Свёрточные нейронные сети [Электронный ресурс] // habr.com. 31.01.2018. URL: https://habr.com/ru/post/348000/.