Частное образовательное учреждение высшего образования
«Международный Институт Дизайна и Сервиса»
(ЧОУВО МИДиС)
Кафедра математики и информатики

УТВЕРЖДАЮ
Ректор
_____ М.В. Усынин
«29» мая 2023 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ Б1.Б.13 ТЕХНОЛОГИИ МАШИННОГО ОБУЧЕНИЯ И АНАЛИЗА ДАННЫХ

Направление подготовки 38.03.05 БИЗНЕС-ИНФОРМАТИКА
Направленность (профиль): Электронный бизнес
Квалификация Бакалавр
Форма обучения: Очная
Год набора - 2020

Автор-составитель: Овсяницкая Л.Ю.

Челябинск 2023

СОДЕРЖАНИЕ

1. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ С УКАЗАНИЕМ ЭТАПОВ ИХ ФОРМИРОВАНИЯ В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Процесс изучения дисциплины «Информатика» направлен на формирование следующих компетенций:

№ п/п	Шифр компетенции	Наименование компетенции	Планируемые результаты изучения учебной дисциплины
1.	ОК-7	способность к самоорганизации и самообразованию	*знать:* – методы самоорганизации и самообразования;
			уметь: − самостоятельно работать с разноплановыми источниками и научной литературой; − планировать реализацию поставленной цели; анализировать результаты деятельности;
			владеть: − навыками планирования, организации и контроля своей учебной и научной деятельности; − навыками ориентации в профессиональных источниках информации (журналы, сайты, образовательные порталы и т.д.); навыками рефлексии, самооценки, самоконтроля.

2. ПОКАЗАТЕЛИ И КРИТЕРИИ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ НА РАЗЛИЧНЫХ ЭТАПАХ ИХ ФОРМИРОВАНИЯ, ОПИСАНИЕ ШКАЛ ОЦЕНИВАНИЯ

№ п/п	Шифр компетенции	Показатели оценивания (содержание компетенции)	Критерии оценивания компетенций на различных этапах формирования	Шкала оценивания
1.	ОПК-7	Способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных	*1 Этап – знать:* – методы самоорганизации и самообразования;	Оценка «ЗАЧТЕНО»: 1. Усвоение программного материала. 2. Правильная формулировка основных определений. 3. Знание классификаций, применяемых в информатике.
			2 Этап – уметь: – самостоятельно работать с разноплановыми источниками и научной литературой; – планировать реализацию поставленной цели; – анализировать результаты деятельности	
			3 Этап – владеть:	

		требований информационной безопасности.	– навыками планирования, организации и контроля своей учебной и научной деятельности; – навыками ориентации в профессиональных источниках информации (журналы, сайты, образовательные порталы и т.д.); – навыками рефлексии, самооценки, самоконтроля.	4. Знание основных информационных показателей. 5. Свободное владение приемами и методами работы за компьютером. 6. Выполнение практического задания. Оценка «НЕ ЗАЧТЕНО»: 1. Незнание значительной части программного материала. 2. Неспособность объяснить основные информационные категории и закономерности. 3. Неумение выделить главное, сделать выводы и обобщения. 4. Грубые ошибки при выполнении практического задания.

3. ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ, НЕОБХОДИМЫЕ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ И (ИЛИ) ОПЫТА ДЕЯТЕЛЬНОСТИ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ ТИПОВЫЕ КОНТРОЛЬНЫЕ ЗАДАНИЯ ИЛИ ИНЫЕ МАТЕРИАЛЫ,
НЕОБХОДИМЫЕ ДЛЯ ОЦЕНКИ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ И (ИЛИ) ОПЫТА ДЕЯТЕЛЬНО ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ
В ПРОЦЕССЕ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

1 ЭТАП – ЗНАТЬ

Комплект тестовых и самостоятельных вопросов

Тема 1. Основные этапы и направления исследований в области систем искусственного интеллекта

1. Этапы развития систем искусственного интеллекта (ИИ).
2. Основные направления развития исследований в области систем искусственного интеллекта.
3. Big Data. Особенности работы с большими данными.
4. Извлечение знаний. Интеграция знаний. Базы знаний. Примеры.

Тема 2. Основы машинного обучения и анализа данных

1. Основы программирования для задач анализа данных.
2. Библиотеки Python для анализа данных.
3. Изучение отдельных направлений анализа данных. Обучение «с учителем», «без учителя», «с подкреплением».
4. Решение задач классификации, кластеризации, регрессии.

Тема 3. Нейронные сети

1. Принцип работы мозга. История открытия нейронов и нейронных сетей.
2. Искусственные нейросети. Схемы и принцип работы.
3. Полносвязные нейронные сети. Однослойный и многослойный персептрон.
4. Глубокое обучение.
5. Различные архитектуры нейросетей.

Тема 4. Обработка естественного языка

1. Извлечение информации. Информационный поиск.
2. Анализ высказываний. Анализ тональности текста.
3. Вопросно-ответные системы. Генерирование текста.
4. Естественно-языковой интерфейс.

Тема 5. Цифровые сервисы создания рекламы, рассылок, опросов и маркетинг в социальных сетях для решения задач профессиональной деятельности

1. Введение в компьютерное зрение.
2. Распознавание изображений людьми. Признаки для категоризации изображений.
3. Возможности библиотеки OpenCV. Машинное обучение в OpenCV.
4. Архитектуры нейросетей для распознавания изображений.

Вопросы тестирования по теоретическим основам:

1. Метод, позволяющий предсказывать значения той или иной непрерывной числовой величины для входных данных называется:

Кластеризация

Классификация

Регрессия

Метод опорных векторов

2. Какой из видов машинного обучения основывается на взаимодействии обучаемой системы со средой?

С учителем
Без учителя
С подкреплением
Глубинное

3. В какие игры нейросеть еще не научилась обыгрывать человека?
Го
Бридж
Шахматы
"Марио"

4. Метод, позволяющий прогнозировать выходы с двумя возможными значениями, помеченными как «0» или «1», называется:
Логистическая регрессия
Метод опорных векторов
Метод k-ближайших соседей
Нейросеть

5. Ошибку 1-го рода иногда называют:
Точность модели
Ложная тревога
Вероятность отказа
Пропуск цели

6. Ошибку 2-го рода иногда называют:
Пропуск цели
Ложная тревога
Вероятность отказа
Точность модели

7. Доля объектов, названных положительными и являющиеся положительным, отражает метрика:
Recall
Precision
Accuracy
Exactly

8. Доля объектов положительного класса из всех объектов положительного класса определяется метрикой
Ошибка 1-го рода
Ошибка 2-го рода
Ошибка 3-го рода
Ошибка 4-го рода

9. "Пропуск" уходящего абонента и ошибочное принятие нулевой гипотезы называется:

Разделяющие
Опорные
Решающие
Гипервектора

10. Под «соседями» в методе k-NN понимаются:
Параметры модели, лучше всего описывающие объект
Рядом находящиеся объекты
Объекты, близкие к исследуемому в том или ином смысле
Объекты, находящиеся напротив исследуемого объекта

11. Кластеризация относится к методу обучения:
С учителем
Без учителя
С подкреплением
С предсказанием

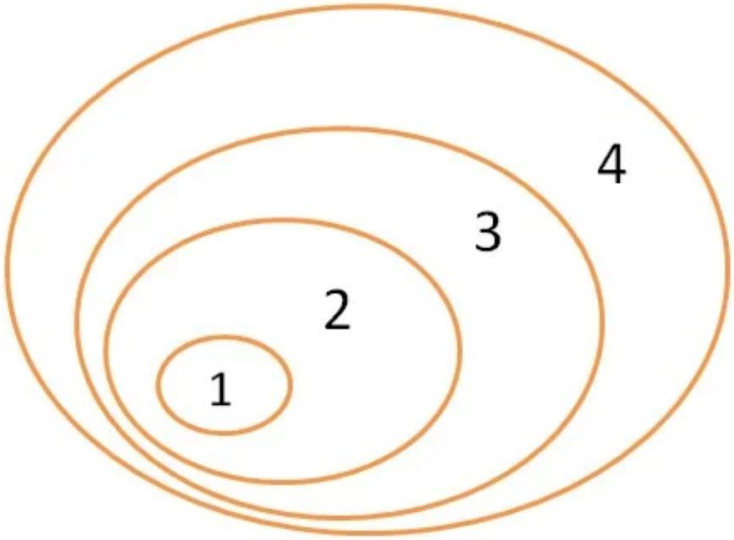
12. Классификация относится к методу обучения:
С учителем
Без учителя
С подкреплением
С предсказанием

13. В каком году был определен термин "искусственный интеллект"?
1945
1956
1981
1990

14. В чем заключается цель машинного обучения?
Предсказать входные данные по результату
Предсказать результат по входным данным
Обучить машину ездить без водителя
Проводить арифметические вычисления

15. Обучение, основанное на маркированных обучающих данных, называется:
Обучение с учителем
Обучение без учителя
Обучение с подкреплением
Машинное обучение

16. Укажите правильное соответствие цифр на рисунке и терминам

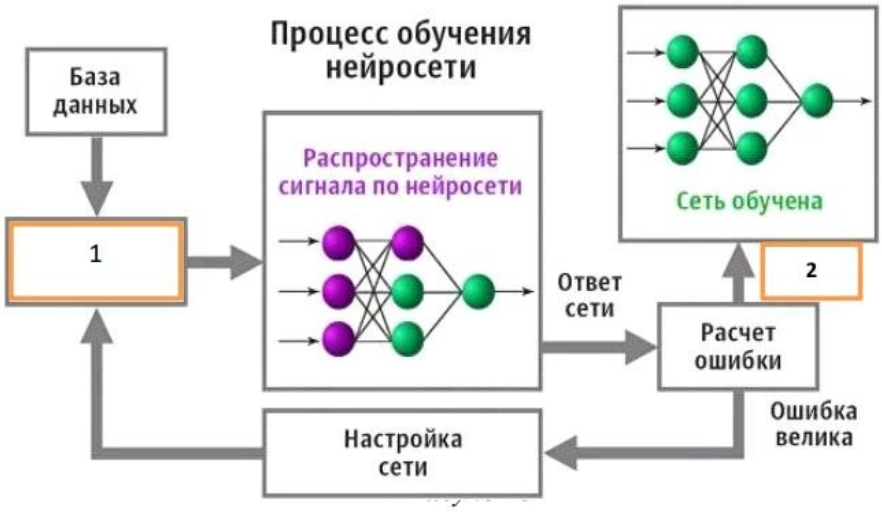


1 - нейросети, 2- машин. обучение, 3-искусственный интеллект, 4 - мозг
1 - статистика, 2- машин. обучение, 3-математика, 4 - наука
1 - глубокое обучение, 2- искусственный интеллект, 3 - нейросети, 4 - модели
1 - глубокое обучение, 2-нейросети, 3- машин. обучение, 4-иск. интеллект

17. Что называется обучением нейронной сети?

процесс настройки синаптических весов
процесс получения результата арифметических действий
процесс присвоения нейронам маркеров выполнения
процесс программирования искусственного интеллекта

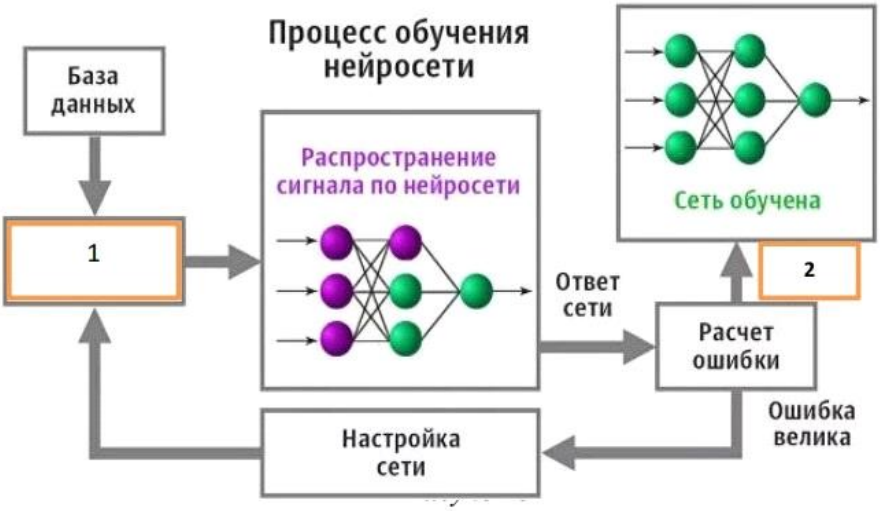
18. Что должно быть написано в схеме обучения нейронной сети методом обратного распространения ошибки вместо цифры 1?



выбор примера
ошибка мала
ошибка велика

ошибки нет

19. Что должно быть написано в схеме обучения нейронной сети методом обратного распространения ошибки вместо цифры 2?



выбор примера
ошибка мала
ошибка велика
ошибки нет

20. Библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых (и очень быстрых) математических функций для операций с этими массивами, называется:
pandas
numpy
matplotlib
sklearn

21. Библиотека Python, позволяющая строить сводные таблицы, выполнять группировки, предоставляет удобный доступ к табличным данным, называется:
pandas
numpy
matplotlib
sklearn

22. Библиотека Python, которая предоставляет множество возможностей , таких как многоступенчатый анализ, регрессия и алгоритмы кластеризации, называется:
pandas
numpy
matplotlib
sklearn

23. Библиотека Python, предназначенная для визуализации данных, называется

pandas
numpy
matplotlib
sklearn

24. Метод, который задает начальные условия для генератора случайных чисел, называется:
random.seed()
np.median()
np.median()
np.arange()

25. "Надстройка» над Matplotlib, которая предоставляет лучшую графику и большее количество возможностей её настройки, называется
plot
graphic
seaborn
diagrams

26. Открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети, называется:
NeiroNet
NeiroLib
TensorFlow
FlowKeras

27. В каком отношении обцчно далят выборку на обучающую и тестоую:
20:80
80:20
50:50
90:10

28. Что называют "эпохой" в нейросетях?
Поколение создания искусственной нейросети
Одна итерация в процессе обучения нейросети
Событие, повлиявшее на развитие нейросети
Процесс расчета ошибки нейросети

29. Что такое "переобучение" модели?
Модель содержит чрезмерно большое число переменных
Модель слишком часто участвовала в обучении
Модель излишне точно соответствует сети конкретному набору обучающих примеров и теряет способность к обобщению.

30. Для решения каких задач создана свёрточная нейронная сеть?
Решение сложных арифметических задач
Предсказание временных рядов
Распознавание образов
Архивирование больших данных

31. Какого слоя нет в архитектуре сверточной нейросети?
Сверточного
Полносвязного
Выходного
Промежуточного

32. Среда программирования Python, включающая набор свободных библиотек машинного обучения, называется:

Cobra
Anaconda
MachineLearning
PythonMLLybrary

33. Инструмент для разработки и представления проектов Data Science в интерактивном виде, объединяющий код, текст, математические уравнения и визуализаци, называется:

Mars Notebook
Jupyter Notebook
Venera Notebook
Pluton Notebook

34. Датасет - это
Сет данных
Обработанная и структурированная информация в табличном виде
Усредненная выборка данных по строкам
Усредненная выборка данных по столбцам

35. Как называется бесплатная среда Google для создания ноутбуков Jupyter, которая полностью работает в облаке?
Laboratory
Googlaboratory
Colaboratory
Neirabolatory

36. Какой тип ячеек в ноутбуках Jupyter предназначен для ввода текста и изображений?
Code
Markdown
Memo

Image

37. Какой тип ячеек в ноутбуках Jupyter предназначен для ввода программного кода?
Markdown
Memo
Image
Code

38. С помощью каких символов в Jupyter ноутбук можно создавать заголовки первого уровня?
С помощью символа # и пробела
С помощью символа #
С помощью символа % и пробела
С помощью символа %

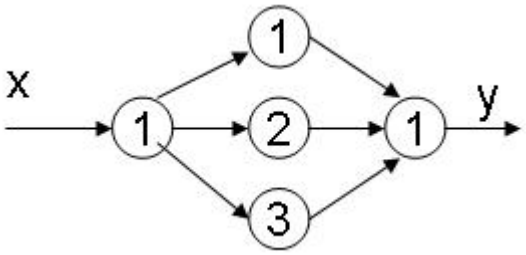
39. С помощью каких символов в Jupyter ноутбук можно создавать шрифт курсив?
с помощью символов * с двух сторон текста
с помощью символов # с двух сторон текста
с помощью символов «K» с двух сторон текста
с помощью символов «/ *» и «*/» с двух сторон текста

40. Какое сочетание клавиш запускает код в Jupyter ноутбук на выполнение?
Enter
Shift+Enter
Ctrl+Shift
Alt+Shift

41. Аксон – это отросток нейрона:
Входной
Выходной
Промежуточный
Преобразующий

42. Что в биологическом нейроне имеет бОльшую длину:
Дендрит
Аксон
Синапс
Тело нейрона

43. Дано: нейронная сеть с одним скрытым слоем. У сети 1 вход, 3 нейрона в скрытом слое и один выход. Что будет на выходе сети в случае, если на входе 1, все веса раны 1?

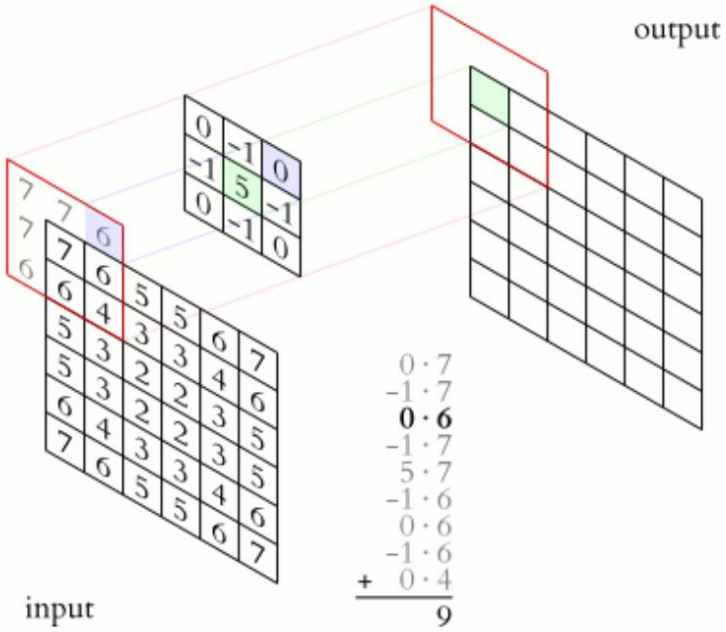


1
3
1/3
0,3

44. Кто создал первую модель искусственных нейронных сетей?

Мак-Каллок и Питтс
Дэвид И. Румельхарт, Дж. Е. Хинтон и Рональд Дж. Вильямс
Фрэнк Розенблатт
Ян Лекун

45. Какой тип искусственной нейронной сети представлен на картинке?



Рекуррентная нейронная сеть
Нейронная сеть Джордана
Матричная нейронная сеть
Сверточная нейронная сеть

46. Для распределенного глубокого машинного обучения (Deep Learning) больше подходит фреймворк
PyTorch
TensorFlow
Flask

Scikit-learn

47. Большие данные – это:

Данные объемом более 1Тб
Данные объемом более 10Тб
Данные объемом более 100Тб
Нет ограничений на минимальный объем

48. Средняя абсолютная ошибка (MAE) получается путем:
вычисления абсолютной разницы между прогнозами модели и истинными (фактическими) значениями.
вычисления относительной разницы между прогнозами модели и истинными (фактическими) значениями.
вычисления квадрата разницы между прогнозами модели и обучающим набором данных (истинные значения).
вычисления квадрата разницы между прогнозами модели и обучающим набором данных (истинные значения) и представления результата в процентном формате.

49. Среднеквадратичная ошибка (MSE) получается путем:
вычисления абсолютной разницы между прогнозами модели и истинными (фактическими) значениями.
вычисления суммы квадратов разницы между прогнозами модели и обучающим набором данных (истинные значения).
вычисления относительной разницы между прогнозами модели и истинными (фактическими) значениями.
вычисления квадрата разницы между прогнозами модели и обучающим набором данных (истинные значения) и представления результата в процентном формате.

50. Средняя абсолютная процентная ошибка (MAPE) получается путем:
вычисления суммы квадратов разницы между прогнозами модели и обучающим набором данных (истинные значения) и представления результата в процентном формате.
вычисления суммы квадратов разницы между прогнозами модели и обучающим набором данных (истинные значения).
вычисления абсолютной разницы между прогнозами модели и истинными (фактическими) значениями.
вычисления относительной разницы между прогнозами модели и истинными (фактическими) значениями.

2 ЭТАП – УМЕТЬ

Комплект практических работ

Практические работы и семинары служат для работы студентов над учебными задачами с целью выработки и закрепления практических навыков.

Тема 1. Основные этапы и направления исследований в области систем искусственного интеллекта.

Задание: изучение основных способов представления данных: продкционные модели, семантические сети, фреймы.

Используя соответствующие дуги построить семантическую сеть (по вариантам):

1. Географии региона. Дуги: государство, страна, континент, широта.
2. Процедуры поиска полезных ископаемых. Дуги: наименование ископаемого, расположение месторождения, глубина залегания, методы добычи.
3. Распределения продуктов по магазинам. Дуги: источник снабжения, наименование продукта, способ транспортировки, конечный пункт транспортировки.
4. Определение принадлежности животного к определенному виду, типу, семейству. Дуги: место обитания, строение, особенности поведения, вид питания.
5. Классификации пищевых продуктов. Дуги: наименование продукта, составляющие части, способ приготовления, срок хранения.
6. Распознавание типа компьютера. Дуги: страна изготовитель, стандартная конфигурация, область применения, используемое программное обеспечение.
7. Иерархической структуры БД. Дуги: система, состояние, назначение, взаимодействие составляющих.

Тема 2. Основы машинного обучения и анализа данных

1. Задание: изучить основы программирования на Python. Основные библиотеки машинного обучения.

Выполнение задания на тему «Линейная регрессия»¶

Постановка задачи¶

Необходимо предсказывать доход от продажи мороженого в зависимости от температуры воздуха. Мы предполагаем, что линейная регрессия позволит решить эту задачу.

Шаг #1: импорт библиотек¶

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Шаг #2: импорт датасета¶

```
# загружаем в Colab с диска ПК.
# from google.colab import files
# files.upload()
# создаем объект - датафрейм - для Colab
# IceCream = pd.read_csv("IceCreamData.csv")
```

считываем первые 10 значений

```
IceCream.head(10)
Out[5]:
```

	Temperature	Revenue
0	24.566884	534.799028
1	26.005191	625.190122
2	27.790554	660.632289
3	20.595335	487.706960
4	11.503498	316.240194
5	14.352514	367.940744
6	13.707780	308.894518
7	30.833985	696.716640
8	0.976870	55.390338
9	31.669465	737.800824

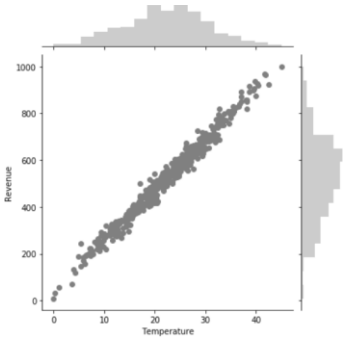
```
In [6]:
# считываем последние 10 значений
IceCream.tail(10)
```

	Temperature	Revenue
490	23.824922	584.399945
491	34.472169	809.352520
492	23.056214	552.819351
493	14.931506	377.430928
494	25.112066	571.434257
495	22.274899	524.746364
496	32.893092	755.818399
497	12.588157	306.090719
498	22.362402	566.217304
499	28.957736	655.660388

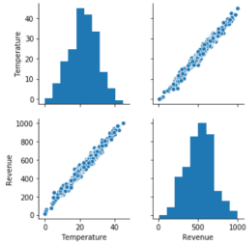
```
# основные сведения описательной статистики: кол-во, среднее, среднеквадратиче-
ское отклонение,
# минимальное значение
IceCream.describe()
```

	Temperature	Revenue
count	500.000000	500.000000
mean	22.232225	521.570777
std	8.096388	175.404751
min	0.000000	10.000000
25%	17.122258	405.558681
50%	22.392791	529.368565
75%	27.740674	642.257922
max	45.000000	1000.000000

```
# то же, но по отдельному столбцу
IceCream['Temperature'].describe()
Out[8]:
count    500.000000
mean      22.232225
std        8.096388
min        0.000000
25%       17.122258
50%       22.392791
75%       27.740674
max       45.000000
Name: Temperature, dtype: float64
In [9]:
# получаем краткие сведения о данных
IceCream.info()
RangeIndex: 500 entries, 0 to 499
Data columns (total 2 columns):
Temperature    500 non-null float64
Revenue        500 non-null float64
dtypes: float64(2)
memory usage: 7.9 KB
Шаг#3: визуализация датасета¶
https://coincase.ru/blog/47592/¶
# строим гибридный двумерный график
sns.jointplot(x='Temperature', y='Revenue', data = IceCream, color = 'gray')
Out[10]:
<seaborn.axisgrid.JointGrid at 0x19761326ef0>
```




```
# Второй способ
sns.pairplot(IceCream)
# диаграмма рассеяния с дополнительной наложенной линией регрессии
sns.lmplot(x='Temperature', y='Revenue', data=IceCream)
```



Шаг #4: разбиение датасета на обучающую и тестовую выборку¶

In [13]:

```
y = IceCream['Revenue']
```

In [14]:

```
X = IceCream[['Temperature']]
```

In [15]:

```
X
```

Out[15]:

	Temperature
0	24.566884
1	26.005191
2	27.790554
3	20.595335
4	11.503498
5	14.352514
498	22.362402
499	28.957736

500 rows × 1 columns

In [16]:

```
# импортируем функцию train_test_split
from sklearn.model_selection import train_test_split
```

In [17]:

```
# функция train_test_split принимает аргументы X_train, X_test, y_train, y_test
# внутри задаем процент тестовой выборки (обычно 25% или 20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

In [18]:

```
X_train
```

Out[18]:

	Temperature
462	12.123014
489	26.964217
210	22.387604
487	32.632858
17	42.515280
406	17.997015
417	27.516646

375 rows × 1 columns

Шаг #5: обучение модели¶

In [19]:

```
# проверим размерность
X_train.shape
```

Out[19]:

```
(375, 1)
```

In [20]:

```
# импорт метода линейной регрессии
from sklearn.linear_model import LinearRegression
```

In [21]:

```
# Следующая операция создаёт переменную model в качестве экземпляра Line-
arRegression.
# Опциональные параметры класса LinearRegression:
#  fit_intercept – логический (True по умолчанию) параметр, который решает,
# вычислять отрезок b0 - от 0 до реальных начальных значений (True) или рассмат-
ривать его как равный нулю (False).
# normalize – логический (False по умолчанию) параметр, который решает, нормали-
зовать входные переменные (True)
# или нет (False).
regressor = LinearRegression(fit_intercept = True)
```

In [22]:

```
# обучение модели - вычисление коэффициентов
regressor.fit(X_train,y_train)
```

Out[22]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [23]:

```
# печать коэффициентов
print('Linear Model Coefficient (m): ', regressor.coef_)
print('Linear Model Coefficient (b): ', regressor.intercept_)
Linear Model Coefficient (m):  [21.42821956]
Linear Model Coefficient (b):  44.69044402743077
```

Шаг #6: Тестирование модели¶

In [24]:

```
y_predict = regressor.predict( X_test)
y_predict
```

Out[24]:

```
array([441.29165419, 797.2527597 , …, 654.03543779,
       732.6880014 ])
```

In [25]:

```
y_test
```

Out[25]:

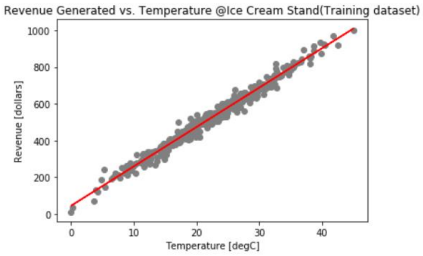
```
164    726.233771
96     474.749392
          ...
108    643.788331
169    773.924755
Name: Revenue, Length: 125, dtype: float64
```

In [26]:

```
plt.scatter(X_train, y_train, color = 'gray')
plt.plot(X_train, regressor.predict(X_train), color = 'red')
plt.ylabel('Revenue [dollars]')
plt.xlabel('Temperature [degC]')
plt.title('Revenue Generated vs. Temperature @Ice Cream Stand(Training da-
taset)')
```

Out[26]:

```
Text(0.5, 1.0, 'Revenue Generated vs. Temperature @Ice Cream Stand(Training da-
taset)')
```



Самостоятельная работа по теме "Линейная регрессия"

Условие задание:

Вы являетесь консультантом крупного производителя автомобилей. Вам было поручено разработать модель для прогнозирования влияния увеличения мощности автомобиля (л.с.) на экономию топлива (пробег миль на галлон (MPG)).

Вы собрали данные:

Независимая переменная X: мощность транспортного средства в лошадиных силах.

Зависимая переменная Y: Пробег миль на галлон (MPG).

Задание:

по аналогии с лабораторной работой, в которой рассчитывалась прибыль от продажи мороженого в зависимости от температуры воздуха, проанализируйте полученные данные, визуализируйте их и постройте модель линейной регрессии. Используйте датасет из файла FuelEconomy.csv.

Тема 3.

Написание реферата является обязательным элементом работы студентов в рамках освоения курса «Обработка естественного языка и компьютерное зрение». Реферат (от лат. «refero» - докладываю, сообщаю) - это самостоятельная исследовательская работа, в которой автор раскрывает суть исследуемой проблемы; приводит различные точки зрения, а также собственные взгляды не нее. Содержание реферата должно быть логичным; изложение материала носит проблемно-тематический характер.

Темы для рефератов:

1. Классификация и виды нейронных сетей.
2. Методы ускорения обучения нейронной сети.
3. Многослойные полносвязные нейронные сети.
4. Нейронные сети для обнаружения вредоносного программного обеспечения.
5. Нейронные сети для анализа финансового рынка.
6. Нейронные сети для распознавания текстов и голоса.
7. Сверточные нейронные сети. Назначение и история создания.
8. Реккурентные нейронные сети. Назначение и история создания.
9. Нейронные сети для распознавания образов.
10. Применение нейронных сетей в экономике и бизнесе.
11. Применение нейронных сетей в медицине.
12. Применение нейронных сетей в автоматизации и робототехнике.
13. Применение нейронных сетей в системах безопасности и охранных системах.
14. Применение нейронных сетей в компьютерных играх.

Структура реферата:

1) Ключевые слова.
2) Аннотация содержания (2-3 предложения).
3) Введение (не более 2 страниц). Во введении необходимо обосновать актуальность темы, очертить область исследования, объект исследования, основные цели и задачи исследования, сформулировать выдвигаемые гипотезы.

4) Основная часть состоит из 2-3 разделов. В них раскрывается суть исследуемой проблемы, проводится обзор информации по предмету исследования. Изложение материала не должно ограничиваться лишь описательным подходом к раскрытию выбранной темы. Оно также должно содержать собственное видение рассматриваемой проблемы.

5) Заключение (1-2 страницы). В заключении кратко излагаются выводы, а также предполагаемые научные результаты и прогнозы.

6) Библиографический список (от 5 до 10 источников) в алфавитном порядке. В данный список рекомендуется включать работы отечественных и зарубежных авторов Библиографический список содержит только те произведения, на которые есть сноски в тексте.

7) Приложение (при необходимости).

Создание презентации по заданной теме

Мультимедийные презентации используются для того, чтобы обучающийся смог наглядно продемонстрировать визуальные (аудио, видео, графические) материалы, освоенные в ходе самостоятельной и практической работы по предмету.

Общие требования к презентации:

Презентация не должна быть меньше 10 слайдов.

Первый слайд – титульный лист, на котором обязательно должны быть представлены: тема; фамилия, имя, автора, номер учебной группы;

Второй слайд – содержание, где представлены основные вопросы разобранные в ходе изучения темы. Желательно, чтобы из содержания по гиперссылке можно перейти на необходимую страницу и вернуться вновь на содержание.

В структуре презентации необходимо использовать: графическую и анимационную информацию: видео и аудио фрагменты, таблицы, диаграммы, инфографику и т.д.

Последний слайд демонстрирует список ссылок на, используемые информационные ресурсы.

Тема 4. Обработка естественного языка

Практическая работа «Детектирование спама»

Коллекция SMS или emeil спама - это набор сообщений с тегами, которые были собраны для исследования SMS-спама. Он содержит набор SMS-сообщений на английском языке, состоящий из 5 574 сообщений, помеченных как ham - «законный» или spam - «спам».

Файлы содержат одно сообщение в строке. Каждая строка состоит из двух столбцов: v1 включает метку (ham или spam), а v2 содержит необработанный текст.

Задача: создать модель, позволяющую определять на основе анализа текста, относится оно к спаму или не содержит подозрительных конструкций.

Шаг #1: Импорт библиотек и датасета¶

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
spam_df = pd.read_csv("emails.csv")
# Смотрим начало и окончание датасета
spam_df.head(10)
```

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1

	text	spam
4	Subject: do not have money , get software cds ...	1
5	Subject: great nnews hello , welcome to medzo...	1

spam_df.tail(5)

	text	spam
5723	Subject: re : research and development charges...	0
5724	Subject: re : receipts from visit jim , than...	0
5725	Subject: re : enron case study update wow ! a...	0
5726	Subject: re : interest david , please , call...	0
5727	Subject: news : aurora 5 . 2 update aurora ve...	0

#Посмотрим описание выборки: общее количество значений и количество ненулевых значений

spam_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5728 entries, 0 to 5727
Data columns (total 2 columns):
text 5728 non-null object
spam 5728 non-null int64
dtypes: int64(1), object(1)
memory usage: 89.6+ KB

Шаг #2: Визуализация датасета¶

ham -это не спам. Слово используется для более быстрого произнесения и написания "no-spam"

ham = spam_df[spam_df['spam']==0]
spam = spam_df[spam_df['spam']==1]
# Выводим сообщения, помеченные как не спам
ham

	text	spam
1368	Subject: hello guys , i ' m " bugging you " f...	0
1369	Subject: sacramento weather station fyi - - ...	0
1370	Subject: from the enron india newsdesk - jan 1...	0
1371	Subject: re : powerisk 2001 - your invitation ...	0
1372	Subject: re : resco database and customer capt...	0
...	...	...
5727	Subject: news : aurora 5 . 2 update aurora ve...	0

4360 rows × 2 columns

Выводим сообщения, помеченные как спам
spam

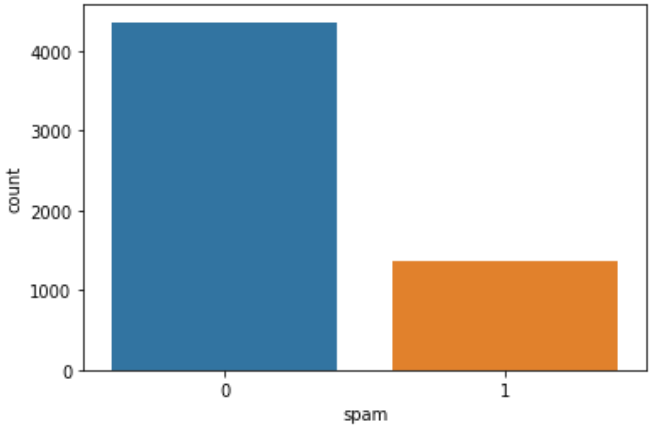
	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
...	...	...
1363	Subject: are you ready to get it ? hello ! v...	1

1368 rows × 2 columns

# Вычисляем процент писем, содержащих спам
print('Spam percentage =', (len(spam) / len(spam_df))*100,"%")
Spam percentage = 23.88268156424581 %

Вычисляем процент писем, НЕ содержащих спам. Самостоятельно: вычислите другим способом.

print('Ham percentage =', (len(ham) / len(spam_df))*100,"%")
Ham percentage = 76.11731843575419 %
# Визуализируем результат
sns.countplot(spam_df['spam'], label = "Count")
<matplotlib.axes._subplots.AxesSubplot at 0xd06c448>

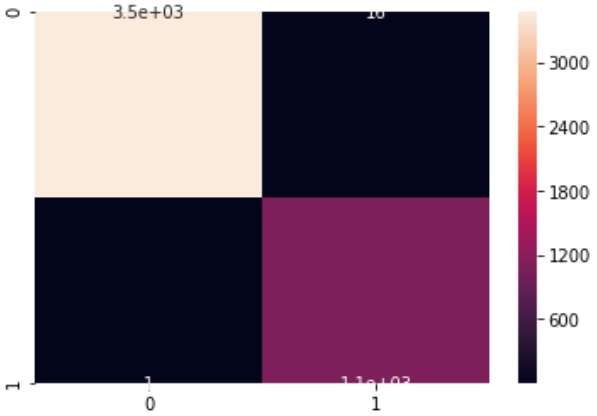


Шаг #3: Создание тестовой и обучающей выборки¶
Пример применения способа извлечения и кодирования текстовых данных COUNT VECTORIZER¶
# CountVectorizer преобразовывает входной текст в матрицу, значениями которой
# являются количества вхождения данного ключа(слова) в текст.
# Приведем простой пример. Допустим есть массив sample_data текстовых значений:
# ['This is the first document.','This document is the second document.',
# 'And this is the third one.','Is this the first document?']
# Ниже значения для удобства написаны в столбец.
from sklearn.feature_extraction.text import CountVectorizer
sample_data = ['This is the first document.',
'This document is the second document.',
'And this is the third one.',
'Is this the first document?']

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(sample_data)
# В первую очередь CountVectorizer собирает уникальные ключи (слова) из всех записей, в нашем примере это будет:
# ['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']. Сортировка по алфавиту.
print(vectorizer.get_feature_names())
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
# Длина списка из уникальных ключей (слов) и будет длиной нашего закодированного текста
# (в нашем случае это 4). А номера элементов будут соответствовать, количеству раз встречи данного ключа
# с данным номером в строке. Соответственно после кодировки и применения данного метода мы получим:
print(X.toarray())
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
Применим COUNT VECTORIZER к нашей задаче¶
from sklearn.feature_extraction.text import CountVectorizer

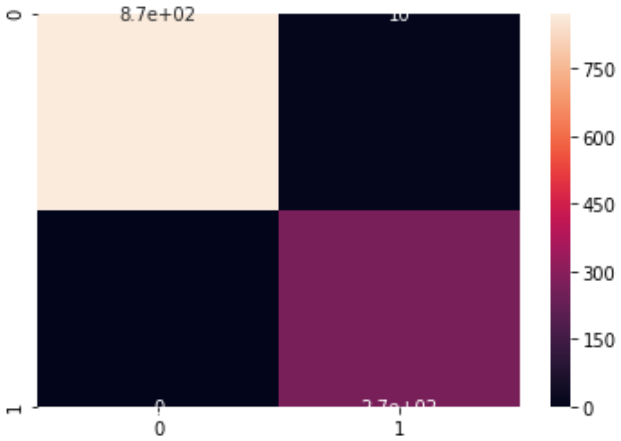
vectorizer = CountVectorizer()
spamham_countvectorizer = vectorizer.fit_transform(spam_df['text'])
In [58]:
# Выводим уникальные ключи
print(vectorizer.get_feature_names())
['00', '000', '0000', '000000', '00000000', '0000000000', '000000000003619', '000000000003991', '000000000003997', ,… 'duenner', 'dues', 'duet', 'duffee', 'duffer', 'duffie', 'dugout', 'duhon', 'duit', 'duke', 'dull', 'duluth', …ky', 'kollaros', 'kolle', …signers', 'signifiantly', 'significance', … 'zzn', 'zzncac-st', 'zzzz']
# Печатаем матрицу текста
print(spamham_countvectorizer.toarray())
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [4 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
Смотрим размерность матрицы: 5728 строк, 37303 столбцов (= числу уникальных значений)
spamham_countvectorizer.shape
 (5728, 37303)
Шаг #4: обучение модели на всем датасете¶
from sklearn.naive_bayes import MultinomialNB
NB_classifier = MultinomialNB()
# Нужно задать параметр - классы, к которым будем относить (классифицировать) результат. Целевой класс указан в столбце Спам
label = spam_df['spam'].values
NB_classifier.fit(spamham_countvectorizer, label)
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
# Задаем тестовый пример и строим матрицу ключей:
testing_sample = ['Free money!!!', "Hi Kim, Please let me know if you need any further information. Thanks"]
testing_sample_countvectorizer = vectorizer.transform(testing_sample)
print(testing_sample_countvectorizer.toarray())
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
#Используем нашу натренированную модель и подставляем в неё полученную тестовую матрицу
# Результат: 1-е выражение спам, второе - не спам. (1 и 0)
test_predict = NB_classifier.predict(testing_sample_countvectorizer)
test_predict
array([1, 0], dtype=int64)
# Тестируем разные варианты - не спам
testing_sample = ['Hello, I am Ryan, I would like to book a hotel']
testing_sample_countvectorizer = vectorizer.transform(testing_sample)
test_predict = NB_classifier.predict(testing_sample_countvectorizer)
test_predict
array([0], dtype=int64)
# Проверяем вариант 2 - оба выражения спам
testing_sample = ['Hello, do you want to buy coffee?', 'free massage']
testing_sample_countvectorizer = vectorizer.transform(testing_sample)

test_predict = NB_classifier.predict(testing_sample_countvectorizer)
test_predict
array([1, 1], dtype=int64)
Самостоятельно: введите несколько своих выражений и проверьте, к какому классу их отнесет классификатор Наивный Байес¶
Шаг #4: Делим данные на обучающую и тестовую выборку перед обучением модели¶
X = spamham_countvectorizer
y = label
In [93]:
X.shape
(5728, 37303)
y.shape
(5728,)
# Обучение обучающей и тестовой выборки 80:20
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
from sklearn.naive_bayes import MultinomialNB
NB_classifier = MultinomialNB()
NB_classifier.fit(X_train, y_train)
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
# from sklearn.naive_bayes import GaussianNB
# NB_classifier = GaussianNB()
# NB_classifier.fit(X_train, y_train)
Шаг #5: улучшение модели¶
# Построим матрицу ошибок при проверке модели на обучающей выборке
from sklearn.metrics import classification_report, confusion_matrix
y_predict_train = NB_classifier.predict(X_train)
y_predict_train
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
# Прекрасный результат
cm = confusion_matrix(y_train, y_predict_train)
sns.heatmap(cm, annot=True)
<matplotlib.axes._subplots.AxesSubplot at 0xcc2fe48>



print(classification_report(y_train, y_predict_train))

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3481
1	0.99	1.00	0.99	1101
accuracy			1.00	4582




```
   macro avg       0.99      1.00      0.99      4582
weighted avg       1.00      1.00      1.00      4582
# Построим матрицу ошибок при проверке модели на тестовой выборке
y_predict_test = NB_classifier.predict(X_test)
cm = confusion_matrix(y_test, y_predict_test)
sns.heatmap(cm, annot=True)
<matplotlib.axes._subplots.AxesSubplot at 0xcc83548>
In [108]:
print(classification_report(y_test, y_predict_test))
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       879
           1       0.96      1.00      0.98       267
    accuracy                           0.99      1146
   macro avg       0.98      0.99      0.99      1146
weighted avg       0.99      0.99      0.99      1146
```

Тема 5. Компьютерное зрение

Формулировка задачи

В данной лабораторной работе мы познакомимся с архитектурой сверточных нейросетей LeNet.
Цель лабораторной работы: обучить нейросеть для решения задач классификации дорожных знаков.
Данная задача необходима для работы беспилотных автомобилей, причем распознавание знаков должно проводиться практически мгновенно. Набор данных содержит 43 различных класса изображений.
Классы перечислены ниже:
0 - Ограничение скорости (20 км / ч)
1 - Ограничение скорости (30 км / ч)
2 - Ограничение скорости (50 км / ч)
3 - Ограничение скорости (60 км / ч)
4 - Ограничение скорости (70 км / ч)
5 - Ограничение скорости (80 км / ч)
6 - Конец ограничения скорости (80 км / ч)
7 - Ограничение скорости (100 км / ч)
8 - Ограничение скорости (120 км / ч)

9 - Обгон запрещен
10 - Запрещается проезд для транспортных средств более 3,5 т
11 - Проезд на следующем перекрестке
12 - Главная дорога
13 - Уступи дорогу
14 - Стоп
15 - Запрещен проезд транспортных средств
16 - Запрещен въезд транспортных средств более 3,5 т
17 - Въезд запрещен
18 - Внимание
19 - Крутой поворот налево
20 - Крутой поворот направо
21 - Двойной поворот
22 - Ухабы
23 - Скользкая дорога
24 - Сужение справа
25 - Дорожные работы
26 - Светофор
27 - Пешеходы
28 - Дети
29 - Пересечение с велосипедной дорогой
30 - Остерегайтесь льда / снега
31 - Дикие животные
32 - Конец всех ограничений
33 - Поворот направо
34 - Поворот налево
35 - Проезд только прямо
36 - Проезд прямо или направо
37 - Проезд прямо или налево
38 - Придерживайтесь правой стороны
39 - Придерживайтесь левой стороны
40 - Круговое движение
41 - Конец зоны ограничения проезда
42 - Конец запрета проезда транспортных средств более 3,5 т


```
# подкоючаем Google диск
from google.colab import drive
drive.mount('/content/drive')
Mounted at /content/drive
Шаг #0: Импорт библиотек¶
import tensorflow
import matplotlib.pyplot as plt
import numpy as np
```

```
#import os
#import PIL
from tensorflow.keras import layers
import pandas as pd
import seaborn as sns
import pickle
```

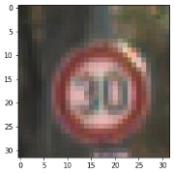
Шаг #1: Импорт и нормализация датасета¶

Файл с расширением .p представляет собой файл pickle, модуль Python, используемый для преобразования объектов Python в последовательность байтов для хранения на диске или передачи по сети. Он позволяет удобно хранить или передавать объекты без предварительного преобразования данных в другой формат.

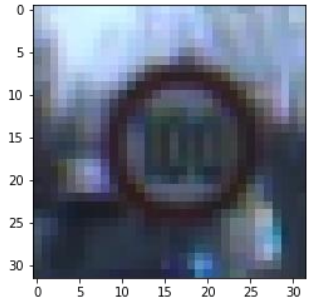
```
! Использование ключевого слова with при работе с файловыми объектами
позволяет правильно закрыть объект после завершения работы с ним. 'rt' -
атрибут, задающий формат чтения в текстовом режим.
! Функция pickle.loads() возвращает восстановленную иерархию объектов из
строкового представления данных.
with open('/content/drive/My Drive/Colab Notebooks/Sign Images/train.p',
mode='rb') as training_data:
    train = pickle.load(training_data)
with open("/content/drive/My Drive/Colab Notebooks/Sign Images/valid.p",
mode='rb') as validation_data:
    valid = pickle.load(validation_data)
with open("/content/drive/My Drive/Colab Notebooks/Sign Images/test.p",
mode='rb') as testing_data:
    test = pickle.load(testing_data)
# обозначаем обучающие, тестовые и проверочные данные датачета
X_train, y_train = train['features'], train['labels']
X_validation, y_validation = valid['features'], valid['labels']
X_test, y_test = test['features'], test['labels']
# оцениваем размерность - 34800 записей, изображение 35*32 пикселя,
цветное (3 - значит тензор RGB)
X_train.shape
(34799, 32, 32, 3)
y_train.shape
 (34799,)
```

Шаг #2: Визуализация датасета¶

```
# подставляем i = случайное число. Функцию matplotlib imshow отображает
графики на основе 2-D массивов (ч/б изображение
# или 3-D массивов (цветное). Важно! Каждый элемент в массиве действует
как пиксель.
i = 3100
plt.imshow(X_train[i])
y_train[i]
1
```

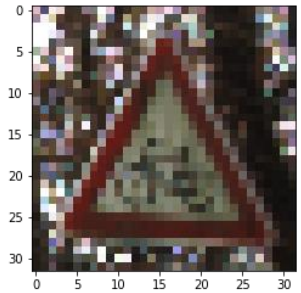


```
i = 3001
plt.imshow(X_validation[i])
y_validation[i]
7
```



```
i = 2100
plt.imshow(X_test[i])
y_test[i]
```

29



Шаг #3: Подготовка данных¶

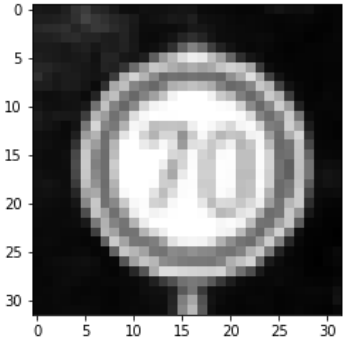
```
# sklearn.utils.shuffle() используется для перемешивания массивов случайным образом, чтобы была возможность получать разные изображения каждый раз, когда мы проводим обучение, чтобы мы не тренировались на одних и тех же изображениях.
from sklearn.utils import shuffle
X_train, y_train = shuffle(X_train, y_train)
```

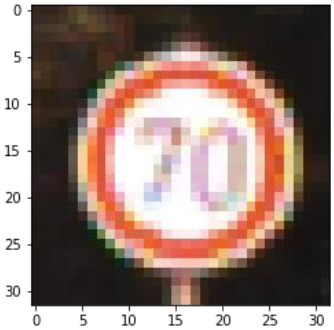
Далее нужно нормализовать набор данных, поскольку денормализованные данные ухудшают качество распознавание. Для этого преобразуем изображение в ч/б формат. Оттенки серого получим как сумму значений RGB слоёв, деленную на 3.

```
X_train_gray = np.sum(X_train/3, axis = 3, keepdims = True)
X_test_gray = np.sum(X_test/3, axis = 3, keepdims = True)
X_validation_gray = np.sum(X_validation/3, axis = 3, keepdims = True)
# проверяем размерность - последнее значение = 1.
X_train_gray.shape
 (34799, 32, 32, 1)
X_validation_gray.shape
 (4410, 32, 32, 1)
```

Посмотрим на изображение в оттенках серого и исходное изображение. Поскольку мы уменьшили размерность изображений, у массива оказалось больше измерений, чем используется, поэтому применяют np.squeeze для уменьшения ненужных размеров. Функция squeeze() удаляет оси с одним элементом (длинной 1), но не сами элементы массива.

```
i=620
plt.imshow(X_train_gray[i].squeeze(),cmap='gray')
plt.figure()
plt.imshow(X_train[i])
<matplotlib.image.AxesImage at 0x7f269a69ea90>
```



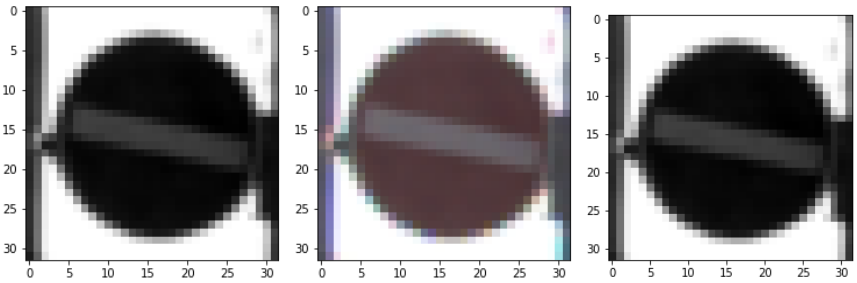


Поскольку это изображение в градациях серого, мы продолжим нормализацию его, вычитая 128, а затем разделив на 128.

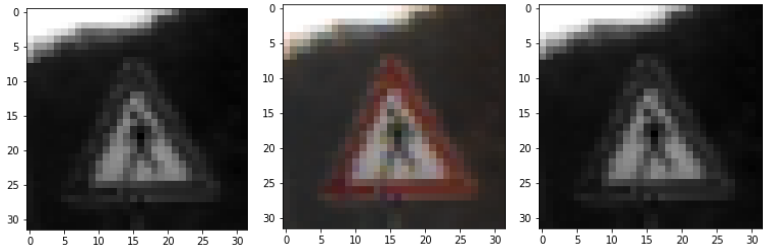
```
X_train_gray_norm = (X_train_gray - 128)/128
X_test_gray_norm = (X_test_gray - 128)/128
X_validation_gray_norm = (X_validation_gray - 128)/128
X_train_gray_norm
array([[[[ 0.25260417],
         [ 0.24739583],
         [ 0.25      ],
         ...,
        [[-0.82552083],
         [-0.828125  ],
         [-0.82291667],
         ...,
         [-0.765625  ],
         [-0.78125   ],
         [-0.80729167]]]])
```

Посмотрим разные варианты изображений

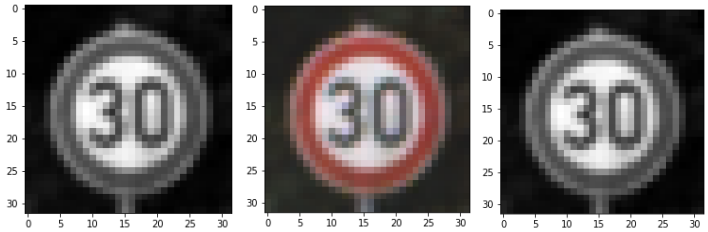
```
i = 60
plt.imshow(X_train_gray[i].squeeze(), cmap = 'gray')
plt.figure()
plt.imshow(X_train[i])
plt.figure()
plt.imshow(X_train_gray_norm[i].squeeze(), cmap = 'gray')
<matplotlib.image.AxesImage at 0x7f269a543d90>
```



```
i = 610
plt.imshow(X_test_gray[i].squeeze(), cmap = 'gray')
plt.figure()
plt.imshow(X_test[i])
plt.figure()
plt.imshow(X_test_gray_norm[i].squeeze(), cmap = 'gray')
<matplotlib.image.AxesImage at 0x7f269a423110>
```



```
i = 500
plt.imshow(X_validation_gray[i].squeeze(), cmap = 'gray')
plt.figure()
plt.imshow(X_validation[i])
plt.figure()
plt.imshow(X_validation_gray_norm[i].squeeze(), cmap = 'gray')
<matplotlib.image.AxesImage at 0x7f269a2f5210>
```



```
Шаг#4: Обучение модели¶
Модель содержит следующие слои:
Шаг 1: первый сверточный слой #1
Input = 32x32x1
Output = 28x28x6
Output = (Input-filter+1)/Stride* => (32-5+1)/1=28
Используем 5x5 фильтр, глубина 3, количество 6
Используем RELU функцию активации на выходе
pooling слой, Input = 28x28x6, Output = 14x14x6
* Stride =1.

Шаг 2: Второй сверточный слой #2
Input = 14x14x6
Output = 10x10x16
Output = (Input-filter+1)/strides => 10 = 14-5+1/1
Применяем RELU активацию на выходе
Pooling слой, Input 10x10x16, Output = 5x5x16
Шаг 3: Разворачиваем нейросеть
5x5x16 разворачивается в Output = 400
Шаг 4: 1-й полносвязный слой
1-й слой: полносвязный слой, Input = 400, Output = 120
применяем RELU функцию активации на выходе
Шаг 5: 2-й полносвязный слой
2-й слой: Input = 120, Output = 84
применяем RELU функцию активации на выходе
Шаг 6: 3-й полносвязный слой
3-й слой: Input = 84, Output = 43
```

```
In [21]:
from tensorflow.keras import datasets, layers, models
LeNet = models.Sequential()
LeNet.add(layers.Conv2D(6, (5,5), activation = 'relu', input_shape =
(32,32,1)))
LeNet.add(layers.AveragePooling2D())
LeNet.add(layers.Conv2D(16, (5,5), activation = 'relu'))
LeNet.add(layers.AveragePooling2D())
LeNet.add(layers.Flatten())
LeNet.add(layers.Dense(120, activation = 'relu'))
LeNet.add(layers.Dense(84, activation = 'relu'))
LeNet.add(layers.Dense(43, activation = 'softmax'))
LeNet.summary()
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 6)         156
_________________________________________________________________
average_pooling2d (AveragePo (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 5, 5, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 400)               0
_________________________________________________________________
dense (Dense)                (None, 120)               48120
_________________________________________________________________
dense_1 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_2 (Dense)              (None, 43)                3655
=================================================================
Total params: 64,511
Trainable params: 64,511
Non-trainable params: 0
_________________________________________________________________
LeNet.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentro-
py', metrics = ['accuracy'])
Внимание! Проверьте соответствие полученных размерностей описанным выше
размерностям!¶
history = LeNet.fit(X_train_gray_norm,
                 y_train,
                 batch_size = 500,
                 epochs = 50,
                 verbose = 1,
                 validation_data = (X_validation_gray_norm, y_valida-
tion))
Epoch 1/50
70/70 [==============================] - 16s 212ms/step - loss: 3.1909 -
accuracy: 0.1737 - val_loss: 2.7206 - val_accuracy: 0.3034
Epoch 2/50
70/70 [==============================] - 15s 210ms/step - loss: 1.7574 -
accuracy: 0.5103 - val_loss: 1.5453 - val_accuracy: 0.5508
Epoch 3/50
…
Epoch 50/50
```

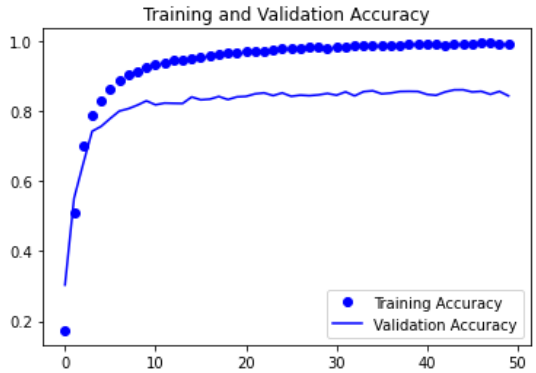
```
70/70 [==============================] - 15s 213ms/step - loss: 0.0304 -
accuracy: 0.9918 - val_loss: 1.1957 - val_accuracy: 0.8433
```

Шаг#5: Оценивание модели¶

```
# оцениваем точность нейросети
score = LeNet.evaluate(X_test_gray_norm, y_test)
print('Test Accuracy: {}'.format(score[1]))
395/395 [==============================] - 3s 8ms/step - loss: 1.6408 -
accuracy: 0.8343
Test Accuracy: 0.8342834711074829
history.history.keys()
Out[25]:
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

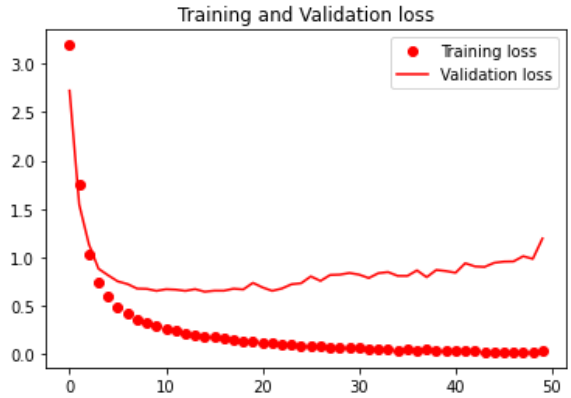
Визуализируем итерации данных обучения и проверки по эпохам. Мы наблюдаем, что точность увеличивается до 70% за несколько эпох, а затем переходит на пологий участок.

```
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training Accuracy')
plt.plot(epochs, val_accuracy, 'b', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
<matplotlib.legend.Legend at 0x7f2693337b10>
```

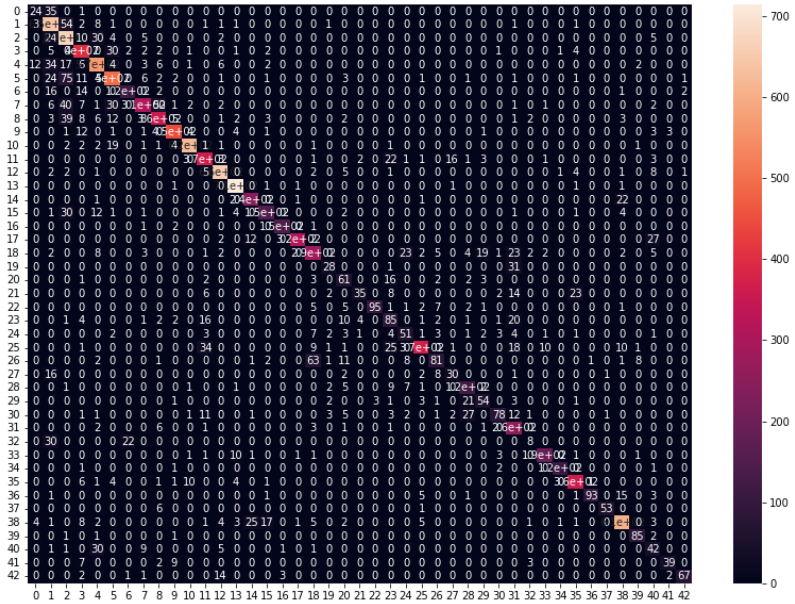


Отображаем потери при проверке и обучении в зависимости от количества эпох и наблюдаем, что оба этих графика имеют обратную зависимость. Мы видим, что процент потерь падает до 50% за 5 эпох.

```
plt.plot(epochs, loss, 'ro', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
<matplotlib.legend.Legend at 0x7f26932fbc10>
```

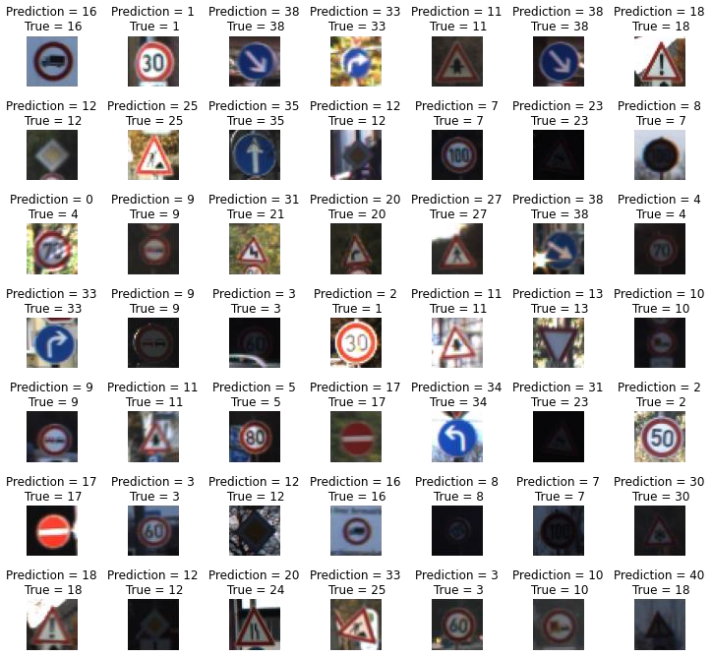


```
predicted_classes=LeNet.predict(X_test_gray_norm)
classes_x=np.argmax(predicted_classes,axis=1)
Строим матрицу ошибок
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, classes_x)
plt.figure(figsize = (14,10))
sns.heatmap(cm, annot=True)
<matplotlib.axes._subplots.AxesSubplot at 0x7f2693f42f50>
```



Назначим y_true для вывода данных тестирования и получим предсказанные значения из нашей модели с помощью функции LeNet.predict (), которая принимает X_test_gray_norm в качестве входных данных. Таким образом мы можем сравнить прогнозы наших моделей с истинным значением.

```
#y_true = y_test
#np.argmax(predicted_classes, axis=1)
L = 7
W = 7
fig, axes = plt.subplots(L, W, figsize = (12, 12))
axes = axes.ravel()
for i in np.arange(0, L*W):
    axes[i].imshow(X_test[i])
    axes[i].set_title('Prediction = {}\n True = {}'.format(classes_x[i],
y_test[i]))
    axes[i].axis('off')
plt.subplots_adjust(wspace = 1)
```



Задание для самостоятельной работы: увеличить точность работы. Обосновать, почему предпринятые меры увеличили точность. Что можно изменить:

1. Функции активации.
2. Добавить сверточный и пуллинг-слой (слои).
3. Изменить average пуллинг на max пуллинг.
4. Добавить слой/слои полносвязной нейросети.
5. Изменить функцию активации полносвязной нейросети.
6. Обоснуйте высокие значения в матрице ошибок, например, на пересечении 18-го столбца и 26-й строки.

3 ЭТАП – ВЛАДЕТЬ

Примерные вопросы для зачета

1. Задачи обработки естественного языка (NLP).
2. Возможности методов машинного обучения в обработке естественного языка.
3. Модель Bag-of-Words.
4. Векторное представление (text embeddings)
5. Модель Word2Vec
6. Рекуррентные нейронные сети.
7. Архитектура и основная идея LSTM-сетей.
8. Машинный перевод.
9. Основные задачи компьютерного зрения.
10. Сверточные нейронные сети.
11. Детектирование объектов.
12. Библиотека OpenCV.

Критерии оценивания знаний на зачете

Оценка «ЗАЧТЕНО»:

1. Хорошее знание программного материала.
2. Достаточно полное изложение теоретического вопроса экзаменационного билета.
3. Наличие незначительных неточностей в употреблении терминов, классификаций.
4. Знание основных пакетов прикладных программ

5. Неполнота представленного иллюстративного материала.
6. Точность и обоснованность выводов.
7. Логичное изложение вопроса, соответствие изложения научному стилю.
8. Негрубая ошибка при выполнении практического задания.
9. Правильные ответы на дополнительные вопросы.

Оценка «НЕ ЗАЧТЕНО»:

1. Незнание значительной части программного материала.
2. Неспособность привести примеры пакетов прикладных программ
3. Неумение выделить главное, сделать выводы и обобщения.
4. Грубые ошибки при выполнении практического задания.
5. Неправильные ответы на дополнительные вопросы.