#### Министерство науки и высшего образования Российской Федерации

#### Федеральное государственное бюджетное образовательное учреждение

#### высшего образования

#### «Владимирский государственный университет

#### имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №1

по дисциплине

"Технологии искусственного интеллекта"

Выполнил:

ст. гр. ПРИм-124

Парахин К.В.

Приняла:

доц. Кафедры ИСПИ

Озерова М.И.

Владимир, 2025 г.

**Цель работы**

Познакомиться с библиотеками Python для реализации предоставленных задач.

Вариант 14

1. **Линейная регрессия**

**Постановка задачи**

Дана обучающая выборка (𝑥( ),𝑦( ))…(𝑥 ,𝑦 ). Необходимо найти оптимальную функцию вида ℎ(𝑥)=𝑎𝑥+𝑏, которая дает наименьшую ошибку на обучающей выборке.

# ЗАДАНИЕ

Реализовать линейную регрессию выражения на языке Python. Также необходимо создать файл из ста строк, содержащий значения входных переменных и выходной переменной. Следует провести регрессию как с введением дополнительных значений, так и без.

Z = x^3 + y^2

Ссылка на Google Colab с решением: <https://colab.research.google.com/drive/1pcGunt9F8C9pTjOeRCBFvXA7ROhhI-fc?usp=sharing>

Для выполнения задания сгенерируем набор значений для функции

Далее откроем новый .csv файл и запишем в него в виде dataset полученные значения – значения для самих переменных x и y были получены с помощью поиска случайных чисел.

После этого запишем это в python dataframe (из библиотеки pandas) – и выведем его начало:

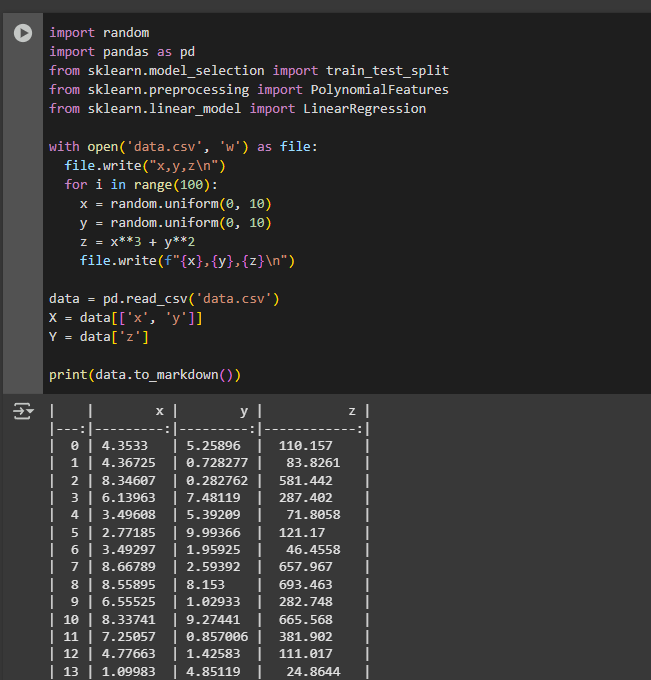


Рисунок 1.1. Вывод Dataframe

Далее будем обучение модели линейной регрессии с разделением выборки на обучающую и тестовую с использованием функции с полиномиальными признаками – то есть признаками, связанными с полиномиальными коэффициентами нелинейной функции – которые могут добиться большей точности

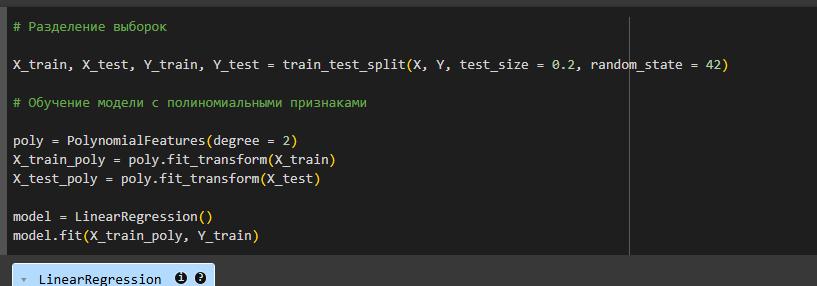


Рисунок 1.2. Обучение модели с полиномиальными признаками

Далее проведем предсказание на модели линейной регрессии, обученной на основе выборки с полиномиальными коэффициентами.

По итогу, после оценки точности модели обученной и тестовой – видно, что они показывают очень хорошие коэффициенты предсказания значений исходной функции – что говорит о большой точности регрессионной модели с полиномиальными признаками.

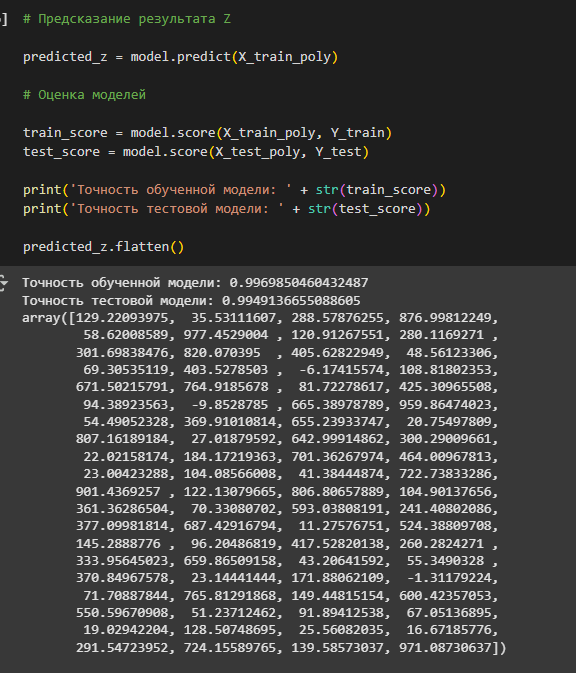


Рисунок 1.3. Оценка моделей

1. Кластеризация

Познакомиться с методами обучения без учителя на языке Python.

# ЗАДАНИЕ

Необходимо выполнить задание и убедиться, что кластеризация действительно работает. А также решить следующую задачу:

*Точки должны быть в трехмерном пространстве и располагаться внутри пяти шаров с радиусами, равными единице, но в различных центрах.*

Код из Google Colab: <https://colab.research.google.com/drive/1pcGunt9F8C9pTjOeRCBFvXA7ROhhI-fc#scrollTo=FrIFQJUDoNpp> (2 кодовый блок)

Для выполнения задания – необходимо выполнить генерацию точек в трехмерном пространстве – пусть точек всего будет 100 штук, шаров по условию должно быть 5 – определим и запишем в список набор координат их центров

import numpy as np

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

# Точки должны быть в трехмерном пространстве и располагаться внутри пяти шаров с радиусами, равными единице, но в различных центрах.

# Создание точек в 3D пространстве

np.random.seed(0)

n\_samples = 100

centres = [[-2, -2, -2], [2, 2, 2], [-2, 2, -2], [2, -2, 2], [0, 0, 0]]

radius = 1.0

Рисунок 2.1. Задание точек

Далее необходимо разработать функцию генерации точек внутри шаров с заданными центрами и радиусом равным 1.

По окончанию – мы в рамках Dataframe X – получим набор координат сгенерированных точек – чтобы далее заниматься с ними задачей кластеризации

def random\_points\_in\_sphere(n, center, radius):

    points = []

    while len(points) < n:

        point = np.random.uniform(-radius, radius, 3)

        if np.linalg.norm(point) <= radius:

            points.append(point + center)

    return np.array(points)

clusters = [random\_points\_in\_sphere(n\_samples, center, radius) for center in centres]

X = np.vstack(clusters)

Рисунок 2.2. Функция генерации точек

Далее выполним кластеризации данных о точках в Dataframe. Выполняется с использованием K-Means clustering функционала

# Кластеризация данных

kmeans = KMeans(n\_clusters = 5, random\_state = 0)

labels = kmeans.fit\_predict(X)

# Визуализация данных

figure = plt.figure(figsize = (8, 6))

ax = figure.add\_subplot(111, projection = '3d')

ax.scatter(X[:,0], X[:,1], X[:,2], c=labels)

ax.set\_xlabel('X')

ax.set\_ylabel('Y')

ax.set\_zlabel('Z')

Далее выполняем визуализацию точек из Dataframe X по их координатам в пространстве – а затем в визуализации в качестве массива, определяющего цвета (colors) – передаем массив labels – кластеризованный на основе переданного Dataframe ранее

Благодаря этому – точки на графике выделяются и подсвечиваются разными цветами – выделяясь в 5 разных кластерах – то есть визуально видно, что задача кластеризации была выполнена корректно

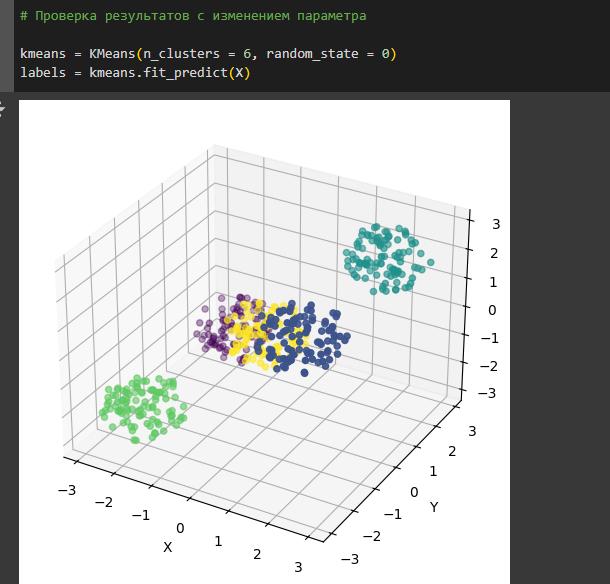


Рисунок 2.3. Визуализация кластеризованных данных

1. Классификация

Познакомиться с методами классификации (логической регрессии).

Построить классификатор для интервала 1. Проверить, насколько точно полученный классификатор распознает обучающую выборку и насколько хорошо он ведет себя на примерах не из обучающей выборки. Сделать это следует с разными параметрами регуляции. Также нужно показать, как именно выглядит график для обучающей выборки.

. (1)

Задание выполнено для вышенаписанной функции (вариант 4) в Google Colab – кодовый блок 3: <https://colab.research.google.com/drive/1pcGunt9F8C9pTjOeRCBFvXA7ROhhI-fc#scrollTo=Xv4n6XNMrjri>

Для начала задаем массив абсцисс X – генерируя случайными числами – а затем задаем функцию для вычисления классификатора для интервала значений X – а именно классификатор Y

    # Генерируем обучающую выборку

    X = np.random.uniform(-5, 5, (1000, 2))

    y = (X[:, 0]\*\*2) + (X[:, 1]) + (X[:, 0]) >= 0

    # Разделяем выборку на обучающую и тестовую

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Далее проводим обучение логистической регрессии с разными параметрами регуляризации (то есть добавления некоторых доп. ограничений к условию с целью решить некорректно поставленную задачу) – для каждого из значений регуляризации используем функцию логистической регрессии и сравниваем точность на обучающей и тестовой выборке:

    # Обучаем логистическую регрессию с разными параметрами регуляризации

    C\_values = [0.001, 0.01, 0.1, 1, 10, 100]

    for C in C\_values:

        clf = LogisticRegression(C=C)

        clf.fit(X\_train, y\_train)

        accuracy\_train = clf.score(X\_train, y\_train)

        accuracy\_test = clf.score(X\_test, y\_test)

        print(f"Точность на обучающей выборке с C={C}: {accuracy\_train}")

        print(f"Точность на тестовой выборке с C={C}: {accuracy\_test}")

По итогу получается набор результатов – из них видно что самое меньшее значение параметра регуляризации C = 0.001 дало в целом самую большую точность как на обучающей, так и на тестовой выборке – то есть значение регуляризации на данной выборке должно стремиться к минимальному

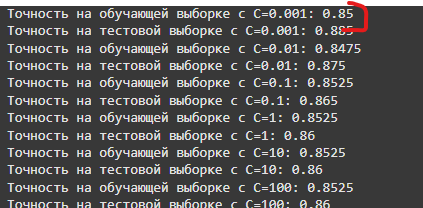


Рисунок 3.1. Сравнение точности при разных параметрах регуляризации

Далее проводим визуализацию данных о классификации обучающей выборки – при этом выделяем 2 разными цветами точки, относящиеся к разным классам – принадлежность к классу определяется по тому, как точки были распределены по заданному Y классификатору

Если провести геометрическую аналогию – то в целом это как раз точки которые при подстановке в функцию дают значение true – класс 1 (синий цвет) – или дают значение false - класс 2 (красный цвет)

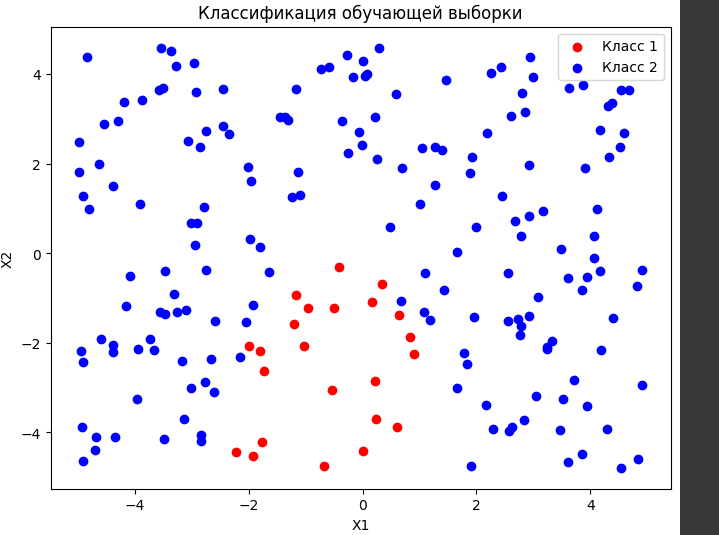


Рисунок 3.2. Визуализация классификации обучающей выборки

Задание из презентации с титаником  
  
Cсылка на Google Colab: <https://colab.research.google.com/drive/1yXYWFPb9DzADs0m1YCbwk0SDQRT_d6ks?usp=sharing>   
  
Все входные данные были загружены в отдельную папку на моем Google Drive – и к ним был оставлен доступ по ссылке

Чтобы смочь грамотно импортировать csv файл в Google Colab – я открыл свой файл, находящийся на диске через Google Sheets – а затем оттуда уже взял ссылку на инструмент по его преобразованию из .gsheet формата в .csv формат.

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

sheet\_id = "1Lho5kk-l3w6nBBFhMzSZ-0a58fLJK2c5XJl8Jq85YH8"

url = f"https://docs.google.com/spreadsheets/d/{sheet\_id}/export?format=csv"

df = pd.read\_csv(url)

#print(df.to\_markdown())

males\_total = len(df.loc[(df['Sex'] == 'male')])

females\_total = len(df.loc[(df['Sex'] == 'female')])

males\_survived = len(df.loc[((df['Sex'] == 'male') & (df['Survived'] == 1))])

females\_survived = len(df.loc[((df['Sex'] == 'female') & (df['Survived'] == 1))])

# Визуализируем

survivors\_counts = {

    'Выжили': [males\_survived, females\_survived],

    'Погибли': [males\_total - males\_survived, females\_total - females\_survived]

}

fig, ax = plt.subplots()

bottom = np.zeros(2)

for key, count in survivors\_counts.items():

    p = ax.bar(('Мужчины', 'Женщины'), count, width=0.6, label=key, bottom=bottom)

    bottom += count

    ax.bar\_label(p, label\_type='center')

ax.set\_title('Распределение выживших в зависимости от пола')

ax.set\_ylabel('Количество человек')

ax.legend()

plt.show()

from collections import Counter

# Выберем списки пассажиров в отдельные группы

survivors = df.loc[df['Survived'] == 1]

male\_survivors = df.loc[(df['Sex'] == 'male') & (df['Survived'] == 1)]

female\_survivors = df.loc[(df['Sex'] == 'female') & (df['Survived'] == 1)]

# Число возрастов

num\_of\_ages = len(dict(Counter(survivors['Age'])).keys())

# Визуализируем

fig, axs = plt.subplots(2, 2)

fig.set\_figwidth(20)

fig.set\_figheight(10)

names = [['Все пассажиры', 'Выжившие'], ['Выжившие мужчины', 'Выжившие женщины']]

for i, surv in enumerate([[df, survivors], [male\_survivors, female\_survivors]]):

    for j, subsurv in enumerate(surv):

        axs[i][j].hist(subsurv['Age'], bins=num\_of\_ages)

        axs[i][j].set\_title(names[i][j])

        axs[i][j].set\_xlabel('Возраст, лет')

        axs[i][j].set\_ylabel('Количество человек')

        axs[i][j].set\_ylim(0, 30)

        axs[i][j].set\_xlim(0, 70)

plt.show()

df.loc[((df['Survived'] == 1) & (df['Age'] > 0) & (df['Age'] > 18))].Age.hist()

В данном случае были загружены данные в Dataframe – а далее по разным измерениям входных данных (параметрам, связанным с распределением людей по возрасту, полу, классификатору выжившего человека – Survived – и остальным) были выполнены простые математические аналитические операции

Например, было посчитано число человек выживших и погибших в зависимости от пола – и это распределение отображено на гистограмме.

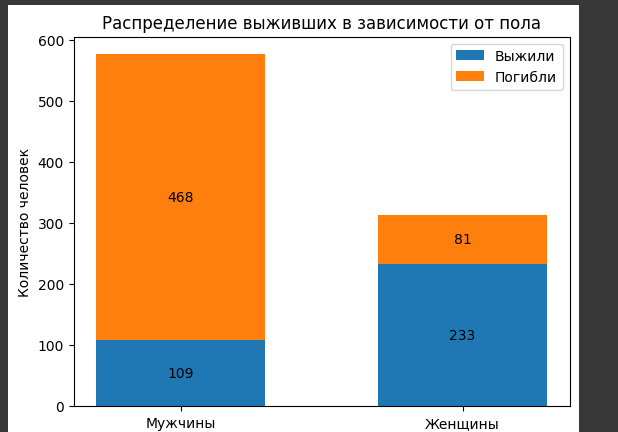
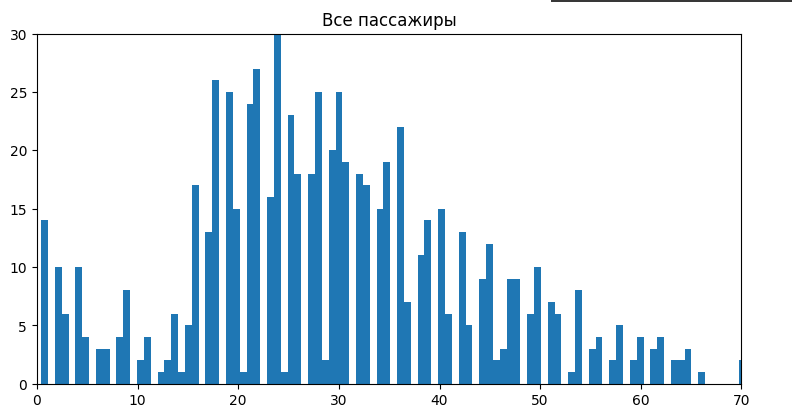


Рисунок 4.1. Распределение выживших в зависимости от пола

Также была в отдельные коллекции отфильтрована выборка выживших людей в зависимости от возраста – в этой выборке по абсциссам идет дискретный интервал возрастов – от 0 лет до 70 лет – а по ординатам на графике – кол-во пассажиров, которые относятся к каждому из значений внутри возрастного интервала



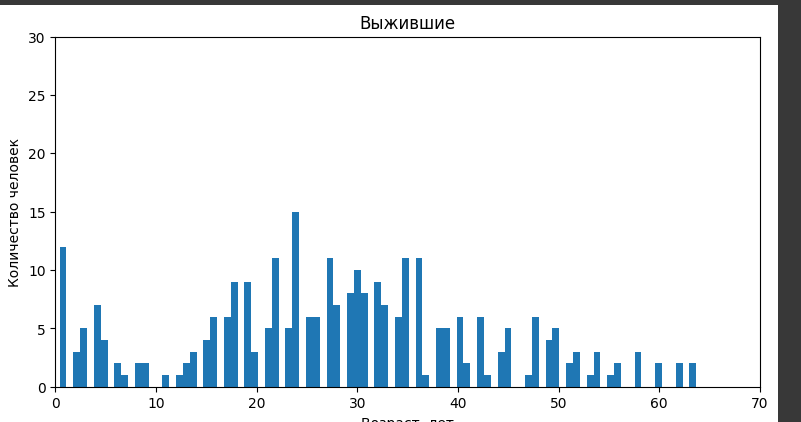


Рисунок 4.2-4.3. Столбчатые диаграммы вне зависимости от пола выживших

Эти же манипуляции были выполнены и в разрезе пола выжившего человека, на одном графике показана стобчатая диаграмма для выживших мужчин – а на другом графике аналогичная диаграмма для выживших женщин

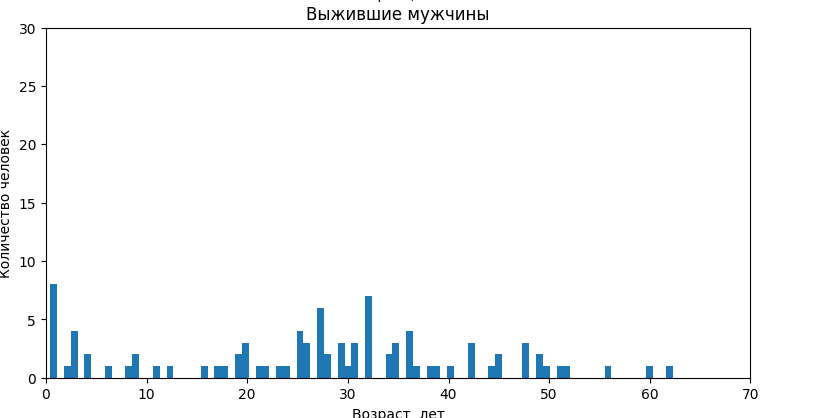




Рисунок 4.4-4.5. Столбчатые диаграммы в зависимости от пола выживших

Вывод

В результате выполнения лабораторных работ и примеров – я познакомился с основными библиотеками и функционалом Python для проведения обучения выборок на основе машинного обучения – с помощью методом линейной и полиномиальной регрессии, кластеризации и классификации.