Курсовая работа на тему "Дата-сеты"

Загоска Максим, группа М80-107-22

Задание:

1. Выбор своего датасета (условие — выполнено в виде интерактивной презентации в юпитере) Нужно попытаться визуализировать экземпляр данных
2. найти публикацию\статью про датасет, внедрить ссылку и кратко описать в презентации (+ какие популярные модели используются)
3. Пример данных с разметкой + пример кода для загрузки тестового набора
4. Пример применения готовой модели на этих данных

Описание дата-сета

Дата-сет - Heart Attack Analysis & Prediction Dataset (ссылка: https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset)

Об этом дата-сете:

- Age : Возраст пациента
- Sex : Пол пациента
- exang: стенокардия, вызванная физической нагрузкой (1 = да; 0 = нет)
- ca: количество крупных сосудов (0-3)
- cp : Тип боли в груди Тип боли в груди

```
Value 1: типичная стенокардия

Value 2: атипичная стенокардия

Value 3: неангинозная боль

Value 4: бессимптомный
```

- trtbps : артериальное давление в покое (в мм рт. ст.)
- chol : холестораль в мг/дл, полученный с помощью датчика ИМТ
- fbs : (уровень сахара в крови натощак > 120 мг/дл) (1 = верно; 0 = неверно)
- rest_ecg : результаты электрокардиографии в покое

```
Value 0: нормальный

Value 1: наличие аномалии ST-T (инверсия зубца T и/или элевация или депрессия ST > 0,05 мВ)

Value 2: указание на возможную или определенную гипертрофию левого желудочка по критериям Эстеса.
```

- thalach : максимальная частота сердечных сокращений
- target : 0 = меньше вероятность сердечного приступа 1 = больше вероятность сердечного приступа

Загрузка дата-сета

Импорт необходимых библиотек

```
import seaborn as sns
import plotly.express as px
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
```

```
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix

heart_df=pd.read_csv('/heart.csv')
heart_df.head()
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Предварительная обработка

Информация о дата-сете: типы данных атрибутов, форма, количество нулей в строке относительно столбца.

```
heart_df.info()
print('Number of rows are',heart_df.shape[0], 'and number of columns are ',heart_df.shape[1])
# Проверка нулевых значений
heart_df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trtbps    303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalachh  303 non-null    int64  
 8   exng      303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slp       303 non-null    int64  
 11  caa       303 non-null    int64  
 12  thall     303 non-null    int64  
 13  output    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
Number of rows are 303 and number of columns are  14
age         0
sex         0
cp          0
trtbps      0
chol        0
fbs         0
restecg     0
thalachh    0
exng        0
oldpeak     0
slp         0
caa         0
thall       0
output      0
dtype: int64
```

```
# Проверка дубликатов
heart_df[heart_df.duplicated()]
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

```
# Удаление дубликатов
heart_df.drop_duplicates(keep='first',inplace=True)
heart_df[heart_df.duplicated()]
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output

Сбор статистической информации

```
heart_df.describe()
```

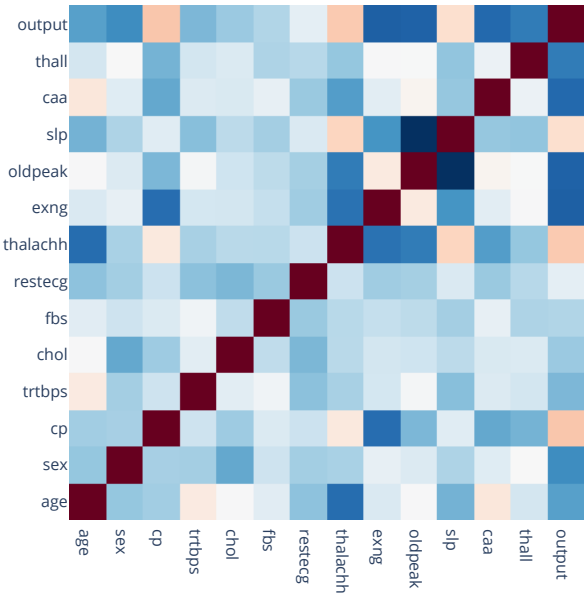
	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa
count	302.00000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000	302.000000
mean	54.42053	0.682119	0.963576	131.602649	246.500000	0.149007	0.526490	149.569536	0.327815	1.043046	1.397351	0.718543
std	9.04797	0.466426	1.032044	17.563394	51.753489	0.356686	0.526027	22.903527	0.470196	1.161452	0.616274	1.006748
min	29.00000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000
25%	48.00000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.250000	0.000000	0.000000	1.000000	0.000000
50%	55.50000	1.000000	1.000000	130.000000	240.500000	0.000000	1.000000	152.500000	0.000000	0.800000	1.000000	0.000000
75%	61.00000	1.000000	2.000000	140.000000	274.750000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	77.00000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000

```
# матрица корреляции
cor_mat = heart_df.corr()
cor_mat
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	outp
age	1.000000	-0.094962	-0.063107	0.283121	0.207216	0.119492	-0.111590	-0.395235	0.093216	0.206040	-0.164124	0.302261	0.065317	-0.2214
sex	-0.094962	1.000000	-0.051740	-0.057647	-0.195571	0.046022	-0.060351	-0.046439	0.143460	0.098322	-0.032990	0.113060	0.211452	-0.2836
cp	-0.063107	-0.051740	1.000000	0.046486	-0.072682	0.096018	0.041561	0.293367	-0.392937	-0.146692	0.116854	-0.195356	-0.160370	0.4320
trtbps	0.283121	-0.057647	0.046486	1.000000	0.125256	0.178125	-0.115367	-0.048023	0.068526	0.194600	-0.122873	0.099248	0.062870	-0.1462
chol	0.207216	-0.195571	-0.072682	0.125256	1.000000	0.011428	-0.147602	-0.005308	0.064099	0.050086	0.000417	0.086878	0.096810	-0.0814
fbs	0.119492	0.046022	0.096018	0.178125	0.011428	1.000000	-0.083081	-0.007169	0.024729	0.004514	-0.058654	0.144935	-0.032752	-0.0268
restecg	-0.111590	-0.060351	0.041561	-0.115367	-0.147602	-0.083081	1.000000	0.041210	-0.068807	-0.056251	0.090402	-0.083112	-0.010473	0.1348
thalachh	-0.395235	-0.046439	0.293367	-0.048023	-0.005308	-0.007169	0.041210	1.000000	-0.377411	-0.342201	0.384754	-0.228311	-0.094910	0.4199
exng	0.093216	0.143460	-0.392937	0.068526	0.064099	0.024729	-0.068807	-0.377411	1.000000	0.286766	-0.256106	0.125377	0.205826	-0.4356
oldpeak	0.206040	0.098322	-0.146692	0.194600	0.050086	0.004514	-0.056251	-0.342201	0.286766	1.000000	-0.576314	0.236560	0.209090	-0.4291
slp	-0.164124	-0.032990	0.116854	-0.122873	0.000417	-0.058654	0.090402	0.384754	-0.256106	-0.576314	1.000000	-0.092236	-0.103314	0.3439
caa	0.302261	0.113060	-0.195356	0.099248	0.086878	0.144935	-0.083112	-0.228311	0.125377	0.236560	-0.092236	1.000000	0.160085	-0.4089
thall	0.065317	0.211452	-0.160370	0.062870	0.096810	-0.032752	-0.010473	-0.094910	0.205826	0.209090	-0.103314	0.160085	1.000000	-0.3431
output	-0.221476	-0.283609	0.432080	-0.146269	-0.081437	-0.026826	0.134874	0.419955	-0.435601	-0.429146	0.343940	-0.408992	-0.343101	1.0000

Тепловая карта, представляющая корреляционную матрицу

```
fig = px.imshow(cor_mat, color_continuous_scale='RdBu_r', origin='lower')
fig.show()
```



Визуализация дата-сета

Возраст

```
heart_df['age'].unique()
```

```
array([63, 37, 41, 56, 57, 44, 52, 54, 48, 49, 64, 58, 50, 66, 43, 69, 59,
       42, 61, 40, 71, 51, 65, 53, 46, 45, 39, 47, 62, 34, 35, 29, 55, 60,
       67, 68, 74, 76, 70, 38, 77])
```

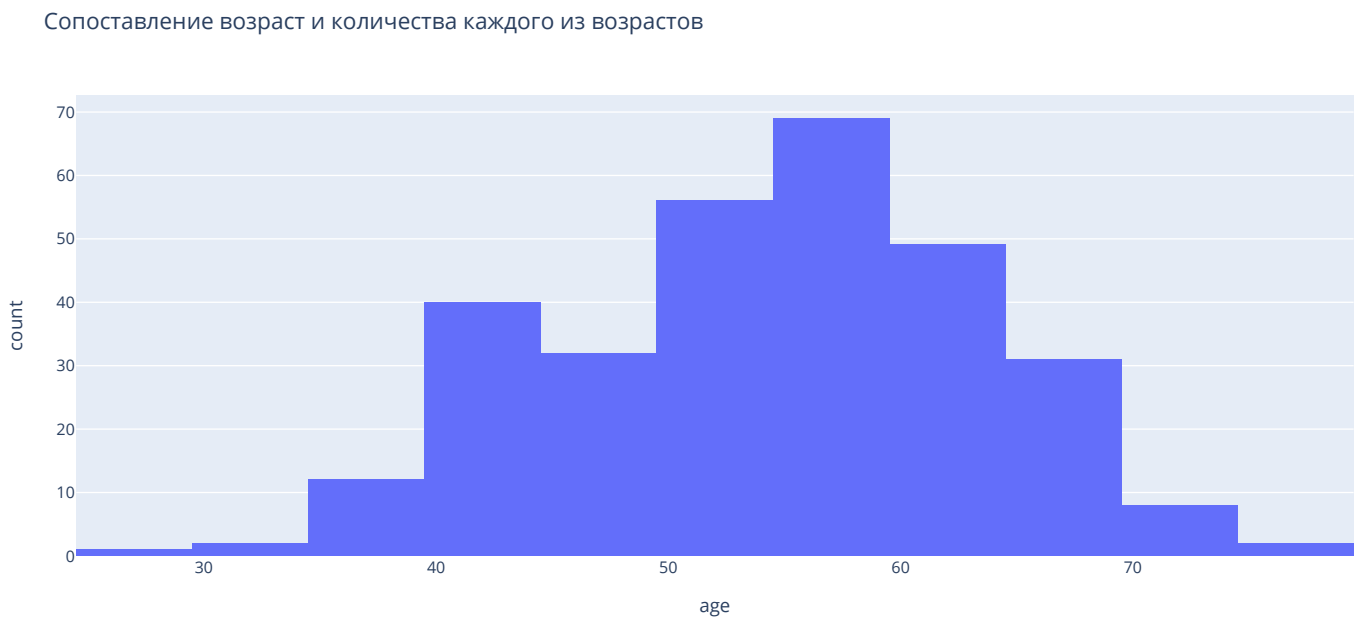
```
heart_df['age'].nunique()
```

```
41
```

```
age_lst = heart_df['age'].value_counts().index
nop_age = heart_df['age'].value_counts().values
print("Уникальный возраст:  ")
print(age_lst)
print("Кол-во уникальных возрастов:  ")
print(nop_age)
```

```
Уникальный возраст:
Int64Index([58, 57, 54, 59, 52, 51, 62, 56, 44, 60, 41, 64, 67, 63, 43, 55, 42,
            61, 65, 53, 45, 50, 48, 46, 66, 47, 49, 70, 39, 68, 35, 71, 40, 69,
            34, 37, 38, 29, 74, 76, 77],
           dtype='int64')
Кол-во уникальных возрастов:
[19 17 16 14 13 12 11 11 11 11 10 10  9  9  8  8  8  8  8  8  7  7  7
  7  5  5  4  4  4  4  3  3  3  2  2  2  1  1  1  1]
```

```
fig = px.histogram(heart_df,
             x='age',
             nbins=10,
             title='Сопоставление возраст и количества каждого из возрастов'
            )
fig.show()
```



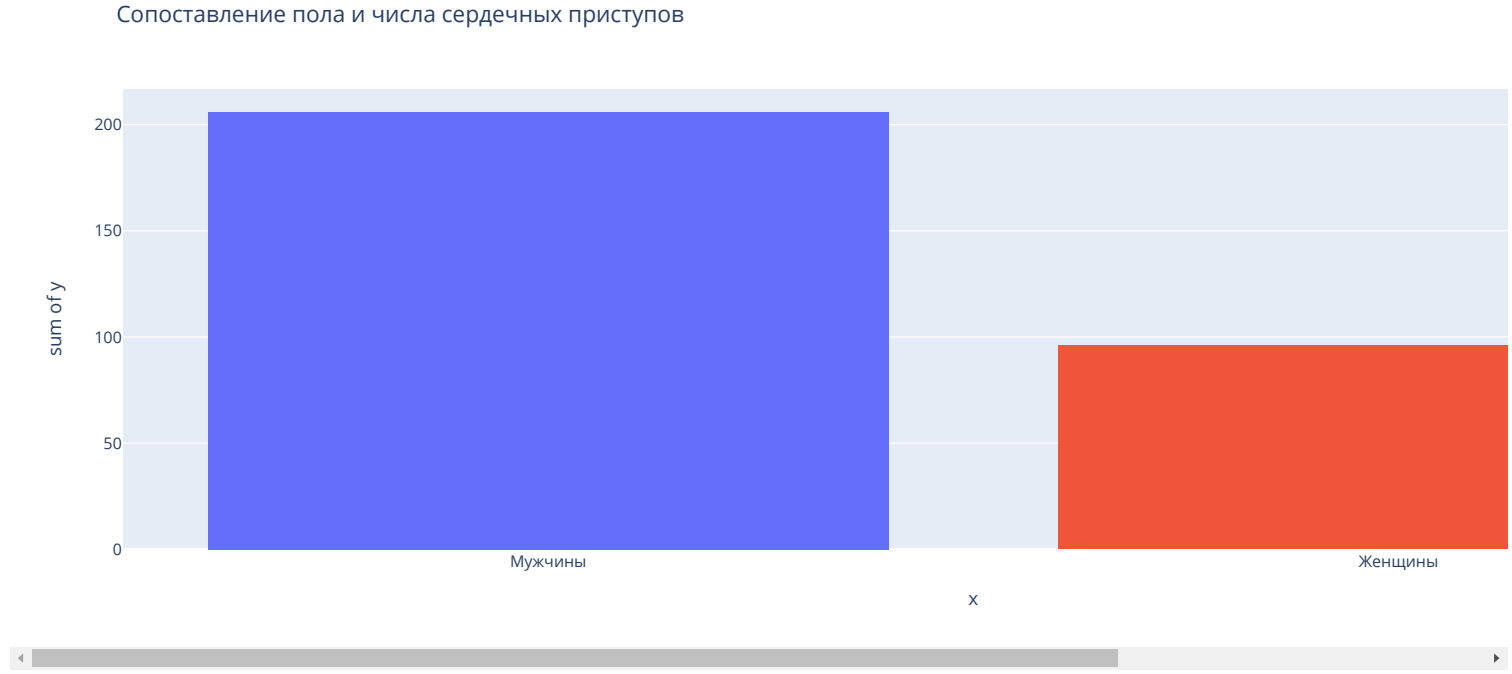

```
fig = px.histogram(heart_df[heart_df['output'] == 1],
             x='age',
             nbins=10,
             title='Сопоставление возраста и числа сердечных приступов'
            )
fig.show()
```

Вывод: Наибольшее количество сердечных приступов наблюдается в возрастной группе 50-54 года.

Пол

```
sex_name = ['Мужчины', 'Женщины']
sex_count = heart_df['sex'].value_counts().values

fig = px.histogram(x=sex_name,
             y=sex_count,
             color = sex_name,
             title='Сопоставление пола и числа сердечных приступов'
            )
fig.show()
```

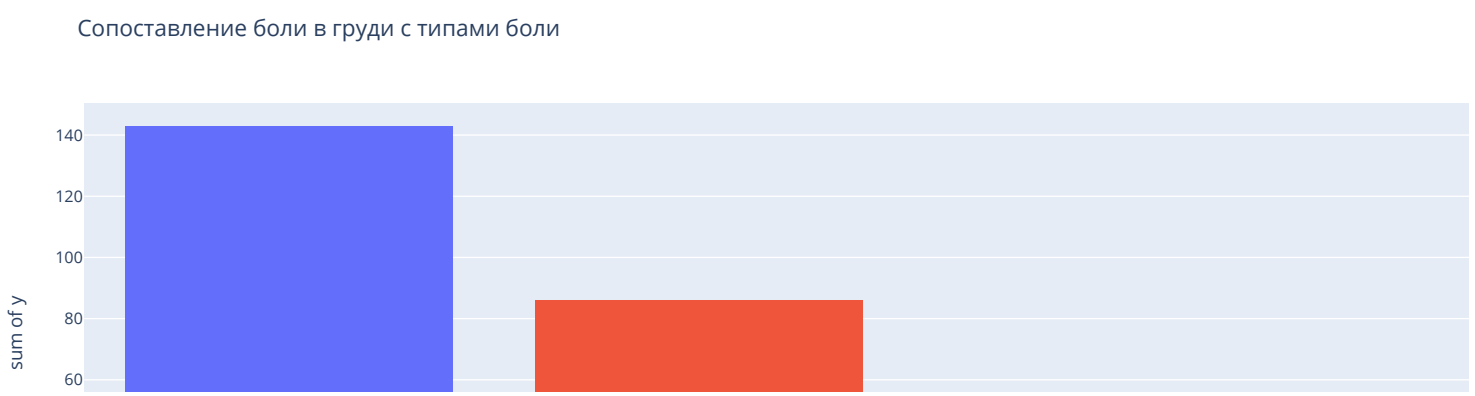


Вывод: явной связи с полом не наблюдается

Боль в груди

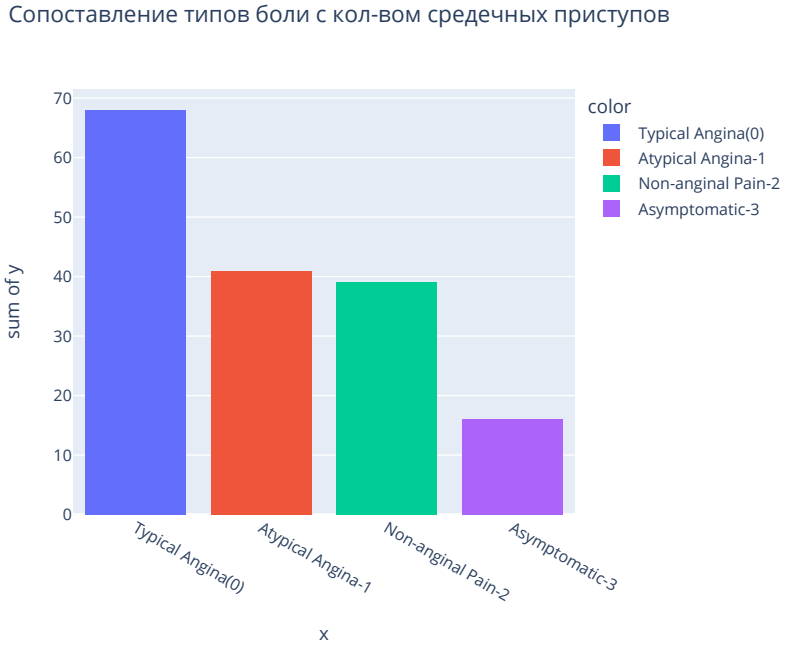
```
cp_type = ['Typical Angina(0)', 'Atypical Angina-1', 'Non-anginal Pain-2', 'Asymptomatic-3']
cp_type_count = heart_df['cp'].value_counts().values


fig = px.histogram(x=cp_type,
             y=cp_type_count,
            color=cp_type,
             title='Сопоставление боли в груди с типами боли'
            )
fig.show()
```




```
cp_type = ['Typical Angina(0)', 'Atypical Angina-1', 'Non-anginal Pain-2', 'Asymptomatic-3']
cp_type_count = heart_df[heart_df['output'] == 1]['cp'].value_counts().values

fig = px.histogram(x=cp_type,
             y=cp_type_count,
            color=cp_type,
             title='Сопоставление типов боли с кол-вом средечных приступов'
            )
fig.show()
```



Вывод: Типичная ангина имеет наибольшее количество сердечных приступов, т.е. 68.

Обучение модели

Обученную модель можно использовать для предсказания вероятности сердечного признака обследуемого на данных, полученных в результате врачебного обследования.

Разделение на тестовую и обучающую выборки Дата-сет был разделен на обучающую и тестовую выборки. Размер обучающей выборки - 80%, тестовой - 20%.

```
X = heart_df.iloc[:, :-1].values
y = heart_df.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state= 0)
print('Shape for training data', X_train.shape, y_train.shape)
print('Shape for testing data', X_test.shape, y_test.shape)
```

```
Shape for training data (241, 13) (241,)
Shape for testing data (61, 13) (61,)
```

Масштабирование

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Обучение модели

Путем подбора наилучшего алгоритма обучения для получения максимальной точности, лучшую точность показал алгоритм - **Support vector machine** (список используемых алгоритмов: Logistic Regression, Gaussian Naive Bayes, Bernoulli Naive Bayes, Support Vector Machine, K Nearest Neighbours, X Gradient Boosting).

```
svm_model = SVC()
svm_model.fit(X_train, y_train)

predicted = svm_model.predict(X_test)
print("The accuracy of SVM is : ", accuracy_score(y_test, predicted)*100, "%")
```

```
The accuracy of SVM is :  93.44262295081968 %
```