



**Министерство науки и высшего образования
Российской Федерации Федеральное государственное
бюджетное образовательное учреждение высшего
образования «Московский государственный
технический университет имени Н.Э. Баумана**

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчёт по РК2

«Технологии машинного обучения»

Вариант 5

Выполнил:

студент группы ИУ5-63Б

Журмилов В.Д.

Преподаватель:

Гапанюк Ю. Е.

2024 г.

Задание: Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Метод №1: Дерево решений

Метод №2: Случайный лес

Датасет: <https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset>

Решение:

Импортируем необходимые модули и загружаем данные из файла 'heart.csv' в объект `df`. Выводим первые строки данных с помощью `head()` и получаем информацию о данных с помощью `info()`. Подсчитываем количество пропущенных значений с помощью `isnull().sum()`.

```
In [1]: import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import sklearn

from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error, r2_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.impute import SimpleImputer, MissingIndicator
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, accuracy_score
from sklearn.svm import SVC, NuSVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV
from sklearn import tree
from IPython.core.display import HTML
from sklearn.tree import export_text
from operator import itemgetter
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_predict
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

```
In [2]: df = pd.read_csv('heart.csv')
```

```
In [3]: df.head()
```

```
Out[3]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
In [4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   age         1025 non-null   int64
1   sex         1025 non-null   int64
2   cp          1025 non-null   int64
3   trestbps    1025 non-null   int64
4   chol        1025 non-null   int64
5   fbs         1025 non-null   int64
6   restecg     1025 non-null   int64
7   thalach     1025 non-null   int64
8   exang       1025 non-null   int64
9   oldpeak     1025 non-null   float64
10  slope       1025 non-null   int64
11  ca          1025 non-null   int64
12  thal        1025 non-null   int64
13  target      1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
In [5]: df.isnull().sum()

Out[5]: age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Создаём матрицу корреляции и визуализируем её с помощью тепловой карты, чтобы оценить связи между признаками в данных.

```
In [6]: corr = df.corr()
fig, ax = plt.subplots(figsize=(12, 8))
ax = sns.heatmap(corr, annot=True, linewidth=.5, fmt=".2f")
plt.show()
```



Наиболее сильная корреляция с целевым признаком "target" наблюдается у признаков "cp", "thalach", "exang", "oldpeak", "ca". При построении модели

машинного обучения эти признаки будут наиболее информативными и важными для прогнозирования результата – есть ли у пациента заболевание.

Далее выполняем удаление столбца 'target', кодируем столбец 'target' в числовые значения и разбиваем данные на обучающую и тестовую выборки в соотношении 80/20.

```
In [7]: X = df.drop(['target'], axis=1) #Наименования признаков
        y = df['target'] # Значения признаков

In [8]: # кодируем категориальные данные из строк в числа
        le = LabelEncoder()
        y = le.fit_transform(y)

In [9]: X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.20, shuffle=False)

In [10]: # Размер обучающей выборки
         X_train.shape, y_train.shape
Out[10]: ((820, 13), (820,))

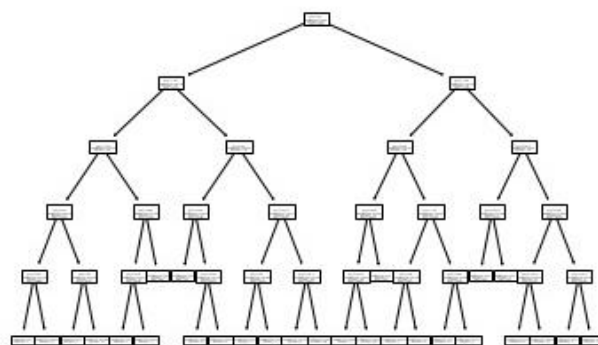
In [11]: # Размер тестовой выборки
         X_test.shape, y_test.shape
Out[11]: ((205, 13), (205,))
```

Затем создаем регрессор `DecisionTreeRegressor` с максимальной глубиной 5 и обучаем его на обучающих данных. Затем мы строим **дерево решений** для визуализации полученной модели. Далее, мы создаем классификатор `DecisionTreeClassifier` с указанным случайным состоянием и обучаем его на обучающих данных. Также определена функция `test_model()` для оценки модели с использованием различных метрик. В конце вызываем функцию `test_model()` для оценки модели `dt_none` на тестовых данных.

```
In [12]: dt_none = DecisionTreeRegressor(max_depth=5)
         dt_none.fit(X_train, y_train)

Out[12]: DecisionTreeRegressor
         DecisionTreeRegressor(max_depth=5)

In [13]: tree.plot_tree(dt_none);
```



```
In [14]: clf = DecisionTreeClassifier(random_state=1)
clf.fit(X_train, y_train)
```

```
Out[14]: DecisionTreeClassifier
DecisionTreeClassifier(random_state=1)
```

```
In [15]: def test_model(model):
print("mean absolute error:",
      mean_absolute_error(y_test, model.predict(X_test)))
print("median absolute error:",
      median_absolute_error(y_test, model.predict(X_test)))
print("r2 score:",
      r2_score(y_test, model.predict(X_test)))
```

```
In [16]: test_model(dt_none)

mean_absolute_error: 0.14916588524552168
median_absolute_error: 0.019512195121951237
r2_score: 0.6253338632194336
```

Определим функцию `draw_feature_importances`, которая выводит график важности признаков на основе модели дерева решений (`clf`) и набора данных признаков (`X_train`). График отображает столбцы с названиями признаков и их важностью, а также выводит значения важности над соответствующими столбцами.

```
In [17]: def draw_feature_importances(tree_model, X_dataset, figsize=(18, 5)):
"""
Вывод важности признаков в виде графика
"""
# Сортировка значений важности признаков по убыванию
list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_importances_))
sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse=True)

# Названия признаков
labels = [x for x, _ in sorted_list]
# Важности признаков
data = [x for _, x in sorted_list]

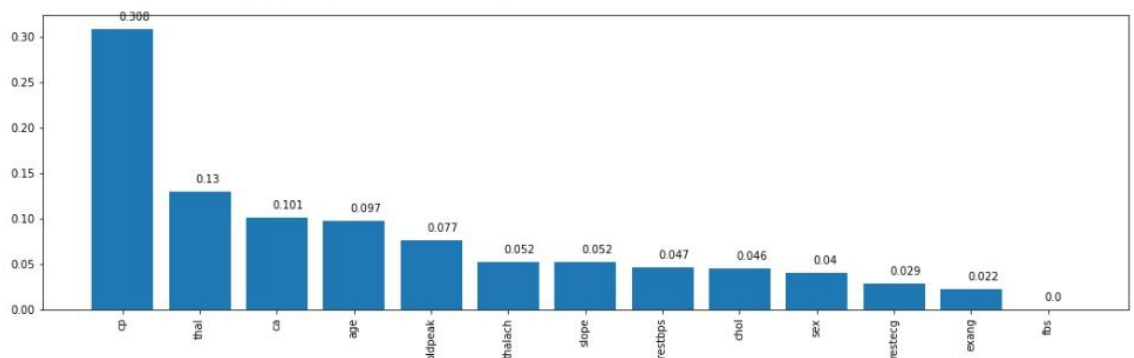
# Вывод графика
fig, ax = plt.subplots(figsize=figsize)
ind = np.arange(len(labels))
plt.bar(ind, data)
plt.xticks(ind, labels, rotation='vertical')

# Вывод значений
for a, b in zip(ind, data):
    plt.text(a - 0.05, b + 0.01, str(round(b, 3)))

plt.show()

return labels, data
```

```
In [18]: dt_fl, dt_fd = draw_feature_importances(clf, X_train)
```



Далее выполняем поиск наилучших параметров для модели `DecisionTreeClassifier` с помощью кросс-валидации и оцениваем ее точность на тестовых данных, а также сравниваем ее с другими моделями.

```
In [19]: tree = DecisionTreeClassifier()

param_grid = {'max_depth': [2, 4, 6, 8, 10],
              'min_samples_split': [2, 4, 6, 8, 10],
              'min_samples_leaf': [1, 2, 3, 4, 5]}
grid_search = GridSearchCV(tree, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
accuracy_tree = grid_search.best_estimator_.score(X_test, y_test)

print("Наилучшие параметры: {} ".format(grid_search.best_params_))
print("Оценка точности на кросс-валидации: {:.2f}".format(grid_search.best_score_))
print(accuracy_tree)

Наилучшие параметры: {} {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2}
Оценка точности на кросс-валидации: 0.98
0.9853658536585366
```

```
In [20]: models = [['DecisionTree', DecisionTreeRegressor()]]
```

```
In [21]: print('Вывод 1')
for name, model in models:
    model = model
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(name, (np.sqrt(mean_squared_error(y_test, predictions))))
```

```
Вывод 1
DecisionTree : 0.12097167578182678
```

```
In [22]: models = [['DecisionTree:', DecisionTreeRegressor(max_depth=6)],
                  ['Линейная регрессия:', LinearRegression()],
                  ['SVC:', SVC(C=1, kernel='linear')]]
```

```
In [23]: print('Вывод 2')
for name, model in models:
    model = model
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(name, (np.sqrt(mean_squared_error(y_test, predictions))))
```

```
Вывод 2
DecisionTree: 0.2141112756653403
Линейная регрессия: 0.3862846119045108
SVC: 0.45799137280190727
```

После создаем модель **случайного леса** (`RandomForestClassifier`), обучаем ее на тренировочных данных, оцениваем ее точность на тестовых данных, а затем выполняем поиск наилучших параметров для модели с помощью кросс-валидации и оцениваем ее точность на тестовых данных с использованием найденных параметров.

```
In [24]: # Создаем модель случайного леса с 100 деревьями
rf_model = RandomForestClassifier(n_estimators=100)
# Обучаем модель на тренировочных данных
rf_model.fit(X_train, y_train)
# Оцениваем качество модели на тестовых данных
accuracy = rf_model.score(X_test, y_test)
print('Accuracy: {:.2f}%'.format(accuracy*100))
```

Accuracy: 98.54%

```
In [25]: model = RandomForestClassifier()
param_grid = {
    'n_estimators': [200, 700],
    'max_features': ['auto', 'sqrt', 'log2']
}

grid_search = GridSearchCV(model, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
accuracy_RandomForestClassifier = grid_search.best_estimator_.score(X_test, y_test)

print("Наилучшие параметры: {} ".format(grid_search.best_params_))
print("Оценка точности на кросс-валидации: {:.2f}".format(grid_search.best_score_))
print(accuracy_tree)
```

Наилучшие параметры: {} {'max_features': 'auto', 'n_estimators': 200}
Оценка точности на кросс-валидации: 0.99
0.9853658536585366