Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

Лабораторная работа №5 по дисциплине «Технологии машинного обучения»

«Ансамбли моделей машинного обучения»

Студент: Коростелев Андрей Михайлович

Группа: ИУ5–64Б

Москва, 2021

Описание задания:

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии;
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков;
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую;
4. Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Текст программы и экранные формы с примерами выполнения программы:

ЛР №5

Ансамбли моделей машинного обучения.

1) Импорт библиотек. Загрузка, первичный анализ и масштабирование данных.

In [17]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score, cross_validate
from typing import Dict, Tuple
from scipy import stats
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import KFold, RepeatedKFold, LeaveOneOut, LeavePOut, ShuffleSplit, StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
```

В качестве набора данных мы будем использовать набор данных, содержащий информацию о прогнозе смертности от сердечной недостаточности

Датасет состоит из одного файла: 'heart.csv'

In [2]:
```
data = pd.read_csv('data/heart.csv', sep=",")
```

In [3]:
```
# Размер датасета (строки, столбцы)
data.shape
```

Out[3]: (299, 13)

In [4]:
```
# Список колонок с типами данных
data.dtypes
```

Out[4]:
```
age                         float64
anaemia                       int64
creatinine_phosphokinase      int64
diabetes                      int64
ejection_fraction             int64
high_blood_pressure           int64
platelets                   float64
serum_creatinine            float64
serum_sodium                  int64
sex                           int64
smoking                       int64
time                          int64
DEATH_EVENT                   int64
dtype: object
```

In [5]:
```
# Количество пропущенных значений
data.isnull().sum()
```

Out[5]:
```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

In [6]:
```
# Первые 5 строк датасета
data.head()
```

Out[6]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	tim
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	

Как видно, приведённый набор данных не имеет пропусков и все колонки имеют численный тип данных. Однако для успешного решения задачи классификации мы должны произвести масштабированние данных в колонках "age", "platelets", "ejection_fraction", "time", "serum_sodium", "serum_creatinine" и "creatinine_phosphokinase".

In [7]:
```
data['time'] = MinMaxScaler().fit_transform(data[['time']])
data['platelets'] = MinMaxScaler().fit_transform(data[['platelets']])
data['serum_sodium'] = MinMaxScaler().fit_transform(data[['serum_sodium']])
data['creatinine_phosphokinase'] = MinMaxScaler().fit_transform(data[['creatinine_phosphokinase']])
data['age'] = MinMaxScaler().fit_transform(data[['age']])
data['ejection_fraction'] = MinMaxScaler().fit_transform(data[['ejection_fraction']])
data['serum_creatinine'] = MinMaxScaler().fit_transform(data[['serum_creatinine']])
```

In [8]:
```
data.head()
```

Out[8]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking
0	0.636364	0	0.071319	0	0.090909	1	0.290823	0.157303	0.485714	1	0
1	0.272727	0	1.000000	0	0.363636	0	0.288833	0.067416	0.657143	1	0
2	0.454545	0	0.015693	0	0.090909	0	0.165960	0.089888	0.457143	1	1
3	0.181818	1	0.011227	0	0.090909	0	0.224148	0.157303	0.685714	1	0
4	0.454545	1	0.017479	1	0.090909	0	0.365984	0.247191	0.085714	0	0

Датасет отмасштабирован и готов к решению задачи классификации

2) Разделение выборки на обучающую и тестовую.

In [10]:
```
data_X = data.drop(columns='DEATH_EVENT')
data_Y = data['DEATH_EVENT']
X_train, X_test, Y_train, Y_test = train_test_split(data_X,  data_Y, test_size=0.25, random_state = 0)
```

In [11]:
```
Y_train=np.ravel(Y_train)
Y_test=np.ravel(Y_test)
```

In [12]:
```
# Проверим правильность разделения выборки на тестовую и обучающую. Посмотрим на размеры матриц.
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(224, 12)
(75, 12)
(224,)
(75,)
```

3) Обучение и тестирование моделей.

In [13]:
```
def accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray) -> Dict[int, float]:
    """
    Вычисление метрики accuracy для каждого класса
    y_true - истинные значения классов
    y_pred - предсказанные значения классов
    Возвращает словарь: ключ - метка класса,
    значение - Accuracy для данного класса
    """
    # Для удобства фильтрации сформируем Pandas DataFrame
    d = {'t': y_true, 'p': y_pred}
    df = pd.DataFrame(data=d)
    # Метки классов
    classes = np.unique(y_true)
    # Результирующий словарь
    res = dict()
    # Перебор меток классов
    for c in classes:
        # отфильтруем данные, которые соответствуют
        # текущей метке класса в истинных значениях
        temp_data_flt = df[df['t']==c]
        # расчет accuracy для заданной метки класса
        temp_acc = accuracy_score(
            temp_data_flt['t'].values,
            temp_data_flt['p'].values)
        # сохранение результата в словарь
        res[c] = temp_acc
    return res

def print_accuracy_score_for_classes(
    y_true: np.ndarray,
    y_pred: np.ndarray):
    """
    Вывод метрики accuracy для каждого класса
    """
    accs = accuracy_score_for_classes(y_true, y_pred)
    if len(accs)>0:
        print('Метка \t Accuracy')
    for i in accs:
        print('{} \t {}'.format(i, accs[i]))
```

In [15]:
```
from operator import itemgetter

def draw_feature_importances(tree_model, X_dataset, figsize=(10,5)):
    """
    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
```

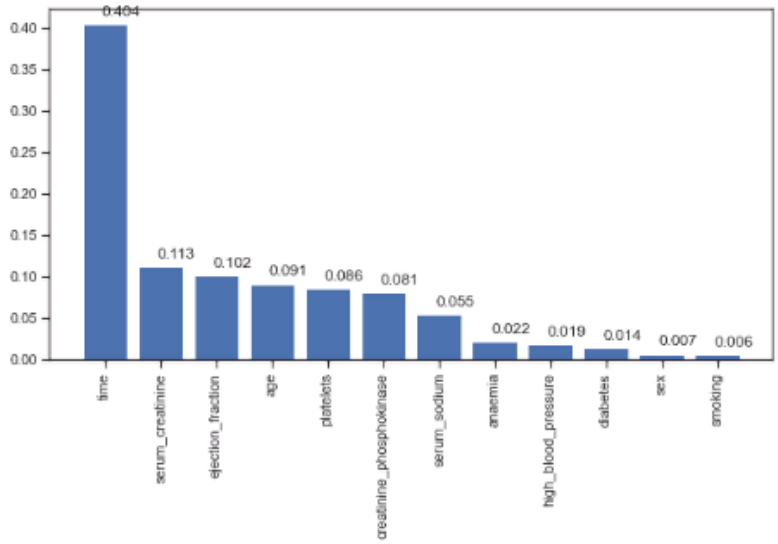
Модель "Случайный лес"

In [19]:
```
# Обучим классификатор на 5 деревьях
tree = RandomForestClassifier(n_estimators=5, oob_score=True, random_state=0)
tree.fit(X_train, Y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:540: UserWarning: Some inputs do not have OOB scores. Th
is probably means too few trees were used to compute any reliable oob estimates.
  warn("Some inputs do not have OOB scores. "
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:544: RuntimeWarning: invalid value encountered in true_d
ivide
  decision = (predictions[k] /
```

Out[19]:
```
RandomForestClassifier(n_estimators=5, oob_score=True, random_state=0)
```

In [22]:
```
_,_ = draw_feature_importances(tree, data)
```



In [23]:
```
target1 = tree.predict(X_test)
```

In [24]:
```
print('Процент точности:',accuracy_score(Y_test, target1))
```

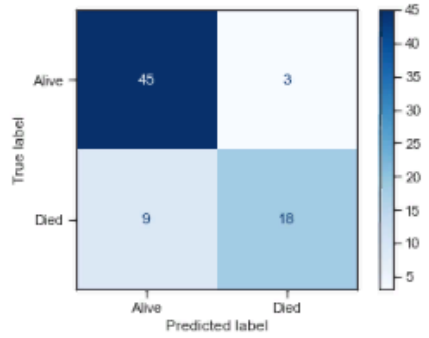
```
Процент точности: 0.84
```

In [25]:
```
print('Процент точности для каждого класса:', print_accuracy_score_for_classes(Y_test, target1))
```

```
Метка    Accuracy
0        0.9375
1        0.6666666666666666
Процент точности для каждого класса: None
```

In [27]:
```
plot_confusion_matrix(tree, X_test, Y_test,
                      display_labels=['Alive','Died'], cmap=plt.cm.Blues)
```

Out[27]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f130ee1d90>
```

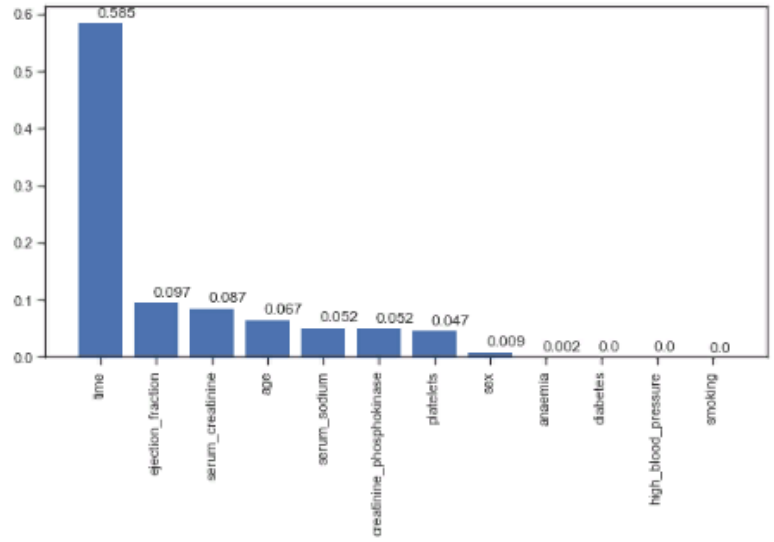


Модель "Градиентный бустинг"

In [28]:
```
gr_boost = GradientBoostingClassifier(random_state=1)
gr_boost.fit(X_train, Y_train)
```

Out[28]:
```
GradientBoostingClassifier(random_state=1)
```

In [29]:
```
_,_ = draw_feature_importances(gr_boost, data)
```



In [30]:
```
target2 = gr_boost.predict(X_test)
```

In [31]:
```
print('Процент точности:',accuracy_score(Y_test, target2))
```

```
Процент точности: 0.8266666666666667
```

In [32]:
```
print('Процент точности для каждого класса:', print_accuracy_score_for_classes(Y_test, target1))
```

```
Метка    Accuracy
0        0.9375
1        0.6666666666666666
Процент точности для каждого класса: None
```

In [33]:
```
plot_confusion_matrix(gr_boost, X_test, Y_test,
                      display_labels=['Alive','Died'], cmap=plt.cm.Blues)
```

Out[33]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1f130e9cf10>
```