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 Системы обработки информации и управления

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Решение задачи машинного обучения

Студент группы ИУ5-64Б _____ _____Коростелев А.М._____
(Подпись, дата) (Фамилия И.О.)

Руководитель курсовой работы _____ _____Гапанюк Ю.Е._____
(Подпись, дата) (Фамилия И.О.)

Москва, 2021 г.

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)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ5
(Индекс)
Черненький В.М.
(Фамилия И.О.)
« __ » _____ 2021 г.

З А Д А Н И Е
на выполнение курсовой работы

По дисциплине Технологии машинного обучения

Решение задачи машинного обучения
(Тема курсовой работы)

Студент Коростелев А.М. Группа ИУ5–64Б
(Фамилия, инициалы, индекс группы)

График выполнения работы: 25% к 10 нед., 50% к 12 нед., 75% к 15 нед., 100% к 17 нед.

Задание

- Типовое исследование – решение задачи машинного обучения на основе материалов дисциплины. Исследование набора данных, обучение базового решения, подбор гиперпараметров для выбранных моделей. Формирование выводов о качестве построенных моделей.

Оформление курсовой работы

- Расчетно-пояснительная записка на 15 страницах формата А4.

Дата выдачи задания « 1 » Апреля 2021 г.

Руководитель курсовой работы _____ _____Гапанюк Ю.Е._____
(Подпись, дата) (Фамилия И.О.)

Студент _____ _____Коростелев А.М._____
(Подпись, дата) (Фамилия И.О.)

Оглавление

1. Импорт библиотек. Поиск и загрузка набора данных для построения моделей машинного обучения. Проведение разведочного анализа данных. Анализ и заполнение пропусков в данных

```
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 KNeighborsClassifier
from sklearn.metrics import 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 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.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
from sklearn.metrics import roc_curve, roc_auc_score
```

В качестве набора данных мы будем использовать набор данных, содержащий информацию о наличии у пациента сердечного заболевания

Датасет состоит из одного файла: 'HeartKR.csv'

В данном датасете имеются следующие признаки:

- Возраст (age);
- Пол (sex);
- Тип боли в груди (cp) - 4 значения;
- Артериальное давление в состоянии покоя (trestbps);
- Уровень холестерина (chol);
- Уровень сахара в крови натощак (fbs) > 120 мг/дл - 2 значения;
- Результаты ЭКГ в покое (restecg) - 3 значения;
- Максимальная частота сердечных сокращений (thalach);
- Стенокардия, вызванная физической нагрузкой (exang) - 2 значения;
- Депрессия ST, вызванная упражнениями по сравнению с отдыхом (oldpeak);
- Наклон сегмента ST при пиковой нагрузке (slope) - 3 значения;
- Количество крупных сосудов с флуорозопией (ca) - 5 значений;
- Талассемия (thal) - 4 значения;
- Наличие сердечного заболевания (target) - 2 значения (0 - нет, 1 - есть);

Целевым признаком выберем 'target'

```
# Загрузка датасета
data = pd.read_csv('data/HeartKR.csv', sep=",")
```

```
# Размер датасета
data.shape
```

```
(303, 14)
```

```
# Список колонок с типами данных
data.dtypes
```

```
age          int64
sex          int64
cp           int64
trestbps     int64
chol         int64
fbs          int64
restecg      int64
thalach      int64
exang        int64
oldpeak    float64
slope        int64
ca           int64
thal         int64
target       int64
dtype: object
```

```
# Количество пропущенных значений
data.isnull().sum()
```

```
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
```

```
# Первые 5 строк датасета
data.head()
```

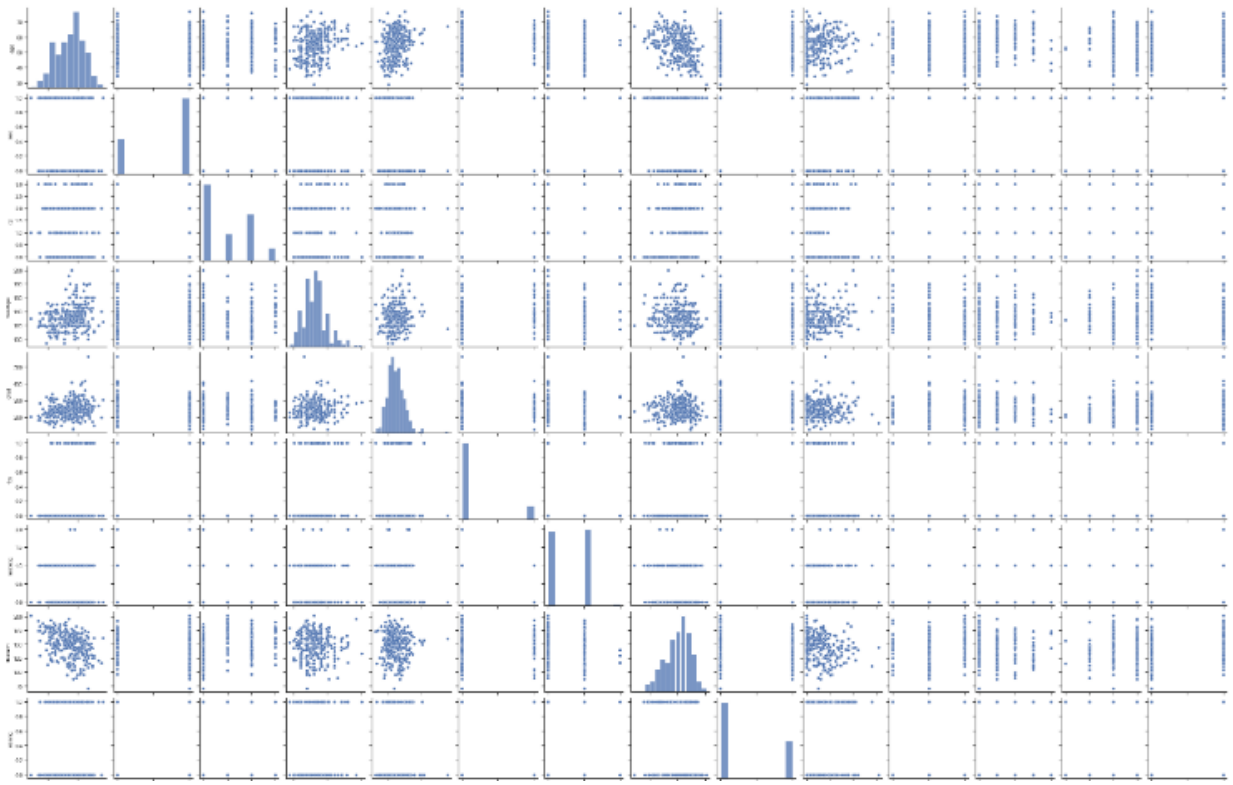
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

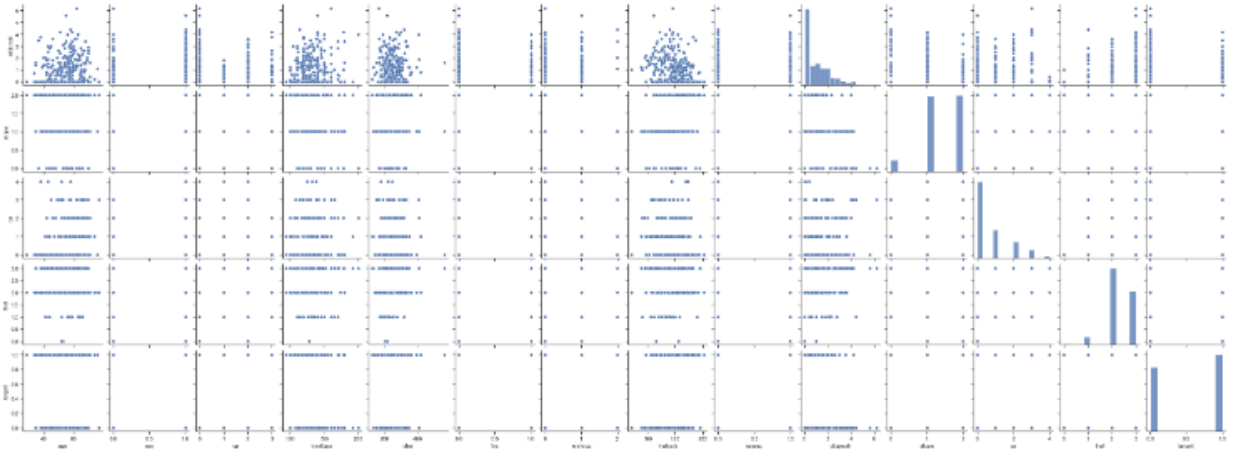
Мы имеем датасет без пропусков и без категориальных признаков

2. Построение графиков, необходимых для понимания структуры данных

```
# Парные диаграммы
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x221bc4f41f0>
```





```
# Скрипичные диаграммы
for col in data.columns:
    sns.violinplot(x=data[col])
    plt.show()
```

3. Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей

Категориальные признаки отсутствуют, их кодирование не требуется.

Вспомогательные признаки для улучшения качества моделей в данном примере мы строить не будем.

Выполним масштабирование данных.

```
# Выполняем масштабирование данных
data['age'] = MinMaxScaler().fit_transform(data[['age']])
data['trestbps'] = MinMaxScaler().fit_transform(data[['trestbps']])
data['chol'] = MinMaxScaler().fit_transform(data[['chol']])
data['thalach'] = MinMaxScaler().fit_transform(data[['thalach']])
```

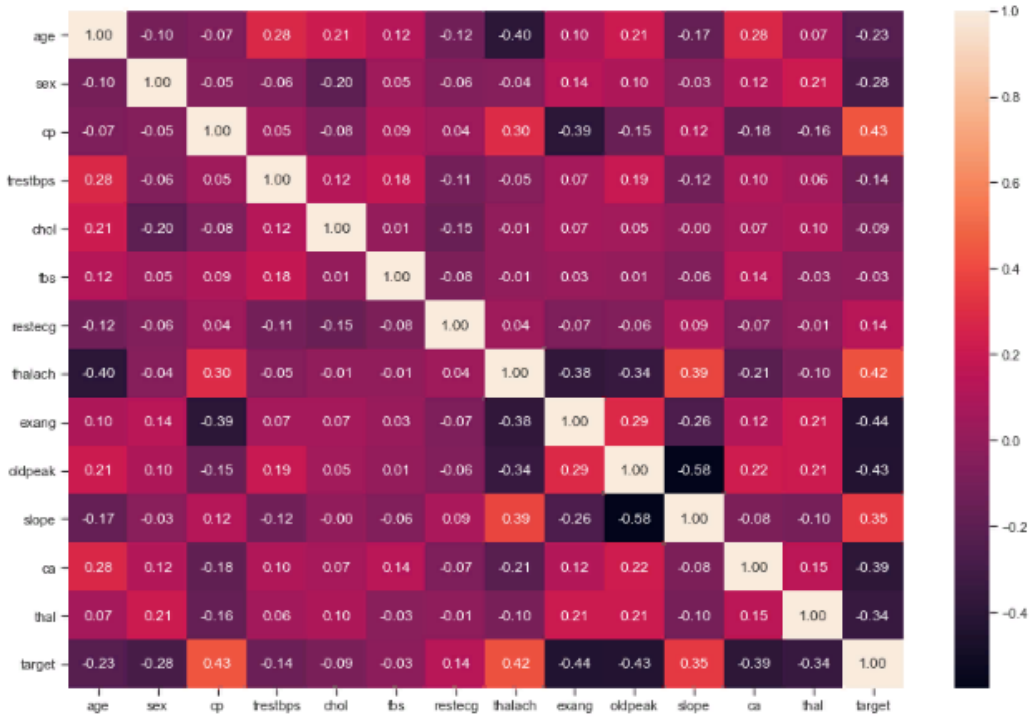
```
# Первые 5 строк датасета
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0.708333	1	3	0.481132	0.244292	1	0	0.603053	0	2.3	0	0	1	1
1	0.166667	1	2	0.339623	0.283105	0	1	0.885496	0	3.5	0	0	2	1
2	0.250000	0	1	0.339623	0.178082	0	0	0.770992	0	1.4	2	0	2	1
3	0.562500	1	1	0.245283	0.251142	0	1	0.816794	0	0.8	2	0	2	1
4	0.583333	0	0	0.245283	0.520548	0	1	0.702290	1	0.6	2	0	2	1

4. Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения

```
fig, ax = plt.subplots(figsize=(15,10))
sns.heatmap(data.corr(), annot=True, fmt='.2f')
```

```
<AxesSubplot:>
```



На основе корреляционной матрицы можно сделать вывод, что сильных линейных зависимостей между признаками нет.

5. Выбор метрик для последующей оценки качества моделей

В качестве метрик для решения задачи регрессии будем использовать:

- Accureacy - процент правильно определённых классов;
- Confusion Matrix - количество верно или ошибочно классифицированных данных, представленное в виде матрица;
- ROC-кривая - показатель качества классификации.

6. Выбор наиболее подходящих моделей для решения задачи классификации

Для задачи классификации мы будем использовать следующие модели:

- Метод ближайших соседей
- Метод опорных векторов
- Решающее дерево
- Случайный лес
- Градиентный бустинг

7. Формирование обучающей и тестовой выборок на основе исходного набора данных

```
data_X = data.drop(columns='target')
data_X
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	0.708333	1	3	0.481132	0.244292	1	0	0.603053	0	2.3	0	0	1
1	0.166667	1	2	0.339623	0.283105	0	1	0.885496	0	3.5	0	0	2
2	0.250000	0	1	0.339623	0.178082	0	0	0.770992	0	1.4	2	0	2
3	0.562500	1	1	0.245283	0.251142	0	1	0.816794	0	0.8	2	0	2
4	0.583333	0	0	0.245283	0.520548	0	1	0.702290	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	0.583333	0	0	0.433962	0.262557	0	1	0.396947	1	0.2	1	0	3
299	0.333333	1	3	0.150943	0.315068	0	1	0.465649	0	1.2	1	0	3
300	0.812500	1	0	0.471698	0.152968	1	1	0.534351	0	3.4	1	2	3
301	0.583333	1	0	0.339623	0.011416	0	1	0.335878	1	1.2	1	1	3
302	0.583333	0	1	0.339623	0.251142	0	0	0.786260	0	0.0	1	1	2

303 rows × 13 columns

```
data_Y = data['target']
data_Y
```

```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: target, Length: 303, dtype: int64
```

```
X_train, X_test, Y_train, Y_test = train_test_split(data_X,  data_Y, test_size=0.20, random_state = 0)
```

```
Y_train=np.ravel(Y_train)
Y_test=np.ravel(Y_test)
```

```
# 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)
```

```
(242, 13)
(61, 13)
(242,)
(61,)
```

8. Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров

```
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]))

def draw_roc_curve(y_true, y_score, pos_label, average):
    fpr, tpr, thresholds = roc_curve(y_true, y_score,
                                     pos_label=pos_label)
    roc_auc_value = roc_auc_score(y_true, y_score, average=average)
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc_value)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
```

Метод ближайших соседей

```
# Решение задачи классификации методом 5 ближайших соседей
Cls_5 = KNeighborsClassifier(n_neighbors = 5)
```

```
Cls_5.fit(X_train, Y_train)
target_1 = Cls_5.predict(X_test)
```

```
print('Процент точности:',accuracy_score(Y_test, target_1))
```

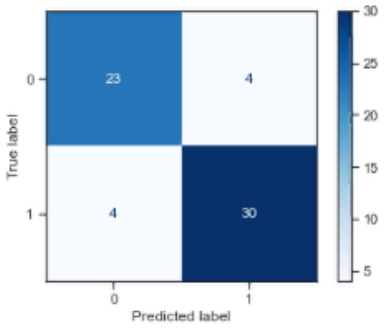
```
Процент точности: 0.8688524590163934
```

```
print('Процент точности для каждого класса:')
print_accuracy_score_for_classes(Y_test, target_1)
```

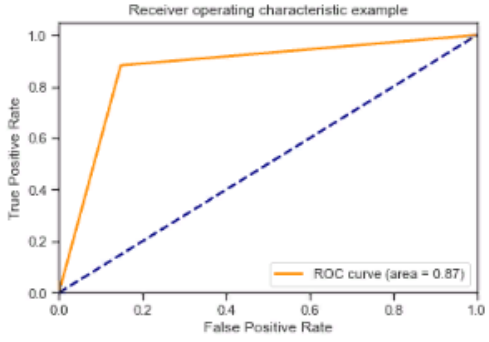
```
Процент точности для каждого класса:
Метка    Accuracy
0        0.8518518518518519
1        0.8823529411764706
```

```
plot_confusion_matrix(Cls_5, X_test, Y_test,
                      display_labels=['0','1'], cmap=plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221c6b9ca30>
```




```
draw_roc_curve(Y_test, target_1, pos_label=1, average='micro')
```



Метод опорных векторов

```
# Решение задачи классификации методом опорных векторов
svc = SVC()
```

```
svc.fit(X_train, Y_train)
target_2 = svc.predict(X_test)
```

```
print('Процент точности:',accuracy_score(Y_test, target_2))
```

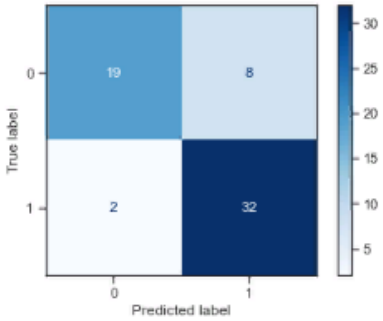
```
Процент точности: 0.8360655737704918
```

```
print('Процент точности для каждого класса:')
print_accuracy_score_for_classes(Y_test, target_2)
```

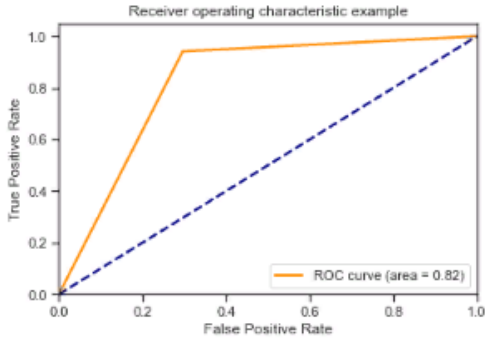
```
Процент точности для каждого класса:
Метка    Accuracy
0        0.7037037037037037
1        0.9411764705882353
```

```
plot_confusion_matrix(svc, X_test, Y_test,
                      display_labels=['0','1'], cmap=plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221cbc737c0>
```




```
draw_roc_curve(Y_test, target_2, pos_label=1, average='micro')
```



Решающее дерево

```
# Решение задачи классификации методом решающего дерева
tree = DecisionTreeClassifier(random_state=0)
```

```
tree.fit(X_train, Y_train)
target_3 = tree.predict(X_test)
```

```
print('Процент точности:',accuracy_score(Y_test, target_3))
```

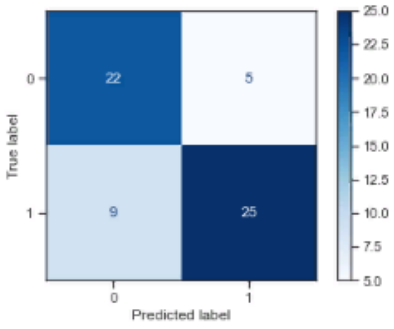
```
Процент точности: 0.7704918032786885
```

```
print('Процент точности для каждого класса:')
print_accuracy_score_for_classes(Y_test, target_3)
```

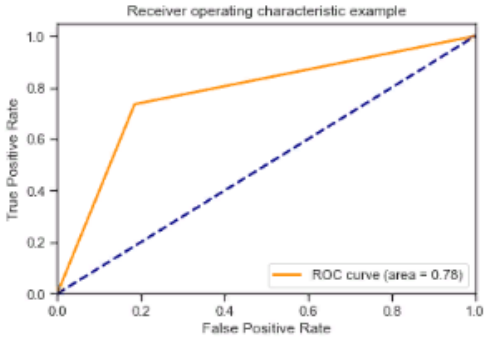
```
Процент точности для каждого класса:
Метка    Accuracy
0        0.8148148148148148
1        0.7352941176470589
```

```
plot_confusion_matrix(tree, X_test, Y_test,
                      display_labels=['0','1'], cmap=plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221cbdeba30>
```




```
draw_roc_curve(Y_test, target_3, pos_label=1, average='micro')
```



Случайный лес

```
# Решение задачи классификации методом случайного леса на 5 деревьях
trees = RandomForestClassifier(n_estimators=5, oob_score=True, random_state=0)
```

```
trees.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] /
```

```
RandomForestClassifier(n_estimators=5, oob_score=True, random_state=0)
```

```
target_4 = trees.predict(X_test)
```

```
print('Процент точности:',accuracy_score(Y_test, target_4))
```

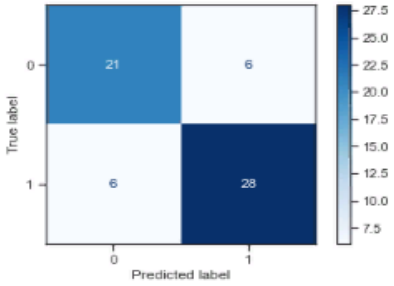
```
Процент точности: 0.8032786885245902
```

```
print('Процент точности для каждого класса:')
print_accuracy_score_for_classes(Y_test, target_4)
```

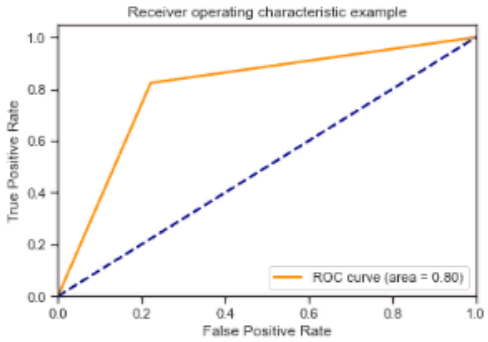
```
Процент точности для каждого класса:
Метка    Accuracy
0        0.7777777777777778
1        0.8235294117647058
```

```
plot_confusion_matrix(trees, X_test, Y_test,
                      display_labels=['0','1'], cmap=plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221cbed8400>
```



```
draw_roc_curve(Y_test, target_4, pos_label=1, average='micro')
```



Градиентный бустинг

```
gr_boost = GradientBoostingClassifier(random_state=0)
```

```
gr_boost.fit(X_train, Y_train)
target_5 = gr_boost.predict(X_test)
```

```
print('Процент точности:',accuracy_score(Y_test, target_5))
```

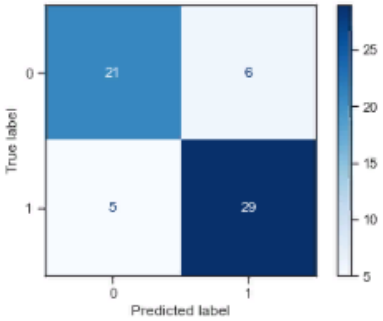
```
Процент точности: 0.819672131147541
```

```
print('Процент точности для каждого класса:')
print_accuracy_score_for_classes(Y_test, target_5)
```

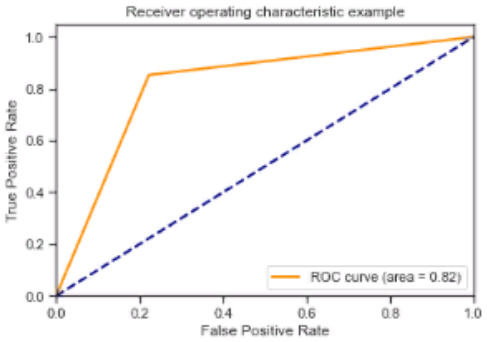
```
Процент точности для каждого класса:
Метка    Accuracy
0        0.7777777777777778
1        0.8529411764705882
```

```
plot_confusion_matrix(gr_boost, X_test, Y_test,
                      display_labels=['0','1'], cmap=plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221cd370d30>
```




```
draw_roc_curve(Y_test, target_5, pos_label=1, average='micro')
```



9. Подбор гиперпараметров для выбранных моделей

Метод ближайших соседей

```
n_range = np.array(range(1,100,1))
tuned_parameters = [{'n_neighbors': n_range}]
tuned_parameters
```

```
[{'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
        18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
        35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
        52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
        69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
        86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])}]
```

```
regr_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5, scoring='accuracy')
regr_gs.fit(X_train, Y_train)
```

```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid=[{'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51,
       52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
       69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
       86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])}],
             scoring='accuracy')
```

```
# Лучшая модель
-regr_gs.best_score_
```

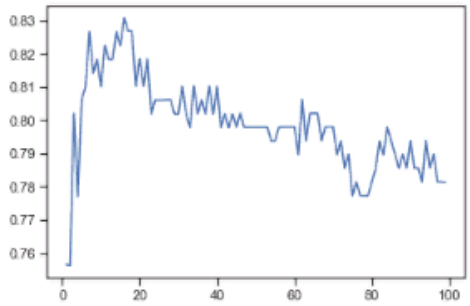
```
-0.8306972789115648
```

```
# Лучшее значение параметров
regr_gs.best_params_
```

```
{'n_neighbors': 16}
```

```
# Изменение качества на тестовой выборке в зависимости от K-соседей
plt.plot(n_range, regr_gs.cv_results_['mean_test_score'])
```

```
[<matplotlib.lines.Line2D at 0x221cd4ec760>]
```



Решающее дерево

```
params = {
    'max_depth': [3, 4, 5, 6],
    'min_samples_leaf': [0.04, 0.06, 0.08],
    'max_features': [0.2, 0.4, 0.6, 0.8]
}
```

```
regr_gs1 = GridSearchCV(estimator=DecisionTreeClassifier(random_state=0),
                        param_grid=params, scoring='accuracy', cv=3, n_jobs=-1)
regr_gs1.fit(X_train, Y_train)
```

```
GridSearchCV(cv=3, estimator=DecisionTreeClassifier(random_state=0), n_jobs=-1,
             param_grid={'max_depth': [3, 4, 5, 6],
                         'max_features': [0.2, 0.4, 0.6, 0.8],
                         'min_samples_leaf': [0.04, 0.06, 0.08]},
             scoring='accuracy')
```

```
# Лучшая модель
-regr_gs1.best_score_
```

```
-0.797685185185185
```

```
# Лучшее значение параметров
regr_gs1.best_params_
```

```
{'max_depth': 4, 'max_features': 0.6, 'min_samples_leaf': 0.04}
```

```
regr_gs1.best_estimator_
```

```
DecisionTreeClassifier(max_depth=4, max_features=0.6, min_samples_leaf=0.04,
                       random_state=0)
```

10. Построение решения для выбранных моделей с подбором гиперпараметров

Метод ближайших соседей

```
# Решение задачи классификации методом 16 ближайших соседей
Cls_16 = KNeighborsClassifier(n_neighbors = 16)
```

```
Cls_16.fit(X_train, Y_train)
target_6 = Cls_16.predict(X_test)
```

```
print('Процент точности:',accuracy_score(Y_test, target_6))
```

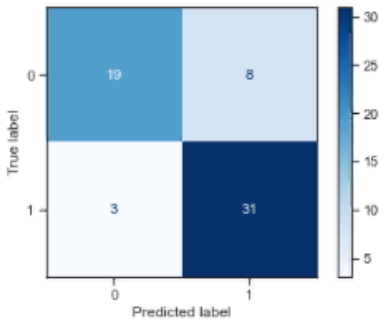
```
Процент точности: 0.819672131147541
```

```
print('Процент точности для каждого класса:')
print_accuracy_score_for_classes(Y_test, target_6)
```

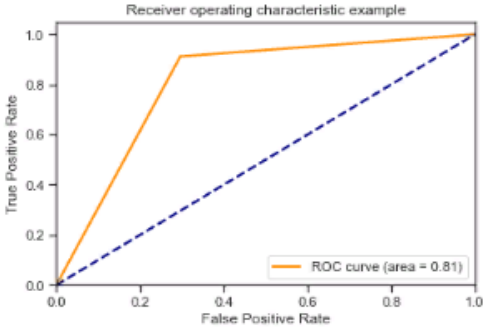
```
Процент точности для каждого класса:
Метка    Accuracy
0        0.7037037037037037
1        0.9117647058823529
```

```
plot_confusion_matrix(Cls_16, X_test, Y_test,
                      display_labels=['0','1'], cmap=plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221cc35ab20>
```




```
draw_roc_curve(Y_test, target_6, pos_label=1, average='micro')
```



Решающее дерево

```
treeNEW = DecisionTreeClassifier(max_depth=4, max_features=0.6, min_samples_leaf=0.04, random_state=0).fit(X_train, Y_train)
```

```
target_7 = treeNEW.predict(X_test)
```

```
print('Процент точности:',accuracy_score(Y_test, target_7))
```

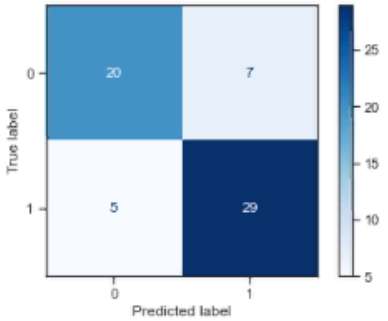
```
Процент точности: 0.8032786885245902
```

```
print('Процент точности для каждого класса:')
print_accuracy_score_for_classes(Y_test, target_7)
```

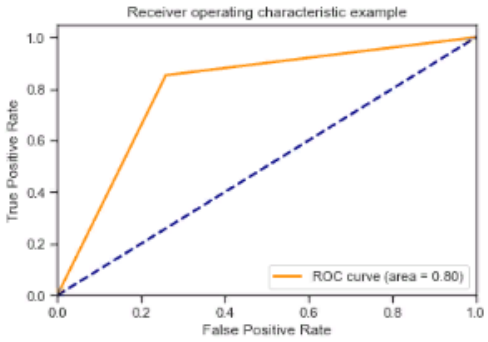
```
Процент точности для каждого класса:
Метка    Accuracy
0        0.7407407407407407
1        0.8529411764705882
```

```
plot_confusion_matrix(treeNEW, X_test, Y_test,
                      display_labels=['0','1'], cmap=plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x221cbba9670>
```




```
draw_roc_curve(Y_test, target_7, pos_label=1, average='micro')
```



11. Формирование выводов о качестве построенных моделей на основе выбранных метрик

Лучшей моделью с точки зрения результатов решения задачи классификации, не смотря на свою простоту, оказался метод ближайших соседей, показавший наибольший процент правильного предсказания принадлежности объектов к классам как в целом, так и для каждого класса в отдельности.