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 КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Решение задач машинного обучения

Студент __ИУ5-64б__ _____ _____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ _____
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

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 _____
(Индекс)
_____ _____
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине _____Технологии машинного обучения_____

Студент группы __ИУ5-64б__

_____Савельев Алексей Александрович_____
(Фамилия, имя, отчество)

Тема курсового проекта ____Решение задач машинного обучения_____
_____
_____

Направленность КП (учебный, исследовательский, практический, производственный, др.)
_____
Источник тематики (кафедра, предприятие, НИР) _____

График выполнения проекта: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Задание___типовое исследование: решение задачи машинного обучения на основе_____
____материалов дисциплины. Выполняется студентом единолично_____
_____
_____
_____

Оформление курсового проекта:

Расчетно-пояснительная записка на _21_ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
_____
_____
_____
_____

Дата выдачи задания « ___ » _____ 20__ г.

Руководитель курсового проекта _____ _____
(Подпись, дата) (И.О.Фамилия)

Студент _____ _____
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Курсовой проект по дисциплине "Технологии машинного обучения"

Выбор набора данных для построения моделей машинного обучения и дальнейшей оценки их качества

В качестве набора данных мы будем использовать набор данных по показателям вина, предназначенные для решения задачи классификации:
Ссылка на набор данных
Набор данных состоит из одного файла wine.csv включающего в себя следующие колонки:

id	parametr	sense
0	type	Тип
1	fixed acidity	Фиксированная кислотность
2	volatile acidity	Летучая кислотность
3	citric acid	Лемонная кислота
4	residual sugar	Остаточный сахар
5	chlorides	Хлориды
6	free sulfur dioxide	Свободный диоксид серы
7	total sulfur dioxide	Общий диоксид серы
8	density	Плотность
9	pH	Кислотность (pH)
10	sulphates	Сульфаты
11	alcohol	Алкоголь
12	quality	Качество (оценка от 0 до 10 param)

Для решения задачи классификации в качестве целевого признака будем использовать параметр "type" (тип (вид) вина). Поскольку признак содержит только два уникальных значения, то это задача бинарной классификации.

Первостепенный импорт базовых библиотек

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

pd.set_option('display.max_columns', None)
```

Чтение набора данных и его первоначальный анализ

In [2]:
```
# читаем данные набора

data_wine = pd.read_csv('/Users/savelevaa/Desktop/ml-labs/Coursework/data/wine.csv', sep=",")

data_wine.head(5)
```

Out[2]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [3]:
```
# срез данных
data_wine.shape
```

Out[3]:
```
(6497, 13)
```

In [4]:
```
# типы данных столбцов
data_wine.dtypes
```

Out[4]:
```
type                   object
fixed acidity         float64
volatile acidity      float64
citric acid           float64
residual sugar        float64
chlorides             float64
free sulfur dioxide   float64
total sulfur dioxide  float64
density               float64
pH                    float64
sulphates             float64
alcohol               float64
quality                 int64
dtype: object
```

In [5]:
```
# суммы пропусков по столбцам
data_wine.isnull().sum()
```

Out[5]:
```
type                   0
fixed acidity         10
volatile acidity       8
citric acid            3
residual sugar         2
chlorides              2
free sulfur dioxide    0
total sulfur dioxide   0
density                0
pH                     9
sulphates              4
alcohol                0
quality                0
dtype: int64
```

В некоторых колонках есть пропуски данных. Оставим в наборе данных только те строки, в которых есть значения:

In [6]:
```
# Оставим только непустые значения
for col in data_wine.columns:
    data_wine = data_wine[data_wine[col].notna()]
data_wine.isnull().sum()
```

Out[6]:
```
type                   0
fixed acidity          0
volatile acidity       0
citric acid            0
residual sugar         0
chlorides              0
free sulfur dioxide    0
total sulfur dioxide   0
density                0
pH                     0
sulphates              0
alcohol                0
quality                0
dtype: int64
```

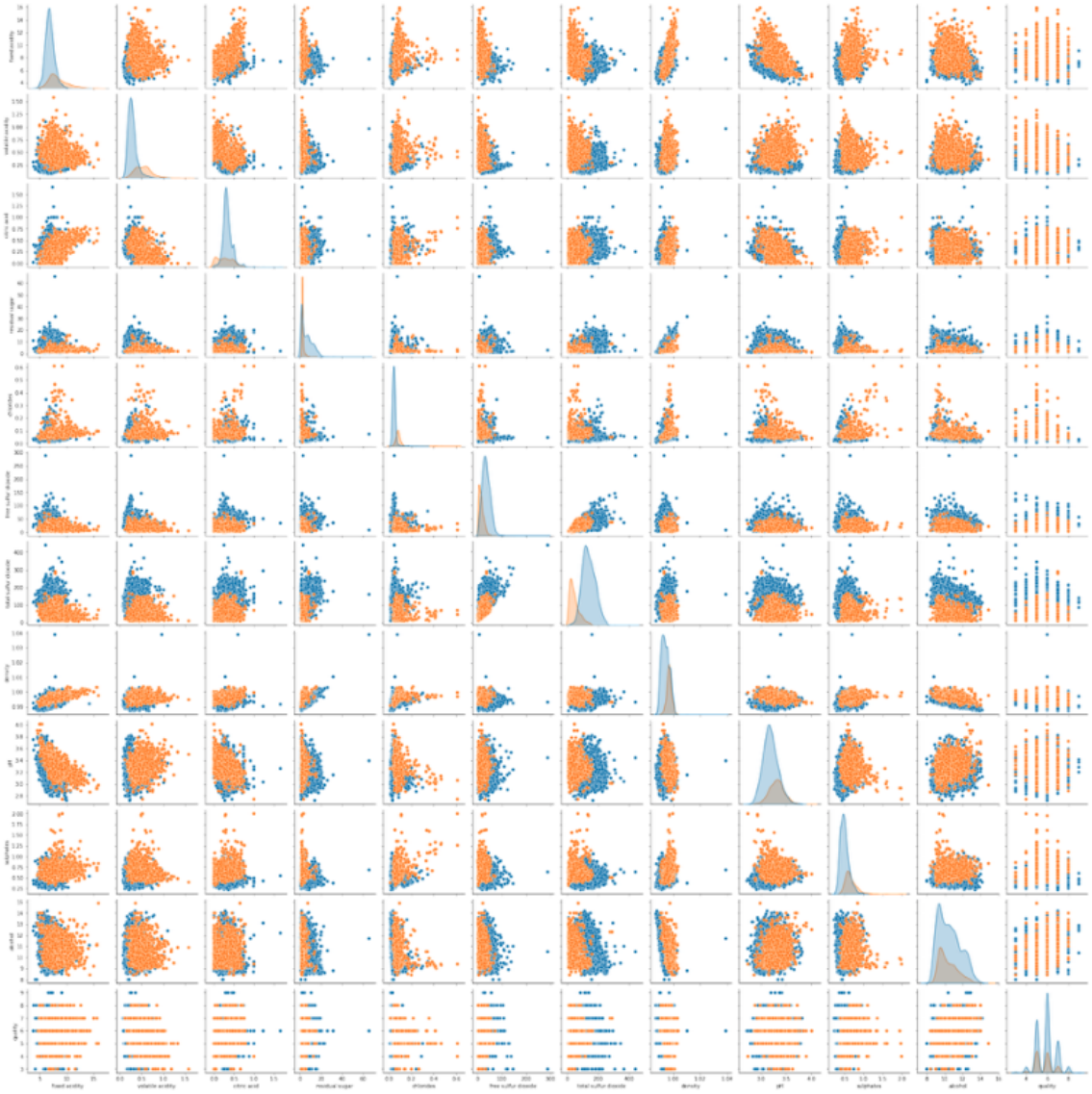
Отлично, набор данных больше не имеет пропусков, поэтому мы можем перейте к анализу набора данных.

Анализ данных

Выведем парные диаграммы и заметить зависимости

In [7]:
```
sns.pairplot(data_wine, hue="type")
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x7f80e041c160>

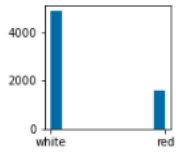


Однозначно зависимости определить не получается, тогда, после настройки целевого признака, определим зависимости по корреляционной матрице.

In [8]:
```
# Проверим значения целевого признака
types = data_wine['type'].unique()
types
```

Out[8]: array(['white', 'red'], dtype=object)

In [9]:
```
# Оценим дисбаланс классов для Occupancy
fig, ax = plt.subplots(figsize=(2,2))
plt.hist(data_wine['type'])
plt.show()
```



Дисбаланс незначительный, можем продолжать.

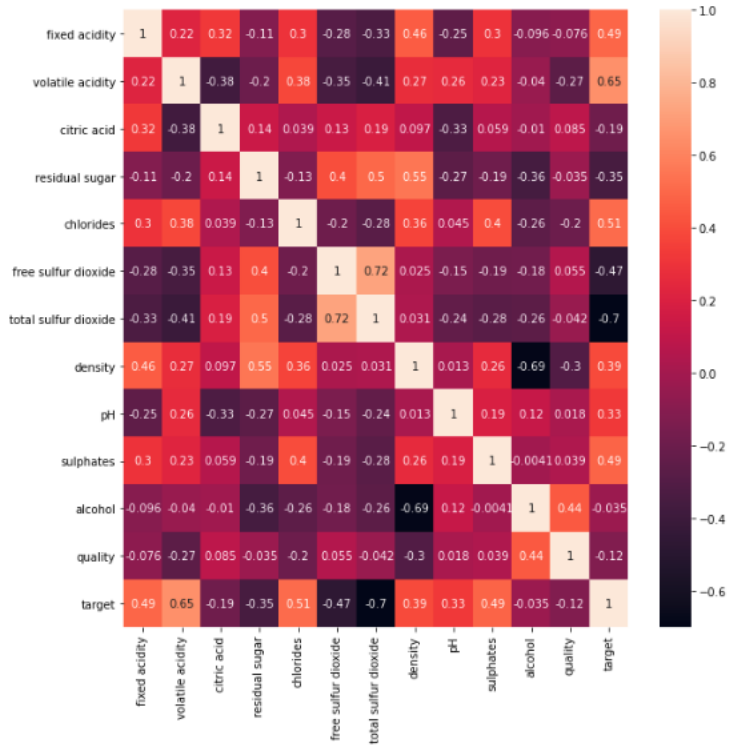
In [10]:
```
# функция классификации целевого пизнака
def type_to_int(w_type: str) -> int:
    if w_type == "white":
        result = 0
    elif w_type == "red":
        result = 1
    return result
```

In [11]:
```
# формирование числового целевого признака для задачи классификации
data_wine['target'] = \
data_wine.apply(lambda row: type_to_int(row['type']), axis=1)
```

Корреляционная матрица

Построим корреляционную матрицу, для проверки зависимостей между параметрами набора данных

In [12]:
```
fig, ax = plt.subplots(figsize=(10,10))
ax = sns.heatmap(data_wine.corr(), annot=True)
plt.show(fig)
```



Можно отметить следующий признаки, коррелирующие с целевым (type (target)):

- volatile acidity (0.65)
- chlorides (0.51)
- fixed acidity (0.49)
- sulphates (0.49)
- density

  Эти признаки будем использовать для решения задачи классификации.

Масштабирование данных

произведем мастабирование данных с использыванием метода MinMaxScaler():

In [13]:
```
# Получим названия колонок
data_cols = list()
temp_cols = data_wine.columns
for col in temp_cols:
    data_cols.append(col)
data_cols.pop(0)
data_cols
```

Out[13]:
```
['fixed acidity',
 'volatile acidity',
 'citric acid',
 'residual sugar',
 'chlorides',
 'free sulfur dioxide',
 'total sulfur dioxide',
 'density',
 'pH',
 'sulphates',
 'alcohol',
 'quality',
 'target']
```

In [14]:
```
from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler()
scaled_data = sc.fit_transform(data_wine[data_cols])
```

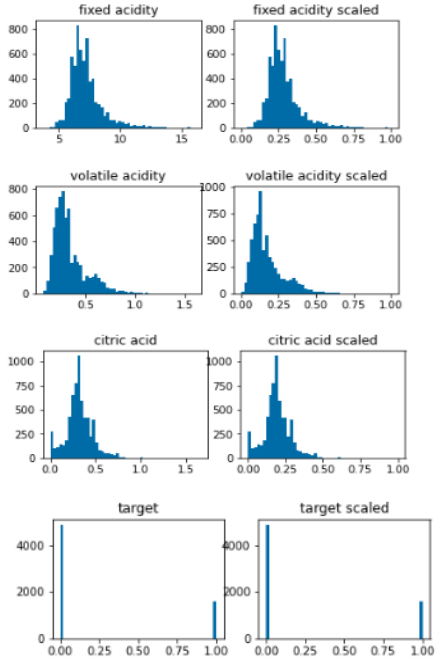
In [15]:
```
# Добавим масштабированные данные в набор данных
for i in range(len(data_cols)):
    col = data_cols[i]
    new_col_name = col + ' scaled'
    data_wine[new_col_name] = scaled_data[:,i]
```

In [16]:
```
data_wine.head(5)
```

Out[16]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	target	fixed acidity scaled	volatile acidity scaled	citric acid scaled	residual sugar scaled
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6	0	0.264463	0.126667	0.216867	0.308282
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6	0	0.206612	0.146667	0.204819	0.015337
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6	0	0.355372	0.133333	0.240964	0.096626
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	0	0.280992	0.100000	0.192771	0.121166
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6	0	0.280992	0.100000	0.192771	0.121166

In [17]:
```
# Проверим, что масштабирование не повлияло на распределение данных
for col in data_cols:
    col_scaled = col + ' scaled'

    fig, ax = plt.subplots(1, 2, figsize=(6,2))
    ax[0].hist(data_wine[col], 50)
    ax[1].hist(data_wine[col_scaled], 50)
    ax[0].title.set_text(col)
    ax[1].title.set_text(col_scaled)
    plt.show()
```



Как видно по графикам распределения значений не изменилось после масшабирования данных.

Метрики

Для оценки качества моделей машинного обучения для задачи классификации используем следующий метрики:

Метрика accuracy:
Метрика вычисляет процент (долю в диапазоне от 0 до 1) правильно определенных классов.

Эту метрику обычно переводят как "точность", но перевод не является удачным, потому что совпадает с переводом для другой метрики - "precision".

Доля верно предсказанных классификатором положительных и отрицательных объектов.

Метрика precision:
Можно переводить как точность, но такой перевод совпадает с переводом метрики "accuracy".

Доля верно предсказанных классификатором положительных объектов, из всех объектов, которые классификатор верно или неверно определил как положительные.

Метрика recall:
Можно переводить как полнота. Доля верно предсказанных классификатором положительных объектов, из всех действительно положительных объектов.

Метрика F-мера:
Это метрика, включающая в себя две предидущих.

Метрика ROC-кривая и ROC AUC:
Используется для оценки качества бинарной классификации. Основана на вычислении следующих характеристик:

- True Positive Rate, откладывается по оси ординат. Совпадает с recall.
- False Positive Rate, откладывается по оси абсцисс. Показывает какую долю из объектов отрицательного класса алгоритм предсказал неверно.

В качестве количественной метрики используется площадь под кривой - ROC AUC (Area Under the Receiver Operating Characteristic Curve). Чем ниже проходит кривая тем меньше ее площадь и тем хуже качество классификатора.

Сохранение и визуализация метрик

Инициализируем специальный класс, который позволит нам сохранять метрики качества построенных моделей машинного обучения и реализует их визуализацию.

In [18]:
```
class MetricLogger:

    def __init__(self):
        self.df = pd.DataFrame(
            {'metric': pd.Series([], dtype='str'),
            'alg': pd.Series([], dtype='str'),
            'value': pd.Series([], dtype='float')})

    def add(self, metric, alg, value):
        """
        Добавление значения
        """
        # Удаление значения если оно уже было ранее добавлено
        self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)].index, inplace = True)
        # Добавление нового значения
        temp = [{'metric':metric, 'alg':alg, 'value':value}]
        self.df = self.df.append(temp, ignore_index=True)

    def get_data_for_metric(self, metric, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        temp_data = self.df[self.df['metric']==metric]
        temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
        return temp_data_2['alg'].values, temp_data_2['value'].values

    def plot(self, str_header, metric, ascending=True, figsize=(5, 5)):
        """
        Вывод графика
        """
        array_labels, array_metric = self.get_data_for_metric(metric, ascending)
        fig, ax1 = plt.subplots(figsize=figsize)
        pos = np.arange(len(array_metric))
        rects = ax1.barh(pos, array_metric,
                         align='center',
                         height=0.5,
                         tick_label=array_labels)
        ax1.set_title(str_header)
        for a,b in zip(pos, array_metric):
            plt.text(0.5, a-0.05, str(round(b,3)), color='white')
        plt.show()
```

Выпишем коррелирующие с целевым признаки в отдельный список:

In [19]:
```
# Признаки для задачи классификации
task_clas_cols = ['volatile acidity scaled', 'density scaled',
                  'sulphates scaled', 'chlorides scaled', 'fixed acidity scaled']
```

Формирование обучающей и тестовой выборок на основе исходного (но уже подготовленного) набора данных

In [20]:
```
# Выборки для задачи классификации
from sklearn.model_selection import train_test_split

cl_X_train, cl_X_test, cl_Y_train, cl_Y_test = train_test_split(
    data_wine[task_clas_cols], data_wine['target'].values, test_size=0.5, random_state=1)

cl_X_train.shape, cl_X_test.shape, cl_Y_train.shape, cl_Y_test.shape
```

Out[20]:
```
((3231, 5), (3232, 5), (3231,), (3232,))
```

Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.

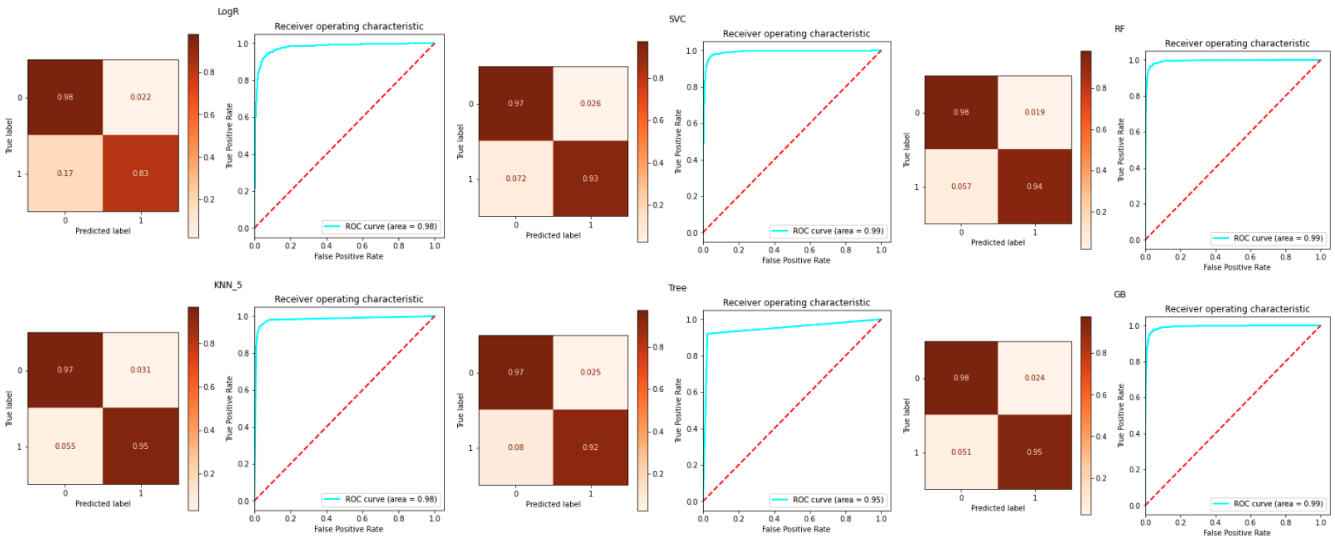
In [21]:
```
# импорт библиотек используемых моделей
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

# Модели
clas_models = {'LogR': LogisticRegression(),
               'KNN_5':KNeighborsClassifier(n_neighbors=5),
               'SVC':SVC(probability=True),
               'Tree':DecisionTreeClassifier(),
               'RF':RandomForestClassifier(),
               'GB':GradientBoostingClassifier()}
```

In [22]:
```
# Создаем хранитель метрик
metricLogger1 = MetricLogger()
```

In [23]:
```
# Отрисовка ROC-кривой
def draw_roc_curve(y_true, y_score, ax, pos_label=1, average='micro'):
    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
    ax.plot(fpr, tpr, color='cyan',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc_value)
    ax.plot([0, 1], [0, 1], color='red', lw=lw, linestyle='--')
    ax.set_xlim([0.0, 1.0])
    ax.set_xlim([0.0, 1.05])
    ax.set_xlabel('False Positive Rate')
    ax.set_ylabel('True Positive Rate')
    ax.set_title('Receiver operating characteristic')
    ax.legend(loc="lower right")
```

In [24]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import plot_confusion_matrix

def clas_train_model(model_name, model, clasMetricLogger):
    model.fit(cl_X_train, cl_Y_train)
    # Предсказание значений
    Y_pred = model.predict(cl_X_test)
    # Предсказание вероятности класса "1" для roc auc
    Y_pred_proba_temp = model.predict_proba(cl_X_test)
    Y_pred_proba = Y_pred_proba_temp[:,1]

    accuracy = accuracy_score(cl_Y_test, Y_pred)
    precision = precision_score(cl_Y_test, Y_pred)
    recall = recall_score(cl_Y_test, Y_pred)
    f1 = f1_score(cl_Y_test, Y_pred)
    roc_auc = roc_auc_score(cl_Y_test, Y_pred_proba)

    clasMetricLogger.add('accuracy', model_name, accuracy)
    clasMetricLogger.add('precision', model_name, precision)
    clasMetricLogger.add('recall', model_name, recall)
    clasMetricLogger.add('f1', model_name, f1)
    clasMetricLogger.add('roc_auc', model_name, roc_auc)

    fig, ax = plt.subplots(ncols=2, figsize=(10,5))
    draw_roc_curve(cl_Y_test, Y_pred_proba, ax[1])
    plot_confusion_matrix(model, cl_X_test, cl_Y_test, ax=ax[0],
                          display_labels=['0','1'],
                          cmap=plt.cm.Oranges, normalize='true')
    fig.suptitle(model_name)
    plt.show()
```

In [25]:
```
for model_name, model in clas_models.items():
    clas_train_model(model_name, model, metricLogger1)
```



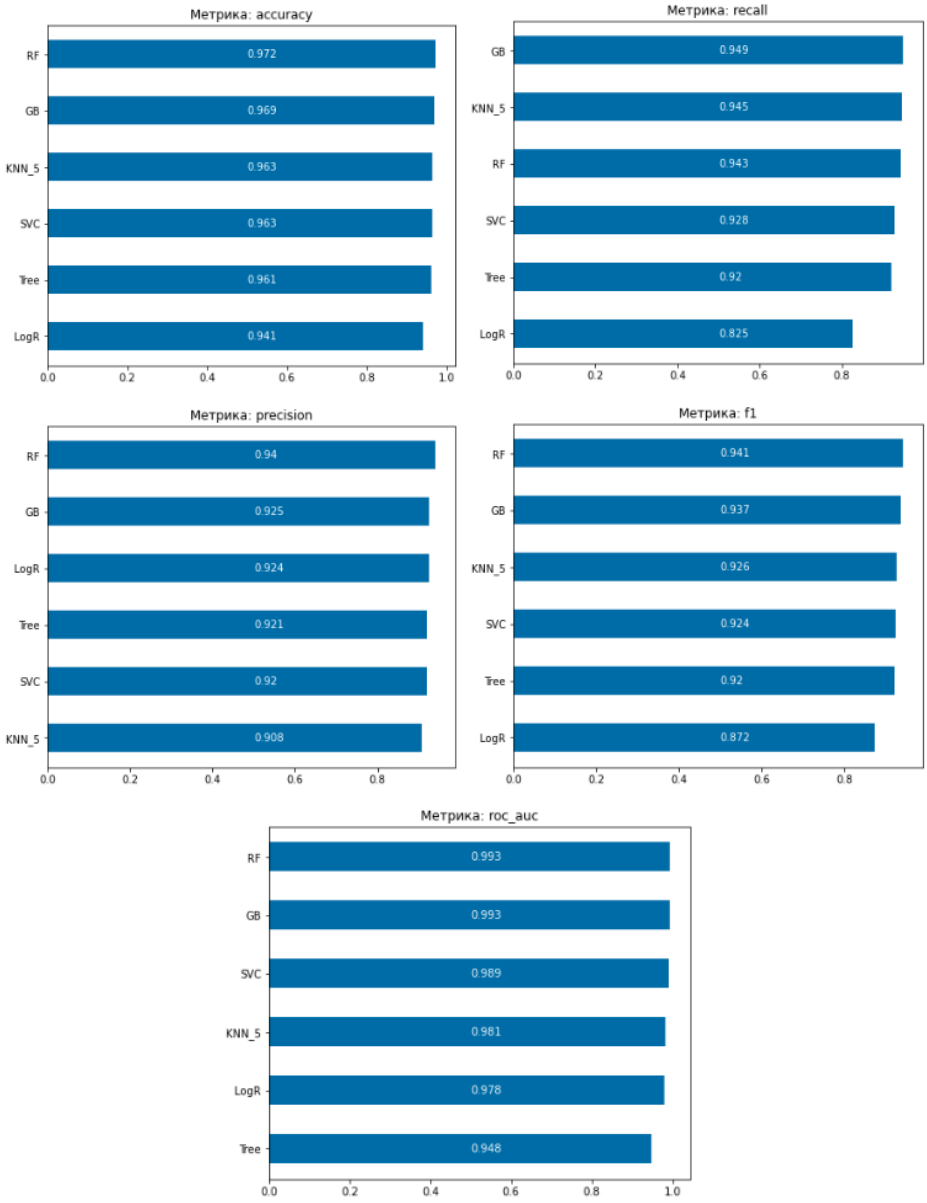
Оценка качества моделей машинного обучения

Далее будут представленны графики различных оценочных метрик, по которым можно будет определить наилучшую модель:

In [26]:
```
# Метрики качества модели
clas_metrics = metricLogger1.df['metric'].unique()
clas_metrics
```

Out[26]:
```
array(['accuracy', 'precision', 'recall', 'f1', 'roc_auc'], dtype=object)
```

In [27]:
```
# Построим графики метрик качества модели
for metric in clas_metrics:
    metricLogger1.plot('Метрика: ' + metric, metric, figsize=(7, 6))
```



Выше представлены промежуточные оценки по baseline-моделям. Финальный вывод по качеству моделей сделаем после подбора гиперпараметров и построению по ним моделей.

Листинг веб приложения:

```
import streamlit as st
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

class MetricLogger:

    def __init__(self):
        self.df = pd.DataFrame(
            {'metric': pd.Series([], dtype='str'),
            'alg': pd.Series([], dtype='str'),
            'value': pd.Series([], dtype='float')})

    def add(self, metric, alg, value):
        """
        Добавление значения
        """
        # Удаление значения если оно уже было ранее добавлено
        self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)].index, inplace = True)
        # Добавление нового значения
        temp = [{'metric':metric, 'alg':alg, 'value':value}]
        self.df = self.df.append(temp, ignore_index=True)

    def get_data_for_metric(self, metric, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        temp_data = self.df[self.df['metric']==metric]
        temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
        return temp_data_2['alg'].values, temp_data_2['value'].values

    def get_metricts_for_alg(self, metrics, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        to_return = list()
        for metric in metrics:
            temp_data = self.df[self.df['metric']==metric]
            temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
            to_return.append((temp_data_2['alg'].values, metric, temp_data_2['value'].values))
        return to_return


    def plot(self, str_header, metric, ascending=True, figsize=(5, 5)):
```

```
        """
        Вывод графика
        """
        array_labels, array_metric = self.get_data_for_metric(metric, ascending)
        figsize = (5, 1*len(array_labels))
        fig, ax1 = plt.subplots(figsize=figsize)
        pos = np.arange(len(array_metric))
        rects = ax1.barh(pos, array_metric,
                         align='center',
                         height=0.5,
                         tick_label=array_labels)
        ax1.set_title(str_header)
        for a,b in zip(pos, array_metric):
            plt.text(0.5, a-0.05, str(round(b,4)), color='white')
        st.pyplot(fig)

    def abs_plot(self, metrics, ascending=True, figsize=(5, 5)):
        """
        Вывод графика
        """
        array_metric = list()
        metrics_data_list = self.get_metricts_for_alg(metrics, ascending)
        for i in metrics_data_list:
            array_metric.append(i[2])
        st.text(metrics_data_list)
        figsize = (5, 1*len(metrics_data_list))
        fig, ax1 = plt.subplots(figsize=figsize)
        pos = np.arange(len(array_metric[0]))
        rects = ax1.barh(pos, metrics_data_list[0][0],
                         align='center',
                         height=0.5,
                         tick_label=metrics)
        ax1.set_title(metrics_data_list[0][0])
        for a,b in zip(pos, array_metric):
            plt.text(0.5, a-0.05, str(round(b,3)), color='white')
        st.pyplot(fig)

# ЗАГРУЗКА ДАННЫХ
@st.cache
def load_data():
    data_wine = pd.read_csv('/Users/savelevaa/Desktop/ml-labs/Coursework/data/wine.csv', sep=",")
    return data_wine

# функция классификации целевого пизнака
def type_to_int(w_type: str) -> int:
    if w_type == "white":
        result = 0
    elif w_type == "red":
        result = 1
    return result

def scaler(data):
    # Получим названия колонок
```

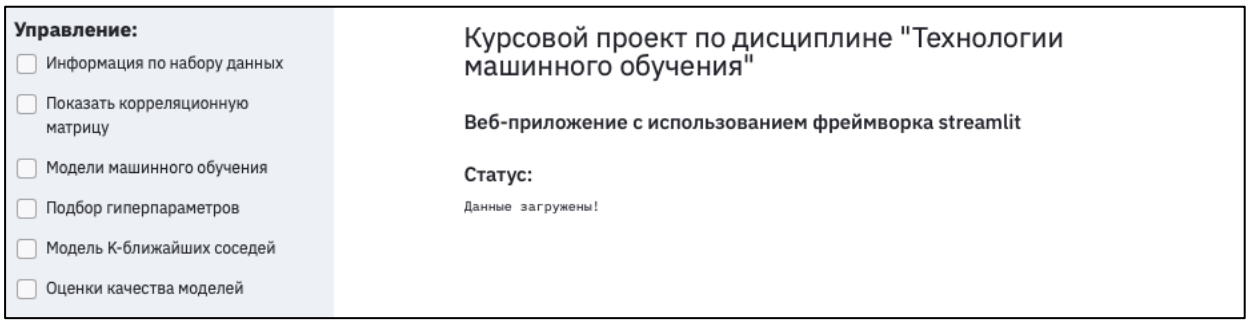
```
    data_cols = list()
    temp_cols = data.columns
    for col in temp_cols:
        data_cols.append(col)
    data_cols.pop(0)
    sc = MinMaxScaler()
    scaled_data = sc.fit_transform(data[data_cols])
    # Добавим масштабированные данные в набор данных
    for i in range(len(data_cols)):
        col = data_cols[i]
        new_col_name = col + ' scaled'
        data[new_col_name] = scaled_data[:,i]
    return data


@st.cache
def prepare_data(data):
    # Оставим только непустые значения
    for col in data.columns:
        data = data[data[col].notna()]
    data['target'] = \
        data.apply(lambda row: type_to_int(row['type']), axis=1)
    data = scaler(data)
    # Признаки для задачи классификации
    task_clas_cols = ['volatile acidity scaled', 'density scaled',
                  'sulphates scaled', 'chlorides scaled', 'fixed acidity scaled']
    # Выборки для задачи классификации
    cl_X_train, cl_X_test, cl_Y_train, cl_Y_test = train_test_split(
        data[task_clas_cols], data['target'].values, test_size=0.5, random_state=1)
    return (cl_X_train, cl_X_test, cl_Y_train, cl_Y_test)


# Отрисовка ROC-кривой
def draw_roc_curve(y_true, y_score, ax, pos_label=1, average='micro'):
    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
    ax.plot(fpr, tpr, color='cyan',
             lw=lw, label='ROC curve (area = %0.4f)' % roc_auc_value)
    ax.plot([0, 1], [0, 1], color='red', lw=lw, linestyle='--')
    ax.set_xlim([0.0, 1.0])
    ax.set_xlim([0.0, 1.05])
    ax.set_xlabel('False Positive Rate')
    ax.set_ylabel('True Positive Rate')
    ax.set_title('Receiver operating characteristic')
    ax.legend(loc="lower right")

@st.cache(suppress_st_warning=True)
def clas_train_model(model_names, X_train, X_test, Y_train, Y_test, clasMetricLogger=None, Kn=None):
    current_models_list = []
    roc_auc_list = []
```

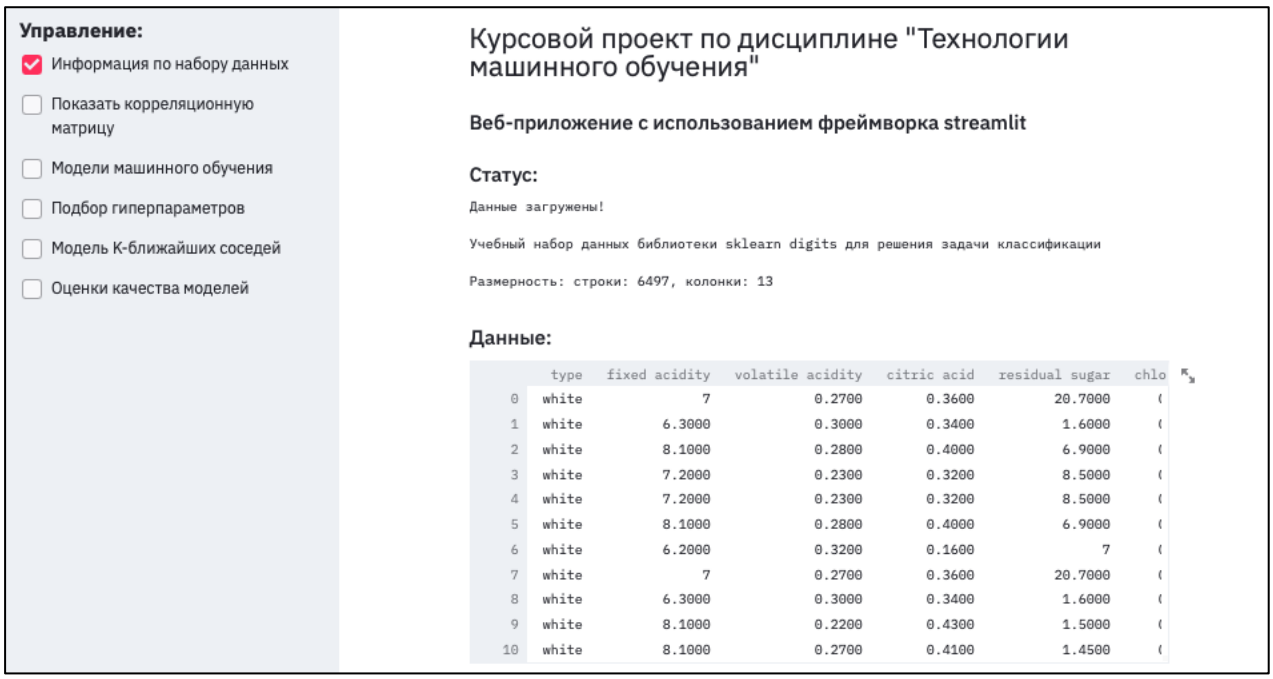
```
    for model_name in model_names:
        model = None
        if Kn == None:
            model = clas_models[model_name]
        else:
            model = KNeighborsClassifier(n_neighbors=Kn)
        model.fit(X_train, Y_train)
        # Предсказание значений
        Y_pred = model.predict(X_test)
        # Предсказание вероятности класса "1" для roc auc
        Y_pred_proba_temp = model.predict_proba(X_test)
        Y_pred_proba = Y_pred_proba_temp[:,1]

        roc_auc = roc_auc_score(Y_test, Y_pred_proba)

        if clasMetricLogger != None:
            accuracy = accuracy_score(Y_test, Y_pred)
            precision = precision_score(Y_test, Y_pred)
            recall = recall_score(Y_test, Y_pred)
            f1 = f1_score(Y_test, Y_pred)
            clasMetricLogger.add('accuracy', model_name, accuracy)
            clasMetricLogger.add('precision', model_name, precision)
            clasMetricLogger.add('recall', model_name, recall)
            clasMetricLogger.add('f1', model_name, f1)
            clasMetricLogger.add('roc_auc', model_name, roc_auc)

        roc_auc = roc_auc_score(Y_test, Y_pred_proba)
        current_models_list.append(model_name)
        roc_auc_list.append(roc_auc)

        #Отрисовка ROC-кривых
        fig, ax = plt.subplots(ncols=2, figsize=(10,5))
        draw_roc_curve(Y_test, Y_pred_proba, ax[1])
        plot_confusion_matrix(model, X_test, Y_test, ax=ax[0],
                        display_labels=['0','1'],
                        cmap=plt.cm.Oranges, normalize='true')
        fig.suptitle(model_name)
        st.pyplot(fig)

        # if Kn != None:
        #     for metric in list(clasMetricLogger.df['metric'].unique()):
        #         clasMetricLogger.plot('Метрика: ' + metric, metric, figsize=(7, 6))
        #     # clasMetricLogger.abs_plot(list(clasMetricLogger.df['metric'].unique()), figsize=(7, 6))

st.sidebar.header('Управление:')


info = st.sidebar.checkbox('Информация по набору данных')
corr = st.sidebar.checkbox('Показать корреляционную матрицу')
print_models = st.sidebar.checkbox('Модели машинного обучения')
gparams = st.sidebar.checkbox('Подбор гиперпараметров')
absolute = st.sidebar.checkbox('Модель K-ближайших соседей')
evaluation = st.sidebar.checkbox('Оценки качества моделей')
```

```
# Модели
models_list = ['LogR', 'KNN_5', 'SVC', 'Tree', 'RF', 'GB']
clas_models = {'LogR': LogisticRegression(),
               'KNN_5':KNeighborsClassifier(n_neighbors=5),
               'SVC':SVC(probability=True),
               'Tree':DecisionTreeClassifier(),
               'RF':RandomForestClassifier(),
               'GB':GradientBoostingClassifier()}

params_list = ['GridSearch', 'RandomSearch']

# Создаем хранитель метрик
metricLogger1 = MetricLogger()

#st.sidebar.header('Параметры:')

st.header('Курсовой проект по дисциплине "Технологии машинного обучения"')
st.subheader('Веб-приложение с использованием фреймворка streamlit')
st.subheader('Статус:')

data_load_state = st.text('Загрузка данных...')
data = load_data()
data_load_state.text('Данные загружены!')

train_test_tpl = prepare_data(data)


if info:
    st.text('Учебный набор данных библиотеки sklearn digits для решения задачи классификации')
    st.text(f'Размерность: строки: {data.shape[0]}, колонки: {data.shape[1]}')
    st.subheader('Данные:')
    st.write(data)

if corr:
    fig1, ax = plt.subplots(figsize=(12,6))
    sns.heatmap(data.corr(), fmt='.2f', annot=True)
    st.pyplot(fig1)

if print_models:
    st.sidebar.header('Модели машинного обучения')
    st.subheader('Модели:')
    models_select = st.sidebar.multiselect('Выберите модели', models_list)
    clas_train_model(models_select, train_test_tpl[0], train_test_tpl[1],
                     train_test_tpl[2], train_test_tpl[3], metricLogger1)

if gparams:
    st.subheader('Подбор гиперпараметров:')
    st.sidebar.header('Параметры:')
    param_select = st.sidebar.multiselect('Выберите метод подбора:', params_list)
    step_slider = st.sidebar.slider('Шаг для соседей:', min_value=1, max_value=100, value=20, step=1)
    max_border_neighbor = st.sidebar.slider('Макс кол-во соседей:', min_value=50, max_value=1000,
value=200, step=5)

    #Количество записей
```

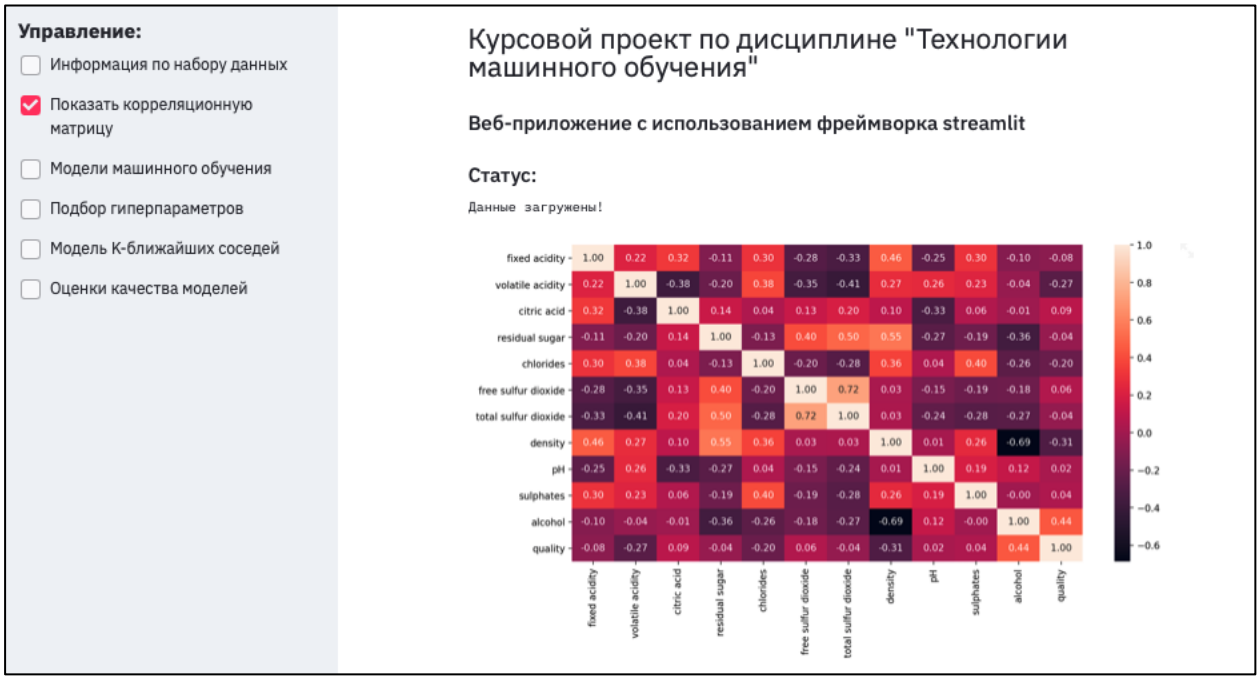
```
    data_len = data.shape[0]
    st.write('Количество строк в наборе данных - {}'.format(data_len))
    # Подбор гиперпараметра
    n_range_list = list(range(3,max_border_neighbor, step_slider))
    n_range = np.array(n_range_list)
    st.write('Возможные значения соседей - {}'.format(n_range))
    tuned_parameters = [{'n_neighbors': n_range}]



    st.subheader('Оценка качества модели')
    if "GridSearch" in param_select:
        clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5, scoring='roc_auc')
        clf_gs.fit(train_test_tpl[0], train_test_tpl[2])

        st.subheader("Grid Search")
        st.write('Лучшее значение параметров - {}'.format(clf_gs.best_params_))
        # Изменение качества на тестовой выборке в зависимости от K-соседей
        fig1, ax1 = plt.subplots(figsize=(10,5))
        ax1 = plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
        st.pyplot(fig1)

    if "RandomSearch" in param_select:
        clf_rs = RandomizedSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5, scoring='roc_auc')
        clf_rs.fit(train_test_tpl[0], train_test_tpl[2])

        st.subheader("Random Search")
        st.write('Лучшее значение параметров - {}'.format(clf_rs.best_params_))
        # Изменение качества на тестовой выборке в зависимости от K-соседей
        fig1, ax1 = plt.subplots(figsize=(10,5))
        ax1 = plt.plot(n_range, clf_rs.cv_results_['mean_test_score'])
        st.pyplot(fig1)

metricLogger2 = MetricLogger()

if absolute:
    st.sidebar.header('Модель K-ближайших соседей:')
    st.subheader("Модель K-ближайших соседей")
    neighbors_num = st.sidebar.slider('K-ближайших соседей:', min_value=1, max_value=100, value=20,
step=1)
    clas_train_model([f"KNN_{neighbors_num}"], train_test_tpl[0], train_test_tpl[1],
                     train_test_tpl[2], train_test_tpl[3], metricLogger1, neighbors_num)
if evaluation:
    # Метрики качества модели
    clas_metrics = list(metricLogger1.df['metric'].unique())

    if len(clas_metrics) == 0:
        st.subheader('Оценок качества моделей нет')
    else:
        st.subheader('Оценки качества моделей:')
        # Построим графики метрик качества модели
        for metric in clas_metrics:
            metricLogger1.plot('Метрика: ' + metric, metric, figsize=(7, 6))
```

Интерфейс веб приложения:





	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlo
0	white	7	0.2700	0.3600	20.7000	(
1	white	6.3000	0.3000	0.3400	1.6000	(
2	white	8.1000	0.2800	0.4000	6.9000	(
3	white	7.2000	0.2300	0.3200	8.5000	(
4	white	7.2000	0.2300	0.3200	8.5000	(
5	white	8.1000	0.2800	0.4000	6.9000	(
6	white	6.2000	0.3200	0.1600	7	(
7	white	7	0.2700	0.3600	20.7000	(
8	white	6.3000	0.3000	0.3400	1.6000	(
9	white	8.1000	0.2200	0.4300	1.5000	(
10	white	8.1000	0.2700	0.4100	1.4500	(



Управление:

- [ ] Информация по набору данных
- [ ] Показать корреляционную матрицу
- [x] Модели машинного обучения
- [ ] Подбор гиперпараметров
- [ ] Модель К-ближайших соседей
- [ ] Оценки качества моделей

Модели машинного обучения

Выберите модели

LogR × KNN_5 × SVC ×

Курсовой проект по дисциплине "Технологии машинного обучения"

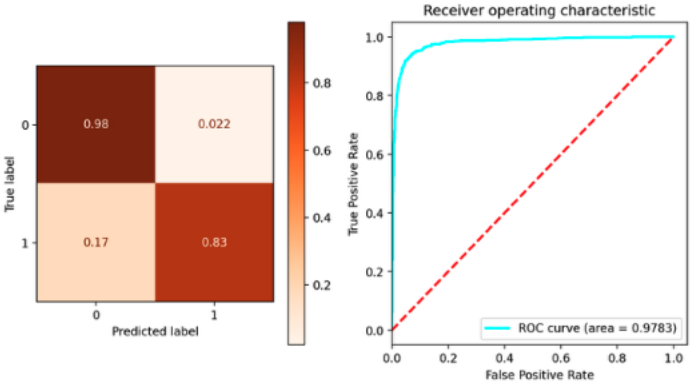
Веб-приложение с использованием фреймворка streamlit

Статус:

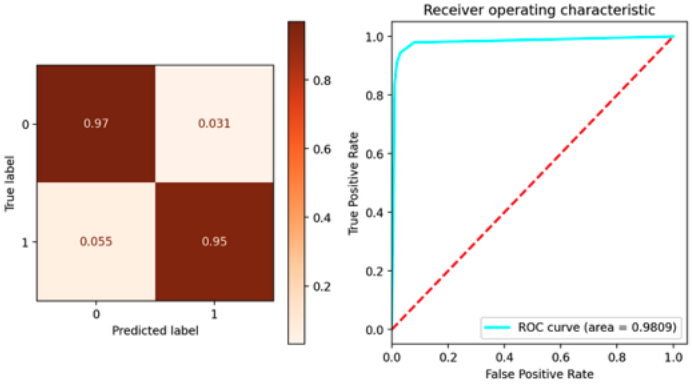
```
Данные загружены!
```

Модели:

LogR



KNN_5



Управление:

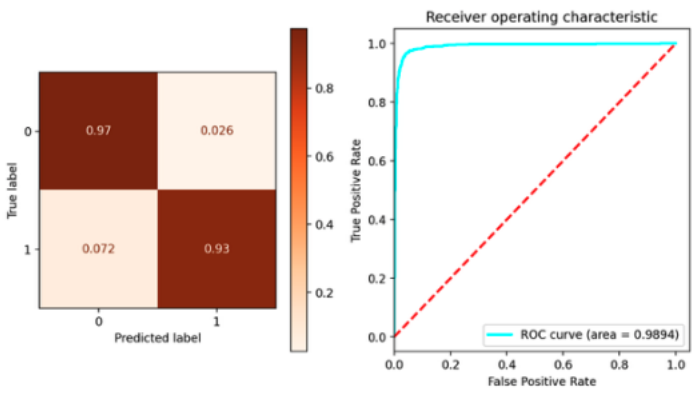
- [ ] Информация по набору данных
- [ ] Показать корреляционную матрицу
- [x] Модели машинного обучения
- [ ] Подбор гиперпараметров
- [ ] Модель К-ближайших соседей
- [ ] Оценки качества моделей

Модели машинного обучения

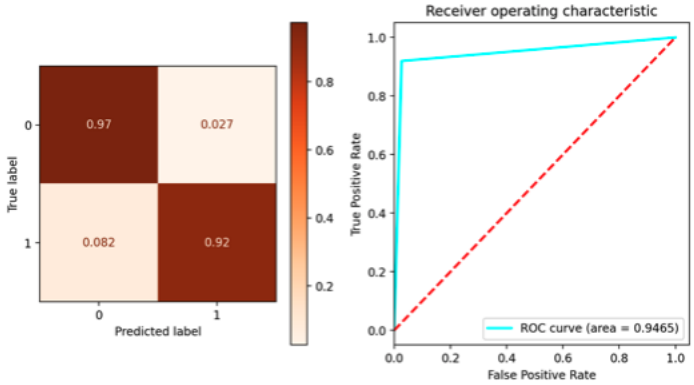
Выберите модели

LogR × KNN_5 × SVC × Tree × RF ×

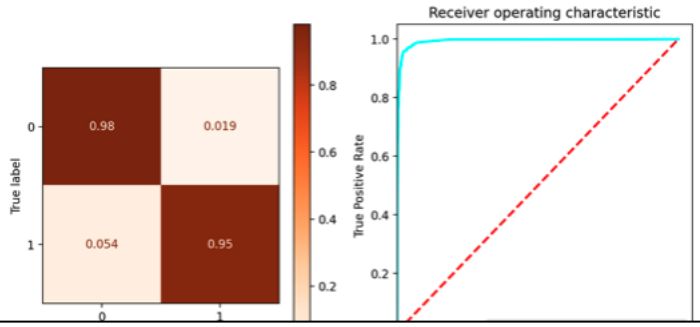
SVC



Tree



RF



- [x] Подбор гиперпараметров
- [ ] Модель К-ближайших соседей
- [ ] Оценки качества моделей

Параметры:

Выберите метод подбора:

GridSearch ×

Шаг для соседей: 17

1 — 100

Макс кол-во соседей: 500

50 — 1000

Подбор гиперпараметров:

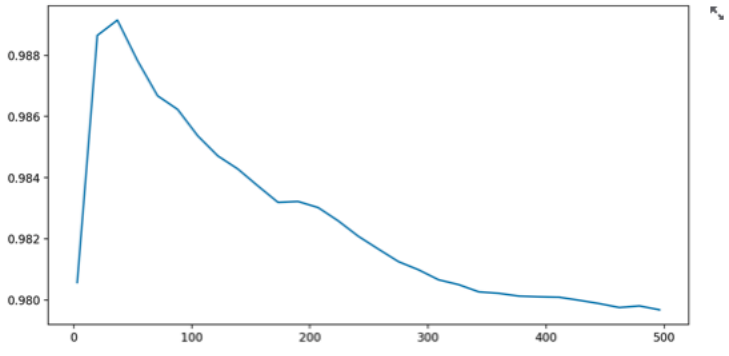
Количество строк в наборе данных - 6497

Возможные значения соседей - [3 20 37 54 71 88 105 122 139 156 173 190 207 224 241 258 275 292 309 326 343 360 377 394 411 428 445 462 479 496]

Оценка качества модели

Grid Search

Лучшее значение параметров - {'n_neighbors': 37}



Управление:

- [ ] Информация по набору данных
- [ ] Показать корреляционную матрицу
- [ ] Модели машинного обучения
- [x] Подбор гиперпараметров
- [ ] Модель К-ближайших соседей
- [ ] Оценки качества моделей

Параметры:

Выберите метод подбора:

RandomSearch ×
GridSearch ×

Шаг для соседей:

21

1 100

Макс кол-во соседей:

190

50 1000

Подбор гиперпараметров:

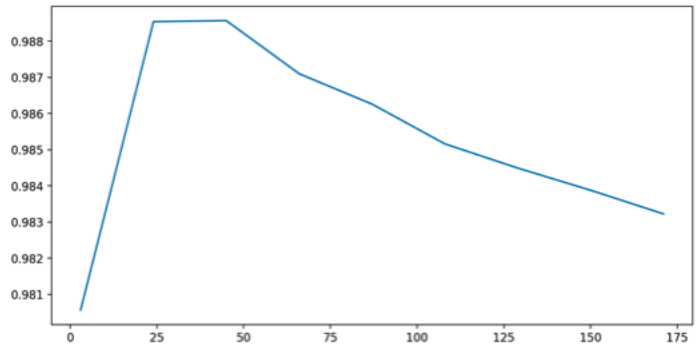
Количество строк в наборе данных - 6497

Возможные значения соседей - [3 24 45 66 87 108 129 150 171]

Оценка качества модели

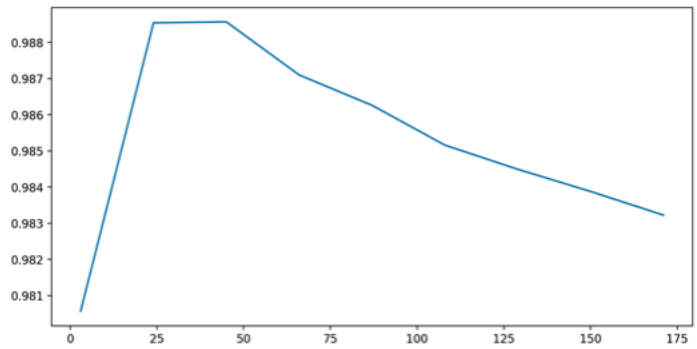
Grid Search

Лучшее значение параметров - {'n_neighbors': 45}



Random Search

Лучшее значение параметров - {'n_neighbors': 45}



Управление:

- [ ] Информация по набору данных
- [ ] Показать корреляционную матрицу
- [ ] Модели машинного обучения
- [ ] Подбор гиперпараметров
- [x] Модель К-ближайших соседей
- [ ] Оценки качества моделей

Модель К-ближайших соседей:

К-ближайших соседей:

37

1 100

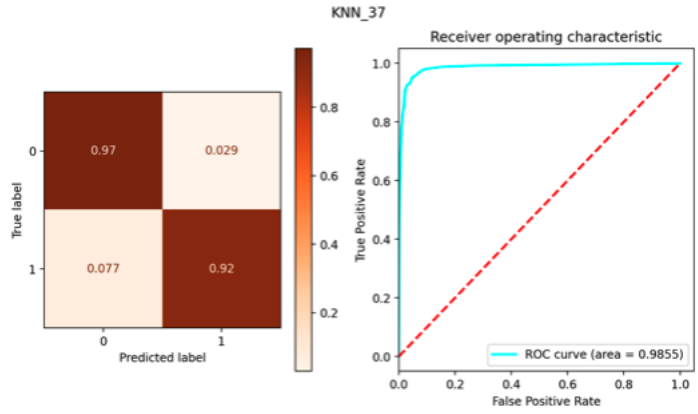
Курсовой проект по дисциплине "Технологии машинного обучения"

Веб-приложение с использованием фреймворка streamlit

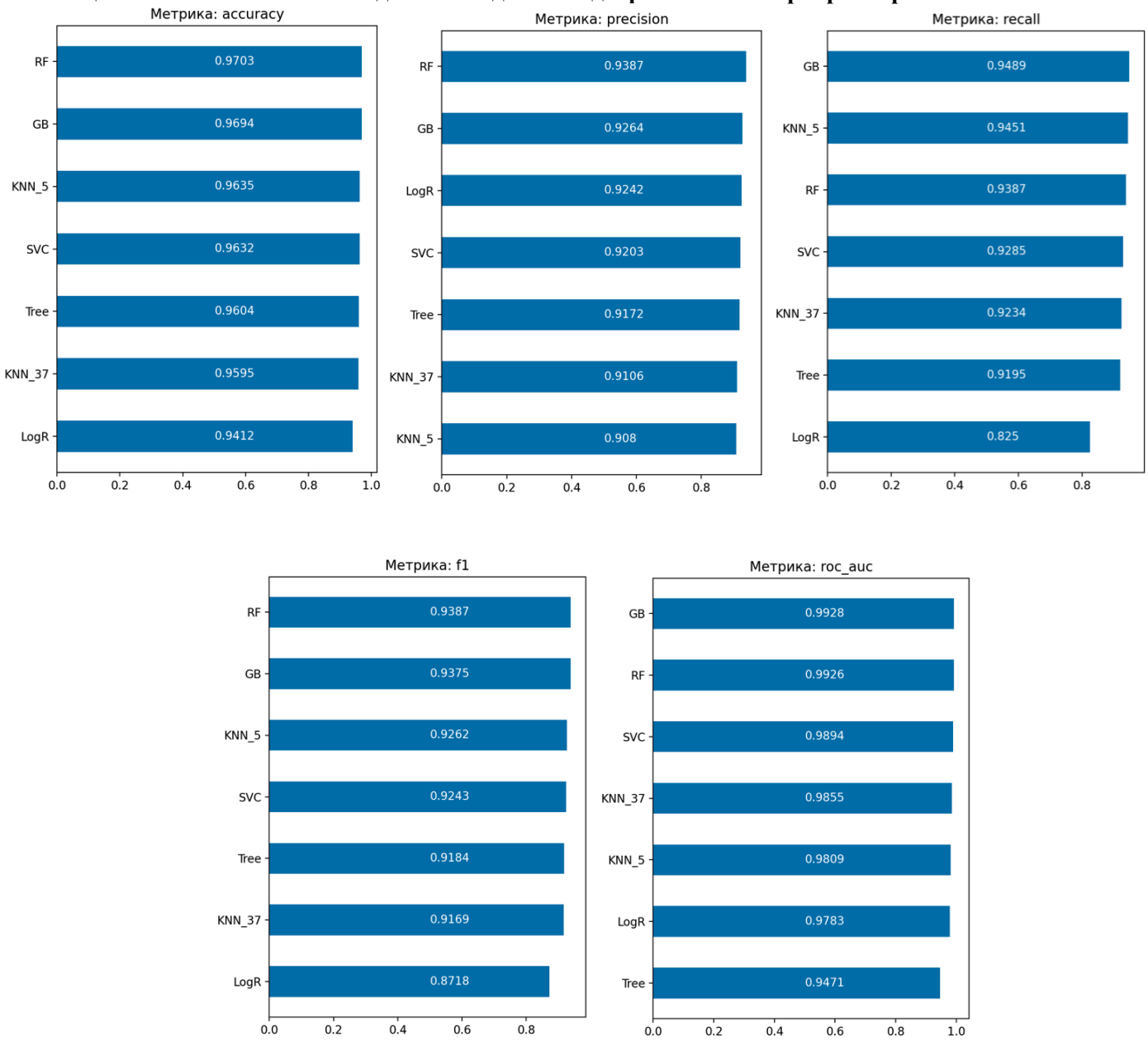
Статус:

`Данные загружены!`

Модель К-ближайших соседей



Оценка качества baseline моделей и модели с подобранными гиперпараметрами:



Выводы:

Модель, построенная по подобранному параметру, оказалось лучше baseline модели по оценкам двух метрик: precision и roc_auc. Лучшими по оценкам трех метрик стала модель Случайного леса, и по оценкам двух других модель Градиентного бустинга.