**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 комплексной задачи машинного обучения***

| | | | |
|---|---|---|---|
| Студент ИУ5Ц-83Б | | | **Донченко М.А.** |
| (Группа) | | (Подпись, дата) | (Фамилия И.О.) |
| Руководитель курсовой работы | | | **Гапанюк Ю.Е.** |
| | | (Подпись, дата) | (Фамилия И.О.) |
| Консультант | | | |
| | | (Подпись, дата) | (Фамилия И.О.) |

*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 курсовой работы

по дисциплине Технологии машинного обучения

Студент группы ИУ5Ц-83Б

Донченко Мария Андреевна
(Фамилия, имя, отчество)

Тема курсовой работы: решение комплексной задачи машинного обучения

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
учебная
Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения работы: 25% к ____нед., 50% к ____нед., 75% к ___нед., 100% к ___нед.

***Задание*** решение задачи машинного обучения на основе материалов дисциплины.

***Оформление курсовой работы:***

Расчетно-пояснительная записка на 19 листах формата А4.

Дата выдачи задания «____»_____20___г.

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

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

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

# Курсовая работа

## по дисциплине "Технологии машинного обучения"

### Задание:

1. Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.
2. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.
3. Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
4. Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения. В зависимости от набора данных, порядок выполнения пунктов 2, 3, 4 может быть изменен.
5. Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.
6. Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.
7. Формирование обучающей и тестовой выборок на основе исходного набора данных.
8. 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.
9. Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.
10. Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.

Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется построение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

```
import plotly.figure_factory as ff
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
StandardScaler, Normalizer
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error,
median_absolute_error, r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn import tree
import re

def load_data():
    '''
    Загрузка данных
    '''
    data = pd.read_csv('data/whitewines.csv')
    return data


@st.cache
def preprocess_data(data_in):
    '''
    Масштабирование признаков, функция возвращает X и y для кросс-валидации
    '''
    data_out = data_in.copy()
    # Числовые колонки для масштабирования
    scale_cols = ['density', 'total sulfur dioxide']
    new_cols = []
    sc1 = MinMaxScaler()
    sc1_data = sc1.fit_transform(data_out[scale_cols])
    for i in range(len(scale_cols)):
        col = scale_cols[i]
        new_col_name = col + '_scaled'
        new_cols.append(new_col_name)
        data_out[new_col_name] = sc1_data[:, i]
    X = data_out[new_cols]
    Y = data_out['residual sugar'].astype(int)
    # Чтобы в тесте получилось низкое качество используем только 0,5% данных
для обучения
    X_train, X_test, y_train, y_test = train_test_split(X, Y, train_size=0.8,
test_size=0.2, random_state=1)
    return X_train, X_test, y_train, y_test, X, Y


data = load_data()



st.sidebar.header('Случайный лес')
n_estimators_1 = st.sidebar.slider('Количество фолдов:', min_value=3,
max_value=10, value=3, step=1)

st.sidebar.header('Градиентный бустинг')
n_estimators_2 = st.sidebar.slider('Количество:', min_value=3, max_value=10,
value=3, step=1)
random_state_2 = st.sidebar.slider('random_state:', min_value=3,
```

```
max_value=15, value=3, step=1)

# Первые пять строк датасета
st.subheader('Первые 5 значений')
st.write(data.head())

st.subheader('Размер датасета')
st.write(data.shape)

st.subheader('Количество нулевых элементов')
st.write(data.isnull().sum())

st.write(data['volatile acidity'].value_counts())

st.subheader('Колонки и их типы данных')
st.write(data.dtypes)

st.subheader('Статистические данные')
st.write(data.describe())

fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(x=data['alcohol'], y=data['quality'])
plt.xlabel("alcohol")
plt.ylabel("quality")
st.pyplot(fig)

f1, ax = plt.subplots()
sns.boxplot(x=data['quality'])
st.pyplot(f1)

f, ax = plt.subplots()
sns.violinplot(x=data['quality'])
st.pyplot(f)

st.subheader('Масштабирование данных')
f, ax = plt.subplots()
plt.hist(data['quality'], 50)
plt.show()
st.pyplot(f)

st.subheader('Показать корреляционную матрицу')
fig1, ax = plt.subplots(figsize=(10, 5))
sns.heatmap(data.corr(), annot=True, fmt='.2f')
st.pyplot(fig1)

X_train, X_test, Y_train, Y_test, X, Y = preprocess_data(data)
forest_1 = RandomForestRegressor(n_estimators=n_estimators_1, oob_score=True,
random_state=10)
forest_1.fit(X, Y)
Y_predict = forest_1.predict(X_test)


st.subheader('RandomForestRegressor')
st.subheader('Средняя абсолютная ошибка:')
st.write(mean_absolute_error(Y_test, Y_predict))
st.subheader('Средняя квадратичная ошибка:')
st.write(mean_squared_error(Y_test, Y_predict))
st.subheader('Median absolute error:')
st.write(median_absolute_error(Y_test, Y_predict))
st.subheader('Коэффициент детерминации:')
st.write(r2_score(Y_test, Y_predict))

fig1 = plt.figure(figsize=(7, 5))
ax = plt.scatter(X_test['density_scaled'], Y_test, marker='o',
```

```
label='Тестовая выборка')
plt.scatter(X_test['density_scaled'], Y_predict, marker='.',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('density_scaled')
plt.ylabel('price')
plt.plot(n_estimators_1)
st.pyplot(fig1)

st.subheader('Нахождение лучшего случайного леса')

params2 = {
    'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50, 75, 100],
}

grid_2 = GridSearchCV(estimator=RandomForestRegressor(oob_score=True,
random_state=10),
                      param_grid=params2,
                      scoring='neg_mean_squared_error',
                      cv=3,
                      n_jobs=-1)
grid_2.fit(X, Y)

st.write(grid_2.best_params_)

forest_3 = RandomForestRegressor(n_estimators=75, oob_score=True,
random_state=5)
forest_3.fit(X, Y)
Y_predict3 = forest_3.predict(X_test)
st.subheader('Средняя абсолютная ошибка:')
st.write(mean_absolute_error(Y_test, Y_predict3))
st.subheader('Средняя квадратичная ошибка:')
st.write(mean_squared_error(Y_test, Y_predict3))
st.subheader('Median absolute error:')
st.write(median_absolute_error(Y_test, Y_predict3))
st.subheader('Коэффициент детерминации:')
st.write(r2_score(Y_test, Y_predict3))

fig1 = plt.figure(figsize=(7, 5))
ax = plt.scatter(X_test['density_scaled'], Y_test, marker='o',
label='Тестовая выборка')
plt.scatter(X_test['density_scaled'], Y_predict3, marker='.',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('density_scaled')
plt.ylabel('price')
plt.plot(n_estimators_1)
st.pyplot(fig1)

st.subheader('Градиентный бустинг')

grad = GradientBoostingRegressor(n_estimators=n_estimators_2,
random_state=random_state_2)
grad.fit(X_train, Y_train)
Y_grad_pred = grad.predict(X_test)
st.subheader('Средняя абсолютная ошибка:')
st.write(mean_absolute_error(Y_test, Y_grad_pred))
st.subheader('Средняя квадратичная ошибка:')
st.write(mean_squared_error(Y_test, Y_grad_pred))
st.subheader('Median absolute error:')
st.write(median_absolute_error(Y_test, Y_grad_pred))
st.subheader('Коэффициент детерминации:')
st.write(r2_score(Y_test, Y_grad_pred))
```

```
fig2 = plt.figure(figsize=(7, 5))
ax = plt.scatter(X_test['density_scaled'], Y_test, marker='o',
label='Тестовая выборка')
plt.scatter(X_test['density_scaled'], Y_grad_pred, marker='.',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('density_scaled')
plt.ylabel('price')
plt.plot(random_state_2)
st.pyplot(fig2)

st.subheader('Нахождение лучшего////')

params = {
    'n_estimators': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 50, 75, 100],
    'max_features': [0.2, 0.3, 0.4, 0.6, 0.8, 0.9, 1.0],
    'min_samples_leaf': [0.01, 0.04, 0.06, 0.08, 0.1]
}

grid_gr = GridSearchCV(estimator=GradientBoostingRegressor(random_state=10),
                       param_grid=params,
                       scoring='neg_mean_squared_error',
                       cv=3,
                       n_jobs=-1)
grid_gr.fit(X_train, Y_train)
st.write(grid_gr.best_params_)

grad1 = GradientBoostingRegressor(n_estimators=50, max_features=1,
min_samples_leaf=0.01, random_state=10)
grad1.fit(X_train, Y_train)
Y_grad_pred1 = grad1.predict(X_test)

st.subheader('Средняя абсолютная ошибка:')
st.write(mean_absolute_error(Y_test, Y_grad_pred1))
st.subheader('Средняя квадратичная ошибка:')
st.write(mean_squared_error(Y_test, Y_grad_pred1))
st.subheader('Median absolute error:')
st.write(median_absolute_error(Y_test, Y_grad_pred1))
st.subheader('Коэффициент детерминации:')
st.write(r2_score(Y_test, Y_grad_pred1))

fig1 = plt.figure(figsize=(7, 5))
ax = plt.scatter(X_test['density_scaled'], Y_test, marker='o',
label='Тестовая выборка')
plt.scatter(X_test['density_scaled'], Y_grad_pred1, marker='.',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('density_scaled')
plt.ylabel('price')
plt.plot(n_estimators_1)
st.pyplot(fig1)

st.subheader('Построение линейной регрессии')

Lin_Reg = LinearRegression().fit(X_train, Y_train)

lr_y_pred = Lin_Reg.predict(X_test)

fig3 = plt.figure(figsize=(7, 5))
plt.scatter(X_test['density_scaled'], Y_test, marker='s', label='Тестовая
выборка')
plt.scatter(X_test['density_scaled'], lr_y_pred, marker='o',
label='Предсказанные данные')
plt.legend(loc='lower right')
```

```
plt.xlabel('density_scaled')
plt.ylabel('price')
plt.show()
st.pyplot(fig3)

st.subheader('Tree')

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, Y_train)

lr_y_pred = Lin_Reg.predict(X_test)

fig5 = plt.figure(figsize=(7, 5))
plt.scatter(X_test['density_scaled'], Y_test, marker='s', label='Тестовая
выборка')
plt.scatter(X_test['density_scaled'], lr_y_pred, marker='o',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('density_scaled')
plt.ylabel('price')
plt.show()
st.pyplot(fig5)

st.subheader('Модель ближайших соседей для произвольного гиперпараметра K')

Regressor_5NN = KNeighborsRegressor(n_neighbors = 5)
Regressor_5NN.fit(X_train, Y_train)

lr_y_pred = Regressor_5NN.predict(X_test)

fig6 = plt.figure(figsize=(7, 5))
plt.scatter(X_test['density_scaled'], Y_test, marker='s', label='Тестовая
выборка')
plt.scatter(X_test['density_scaled'], lr_y_pred, marker='o',
label='Предсказанные данные')
plt.legend(loc='lower right')
plt.xlabel('density_scaled')
plt.ylabel('price')
plt.show()
st.pyplot(fig6)
```

## Случайный лес

Количество фолдов:

3

3 10

## Градиентный бустинг

Количество:

3

3 10

random_state:

3

3 15

### Первые 5 значений

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | fr |
|---|---|---|---|---|---|---|
| 0 | 7 | 0.2700 | 0.3600 | 20.7000 | 0.0450 | |
| 1 | 6.3000 | 0.3000 | 0.3400 | 1.6000 | 0.0490 | |
| 2 | 8.1000 | 0.2800 | 0.4000 | 6.9000 | 0.0500 | |
| 3 | 7.2000 | 0.2300 | 0.3200 | 8.5000 | 0.0580 | |
| 4 | 7.2000 | 0.2300 | 0.3200 | 8.5000 | 0.0580 | |

### Размер датасета

```
(4898, 12)
```

## Количество нулевых элементов

| | 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 |

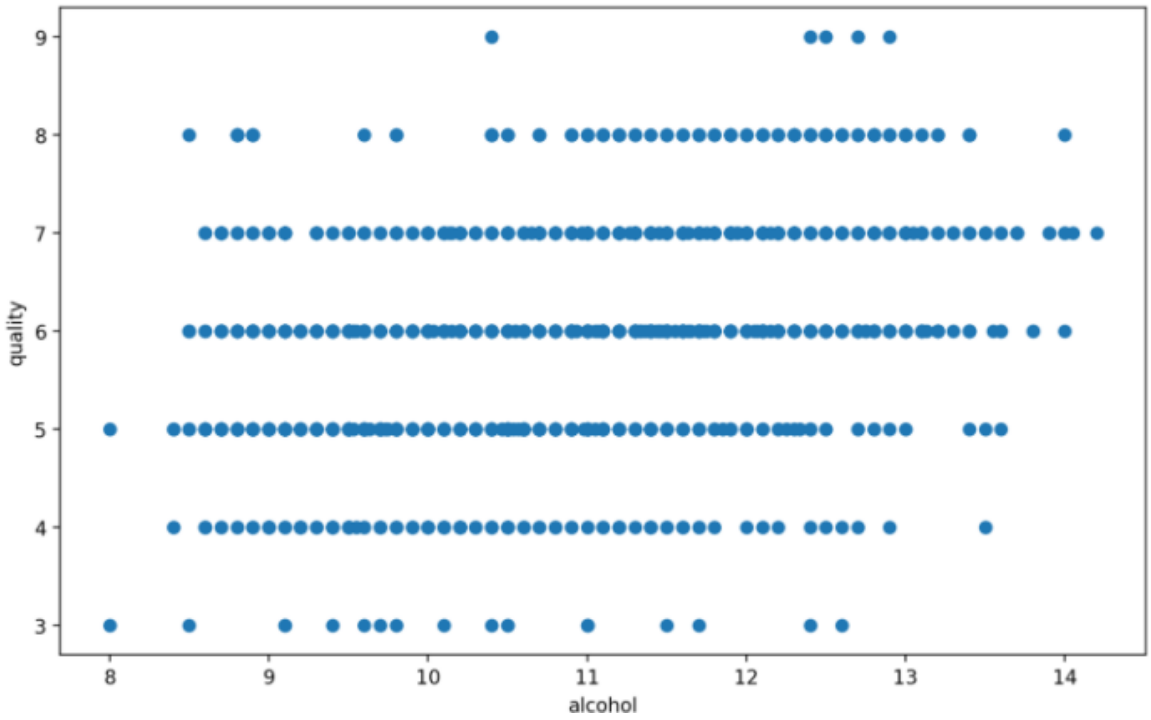
| | volatile acidity |
|---|---|
| 0.2800 | 263 |
| 0.2400 | 253 |
| 0.2600 | 240 |
| 0.2500 | 231 |
| 0.2200 | 229 |
| 0.2700 | 218 |
| 0.2300 | 216 |
| 0.2000 | 214 |
| 0.3000 | 198 |
| 0.2100 | 191 |
| 0.3200 | 182 |

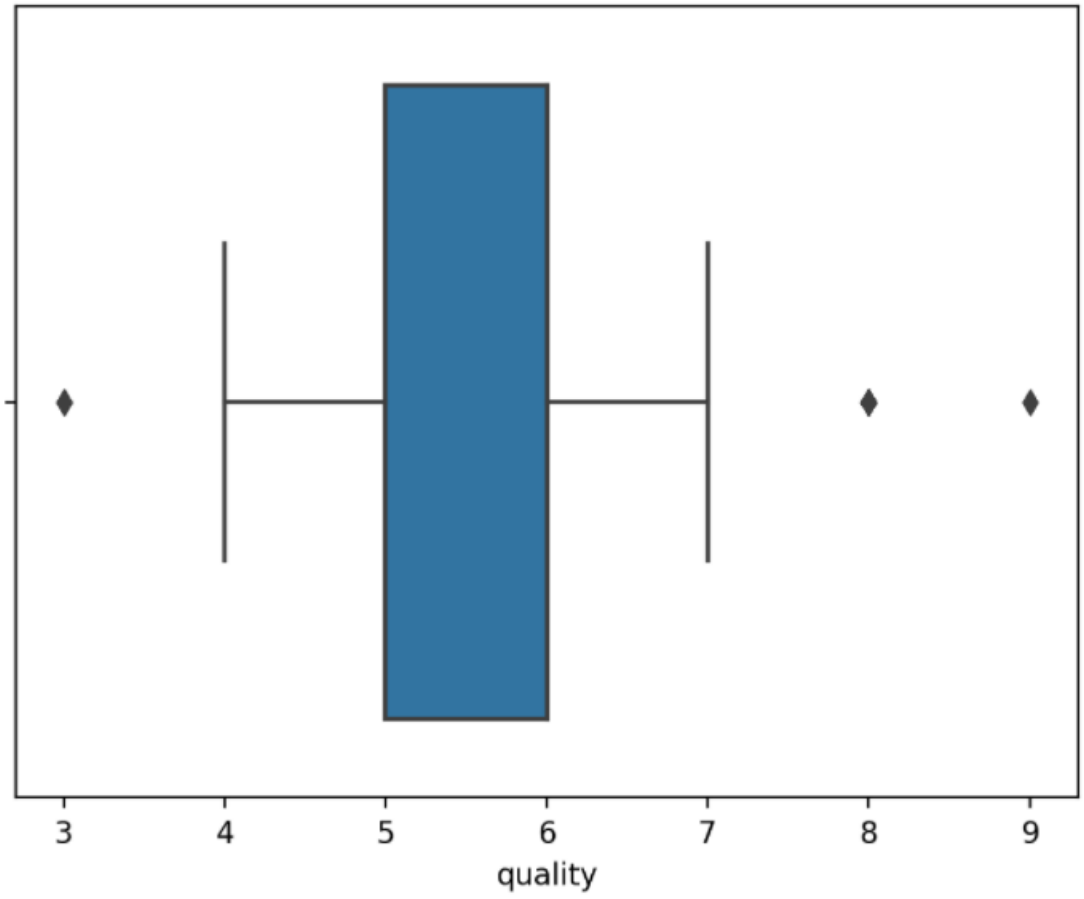
## Колонки и их типы данных

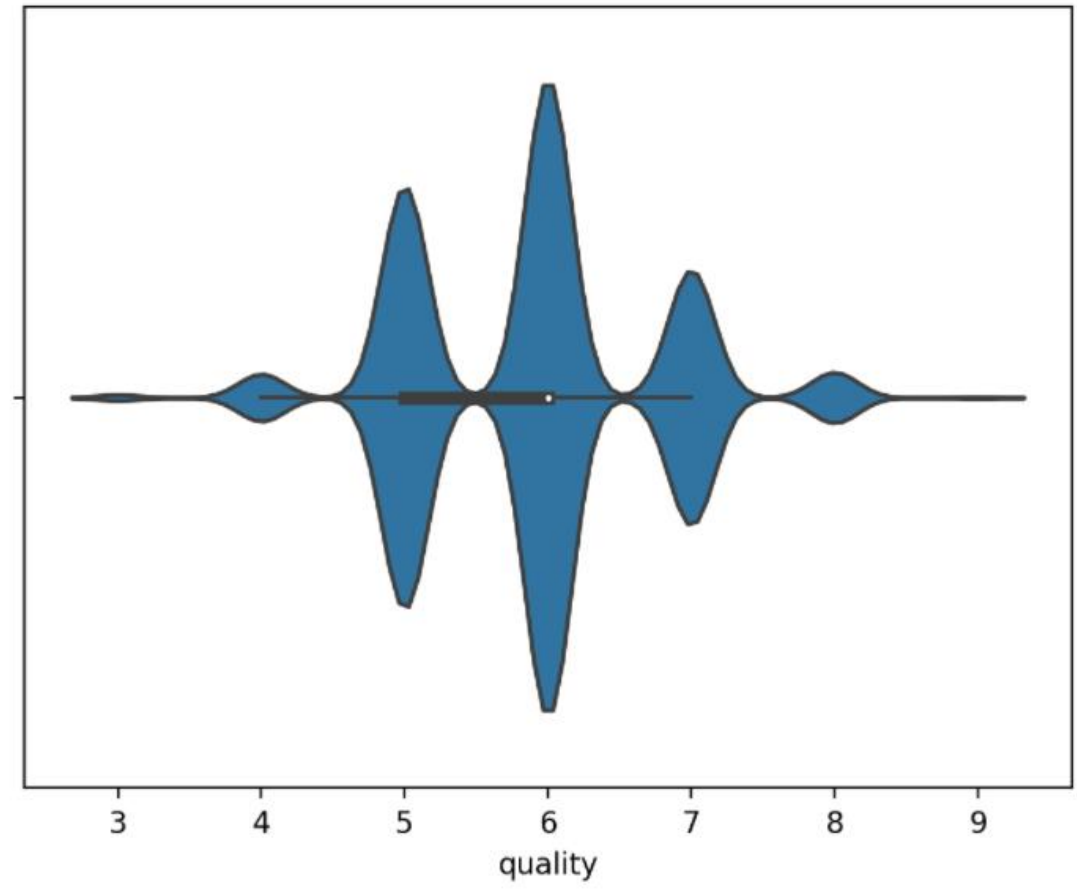
| | 0 |
|---|---|
| fixed acidity | float64 |
| volatile acidity | float64 |
| citric acid | float64 |
| residual sugar | float64 |
| chlorides | float64 |
| free sulfur dioxide | float64 |
| total sulfur dioxide | float64 |
| density | float64 |
| pH | float64 |
| sulphates | float64 |
| alcohol | float64 |

## Статистические данные

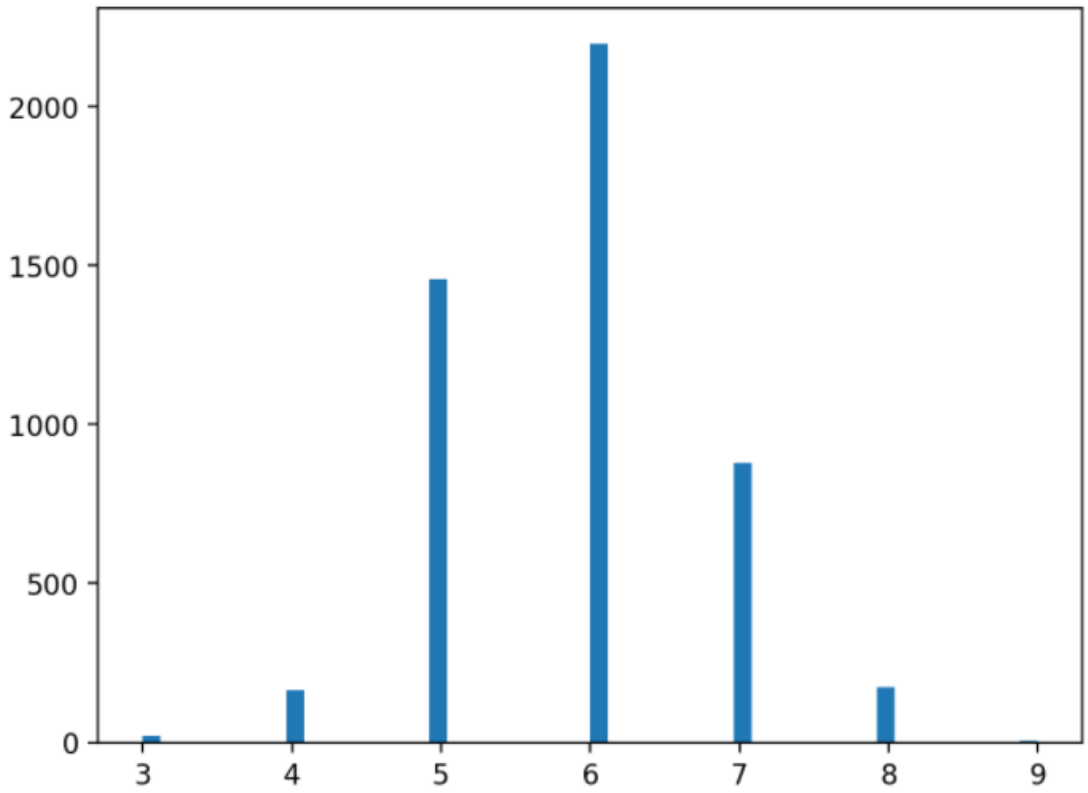
| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides |
|---|---|---|---|---|---|
| count | 4898 | 4898 | 4898 | 4898 | 4898 |
| mean | 6.8548 | 0.2782 | 0.3342 | 6.3914 | 0.0458 |
| std | 0.8439 | 0.1008 | 0.1210 | 5.0721 | 0.0218 |
| min | 3.8000 | 0.0800 | 0 | 0.6000 | 0.0090 |
| 25% | 6.3000 | 0.2100 | 0.2700 | 1.7000 | 0.0360 |
| 50% | 6.8000 | 0.2600 | 0.3200 | 5.2000 | 0.0430 |
| 75% | 7.3000 | 0.3200 | 0.3900 | 9.9000 | 0.0500 |
| max | 14.2000 | 1.1000 | 1.6600 | 65.8000 | 0.3460 |




3
4
5
6
7
8
9
quality


3
4
5
6
7
8
9
quality

## Масштабирование данных




## Показать корреляционную матрицу

RandomForestRegressor

### Средняя абсолютная ошибка:

```
0.6350566893424037
```

### Средняя квадратичная ошибка:

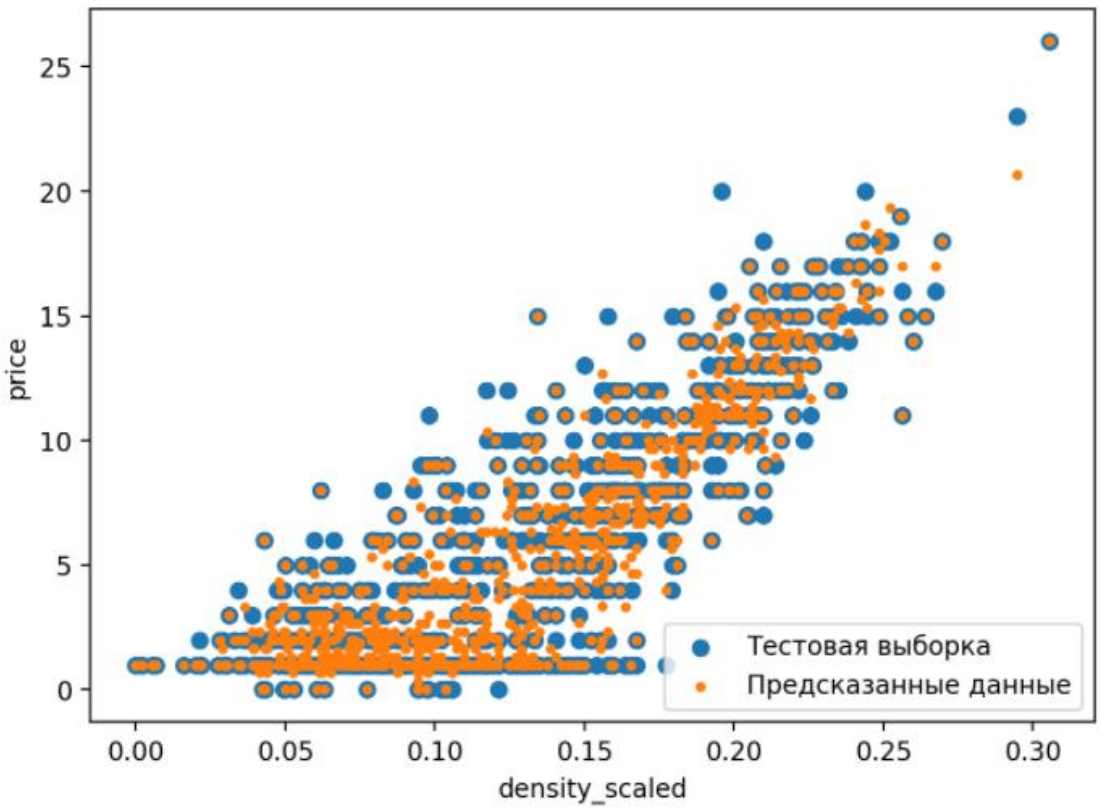
```
1.606127461358693
```

### Median absolute error:

```
0.0
```

### Коэффициент детерминации:

```
0.93635702085307
```



### Нахождение лучшего случайного леса


```
{
  "n_estimators" : 75
}
```


### Средняя абсолютная ошибка:

`0.6087955330761453`

### Средняя квадратичная ошибка:

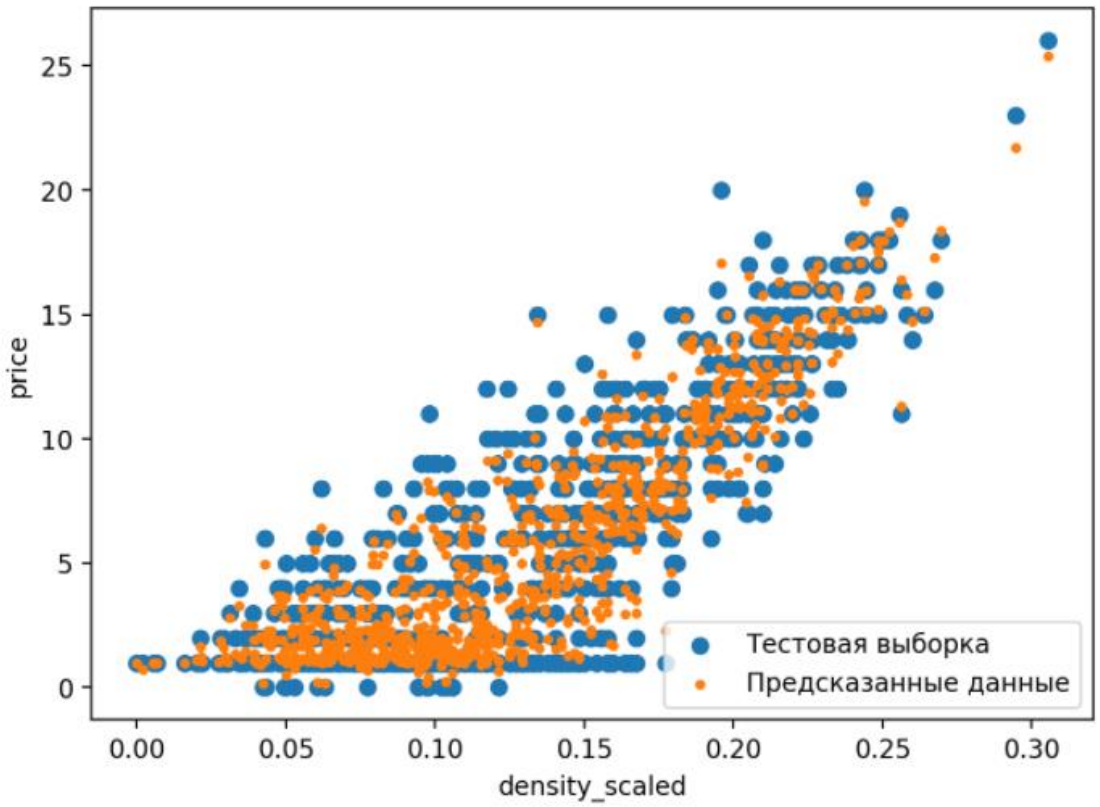
`0.9310949482596712`

### Median absolute error:

`0.3600000000000003`

### Коэффициент детерминации:

`0.9631052592016741`



## Градиентный бустинг

### Средняя абсолютная ошибка:

```
3.439577455813961
```

### Средняя квадратичная ошибка:

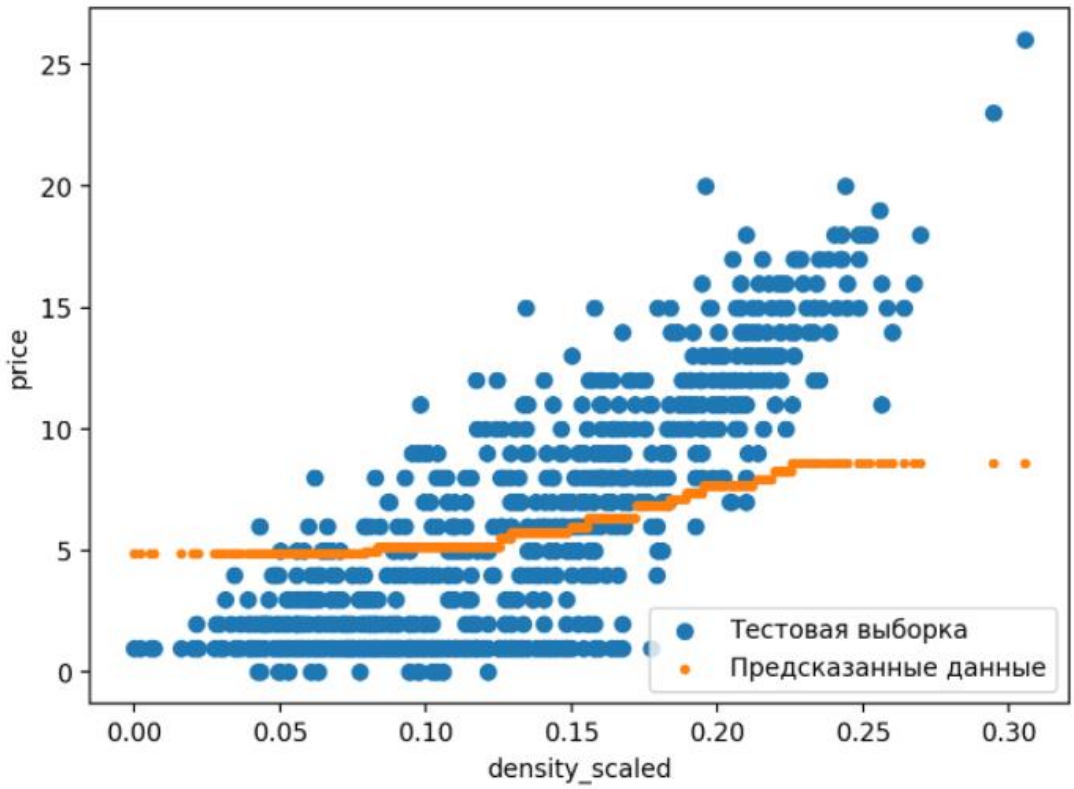
```
16.36672227850483
```

### Median absolute error:

```
3.858982186947001
```

### Коэффициент детерминации:

```
0.3514668108636161
```



## Нахождение лучшего////


```
{
  "max_features" : 1
  "min_samples_leaf" : 0.01
  "n_estimators" : 50
}
```


## Средняя абсолютная ошибка:

1.8667782240435924

## Средняя квадратичная ошибка:

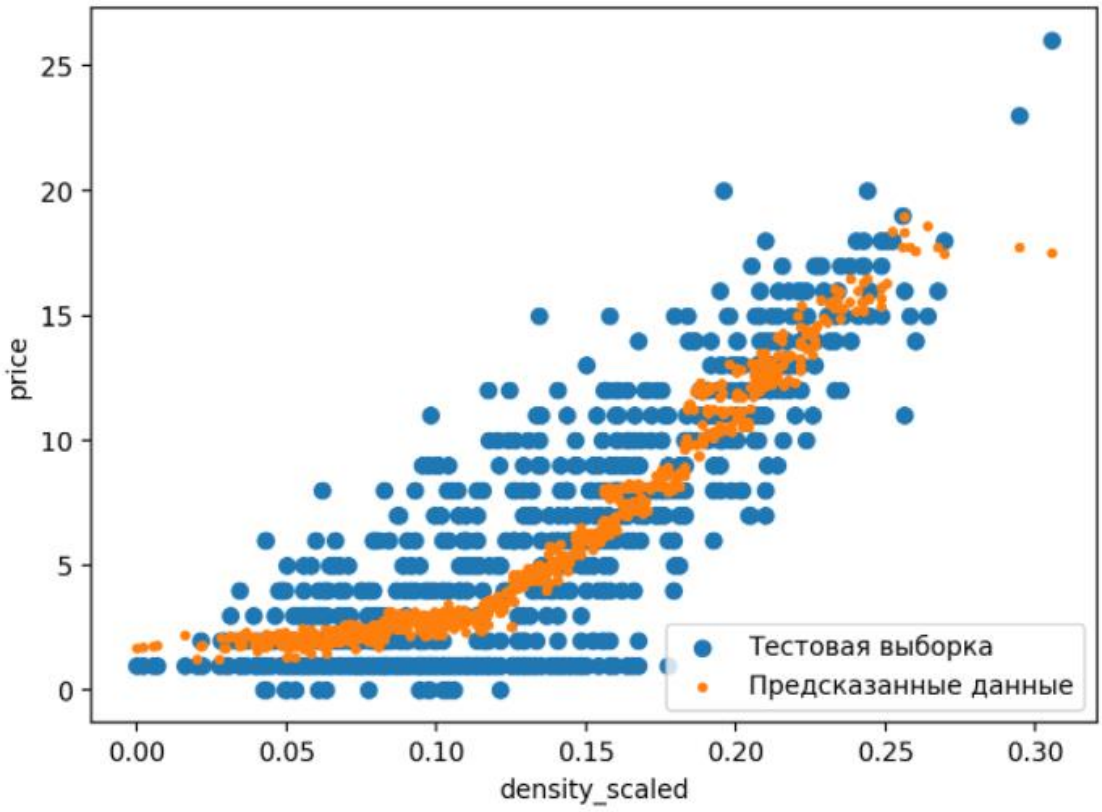
5.9103563475457905

## Median absolute error:

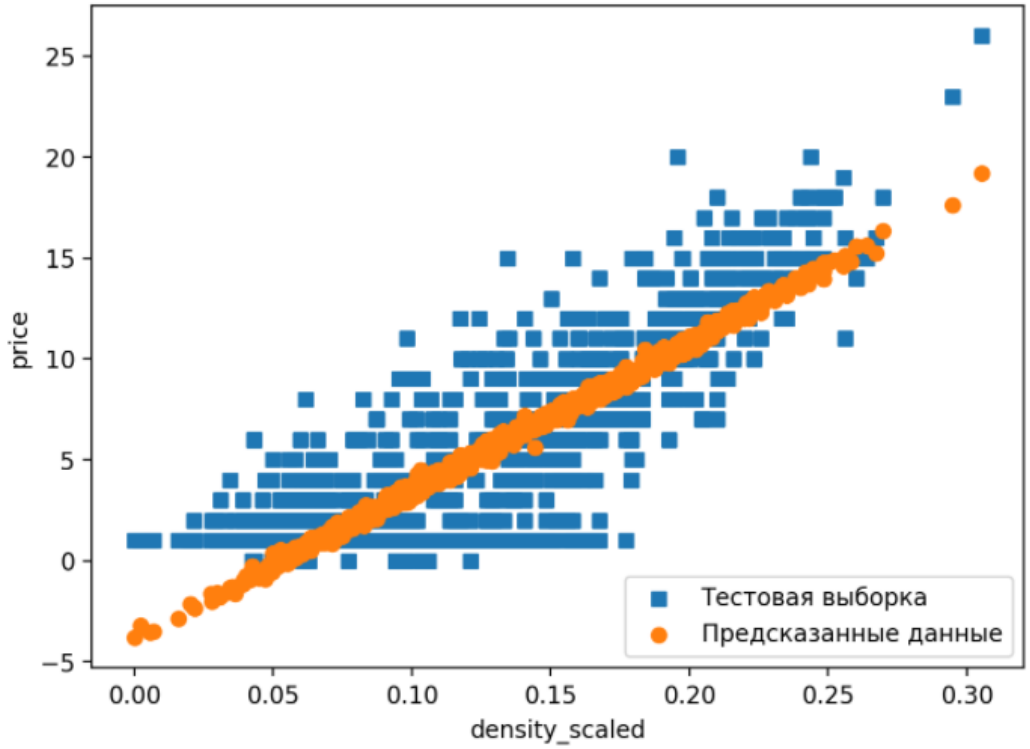
1.5414564804505195

## Коэффициент детерминации:

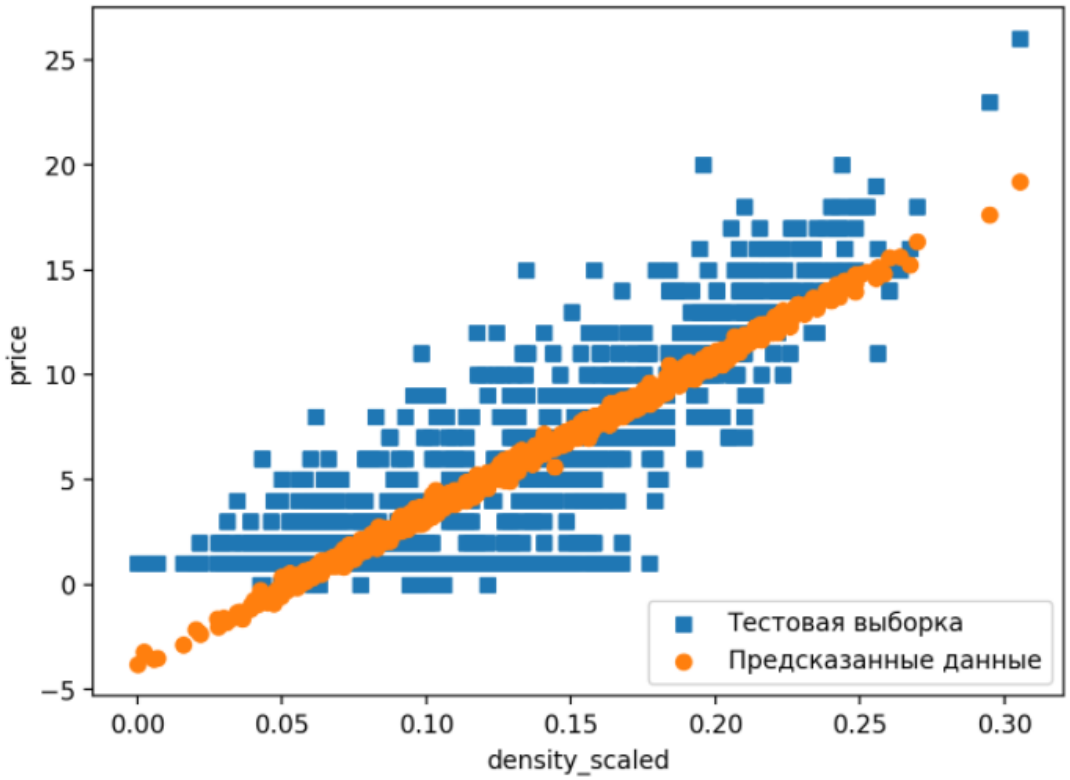
0.7658014729045364



Построение линейной регрессии




Tree



Модель ближайших соседей для произвольного гиперпараметра K