Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Домашнее задание по
дисциплине
«Методы машинного обучения»
на тему
«Решение задачи классификации с помощью трёх моделей»

Выполнл:
студент группы ИУ5-23М
Богомолов Д.Н.

Москва — 2020 г.

Богомолов Дмитрий ИУ5-23М

Домашнее задание по курсу Методы машинного обучен

Домашнее задание включает выполнение следующих шагов:

1. Поиск и выбор набора данных для построения моделей машинного обучения. На осно
должен построить модели машинного обучения для решения или задачи классифика
2. Проведение разведочного анализа данных. Построение графиков, необходимых для
заполнение пропусков в данных.
3. Выбор признаков, подходящих для построения моделей. Кодирование категориальны
Формирование вспомогательных признаков, улучшающих качество моделей.
4. Проведение корреляционного анализа данных. Формирование промежуточных выво
машинного обучения. В зависимости от набора данных, порядок выполнения пунктов
5. Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не м
6. Выбор наиболее подходящих моделей для решения задачи классификации или регре
трех моделей, хотя бы одна из которых должна быть ансамблевой.
7. Формирование обучающей и тестовой выборок на основе исходного набора данных.
8. Построение базового решения (baseline) для выбранных моделей без подбора гиперп
моделей на основе обучающей выборки и оценка качества моделей на основе тестов
9. Подбор гиперпараметров для выбранных моделей. Рекомендуется подбирать не боле
использовать методы кросс-валидации. В зависимости от используемой библиотеки
использовать перебор параметров в цикле, или использовать другие методы.
10. Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравн
качеством baseline-моделей.
11. Формирование выводов о качестве построенных моделей на основе выбранных метр

Поиск и выбор набора данных для построения моделей машин выбранного набора данных студент должен построить модели решения или задачи классификации, или задачи регрессии.

Импорт библиотек

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from plotly.subplots import make_subplots
import plotly.graph_objects as go
```

```
import plotly.graph_objects as go
import plotly.express as pltl
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, Normalizer
from sklearn.model_selection import train_test_split
```

Датасет wine quality на сайте Kaggle

Сведения о датасете Эти два набора данных относятся к красному и белому вариантам по
[Cortez et al., 2009]. Из-за проблем конфиденциальности и логистики доступны только физи
сенсорные (выходные данные) переменные (например, нет данных о типах винограда, мар

Эти наборы данных можно рассматривать как задачи классификации или регрессии. Клас
(например, есть Мунк больше нормальных вин, чем отличных или плохих). Для обнаружени
вин можно было бы использовать более сложные алгоритмы обнаружения выбросов. Кро
переменные релевантны. Поэтому было бы интересно протестировать методы выбора фун

Два набора данных были объединены, и несколько значений были случайным образом уда

атрибутивная информация:

Входные переменные (на основе физико-химических тестов): 1-тип (белое/красное) 2-фикс
кислотность 4-лимонная кислота 5-остаточный сахар 6-хлориды 7-свободный диоксид сер
pH 11-сульфаты 12-алкоголь 12-качество (оценка от 0 до 10 баллов)

```
data1 = pd.read_csv('/content/drive/My Drive/MMO_Datasets/winequalityN.csv', sep = ',
data = pd.DataFrame(data = data1)
```

Выведем несколько строк

```
data.head()
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	t
0	white	7.0	0.27	0.36	20.7	0.045	45.0	
1	white	6.3	0.30	0.34	1.6	0.049	14.0	
2	white	8.1	0.28	0.40	6.9	0.050	30.0	
3	white	7.2	0.23	0.32	8.5	0.058	47.0	
4	white	7.2	0.23	0.32	8.5	0.058	47.0	

Проведение разведочного анализа данных. Построение график понимания структуры данных. Анализ и заполнение пропусков

```
data.dtypes
```

```
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
```

Видим, что поле тип представляет собой текст типа object, поэтому применим Label encode уникальное числовое значение. Т.е. белое вино - 1, красное - 0

```
le = LabelEncoder()
for col in data.columns:
  cat_enc_le = le.fit_transform(data[col])
  data[col] = cat_enc_le
  data[col].unique()
```

Поиск пустых значений в колонках

```
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
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
```

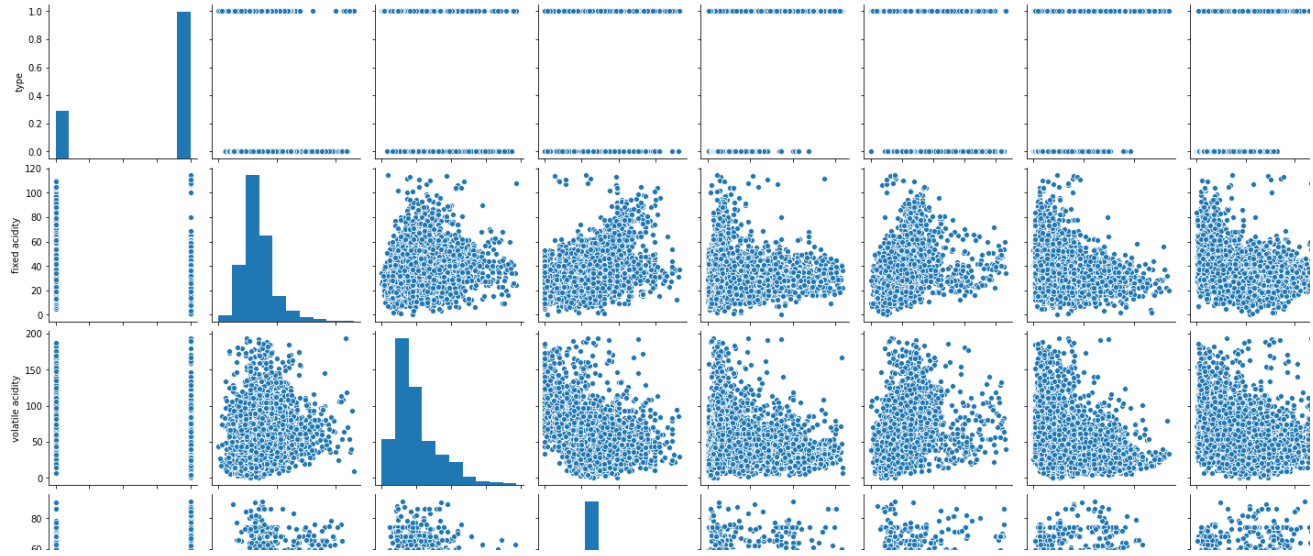
```
data.describe() #Описательные статистики
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.00000	649
mean	0.753886	33.276589	49.403109	31.861013	80.555795	44.03417	3
std	0.430779	13.988846	31.271424	14.386978	75.358225	27.56434	2
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	
25%	1.000000	24.000000	28.000000	25.000000	21.000000	27.00000	1

Построим парные диаграммы для понимания структуры данных

```
1  sns.pairplot(data)
```

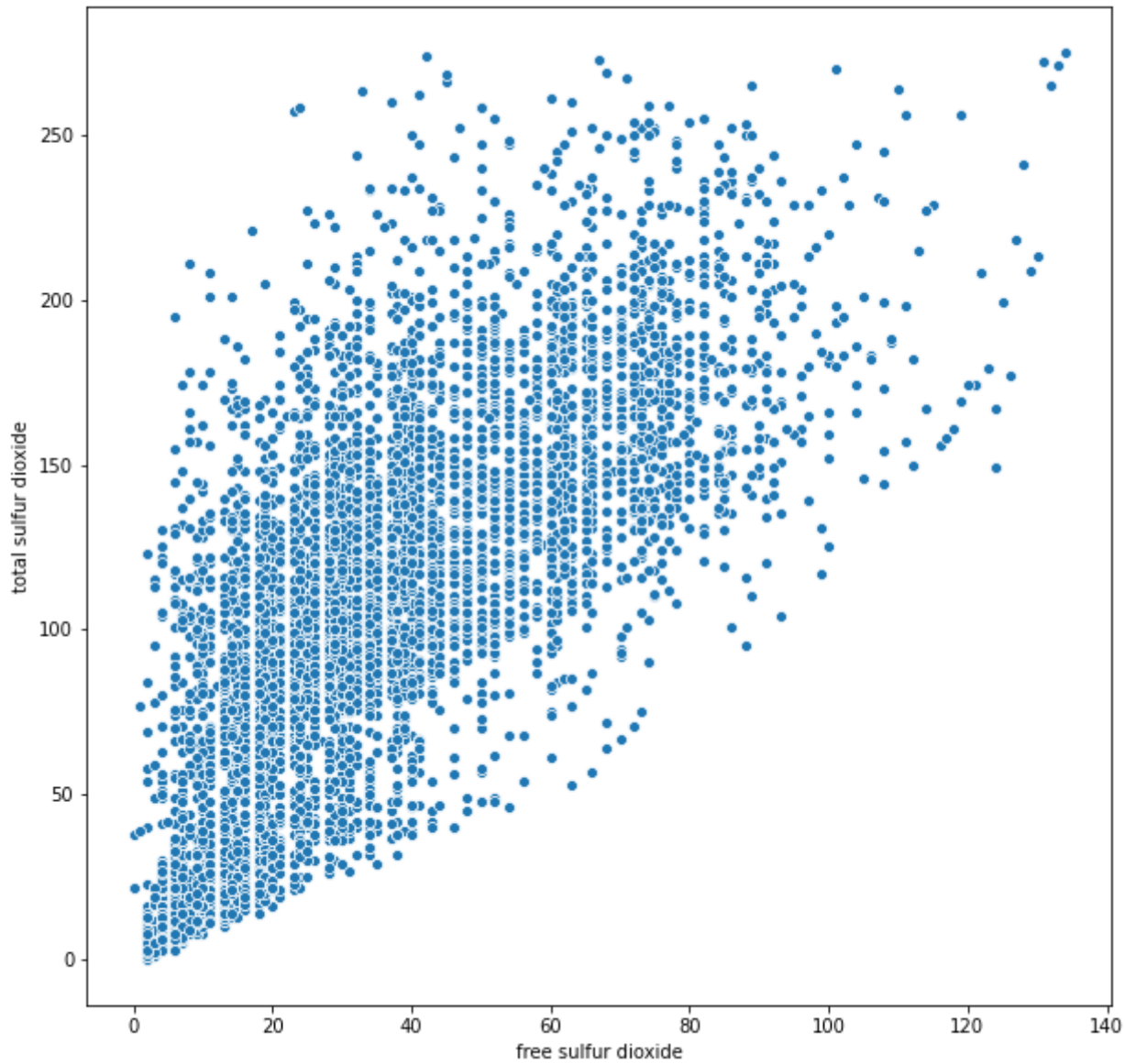
```
<seaborn.axisgrid.PairGrid at 0x7f08f46cfba8>
```



Находим почти линейную зависимость между значениями двух колонок с содержанием "в

```
1  fig, ax = plt.subplots(figsize=(10,10))
2  sns.scatterplot(ax=ax, x='free sulfur dioxide', y='total sulfur dioxide', data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f08f118ea58>
```



Проведение корреляционного анализа данных. Формирование возможности построения моделей машинного обучения. В зави порядок выполнения пунктов 2, 3, 4 может быть изменен.

Построим корреляционную матрицу

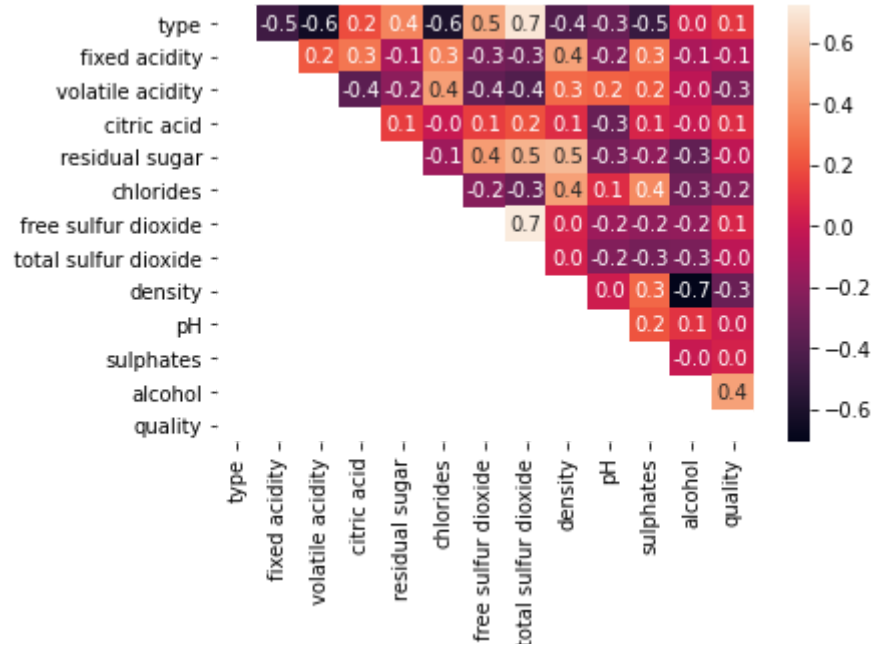
```
data.corr()
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	d
type	1.000000	-0.471746	-0.644529	0.187353	0.352260	-0.601578	0.469138	0.
fixed acidity	-0.471746	1.000000	0.221546	0.309189	-0.103445	0.345455	-0.270991	-0
volatile acidity	-0.644529	0.221546	1.000000	-0.369614	-0.201820	0.436801	-0.350387	-0.
citric acid	0.187353	0.309189	-0.369614	1.000000	0.142452	-0.000137	0.143170	0.
residual sugar	0.352260	-0.103445	-0.201820	0.142452	1.000000	-0.146735	0.429815	0.
chlorides	-0.601578	0.345455	0.436801	-0.000137	-0.146735	1.000000	-0.222113	-0.
free sulfur dioxide	0.469138	-0.270991	-0.350387	0.143170	0.429815	-0.222113	1.000000	0.
total sulfur dioxide	0.696524	-0.311061	-0.403837	0.196154	0.507789	-0.323331	0.718504	1.
density	-0.406289	0.447218	0.272070	0.086309	0.531107	0.443052	0.045610	0.
pH	-0.324525	-0.239439	0.247969	-0.324736	-0.267738	0.084952	-0.150079	-0.
sulphates	-0.491583	0.295957	0.230027	0.052697	-0.191229	0.369757	-0.190768	-0.
alcohol	0.041491	-0.104516	-0.042641	-0.003541	-0.348267	-0.297910	-0.185907	-0.
quality	0.119323	-0.078996	-0.259562	0.086396	-0.035623	-0.231030	0.057618	-0.

Также построим матрицу корреляций по Пирсону

```
# Треугольный вариант матрицы Пирсона
mask = np.zeros_like(data.corr(), dtype=np.bool)
mask[np.tril_indices_from(mask)] = True
sns.heatmap(data.corr(), mask=mask, annot=True, fmt='.1f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f08f123ae10>
```



По матрице видим корреляцию целевого поля type с несколькими параметрами:

- total sulfur dioxide (0,7)
- free sulfur dioxide (0,5)
- volatile acidity (-0,6)
- chlorides(-0,6)

```
1  data['type'].unique()
```

```
array([1, 0])
```

Выбор метрик для последующей оценки качества моделей. Нео двух метрик и обосновать выбор.

Модель классификации должна будет предсказать один из двух классов (белое/красное) Д использованы метрики accuracy, precision и recall.

- Accuracy — доля правильных ответов алгоритма:
- Precision можно интерпретировать как долю объектов, названных классификатором п действительно являющимися положительными
- Recall показывает, какую долю объектов положительного класса из всех объектов по

Выбор наиболее подходящих моделей для решения задачи кла Необходимо использовать не менее трех моделей, хотя бы одна ансамблевой.

Для решения задачи классификации были выбраны 3 модели:

1. Логистическая регрессия
2. SVC - Support Vector Regression
3. Ансамблевая модель RandomForestClassifier (RFC)

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

Создаём 3 модели без изменений гиперпараметров для построения базового решения

```
model1 = LogisticRegression()
model2 = SVC()
model3 = RandomForestClassifier()
result_df = pd.DataFrame()
```

```
from sklearn.model_selection import train_test_split
```

Формирование обучающей и тестовой выборок на основе исход

```
X_train, X_test, y_train, y_test = train_test_split(data[['volatile acidity','chlorid
                                                          'free sulfur dioxide' ]], \
                                                    data['type'], \
                                                    test_size=0.3, \
                                                    random_state=42)
```

```
data.dropna(inplace=True)
```

```
X_train.isna().sum()
```

```
volatile acidity       0
chlorides              0
free sulfur dioxide    0
dtype: int64
```

Построение базового решения (baseline) для выбранных модел гиперпараметров. Производится обучение моделей на основе с качества моделей на основе тестовой выборки.

```
%time model1.fit(X_train,y_train)
```

```
CPU times: user 31.5 ms, sys: 18.5 ms, total: 50 ms
Wall time: 30 ms
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
%time model2.fit(X_train,y_train)
```

```
CPU times: user 163 ms, sys: 91.9 ms, total: 255 ms
Wall time: 153 ms
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
%time model3.fit(X_train,y_train)
```

```
CPU times: user 313 ms, sys: 1.09 ms, total: 314 ms
Wall time: 315 ms
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
from sklearn.metrics import recall_score, precision_score, f1_score, accuracy_score
```

```
result_df.loc['LR', 'AS train'] = accuracy_score(y_train, model1.predict(X_train))
result_df.loc['LR', 'AS test'] = accuracy_score(y_test, model1.predict(X_test))
result_df.loc['LR', 'PS train'] = precision_score(y_train, model1.predict(X_train))
result_df.loc['LR', 'PS test'] = precision_score(y_test, model1.predict(X_test))
result_df.loc['LR', 'RC train'] = recall_score(y_train, model1.predict(X_train))
result_df.loc['LR', 'RC test'] = recall_score(y_test, model1.predict(X_test))
result_df.loc['LR', 'F1 train'] = f1_score(y_train, model1.predict(X_train))
result_df.loc['LR', 'F1 test'] = f1_score(y_test, model1.predict(X_test))
```

```
result_df.loc['SVC', 'AS train'] = accuracy_score(y_train, model2.predict(X_train))
result_df.loc['SVC', 'AS test'] = accuracy_score(y_test, model2.predict(X_test))
result_df.loc['SVC', 'PS train'] = precision_score(y_train, model2.predict(X_train))
result_df.loc['SVC', 'PS test'] = precision_score(y_test, model2.predict(X_test))
result_df.loc['SVC', 'RC train'] = recall_score(y_train, model2.predict(X_train))
result_df.loc['SVC', 'RC test'] = recall_score(y_test, model2.predict(X_test))
result_df.loc['SVC', 'F1 train'] = f1_score(y_train, model2.predict(X_train))
result_df.loc['SVC', 'F1 test'] = f1_score(y_test, model2.predict(X_test))
```

```
result_df.loc['RFC', 'AS train'] = accuracy_score(y_train, model3.predict(X_train))
result_df.loc['RFC', 'AS test'] = accuracy_score(y_test, model3.predict(X_test))
result_df.loc['RFC', 'PS train'] = precision_score(y_train, model3.predict(X_train))
result_df.loc['RFC', 'PS test'] = precision_score(y_test, model3.predict(X_test))
```

```
result_df.loc['RFC', 'PS test'] = precision_score(y_test, model3.predict(X_test))
result_df.loc['RFC', 'RC train'] = recall_score(y_train, model3.predict(X_train))
result_df.loc['RFC', 'RC test'] = recall_score(y_test, model3.predict(X_test))
result_df.loc['RFC', 'F1 train'] = f1_score(y_train, model3.predict(X_train))
result_df.loc['RFC', 'F1 test'] = f1_score(y_test, model3.predict(X_test))
```

```
result_df
```

	AS train	AS test	PS train	PS test	RC train	RC test	F1 train	F1 test
LR	0.930724	0.925128	0.946070	0.943813	0.962920	0.957909	0.954420	0.950809
SVC	0.963492	0.951795	0.974658	0.967458	0.976934	0.968771	0.975795	0.968114
RFC	0.999120	0.957949	0.999125	0.968982	0.999708	0.975560	0.999416	0.972260

Вывод

При сравнении трёх моделей, лучше всего с данной задачей классификации справилась ансамблевая модель Random Forest Classifier. Точность составила 95,7%