Домашняя работа

ФИО: Ваняшкин Ю.Ю.

Группа: ИУ5-24М

Задание

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. Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных. Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков. Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей. 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. Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее двух метрик и обосновать выбор. Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее трех моделей, хотя бы одна из которых должна быть ансамблевой. Формирование обучающей и тестовой выборок на основе исходного набора данных. 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. Подбор гиперпараметров для выбранных моделей. Рекомендуется подбирать не более 1-2 гиперпараметров. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы. 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.

Описание данных

In [83]:

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

В качестве набора данных будем использовать датасет качетсва красного вина. Будем решать задачу регрессии.
В качестве целевого признака возьмем колонку "Качество вина" (Quality)

In [84]:

```
data=pd.read_csv("dataset.csv")
```

In [85]:

```
data.shape
```

Out[85]:

```
(1599, 12)
```

In [86]:

```
data.isna().sum()
```

Out[86]:

```
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
```

Отсутствующих данных нет

Разведочный анализ

In [87]:

```
data.head()
data.describe()
data.columns
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64  
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

Проверим корреляцию между признаками

Корреляционный анализ, выбор подходящих признаков

In [88]:

```
corr = data.corr()
```

In [89]:

```
corr
```

Out[89]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.256131	0.671703	0.114777	0.093705	-0.153794	-0.113181	0.668047	-0.682978	0.183006	-0.061668	0.124052
volatile acidity	-0.256131	1.000000	-0.552496	0.001918	0.061298	-0.010504	0.076470	0.022026	0.234937	-0.260987	-0.202288	-0.390558
citric acid	0.671703	-0.552496	1.000000	0.143577	0.203823	-0.060978	0.035533	0.364947	-0.541904	0.312770	0.109903	0.226373
residual sugar	0.114777	0.001918	0.143577	1.000000	0.055610	0.187049	0.203028	0.355283	-0.085652	0.005527	0.042075	0.013732

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
chlorides	0.090709	0.061298	0.203823	0.055610	1.000000	0.005562	0.047400	0.200632	-0.265026	0.371260	-0.221141	-0.128907
free sulfur dioxide	-0.153794	-0.010504	-0.060978	0.187049	0.005562	1.000000	0.667666	-0.021946	0.070377	0.051658	-0.069408	-0.050656
total sulfur dioxide	-0.113181	0.076470	0.035533	0.203028	0.047400	0.667666	1.000000	0.071269	-0.066495	0.042947	-0.205654	-0.185100
density	0.668047	0.022026	0.364947	0.355283	0.200632	-0.021946	0.071269	1.000000	-0.341699	0.148506	-0.496180	-0.174919
pH	-0.682978	0.234937	-0.541904	-0.085652	-0.265026	0.070377	-0.066495	-0.341699	1.000000	-0.196648	0.205633	-0.057731
sulphates	0.183006	-0.260987	0.312770	0.005527	0.371260	0.051658	0.042947	0.148506	-0.196648	1.000000	0.093595	0.251397
alcohol	-0.061668	-0.202288	0.109903	0.042075	-0.221141	-0.069408	-0.205654	-0.496180	0.205633	0.093595	1.000000	0.476166
quality	0.124052	-0.390558	0.226373	0.013732	-0.128907	-0.050656	-0.185100	-0.174919	-0.057731	0.251397	0.476166	1.000000

Построим тепловую карту корреляции для более наглядного представления

In [90]:

```
sns.heatmap(corr, cmap="YlGnBu", annot=True)
```

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1356f2390>
```

Построим графики, чтобы понять структуру данных

In [91]:

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

Out[91]:

```
<seaborn.axisgrid.PairGrid at 0x135b25a90>
```

In [92]:

```
corr = data.corr()
sns.heatmap(corr, square=True, vmin=0.4, vmax=0.8,cmap="YlGnBu",annot=True)
```

Out[92]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1399a0810>
```

Мы можем решать задачу регрессии, пытаясь предсказать шанс (%) поступления

Выделим целевой признак и нормализуем данные

In [93]:

```
target = data['quality']
data = data.drop(['quality'], axis=1)
```

In [94]:

In [94]:

```
from sklearn import preprocessing
data = preprocessing.scale(data)
```

Метрики качества

В качестве метрик качества мы будет использовать среднюю квадратичную ошибку, среднюю абсолютную ошибку и коэффициент детерминации

Средняя квадратичная ошибка:

$$MAE(y, \hat{y}) = \frac{1}{N} \cdot \sum_{i=1}^{N} |y_i - \hat{y_i}|$$

где:

y - истинное значение целевого признака

$\hat{y}$ - предсказанное значение целевого признака

N - размер тестовой выборки

Чем ближе значение к нулю, тем лучше качество регрессии.

Основная проблема метрики состоит в том, что она не нормирована.

Средняя абсолютная ошибка:

$$MSE(y, \hat{y}) = \frac{1}{N} \cdot \sum_{i=1}^{N} (y_i - \hat{y_i})^2$$

где:

y - истинное значение целевого признака

$\hat{y}$ - предсказанное значение целевого признака

N - размер тестовой выборки

Коэффициент детерминации:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y_i})^2}{\sum_{i=1}^{N} (y_i - \bar{y_i})^2}$$

где:

y - истинное значение целевого признака

$\hat{y}$ - предсказанное значение целевого признака

N - размер тестовой выборки

$$\bar{y_i} = \frac{1}{N} \cdot \sum_{i=1}^{N} y_i$$

In [95]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

Формирование обучающей и тестовой выборки

В качестве моделей регрессии выберем модель BaggingRegressor, KneighborsRegressor и RandomForestRegressor

In [96]:

```
from sklearn.ensemble import BaggingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
```

Разделим выборку в пропорции 1:4

In [97]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.2, random_state=1)
```

In [98]:

```
X_train.shape, y_train.shape
```

Out[98]:

```
((1279, 11), (1279,))
```

In [99]:

```
X_test.shape, y_test.shape
```

Out[99]:

```
((320, 11), (320,))
```

Подбор гиперпараметров моделей

In [100]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score, cross_validate
```

Подбор гиперпараметров для модели BaggingRegressor

In [101]:

```
param_grid = {
    'n_estimators' : [1, 3, 6, 9, 12, 15, 20, 25, 30],
    'max_samples' : [0.01, 0.1, 0.4, 0.7],
    'max_features' : [2, 4, 8, 10]
}

bagging = BaggingRegressor()
grid = GridSearchCV(estimator=bagging, param_grid=param_grid)
grid.fit(X_train, y_train)
print(grid)

print(grid.best_score_)
print(grid.best_estimator_)
```

```
GridSearchCV(estimator=BaggingRegressor(),
             param_grid={'max_features': [2, 4, 8, 10],
                         'max_samples': [0.01, 0.1, 0.4, 0.7],
                         'n_estimators': [1, 3, 6, 9, 12, 15, 20, 25, 30]})
0.4624483461984088
BaggingRegressor(max_features=10, max_samples=0.7, n_estimators=25)
```

Подбор параметров для KNeighborsRegressor

In [102]:

```
grid_params = {
    'n_neighbors': [3, 9, 15, 20, 25],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}

grid = GridSearchCV(KNeighborsRegressor(), grid_params, verbose=1, cv=3, n_jobs=-1)
grid.fit(X_train, y_train)
print(grid)

print(grid.best_score_)
print(grid.best_estimator_)
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
GridSearchCV(cv=3, estimator=KNeighborsRegressor(), n jobs=-1,
```

```
             param_grid={'metric': ['euclidean', 'manhattan'],
                         'n_neighbors': [3, 9, 15, 20, 25],
                         'weights': ['uniform', 'distance']},
             verbose=1)
0.4295060180466537
KNeighborsRegressor(metric='manhattan', n_neighbors=20, weights='distance')
[Parallel(n_jobs=-1)]: Done  60 out of  60 | elapsed:    1.0s finished
```

Подбор параметров для RandomForestRegressor

In [103]:

```
grid_params= {
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [3, 5, 9, 12, 15],
    'min_samples_split': [2, 4, 6, 8, 10],
    'min_samples_leaf': [1, 2, 4, 6]
}
grid = GridSearchCV(RandomForestRegressor(), grid_params, cv=2, n_jobs=-1)
grid.fit(X_train, y_train)
print(grid)

print(grid.best_score_)
print(grid.best_estimator_)
```

```
GridSearchCV(cv=2, estimator=RandomForestRegressor(), n_jobs=-1,
             param_grid={'max_depth': [3, 5, 9, 12, 15],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 2, 4, 6],
                         'min_samples_split': [2, 4, 6, 8, 10]})
0.4361209272198616
RandomForestRegressor(max_depth=15, max_features='sqrt', min_samples_leaf=2,
                      min_samples_split=6)
```

Обучение с оптимальными значениями гиперпараметров

In [104]:

```
def quality(test, predicted):
    print("Метрики качества:")
    print("Средняя квадратичная ошибка: " + str(mean_squared_error(test, predicted)))
    print("Средняя абсолютная ошибка: " + str(mean_absolute_error(test, predicted)))
    print("Коэффициент детерминации: " + str(r2_score(test, predicted)))

models = [BaggingRegressor(base_estimator=None, bootstrap=True, bootstrap_features=False,
                  max_features=4, max_samples=0.5, n_estimators=25, n_jobs=None,
                  oob_score=False, random_state=None, verbose=0,
                  warm_start=False),
          KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='manhattan',
                     metric_params=None, n_jobs=None, n_neighbors=11, p=2,
                     weights='distance'),
          RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=9, 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=4, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
         ]

for model in models:
    print("========================================")
    print("Обучение модели " + type(model).__name__)
    model.fit(X_train, y_train)
    predicted = model.predict(X_test)
    plt.figure(figsize=(4, 4))
    plt.scatter(y_test,predicted)
    plt.title(type(model).__name__)
    plt.xlabel('Actual wine quality')
    plt.ylabel('Predicted wine quality')
    plt.tight_layout()
    quality(y_test, predicted)
```

```
    quality(y_test, predicted)
```

```
========================================
Обучение модели BaggingRegressor
Метрики качества:
Средняя квадратичная ошибка: 0.37305125
Средняя абсолютная ошибка: 0.46281250000000007
Коэффициент детерминации: 0.3442546047549566
========================================
Обучение модели KNeighborsRegressor
Метрики качества:
Средняя квадратичная ошибка: 0.34438313523442743
Средняя абсолютная ошибка: 0.39607220053975806
Коэффициент детерминации: 0.3946471024288839
========================================
Обучение модели RandomForestRegressor
Метрики качества:
Средняя квадратичная ошибка: 0.3410027855511446
Средняя абсолютная ошибка: 0.44163308917227884
Коэффициент детерминации: 0.4005890440230503
```

Лучшей оказалась модель случайного леса. Оптимизация гиперпараметров не дала большого эффекта.
Метрики качества показывают, что все модели, построенные в результате выполнения проекта, являются недостаточно хорошими для их использования. При этом метод леса показал себя лучше классического алгоритма.