# Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»
Отчет по рубежному контролю №2
«Методы построения моделей машинного обучения»
Вариант №2


Выполнил:
студент группы ИУ5-61Б
Бондаренко Денис
Константинович

Подпись:_____

Дата:_____

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич

Подпись:_____

Дата:_____


Москва, 2023 г.

# Выполнение работы

Для выполнения задачи построения моделей классификации был представлен набор данных sklearn wine dataset, загруженный с помощью функции load_wine().

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_wine
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from typing import Dict
```

In [2]:
```
wine = load_wine()
```

Был создан датафрейм, содержащий 13 нецелевых признаков и 1 целевой – класс вина.

In [3]:
```
wine_x_ds = pd.DataFrame(data=wine['data'], columns=wine['feature_names'])
wine_x_ds
```

Out[3]:

| | alcohol | malic_acid | ash | alcalinity_of_ash | magnesium | total_phenols | flavanoids | nonflavanoid_phenols | proanthocyanins | color_intensity | hue | od280/od3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 14.23 | 1.71 | 2.43 | 15.6 | 127.0 | 2.80 | 3.06 | 0.28 | 2.29 | 5.64 | 1.04 | |
| 1 | 13.20 | 1.78 | 2.14 | 11.2 | 100.0 | 2.65 | 2.76 | 0.26 | 1.28 | 4.38 | 1.05 | |
| 2 | 13.16 | 2.36 | 2.67 | 18.6 | 101.0 | 2.80 | 3.24 | 0.30 | 2.81 | 5.68 | 1.03 | |
| 3 | 14.37 | 1.95 | 2.50 | 16.8 | 113.0 | 3.85 | 3.49 | 0.24 | 2.18 | 7.80 | 0.86 | |
| 4 | 13.24 | 2.59 | 2.87 | 21.0 | 118.0 | 2.80 | 2.69 | 0.39 | 1.82 | 4.32 | 1.04 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 173 | 13.71 | 5.65 | 2.45 | 20.5 | 95.0 | 1.68 | 0.61 | 0.52 | 1.06 | 7.70 | 0.64 | |
| 174 | 13.40 | 3.91 | 2.48 | 23.0 | 102.0 | 1.80 | 0.75 | 0.43 | 1.41 | 7.30 | 0.70 | |
| 175 | 13.27 | 4.28 | 2.26 | 20.0 | 120.0 | 1.59 | 0.69 | 0.43 | 1.35 | 10.20 | 0.59 | |
| 176 | 13.17 | 2.59 | 2.37 | 20.0 | 120.0 | 1.65 | 0.68 | 0.53 | 1.46 | 9.30 | 0.60 | |
| 177 | 14.13 | 4.10 | 2.74 | 24.5 | 96.0 | 2.05 | 0.76 | 0.56 | 1.35 | 9.20 | 0.61 | |

178 rows × 13 columns

In [4]:
```
wine_y_ds = pd.DataFrame(data=wine['target'])
wine_y_ds
```

Out[4]:

| | 0 |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| ... | ... |
| 173 | 2 |
| 174 | 2 |
| 175 | 2 |
| 176 | 2 |
| 177 | 2 |

178 rows × 1 columns

In [5]:
```
wine_ds = pd.DataFrame(data=wine['data'], columns=wine['feature_names'])
wine_ds['target'] = wine['target']
wine_ds
```

Out[5]:

| :alinity_of_ash | magnesium | total_phenols | flavanoids | nonflavanoid_phenols | proanthocyanins | color_intensity | hue | od280/od315_of_diluted_wines | proline | target |
|---|---|---|---|---|---|---|---|---|---|---|
| 15.6 | 127.0 | 2.80 | 3.06 | 0.28 | 2.29 | 5.64 | 1.04 | 3.92 | 1065.0 | 0 |
| 11.2 | 100.0 | 2.65 | 2.76 | 0.26 | 1.28 | 4.38 | 1.05 | 3.40 | 1050.0 | 0 |
| 18.6 | 101.0 | 2.80 | 3.24 | 0.30 | 2.81 | 5.68 | 1.03 | 3.17 | 1185.0 | 0 |
| 16.8 | 113.0 | 3.85 | 3.49 | 0.24 | 2.18 | 7.80 | 0.86 | 3.45 | 1480.0 | 0 |
| 21.0 | 118.0 | 2.80 | 2.69 | 0.39 | 1.82 | 4.32 | 1.04 | 2.93 | 735.0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 20.5 | 95.0 | 1.68 | 0.61 | 0.52 | 1.06 | 7.70 | 0.64 | 1.74 | 740.0 | 2 |
| 23.0 | 102.0 | 1.80 | 0.75 | 0.43 | 1.41 | 7.30 | 0.70 | 1.56 | 750.0 | 2 |
| 20.0 | 120.0 | 1.59 | 0.69 | 0.43 | 1.35 | 10.20 | 0.59 | 1.56 | 835.0 | 2 |
| 20.0 | 120.0 | 1.65 | 0.68 | 0.53 | 1.46 | 9.30 | 0.60 | 1.62 | 840.0 | 2 |
| 24.5 | 96.0 | 2.05 | 0.76 | 0.56 | 1.35 | 9.20 | 0.61 | 1.60 | 560.0 | 2 |

Типы данных всех полей являются числовыми.

In [6]:
```
wine_ds.dtypes
```

Out[6]:
```
alcohol                         float64
malic_acid                      float64
ash                             float64
alcalinity_of_ash               float64
magnesium                       float64
total_phenols                   float64
flavanoids                      float64
nonflavanoid_phenols            float64
proanthocyanins                 float64
color_intensity                 float64
hue                             float64
od280/od315_of_diluted_wines    float64
proline                         float64
target                            int32
dtype: object
```

В наборе данных отсутствуют пропуски и дубликаты.

In [7]:
```
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in wine_ds.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = wine_ds[wine_ds[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
alcohol - 0
malic_acid - 0
ash - 0
alcalinity_of_ash - 0
magnesium - 0
total_phenols - 0
flavanoids - 0
nonflavanoid_phenols - 0
proanthocyanins - 0
color_intensity - 0
hue - 0
od280/od315_of_diluted_wines - 0
proline - 0
target - 0
```

In [8]:
```
wine_ds.duplicated().sum()
```

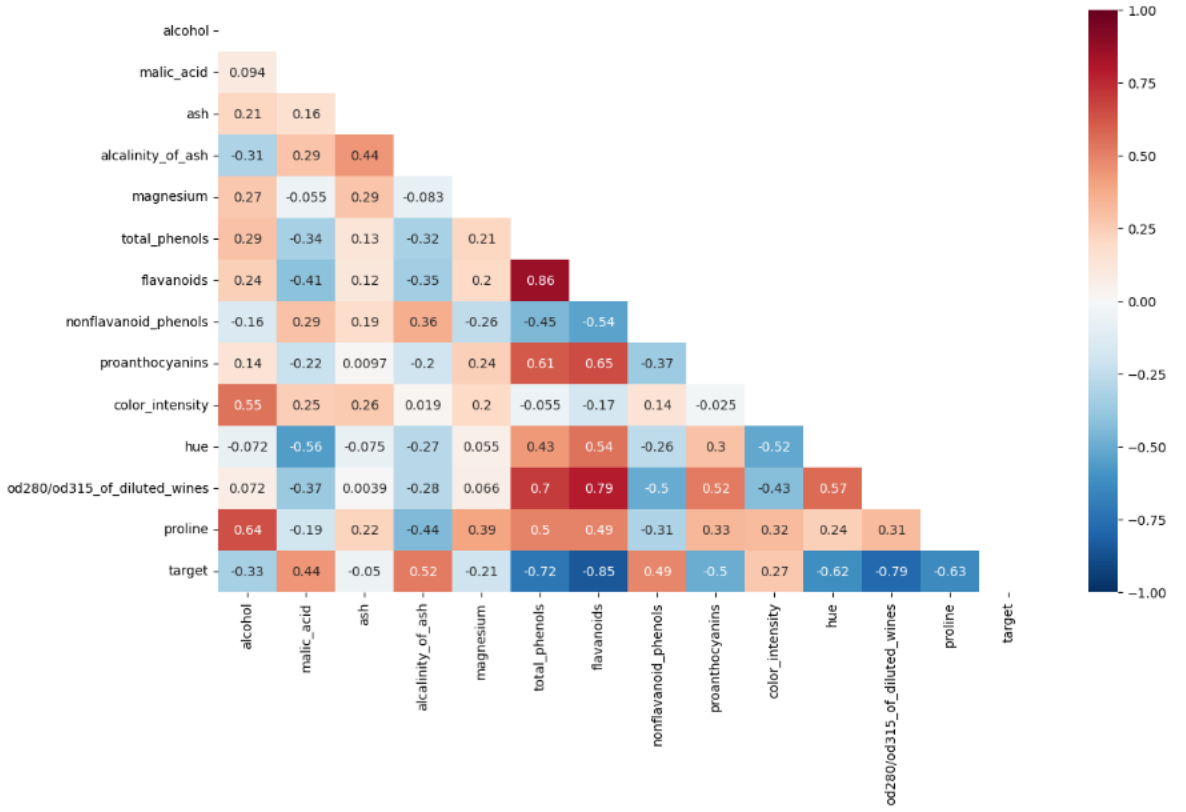
Out[8]:
```
0
```

Проведем корреляционный анализ, чтобы оценить вклад признаков для построения моделей классификации. Для визуализации корреляционной матрицы была использована “тепловая карта”.

```
In [9]: plt.figure(figsize = (14,8))

        m =np.triu(np.ones_like(wine_ds.corr(), dtype=bool))

        sns.heatmap(wine_ds.corr(), mask = m, annot = True, vmin= -1.0, vmax= 1.0, center = 0, cmap = 'RdBu_r');
```




С целевым признаком наиболее сильную корреляцию имеют признаки “flavanoids” (-0,85), “od280/od315_of_diluted_wines” (-0,79), “total_phenols” (-0,72), “proline” (-0,63) и “hue” (-0,62). Эти признаки будут наиболее информативными при построении моделей машинного обучения. Целевой признак отчасти коррелирует с признаками “alcalinity_of_ash” (0,52), “proanthocyanins” (-0,5), “nonflavanoid_fenols” (0,49) и “malic_acid” (0,44). Эти признаки также стоит использовать при обучении модели. Признаки “alcohol” (-0,33), “color_intensity” (0,27), “magnesium” (-0,21) и “ash” (-0,05) слабо коррелируют с целевым признаком и могут негативно сказаться на модели машинного обучения, поэтому, скорее всего, их стоит исключить из модели.

Но не все признаки, которые имеют сильную и среднюю корреляцию с целевым признаком, стоит использовать для построения модели машинного обучения. Между признаками “flavanoids” и “total_phenols” наблюдается очень сильная корреляция (0,86). Это связано с тем, что флавоноиды относятся к классу полифенолов. Поэтому из этих двух признаков стоит оставить тот, который имеет наибольшую корреляцию с целевым признаком, т.е.

“flavanoids”. Остальные нецелевые признаки не коррелируют друг с другом так сильно и между ними не наблюдается почти линейной зависимости.

Таким образом, на основе признаков “flavanoids”, “od280/od315_of_diluted_wines”, “proline”, “hue”, “alcalinity_of_ash”, “proanthocyanins”, “nonflavanoid_phenols” и “malic_acid” могут быть построены модели машинного обучения, первые четыре признака могут иметь наиболее весомый вклад в их обучение. Для обучения моделей классификации будут использоваться эти 8 нецелевых признаков.

Выборка экземпляров вина, принадлежащих разным классам, является сбалансированной.

```
In [10]: wine_y_ds.value_counts()

Out[10]: 1    71
         0    59
         2    48
         dtype: int64
```

Разобьем исходную выборку на обучающую и тестовую.

```
In [11]: wine_X_train, wine_X_test, wine_y_train, wine_y_test = train_test_split(
             wine_ds[['malic_acid', 'alcalinity_of_ash', 'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'hue', 'od280/od315_of_
             wine_ds['target'].values, test_size=0.2, random_state=2)
```

Было произведено MinMax масштабирование данных.

```
In [12]: mms = MinMaxScaler()

In [13]: wine_X_train_scaled = mms.fit_transform(wine_X_train)
         wine_X_test_scaled = mms.transform(wine_X_test)
```

Была обучена модель логистической регрессии.

```
In [14]: cl = LogisticRegression(multi_class='multinomial')

In [15]: cl.fit(wine_X_train_scaled, wine_y_train)

Out[15]: LogisticRegression(multi_class='multinomial')
```

Результаты классификации с использованием модели логистической регрессии:

```
In [16]: pred_wine_y_test = cl.predict(wine_X_test_scaled)
         pred_wine_y_test

Out[16]: array([0, 0, 2, 1, 0, 0, 1, 2, 1, 0, 1, 0, 0, 2, 2, 1, 0, 0, 0, 2, 2, 0,
                1, 1, 0, 0, 1, 0, 0, 1, 2, 1, 2, 2, 0, 1])
```

Для оценки качества моделей машинного обучения были использованы метрики accuracy и F1-мера. Метрика accuracy подходит для оценки качества моделей классификации для заданного набора данных, так как классификация производится по трем равноценным классам и нет необходимости в более точном определении того или иного класса. Также она подходит, так как выборка является сбалансированной, поэтому точность по всем классам, которую и отражает accuracy, не будет скрывать малую точность для отдельного класса. Метрика F1-мера подходит для оценки качества моделей классификации для заданного набора данных, так как в случае классификации по трем равноценным классам precision и recall имеют равное значение, поэтому их оценку можно совместить в метрике F1-мера. Распределение экземпляров вина из набора данных по классам не будет иметь отрицательного влияния на значение метрики F1-мера, так как выборка является сбалансированной.

Значение метрики accuracy для модели логистической регрессии:

```
In [17]: accuracy_score(wine_y_test, pred_wine_y_test)

Out[17]: 0.9444444444444444
```

Функции для вывода значения метрики accuracy для каждого класса:

In [18]:
```
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]))
```

Значение метрики accuracy для каждого класса:

In [19]:
```
print_accuracy_score_for_classes(wine_y_test, pred_wine_y_test)
```

```
Метка    Accuracy
0        0.8888888888888888
1        1.0
2        1.0
```

Значение метрики F1-мера для модели логистической регрессии для каждого класса:

In [20]:
```
f1_score(wine_y_test, pred_wine_y_test, average=None)
```

Out[20]:
```
array([0.94117647, 0.9       , 1.        ])
```

Была обучена модель случайного леса.

```
In [22]: wine_rf_cl = RandomForestClassifier(random_state=2)

In [23]: wine_rf_cl.fit(wine_X_train_scaled, wine_y_train)
Out[23]: RandomForestClassifier(random_state=2)
```

Результаты классификации с использованием модели случайного леса:

```
In [24]: pred_wine_rf_y_test = wine_rf_cl.predict(wine_X_test_scaled)
         pred_wine_rf_y_test
Out[24]: array([0, 0, 2, 1, 0, 0, 1, 2, 1, 0, 1, 0, 0, 2, 2, 1, 0, 0, 0, 2, 2, 0,
                1, 1, 0, 0, 1, 0, 0, 0, 2, 1, 2, 2, 0, 1])
```

Значение метрики accuracy для модели случайного леса:

```
In [25]: accuracy_score(wine_y_test, pred_wine_rf_y_test)
Out[25]: 0.9722222222222222
```

Значение метрики accuracy для каждого класса:

```
In [26]: print_accuracy_score_for_classes(wine_y_test, pred_wine_rf_y_test)
Метка    Accuracy
0        0.9444444444444444
1        1.0
2        1.0
```

Значение метрики F1-мера для модели случайного леса для каждого класса:

```
In [27]: f1_score(wine_y_test, pred_wine_rf_y_test, average=None)
Out[27]: array([0.97142857, 0.94736842, 1.        ])
```

Таким образом, каждая из моделей машинного обучения классифицирует вино с высокой точностью. Обе модели безошибочно определяют вино первого и второго класса, но в малом количестве случаев определяют вино класса 0 как вино класса 1. Модель случайного леса производит классификацию лучше модели логистической регрессии, так как значение каждой метрики для модели случайного леса не хуже значения соответствующей метрики для модели логистической регрессии.