РК-2

Бушуев В.М. РТ5-61Б

В качестве датасета будем использовать "игрушечный" датасет wineиз sklearn.

Атрибуты:

Alcohol

Malic acid

Ash

Alcalinity of ash

Magnesium

Total phenols

Flavanoids

Nonflavanoid phenols

Proanthocyanins

Color intensity

Hue

OD280/OD315 of diluted wines

Proline

Ввод [ ]:
```
import matplotlib
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn import datasets, preprocessing
from xgboost import XGBClassifier
from sklearn.metrics import f1_score,  balanced_accuracy_score, ConfusionMatrixDisplay
from sklearn.preprocessing import StandardScaler, MinMaxScaler


%matplotlib inline
sns.set(style="dark")
```

Ввод [ ]:
```
# Загрузка датасета
wine = datasets.load_wine(as_frame=True)
df : pd.DataFrame = wine.frame
```

Основные харакеристики датасета

Размер набора данных

Ввод [ ]:
```
df.shape
```

Out[5]:
```
(178, 14)
```

Типы колонок

Ввод [ ]:
```
df.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                            int64
dtype: object
```

Проверка наличие пропусков

Ввод [ ]:
```
df.isnull().sum()
```

Out[8]:
```
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
dtype: int64
```

Пропусков нет

Ввод [ ]:
```
df.head(10)
```

Out[10]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315
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	
5	14.20	1.76	2.45	15.2	112.0	3.27	3.39	0.34	1.97	6.75	1.05	
6	14.39	1.87	2.45	14.6	96.0	2.50	2.52	0.30	1.98	5.25	1.02	
7	14.06	2.15	2.61	17.6	121.0	2.60	2.51	0.31	1.25	5.05	1.06	
8	14.83	1.64	2.17	14.0	97.0	2.80	2.98	0.29	1.98	5.20	1.08	
9	13.86	1.35	2.27	16.0	98.0	2.98	3.15	0.22	1.85	7.22	1.01	

Ввод [ ]:
```
# Копия датасета для визуализации
d_to_viz = df.copy()
d_to_viz['target'] = d_to_viz['target'].\
 map({i : wine.target_names[i] for i in range(0, len(wine.target_names))})
d_to_viz.sample(5, random_state=1)
```

Типы колонок

Ввод [ ]:
```
df.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                            int64
dtype: object
```

Проверка наличие пропусков

Ввод [ ]:
```
df.isnull().sum()
```

Out[8]:
```
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
dtype: int64
```

Пропусков нет

Ввод [ ]:
```
df.head(10)
```

Out[10]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315
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	
5	14.20	1.76	2.45	15.2	112.0	3.27	3.39	0.34	1.97	6.75	1.05	
6	14.39	1.87	2.45	14.6	96.0	2.50	2.52	0.30	1.98	5.25	1.02	
7	14.06	2.15	2.61	17.6	121.0	2.60	2.51	0.31	1.25	5.05	1.06	
8	14.83	1.64	2.17	14.0	97.0	2.80	2.98	0.29	1.98	5.20	1.08	
9	13.86	1.35	2.27	16.0	98.0	2.98	3.15	0.22	1.85	7.22	1.01	

Ввод [ ]:
```
# Копия датасета для визуализации
d_to_viz = df.copy()
d_to_viz['target'] = d_to_viz['target'].\
 map({i : wine.target_names[i] for i in range(0, len(wine.target_names))})
d_to_viz.sample(5, random_state=1)
```

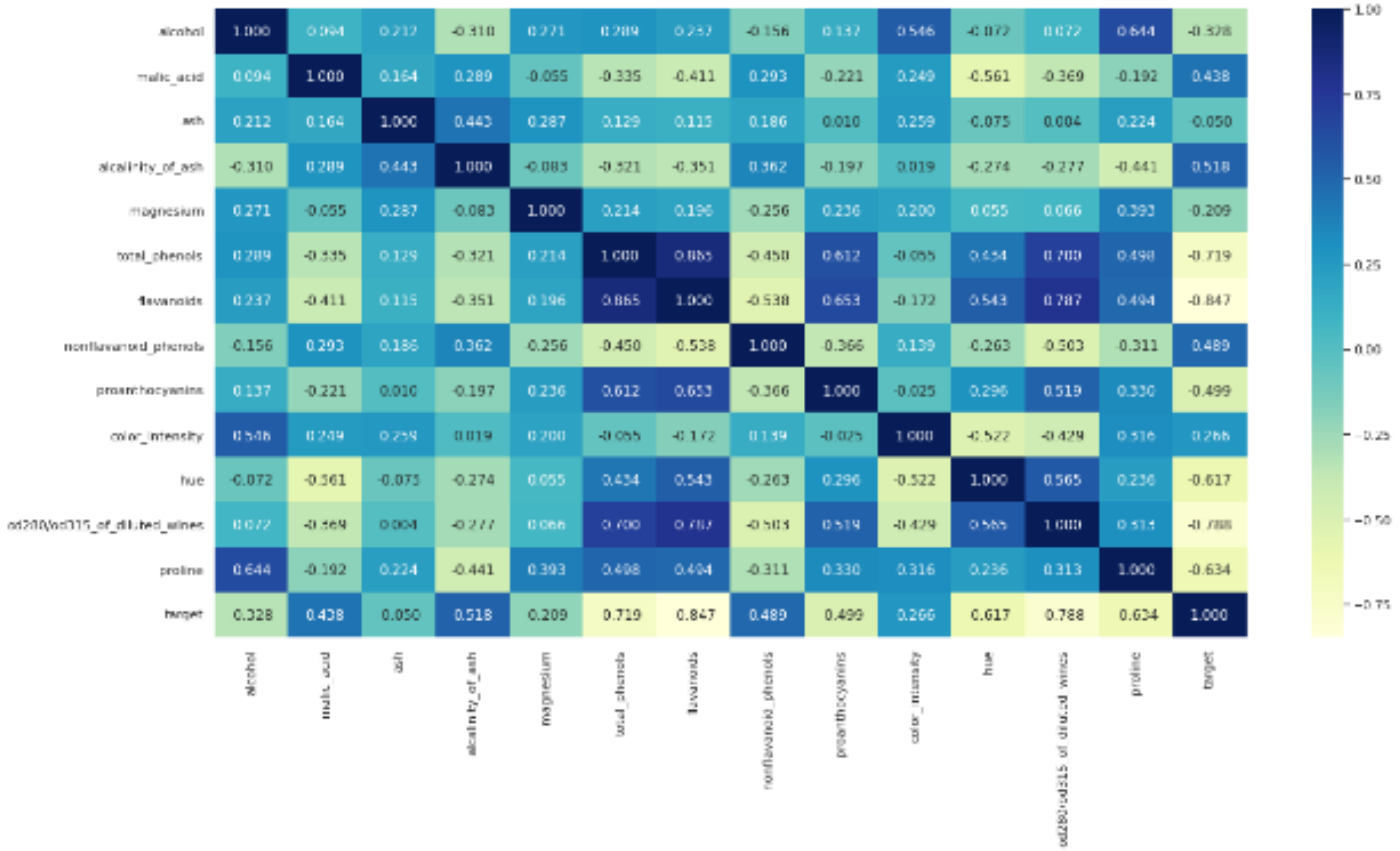
Out[14]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od3
161	13.69	3.26	2.54	20.0	107.0	1.83	0.56	0.50	0.80	5.88	0.96	
117	12.42	1.61	2.19	22.5	108.0	2.00	2.09	0.34	1.61	2.06	1.06	
19	13.64	3.10	2.56	15.2	116.0	2.70	3.03	0.17	1.66	5.10	0.96	
69	12.21	1.19	1.75	16.8	151.0	1.85	1.28	0.14	2.50	2.85	1.28	
53	13.77	1.90	2.68	17.1	115.0	3.00	2.79	0.39	1.68	6.30	1.13	

Корреляционная матрица

Ввод [ ]:
```
ig, ax = plt.subplots(figsize=(20,10))
sns.heatmap(df.corr(method='pearson'), ax=ax, annot=True, fmt='.3f', cmap="YlGnBu")
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f02ece39290>

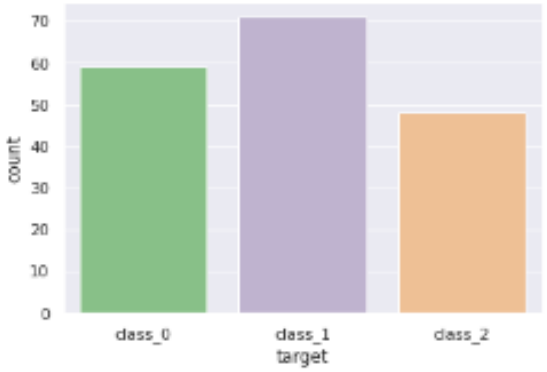


Наблюдается довольно сильная корреляция между некоторыми нецелевыми признаками. Это может повлиять на результаты использования моделей. Стоит задуматься об исключении пар сильно коррелирующих параметров.

Ввод [ ]:
```
df.target.value_counts()
```

Out[15]:
```
1    71
0    59
2    48
Name: target, dtype: int64
```

Ввод [ ]:
```
# График распределения классов
with sns.axes_style('darkgrid'):
 sns.countplot(x='target', data=d_to_viz, palette='Accent')
 plt.show()
```



Обучение моделей

Используемые метрики

Для получения обобщенной эффективности используем кросс-валидацию. Используем метрики balanced_accuracy И f1.

Делим данные на контролную и обучающую выборки

Ввод [ ]:
```
X = df.drop(columns=['target'])
y = df['target']
# разделим данные с помощью Scikit-Learn's train_test_split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

Градиентный бустинг

Ввод [ ]:
```
model = XGBClassifier(eval_metric='mlogloss')
#метрики

f1 = cross_val_score(model, X_train, y_train, cv=5, scoring='f1_weighted').mean()
recall_weighted = cross_val_score(model, X_train, y_train, cv=5, scoring='recall_weighted').mean()
balanced_accuracy = cross_val_score(model, X_train, y_train, cv=5, scoring='balanced_accuracy').mean()
print('f1_weighted: '+ str(round(f1,2)) + ' balanced_accuracy ' + str(round(balanced_accuracy,2)) + ' recall_weighted ' + str(ro
```

```
f1_weighted: 0.96 balanced_accuracy 0.97 recall_weighted 0.96
```

k-NN

Ввод [ ]:
```
from sklearn.neighbors import KNeighborsClassifier
logreg = KNeighborsClassifier(n_neighbors=6)

# метрики
f1 = cross_val_score(logreg, X_train, y_train, cv=5, scoring='f1_weighted').mean()
recall_weighted = cross_val_score(logreg, X_train, y_train, cv=5, scoring='recall_weighted').mean()
balanced_accuracy = cross_val_score(logreg, X_train, y_train, cv=5, scoring='balanced_accuracy').mean()
print('f1_weighted: '+ str(round(f1,2)) + ' balanced_accuracy ' + str(round(balanced_accuracy,2)) + ' recall_weighted ' + str(ro
```

```
f1_weighted: 0.63 balanced_accuracy 0.65 recall_weighted 0.65
```

Видно, что k-NN показывает себя хуже, чем градиентный бустинг, как более протсая и менне наежная модель.

Проверка на контрольной выборке

Градиентный бустинг

Градиентный бустинг

Ввод [ ]:
```
model.fit(X_train, y_train)
XGB_res = model.predict(X_test)
bascore = balanced_accuracy_score(y_test, XGB_res)
f1 = f1_score(y_test, XGB_res, average='weighted')
print('f1_weighted: '+ str(round(f1,2)) + ' balanced_accuracy ' + str(round(bascore,2)))
```

```
f1_weighted: 0.96 balanced_accuracy 0.97
```

k-NN

Ввод [ ]:
```
logreg.fit(X_train, y_train)
knn_res = logreg.predict(X_test)
bascore = balanced_accuracy_score(y_test, knn_res)
f1 = f1_score(y_test, knn_res, average='weighted')
print('f1_weighted: '+ str(round(f1,2)) + ' balanced_accuracy ' + str(round(bascore,2)))
```

```
f1_weighted: 0.75 balanced_accuracy 0.7
```

Метрики показывают, то градиентный бустинг является более предпочтительной моделью для данного датасета.