Московский государственный технический
университет им. Н.Э. Баумана

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

Курс «Технологии машинного обучения»

Рубежный контроль №2
Вариант 3

Выполнил:
студент группы ИУ5-62Б
Барышников Михаил

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2022 г.

Описание задания:

Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Методы 1 и 2 для группы:

- ИУ5-62Б, ИУ5Ц-82Б 1) Метод опорных векторов 2) Случайный лес

Вариант №3

Задание

Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Датасет https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html#sklearn.datasets.load_win - набор данных о вине.

In [12]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,
```

In [13]:
```
# Импортируем датасет из sklearn.
from sklearn.datasets import load_wine
wine=load_wine()

# Преобразуем в датасет для pandas.
data=pd.DataFrame(data=np.c_[wine['data'],wine['target']],columns=wine['fea

# Проверяем данные после преобразования.
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    float64
dtypes: float64(14)
memory usage: 19.6 KB
```

Out[13]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavar
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

In [16]:
```
# Ищем пропуски.
(data.isnull() | data.empty | data.isna()).sum()
```

Out[16]:
```
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
```

Выбор метрик и подготовка данных

Так как выполняется задача небинарной классификации и в тестовой выборке возможен дисбаланс классов, были выбраны следующие метрики:

- precision;
- recall;
- f1-score.

Всем метрикам был задан уровень детализации average='weighted'.

In [17]:
```
def print_metrics(y_test, y_pred):
    rep = classification_report(y_test, y_pred, output_dict=True)
    print("weighted precision:", rep['weighted avg']['precision'])
    print("weighted recall:", rep['weighted avg']['recall'])
    print("weighted f1-score:", rep['weighted avg']['f1-score'])
    plt.figure(figsize=(4, 3))
    plt.title('Матрица ошибок')
    sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap="Blues")
```

In [135…
```
x_train, x_test, y_train, y_test = train_test_split(data.drop(['target'], a
```

Масштабирование данных

In [136…
```
scaler = StandardScaler().fit(x_train)
x_train_scaled = pd.DataFrame(scaler.transform(x_train), columns=x_train.co
x_test_scaled = pd.DataFrame(scaler.transform(x_test), columns=x_train.colu
x_train_scaled.describe()
```

Out[136]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phe
count	8.900000e+01	8.900000e+01	8.900000e+01	8.900000e+01	8.900000e+01	8.900000
mean	-1.616684e-15	-4.004288e-16	1.182575e-15	6.711236e-16	1.496930e-17	1.79631
std	1.005666e+00	1.005666e+00	1.005666e+00	1.005666e+00	1.005666e+00	1.005666
min	-2.121506e+00	-1.565042e+00	-2.311900e+00	-2.434169e+00	-1.549464e+00	-1.705957
25%	-7.679090e-01	-6.076227e-01	-5.943782e-01	-6.139109e-01	-7.921375e-01	-9.58416
50%	-2.726134e-02	-4.182430e-01	2.392951e-02	-3.749595e-02	-1.725065e-01	1.54940
75%	7.772352e-01	5.602187e-01	7.796389e-01	5.389190e-01	4.471245e-01	7.91145
max	2.143602e+00	2.622353e+00	2.909365e+00	3.269306e+00	4.233758e+00	2.429371

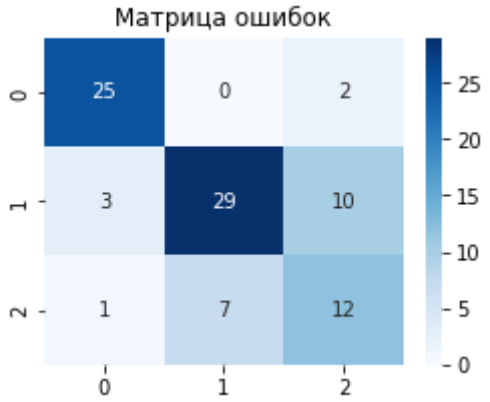
SVC

Базовая модель

Без масштабирования:

In [103…
```
svm_model = SVC()
svm_model.fit(x_train, y_train)
y_pred_svm = svm_model.predict(x_test)
print_metrics(y_test, y_pred_svm)
```

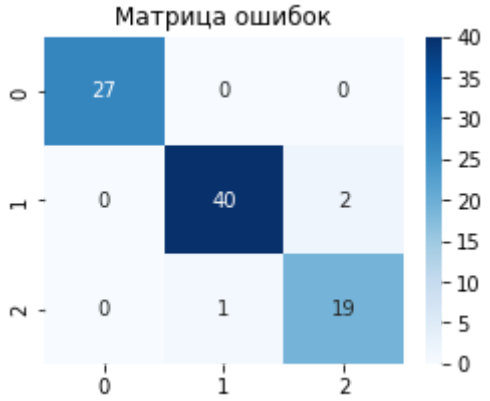
```
weighted precision: 0.7540359033966163
weighted recall: 0.7415730337078652
weighted f1-score: 0.7443483482809325
```



С масштабированием:

In [104…
```
svm_model = SVC()
svm_model.fit(x_train_scaled, y_train)
y_pred_svm = svm_model.predict(x_test_scaled)
print_metrics(y_test, y_pred_svm)
```

```
weighted precision: 0.967088178104895
weighted recall: 0.9662921348314607
weighted f1-score: 0.96650014692918
```



Подбор гиперпараметров

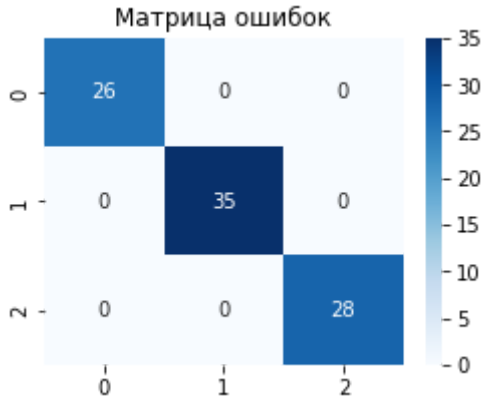
In [105…
```
params = {'C': np.concatenate([np.arange(0.1, 2, 0.03), np.arange(2, 20, 1)
grid_cv = GridSearchCV(estimator=svm_model, param_grid=params, cv=10, n_job
grid_cv.fit(x_train_scaled, y_train)
print(grid_cv.best_params_)
```

```
{'C': 0.13}
```

Лучшая модель

In [137…
```
best_svm_model = grid_cv.best_estimator_
best_svm_model.fit(x_train_scaled, y_train)
y_pred_svm = best_svm_model.predict(x_test_scaled)
print_metrics(y_test, y_pred_svm)
```

```
weighted precision: 1.0
weighted recall: 1.0
weighted f1-score: 1.0
```

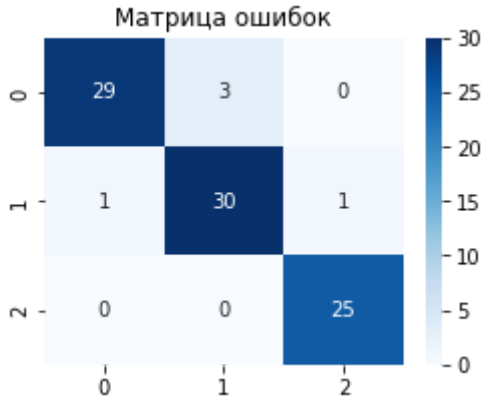


RandomForestClassifier

Базовая модель

In [130…
```
rfc_model = RandomForestClassifier()
rfc_model.fit(x_train, y_train)
y_pred_rfc = rfc_model.predict(x_test)
print_metrics(y_test, y_pred_rfc)
```

```
weighted precision: 0.9445247636258872
weighted recall: 0.9438202247191011
weighted f1-score: 0.9436376329325608
```



Подбор гиперпараметров

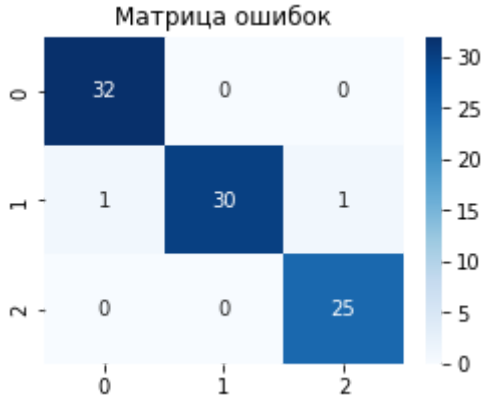
In [131…
```
params = {'n_estimators': [5, 10, 50, 100], 'max_features': [2, 3, 4], 'cri
grid_cv = GridSearchCV(estimator=rfc_model, param_grid=params, cv=10, n_job
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'criterion': 'gini', 'max_features': 2, 'min_samples_leaf': 1, 'n_estimato
rs': 100}
```

Лучшая модель

In [132…
```
best_rfc_model = grid_cv.best_estimator_
best_rfc_model.fit(x_train, y_train)
y_pred_rfc = best_rfc_model.predict(x_test)
print_metrics(y_test, y_pred_rfc)
```

```
weighted precision: 0.9783007254917369
weighted recall: 0.9775280898876404
weighted f1-score: 0.9773622265701662
```



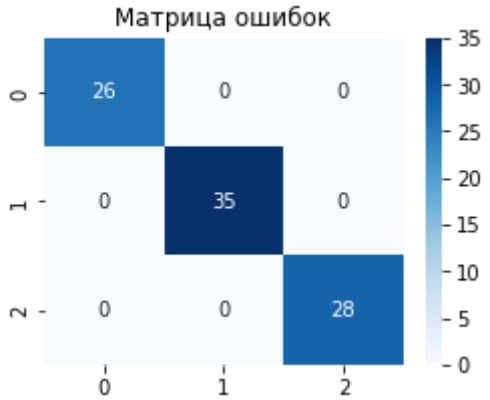
Сравнение результатов

In [138…

```
print("SVC result\n")
print_metrics(y_test, y_pred_svm)
```

```
SVC result

weighted precision: 1.0
weighted recall: 1.0
weighted f1-score: 1.0
```

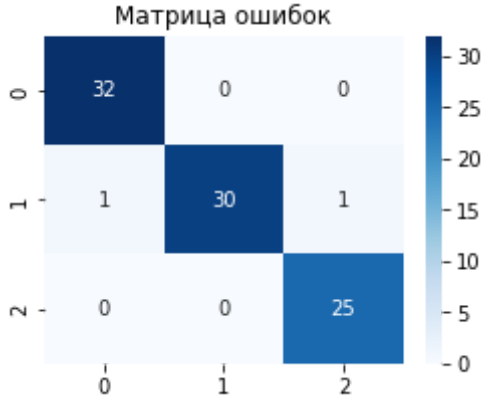


In [134…

```
print("RandomForestClassifier result\n")
print_metrics(y_test, y_pred_rfc)
```

```
RandomForestClassifier result

weighted precision: 0.9783007254917369
weighted recall: 0.9775280898876404
weighted f1-score: 0.9773622265701662
```



Выводы:

Модели с подобранными гиперпараметрами оказались лучше базовых моделей. Обе конечные модели показали очень высокую точность прогноза, что объясняется спецификой используемого "игрушечного" датасета. Из матриц ошибок видим, что модель RandomForestClassifier совершила 2 неверных прогноза из 89. Метрики показывают, что качество рассматриваемой модели RandomForestClassifier хуже SVC, что обосновывается размером тестовой выборки данных.