РК2 Перова Анна ИУ5-62Б

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.

Методы: опорных векторов и случайный лес

Перова А.Е.

ИУ5-62Б Вариант №16

Импортируем библиотеки:

```
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, f1_score, confusion_matrix, classification_r
```

```
data = pd.read_csv('restaurant-scores-lives-standard.csv', sep=",")
```

```
data.head()
```

	business_id	business_name	business_address	business_city	business_state	business_postal_code	business_latitude	business_longitude	business_location
0	101192	Cochinita #2	2 Marina Blvd Fort Mason	San Francisco	CA	NaN	NaN	NaN	NaN
1	97975	BREADBELLY	1408 Clement St	San Francisco	CA	94118	NaN	NaN	NaN
2	92982	Great Gold Restaurant	3161 24th St.	San Francisco	CA	94110	NaN	NaN	NaN
3	101389	HOMAGE	214 CALIFORNIA ST	San Francisco	CA	94111	NaN	NaN	NaN
4	85986	Pronto Pizza	798 Eddy St	San Francisco	CA	94109	NaN	NaN	NaN

5 rows × 23 columns

Обработка пропусков

```
data.isnull().sum()
```

```
business_id                       0
business_name                     0
business_address                  0
business_city                     0
business_state                    0
business_postal_code           1018
business_latitude             19556
business_longitude            19556
business_location             19556
business_phone_number         36938
inspection_id                     0
inspection_date                   0
inspection_score              13610
inspection_type                   0
violation_id                  12870
violation_description         12870
risk_category                 12870
Neighborhoods (old)           19594
Police Districts              19594
Supervisor Districts          19594
Fire Prevention Districts     19646
Zip Codes                     19576
Analysis Neighborhoods        19594
dtype: int64
```

```
data.shape
```

```
(53973, 23)
```

```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 53973
```

```
data = data.dropna(axis=0, how='any')
data.shape
```

Снимок экрана

```
data.head()
```

	business_id	business_name	business_address	business_city	business_state	business_postal_code	business_latitude	business_longitude	business_locatior
11	4794	VICTOR'S	210 TOWNSEND St	San Francisco	CA	94107	37.778634	-122.393089	{'type': 'Point' 'coordinates' [-122.393089,..
172	63652	SFDH - Banquet Main Kitchen	450 Powell St 2nd Floor	San Francisco	CA	94102	37.788918	-122.408507	{'type': 'Point' 'coordinates' [-122.408507,..
327	328	Miyako	1470 Fillmore St	San Francisco	CA	94115	37.783017	-122.432584	{'type': 'Point' 'coordinates' [-122.432584,..
372	2684	ERIC'S RESTAURANT	1500 Church St	San Francisco	CA	94131	37.746759	-122.426995	{'type': 'Point' 'coordinates' [-122.426995,..
397	328	Miyako	1470 Fillmore St	San Francisco	CA	94115	37.783017	-122.432584	{'type': 'Point' 'coordinates' [-122.432584,..

5 rows × 23 columns

Кодируем категориальные признаки

Удалим колонки, которые не влияют на целевой признак:

```
data = data.drop(columns='business_name')
data = data.drop(columns='business_address')
data = data.drop(columns='business_city')
data = data.drop(columns='business_state')
data = data.drop(columns='business_location')
```

Снимок экрана

```
data = data.drop(columns='business_phone_number')
data = data.drop(columns='violation_description')
```

```
data.shape
```

```
(6566, 16)
```

```
data.head()
```

	business_id	business_postal_code	business_latitude	business_longitude	inspection_id	inspection_date	inspection_score	inspection_type	vi
11	4794	94107	37.778634	-122.393089	4794_20181030	2018-10-30T00:00:00.000	71.0	Routine - Unscheduled	4794_201810
172	63652	94102	37.788918	-122.408507	63652_20190904	2019-09-04T00:00:00.000	94.0	Routine - Unscheduled	63652_201909
327	328	94115	37.783017	-122.432584	328_20161122	2016-11-22T00:00:00.000	81.0	Routine - Unscheduled	328_201611
372	2684	94131	37.746759	-122.426995	2684_20190715	2019-07-15T00:00:00.000	87.0	Routine - Unscheduled	2684_201907
397	328	94115	37.783017	-122.432584	328_20161122	2016-11-22T00:00:00.000	81.0	Routine - Unscheduled	328_201611

```
data.dtypes
```

```
business_id                    int64
business_postal_code          object
business_latitude            float64
business_longitude           float64
inspection_id                 object
inspection_date               object
inspection_score             float64
inspection_type               object
violation_id                  object
risk_category                 object
Neighborhoods (old)          float64
Police Districts             float64
Supervisor Districts         float64
Fire Prevention Districts    float64
Zip Codes                    float64
Analysis Neighborhoods       float64
dtype: object
```

Снимок экрана

Кодируем категориальные признаки:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
df_int = le.fit_transform(data['business_postal_code'])
data['business_postal_code'] = df_int
df_int = le.fit_transform(data['inspection_id'])
data['inspection_id'] = df_int
df_int = le.fit_transform(data['inspection_date'])
data['inspection_date'] = df_int
df_int = le.fit_transform(data['inspection_type'])
data['inspection_type'] = df_int
df_int = le.fit_transform(data['violation_id'])
data['violation_id'] = df_int
df_int = le.fit_transform(data['risk_category'])
data['risk_category'] = df_int
data.head()
```

	business_id	business_postal_code	business_latitude	business_longitude	inspection_id	inspection_date	inspection_score	inspection_type	violation_id	risk
11	4794	5	37.778634	-122.393089	829	440	71.0	0	2734	
172	63652	1	37.788918	-122.408507	1335	604	94.0	0	3895	
327	328	13	37.783017	-122.432584	564	28	81.0	0	1925	
372	2684	22	37.746759	-122.426995	405	576	87.0	0	1406	
397	328	13	37.783017	-122.432584	564	28	81.0	0	1929	

Масштабируем числовые данные:

```
scl = MinMaxScaler()
data['business_id'] = scl.fit_transform(data[['business_id']])
data['business_latitude'] = scl.fit_transform(data[['business_latitude']])
data['business_longitude'] = scl.fit_transform(data[['business_longitude']])
data['inspection_score'] = scl.fit_transform(data[['inspection_score']])
data['Neighborhoods (old)'] = scl.fit_transform(data[['Neighborhoods (old)']])
data['Police Districts'] = scl.fit_transform(data[['Police Districts']])
data['Supervisor Districts'] = scl.fit_transform(data[['Supervisor Districts']])
data['Fire Prevention Districts'] = scl.fit_transform(data[['Fire Prevention Districts']])
data['Zip Codes'] = scl.fit_transform(data[['Zip Codes']])
data['Analysis Neighborhoods'] = scl.fit_transform(data[['Analysis Neighborhoods']])
data.head()
```

	business_id	business_postal_code	business_latitude	business_longitude	inspection_id	inspection_date	inspection_score	inspection_type	violation_id	risk
11	0.065966	5	0.700222	0.903650	829	440	0.480769	0	2734	
172	0.885088	1	0.803468	0.784522	1335	604	0.923077	0	3895	
327	0.003813	13	0.744225	0.598490	564	28	0.673077	0	1925	
372	0.036601	22	0.380214	0.641674	405	[illegible]	[illegible]	0	1406	

Делим выборку на обучающую и тестовую

```
x_train, x_test, y_train, y_test = train_test_split(data.drop(['risk_category'], axis=1), data['risk_category'], test_s
```

Масштабирование данных

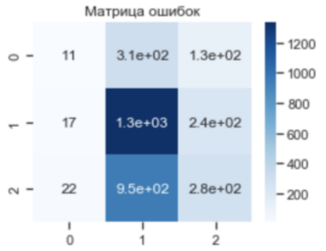
```
scaler = StandardScaler().fit(x_train)
x_train_scaled = pd.DataFrame(scaler.transform(x_train), columns=x_train.columns)
x_test_scaled = pd.DataFrame(scaler.transform(x_test), columns=x_train.columns)
x_train_scaled.describe()
```

	business_id	business_postal_code	business_latitude	business_longitude	inspection_id	inspection_date	inspection_score	inspection_type	violation_
count	3.283000e+03	3.283000e+03	3.283000e+03	3.283000e+03	3.283000e+03	3.283000e+03	3.283000e+03	3283.0	3.283000e+
mean	-5.059073e-17	-3.151775e-17	-1.812609e-17	4.240693e-16	-4.490308e-17	-8.724872e-18	-3.872423e-16	0.0	-1.615200e-
std	1.000152e+00	1.000152e+00	1.000152e+00	1.000152e+00	1.000152e+00	1.000152e+00	1.000152e+00	0.0	1.000152e+
min	-7.166362e-01	-1.660029e+00	-2.307897e+00	-3.256321e+00	-1.604104e+00	-1.768765e+00	-4.811108e+00	0.0	-1.756056e+
25%	-6.191142e-01	-6.446684e-01	-6.091462e-01	-3.247809e-01	-8.625846e-01	-7.637860e-01	-5.953532e-01	0.0	-8.657228e-
50%	-5.214277e-01	-6.446242e-02	5.839182e-02	3.181117e-01	-1.272187e-01	-1.005139e-02	2.477978e-01	0.0	-7.923122e-
75%	1.858585e-01	8.058465e-01	8.361879e-01	5.994102e-01	9.537794e-01	8.978562e-01	7.295984e-01	0.0	8.607211e-
max	2.240713e+00	2.111310e+00	1.893510e+00	2.000217e+00	1.784527e+00	1.788633e+00	1.452299e+00	0.0	1.737318e+

Метод опорных векторов

```
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: 0.44530918329069746
weighted recall: 0.4952787084983247
weighted f1-score: 0.42810029971890773
```



Подбор гиперпараметров

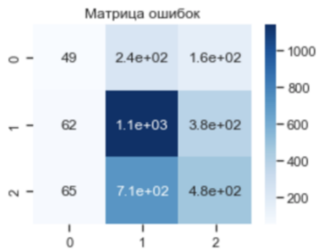
```
params = {'C': np.concatenate([np.arange(0.1, 2, 0.03), np.arange(2, 20, 1)])}
grid_cv = GridSearchCV(estimator=svm_model, param_grid=params, cv=10, n_jobs=-1, scoring='f1_macro')
grid_cv.fit(x_train_scaled, y_train)
print(grid_cv.best_params_)
```

```
{'C': 17.0}
```

Лучшая модель

```
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: 0.47934527484451267
weighted recall: 0.5068534876637222
weighted f1-score: 0.48081097719528215
```

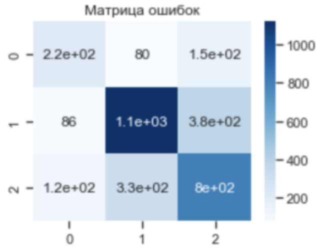


Случайный лес

```
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");
```

```
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: 0.6542897441379976
weighted recall: 0.652452025586354
weighted f1-score: 0.6529870082268522
```



Подбор гиперпараметров

```
params = {'n_estimators': [5, 10, 50, 100], 'max_features': [2, 3, 4], 'criterion': ['gini', 'entropy'], 'min_samples_l
grid_cv = GridSearchCV(estimator=rfc_model, param_grid=params, cv=10, n_jobs=-1, scoring='f1_weighted')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

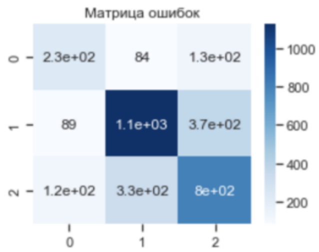
```
{'criterion': 'entropy', 'max_features': 4, 'min_samples_leaf': 1, 'n_estimators': 100}
```

Лучшая модель

```
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)
```

Снимок экрана

```
weighted precision: 0.6586578132508267
weighted recall: 0.6567164179104478
weighted f1-score: 0.6574876194933115
```



Сравнение результатов

Модели с подобранными гиперпараметрами оказались лучше базовых моделей. Обе конечные модели показали довольно высокую точность прогноза. Метрики показывают, что точность модели случайный лес гораздо выше модели опорных векторов.