Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №6

Ансамбли моделей машинного обучения

Группа: РТ5-61

Студент: Савушкин Д. А.

Преподаватель: Гапанюк Ю. Е.

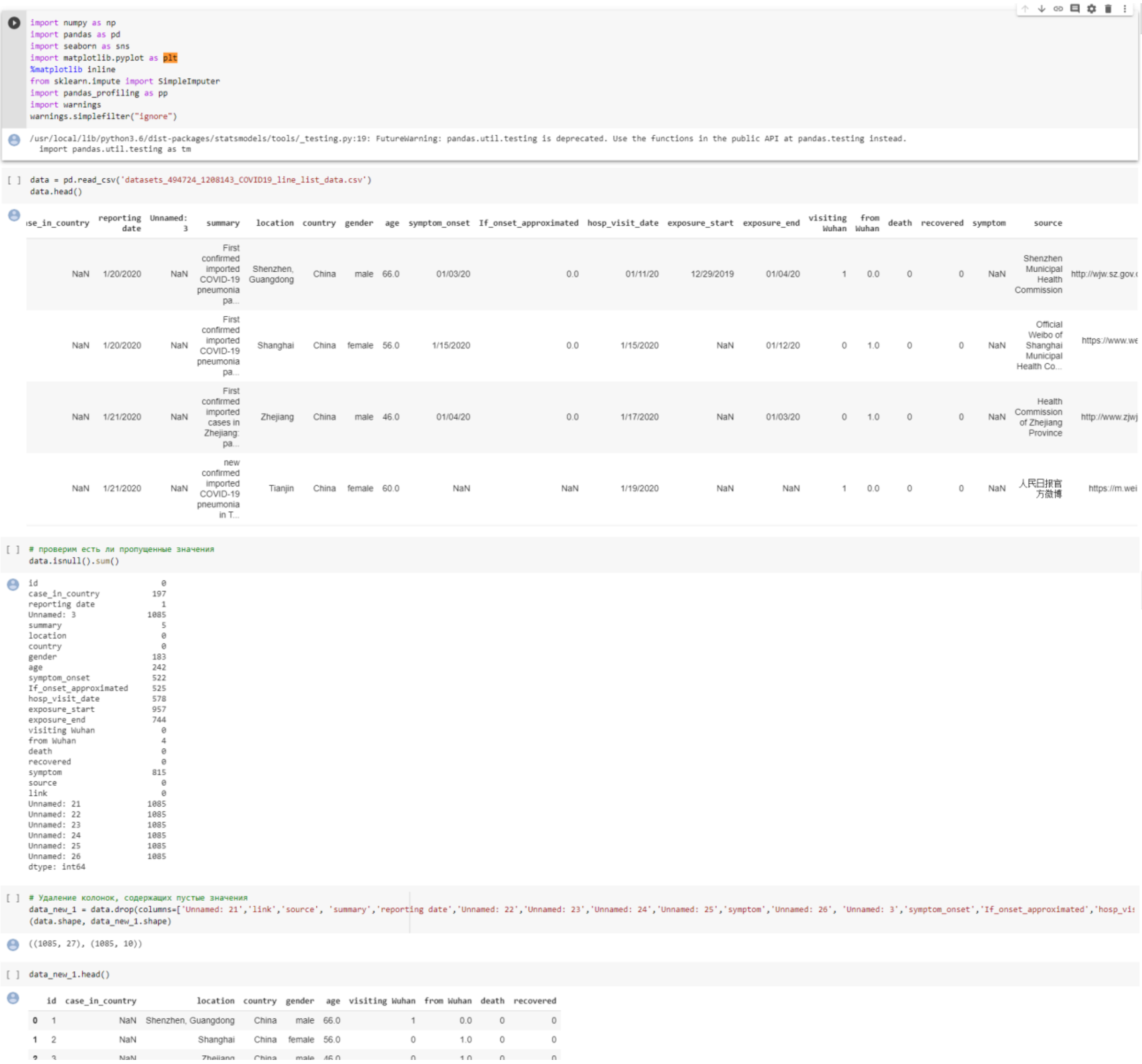
Москва, 2020 г.

Цель лабораторной работы: изучение ансамблей моделей машинного обучения.

Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Текст программы и экранные формы с примерами выполнения программы:



```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.impute import SimpleImputer
import pandas_profiling as pp
import warnings
warnings.simplefilter("ignore")
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

```
[ ] data = pd.read_csv('datasets_494724_1208143_COVID19_line_list_data.csv')
    data.head()
```

se_in_country	reporting date	Unnamed: 3	summary	location	country	gender	age	symptom_onset	If_onset_approximated	hosp_visit_date	exposure_start	exposure_end	visiting Wuhan	from Wuhan	death	recovered	symptom	source	
NaN	1/20/2020	NaN	First confirmed imported COVID-19 pneumonia pa...	Shenzhen, Guangdong	China	male	66.0	01/03/20	0.0	01/11/20	12/29/2019	01/04/20	1	0.0	0	0	NaN	Shenzhen Municipal Health Commission	http://wjw.sz.gov.c
NaN	1/20/2020	NaN	First confirmed imported COVID-19 pneumonia pa...	Shanghai	China	female	56.0	1/15/2020	0.0	1/15/2020	NaN	01/12/20	0	1.0	0	0	NaN	Official Weibo of Shanghai Municipal Health Co...	https://www.we
NaN	1/21/2020	NaN	First confirmed imported cases in Zhejiang: pa...	Zhejiang	China	male	46.0	01/04/20	0.0	1/17/2020	NaN	01/03/20	0	1.0	0	0	NaN	Health Commission of Zhejiang Province	http://www.zjwj
NaN	1/21/2020	NaN	new confirmed imported COVID-19 pneumonia in T...	Tianjin	China	female	60.0	NaN	NaN	1/19/2020	NaN	NaN	1	0.0	0	0	NaN	人民日报官方微博	https://m.wei

```
[ ] # проверим есть ли пропущенные значения
    data.isnull().sum()
```

```
id                          0
case_in_country           197
reporting date              1
Unnamed: 3               1085
summary                     5
location                    0
country                     0
gender                    183
age                       242
symptom_onset             522
If_onset_approximated     525
hosp_visit_date           578
exposure_start            957
exposure_end              744
visiting Wuhan              0
from Wuhan                  4
death                       0
recovered                   0
symptom                   815
source                      0
link                        0
Unnamed: 21              1085
Unnamed: 22              1085
Unnamed: 23              1085
Unnamed: 24              1085
Unnamed: 25              1085
Unnamed: 26              1085
dtype: int64
```

```
[ ] # Удаление колонок, содержащих пустые значения
    data_new_1 = data.drop(columns=['Unnamed: 21','link','source', 'summary','reporting date','Unnamed: 22','Unnamed: 23','Unnamed: 24','Unnamed: 25','symptom','Unnamed: 26', 'Unnamed: 3','symptom_onset','If_onset_approximated','hosp_vis
    (data.shape, data_new_1.shape)
```

```
((1085, 27), (1085, 10))
```

```
[ ] data_new_1.head()
```

	id	case_in_country	location	country	gender	age	visiting Wuhan	from Wuhan	death	recovered
0	1	NaN	Shenzhen, Guangdong	China	male	66.0	1	0.0	0	0
1	2	NaN	Shanghai	China	female	56.0	0	1.0	0	0
2	3	NaN	Zhejiang	China	male	46.0	0	1.0	0	0

```
[ ] data_new_1.isnull().sum()
```

```
id                  0
case_in_country   197
location            0
country             0
gender            183
age               242
visiting Wuhan      0
from Wuhan          4
death               0
recovered           0
dtype: int64
```

```
[ ] # Удаление строки, содержащих пустые значения
    data_new_2 = data_new_1.dropna(axis=0,how='any',subset=['case_in_country','gender', 'age'])
    (data_new_1.shape, data_new_2.shape)
```

```
((1085, 10), (635, 10))
```

```
[ ] data_new_2.head()
```

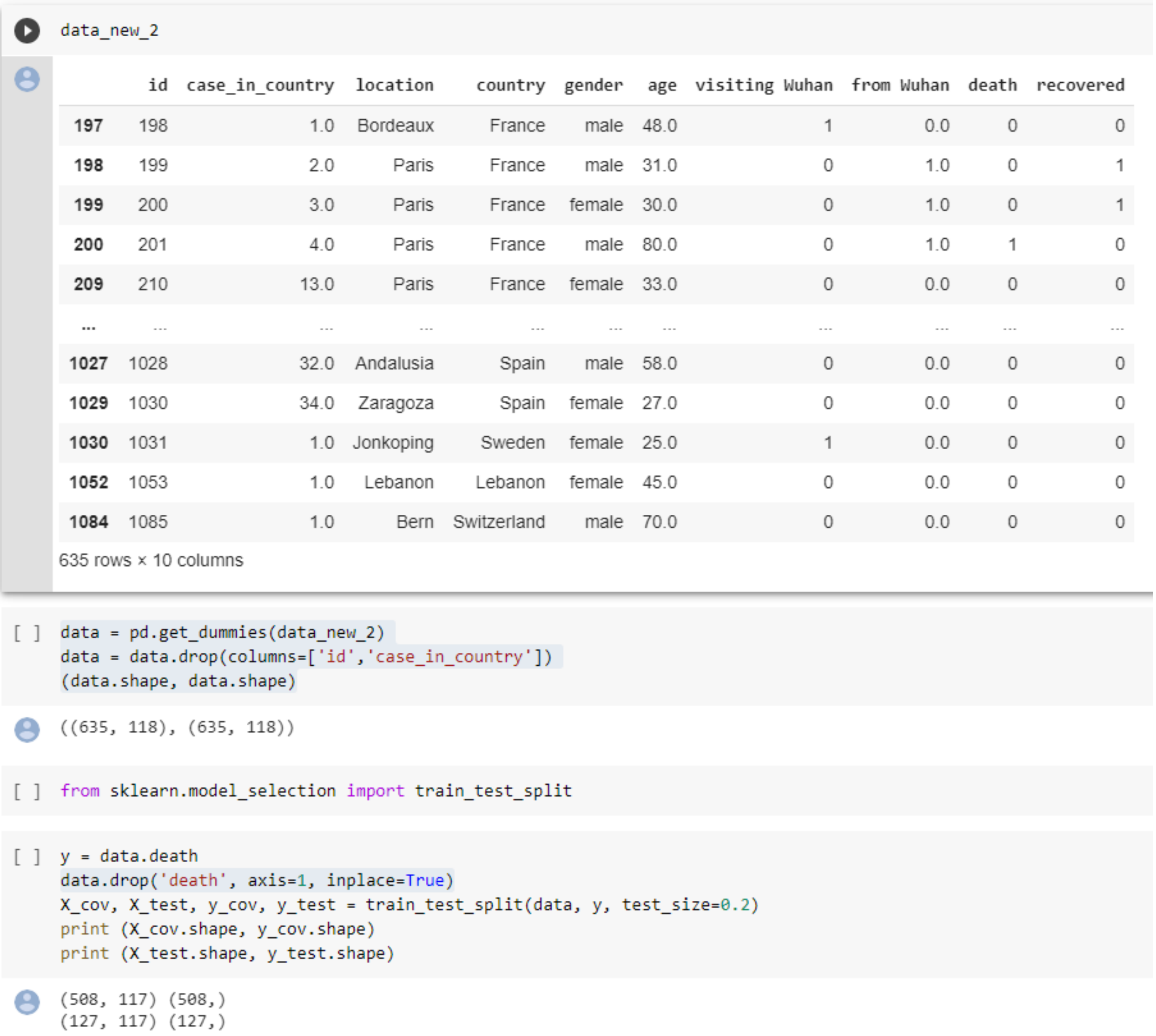
	id	case_in_country	location	country	gender	age	visiting Wuhan	from Wuhan	death	recovered
197	198	1.0	Bordeaux	France	male	48.0	1	0.0	0	0
198	199	2.0	Paris	France	male	31.0	0	1.0	0	02/12/20
199	200	3.0	Paris	France	female	30.0	0	1.0	0	02/12/20
200	201	4.0	Paris	France	male	80.0	0	1.0	2/14/2020	0
209	210	13.0	Paris	France	female	33.0	0	0.0	0	0

```
[ ] data_new_2.shape[0]
```

```
635
```

```
[ ] data_new_2['death'] = data_new_2['death'].apply(lambda x: 0 if x=='0' else 1)
```

```
[ ] data_new_2['recovered'] = data_new_2['recovered'].apply(lambda x: 0 if x=='0' else 1)
```



```
data_new_2
```

	id	case_in_country	location	country	gender	age	visiting Wuhan	from Wuhan	death	recovered
197	198	1.0	Bordeaux	France	male	48.0	1	0.0	0	0
198	199	2.0	Paris	France	male	31.0	0	1.0	0	1
199	200	3.0	Paris	France	female	30.0	0	1.0	0	1
200	201	4.0	Paris	France	male	80.0	0	1.0	1	0
209	210	13.0	Paris	France	female	33.0	0	0.0	0	0
...	...	...	...	...	...	...	...	...	...	...
1027	1028	32.0	Andalusia	Spain	male	58.0	0	0.0	0	0
1029	1030	34.0	Zaragoza	Spain	female	27.0	0	0.0	0	0
1030	1031	1.0	Jonkoping	Sweden	female	25.0	1	0.0	0	0
1052	1053	1.0	Lebanon	Lebanon	female	45.0	0	0.0	0	0
1084	1085	1.0	Bern	Switzerland	male	70.0	0	0.0	0	0

635 rows × 10 columns

[ ]

```
data = pd.get_dummies(data_new_2)
data = data.drop(columns=['id','case_in_country'])
(data.shape, data.shape)
```

```
((635, 118), (635, 118))
```

[ ]

```
from sklearn.model_selection import train_test_split
```

[ ]

```
y = data.death
data.drop('death', axis=1, inplace=True)
X_cov, X_test, y_cov, y_test = train_test_split(data, y, test_size=0.2)
print (X_cov.shape, y_cov.shape)
print (X_test.shape, y_test.shape)
```

```
(508, 117) (508,)
(127, 117) (127,)
```

Обучение модели

```
[ ] from sklearn.neighbors import KNeighborsClassifier
```

```
[ ] KNeighborsClassifierObj = KNeighborsClassifier(n_neighbors=10)
```

```
[ ] KNeighborsClassifierObj.fit(X_cov, y_cov)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                     weights='uniform')
```

```
[ ] y_predicted = KNeighborsClassifierObj.predict(X_test)
```

Метрика качества

```
[ ] from sklearn.metrics import accuracy_score, balanced_accuracy_score, precision_score, f1_score, classification_report
```

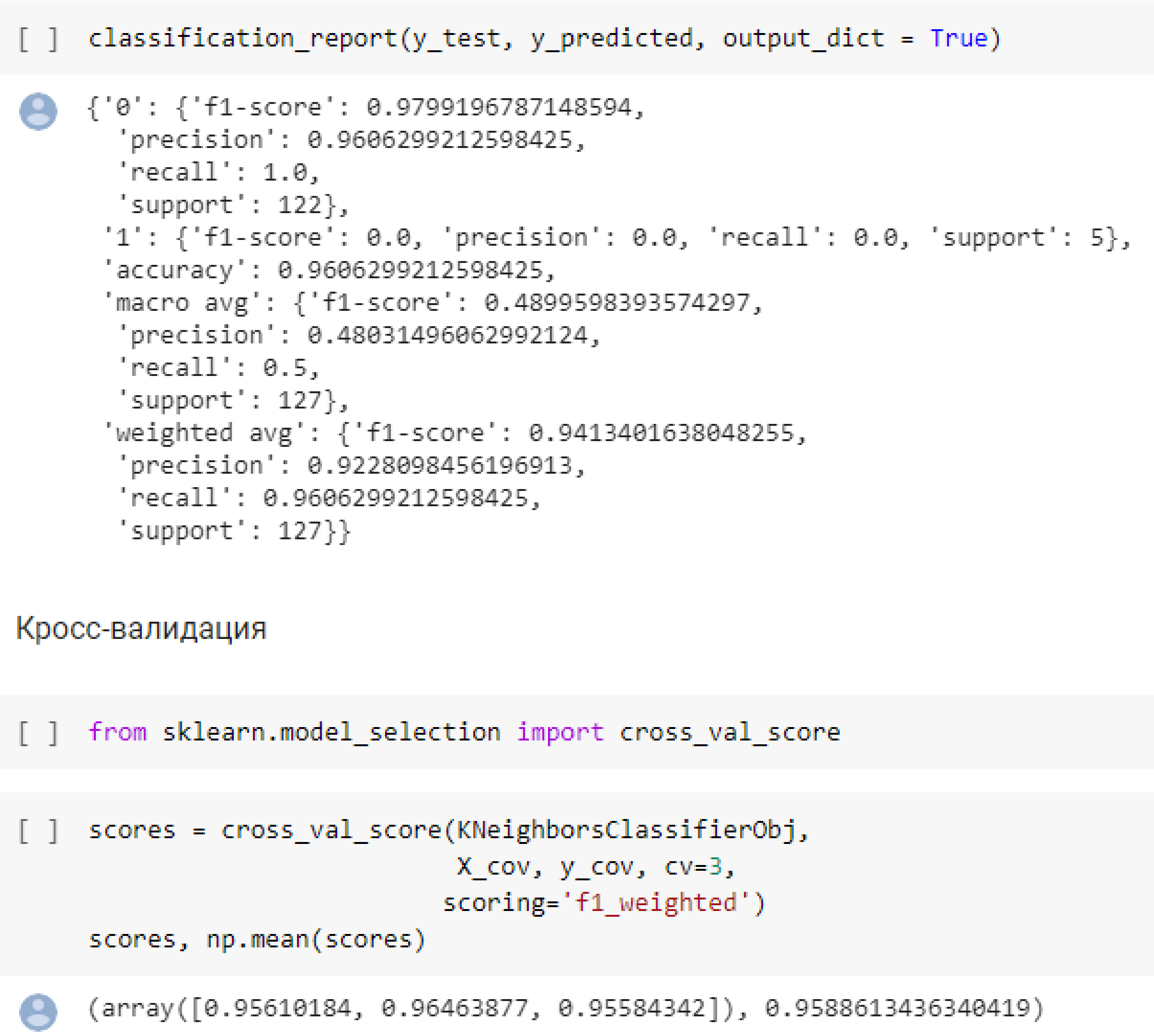
```
[ ] accuracy_score(y_test, y_predicted)
```

```
0.9606299212598425
```

```
[ ] precision_score(y_test, y_predicted)
```

```
0.0
```

Смертность не зависит от других параметров.



```
[ ]  classification_report(y_test, y_predicted, output_dict = True)
```

```
{'0': {'f1-score': 0.9799196787148594,
   'precision': 0.9606299212598425,
   'recall': 1.0,
   'support': 122},
  '1': {'f1-score': 0.0, 'precision': 0.0, 'recall': 0.0, 'support': 5},
  'accuracy': 0.9606299212598425,
  'macro avg': {'f1-score': 0.4899598393574297,
   'precision': 0.48031496062992124,
   'recall': 0.5,
   'support': 127},
  'weighted avg': {'f1-score': 0.9413401638048255,
   'precision': 0.9228098456196913,
   'recall': 0.9606299212598425,
   'support': 127}}
```

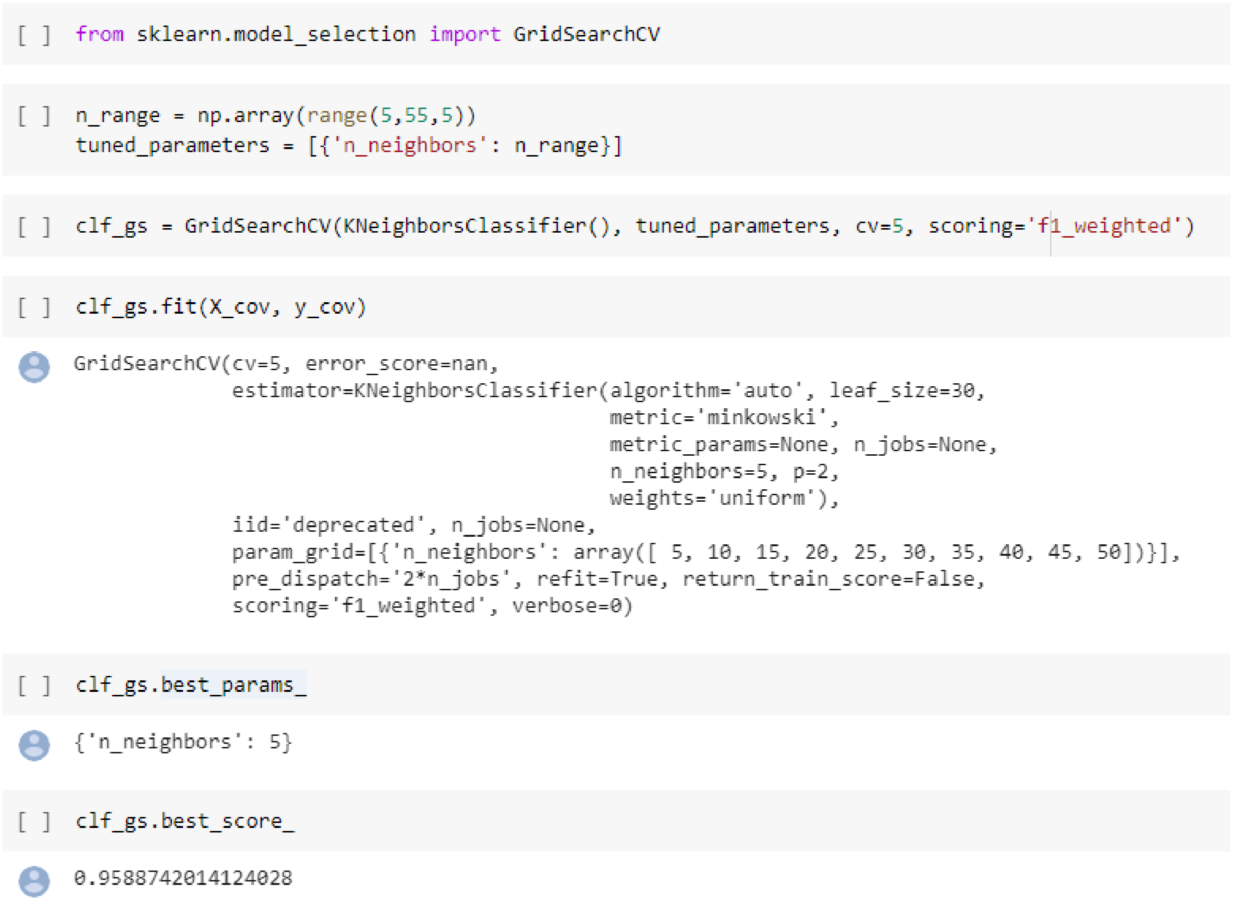
Кросс-валидация

```
[ ]  from sklearn.model_selection import cross_val_score
```

```
[ ]  scores = cross_val_score(KNeighborsClassifierObj,
                             X_cov, y_cov, cv=3,
                            scoring='f1_weighted')
     scores, np.mean(scores)
```

```
(array([0.95610184, 0.96463877, 0.95584342]), 0.9588613436340419)
```

Подбор гиперпараметров



```
[ ]  from sklearn.model_selection import GridSearchCV
```

```
[ ]  n_range = np.array(range(5,55,5))
     tuned_parameters = [{'n_neighbors': n_range}]
```

```
[ ]  clf_gs = GridSearchCV(KNeighborsClassifier(), tuned_parameters, cv=5, scoring='f1_weighted')
```

```
[ ]  clf_gs.fit(X_cov, y_cov)
```

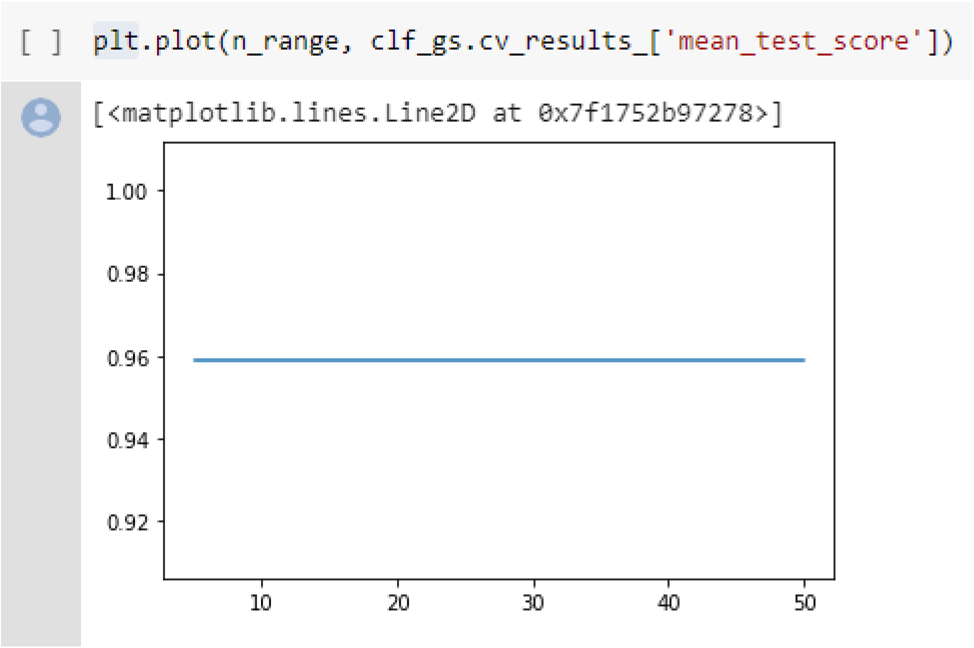
```
GridSearchCV(cv=5, error_score=nan,
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=None,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='deprecated', n_jobs=None,
             param_grid=[{'n_neighbors': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50])}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='f1_weighted', verbose=0)
```

```
[ ]  clf_gs.best_params_
```

```
{'n_neighbors': 5}
```

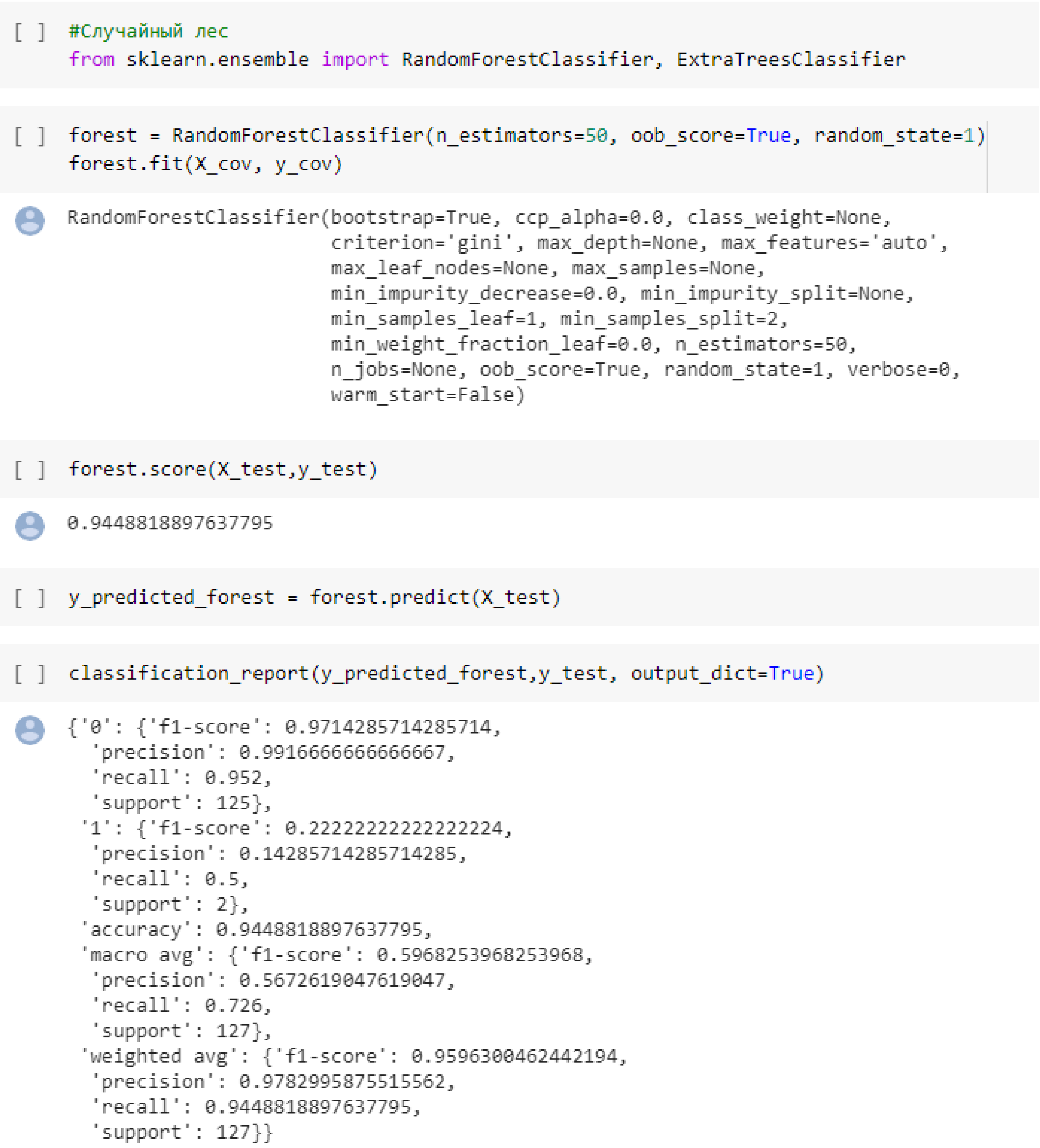
```
[ ]  clf_gs.best_score_
```

```
0.9588742014124028
```



```
[ ]  plt.plot(n_range, clf_gs.cv_results_['mean_test_score'])
```

```
[<matplotlib.lines.Line2D at 0x7f1752b97278>]
```



```
#Случайный лес
from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier
```

```
forest = RandomForestClassifier(n_estimators=50, oob_score=True, random_state=1)
forest.fit(X_cov, y_cov)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, 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=2,
                       min_weight_fraction_leaf=0.0, n_estimators=50,
                       n_jobs=None, oob_score=True, random_state=1, verbose=0,
                       warm_start=False)
```

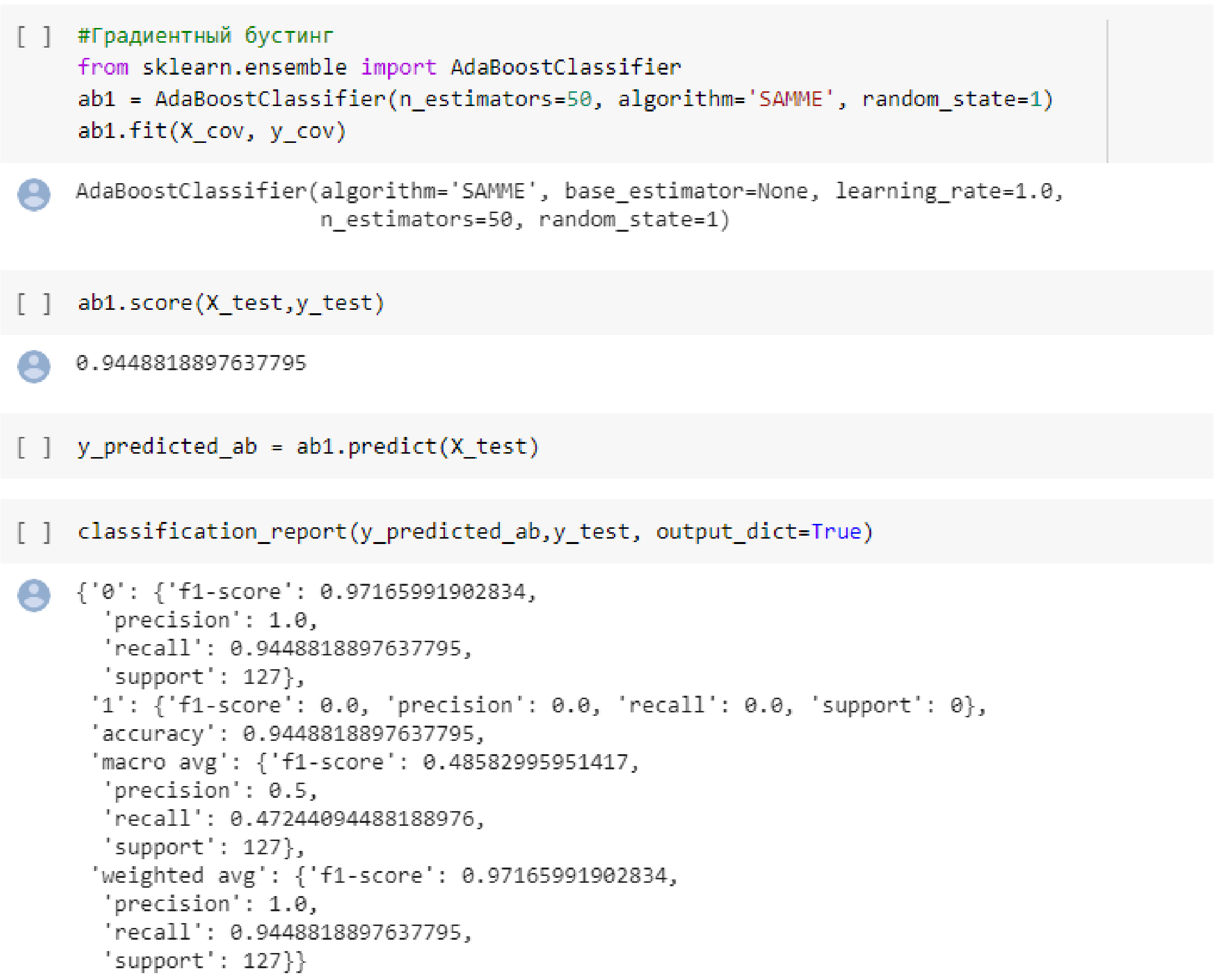
```
forest.score(X_test,y_test)
```

```
0.9448818897637795
```

```
y_predicted_forest = forest.predict(X_test)
```

```
classification_report(y_predicted_forest,y_test, output_dict=True)
```

```
{'0': {'f1-score': 0.9714285714285714,
  'precision': 0.9916666666666667,
  'recall': 0.952,
  'support': 125},
 '1': {'f1-score': 0.2222222222222224,
  'precision': 0.14285714285714285,
  'recall': 0.5,
  'support': 2},
 'accuracy': 0.9448818897637795,
 'macro avg': {'f1-score': 0.5968253968253968,
  'precision': 0.5672619047619047,
  'recall': 0.726,
  'support': 127},
 'weighted avg': {'f1-score': 0.9596300462442194,
  'precision': 0.9782995875515562,
  'recall': 0.9448818897637795,
  'support': 127}}
```



```
[ ] #Градиентный бустинг
    from sklearn.ensemble import AdaBoostClassifier
    ab1 = AdaBoostClassifier(n_estimators=50, algorithm='SAMME', random_state=1)
    ab1.fit(X_cov, y_cov)
```

```
AdaBoostClassifier(algorithm='SAMME', base_estimator=None, learning_rate=1.0,
                   n_estimators=50, random_state=1)
```

```
[ ] ab1.score(X_test,y_test)
```

```
0.9448818897637795
```

```
[ ] y_predicted_ab = ab1.predict(X_test)
```

```
[ ] classification_report(y_predicted_ab,y_test, output_dict=True)
```

```
{'0': {'f1-score': 0.97165991902834,
  'precision': 1.0,
  'recall': 0.9448818897637795,
  'support': 127},
 '1': {'f1-score': 0.0, 'precision': 0.0, 'recall': 0.0, 'support': 0},
 'accuracy': 0.9448818897637795,
 'macro avg': {'f1-score': 0.48582995951417,
  'precision': 0.5,
  'recall': 0.47244094488188976,
  'support': 127},
 'weighted avg': {'f1-score': 0.97165991902834,
  'precision': 1.0,
  'recall': 0.9448818897637795,
  'support': 127}}
```

Все модели дают одинаковые хорошие результаты по всем метрикам качества модели.