

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»

Факультет информационных систем и технологий
Кафедра «Измерительно-вычислительные комплексы»

Отчет по лабораторной работе №5

«Исследование инструментов классификации библиотеки Scikit-learn»

по дисциплине «Методы искусственного интеллекта»

Выполнила:
ст. гр. ИСТбд-41 Гильметдинова Е.Д.

Ульяновск

2022

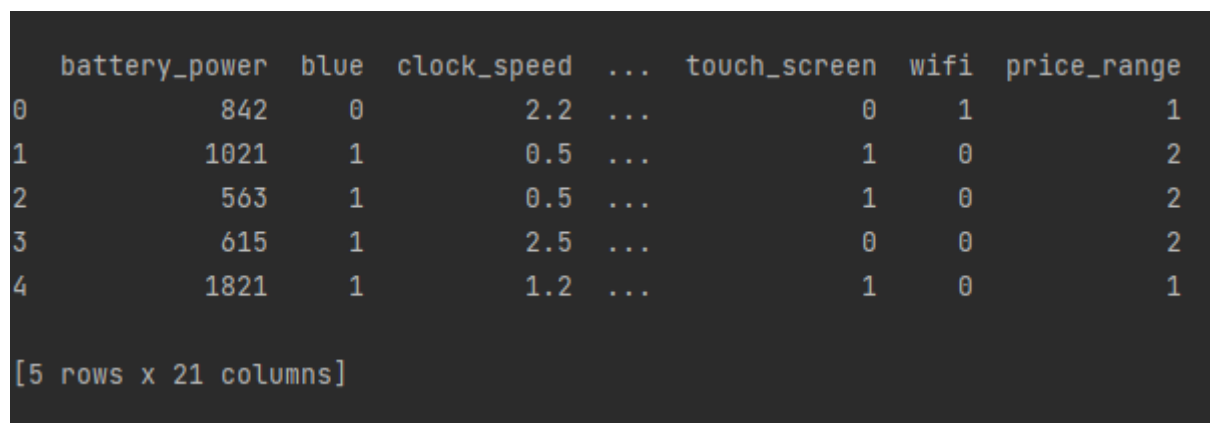
Задание:

1. Ознакомиться с классификаторами библиотеки Scikit-learn
2. Выбрать для исследования не менее 3 классификаторов

Выбраны классификаторы: Классификатор дерева решений, Линейный дискриминантный анализ, Метод опорных векторов, Метод ближайших соседей

3. Выбрать набор данных для задач классификации из открытых источников

Выбран dataset: <https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?select=train.csv>

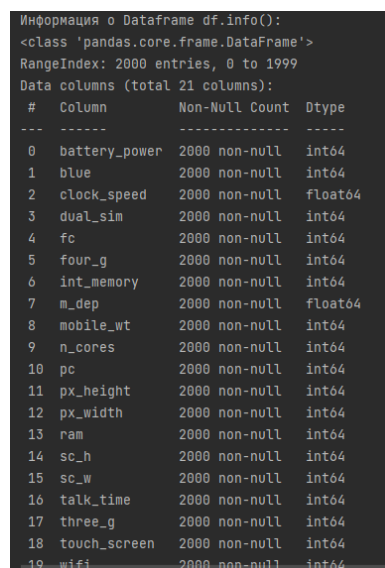


	battery_power	blue	clock_speed	...	touch_screen	wifi	price_range
0	842	0	2.2	...	0	1	1
1	1021	1	0.5	...	1	0	2
2	563	1	0.5	...	1	0	2
3	615	1	2.5	...	0	0	2
4	1821	1	1.2	...	1	0	1

[5 rows x 21 columns]

Датасет определяет стоимость телефона по его характеристикам. Цена принимает значения: 0 (низкая стоимость), 1 (средняя стоимость), 2 (высокая стоимость) и 3 (очень высокая стоимость).

4. Была выведена информация о датасете для проверки, что данные подготовлены:

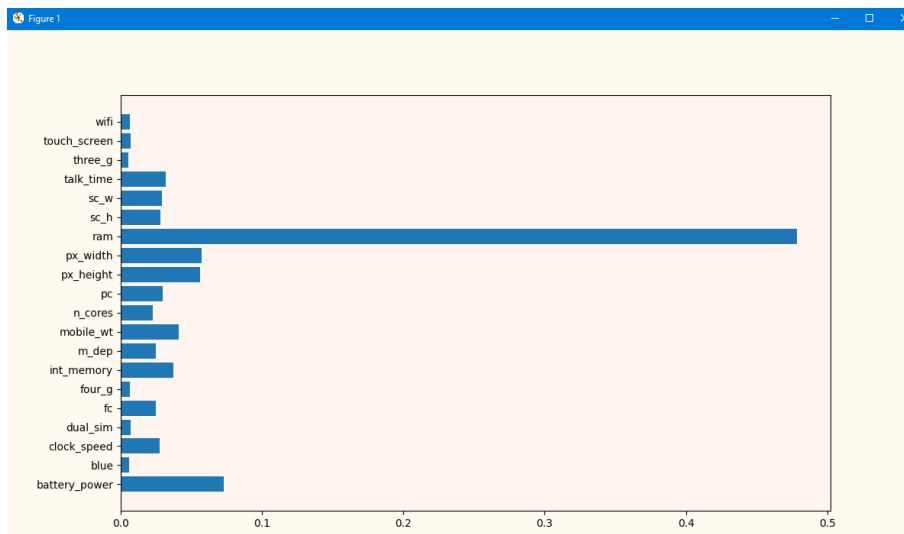


```
Информация о Dataframe df.info():
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   battery_power    2000 non-null   int64
1   blue             2000 non-null   int64
2   clock_speed      2000 non-null   float64
3   dual_sim         2000 non-null   int64
4   fc               2000 non-null   int64
5   four_g           2000 non-null   int64
6   int_memory       2000 non-null   int64
7   m_dep            2000 non-null   float64
8   mobile_wt        2000 non-null   int64
9   n_cores          2000 non-null   int64
10  pc               2000 non-null   int64
11  px_height        2000 non-null   int64
12  px_width         2000 non-null   int64
13  ram              2000 non-null   int64
14  sc_h             2000 non-null   int64
15  sc_w             2000 non-null   int64
16  talk_time        2000 non-null   int64
17  three_g          2000 non-null   int64
18  touch_screen     2000 non-null   int64
19  wifi             2000 non-null   int64
```

```
20 price_range    2000 non-null    int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
None
```

Пустых значений нет, предобработка не требуется.

5. Целевым столбцом для всех классификаторов будет `price_range`. Для определения ключевых признаков, имеющих наибольшее влияние, была построена диаграмма:



И выбраны 6 ключевых признаков: 'ram', 'battery_power', 'px_width', 'px_height', 'mobile_wt', 'int_memory'.

6. Данные были разделены на тестовые и тренировочные:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

И для каждого классификатора был создан его экземпляр и обучен:

```
SVC_model = svm.SVC()
KNN_model = KNeighborsClassifier(n_neighbors=5)
TREE_model = DecisionTreeClassifier(max_depth=12)
LOG_model = LinearDiscriminantAnalysis()

SVC_model.fit(X_train, y_train)
KNN_model.fit(X_train, y_train)
TREE_model.fit(X_train, y_train)
LOG_model.fit(X_train, y_train)

SVC_prediction = SVC_model.predict(X_test)
KNN_prediction = KNN_model.predict(X_test)
TREE_prediction = TREE_model.predict(X_test)
LOG_prediction = LOG_model.predict(X_test)
```

Также созданы классификационные отчеты:

```
++++KNN_prediction+++++
      precision    recall  f1-score   support

     0       0.95      0.93      0.94        101
     1       0.93      0.92      0.92        118
     2       0.86      0.94      0.90         88
     3       0.95      0.90      0.93         93

 accuracy          0.92        400
 macro avg       0.92      0.92      0.92        400
weighted avg       0.92      0.92      0.92        400

++++SVC_prediction+++++
      precision    recall  f1-score   support

     0       0.97      0.99      0.98         97
     1       0.96      0.94      0.95        118
     2       0.90      0.92      0.91         95
     3       0.95      0.93      0.94         90

 accuracy          0.94        400
 macro avg       0.94      0.94      0.94        400
weighted avg       0.95      0.94      0.95        400
```

```
++++TREE_prediction+++++
      precision    recall  f1-score   support

     0       0.87      0.92      0.90         93
     1       0.82      0.81      0.82        117
     2       0.69      0.71      0.70         95
     3       0.84      0.78      0.81         95

 accuracy          0.81        400
 macro avg       0.80      0.81      0.80        400
weighted avg       0.81      0.81      0.80        400

++++LOG_prediction+++++
      precision    recall  f1-score   support

     0       0.96      1.00      0.98         95
     1       0.93      0.92      0.93        117
     2       0.95      0.83      0.88        111
     3       0.88      1.00      0.93         77

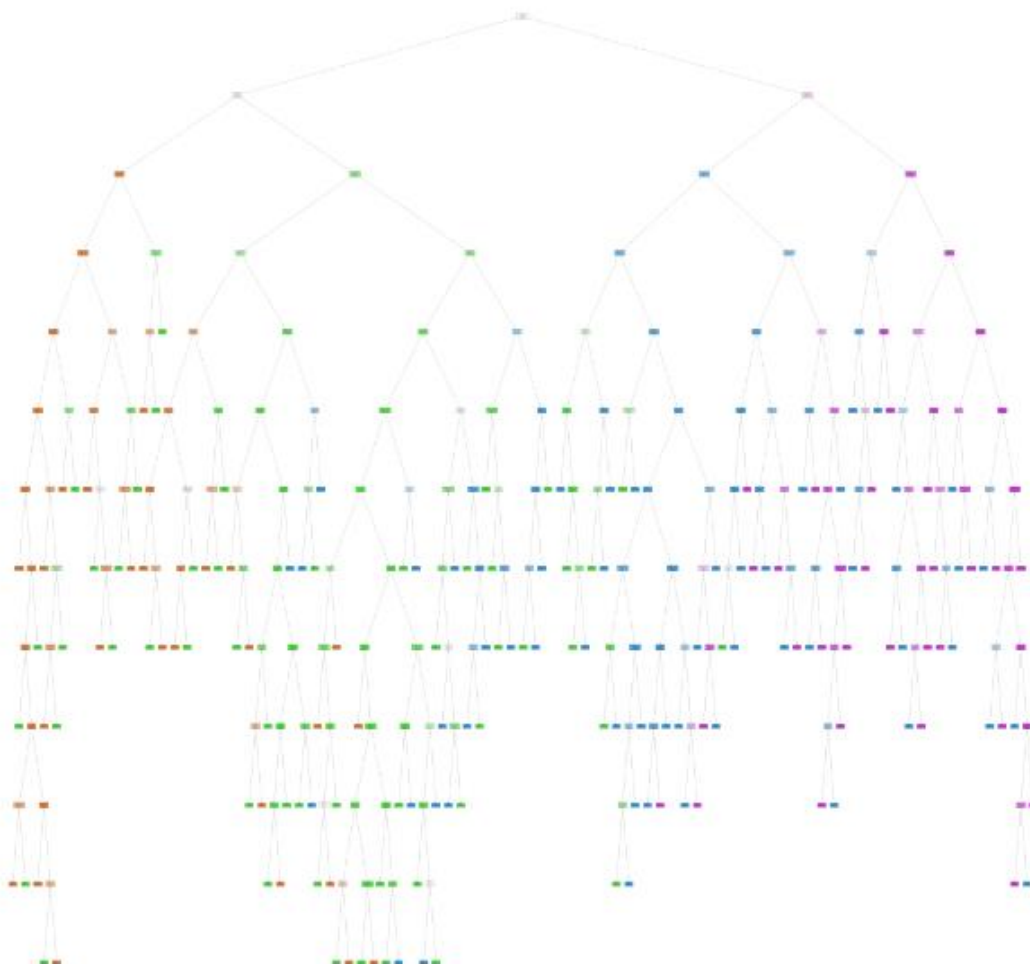
 accuracy          0.93        400
 macro avg       0.93      0.94      0.93        400
weighted avg       0.93      0.93      0.93        400
```

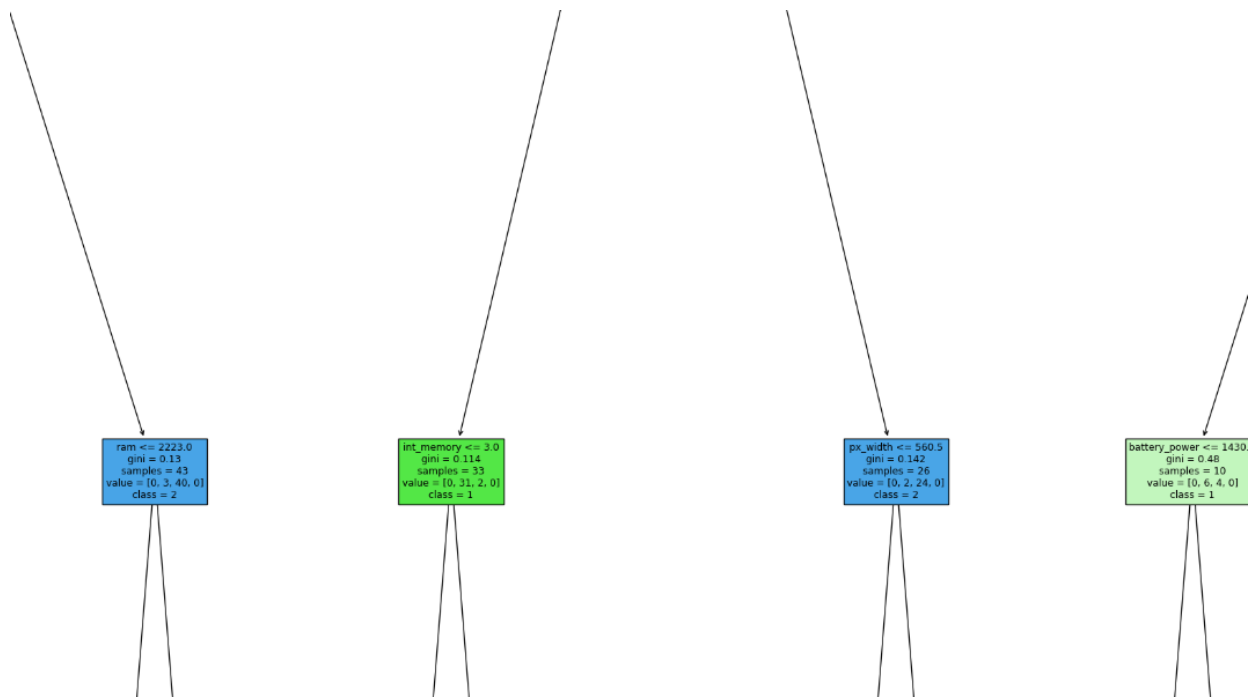
7. Для оценки точности используют функцию `accuracy_score`, которая возвращает точность подмножества. Если весь набор предсказанных меток для выборки строго соответствует истинному набору меток, то точность подмножества равна 1,0; в противном случае — 0,0.

Результаты представлены в табличной форме:

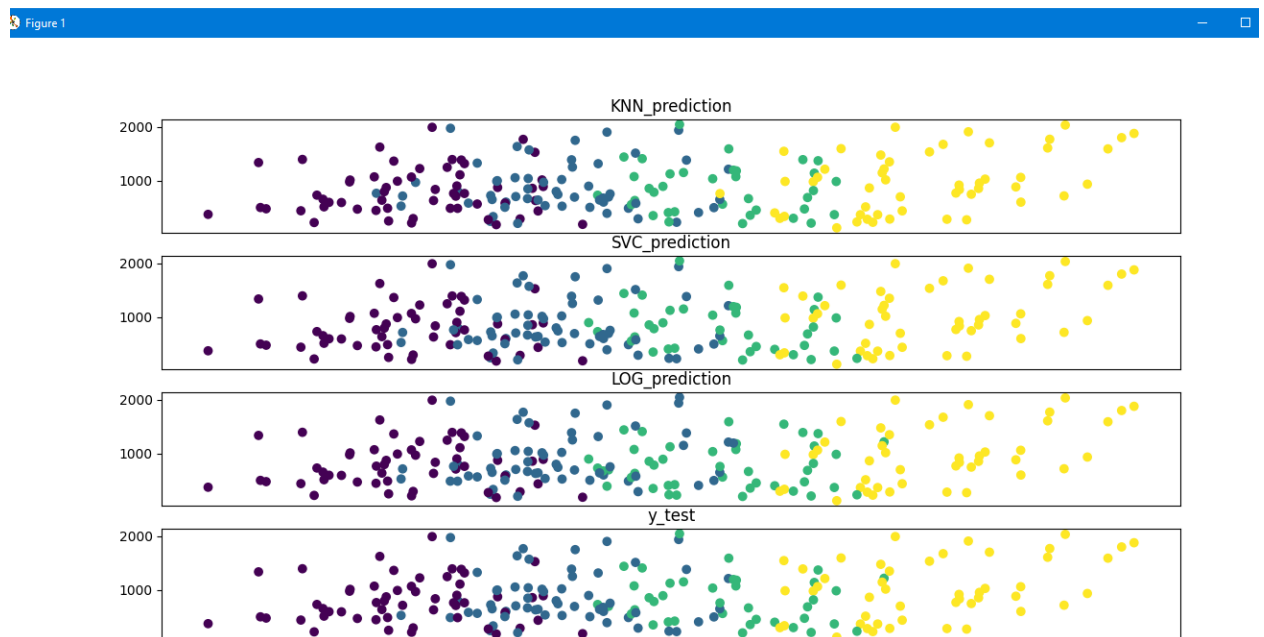
Классификатор	Accuracy
Классификатор дерева решений	0.805
Линейный дискриминантный анализ	0.93
Метод опорных векторов	0.945
Метод ближайших соседей	0.922

8. Также результаты были визуализированы. Классификатор дерева решений в виде дерева:





Другие в одном графике (также выведены реальные данные для сравнения):



Вывод: по результатам исследования наиболее точным оказался классификатор опорных векторов.