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»

Дисциплина: «Методы искусственного интеллекта»

Отчет

по лабораторной работе № 5

по теме: **«Исследование инструментов классификации библиотеки Scikit-learn»**

Выполнила:

студентка гр. ИСТбд-4

Тагашев И.

Проверил:

к.т.н., доцент

Шишкин В.В.

Ульяновск 2022 г.

Выполнение лабораторной работы по теме:
«Исследование инструментов классификации библиотеки Scikit-learn»

Классификаторы: классификация с помощью стохастического градиентного спуска, с помощью опорных векторов и метод случайного леса.

Датасет: «Covid19» https://www.kaggle.com/datasets/meirnizri/covid19-dataset

Данный датасет содержит сырые данные и состоит из следующих полей:

1. USMER - уровень подразделения медицинского учреждения (1, 2, 3)
2. MEDICAL_UNIT - тип учреждения Национальной системы здравоохранения, которое оказывало медицинскую помощь (1-13)
3. SEX – пол (женский – 1; мужской – 2)
4. PATIENT_TYPE - вид медицинской помощи, которую пациент получал в отделении (1 - для возвращения домой; 2 - для госпитализации)
5. DATE_DIED – дата смерти (указывается дата или 9999-99-99, если пациент не умер)
6. INTUBED – был ли пациент подключен к аппарату искусственной вентиляции легких (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
7. PNEUMONIA – была ли у пациента уже пневмония (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
8. AGE – возраст пациента
9. PREGANT – беременность (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
10. DIABETES – болен ли пациент диабетом (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
11. COPD - есть ли у пациента хроническая обструктивная болезнь легких (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
12. ASTHMA – есть ли у пациента астма (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
13. INMSUPR – имеет ли пациент подавленный иммунитет (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
14. HIPERTENSION – есть ли у пациента гипертония (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
15. OTHER_DISEASE – есть ли у пациента другие заболевания (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
16. CARDIOVASCULAR – есть ли у пациента заболевания, связанные с сердцем или кровеносными сосудами (1 – да, 2 – нет, 97 и 99 – отсутствие данных)

17. OBESITY- страдает ли пациент ожирением (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
18. RENAL_CHRONIC – есть ли у пациента хроническое заболевание почек (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
19. TOBACCO – пациент-курильщик (1 – да, 2 – нет, 97 и 99 – отсутствие данных)
20. CLASSIFICATION – результат теста на Covid-19 (значения 1-3 означают, что у пациента был диагностирован covid в разной степени; 4 и выше означает, что пациент не является носителем covid или что тест не дает результатов)
21. ICU – был ли пациент госпитализирован в отделение интенсивной терапии (1 – да, 2 – нет, 97 и 99 – отсутствие данных)

Для данного датасета для всех классификаторов в качестве целевого столбца выберем столбец «CLASIFFICATION_FINAL» (значения: «1», «2», «3», «4»), отберем 5 значимых определяющих признаков на основе важности признаков, используя классификатор ExtraTreesClassifier.

Перед обучением на сырых данных во избежание ошибки несоответствия типов уберем единственный столбец с датой смерти (также данный столбец не будет влиять на результат теста на Covid19), сократим количество записей до 10 тысяч, так как в первоначальном датасете содержится около миллиона записей, что значительно замедляет работу программы для учебного проекта, и проведем отбор 5 значимых признаков.

Код:

```
#загрузка и выбор первых 10000 строк датасета
dataset =
pd.read_csv(r"/Users/katyaanosova/Desktop/Covid_Data.csv")
dataset.drop('DATE_DIED', axis = 1, inplace = True)
dataset = dataset.head(10000)

#выделение целевого столбца и выбор первых 10000 строк
aim_label = dataset['CLASIFFICATION_FINAL']
aim_label = aim_label.head(10000)
dataset.drop('CLASIFFICATION_FINAL', axis = 1, inplace = True)

#отбор значимых признаков
model = ExtraTreesClassifier()
model.fit(dataset, aim_label)
print(model.feature_importances_)
```

Результат отбора:

```
[0.02172063 0.20229271 0.0109869  0.00663675 0.01706895 0.03094936
 0.49400901 0.01510243 0.02543232 0.0115651  0.00968934 0.01342576
 0.02637528 0.01563012 0.01926105 0.02209054 0.01996494 0.0193579
 0.0184409 ]
```

Зная, что чем больше значение оценки, тем важнее признак, выберем 10 значений:

PNEUMONIA – наличие у пациента пневмонии

AGE – возраст пациента

COPD - наличие у пациента хронической обструктивной болезни легких

HIPERTENSION - наличие у пациента гипертонии

OBESITY – наличие у пациента ожирения

Таким образом, сформируем отдельно выборку данных по интересующим признакам, затем с помощью train_test_split разделим данные на обучающую и тестовую выборку.

Код:

```
#формирование новой выборки значимых признаков
columns = [0, 1, 2, 3, 4, 7, 8, 10, 11, 13, 14, 16, 17, 18]
dataset.drop(dataset.columns [columns], axis = 1, inplace = True)

#разделение датасета на обучающую и тестовую выборки
X_train,  X_test,  Y_train,  Y_test  =  train_test_split(dataset,
aim_label, test_size=0.1, random_state=0)
```

Результат – выборка из значимых признаков, обучающая и тестовая выборки:

```
   PNEUMONIA  AGE  COPD  HIPERTENSION  OBESITY
0          1   65     2             1        2
1          1   72     2             1        1
2          2   55     2             2        2
3          2   53     2             2        2
4          2   68     2             1        2

Обучающая выборка
      PNEUMONIA  AGE  COPD  HIPERTENSION  OBESITY
1554          1   77     2             2        2
2087          2   52     2             1        2
5470          2   43     2             2        2
2363          2   53     2             2        2
7570          2   58     2             2        2
...         ...  ...   ...           ...      ...
9225          1   52     2             2        2
4859          1   51     2             2        2
3264          2   42     2             2        2
```

```
9845          1   59     2             2        2
2732          2   34     2             2        2
[9000 rows x 5 columns]

Тестовая выборка
      PNEUMONIA  AGE  COPD  HIPERTENSION  OBESITY
9394          2   41     2             2        2
898           1   78     2             1        2
2398          2   37     2             2        2
5906          1   47     2             2        1
2343          2   32     2             2       98
...         ...  ...   ...           ...      ...
9319          2   29     2             2        2
2662          2   57     2             1        2
6925          2   26     2             2        2
8070          2   35     2             2        2
3651          2   55     2             2        2
[1000 rows x 5 columns]
```

Стохастический градиентный спуск

Проведем обучение и оценку модели, используя метод стохастического градиентного спуска:

Код:

```
#обучение модели
sgd = SGDClassifier (loss='hinge', penalty='l2', alpha=1e-3,
random_state=42, max_iter=5, tol=None)
sgd.fit(X_train, Y_train)
y_pred = sgd.predict(X_test)
score = accuracy_score(Y_test, y_pred)
print(f'Точность SGD-классификатора: {round(score * 100, 2)}%')
```

Результат:

```
Точность SGD-классификатора: 84.9%
```

Затем произведем очистку данных – приведем строки с отсутствующими данными (97, 98, 99) к единому формату (1 – да, 2 – нет, 3 – данные отсутствуют), мы не будем удалять эти строки, так как эти данные тоже важны.

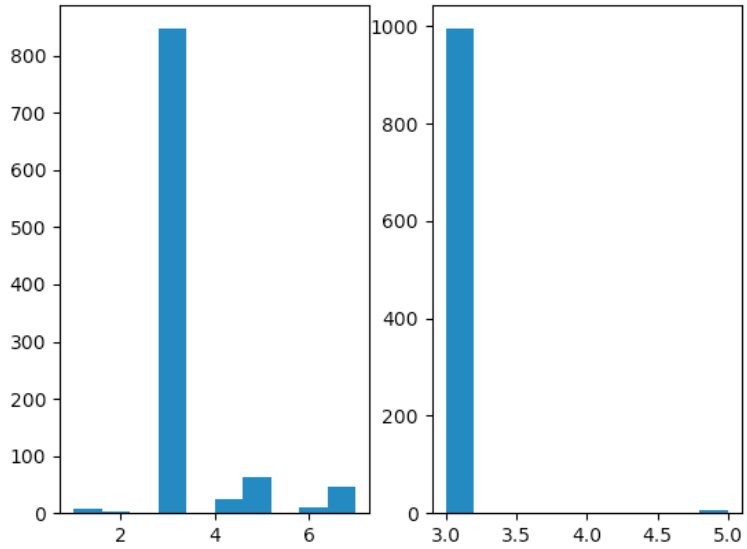
Код:

```
#очистка выборки значимых признаков
dataset.loc[dataset['OBESITY'] > 2, 'OBESITY'] = 3
dataset.loc[dataset['PNEUMONIA'] > 2, 'PNEUMONIA'] = 3
dataset.loc[dataset['COPD'] > 2, 'COPD'] = 3
dataset.loc[dataset['HIPERTENSION'] > 2, 'HIPERTENSION'] = 3
dataset = dataset.fillna(3)
```

После очистки данных снова произведем обучение и выведем оценку точности модели:

```
Точность SGD-классификатора: 85.0%
```

Далее произведем визуализацию данных.



	Значения тестовой выборки	Предсказанные значения
9394	4	3
898	3	3
2398	3	3
…	…	…
6925	3	3
8070	3	3
3651	3	3

Можно заметить, что данный классификатор имеет относительно небольшую точность. В основном, хорошо определяет результат Covid-19 со степенью тяжести «3». Скорее всего ошибка в выборке данных, они не являются уравновешенными, идет большой перевес класса «3».

Линейный метод опорных векторов

Проведем обучение и оценку модели, используя линейный метод опорных векторов:

Код:

```
svc = LinearSVC()
```

```
svc.fit(X_train, Y_train)
y_pred = svc.predict(X_test)
score = accuracy_score(Y_test, y_pred)
print(f'Точность SVC-классификатора: {round(score * 100, 2)}%')
```

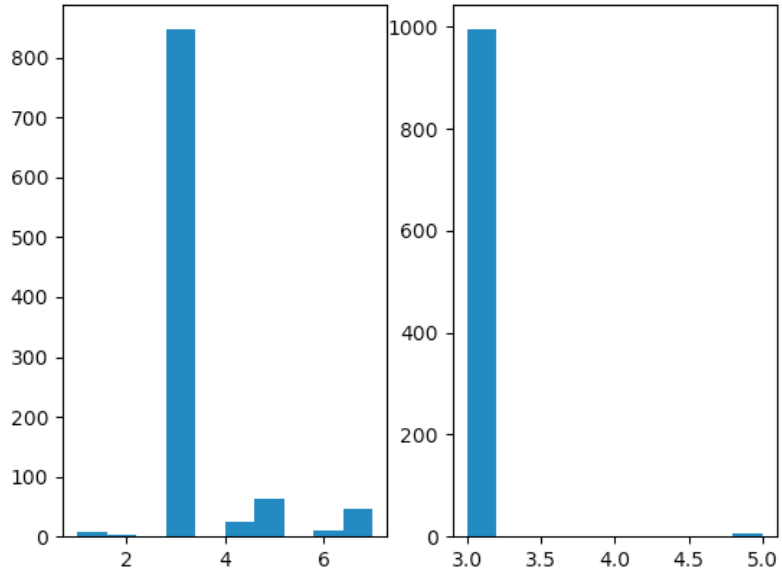
Результат:

```
Точность SVC-классификатора: 84.6%
```

Затем сделаем очистку данных и снова произведем обучение с оценкой точности модели:

```
Точность SVC-классификатора: 84.7%
```

Визуализация данных:



	Значения тестовой выборки	**Предсказанные значения**
9394	4	3
898	3	3
2398	3	3
…	…	…
6925	3	3
8070	3	3
3651	3	3

Данный классификатор имеет меньшую точность, чем предыдущий, а также снова видно ошибку в выборке неуравновешенных данных (идет большой перевес класса «3»).

Классификация случайного леса

Проведем обучение и оценку модели, используя метод случайного леса.
Код:

```
rf = RandomForestClassifier(max_depth = 2, random_state = 0)
rf.fit(X_train, Y_train)
y_pred = rf.predict(X_test)
score = accuracy_score(Y_test, y_pred)
print(f'Точность случайного леса: {round(score * 100, 2)}%')
```

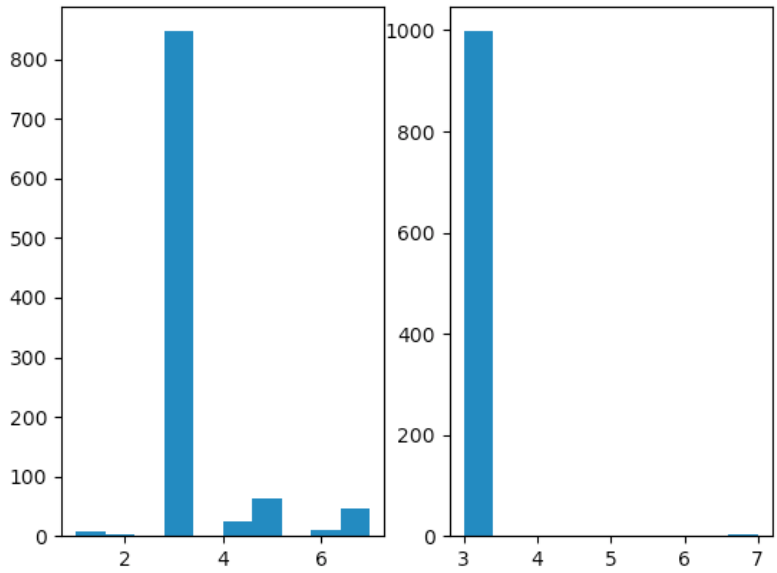
Результат:

```
Точность случайного леса: 84.8%
```

Затем сделаем очистку данных и снова произведем обучение с оценкой точности модели:

```
Точность случайного леса: 84.8%
```

Визуализация:



	Значения тестовой выборки	**Предсказанные значения**
9394	4	3
898	3	3
2398	3	3
…	…	…
6925	3	3
8070	3	3
3651	3	3

Данный классификатор имеет большую точность, чем предыдущий, но меньшую, чем первый, а также снова видно ошибку в выборке неуравновешенных данных (идет большой перевес класса «3»).

Вывод

Таким образом, в результате выполнения лабораторной работы мы исследовали классификаторы библиотеки Sklearn. В результате исследования для данного датасета и при проведенной обработке данных наиболее точным оказался метод стохастического градиентного спуска. Также на основе данного датасета стало понятно, почему важно выбирать уравновешенные данные без перевеса какого-либо класса.