**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»**

Выполнил:

студент гр. ИСТбд-42

Курбаниязов. Ш.Р

Проверил:

к.т.н., доцент

*Шишкин В.В.*

Ульяновск 2022 г.

Выполнение лабораторной работы по теме:

«Исследование инструментов классификации

библиотеки Scikit-learn»

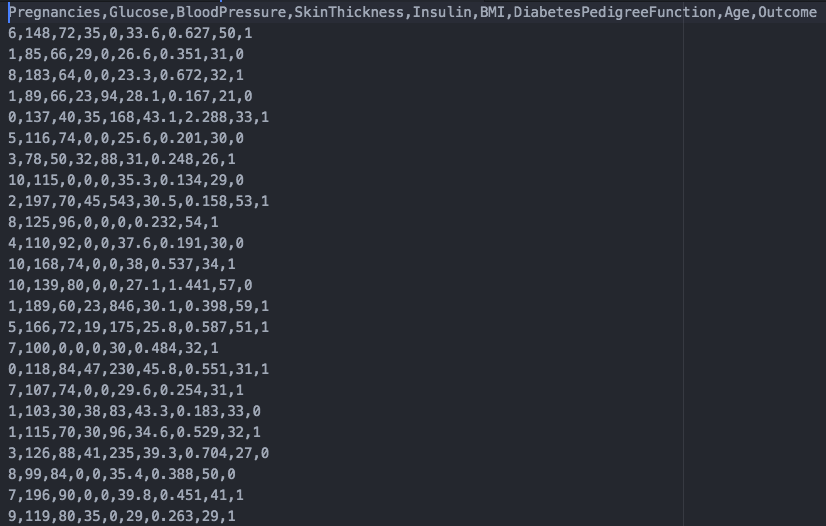
1. Ознакомиться с классификаторами библиотеки Scikit-learn. Выбрать для исследования не менее 3 классификаторов. Выбрать набор данных для задач классификации из открытых источников.

Классификаторы:

1. - метод случайного леса;
2. - стохастический градиентный спуск;
3. - линейный дискриминантный анализ.

Набор данных взят с сайта Kaggle (диагностирование диабета на основе общего анализа и обследований)

https://www.kaggle.com/datasets/whenamancodes/predict-diabities?resource=download



1. Для всех классификаторов определить целевой столбец и набор признаков. Целевой столбец: Outcome (есть диабет – 1, нет - 0). Набор признаков: остальные столбцы.
2. Подготовить данные к обучению: разделить выборку на набор признаков и целевой столбец и разделить выборку на обучающую и тестовую части.

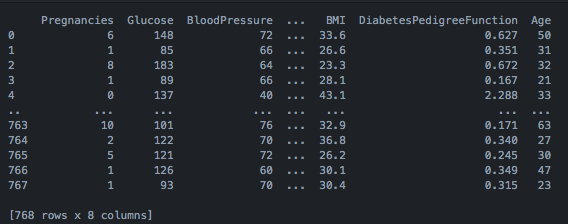
dataset = pd.read\_csv("diabetes.csv")

target = dataset["Outcome"]

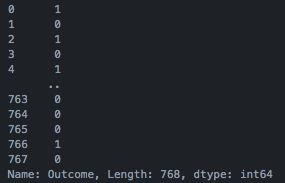
dataset.drop("Outcome", axis = 1, inplace = True)

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(dataset, target, random\_state=0)

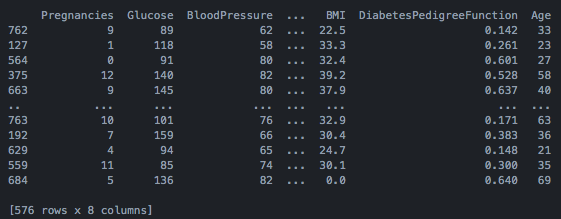
Набор признаков:



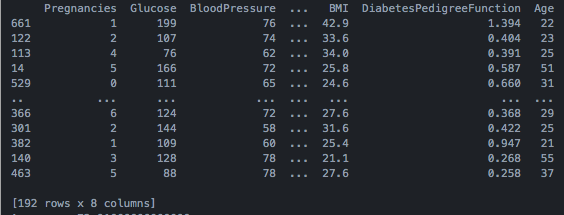
Целевой столбец:



Обучающая выборка:



Тестовая выборка:



*Метод случайного леса*

1. Обучение и оценка модели на сырых данных.

rf\_model = RandomForestClassifier(n\_estimators=100)

rf\_model.fit(X\_train, Y\_train)

predictions = rf\_model.predict(X\_test)

print("Точность классификатора: {}%".format((rf\_model.score(X\_test,Y\_test))\*100))

Точность классификатора: 78.64583333333334%

1. Предобработка данных: нормализация и стандартизация

scaler = StandardScaler().fit(X\_train)

scaler.transform(X\_train)

scaler.transform(X\_test)

scaler = Normalizer().fit(X\_train)

scaler.transform(X\_train)

scaler.transform(X\_test)

1. Обучение и оценка точности модели на очищенных данных.

rf\_model = RandomForestClassifier(n\_estimators=100)

rf\_model.fit(X\_train, Y\_train)

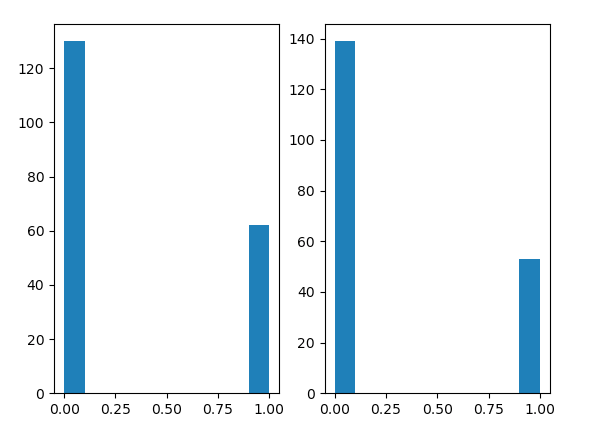
predictions1 = rf\_model.predict(X\_test)

print("Точность классификатора: {}%".format((rf\_model.score(X\_test,Y\_test))\*100))

Точность классификатора: 81.25%

1. Визуализация данных:

Первая группа гистограмм – значения целевого столбца тестовой выборки, вторая – предсказанные значения:



|  |  |
| --- | --- |
| **Значения тестовой выборки** | **Предсказанные значения** |
| 1 | 1 |
| 0 | 0 |
| 0 | 0 |
| … | … |
| *1* | *0* |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |

Таким образом, у данного классификатора довольно высокий показатель точности. А после масштабирования данных показатель точности увеличивается.

*Стохастический градиентный спуск*

1. Обучение и оценка модели на сырых данных.

sgd = SGDClassifier (loss='hinge', penalty='l2', alpha=1e-3, random\_state=42, max\_iter=5, tol=None)

sgd.fit(X\_train, Y\_train)

predictions = sgd.predict(X\_test)

print("Точность классификатора: {}%".format((sgd.score(X\_test,Y\_test))\*100))

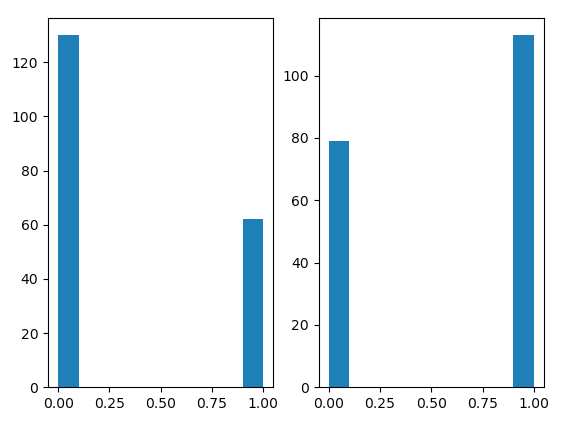
Точность классификатора: 55.729166666666664%

1. После стандартизации и нормализации данных была получена следующая оценка точности работы классификатора:

Точность классификатора: 55.729166666666664%

1. Визуализация данных:

Первая группа гистограмм – значения целевого столбца тестовой выборки, вторая – предсказанные значения:



|  |  |
| --- | --- |
| **Значения тестовой выборки** | **Предсказанные значения** |
| 1 | 1 |
| *0* | *1* |
| 0 | 0 |
| … | … |
| 1 | 1 |
| *0* | *1* |
| 0 | 0 |
| 0 | 0 |

Таким образом, у данного классификатора низкий показатель точности для данного набора данных.

*Линейный дискриминантный анализ*

1. Обучение и оценка модели на сырых данных.

lda\_model = LinearDiscriminantAnalysis()

lda\_model.fit(X\_train, Y\_train)

predictions = lda\_model.predict(X\_test)

print("Точность классификатора: {}%".format((lda\_model.score(X\_test,Y\_test))\*100))

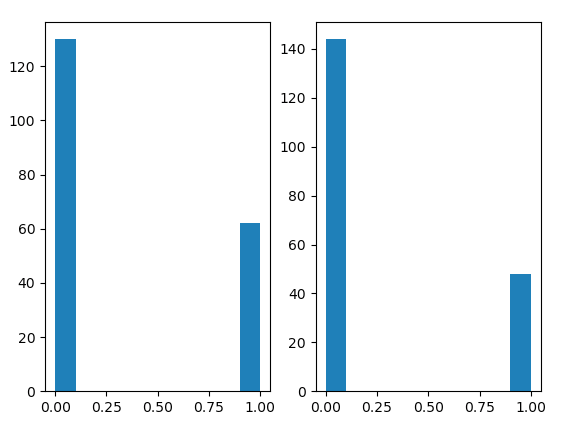
Точность классификатора: 80.20833333333334%

1. После стандартизации и нормализации данных была получена следующая оценка точности работы классификатора:

Точность классификатора: 80.20833333333334%

1. Визуализация данных:

Первая группа гистограмм – значения целевого столбца тестовой выборки, вторая – предсказанные значения:



|  |  |
| --- | --- |
| **Значения тестовой выборки** | **Предсказанные значения** |
| 1 | 1 |
| 0 | 0 |
| 0 | 0 |
| … | … |
| *1* | *0* |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |

Таким образом, у данного классификатора высокий показатель точности для данного набора данных, но масштабирование данных не повлияло на точность классификатора.

**Вывод**

В ходе лабораторной работы мы исследовали классификаторы, входящие в библиотеку Scikit-learn, а также научились визуализировать данные для различных классификаторов. Анализируя полученные результаты, можно сделать вывод о том, что в данном случае для данного датасета наилучшую точность имеет классификатор на основе линейного дискриминантного анализа, а наименьшую точность имеет стохастический градиентный спуск.