**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-41

Бакунькина Н.М.

Проверил:

к.т.н., доцент

*Шишкин В.В.*

Ульяновск 2022 г.

**Выполнение лабораторной работы по теме:**

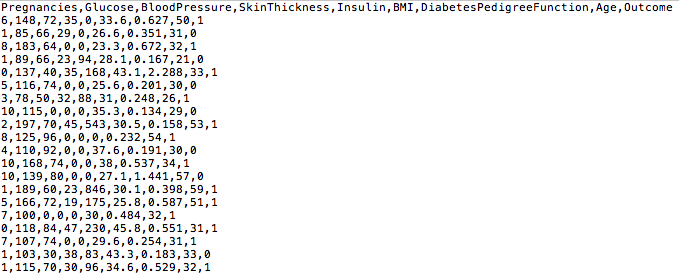
**«Исследование инструментов классификации библиотеки Scikit-learn»**

1. После ознакомления с классификаторами библиотеки Scikit-learn были выбраны следующие три классификатора, которые наиболее подходят для работы с числовыми значениями:

* дерево решений (Decision Tree Classifier);
* метод опорных векторов (SVM);
* линейный дискриминантный анализ (LDA).

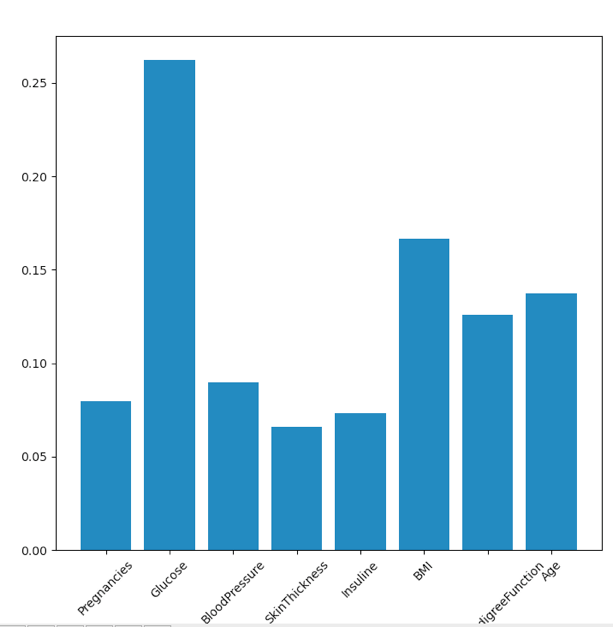
1. Затем был выбран следующий набор данных о диагностировании диабета на основе общего анализа и обследований:

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



1. Для всех классификаторов целевой столбец будет один – Outcome, который определяет – есть диабет (1) или нет (0).

Набор признаков, на основе которых будет производиться классификация, определим с помощью отбора признаков random forest. Получим следующую гистограмму влияния каждого признака на целевой столбец:



По заданию для каждого классификатора определим свой набор признаков:

* первый классификатор «Древо решений» исследуем на основе всего набора признаков (данный классификатор чаще всего используется именно для классификации на основе нескольких признаков);
* второй классификатор «Метод опорный векторов» исследуем на наиболее влияющих признаках: уровень глюкозы и индекса массы тела;
* третий классификатор «Линейный дискриминантный анализ» исследуем второй паре влияющих признаков на возрасте и значении функции вероятности возникновения диабета по родословной.

*Древо решений*

1. Для начала подготовим данные к обучению, а именно сначала разделим выборку на набор признаков и целевой столбец, а затем разделим выборку на обучающую и тестовую часть.

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

target = dataset["Outcome"]

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

Pregnancies Glucose BloodPressure ... BMI DiabetesPedigreeFunction Age

0 6 148 72 ... 33.6 0.627 50

1 1 85 66 ... 26.6 0.351 31

2 8 183 64 ... 23.3 0.672 32

3 1 89 66 ... 28.1 0.167 21

4 0 137 40 ... 43.1 2.288 33

.. ... ... ... ... ... ... ...

763 10 101 76 ... 32.9 0.171 63

764 2 122 70 ... 36.8 0.340 27

765 5 121 72 ... 26.2 0.245 30

766 1 126 60 ... 30.1 0.349 47

767 1 93 70 ... 30.4 0.315 23

[768 rows x 8 columns]

0 1

1 0

2 1

3 0

4 1

..

763 0

764 0

765 0

766 1

767 0

Name: Outcome, Length: 768, dtype: int64

X\_train1, X\_test1, Y\_train1, Y\_test1 = train\_test\_split(dataset, target, random\_state=0)

Выборка для обучения

Pregnancies Glucose BloodPressure ... BMI DiabetesPedigreeFunction Age

762 9 89 62 ... 22.5 0.142 33

127 1 118 58 ... 33.3 0.261 23

564 0 91 80 ... 32.4 0.601 27

375 12 140 82 ... 39.2 0.528 58

663 9 145 80 ... 37.9 0.637 40

.. ... ... ... ... ... ... ...

763 10 101 76 ... 32.9 0.171 63

192 7 159 66 ... 30.4 0.383 36

629 4 94 65 ... 24.7 0.148 21

559 11 85 74 ... 30.1 0.300 35

684 5 136 82 ... 0.0 0.640 69

[576 rows x 8 columns]

Тестовая выборка

Pregnancies Glucose BloodPressure ... BMI DiabetesPedigreeFunction Age

661 1 199 76 ... 42.9 1.394 22

122 2 107 74 ... 33.6 0.404 23

113 4 76 62 ... 34.0 0.391 25

14 5 166 72 ... 25.8 0.587 51

529 0 111 65 ... 24.6 0.660 31

.. ... ... ... ... ... ... ...

366 6 124 72 ... 27.6 0.368 29

301 2 144 58 ... 31.6 0.422 25

382 1 109 60 ... 25.4 0.947 21

140 3 128 78 ... 21.1 0.268 55

463 5 88 78 ... 27.6 0.258 37

[192 rows x 8 columns]

1. Проведем обучение и оценку модели на сырых данных.

tree\_model = tree.DecisionTreeClassifier()

tree\_model.fit (X\_train1, Y\_train1)

predictions = tree\_model.predict(X\_test1)

print("Accuracy: {}".format((tree\_model.score(X\_test1,Y\_test1))\*100))

Accuracy: 70.3125

1. Проведем предобработку данных, а именно масштабируем их с помощью стандартизации и нормализации (нам не нужно их унифицировать, так как в данном датасете не используются категориальные целочисленные переменные), затем повторим обучение и оценку модели на очищенных данных:

scaler = StandardScaler().fit(X\_train1)

scaler.transform(X\_train1)

scaler.transform(X\_test1)

scaler = Normalizer().fit(X\_train1)

scaler.transform(X\_train1)

scaler.transform(X\_test1)

tree\_model = tree.DecisionTreeClassifier()

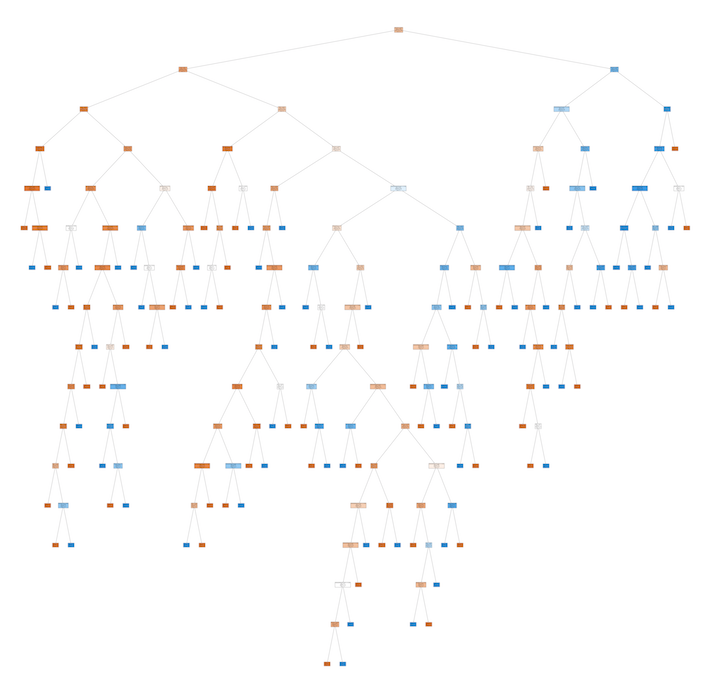
tree\_model.fit (X\_train1, Y\_train1)

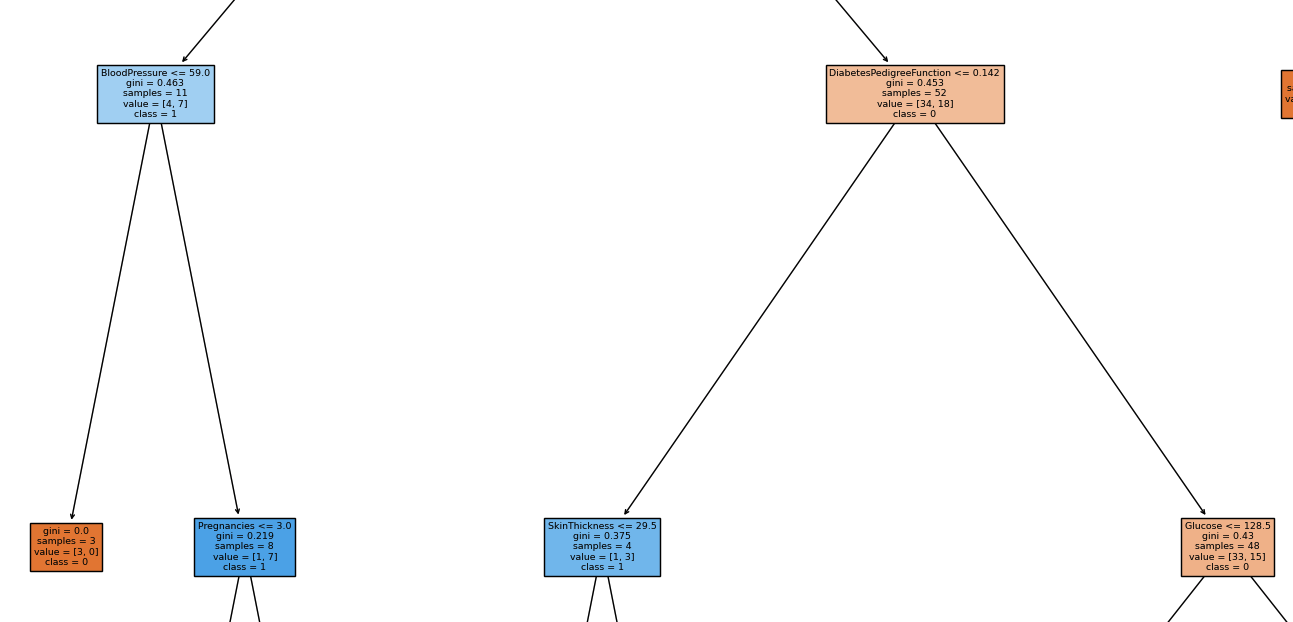
predictions = tree\_model.predict(X\_test1)

print("Accuracy: {}".format((tree\_model.score(X\_test1,Y\_test1))\*100))

Accuracy: 72.91666666666666

1. Так как мы делали исследование на всем наборе признаков, то лучшим способом визуализации решения будет построение непосредственно дерева решений:



Если мы приблизим данный рисунок, то можем увидеть, какие данные использовал классификатор: 

Также представим результаты решения в табличной форме. Ниже представлена выдержка из таблицы сравнения результатов полученных классов и классов тестовой выборки (несоответствия подчеркнуты):

|  |  |  |
| --- | --- | --- |
|  | **Предсказанные классы** | **Классы тестовой выборки** |
| 661 | 1 | 1 |
| 122 | 0 | 0 |
| 113 | 0 | 0 |
| *14* | *0* | *1* |
| 529 | 0 | 0 |
| … | … | … |
| *366* | *0* | *1* |
| 301 | 1 | 1 |
| 382 | 0 | 0 |
| 140 | 0 | 0 |
| 463 | 0 | 0 |

Таким образом, можно заметить, что после масштабирования данных точность прогнозирования повышается, но все же процент точности данного алгоритма классификации невысокий.

*Метод опорных векторов*

1. Для начала подготовим данные к обучению, а именно сначала разделим выборку на набор признаков и целевой столбец, а затем разделим выборку на обучающую и тестовую часть.

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

target = dataset["Outcome"]

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

dataset\_glucose\_bmi = dataset[["Glucose", "BMI"]]

Glucose BMI

0 148 33.6

1 85 26.6

2 183 23.3

3 89 28.1

4 137 43.1

.. ... ...

763 101 32.9

764 122 36.8

765 121 26.2

766 126 30.1

767 93 30.4

[768 rows x 2 columns]

0 1

1 0

2 1

3 0

4 1

..

763 0

764 0

765 0

766 1

767 0

Name: Outcome, Length: 768, dtype: int64

X\_train2, X\_test2, Y\_train2, Y\_test2 = train\_test\_split(dataset\_glucose\_bmi, target, random\_state=0)

Выборка для обучения

Glucose BMI

762 89 22.5

127 118 33.3

564 91 32.4

375 140 39.2

663 145 37.9

.. ... ...

763 101 32.9

192 159 30.4

629 94 24.7

559 85 30.1

684 136 0.0

[576 rows x 2 columns]

Тестовая выборка

Glucose BMI

661 199 42.9

122 107 33.6

113 76 34.0

14 166 25.8

529 111 24.6

.. ... ...

366 124 27.6

301 144 31.6

382 109 25.4

140 128 21.1

1. 88 27.6
2. Проведем обучение и оценку модели на сырых данных.

svm\_model = svm.SVC()

svm\_model.fit(X\_train2, Y\_train2)

predictions = svm\_model.predict(X\_test2)

print("Accuracy: {}".format((svm\_model.score(X\_test2,Y\_test2))\*100))

Accuracy: 76.5625

1. Проведем предобработку данных, а именно масштабируем их с помощью стандартизации и нормализации, затем повторим обучение и оценку модели на очищенных данных:

scaler = StandardScaler().fit(X\_train2)

scaler.transform(X\_train2)

scaler.transform(X\_test2)

scaler = Normalizer().fit(X\_train2)

scaler.transform(X\_train2)

scaler.transform(X\_test2)

svm\_model = svm.SVC()

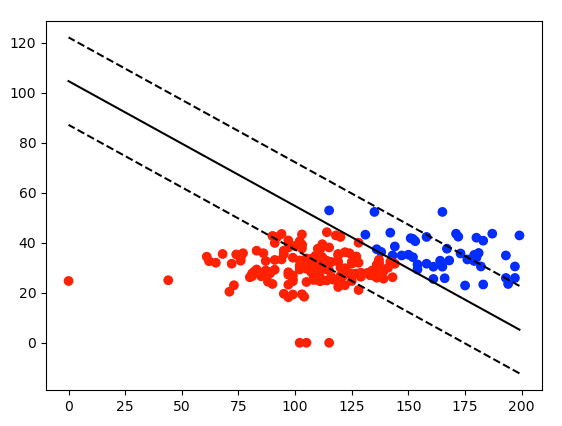
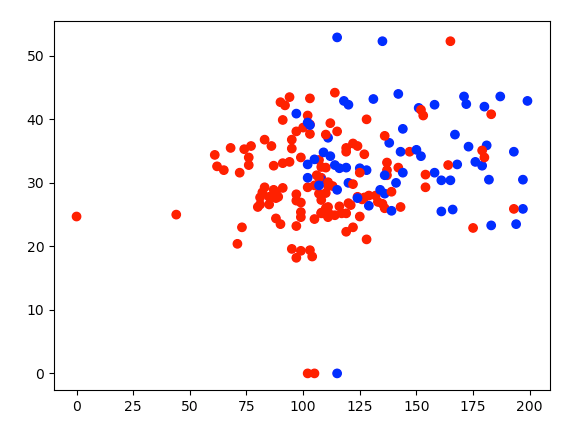
svm\_model.fit(X\_train2, Y\_train2)

predictions = svm\_model.predict(X\_test2)

print("Accuracy: {}".format((svm\_model.score(X\_test2,Y\_test2))\*100))

Accuracy: 76.5625

1. Визуализируем результат. На первом рисунке представлен набор данных до классификации, на втором после классификации методом опорных векторов (соответственно изображена оптимальная разделяющая гиперплоскость).



Также представим результаты решения в табличной форме. Ниже представлена выдержка из таблицы сравнения результатов полученных классов и классов тестовой выборки (несоответствия подчеркнуты):

|  |  |  |
| --- | --- | --- |
|  | **Предсказанные классы** | **Классы тестовой выборки** |
| 661 | 1 | 1 |
| 122 | 0 | 0 |
| 113 | 0 | 0 |
| 14 | 1 | 1 |
| 529 | 0 | 0 |
| … | … | … |
| *366* | *0* | *1* |
| *301* | *0* | *1* |
| 382 | 0 | 0 |
| 140 | 0 | 0 |
| 463 | 0 | 0 |

Делая вывод по классификации с помощью метода опорных векторов, можно отметить, что в данном случае масштабирование данных не повлияло на результат точности алгоритма. Анализируя визуальную информацию, также отметим, что точность данного способа невысока.

*Линейный дискриминантный анализ*

1. Подготовим данные к обучению, а именно разделим выборку на набор признаков и целевой столбец, а затем разделим выборку на обучающую и тестовую часть.

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

target = dataset["Outcome"]

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

dataset\_age\_family = dataset[["DiabetesPedigreeFunction", "Age"]]

DiabetesPedigreeFunction Age

0 0.627 50

1 0.351 31

2 0.672 32

3 0.167 21

4 2.288 33

.. ... ...

763 0.171 63

764 0.340 27

765 0.245 30

766 0.349 47

767 0.315 23

[768 rows x 2 columns]

0 1

1 0

2 1

3 0

4 1

..

763 0

764 0

765 0

766 1

767 0

Name: Outcome, Length: 768, dtype: int64

X\_train3, X\_test3, Y\_train3, Y\_test3 = train\_test\_split(dataset\_age\_family, target, random\_state=0)X\_train2,

Выборка для обучения

DiabetesPedigreeFunction Age

762 0.142 33

127 0.261 23

564 0.601 27

375 0.528 58

663 0.637 40

.. ... ...

763 0.171 63

192 0.383 36

629 0.148 21

559 0.300 35

684 0.640 69

[576 rows x 2 columns]

Тестовая выборка

DiabetesPedigreeFunction Age

661 1.394 22

122 0.404 23

113 0.391 25

14 0.587 51

529 0.660 31

.. ... ...

366 0.368 29

301 0.422 25

382 0.947 21

140 0.268 55

463 0.258 37

[192 rows x 2 columns]

1. Проведем обучение и оценку модели на сырых данных.

lda\_model = LinearDiscriminantAnalysis()

lda\_model.fit(X\_train3, Y\_train3)

predictions = lda\_model.predict(X\_test3)

print("Accuracy: {}".format((lda\_model.score(X\_test3,Y\_test3))\*100))

Accuracy: 66.66666666666666

1. Проведем предобработку данных, а именно масштабируем их с помощью стандартизации и нормализации, затем повторим обучение и оценку модели на очищенных данных:

scaler = StandardScaler().fit(X\_train3)

scaler.transform(X\_train3)

scaler.transform(X\_test3)

scaler = Normalizer().fit(X\_train3)

scaler.transform(X\_train3)

scaler.transform(X\_test3)

lda\_model = LinearDiscriminantAnalysis()

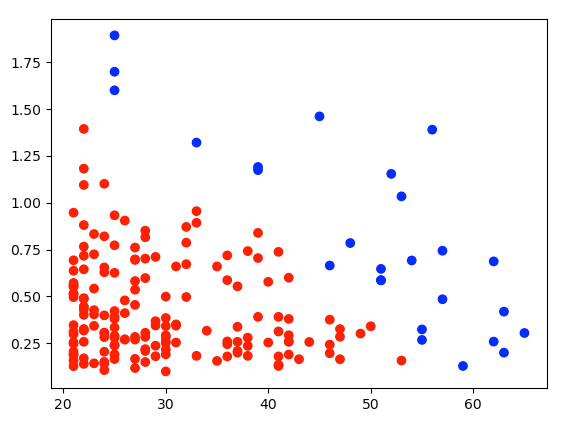
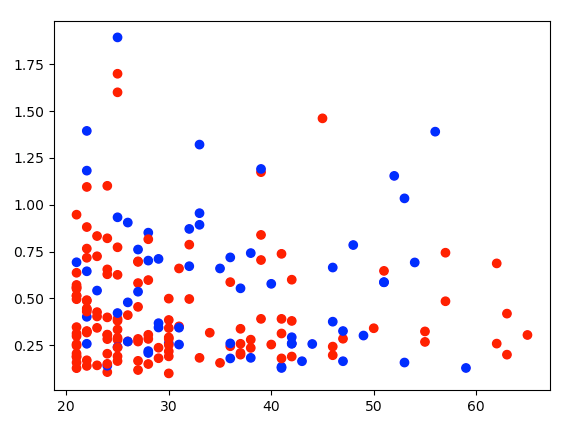
lda\_model.fit(X\_train3, Y\_train3)

predictions = lda\_model.predict(X\_test3)

print("Accuracy: {}".format((lda\_model.score(X\_test3,Y\_test3))\*100))

Accuracy: 66.66666666666666

1. Визуализируем результат. На первом рисунке представлен набор данных до классификации, на втором после классификации линейным дискриминантным анализом.



Также представим результаты решения в табличной форме. Ниже представлена выдержка из таблицы сравнения результатов полученных классов и классов тестовой выборки (несоответствия подчеркнуты):

|  |  |  |
| --- | --- | --- |
|  | **Предсказанные классы** | **Классы тестовой выборки** |
| *661* | *0* | *1* |
| 122 | 0 | 0 |
| 113 | 0 | 0 |
| 14 | 1 | 1 |
| 529 | 0 | 0 |
| … | … | … |
| *366* | *0* | *1* |
| *301* | *0* | *1* |
| 382 | 0 | 0 |
| *140* | *1* | *0* |
| 463 | 0 | 0 |

Таким образом, мы получили классификацию на основе линейного дискриминантного анализа. Данный способ имеет наименьшую точность, так как подобранный набор признаков не имеет сильного влияния на целевой столбец.

**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 дерево решений, в основном, из-за того, что использовались все признаки, а наименьшую точность имеет линейный дискриминантный анализ, который использовал набор признаков, не имеющий высокого влияния на целевой столбец.

Таким образом, можно сделать вывод, что при обработке данных важно не только выбрать подходящий классификатор, но и правильно обработать данные и выбрать наиболее определяющие признаки.