МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по ИДЗ
по дисциплине «Машинное обучение»
Тема: Классификация статуса заявки на кредит

Студент гр. 0304 _____ Сергеев Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2024

Цель работы.

Определить модель, демонстрирующую наибольшую точность предсказания статуса заявки на кредит.

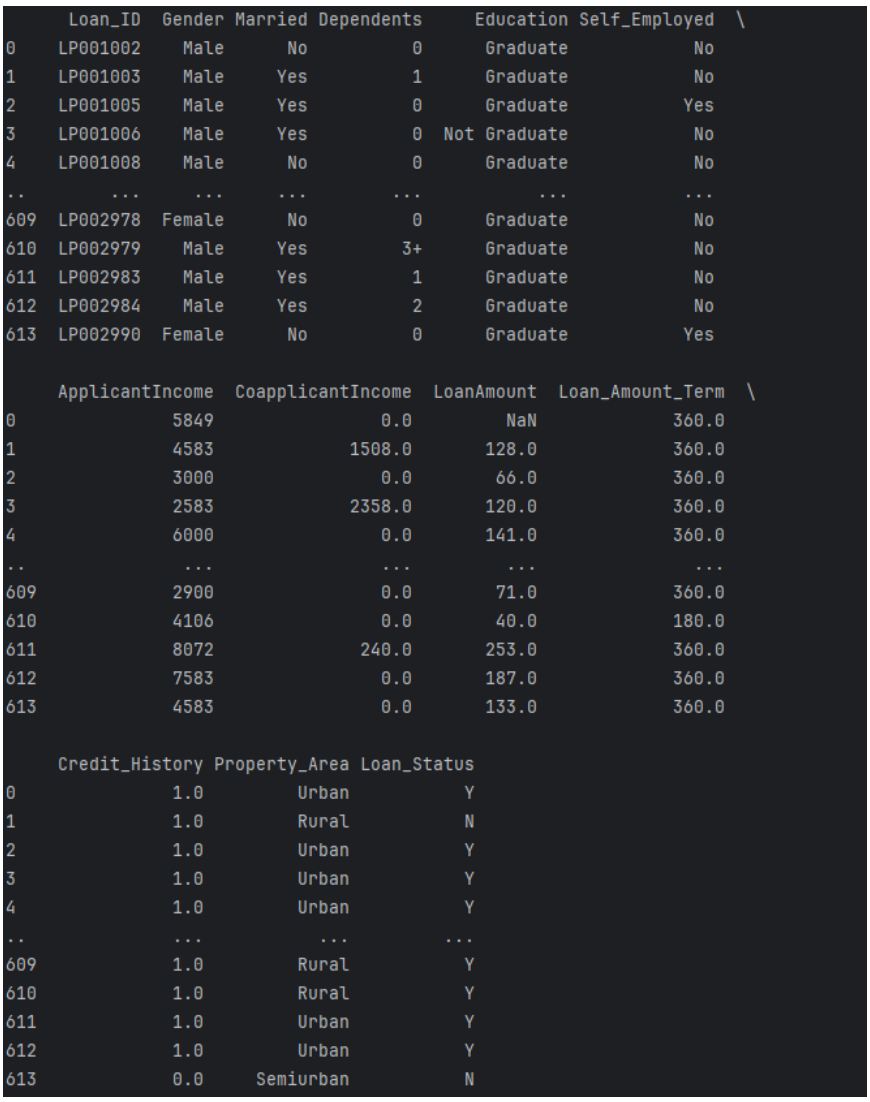
Выполнение работы.

Описание датасета

Датасет «Loan Prediction Problem Dataset» состоит из 614 наблюдений, каждое из которых содержит данные по 12 признакам:

1. Loan_ID – уникальный номер заявки на кредит;
2. Gender – пол, подавшего заявки;
3. Married – находиться ли человек, подавший заявки, в браке;
4. Dependents – количество человек, которые финансово зависят от человека, подавшего заявку;
5. Education – статус образования;
6. Self_Employed – является ли заявитель самозанятым;
7. ApplicantIncome – доход заявителя;
8. CoapplicantIncome – доход человека, которого заявитель объявил доверенным;
9. LoanAmount – сумма займа в тысячах;
10. Loan_Amount_Term – время, отведённое на выплату займа, в месяцах;
11. Credit_History – отвечает ли кредитная история заявителя требованиям;
12. Property_Area – в какой местности проживает заявитель (городская, полугородская, деревенская).

Также имеется поле, соответствующее классу – Loan_Status – одобрен ли кредит. На рисунке 1 представлен исходный вид датасета.



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
..	...	...	...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

Рисунок 1 – Исходный вид датасета

Предобработка датасета

- Признак Loan_ID не несёт полезной информации, а просто является уникальным идентификаторов заявки, поэтому он был вынесен из рассмотрения;
- В датасете отсутствуют данные по признакам у некоторых наблюдений, на рисунке 2 приведено количество пропущенных значений по каждому признаку.

Рисунок 2 – Количество пропущенных значений по каждому признаку

Чтобы справиться с этой проблемой вместо пропущенных значений было решено подставить среднее значение для числовых признаков и моду для категориальных признаков. На рисунке 3 представлен поиск пропущенных значений после замены.

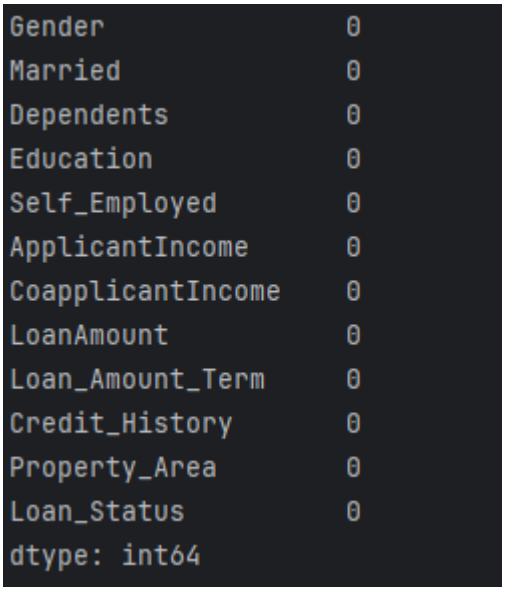
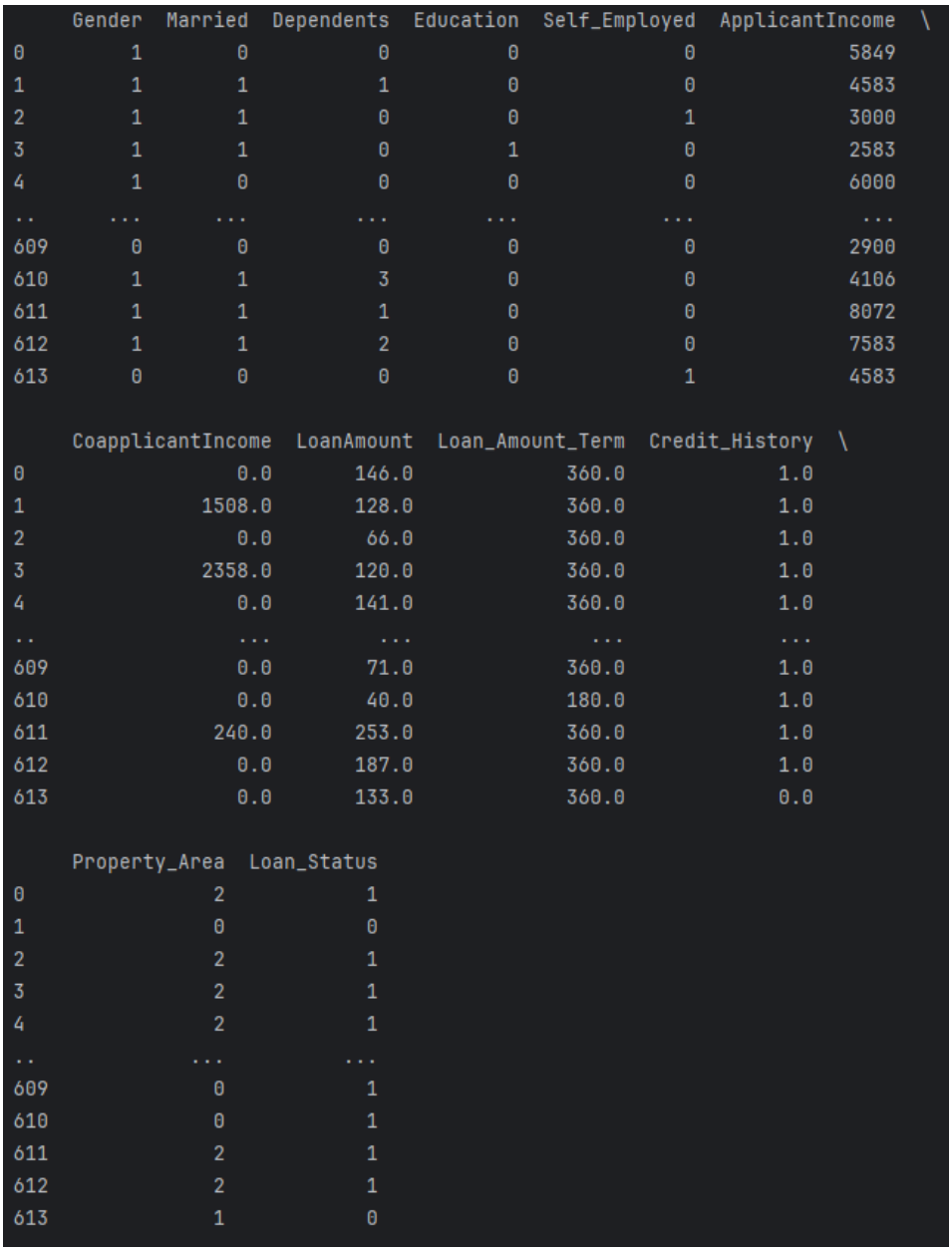


Рисунок 3 – Количество пропущенных значений по каждому признаку после замены

- Также так как признаки Gender, Married, Dependents, Education, Self_Employed, Property_Area, Loan_Status являются

категориальными, то с помощью LabelEncoder они были преобразованы к числовым. Преобразованный датасет представлен на рисунке 4.



	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	\
0	1	0	0	0	0	5849	
1	1	1	1	0	0	4583	
2	1	1	0	0	1	3000	
3	1	1	0	1	0	2583	
4	1	0	0	0	0	6000	
..	...	...	...	...	...	...	
609	0	0	0	0	0	2900	
610	1	1	3	0	0	4106	
611	1	1	1	0	0	8072	
612	1	1	2	0	0	7583	
613	0	0	0	0	1	4583	

	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	\
0	0.0	146.0	360.0	1.0	
1	1508.0	128.0	360.0	1.0	
2	0.0	66.0	360.0	1.0	
3	2358.0	120.0	360.0	1.0	
4	0.0	141.0	360.0	1.0	
..	...	...	...	...	
609	0.0	71.0	360.0	1.0	
610	0.0	40.0	180.0	1.0	
611	240.0	253.0	360.0	1.0	
612	0.0	187.0	360.0	1.0	
613	0.0	133.0	360.0	0.0	

	Property_Area	Loan_Status
0	2	1
1	0	0
2	2	1
3	2	1
4	2	1
..	...	...
609	0	1
610	0	1
611	2	1
612	2	1
613	1	0

Рисунок 4 – Преобразованный датасет

Статистический анализ

Для преобразованного датасета был проведён статистический анализ с помощью метода *describe()*. На рисунке 5 представлены данные о среднем значении, стандартном отклонении, минимальном и максимальном значении, а также 25, 50 и 75 перцентиль для каждого признака.

	Gender	Married	Dependents	Education	Self_Employed	\
count	614.000000	614.000000	614.000000	614.000000	614.000000	
mean	0.817590	0.653094	0.744300	0.218241	0.133550	
std	0.386497	0.476373	1.009623	0.413389	0.340446	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	0.000000	0.000000	0.000000	0.000000	
50%	1.000000	1.000000	0.000000	0.000000	0.000000	
75%	1.000000	1.000000	1.000000	0.000000	0.000000	
max	1.000000	1.000000	3.000000	1.000000	1.000000	

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	\
count	614.000000	614.000000	614.000000	614.000000	
mean	5403.459283	1621.245798	146.397394	342.000000	
std	6109.041673	2926.248369	84.037503	64.372489	
min	150.000000	0.000000	9.000000	12.000000	
25%	2877.500000	0.000000	100.250000	360.000000	
50%	3812.500000	1188.500000	129.000000	360.000000	
75%	5795.000000	2297.250000	164.750000	360.000000	
max	81000.000000	41667.000000	700.000000	480.000000	

	Credit_History	Property_Area	Loan_Status
count	614.000000	614.000000	614.000000
mean	0.773616	1.037459	0.687296
std	0.418832	0.787482	0.463973
min	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000
50%	1.000000	1.000000	1.000000
75%	1.000000	2.000000	1.000000
max	1.000000	2.000000	1.000000

Рисунок 5 – Статистические данные датасета

Описание рассматриваемых моделей

Для рассмотрения были выбраны стандартные методы для задачи классификации – наивный байесовский метод, решающее дерево, линейный дискриминантный анализ и метод опорных векторов.

- Наивный байесовский метод
  - Основан на теореме Байеса:

$$P(c_i|x) = \frac{P(x|c_i) * P(c_i)}{P(x)}$$

  - Таким образом для определения класса y используют

$$\hat{y} = argmax_{c_i}\{P(x|c_i) * P(c_i)\}$$

  - Наивный байесовский классификатор основан на предположении, что все атрибуты независимы. И таким образом:

$$P(x|c_i) = P(x_1, x_2, \ldots, x_d|c_i) = \prod_{j=1}^{d} P(x_j|c_i)$$

- Решающее дерево
  - Делит данные на несколько подмножеств, используя условия, основанные на признаках объектов
  - Разбиение делит данные на две группы, то есть точка разбиения вида $x_j \leq v$ вызывает разбиение:

$$D_Y = \{x \,|x \ \in D,\ x_j \leq v\}$$

$$D_N = \{x \,|x \ \in D,\ x_j > v\}$$

  - Выбор условия разбиения основывается на критериях:
    - Энтропия:

$$H(D) = -\sum_{i=1}^{k} P(c_i|D) log_2 P(c_i|D)$$

    - Критерий Джини:

$$G(D) = 1 - \sum_{i=1}^{k} P(c_i|D)^2$$

- Линейный дискриминантный анализ
  - Используется для классификации и снижения размерности данных, цель которого – найти направления в пространстве признаков, которые максимально разделяют данные между классами
  - Межклассовая дисперсия – описывает, насколько далеко центры классов расположены друг от друга:

    $S_B = \sum_{i=1}^{k} N_i(\mu_i - \mu)(\mu_i - \mu)^T$, где N_i – количество точек в классе i, μ_i – вектор среднего класса i, μ – общее среднее
  - Внутриклассовая дисперсия – описывает, насколько сильно разбросаны данные внутри класса:

$$S_W = \sum_{i=1}^{k} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$$

- Направление w, которое максимизирует отношение межклассовой дисперсии к внутриклассовой $J(w) = \frac{w^T S_B w}{w^T S_W w}$
- Решение этой задачи сводится к нахождение собственных векторов и собственных значений для матрицы $S_W^{-1} S_B$, на основе которых и происходит проекция данных

- Метод опорных векторов
  - Используется как для классификации, так и для регрессии. Основная идея – найти гиперплоскость, которая максимально разделяет данные разных классов.
  - Функция гиперплоскости: $h(x) = w^T x + b$
  - Функция гиперплоскости $h(x)$ служит линейным классификатором или линейным дискриминантом, который предсказывает класс y для любой заданной точки x в соответствии с решающим правилом:

$$y = \begin{cases} +1, if\ h(x) > 0 \\ -1, if\ h(x) < 0 \end{cases}$$

  - Опорный вектор x* - точка, которая лежит точно на границе классификатора и, таким образом, удовлетворяет условию:

$$\delta^* = \frac{y^*(w^T x^* + b)}{\|w\|}$$

Составление обучающей и тестирующей выборки.

Для каждого из вышеописанных методов было 3 раза (с использованием различных значений параметра random_state у функции train_test_split) проведено исследование, которое заключалось в изменении размера тестовой выборки от 0.05 до 0.95 размера исходного датасета и замере точности модели. Результаты этого исследования с random_state=0 приведены на рисунках 6-9.

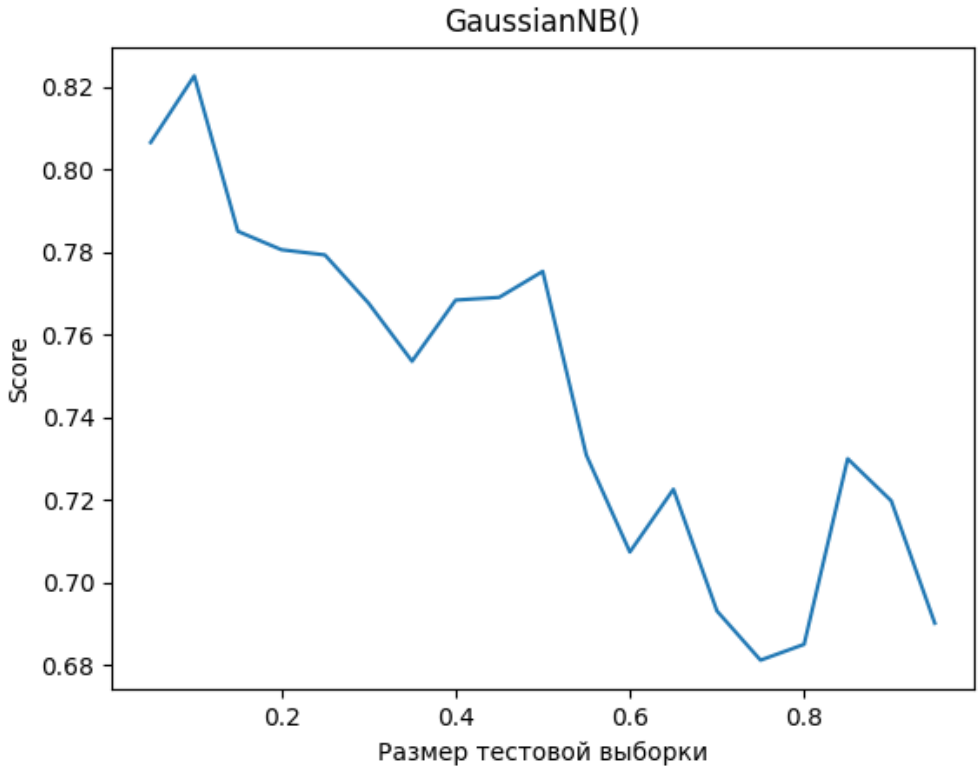


Рисунок 6 – График зависимости точности от размера тестовой выборки для GaussianNB()

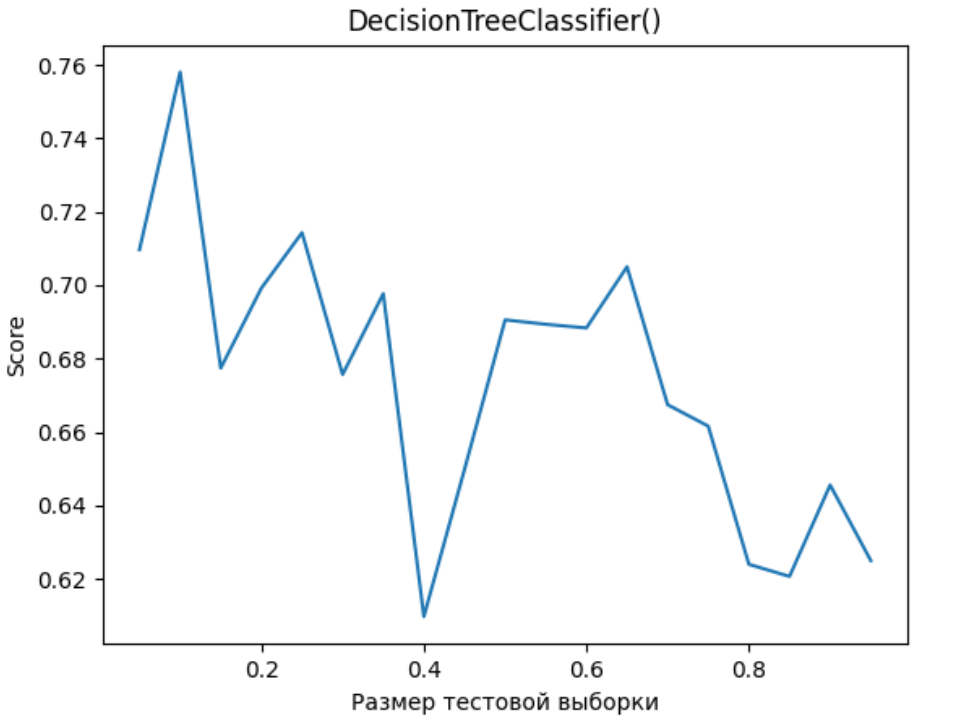


Рисунок 7 – График зависимости точности от размера тестовой выборки для DecisionTreeClassifier()

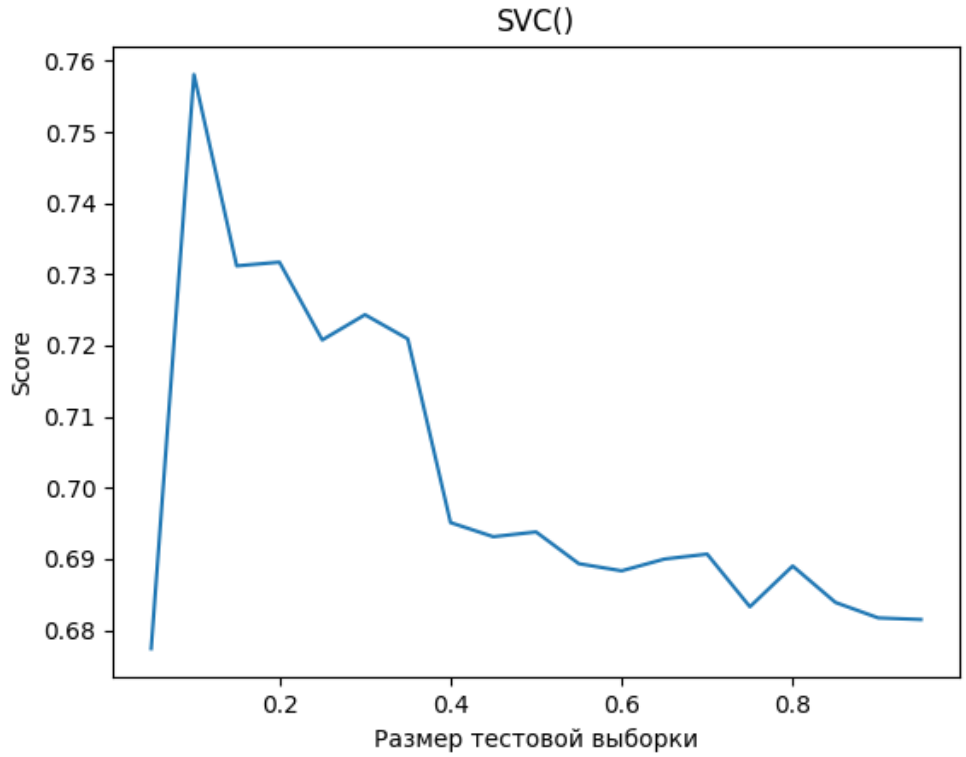


Рисунок 8 – График зависимости точности от размера тестовой выборки для SVC()

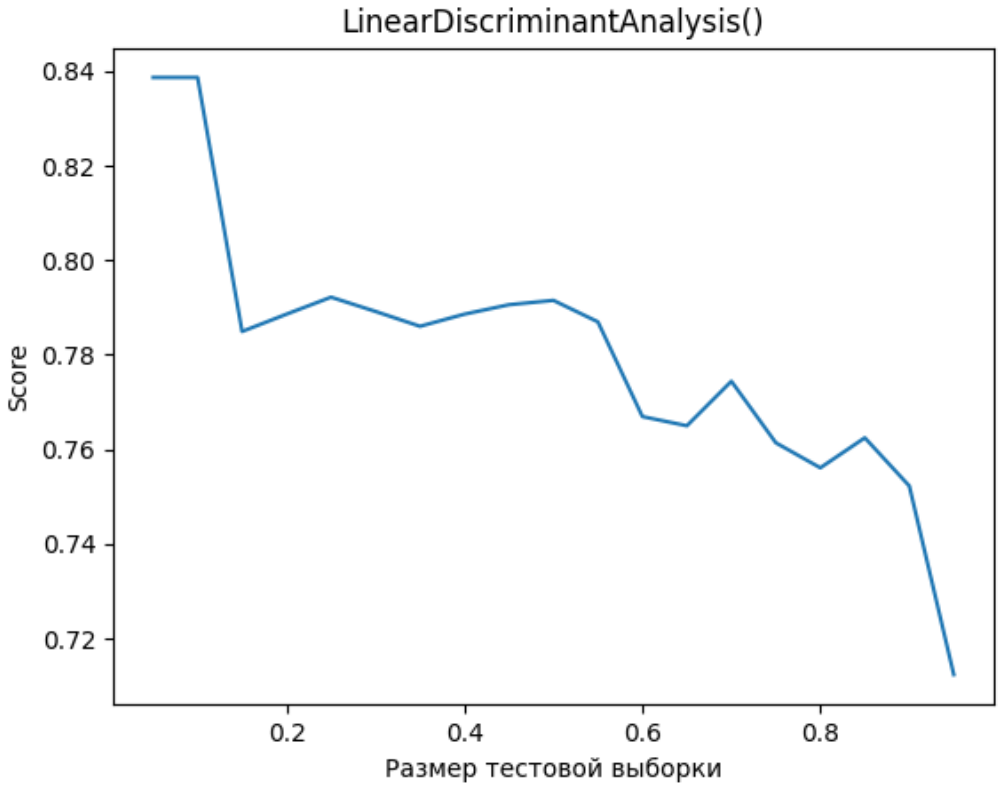


Рисунок 9 – График зависимости точности от размера тестовой выборки для LinearDiscriminantAnalysis()

Таким образом, для каждого из методов были подобраны размеры тестовых выборок, которые способствуют наилучшей точности модели. Эти данные представлены на рисунке 10.

```
Random state = 0
GaussianNB(): max_value = 0.8225806451612904, size = 0.1
DecisionTreeClassifier(): max_value = 0.7272727272727273, size = 0.25
SVC(): max_value = 0.7580645161290323, size = 0.1
LinearDiscriminantAnalysis(): max_value = 0.8387096774193549, size = 0.05
-----------------
Random state = 30410
GaussianNB(): max_value = 0.8548387096774194, size = 0.1
DecisionTreeClassifier(): max_value = 0.8064516129032258, size = 0.05
SVC(): max_value = 0.7741935483870968, size = 0.05
LinearDiscriminantAnalysis(): max_value = 0.8548387096774194, size = 0.1
-----------------
Random state = 8723
GaussianNB(): max_value = 0.7558139534883721, size = 0.7000000000000001
DecisionTreeClassifier(): max_value = 0.7741935483870968, size = 0.05
SVC(): max_value = 0.7272727272727273, size = 0.25
LinearDiscriminantAnalysis(): max_value = 0.7570498915401301, size = 0.7500000000000001
-----------------
```

Рисунок 10 – Размеры тестовых выборок, способствующие лучшей точности, для каждого метода

Результаты исследования.

Далее для каждого значения параметра random_state был определён метод и размер тестовой выборки, демонстрирующие наилучшую точность (рисунок 11).

```
Random state = 0
Best method = LinearDiscriminantAnalysis(), best size = 0.05
------------------
Random state = 30410
Best method = GaussianNB(), best size = 0.1
------------------
Random state = 8723
Best method = LinearDiscriminantAnalysis(), best size = 0.7500000000000001
------------------
```

Рисунок 11 – Методы и размеры тестовых выборок, демонстрирующие наилучшую точность

Как можно заметить, лучшие результаты показали GaussianNB() и LDA()

Анализ результатов исследования.

Для каждого из вышеприведенных методов было проведено вычисление Precision, Recall и F1-score.

Для придания формулам компактного вида введены следующие обозначения:

1. True Positive (TP) - данные, которые модель определила, как “положительные” и которые на самом деле являлись положительными;
2. False Positive (FP) - данные, которые модель определила, как “отрицательные” и которые на самом деле являлись положительными;
3. True Negative (TN) - данные, которые модель определила, как “негативные” и которые на самом деле являлись негативными;
4. False Negative (FN) – данные, которые модель определила, как “негативные” и которые на самом деле являлись положительными.

Соответственно:

- Precision
    - Precision определяется формулой: TP/(TP+FP);
    - Таким образом, Precision - отношение количества верно распознанных “положительных” объектов к общему количеству объектов, помеченных как “положительные”;
    - Однако, как можно заметить из формулы, Precision не учитывает “отрицательные” данные, что позволяет на специально размеченных датасетах получать высокие значения метрики (если TP=1, а остальные данные - FN, то Precision всё равно будет равен 1).
- Recall
    - Recall определяется формулой: TP/(TP+FN);

- Таким образом, Recall - отношение количества верно распознанных “положительных” объектов к общему количеству “положительных” объектов в выборке;
- Однако, если модель отметила все данные, как TP и FP, то Recall будет равен 1, что в свою очередь не будет означать стопроцентную точность модели.

- F1-score
  - Чтобы совместить в одной метрике значения и Precision, и Recall была создана данная создана формула: (B^2+1)PR/(B^2 P+R). В том случае, когда B=1, данная метрика носит название - гармоническое среднее (F1);
  - Использование сразу двух метрик помогает избавиться от случаев, когда на определенных датасетах модель могла показывать чрезвычайно высокие показатели.

На рисунках 12-14 представлены значения данных метрик для использованных методов.

```
Random state = 0
Best method = LinearDiscriminantAnalysis(), best size = 0.05
              precision    recall  f1-score   support

           0       0.78      0.70      0.74        10
           1       0.86      0.90      0.88        21

    accuracy                           0.84        31
   macro avg       0.82      0.80      0.81        31
weighted avg       0.84      0.84      0.84        31
```

Рисунок 12 – Значения метрик для random_state = 0

```
Random state = 30410
Best method = GaussianNB(), best size = 0.1
              precision    recall  f1-score   support

           0       0.88      0.47      0.61        15
           1       0.85      0.98      0.91        47

    accuracy                           0.85        62
   macro avg       0.86      0.72      0.76        62
weighted avg       0.86      0.85      0.84        62
```

Рисунок 13 – Значения метрик для random_state = 30410

```
Random state = 8723
Best method = LinearDiscriminantAnalysis(), best size = 0.7500000000000001
              precision    recall  f1-score   support

           0       0.63      0.50      0.56       141
           1       0.80      0.87      0.83       320

    accuracy                           0.76       461
   macro avg       0.71      0.68      0.69       461
weighted avg       0.75      0.76      0.75       461
```

Рисунок 14 – Значения метрик для random_state = 8723

Как можно заметить на рисунках 12-14 класс «1» определяется с большей точностью, чем класс «0».

Возникшие проблемы.

Как было описано в разделе, посвящённому датасету, в наборе данных отсутствовали данные по некоторым из признаков, и чтобы восполнить пробел в данных, на их место были добавлены средние значения по каждому из признаков.

Предложения по улучшению модели.

При вычислении результатов исследования были использованы стандартные реализации методов из библиотеки scikit-learn со стандартными

параметрами. При подборе параметров точность классификации может увеличиться.

Заключение.

В ходе выполнения ИДЗ были найдены методы, при использовании которых точность классификации достигает максимума. Наилучшие результаты показали наивный байесовский алгоритм с точностью до 85% и линейный дискриминантный анализ с точностью до 84%.

Для достижения данного результата были выполнены следующие действия: проанализирован и дополнен датасет; категориальные признаки в датасете были заменены на числовые для использования в моделях; рассмотрены 4 метода классификации: наивный байесовский алгоритм, решающие деревья, метод опорных векторов и линейный дискриминантный анализ; для каждого из методов была вычислена точность при различных размерах тестовой выборки; затем лучшие из методов были проанализированы с помощью метрик precision, recall и f1-score по классификации каждого из классов.

Также в рамках исследования были описаны проблемы, возникшие при решении задач, и способы их решения, а также выдвинуты предложения по улучшению модели.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.py

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.svm import SVC
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import classification_report

pd.set_option('display.max_columns', None)

train = pd.read_csv('train.csv')
# print(train)
train = train.iloc[:, 1:13] # Удаление Loan_ID

na = train.isna().sum().sort_values(ascending=False) # Поиск количества
NaN-значений для каждого изи признаков
# print(na)

# Замена NaN значений на средние
for column, na_count in na.items():
    if na_count > 0:
        if train.dtypes[column] == 'float64':
            train[column]                                                =
train[column].fillna(float(int(train[column].mean()))) # mean для числовых
признаков
        else:
            train[column] = train[column].fillna(train[column].mode()[0])
# мода для категориальных

# print(train.isna().sum().sort_values(ascending=False))

# Замена категориальных признаков на числовое представление
categorical_features = ['Gender', 'Married', 'Dependents', 'Education',
'Self_Employed', 'Property_Area', 'Loan_Status']
for feature in categorical_features:
    train[feature] = LabelEncoder().fit_transform(train[feature])

# print(train)

stats = train.describe() # Статистический анализ датасета
# print(stats)

# отделение признаков от меток классов
X = train.iloc[:, 1:11].to_numpy()
Y = train.iloc[:, 11].to_numpy()

sizes = np.arange(start=0.05, stop=1, step=0.05)

def score_size(m, train_x, test_x, train_y, test_y):
```

```
    m.fit(train_x, train_y)
    scores.append(m.score(test_x, test_y))


methods = [GaussianNB(), tree.DecisionTreeClassifier(), SVC(),
LinearDiscriminantAnalysis()]
score_size_df = pd.DataFrame()
score_size_df['size'] = sizes

# Случайные состояния
states = [0, 30410, 8723]
for state in states:
    for method in methods:
        scores = []
        for size in sizes:
            X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=size, random_state=state) # Подготовка обучающей и тестовой
выборки
            score_size(method, X_train, X_test, y_train, y_test) #
вычисление точности

        score_size_df[method] = scores
        # plt.plot(sizes, scores)
        # plt.title(str(method))
        # plt.xlabel("Размер тестовой выборки")
        # plt.ylabel("Score")
        # plt.show()

    print(f"Random state = {state}")

    # Поиск размеров выборок для каждого метода, при которых точность
наилучшая
    max_values = {}
    for method in methods:
        max_index = score_size_df[method].idxmax()
        max_values[method] = {
            'max_value': score_size_df.loc[max_index, method],
            'size': score_size_df.loc[max_index, 'size']
        }

    best_method = GaussianNB()
    best_score = 0
    best_size = None

    # Поиск наилучшей комбинации метода и размера выборки для каждого из
random_state
    for method, values in max_values.items():
        if values['max_value']>best_score:
            best_method = method
            best_score = values['max_value']
            best_size = values['size']
        # print(f"{method}: max_value = {values['max_value']}, size =
{values['size']}")

    print(f"Best method = {best_method}, best size = {best_size}")
```

```
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=best_size, random_state=state)
    best_method.fit(X_train,y_train)
    y_pred = best_method.predict(X_test)
    print(classification_report(y_test,y_pred)) # Анализ по Precision,
Recall, F1
    print("------------------")
```