МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Допущен к защите

Заведующий кафедрой ПМИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Разова

**Исследование моделей кредитного скоринга**

Курсовой проект по дисциплине  
«Машинное обучение»

Выполнил студент группы ПИм-1301-03-00     / М.Д. Куликов /

Руководитель д.т.н., профессор кафедры ПМИ     / Е.В. Котельников /

Работа защищена с оценкой     \_\_\_\_.\_\_\_\_.2024 г.

Члены комиссии:   /     /

    /     /

Киров 2024

Содержание

[1 Введение 3](#_Toc163492111)

[1 Подготовка данных 5](#_Toc163492112)

[1.1 Предобработка признаков 5](#_Toc163492113)

[1.1.1 Описание исходных данных 5](#_Toc163492114)

[1.1.2 Предобработка данных 6](#_Toc163492115)

[1.1.3 Отбор признаков. 11](#_Toc163492116)

[1.1.4 Устранение дисбаланса в данных. 11](#_Toc163492117)

[1.2 Выводы по разделу 12](#_Toc163492118)

[2 Сравнение различных моделей 13](#_Toc163492119)

[2.1 Обучение и тестирование моделей 13](#_Toc163492120)

[2.1.1 Кросс-валидация 13](#_Toc163492121)

[2.1.2 Тестирование лучших моделей на тестовой выборке 15](#_Toc163492122)

[2.2 Выводы по разделу 16](#_Toc163492123)

[3 Заключение 17](#_Toc163492124)

[Библиографический список 18](#_Toc163492125)

[Приложения 19](#_Toc163492126)

[Приложение А. Листинги программы 19](#_Toc163492127)

# Введение

В современном мире, где финансовые кредиты играют довольно большую роль в жизни и финансовой устойчивости многих людей, важно обеспечить эффективное управление рисками, связанными с выдачей кредитов. Одним из наиболее важных инструментов для этой цели является кредитный скоринг, который позволяет оценить вероятность возврата кредита заёмщиком на основе различных характеристик и истории его кредитного поведения. И, несмотря на уже существующие модели и методы в данной области, вопрос о повышении эффективности кредитного скоринга и адаптации для нужд конкретных банков до сих пор остаётся открытым.

Таким образом, проблема данного исследования заключается в отстутствии универсального метода для предсказания кредитного скоринга с высокой эффективностью и необходимости исследования и сравнения различных моделей с целью определения наиболее точной.

Целью курсового проекта является обучение нескольких моделей кредитного скоринга и их сравнение.

Для достижения поставленной цели необходимо решить следующие задачи:

* Изучить исходные данные.
* Провести анализ данных и выполнить их предобработку.
* Обучить различные модели машинного обучения на основе выбранного набора данных.
* Провести сравнительный анализ эффективности и точности различных моделей.

Курсовой проект включает два основных раздела. В первом будут описаны методы, использованные для предобработки данных и отбора признаков. Во втором разделе будет описано обучение и сравнение различных моделей машинного обучения.

# Подготовка данных

## Предобработка признаков

### Описание исходных данных

Исходные данные представляли собой csv файл, содержащий 100 тысяч записей о взятых кредитах. Данные содержали 27 признаков, а именно:

1. ID – уникальный идентификатор записи
2. Customer\_ID - уникальный идентификатор человека
3. Month – месяц года
4. Name – имя человека
5. Age - возраст человека
6. SSN - номер социального страхования человека
7. Occupation - род занятий человека
8. Annual\_Income - годовой доход человека
9. Monthly\_Inhand\_Salary - ежемесячная базовая зарплата человека
10. Num\_Bank\_Accounts - количество банковских счетов человека
11. Num\_Credit\_Card - количество других кредитных карт, имеющихся у человека
12. Interest\_Rate – процентная ставка по кредитной карте
13. Num\_of\_Loan - количество кредитов, взятых в банке
14. Type\_of\_Loan - типы кредитов, взятых человеком
15. Delay\_from\_due\_date - среднее количество дней задержки с даты платежа
16. Num\_of\_Delayed\_Payment - среднее количество платежей, задержанных человеком
17. Changed\_Credit\_Limit - процентное изменение лимита кредитной карты
18. Num\_Credit\_Inquiries - количество запросов по кредитной карте
19. Credit\_Mix - классификация структуры кредитов
20. Outstanding\_Debt - оставшийся долг, который необходимо выплатить (в долларах США)
21. Credit\_Utilization\_Ratio - коэффициент использования кредитной карты
22. Credit\_History\_Age - возраст кредитной истории человека
23. Payment\_of\_Min\_Amount - была ли выплачена человеком только минимальная сумма
24. Total\_EMI\_per\_month - ежемесячные платежи EMI (в долларах США)
25. Amount\_invested\_monthly - ежемесячная сумма, инвестированная клиентом (в USD)
26. Payment\_Behaviour - платежное поведение клиента (в USD)
27. Monthly\_Balance - ежемесячная сумма баланса клиента (в USD)

Целевой переменной являлся кредитный скоринг заёмщика, принимающий значения Poor, Standard и Good (задача классификации).

### Предобработка данных

Для начала данные были разделены на обучающую и тестовую выборку с помощью функции библиотеки sklearn[1] train\_test\_split[2]. Обучающая выборка составила 80 процентов данных, тестовая оставшиеся 20.

Были скорректированы признаки, содержащие неправильную информацию. Например у пользователей, возраст которых был написан неправильно, например с лишними символами, он был скорректирован, а у тех, которых по какой-то причине отстутствовал, был взят из информации о предыдущих взятых кредитах. Код можно посмотреть в приложении А.1 (Метод замены неправильных значений признака).

Было замечено, что совсем юные заёмщики имеют плохой кредитный скоринг, а заёмщики с возрастом более 45 стандартный или хороший (рисунки 1.1-1.4):

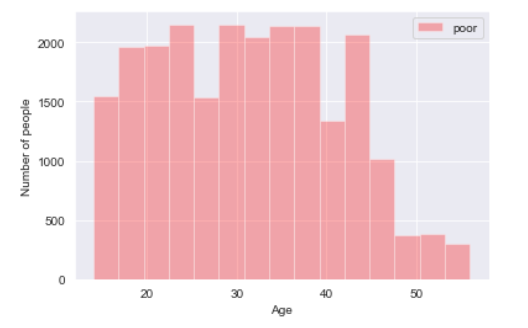


Рисунок 1.1 – Распределение по возрасту у заёмщиков с плохим кредитным скорингом

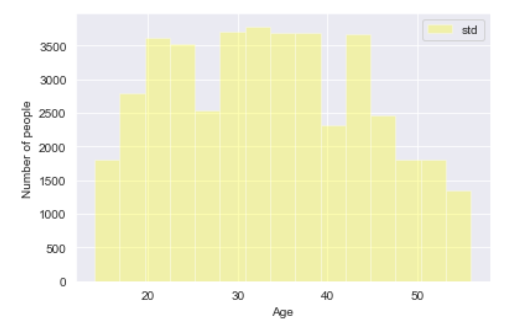


Рисунок 1.2 – Распределение по возрасту у заёмщиков со стандартным кредитным скорингом

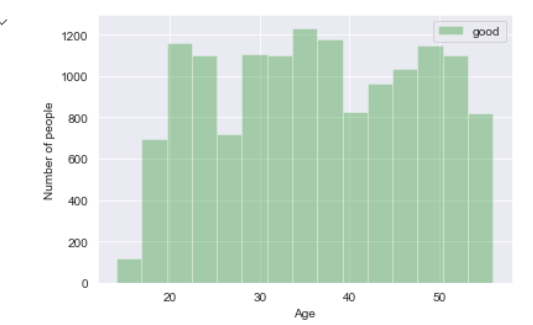


Рисунок 1.3 – Распределение по возрасту у заёмщиков с хорошим кредитным скорингом

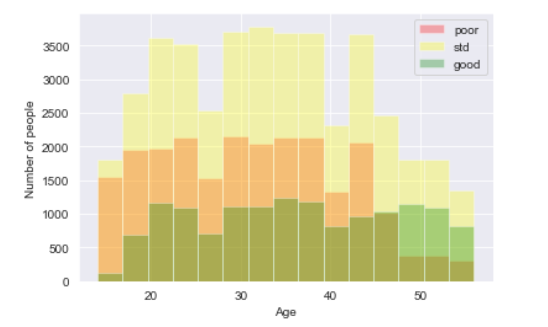


Рисунок 1.4 – Распределение по возрасту у всех заёмщиков

В результате признак Age был закодирован методом one-hot encoding[3] в несколько категорий, а именно Age\_Teenager, Age\_Youth, Age\_Young\_Adult, Age\_Adult, Age\_Senior.

Были отформатированы такие признаки, как Annual\_Income, Num\_of\_Loan, Num\_Bank\_Accounts, Num\_of\_Delayed\_Payment (рисунок 1.5):

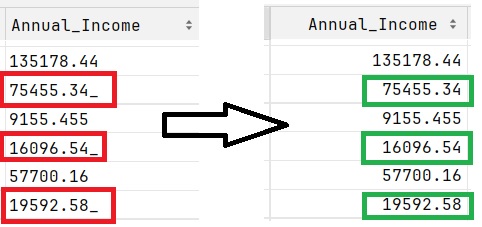


Рисунок 1.5 – Пример форматирования признака Annual\_Income

Credit\_History\_Age был преобразован из строки в float (рисунок 1.6):

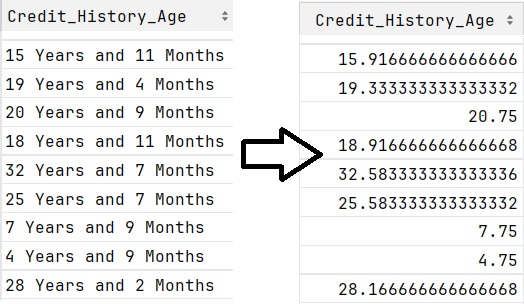


Рисунок 1.6 – Пример преобразования возраста в тип float

Кроме того, были закодированы методом one-hot encoding такие категориальные признаки, как Occupation, Age\_categories, Credit\_Mix, Payment\_Behaviour, Payment\_of\_Min\_Amount, что добавило 16 новых признаков:

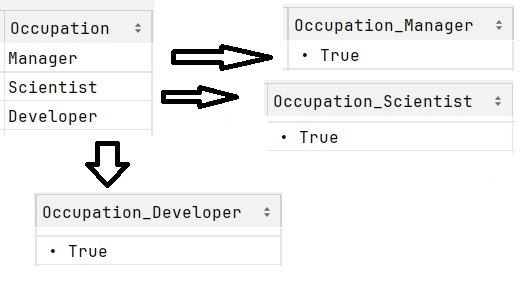


Рисунок 1.7 – Пример кодирования признака Occupation

Также был закодирован признак Type\_of\_Loan (рисунок 1.8), в результате чего прибавилось ещё 9 признаков, код можно посмотреть в приложении А.2:

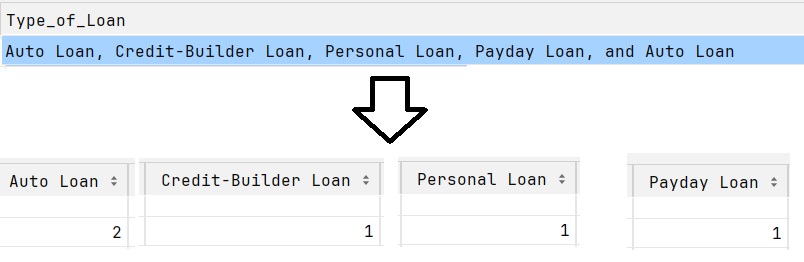


Рисунок 1.8 – Пример кодирования признака Type\_of\_Loan

Были удалены выбросы (код удаления выбросов можно посмотреть в приложении A.3) и ненужные признаки, такие как Name, Month, ID, SSN, Customer\_ID, а также коррелирующие с другими (Monthly\_Inhand\_Salary, коррелировавший с Annual\_Income), в результате чего количество признаков вместе с новыми составило 59.

### Отбор признаков.

Для определения нерелевантных признаков были использованы коэффициенты корреляции Пирсона [4], ANOVA F-value [5] и RFECV [6] (код можно посмотреть в приложении A.4).

Лучше всего себя показал метод RFECV, и в итоге были удалены ещё 7 признаков, а именно Annual\_Income, Outstanding\_Debt, Amount\_invested\_monthly, Monthly\_Balance, Total\_EMI\_per\_month, Credit\_Utilization\_Ratio, Occupation\_Entrepreneur, и осталось 52 признака.

### Устранение дисбаланса в данных.

Так как более чем половина объектов как в обучающей, так и в тестовой выборке имеет стандартный скоринг (распределия можно посмотреть на рисунках 1.9 и 1.10 ниже), был устранён дисбаланс в данных методом апсемплинга для объектов из классов Poor и Good (рисунок 1.11), в результате значение f1 macro для логистической регрессии на кросс-валидации возросло с 0.618 до 0.683:

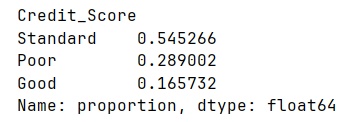


Рисунок 1.9 – Распределение по кредитному скорингу на обучающей выборке

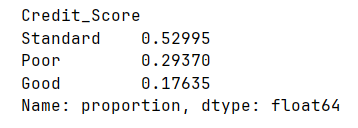


Рисунок 1.10 – Распределение по кредитному скорингу на тестовой выборке

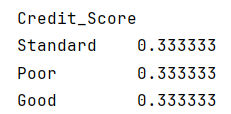


Рисунок 1.11 – Распределение по кредитному скорингу на обучающей выборке после апсемплинга

## Выводы по разделу

В итоге, после предварительной обработки данных, было удалено множество нерелевантных признаков, а также были сформированы новые признаки, устранён дисбаланс в данных. Итоговое количество признаков составило 52.

# Сравнение различных моделей

## Обучение и тестирование моделей

### Кросс-валидация

Было проведено обучение и тестирование моделей на кросс-валидации [7] (3-fold). Были попробованы такие модели, как LogisticRegression, SVC, KNeighborsClassifier, MultinomialNB, DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, BaggingClassifier на своих параметрических сетках. Результаты получились следующие (таблица 2.1):

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | F1 macro | Метод скейлинга | Параметры |
| BaggingClassifier | 0.892 | StandardScaler | n\_estimators:100 |
| BaggingClassifier | 0.891 | MinMaxScaler | n\_estimators:100 |
| RandomForestClassifier | 0.888 | StandardScaler | criterion:'gini', max\_features:'sqrt', min\_samples\_leaf:1, min\_samples\_split:2, n\_estimators:100 |
| RandomForestClassifier | 0.888 | MinMaxScaler | criterion:'gini', max\_features:'sqrt', min\_samples\_leaf:1, min\_samples\_split:2, n\_estimators:100 |
| KNeighborsClassifier | 0.862 | StandardScaler | algorithm:'auto', n\_neighbors:2,  p:1,  weights:'distance' |
| DecisionTreeClassifier | 0.846 | MinMaxScaler | criterion:'entropy', max\_depth:None, max\_features:'sqrt', min\_samples\_leaf:1, min\_samples\_split:2, dt\_\_splitter:'best' |
| DecisionTreeClassifier | 0.846 | StandardScaler | criterion:'entropy', max\_depth:None, max\_features:'sqrt', min\_samples\_leaf:1, min\_samples\_split:2, dt\_\_splitter:'best' |
| SVC | 0.841 | StandardScaler | C:50 |
| KNeighborsClassifier | 0.834 | MinMaxScaler | algorithm:'auto', n\_neighbors:2,  p:1,  weights:'distance' |
| SVC | 0.802 | MinMaxScaler | C:50 |
| GradientBoostingClassifier | 0.730 | StandardScaler | learning\_rate:1.0, max\_features:'sqrt' |
| GradientBoostingClassifier | 0.730 | MinMaxScaler | learning\_rate:1.0, max\_features:'sqrt' |
| LogisticRegression | 0.683 | MinMaxScaler | C:1,  penalty:'none' |
| LogisticRegression | 0.683 | StandardScaler | C:1,  penalty:'l2' |
| MultinomialNB | 0.649 | MinMaxScaler | alpha:2.0, fit\_prior:True |

Таблица 2.1 – Сравнение результатов различных моделей на кросс-валидации (3-fold)

Как мы можем видеть из таблицы 2.1 лучше всего себя показал bagging из деревьев решений, на втором месте расположился random forest, а на третьем knn.

### Тестирование лучших моделей на тестовой выборке

Результаты на тестовых данных у моделей, показавших себя лучше всего на кросс-валидации, получились следующими (таблица 2.2). Видно, что лучше всего показал себя KNN, который довольно неплохо предсказывал все три класса:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | F1 Good | F1 Poor | F1 Standard | F1 macro |
| BaggingClassifier  (100 DecisionTree) (standard scaling) | 0.57 | 0.08 | 0.68 | 0.44 |
| BaggingClassifier  (100 DecisionTree) (minmax scaling) | 0.57 | 0.06 | 0.67 | 0.43 |
| RandomForestClassifier  (standard scaling) | 0.59 | 0.07 | 0.69 | 0.45 |
| RandomForestClassifier  (minmax scaling) | 0.58 | 0.02 | 0.67 | 0.42 |
| KNeighborsClassifier  (standard scaling) | 0.49 | 0.53 | 0.70 | **0.57** |

Таблица 2.2 – Сравнение результатов лучших моделей на тестовых данных

## Выводы по разделу

Таким образом, наилучшей моделью оказалась KNN. Она довольно хорошо показала себя на кросс-валидации (5 место) и лучше всех показала себя на тестовой выборке.

# Заключение

В заключение, была проведена работа над набором данных по кредитному скорингу. Этот процесс включал в себя не только предварительную обработку данных, но и создание новых признаков. Также был произведён отбор наиболее информативных признаков, что позволило уменьшить размерность данных и повысить эффективность моделей.

Основным этапом исследования было тестирование различных моделей машинного обучения на предсказательной способности. Наилучшие результаты показала модель k-ближайших соседей (KNN), что подтверждает её эффективность в решении задачи кредитного скоринга.

# Библиографический список

1. Библиотека машинного обучения на языке Python [Электронный ресурс]. URL: <https://scikit-learn.org>.
2. Метод train\_test\_split из sklearn [Электронный ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>.
3. Raschka S. Python Machine Learning, 2015 – P. 106-108
4. Cohen, I., Huang, Y., Chen, J., Benesty, J., Benesty, J., Chen, J., Huang, Y., Cohen, I. Pearson correlation coefficient // Noise reduction in speech processing. 2009 – P. 1-4.
5. Lars St., Wold S. Analysis of variance (ANOVA) // Chemometrics and intelligent laboratory systems. 1989. Vol. 6. No. 4 – P. 259-272.
6. Класс RFECV из sklearn [Электронный ресурс]. URL: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html>.
7. Berrar, D. Cross-validation // Encyclopedia of bioinformatics and computational biology. 2019. Vol. 1 – P. 542-545.

# Приложения

## Приложение А. Листинги программы

**1. Метод замены неправильных значений признака:**

def find\_correct\_column\_value\_by\_id(data, id, column, wrong\_value):  
 data\_of\_id = data[(data["Customer\_ID"] == id) & (data[column] != wrong\_value)]  
 if len(data\_of\_id) > 0:  
 return data\_of\_id[column].value\_counts().head(1).index[0]  
 else:  
 return np.NaN  
  
def is\_convertible\_to\_int(value):  
 try:  
 int(value)  
 return True  
 except ValueError:  
 return False  
  
def correct\_ages(train\_data, test\_data):  
 train\_data["Age"] = train\_data.apply(lambda el: re.findall(r'-?\d+', el["Age"])[0], axis=1)  
 test\_data["Age"] = test\_data.apply(lambda el: re.findall(r'-?\d+', el["Age"])[0], axis=1)  
 all\_data = pd.concat([train\_data, test\_data])  
   
 for i in range(len(train\_data)):  
 if int(train\_data["Age"][i]) < 14 or int(train\_data["Age"][i]) > 80:  
 age = find\_correct\_column\_value\_by\_id(all\_data, train\_data["Customer\_ID"][i], "Age", train\_data["Age"][i])  
 train\_data["Age"][i] = age  
   
 for i in range(len(test\_data)):  
 if int(test\_data["Age"][i]) < 14 or int(test\_data["Age"][i]) > 80:  
 age = find\_correct\_column\_value\_by\_id(all\_data, test\_data["Customer\_ID"][i], "Age", test\_data["Age"][i])  
 test\_data["Age"][i] = age  
   
 return train\_data, test\_data  
*#%%*train\_data, test\_data = correct\_ages(train\_data, test\_data)

**2. Кодирование взятых ранее займов:**

def extract\_loan\_info(row):  
 if pd.isna(row["Type\_of\_Loan"]):  
 return np.nan  
 else:  
 answer = {}  
 all\_person\_loans = np.array(re.findall(r'[A-Z][a-z]+[-\s]?[A-Z]\*[a-z]\*\s[L]\*[A-Za-z]\*', row["Type\_of\_Loan"]))  
 for loan in all\_person\_loans:  
 if loan not in answer:  
 answer[loan] = 1  
 else:  
 answer[loan] += 1  
 return answer  
  
  
def process\_loan\_data(data):  
 data["Taken\_Loans"] = data.apply(extract\_loan\_info, axis=1)  
 return data  
  
  
train\_data = process\_loan\_data(train\_data)  
test\_data = process\_loan\_data(test\_data)  
  
additional\_columns : pd.DataFrame = train\_data["Taken\_Loans"].apply(pd.Series).fillna(0).astype(int)  
print(list(additional\_columns.keys()))  
train\_data = pd.concat([train\_data, additional\_columns], axis=1)  
additional\_columns : pd.DataFrame = test\_data["Taken\_Loans"].apply(pd.Series).fillna(0).astype(int)  
print(list(additional\_columns.keys()))  
test\_data = pd.concat([test\_data, additional\_columns], axis=1)  
  
train\_data.drop(["Type\_of\_Loan", "Taken\_Loans"], axis=1, inplace=True)  
test\_data.drop(["Type\_of\_Loan", "Taken\_Loans"], axis=1, inplace=True)

**3. Удаление выбросов:**

def remove\_outliers\_iqr(data):  
 numeric\_cols = data.select\_dtypes(include=['number']).columns  
 print(numeric\_cols)  
   
 Q1 = data[numeric\_cols].quantile(0.25)  
 Q3 = data[numeric\_cols].quantile(0.75)  
 IQR = Q3 - Q1  
  
 outliers\_mask = ((data[numeric\_cols] < (Q1 - 1.5 \* IQR)) | (data[numeric\_cols] > (Q3 + 1.5 \* IQR))).any(axis=1)  
  
 data = data[~outliers\_mask]  
  
 return data

**4. Отбор оптимального количества признаков с помощью RFECV:**

from sklearn.feature\_selection import RFECV  
  
rfecv = RFECV(  
 estimator=model,  
 scoring="accuracy",  
 n\_jobs=-1,  
 min\_features\_to\_select=10  
)  
rfecv.fit(train\_x, train\_y)  
print(f"Optimal number of features: {rfecv.n\_features\_}")  
rfecv.support\_