CONTENT

INTRODUCTION

Financial institutes like banking are always essential in both global and local economic development. For the past 40 years share of households' loans and debts in GDP in most countries are constantly growing according to International Monetary Fund statistic (2022). Credit is shown to be likely reliable instrument for people.

On the other hand, banks have to generate revenue. To avoid risks they need to asses each customer and decide whether to borrow money or not. For this purpose, managers collect different information, as a result calculating credit score as Hooman A. et al stayed (2016). Non-repayment of loans reduces turnover, leads to loss of funds, and the loan is not issued to a new client. Every bank is faced the problem of assessing the reliability of potential customers to be financial stable and attractive (Florez-Lopez R. and Ramon-Jeronimo J.M., 2014).

Researchers claimed that at the end there is huge amount of data to be analyzed to make a decision of borrowing (Hooman A. et al, 2016). That is why banks are tend to use modern and efficient technology of Machine Learning such as Discriminant Analysis, Logistic Regression, K-Nearest Neighbor, Bayesian Classifier, Decision Tree and others, that helps to apply sorting methods, risk assessment and some others.

The goal of this work is to provide recommendations for the Artificial Bank for its more financial sustainability, avoiding credit risk using Logistic Regression and K-Nearest Neighbors.

The object of this study is a process of credit risk score evaluation in a banking system. And the subject is an Artificial Bank for study purposes. To analyze the current state of the credit risk evaluation process, Logistic Regression and K-Nearest Neighbors are used as main research methods along with visualization data analysis.

CHAPTER ONE. Problem and dataset

1. Problem formulation

Many banks now offer a wide variety of credit services. Loans can be issued for various purposes, and interest on loans is the main source of income for banks. Therefore, in order not to lose profits, the flow of credit payments must be constant (Miguéis V.L., Benoit D.F. and Van den Poel D. 2013). But how can banks ensure this? The answer is a client's credit risk assessment. Thus, for example, the researchers studied Machine Learning classifiers techniques based on probability of default value. As a result, they got promising methods, providing accurate credit risk scores (Florez-Lopez R. and Ramon-Jeronimo J.M. 2014, Hooman A. et al 2016).

The researchers Subburayan B. et al emphasize the inevitability of the transformation of the banking sector with the help of Machine Learning (2023). For example, quality assessment, bank transactions trends, client traffics. It leads to increasing in customers happiness and income.

Although, one of the most important factors in banking operation system is risk management. It causes improvements of bank's loan portfolio, decreases credit losses from unsecured customers, and even lower default rates. Also, Witzel M., and Bhargava N. claim that the implementation is risk-driven approaches based on Machine Learning may lead not only to time savings and big data problems solutions, but also to reputational and financial impacts (2023).

In addition, banks have limited information and at the same time must be confident in its reliability. Even if the client has a stable job and has bought a house, this is not enough. There are many factors that need to be taken into account. At the same time, all banks collect different information about their potential customers. Therefore, each such financial institution needs to develop its own methods to assess credit risks.

From all this it follows that banks face an important problem: how to assess the credit risk for each new customer with information about their past default experience? This work provides an answer to this question.

2. Dataset description and transforming

This research is based on a dataset extracted from a website https://www.kaggle.com, the name is 'Credit Risk Analysis' (2019). It contains 700 samples with 9 features. These features are:

1. age – customer age, years;

2. ed – customer education level, coded (1 – university degree, 2 – high school, 3 – illiterate, 4 – basic, 5 – professional course)
3. employ – time with current employer, years;
4. address – time at the current address, years;
5. income – annual income, USD;
6. debtinc – debt-to-income ratio, %;
7. creddebt – debt on the credit card, USD;
8. othdebt – other debts, USD;
9. default – indicates whether a customer defaulted in the past if = 1 than yes, if = 0 does not.

The analysis is based on the hypothesis that unpaid debts in the past leads to unpaid debts in the future. That is why the 'default' variable is dependent in the models in this paper. Moreover, it presents two classes of customers: reliable if 'default'=1 and unreliable if 'default'=0.

All further calculations are made in Python. The data is partly presented at the figure 1.

In order to prepare data few steps are made.

- to detect Nan values;
- to detect abnormal value, for example, age cannot be 136;
- to check all values are with expected type;

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41.0	3.0	17	12	176.0	9.3	11.359392	5.008608	1
1	27.0	1.0	10	6	31.0	17.3	1.362202	4.000798	0
2	40.0	1.0	15	7	NaN	5.5	0.856075	2.168925	0
3	41.0	NaN	15	14	120.0	2.9	2.658720	0.821280	0
4	24.0	2.0	2	0	28.0	17.3	1.787436	3.056564	1
..	...	...	...	...	...	...	...	...	...
695	36.0	2.0	6	15	27.0	4.6	0.262062	0.979938	1
696	29.0	2.0	6	4	21.0	11.5	0.369495	2.045505	0
697	33.0	1.0	15	3	32.0	7.6	0.491264	1.940736	0
698	45.0	1.0	19	22	77.0	8.4	2.302608	4.165392	0
699	37.0	1.0	12	14	NaN	14.7	2.994684	3.473316	0

[700 rows x 9 columns]

Figure 1. Partly illustration of the dataset

Figure 1 illustrates, the date is quantitative, numbers based. Moreover, there is missing data to be managed. In addition, the functions 'describe()' and info()' are used, see appendix 1.

Value ‘ed’ after the transformation has wrong datatype. Also, one column as object type, it is not expected since the ‘default’ are boolean values.

The following functions are used to manage the missing data.

- ‘fillna(dataset.mean())’ to replace ‘Nan’ missing data with their mean values;
- ‘astype(int)’ to make educational level integer type, because it is coded;
- ‘replace()’ (for abnormal ‘136’ years and ‘default’ string ‘ :0 ’) to make data valid;

The researched assumes that function ‘describe()’ provides not enough details, and decided to create its own table with more coefficients, see figure 2.

	age	ed	employ	address	income	debtinc	creddebt	othdebt
Mean	34.752	1.697	8.389	8.269	45.744	10.261	1.554	3.058
Harmonic Mean	33.011	1.354	0.000	0.000	32.733	5.687	0.352	1.143
Range	36.000	4.000	31.000	34.000	432.000	40.900	20.550	26.988
Median	34.000	1.000	7.000	7.000	36.000	8.600	0.855	1.988
Mode	29.000	1.000	0.000	2.000	45.744	4.500	0.402	7.823
Standard Deviation	7.853	0.919	6.653	6.817	36.411	6.822	2.116	3.285
Variance	61.667	0.845	44.266	46.468	1325.735	46.545	4.476	10.793
Skewness	0.374	1.252	0.829	0.941	3.953	1.094	3.890	2.722
Kurtosis	-0.535	0.881	0.222	0.322	27.114	1.201	21.815	10.247
Entropy	6.526	6.421	6.217	6.204	6.334	6.341	5.968	6.130

Figure 2. Self-formed information of the cleaned dataset

Average values are seen from the figure 2. ‘Default’ values are not included, because does not have any need. Skewness is mostly positive, Kurtosis also mostly have a very big and positive number, means the curves with high sharp peaks and long outliers. The rest have more flatter vertices and lighter tails. The entropy is 6.5-5, meaning a high degree of uncertainty or diversity.

CHAPTER TWO. Common and by classes visualization

1. Common visualization

For deep analysis and finding hidden patterns, a visual and statistical analysis of the data is carried out. For visualization, a matrix has been built with trend lines at the figure 3.

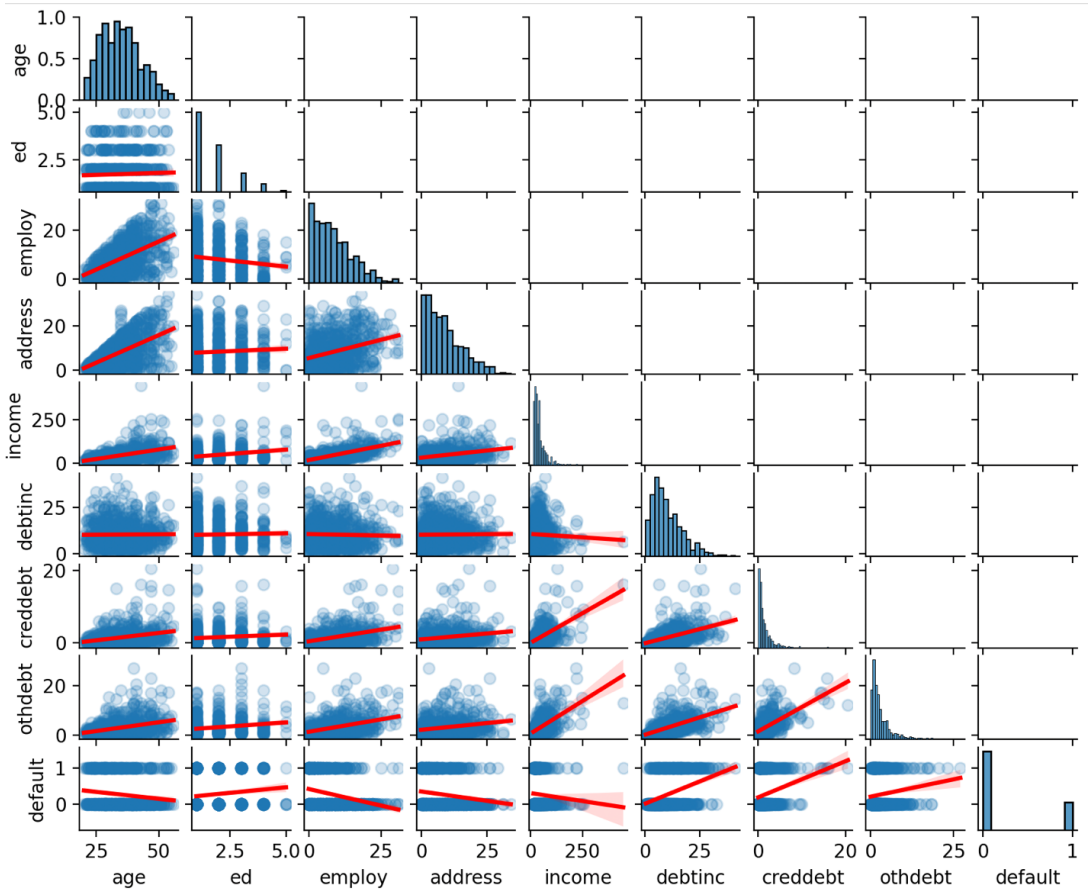

Figure 3. Scatter Plot Matrix for the whole data

Figure 3 illustrates that the spread of values is mostly significant. Majority of the trend lines are strongly or slightly positive, covariance (see appendix 2) and correlation explain more.

The covariance with ‘default’ value of greater interest. Figure 4 shows that older people who work and live in one place for a long time, having small credit card or other debts are of greater interest to the bank as potential and reliable customers. And the level of education most likely does not have a significant impact. In addition, the value ‘debtinc’ has the most positive effect, the less effect is from ‘income’.

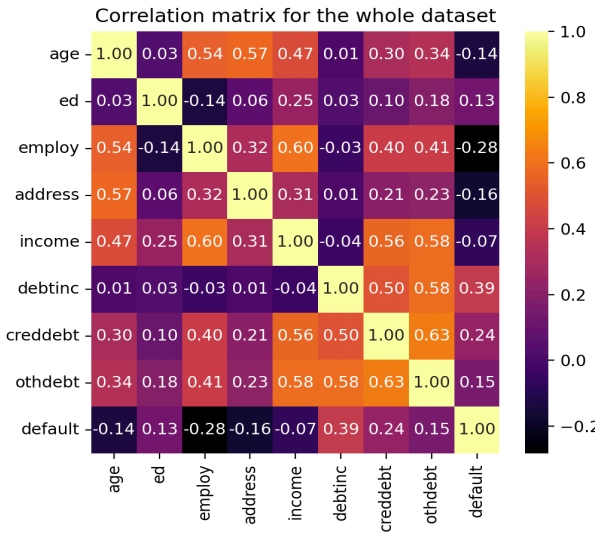

Figure 4. Correlation Matrix for the whole dataset

2. Visualization for classes

To learn more about the behavior patterns within reliable and unreliable customers, the metrics of these two groups is considered. Moreover, number of observations in a reliable group is 3 times more than in an unreliable one (183 and 517, respectively). Their trends are the same or differ slightly, see appendix 3. However, the distributions vary, see figure 5, 6.

	age	ed	employ	address	income	debtinc	creddebt	othdebt
Mean	35.395	1.627	9.509	8.932	47.350	8.679	1.245	2.773
Harmonic Mean	33.765	1.312	0.000	0.000	34.388	4.968	0.301	1.053
Range	36.000	4.000	31.000	34.000	239.000	32.100	9.865	18.224
Median	35.000	1.000	9.000	8.000	38.000	7.300	0.729	1.880
Mode	29.000	1.000	6.000	1.000	45.744	4.400	0.402	4.001
Standard Deviation	7.580	0.887	6.657	6.991	33.863	5.610	1.421	2.811
Variance	57.459	0.787	44.320	48.872	1146.688	31.469	2.019	7.903
Skewness	0.273	1.434	0.649	0.842	2.633	1.065	2.446	2.271
Kurtosis	-0.558	1.538	-0.093	0.081	9.758	1.130	8.093	6.425
Entropy	6.225	6.119	5.982	5.932	6.053	6.049	5.754	5.852

Figure 5. Self-formed information for the reliable group

Table of a "Bad" score group:

	age	ed	employ	address	income	debtinc	creddebt	othdebt
Mean	32.934	1.896	5.224	6.393	41.206	14.728	2.424	3.863
Harmonic Mean	31.051	1.487	0.000	0.000	28.816	9.614	0.675	1.506
Range	35.000	4.000	31.000	29.000	432.000	40.400	20.488	26.873
Median	31.000	2.000	3.000	5.000	31.000	13.800	1.377	2.530
Mode	28.000	1.000	0.000	0.000	45.744	16.000	11.359	5.009
Standard Deviation	8.312	0.978	5.528	5.909	42.470	7.881	3.224	4.252
Variance	69.085	0.956	30.556	34.916	1803.687	62.113	10.392	18.080
Skewness	0.743	0.840	1.718	1.235	5.876	0.631	2.945	2.623
Kurtosis	-0.164	-0.220	3.843	1.337	46.520	0.088	10.000	8.231
Entropy	5.179	5.084	4.695	4.790	4.927	5.064	4.622	4.775

Figure 6. Self-formed information for the unreliable group

A comparison of the correlation allows to say about the strength of the influence of individual factors within each group, see figure 7 and 8.

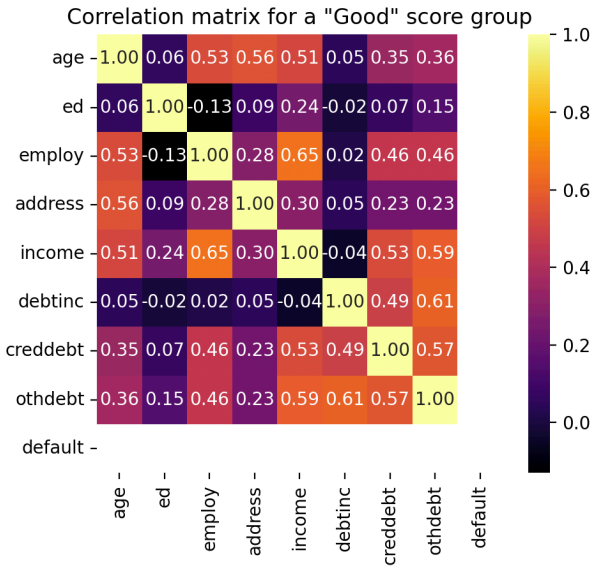

Figure 7. Correlation Matrix for the reliable group

Figure 7 shows a reliable group associate with long work experience and higher earnings. Figure 8 illustrates with age incomes grow slowly, which can lead to frequent loans. Income growth is associated with an increase in debts and debt-to-income ratio. People are likely to face a lack of funds despite the salary increase. This is not observed in the other group.

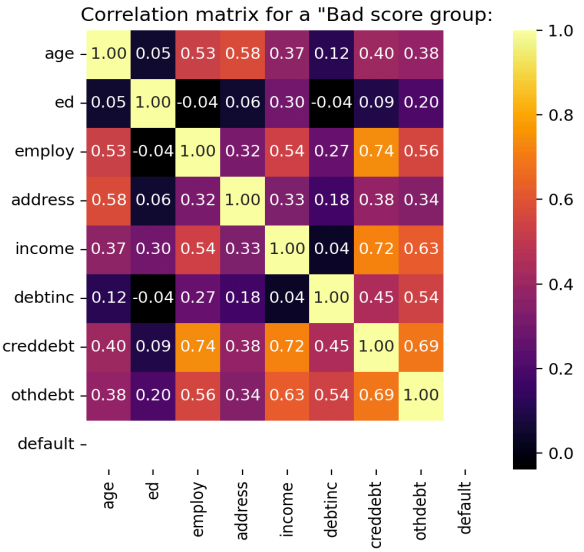

Figure 8. Correlation Matrix for the unreliable group

Thus, the client is likely seemed to be reliable if time with current employer on average 9-10 years. A salary increase should exceed the increase in credits. Next, a program modeling is presented, which helps managers to determine the group score of the future client.

CHAPTER THREE. Modeling and assessment

1. Models implementation

For the classification task Logistic regression is chosen. It works well on large datasets. It is also easy to interpret, and can predict the probability of belonging to a class, which is useful for decision-making. But, if it fluctuates around 0.5, this is difficult to interpret.

The second excellent method is the K-Nearest Neighbors. Unfortunately, it is difficult to identify whether the studied data have linear dependency, so the method deals well with nonlinear dependencies between features and the target variable. On the other hand, the method may be less effective on large datasets. Despite this, it is worth trying to evaluate both models.

The following Python libraries are used to program the modelss: numpy, pandas, matplotlib.pyplot, seaborn, skew, kurtosis, entropy, sklearn.metrics, train_test_split, StandardScaler, LogisticRegression, and KNeighborsClassifier.

First, the data is transformed and prepared. Next, the metrics needed for analysis are calculated using ‘pd.DataFrame’. Then, correlation, covariance, and scattering matrices are constructed using the ‘cov()’, ‘corr()’, ‘sns.heatmap()’, and ‘sns.PairGrid()’. Further, all data is divided into two groups: with 'default' =1 and 'default' = 0. The same measurements as for the whole data are calculated. Then, all the data is divided into training and testing parts by the train_test_split(). They are used to train LogisticRegression() model and calculate the quality. Next, the new text object is used to establish belonging in the group with predict() function. Similar steps are taken to train and use for prediction the KNeighborsClassifier() model. For some functions, the data type or dimensions are changed.

These are all the basic steps of the code used. The full code is provided in appendix 4.

2. Models assessment

For models’ evaluations, the estimates are calculated. Figure 9, demonstrates high accuracy l of 84% for Logistic Regression model. For reliable group model in 15% cases detects unreliable customers. And only 6% of cases, it confuses reliable with the unreliable ones. In unreliable group, tendency is the same, but the percentage is bigger, 45% and 23% respectively. In general, the model recognizes reliable customers better, this may be due to a data imbalance.

Classification Report:

	precision	recall	f1-score	support
0	0.85	0.94	0.89	103
1	0.77	0.54	0.63	37
accuracy			0.84	140
macro avg	0.81	0.74	0.76	140
weighted avg	0.83	0.84	0.83	140

Classification Report:

	precision	recall	f1-score	support
0	0.83	0.90	0.87	103
1	0.64	0.49	0.55	37
accuracy			0.79	140
macro avg	0.74	0.69	0.71	140
weighted avg	0.78	0.79	0.78	140

Figure 9. Classification report for the Logistic Regression (Left) and for the K-Nearest Neighbors (right)

The K-Nearest Neighbors model has exactly the same results, but slightly poorer. The accuracy is lower, up to 79%. There are Confusion Matrixes and predicted values in appendix 5.

The models can be tested for a real customer. For example, a client's portfolio as: 'age'=58, 'ed'=2, 'employ'=24, 'address'=2, 'income'=59,600, 'debtinc'=25.49, 'creddebt'=8.06, 'othdebt'=12.74. Both models predict the unreliable group. The results are shown in Figure 10.

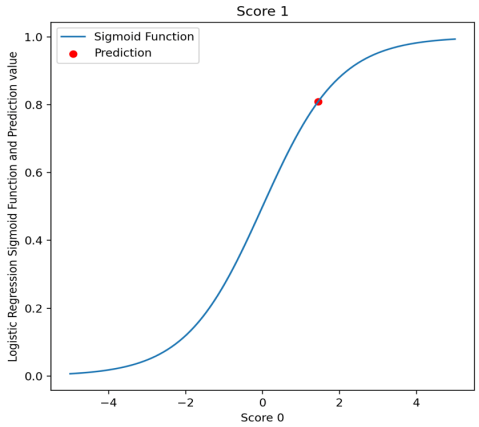

Figure 10. Sigmoid function

Figure 10 demonstrates the prediction of Logistic Regression model that Predicted Score group: 1 (unreliable) with 80,88% of probability, the result is valid.

Logistic Regression model is proved to be better suited for implementation into the bank's work, since it makes fewer mistakes.

Thus, using a trained model, the prediction may be made with a high degree of accuracy whether a potential client will repay the debt or not. This model can be used in practice, which is presented below.

CONCLUDING REMARKS

After the analysis, the main findings can be practically applied in the Artificial Bank operation. In particular, the distinctive characteristics of reliable and unreliable groups, as well as the main factors influencing group belonging.

Firstly, to use the proposed model in the web application on daily basis for decision making. Just entering the client's basic data, such as age, education level, work experience, residence at the same address, annual income, debt-to-income ratio, credit card and other debts, and the probability of being unreliable is calculated.

Secondly, to design its own credit scores, for example, the “ABC” system, dividing customers into three groups depended on the probability of being unreliable. Group "A" includes the most reliable clients (0.45–0), "C" - the least reliable (1–0.56), and "B" (0.55–0.46) is considered a “Gray zone” where changes are required to improve the assessment, for example, an find new income sources or repayment of existing debts.

Thirdly, to supplement the database with new loan parameters to improve the model such as term, interest rate, and amount. It may extend the model.

These recommendations may reduce the percentage of non-repayment loans and increase the bank's assets turnover, making it more financial stable.

The study successfully answered the question of customer risk assessment and helped to develop skills in working with different types of data and data cleaning methods. Further improvement of the model is possible by expanding the sample of unreliable customers or eliminating insignificant parameters, for example, income.

BIBLIOGRAPHY

1. Kaggle (2019) *Credit Risk Analysis* [online] available from <https://www.kaggle.com/datasets/karanagarwal/bankloans [10 January 2024]
2. International Monetary Fund (2022) *Household debt, loans and debt securities* [online] available from <https://www.imf.org/external/datamapper/HH_LS@GDD/CAN/GBR/USA/DEU/ITA/FRA/JPN/VNM.> [25 January 2024]
3. Witzel M., and Bhargava N. (2023), '*AI-Related Risk: The Merits of an ESG-Based Approach to Oversight*.' Centre for International Governance Innovation
4. Subburayan B. et al (2023) *Patent - Transforming Traditional Banking The AI Revolution* Research Gate [online] available from https://www.researchgate.net/publication/374504440_Patent_-_Transforming_Traditional_Banking_The_AI_Revolution/.> [20 January 2024]
5. Hooman A. et al (2016) 'Stat*istical and data mining methods in credit scoring.*' The Journal of Developing Areas 50 (5) 371-381
6. Florez-Lopez R. and Ramon-Jeronimo J.M. (2014) '*Modelling credit risk with scarce default data: on the suitability of cooperative bootstrapped strategies for small low-default portfolios.*' The Journal of the Operational Research Society 65 (3) 416-434
7. Miguéis V.L., Benoit D.F. and Van den Poel D. (2013) '*Enhanced decision support in credit scoring using Bayesian binary quantile regression*.' The Journal of the Operational Research Society 64 (9) 1374-1383

APPENDIX 1. Dataset information and description

	age	ed	employ	address	income	debtinc	creddebt	othdebt
count	681.000000	680.000000	700.000000	700.000000	663.00000	700.000000	700.000000	700.000000
mean	34.898678	1.717647	8.388571	8.268571	45.74359	10.260571	1.553553	3.058209
std	8.861849	0.925652	6.658039	6.821609	37.44108	6.827234	2.117197	3.287555
min	20.000000	1.000000	0.000000	0.000000	14.00000	0.400000	0.011696	0.045584
25%	28.000000	1.000000	3.000000	3.000000	24.00000	5.000000	0.369059	1.044178
50%	34.000000	1.000000	7.000000	7.000000	34.00000	8.600000	0.854869	1.987567
75%	40.000000	2.000000	12.000000	12.000000	54.50000	14.125000	1.901955	3.923065
max	136.000000	5.000000	31.000000	34.000000	446.00000	41.300000	20.561310	27.033600

Figure 11. Original data description

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       681 non-null    float64
 1   ed        680 non-null    float64
 2   employ    700 non-null    int64
 3   address   700 non-null    int64
 4   income    663 non-null    float64
 5   debtinc   700 non-null    float64
 6   creddebt  700 non-null    float64
 7   othdebt   700 non-null    float64
 8   default   700 non-null    object
dtypes: float64(6), int64(2), object(1)
memory usage: 49.3+ KB
```

Figure 12. Information of the original dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       700 non-null    float64
 1   ed        700 non-null    int64
 2   employ    700 non-null    int64
 3   address   700 non-null    int64
 4   income    700 non-null    float64
 5   debtinc   700 non-null    float64
 6   creddebt  700 non-null    float64
 7   othdebt   700 non-null    float64
 8   default   700 non-null    int64
dtypes: float64(5), int64(4)
memory usage: 49.3 KB
```

Figure 13. Information of the cleaned dataset

APPENDIX 2. Covariance Matrixes

Covariance Matrix for the whole dataset:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
age	61.76	0.26	28.23	30.54	134.57	0.58	5.01	8.76	-0.48
ed	0.26	0.83	-0.82	0.36	8.32	0.18	0.20	0.54	0.05
employ	28.23	-0.82	44.33	14.58	146.30	-1.42	5.69	8.89	-0.83
address	30.54	0.36	14.58	46.53	76.71	0.58	3.02	5.09	-0.49
income	134.57	8.32	146.30	76.71	1327.63	-9.86	43.28	69.91	-1.19
debtinc	0.58	0.18	-1.42	0.58	-9.86	46.61	7.25	13.13	1.17
creddebt	5.01	0.20	5.69	3.02	43.28	7.25	4.48	4.41	0.23
othdebt	8.76	0.54	8.89	5.09	69.91	13.13	4.41	10.81	0.21
default	-0.48	0.05	-0.83	-0.49	-1.19	1.17	0.23	0.21	0.19

Figure 14. Covariance matrix for the whole dataset

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
age	69.46	0.42	24.61	28.51	132.78	7.62	10.78	13.53	0.0
ed	0.42	0.96	-0.21	0.30	12.63	-0.36	0.28	0.83	0.0
employ	24.61	-0.21	30.72	10.48	127.38	11.76	13.28	13.16	0.0
address	28.51	0.30	10.48	35.11	83.66	8.23	7.35	8.59	0.0
income	132.78	12.63	127.38	83.66	1813.60	12.45	99.07	113.56	0.0
debtinc	7.62	-0.36	11.76	8.23	12.45	62.45	11.44	18.26	0.0
creddebt	10.78	0.28	13.28	7.35	99.07	11.44	10.45	9.55	0.0
othdebt	13.53	0.83	13.16	8.59	113.56	18.26	9.55	18.18	0.0
default	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0

Figure 15. Covariance Matrix for a “Bad” score group

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
age	57.57	0.42	26.80	29.68	131.50	2.00	3.74	7.79	0.0
ed	0.42	0.79	-0.70	0.55	7.38	-0.05	0.09	0.39	0.0
employ	26.80	-0.70	44.41	13.21	146.36	0.72	4.35	8.62	0.0
address	29.68	0.55	13.21	48.97	70.31	1.90	2.28	4.59	0.0
income	131.50	7.38	146.36	70.31	1148.91	-8.02	25.59	56.40	0.0
debtinc	2.00	-0.05	0.72	1.90	-8.02	31.53	3.92	9.62	0.0
creddebt	3.74	0.09	4.35	2.28	25.59	3.92	2.02	2.26	0.0
othdebt	7.79	0.39	8.62	4.59	56.40	9.62	2.26	7.92	0.0
default	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.0

Figure 16. Covariance Matrix for a “Good” score group

APPENDIX 3. Scatter Matrixes for two score groups

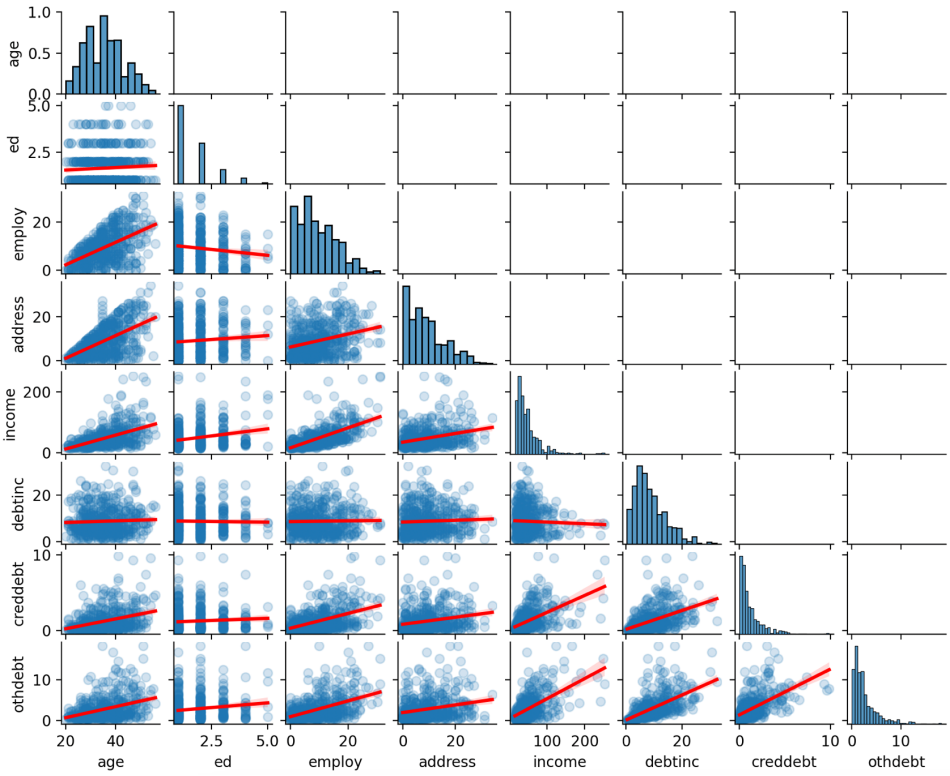

Figure 17. Scatter Plot Matrix for the "Good" score group

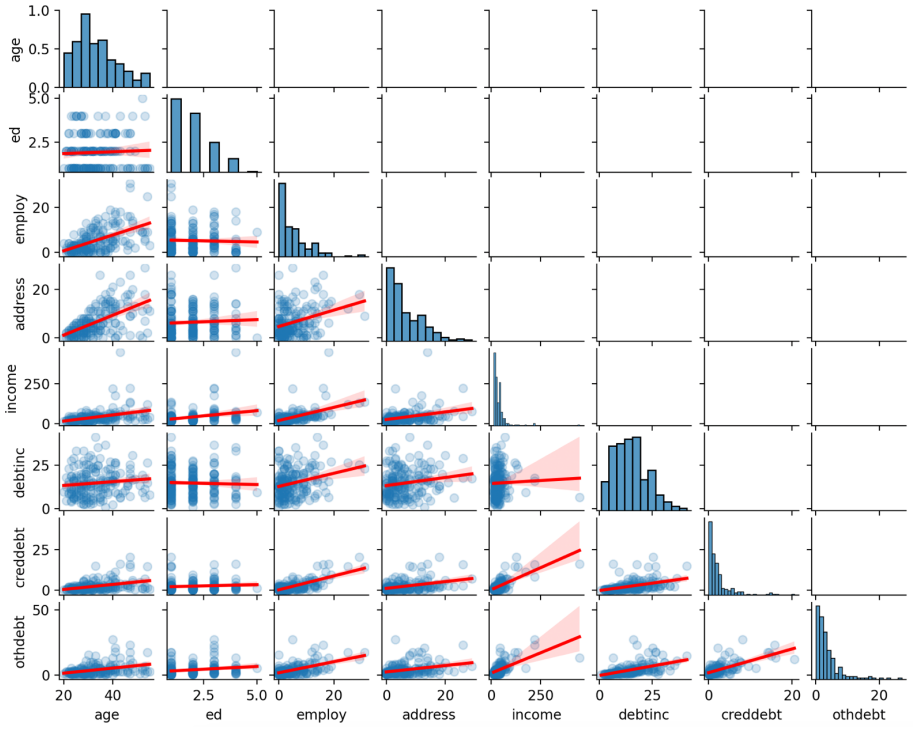

Figure 18. Scatter Plot Matrix for the "Bad" score group

APPENDIX 4. Full code

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew, kurtosis, entropy
import numpy as np
import statistics as st
from sklearn.metrics import (mean_squared_error, r2_score, accuracy_score,
                             confusion_matrix, classification_report)
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier

#input data + describing
dataset =
pd.read_csv('/Users/tatiana_bazaleva/python_projects/Studying/Bankloan.txt')
print(dataset.info())
print(dataset.describe().to_string())

#cleaning
print(dataset['default'].unique())
dataset.loc[:, 'default'] = dataset['default'].replace(':0', 0)
dataset.loc[:, 'age'] = dataset['age'].replace(136, 36)
dataset.loc[:, 'default'] = pd.to_numeric(dataset['default'],
errors='coerce')

#replace NAN values to mean values
dataset_cleaned = dataset.fillna(dataset.mean())
dataset_cleaned['ed'] = dataset_cleaned['ed'].astype(int)
print('\nDataset Cleaned\n',dataset_cleaned)

#design descroptions of the whole data
result_table = pd.DataFrame(index=['Mean', 'Harmonic Mean', 'Range',
'Median',
                                   'Mode', 'Standard Deviation', 'Variance',
                                   'Skewness', 'Kurtosis', 'Entropy'])
#choosing only numerical values of the whole data
numeric_columns = dataset_cleaned.iloc[:, :-1]

#mesuments calculation of the whole data
for column in numeric_columns:
    result_table[column] = [np.mean(dataset_cleaned[column]),

st.harmonic_mean(dataset_cleaned[column].tolist()),
                            np.ptp(dataset_cleaned[column]),
                            np.median(dataset_cleaned[column]),
                            st.mode(dataset_cleaned[column].tolist()),
                            np.std(dataset_cleaned[column]),
                            np.var(dataset_cleaned[column]),
                            skew(dataset_cleaned[column]),
                            kurtosis(dataset_cleaned[column]),
                            entropy(dataset_cleaned[column].dropna())]

result_table_rounded = result_table.round(3)
print('\n',result_table_rounded.to_string())
```

APPENDIX 4. Full code (continued)

```
#designed cov and cor matrix of the whole data
covariance_matrix = dataset_cleaned.cov()
correlation_matrix = dataset_cleaned.corr()
covariance_matrix=covariance_matrix.round(2)
print('\n', 'Covariance Matrix for the whole dataset:')
print(covariance_matrix.to_string())

sns.heatmap(dataset_cleaned.corr())
sns.heatmap(correlation_matrix, annot=True, cmap='inferno', fmt=".2f")
plt.title('Correlation matrix for the whole dataset')
plt.show()

#design Scatter Matrix of the whole data
scatter_grid = sns.PairGrid(dataset_cleaned, diag_sharey=False)
scatter_grid.map_lower(sns.regplot, scatter_kws={'alpha':0.2},
                       line_kws={'color': 'red'})
scatter_grid.map_diag(sns.histplot, kde_kws={'color': 'blue'})
plt.show()

#splitting
bad_score_1 = dataset_cleaned[dataset_cleaned['default'] == 1]
good_score_0 = dataset_cleaned[dataset_cleaned['default'] == 0]

#description of a Good group
result_table_good = pd.DataFrame(index=['Mean', 'Harmonic Mean', 'Range',
                                        'Median', 'Mode', 'Standard
Deviation',
                                        'Variance', 'Skewness', 'Kurtosis',
                                        'Entropy'])
#choosing only numerical values
numeric_columns_good = good_score_0.iloc[:, :-1]

#mesuments calculation
for column in numeric_columns_good:
    result_table_good[column] = [np.mean(numeric_columns_good[column]),

st.harmonic_mean(numeric_columns_good[column].tolist()),
                            np.ptp(numeric_columns_good[column]),
                            np.median(numeric_columns_good[column]),
                            st.mode(numeric_columns_good[column].tolist()),
                            np.std(numeric_columns_good[column]),
                            np.var(numeric_columns_good[column]),
                            skew(numeric_columns_good[column]),
                            kurtosis(numeric_columns_good[column]),
                            entropy(numeric_columns_good[column].dropna())]

result_table_rounded_good = result_table_good.round(3)
print('\nTable for a "Good" score
group:\n',result_table_rounded_good.to_string())

#designed cor and cor matrix
covariance_matrix_good = good_score_0.cov()
correlation_matrix_good = good_score_0.corr()
```

APPENDIX 4. Full code (continued)

```
covariance_matrix_good=covariance_matrix_good.round(2)
print('\n', 'Covariance Matrix for a "Good" score group:')
print(covariance_matrix_good.to_string())

sns.heatmap(correlation_matrix_good)
sns.heatmap(correlation_matrix_good, annot=True, cmap='inferno', fmt=".2f")
plt.title('Correlation matrix for a "Good" score group')
plt.show()

#description of a Bad group
result_table_bad = pd.DataFrame(index=['Mean', 'Harmonic Mean', 'Range',
                                       'Median', 'Mode', 'Standard
Deviation',
                                       'Variance', 'Skewness', 'Kurtosis',
                                       'Entropy'])
#choosing only numerical values
numeric_columns_bad = bad_score_1.iloc[:, :-1]

#mesuments calculation
for column in numeric_columns_bad:
    result_table_bad[column] = [np.mean(numeric_columns_bad[column]),

st.harmonic_mean(numeric_columns_bad[column].tolist()),
                                np.ptp(numeric_columns_bad[column]),
                                np.median(numeric_columns_bad[column]),

st.mode(numeric_columns_bad[column].tolist()),
                                np.std(numeric_columns_bad[column]),
                                np.var(numeric_columns_bad[column]),
                                skew(numeric_columns_bad[column]),
                                kurtosis(numeric_columns_bad[column]),

entropy(numeric_columns_bad[column].dropna())]

result_table_rounded_bad = result_table_bad.round(3)
print('\nTable of a "Bad" score group:\n',
result_table_rounded_bad.to_string())

#design Scatter Matrix for two classes
scatter_grid = sns.PairGrid(numeric_columns_good, diag_sharey=False)
scatter_grid.map_lower(sns.regplot, scatter_kws={'alpha':0.2},
                       line_kws={'color': 'red'})
scatter_grid.map_diag(sns.histplot, kde_kws={'color': 'blue'})
plt.show()

scatter_grid = sns.PairGrid(numeric_columns_bad, diag_sharey=False)
scatter_grid.map_lower(sns.regplot, scatter_kws={'alpha':0.2},
                       line_kws={'color': 'red'})
scatter_grid.map_diag(sns.histplot, kde_kws={'color': 'blue'})
plt.show()

#designed cor and cor matrix
covariance_matrix_bad = bad_score_1.cov()
correlation_matrix_bad = bad_score_1.corr()
```

APPENDIX 4. Full code (continued)

```
print('\n', 'Covariance Matrix for a "Bad" score group:')
print(covariance_matrix_bad.round(2).to_string())

sns.heatmap(correlation_matrix_bad)
sns.heatmap(correlation_matrix_bad, annot=True, cmap='inferno', fmt=".2f")
plt.title('Correlation matrix for a "Bad score group:')
plt.show()

#splitting dataset for train and test parts
X_train, X_test, y_train, y_test = train_test_split(
    dataset_cleaned.drop('default', axis=1),
    dataset_cleaned['default'], test_size=0.2, random_state=1)

y_train = y_train.astype(int)
y_test = y_test.astype(int)

#creating and teaching a model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
df = pd.DataFrame({'Actual for a Logistic Regression': y_test,
                   'Predicted for a Logistic Regression': predictions})

print(df)
print("\nClassification Report:\n", classification_report(y_test,
predictions))

#prediction about next customer
new_data=np.array([58, 2, 24, 2, 59.6, 25.49, 8.06, 12.74]).reshape(1, -1)
predicted_log_reg = model.predict(new_data)
log_odds = model.intercept_[0] + np.dot(model.coef_[0], new_data[0])

#calculating probability and sigmoind function
probability = 1 / (1 + np.exp(-log_odds))
x_values = np.linspace(-5, 5, 100)
sigmoid_values = 1 / (1 + np.exp(-x_values))
print(f'Probability of "Bad" score group: {probability:.4f}')
print(f'Predicted Score group: {predicted_log_reg[0]}')

#design a sigmoind
plt.figure(figsize=(8, 8))
plt.plot(x_values, sigmoid_values, label='Sigmoid Function')
plt.scatter(log_odds, probability, color='red', marker='o',
label='Prediction')
plt.title('Score 1')
plt.xlabel('Score 0')
plt.ylabel('Logistic Regression Sigmoid Function and Prediction value')
plt.legend()
plt.show()

#design a fonfusion matrix
cm_log_reg = confusion_matrix(y_test, predictions)
sns.heatmap(cm_log_reg, annot=True, fmt="d", cmap='inferno', xticklabels=[
    'Predicted 0', 'Predicted 1'], yticklabels=['Actual 0', 'Actual 1'])
```

APPENDIX 4. Full code (continued)

```
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix of a Logistic Regression')
plt.show()

#scaling of features for KNeighborsClassifierk
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#design KNeighborsClassifierk
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train_scaled, y_train)

#model evaluation
y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy for a K-Nearest Neighbors model:", accuracy)
print("\nClassification Report for a K-Nearest Neighbors model:\n",
      classification_report(y_test, y_pred))

df = pd.DataFrame({'Actual for a K-Nearest Neighbors model': y_test,
                   'Predicted for a K-Nearest Neighbors': y_pred})
print(df)

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap='inferno', xticklabels=[
    'Predicted 0', 'Predicted 1'], yticklabels=['Actual 0', 'Actual 1'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix of K-means module')
plt.show()

new_data = np.array([58, 2, 24, 2, 59.6, 25.49, 8.06, 12.74]).reshape(1, -1)
new_data_kmeans = model.predict(new_data)
print('Predicted value of a K-means module:', new_data_kmeans)
```

APPENDIX 5. Confusion matrixes and predicted values

	Actual for a Logistic Regression	Predicted for a Logistic Regression
681	1	0
626	0	0
329	0	0
620	0	0
399	1	0
..	...	...
649	0	1
427	0	0
119	1	0
82	0	1
218	1	0

Figure 19. Actual and predicted values of the Logistic Regression model

	Actual for a K-Nearest Neighbors model	Predicted for a K-Nearest Neighbors
681	1	0
626	0	0
329	0	0
620	0	1
399	1	0
..	...	...
649	0	1
427	0	0
119	1	1
82	0	1
218	1	0

Figure 20. Actual and predicted values of the K-Nearest Neighbors

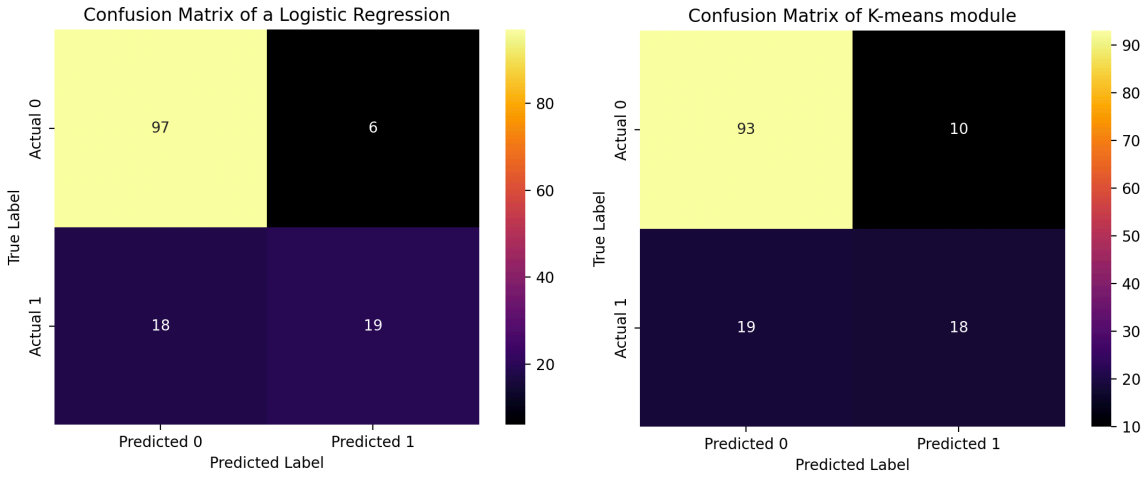

Figure 21. Confusion Matrix of the Logistic Regression model (left) and of the K-Nearest Neighbors (right)