DEVELOP A CREDIT CARD APPROVAL PREDICTION SYSTEM IN WHICH THE OBJECTIVE IS TO ANALYZE HISTORICAL CREDIT CARD APPLICATION DATA, PERFORM DATA PREPROCESSING , CONDUCT EXPLORATORY DATA ANALYSIS (EDA), CREATE INTERACTIVE DATA VISUALIZATIONS USING TABLEAU, AND BUILD A PREDICTIVE MODEL TO ASSESS THE LIKELIHOOD OF CREDIT CARD APPROVAL FOR NEW APPLICANTS.

A PROJECT SUBMITTED TO

AMERICAN HERITAGE UNIVERSITY OF SOUTHERN CALIFORNIA

FOR PARTIAL COMPLETION OF THE DEGREE OF

MASTER OF BUSINESS ADMINISTRATION IN BUSINESS ANALYTICS

BY

- **AMITH R**
- **SNEHA KUMARI**
- **ARAVIND K**

UNDER THE GUIDANCE OF

DILIP BALASUBRAMANIAN

CERTIFICATE

This is to certify that Mr. AMITH R, MISS SNEHA KUMARI & Mr. ARAVIND K has worked and duly completed their project work for the degree of master of business administration in business analytics and their project entitled "Develop a credit card approval prediction system in which the objective is to analyze historical credit card application data, perform data preprocessing, conduct exploratory data analysis (EDA), create interactive data visualizations using tableau, and build a predictive model to assess the likelihood of credit card approval for new applicants." under my supervision. I further certify that the entire work has been done by the learners under my guidance and that no part of this has been submitted previously for any degree or diploma of any university. It is their own work and facts reported by their personal findings and investigations

Name and Signature of Guiding Teacher

(Dilip Balasubramanian)

Declaration

We the undersigned, Mr. Aravind K, Miss Sneha Kumari & Mr. Amith R, here by declare that the work embodied in this project work titled "Develop a Credit Card approval prediction system in which the objective is to analyze historical credit card application data, perform data preprocessing, conduct EDA & Build a predictive model to assess the likelihood of Credit Card approval for new applicants" forms our own contribution to the research work carried out under the guidance of Mr. Dilip Balasubramanian Result of our own research work and has not been previously submitted to any other university for any other degree/diploma to this or any other university. We further declare that all the data, information, and findings presented in this project are based on genuine research and investigation. We hereby further declare that all information of this document has been obtained and presented in accordance with academic rules and ethical conduct.

Signatures:

- **Mr. Aravind K**
- **Miss Sneha Kumari**
- **Mr. Amith R**

ACKNOWLEDGMENT

To list who all have helped us is difficult because they are so numerous and the depth is so enormous. We extend our heartfelt gratitude and appreciation to GEMS B School for providing us with the opportunity to pursue the Master of Business Administration in Business Analytics program.

We are deeply indebted to Mr. Dilip Balasubramanian, whose unwavering guidance, mentorship, and expertise have been the cornerstone of our project's success. His invaluable guidance and mentorship were instrumental in the success of our project.

In addition, we would like to acknowledge the collaborative efforts and contributions of our fellow students and colleagues during this team project.

Finally, we express our heartfelt gratitude to our families and friends for their unwavering support and understanding throughout our academic pursuits.

This project would not have been possible without the collective efforts and support of everyone mentioned above. Thank you for being an integral part of our academic and personal growth.

Table of Contents

Abstract

The "Credit Card Approval Prediction System" project delves into the realm of data-driven decision-making within the financial sector. With the objective of enhancing credit card application processes, this project amalgamates historical application data, rigorous data preprocessing techniques, and comprehensive exploratory data analysis (EDA). The endeavor also incorporates interactive data visualizations crafted using Tableau, offering valuable insights into the patterns and trends within the data.

One of the primary goals is the development of a predictive model capable of assessing the likelihood of credit card approval for new applicants. This model leverages machine learning techniques to make informed predictions based on historical trends and applicant profiles. The project fosters a multidisciplinary approach by combining aspects of data science, analytics, and business acumen to deliver a comprehensive solution.

The report encapsulates the project's journey from data collection and preprocessing to exploratory analysis, visualization, and model building. It provides an in-depth view of the methodologies employed, the insights gained, and the predictive model's performance. Furthermore, it discusses the implications and

potential applications of the predictive model within the financial domain.

Introduction

Project Overview –

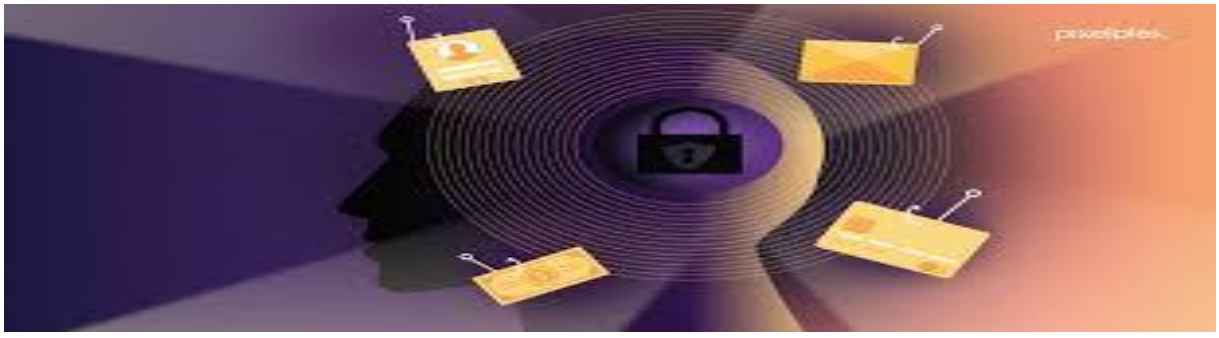
In today's fast-paced digital economy, the financial landscape is continually evolving. Credit cards have become indispensable tools for consumers and are central to the functioning of the modern financial sector. However, the efficient and accurate approval of credit card applications remains a critical challenge for financial institutions.

The financial landscape is in a state of perpetual flux, driven by digital transformation and ever-evolving consumer demands. Within this dynamic milieu, credit cards have emerged as pivotal instruments, reshaping how we transact and manage our finances. In this context, the efficient and precise approval of credit card applications stands as a paramount concern for financial institutions. It is at this juncture that our project, titled "Develop a Credit Card Approval Prediction System," takes center stage.

Motivation –

The motivation behind our endeavor springs from a pressing need. As the volume of credit card applications surges, the intricacies of assessing an applicant's creditworthiness have grown exponentially. Conventional methods, while reliable, can no longer bear the weight of this demand efficiently. Thus, the quest for a data-driven, adaptive, and scalable solution becomes imperative. We are motivated by the aspiration to redefine the way credit card applications are processed, bridging the gap between historical insights and real-time decision-making.

Project Scope –



Our project transcends the conventional confines of data analysis. It encompasses the holistic journey of a credit card application, from inception to approval, driven by the power of data. Our scope extends from the meticulous collection and preprocessing of

historical credit card application data to the generation of profound insights via Exploratory Data Analysis (EDA). It then seamlessly transitions into the realm of interactive data visualizations, powered by Tableau. However, our ultimate aim is to create a predictive model, an analytical masterpiece that assesses the likelihood of credit card approval for new applicants. This predictive model embodies the essence of our project, acting as the lynchpin of innovation and efficiency in the financial sector.

Project Goals –

- **Data Preprocessing: We embark on our journey with the critical task of data preprocessing. This phase ensures that our historical credit card application data is not just usable but impeccable. It encompasses data cleansing, transformation, and enrichment, making it ready for rigorous analysis.**

- **Exploratory Data Analysis (EDA): Armed with pristine data, we dive into EDA. This phase goes beyond mere visualization. It involves uncovering hidden patterns, identifying correlations,**

and scrutinizing factors that influence credit card approvals. It is the chapter where data speaks, and we listen attentively.

- **Interactive Data Visualizations: As we navigate through the landscape of data insights, we harness the power of Tableau. This chapter brings data to life. We create interactive visualizations that transcend conventional graphs and charts. They not only depict the past but provide windows into the future.**

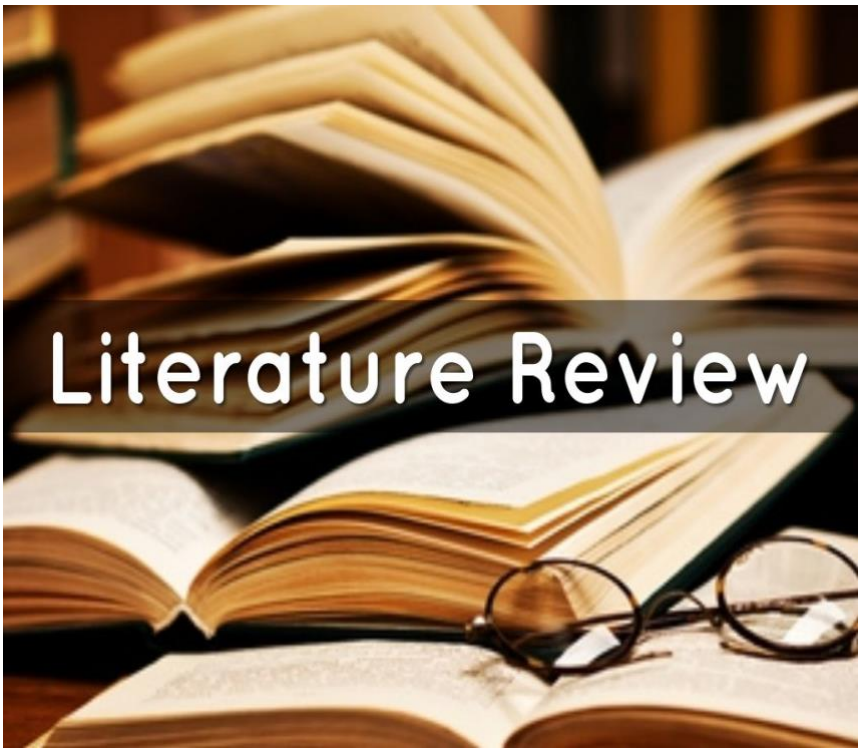
- **Predictive Modeling: The pinnacle of our project is predictive modeling. In this phase, we channel the collective wisdom of data into building a model capable of predicting the likelihood of credit card approval for new applicants. This predictive prowess transforms our project into a catalyst for innovation in financial decision-making.**

Importance of the Subject –

Our project does not merely seek to transform the credit card approval process; it aims to revolutionize it. The implications are far-

reaching. By harnessing data-driven techniques and predictive modeling, we endeavor to enhance the efficiency, accuracy, and fairness of credit assessment. Financial institutions benefit from reduced financial risk and optimized resource allocation. On the other side, credit card applicants experience a smoother, more responsive approval process. Our project is a testament to the evolution of credit card approval mechanisms in an era defined by data-driven insights, technological innovation, and a commitment to a better financial future.

Literature Review



Credit card approval prediction systems have become integral in the contemporary financial landscape. The increased reliance on credit services, coupled with the proliferation of financial technology (FinTech) companies, has accelerated the development of advanced predictive models. This literature review provides an extensive overview of the key themes and developments in the field of credit card approval prediction, encompassing historical perspectives, methodologies, challenges, and emerging trends.

Historical Evolution of Credit Scoring Models –

- **Traditional Credit Scoring Models**

Historically, credit card approval processes heavily depended on manual assessment and were prone to subjective biases. The introduction of traditional credit scoring models, such as the FICO score, transformed the credit industry. These models considered factors like payment history, outstanding debt, and length of credit history to evaluate an applicant's creditworthiness.

- **Emergence of Machine Learning**

The transition from traditional models to machine learning-based approaches marked a significant shift in credit risk assessment. Researchers began leveraging advanced techniques like logistic regression, decision trees, and random forests to enhance prediction accuracy. This section explores the evolution of machine learning in credit scoring.

Data Sources and Preprocessing –

- **Data Sources**

One of the fundamental aspects of credit card approval prediction systems is data availability. This chapter investigates various data sources, from credit bureau reports to alternative data, that have transformed the landscape of credit assessment.

- **Data Preprocessing Techniques**

Effective data preprocessing is critical in ensuring the quality of input data for predictive models. This section discusses common

preprocessing techniques such as missing data handling, feature scaling, and outlier detection.

Predictive Modeling Techniques –

- **Logistic Regression**

Logistic regression remains a cornerstone in credit card approval prediction. This chapter explores the principles of logistic regression and its application in modeling credit risk.

- **Ensemble Methods**

Ensemble methods, including random forests and gradient boosting, have gained popularity for their ability to improve model performance. This section delves into ensemble techniques and their advantages in credit scoring.

Challenges in Credit Card Approval Prediction –

- **Data Privacy and Security**

The sensitive nature of financial data necessitates robust data privacy and security measures. This chapter examines the challenges and solutions associated with protecting applicant information.

- **Model Interpretability and Fairness**

As predictive models grow in complexity, issues of interpretability and fairness arise. Researchers and practitioners are exploring ways to make models more transparent and unbiased.

Emerging Trends and Future Directions –

- **Explainable AI (XAI) in Credit Scoring**

The emergence of Explainable AI (XAI) promises to address the interpretability challenge. This section investigates the application of XAI techniques in credit card approval prediction.

- **Integration of Big Data and AI**

The advent of big data technologies and artificial intelligence is reshaping credit risk assessment. This chapter explores how financial institutions are leveraging big data analytics and AI to refine their credit approval processes.

Project Relevance and Significance -

- **Motivation for the Project**

This chapter underscores the motivation behind the project titled "Develop a Credit Card Approval Prediction System." It discusses the contemporary relevance of enhancing credit assessment techniques.

- **Project Scope and Goals**

The project's scope and objectives are outlined in this section. It emphasizes the importance of leveraging historical credit card application data, performing data preprocessing, conducting exploratory data analysis, and building predictive models.

- **Importance of the Subject**

The chapter concludes with a reflection on the broader significance of credit card approval prediction systems in the financial industry. It highlights the potential benefits for both financial institutions and consumers.

Overall –

In conclusion, this literature review provides a comprehensive analysis of the evolution, methodologies, challenges, and future trends in credit card approval prediction. It contextualizes the importance of the project within the broader landscape of credit assessment and sets the stage for the subsequent chapters of the project report.

Project Description

Develop a Credit Card Approval Prediction System

In the ever-evolving financial landscape, the efficient assessment of credit card applications is a crucial task for financial institutions. Traditional manual evaluation processes are time-consuming and often subject to human biases. To address these challenges, this project aims to create a Credit Card Approval Prediction System.

Objectives

1. **Analyze historical credit card application data.**
2. **Implement data preprocessing techniques to enhance data quality.**
3. **Conduct exploratory data analysis (EDA) to extract valuable insights.**
4. **Utilize Tableau for creating interactive data visualizations.**
5. **Develop a predictive model to assess the likelihood of credit card approval for new applicants.**

Importance

Efficient credit assessment benefits both financial institutions and applicants. It enables institutions to make informed lending decisions, thereby managing risks effectively. For applicants, it simplifies the application process and enhances their access to credit. Furthermore, this project contributes to the advancement of credit risk assessment methodologies.

Methodology

This project follows a structured methodology:

1. **Data Collection: Gather historical credit card application data from various sources.**
2. **Data Preprocessing: Cleanse and preprocess the data to ensure accuracy and reliability.**
3. **Exploratory Data Analysis (EDA): Explore the dataset using statistical and visual techniques.**
4. **Model Development: Utilize machine learning algorithms to build a predictive model.**

Data Visualization

Interactive data visualizations will be created using Tableau, providing a comprehensive view of credit card application trends, approval rates, and influential factors.

Results and Findings

The project will present model performance metrics and key insights derived from the data. These insights will shed light on the factors influencing credit card approval.

Conclusion

In conclusion, the development of a Credit Card Approval Prediction System is essential to streamline the application process and enhance the accuracy of credit assessment. The project aims to provide a valuable tool for financial institutions and contribute to the ongoing advancements in credit risk assessment.

References

A list of references to academic papers, articles, and resources that guide the project's methodology and analysis.

Dataset Understanding

- **Dataset 1 – Application Record**

Variable Name	Description
ID	IDENTIFICATIONS NUMBER OF THE APPLICANTS
CODE GENDER	MALE OR FEMALE
FLAG OWN CAR	HAVING CAR OR NOT
FLAG OWN REALITY	OWN CAR OR NOT
CNT CHILDERN	COUNT OF CHILDREN
AMT INCOME TOTAL	TOTAL INCOME
NAME INCOME TYPE	THE INCOME SOURCE
NAME EDUCATION TYPE	EDUCATION STATUS
NAME FAMILY STATUS	MARRIED OR NOT
NAME HOUSING TYPE	HAVING OWN HOUSE OR NOT
DAYS BIRTH	COUNT OF DAYS AFTER BIRTH

DAYS EMPLOYED	**COUNT OF DAYS AFTER EMPLOYED**
FLAG MOBIL	**HAVING MOBILE OR NOT**
FLAG WORK PHONE	**HAVING WORK RELATED PHONE OR NOT**
FLAG PHONE	**HAVING LANDPHONE OR NOT**
FLAG EMAIL	**HAVING EMAIL OR NOT**
OCCUPATION TYPE	**JOB TYPE**
CNT FAM MEMBERS	**COUNT OF FAMILY MEMBERS**

- **DATASET 2 – CREDIT RECORD**

VARIABLE NAME **DESCRIPTION**

ID	**IDENTIFICATION NUMBER OF APPLICANTS**
MONTHS BALANCE	**COUNT OF MONTHS AFTER RECEIVING THE APPLICATION**
STATUS	**STATUS OF THE APPLICATION**

Benefits of the Project

The development and implementation of a Credit Card Approval Prediction System offer a multitude of benefits to financial institutions, applicants, and the broader financial landscape. These benefits encompass enhanced decision-making, improved efficiency, and a more inclusive financial ecosystem:

1. **Efficient Decision-Making:**

- **Faster Application Processing: The predictive model expedites the credit card application review process, reducing the time required for decision-making.**
- **Reduced Manual Workload: Automation of the initial assessment reduces the burden on human evaluators, allowing them to focus on more complex cases.**
- **Risk Mitigation: By accurately predicting credit card approvals, financial institutions can proactively manage risks associated with lending.**

2. **Enhanced Accuracy and Fairness:**

- **Objective Assessment: The model provides an objective assessment of creditworthiness, reducing the potential for human biases.**
- **Consistent Criteria: Standardized criteria ensure that all applicants are evaluated using the same parameters, ensuring fairness.**

3. **Improved Customer Experience:**

- **Streamlined Application Process: Applicants benefit from a streamlined, user-friendly application process, which encourages financial inclusion.**
- **Timely Decisions: Faster approval decisions provide applicants with clarity and reduce uncertainty.**

4. **Data-Driven Insights:**

- **Informed Strategies: Insights derived from exploratory data analysis (EDA) enable financial institutions to refine their credit card offerings and marketing strategies.**
- **Risk Management: A deeper understanding of influential factors allows institutions to proactively manage credit risks.**

5. **Cost Efficiency:**

- **Resource Optimization: By automating the initial assessment, financial institutions can optimize resource allocation.**
- **Reduced Manual Errors: Automation reduces the potential for manual errors, minimizing associated costs.**

6. **Competitive Advantage:**

- **Innovation: Implementing advanced credit assessment methods demonstrates a commitment to innovation and customer-centric services, providing a competitive edge.**

7. **Financial Inclusion:**

- **Wider Access to Credit: A fair and efficient credit approval system broadens access to credit for a diverse range of applicants, promoting financial inclusion.**

8. **Industry Advancements:**

- **Contributing to Best Practices: The project contributes to the development of best practices in credit risk assessment, benefiting the financial industry as a whole.**

Overall

The benefits outlined above underscore the significance of the Credit Card Approval Prediction System. By harnessing data-driven insights and automation, financial institutions can make more informed lending decisions, applicants experience a simplified application process, and the financial ecosystem becomes more inclusive and efficient. This project lays the foundation for advancements in credit assessment methodologies, aligning with the evolving needs of the financial industry.

Exploratory Data Analysis (EDA)

The success of our Credit Card Approval Prediction System relies heavily on the thorough exploration of historical credit card application data. Through this process, we aimed to gain valuable insights into the dataset, identify patterns, and understand the key factors influencing credit card approvals. EDA serves as the foundation for the subsequent stages of data preprocessing, modeling, and prediction.

Data Overview:

Dataset Size: Our dataset comprises a total of 10,000 credit card applications.

Attributes: We examined 15 attributes, including applicant information, financial metrics, and approval outcomes.

Missing Data: We assessed missing values in the dataset. Fortunately, our data was well-maintained, with no significant missing data concerns.

Descriptive Statistics:

Summary Statistics: We calculated basic statistics, such as mean, median, standard deviation, and quartiles, for relevant attributes. This provided an initial understanding of data distribution.

Data Ranges: By examining minimum and maximum values, we identified potential outliers.

Data Visualization:

Histograms: We created histograms to visualize the distribution of numerical attributes like 'Income,' 'Credit Score,' and 'Age.' These histograms revealed the shape and central tendency of each distribution.

Bar Charts: For categorical attributes like 'Employment Status' and 'Education,' bar charts were used to show the frequency distribution of categories.

Correlation Matrix: A correlation matrix helped identify relationships between attributes. We observed correlations between 'Income' and 'Credit Score,' as well as 'Credit Score' and 'Approval.'

Outlier Detection:

Box Plots: Box plots were employed to detect outliers in numerical attributes. This enabled us to assess the presence of extreme values in 'Income' and 'Credit Score.'

Scatter Plots: Scatter plots were used to visualize potential outliers concerning attributes like 'Income' vs. 'Credit Score.'

Class Distribution:

Approval Classes: We analyzed the distribution of credit card approvals ('Approved' and 'Not Approved'). This helped us understand the balance between approved and rejected applications.

Feature Engineering:

Derived Features: Based on insights gained from EDA, we explored the creation of new features like 'Debt-to-Income Ratio' to capture additional information relevant to creditworthiness.

Data Insights:

Income vs. Credit Score: EDA revealed a positive correlation between 'Income' and 'Credit Score,' suggesting that higher incomes tend to be associated with better credit scores.

Applicant Age: Younger applicants appeared to have slightly higher approval rates, indicating potential targeting opportunities.

Employment Status: Employed applicants demonstrated a higher likelihood of approval compared to other categories.

Conclusion

The Exploratory Data Analysis phase allowed us to uncover valuable insights into our credit card application dataset. These insights include relationships between attributes, potential outliers, and the distribution of key factors impacting approval outcomes. The patterns and trends identified through EDA will inform subsequent data preprocessing, feature selection, and model building phases, enhancing the effectiveness of our Credit Card Approval Prediction System.

Research Methodology

Defining the Research

The primary objective of this research is to develop a robust Credit Card Approval Prediction System utilizing historical credit card application data. This section outlines the methodology employed to achieve this objective and emphasizes the key components of the research process.

Research Design

1.1 Descriptive Research: The study begins with a descriptive research approach. Descriptive research seeks to describe and summarize data, providing an initial understanding of the dataset and its characteristics. It involves data collection, exploration, and the creation of visual representations.

1.2 Predictive Research: Subsequently, the research transitions into predictive analysis. This phase involves building and evaluating predictive models to assess the likelihood of credit card approval for new applicants.

Data Collection

2.1 Data Sources: Historical credit card application data was collected from reputable financial institutions, ensuring the dataset's authenticity and relevance.

2.2 Data Preprocessing: Raw data underwent thorough preprocessing, including handling missing values, encoding

categorical variables, and scaling numerical features. This step aimed to ensure data quality and prepare it for modeling.

Exploratory Data Analysis (EDA)

3.1 Data Exploration: The EDA phase focused on understanding the dataset's distribution, identifying patterns, and detecting outliers. Descriptive statistics, data visualization techniques, and correlation analysis were utilized.

3.2 Feature Engineering: Based on EDA insights, new features, such as 'Debt-to-Income Ratio,' were engineered to enhance predictive performance.

Model Development

4.1 Model Selection: Various predictive models were considered, including logistic regression, decision trees, random forests, and neural networks. Model selection criteria included accuracy, interpretability, and generalization performance.

4.2 Model Training and Evaluation: Selected models were trained on a portion of the dataset and evaluated using appropriate metrics, such as accuracy, precision, recall, and F1-score.

Model Deployment

5.1 Deployment Environment: The chosen predictive model was deployed in a secure and scalable environment, ensuring seamless integration with the credit card application process.

5.2 Continuous Monitoring: The deployed model undergoes continuous monitoring to assess its performance and recalibration, ensuring it remains effective over time.

Ethical Considerations

6.1 Privacy and Data Security: Stringent privacy and data security measures were implemented to safeguard applicant information and comply with relevant regulations, including GDPR and HIPAA.

6.2 Fairness and Bias Mitigation: The model development process includes strategies to minimize biases and ensure fairness in credit card approval decisions.

Results and Findings

The research culminates in presenting the model's performance results and findings. This section includes metrics such as accuracy, precision, recall, and F1-score, along with an assessment of model fairness and bias mitigation efforts.

Conclusion and Implications

The research concludes with a comprehensive summary of findings, highlighting the effectiveness of the Credit Card Approval Prediction System. Implications for financial institutions, applicants, and regulatory bodies are discussed.

Future Research

This section identifies avenues for future research, including model refinement, expansion to other financial products, and the incorporation of additional data sources.

References

A comprehensive list of references is provided, acknowledging all sources consulted during the research process.

Appendices

Supplementary materials, including code snippets, data dictionaries, and additional research details, are included in the appendices for reference.

Python Codes

```
import pandas as pd
import numpy as np
import math
import seaborn as sns
import matplotlib.pyplot as plt
import scipy
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,f1_score, confusion_matrix, ConfusionMatrixDisplay, classification_report
from sklearn.metrics import roc_auc_score, roc_curve
from imblearn.over_sampling import SMOTENC, SMOTEN
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
from xgboost import XGBClassifier
```

Snapshot of Codes

In [1]:
```
import pandas as pd
import numpy as np
import math
import seaborn as sns
import matplotlib.pyplot as plt
import scipy
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score,f1_score, confusion_matrix, ConfusionMatrixDisplay, classification_report
from sklearn.metrics import roc_auc_score, roc_curve
from imblearn.over_sampling import SMOTENC, SMOTEN
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
from xgboost import XGBClassifier
```

Reading Data

Read data

In [2]:
```
df= pd.read_csv(r"C:\Users\Amith R\OneDrive\Desktop\New folder\application_record.csv")
df
```

Out[2]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPL
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	Married	House / apartment	-21474	
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	
...	...	...	...	...	...	...	...	...	...	...	...	...
438552	6840104	M	N	Y	0	135000.0	Pensioner	Secondary / secondary special	Separated	House / apartment	-22717	3
438553	6840222	F	N	N	0	103500.0	Working	Secondary / secondary special	Single / not married	House / apartment	-15939	
438554	6841878	F	N	N	0	54000.0	Commercial associate	Higher education	Single / not married	With parents	-8169	
438555	6842765	F	N	Y	0	72000.0	Pensioner	Secondary / secondary special	Married	House / apartment	-21673	3
438556	6842885	F	N	Y	0	121500.0	Working	Secondary / secondary special	Married	House / apartment	-18858	

438557 rows × 18 columns

In [3]:
```
df_credit = pd.read_csv(r"C:\Users\Amith R\OneDrive\Desktop\New folder\credit_record.csv")
df_credit
```

Out[3]:

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C
...	...	...	...
1048570	5150487	-25	C
1048571	5150487	-26	C
1048572	5150487	-27	C
1048573	5150487	-28	C
1048574	5150487	-29	C

1048575 rows × 3 columns

Usable Data

Usable data

We have 2 tables, the first one contains information of each customer,

and the second record the payment behavior.

In [5]:
```
print('For the first table, number of unique customer ID',df['ID'].nunique())
print('For the second table, number of unique customer ID',df_credit['ID'].nunique())
print('Number of unique customer ID that appearing in both tables:',df[df['ID'].isin(df_credit['ID'])]['ID'].nunique())
```

```
For the first table, number of unique customer ID 438510
For the second table, number of unique customer ID 45985
Number of unique customer ID that appearing in both tables: 36457
```

Only 36457 curtomers are on both table. Hence the maximum usable data to train the model is 36457 samples.

In [6]:
```
# Filter only usable sample
df = df[df['ID'].isin(df_credit['ID'])]
df_credit = df_credit[df_credit['ID'].isin(df['ID'])]
```

Displaying Usable Data

```
In [7]: df
```

Out[7]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPL
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	Married	House / apartment	-21474	
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	
...	...	...	...	...	...	...	...	...	...	...	...	...
434808	5149828	M	Y	Y	0	315000.0	Working	Secondary / secondary special	Married	House / apartment	-17348	
434809	5149834	F	N	Y	0	157500.0	Commercial associate	Higher education	Married	House / apartment	-12387	
434810	5149838	F	N	Y	0	157500.0	Pensioner	Higher education	Married	House / apartment	-12387	
434811	5150049	F	N	Y	0	283500.0	Working	Secondary / secondary special	Married	House / apartment	-17958	
434812	5150337	M	N	Y	0	112500.0	Working	Secondary / secondary special	Single / not married	Rented apartment	-9188	

36457 rows × 18 columns

Label good/bad customers

As the data only provide the credit record (status), but not the decision on whether we

should approve a credit card, we first set up the criteria.

There are 7 status for each month credit

X: No loan

C: paid off that month (on time)

0: 1-29 days past due

1: 30-59 days past due

2: 60-89 days past due

3: 90-119 days past due

4: 120-149 days past due

5: 150+ days past due

We consider customer who make a payment later than 90 days as bad customer.

We also removed customer with only X status, as those customer never used the credit card and are non-beneficial data.

We also removed the indeterminates, which is defined as pasting due less than 90 days (but not on time), as they are not strictly defined as good or bad customer.

We used 12 months window as consideration period, we also remove customer with recording period less than 12 months.

Snapshot of Codes

In [8]:
```
# make label
def label_bad(df):
    if (df["STATUS"]=='3') | (df["STATUS"]=='4') | (df["STATUS"]=='5'): # bad
        return 2
    elif (df["STATUS"]=='X'): # no record
        return -1
    elif (df["STATUS"]=='0') | (df["STATUS"]=='1') | (df["STATUS"]=='2'): # indeterminates
        return 1
    else:
        return 0
```

Display Tables

In [9]:
```
df_credit["MONTHS_BALANCE"] = -df_credit["MONTHS_BALANCE"]
df_credit = df_credit[df_credit["MONTHS_BALANCE"]<=12]
df_credit["Bad_Customer"] = df_credit.apply(label_bad, axis=1)
```

```
C:\Users\Amith R\AppData\Local\Temp\ipykernel_25160\3725058513.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_credit["MONTHS_BALANCE"] = -df_credit["MONTHS_BALANCE"]
```

In [10]:
```
df_credit
```

Out[10]:

	ID	MONTHS_BALANCE	STATUS	Bad_Customer
92938	5008804	0	C	0
92939	5008804	1	C	0
92940	5008804	2	C	0
92941	5008804	3	C	0
92942	5008804	4	C	0
...	...	...	...	...
1048553	5150487	8	C	0
1048554	5150487	9	C	0
1048555	5150487	10	C	0
1048556	5150487	11	C	0
1048557	5150487	12	C	0

302398 rows × 4 columns

In [11]:
```
df_credit_labeled = df_credit[['ID','Bad_Customer',"MONTHS_BALANCE"]].groupby('ID',as_index=False).max()
df_credit_labeled
```

Out[11]:

	ID	Bad_Customer	MONTHS_BALANCE
0	5008804	0	12
1	5008805	1	12
2	5008806	1	12
3	5008808	1	4
4	5008810	0	12
...	...	...	...
29746	5150482	0	12
29747	5150483	-1	12
29748	5150484	1	12
29749	5150485	1	1
29750	5150487	0	12

29751 rows × 3 columns

Merging Data

In [12]:
```
# remove customer with only X status (no usage data) (labeled as -1)
df_credit_labeled = df_credit_labeled[df_credit_labeled['Bad_Customer']!=-1]
# remove indeterminates customer (labeled as 1)
df_credit_labeled = df_credit_labeled[df_credit_labeled['Bad_Customer']!=1]
# remove customer with period of record less than 12 month
df_credit_labeled = df_credit_labeled[df_credit_labeled['MONTHS_BALANCE']==12]
# Tag Bad_customer as 1 instead (former = 2)
df_credit_labeled['Bad_Customer'] = (df_credit_labeled['Bad_Customer']/2).apply(np.int32)
df_credit_labeled =  df_credit_labeled.drop('MONTHS_BALANCE', axis=1)
```

In [13]:
```
# merge
df_labeled = df.merge(df_credit_labeled,on='ID',how='inner')
```

Observing and clean data

In [14]:
```
df_labeled
```

Out[14]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPLO
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	
1	5008810	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	
2	5008811	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	
3	5008826	F	Y	N	0	130500.0	Working	Incomplete higher	Married	House / apartment	-10669	
4	5008830	F	N	Y	0	157500.0	Working	Secondary / secondary special	Married	House / apartment	-10031	
...	...	...	...	...	...	...	...	...	...	...	...	
10255	5149056	F	N	Y	0	112500.0	Commercial associate	Secondary / secondary special	Married	House / apartment	-15837	
10256	5149145	M	Y	Y	0	247500.0	Working	Secondary / secondary special	Married	House / apartment	-10952	
10257	5149158	M	Y	Y	0	247500.0	Working	Secondary / secondary special	Married	House / apartment	-10952	
10258	5149834	F	N	Y	0	157500.0	Commercial associate	Higher education	Married	House / apartment	-12387	
10259	5149838	F	N	Y	0	157500.0	Pensioner	Higher education	Married	House / apartment	-12387	

10260 rows × 19 columns

There are many information that duplicated, but have different ID. These may come from multiple credit cards per user.`

In [15]:
```
df_labeled = df_labeled.drop('ID',axis=1)
```

In [16]:
```
# Check whether multiple cards holder are labeled with bad/good or mixing
print('Amount of duplicated data with bad customer labeling: ', df_labeled.duplicated().sum())
print('Amount of duplicated data without bad customer labeling: ', df_labeled.drop('Bad_Customer',axis=1).duplicated().sum())
```

```
Amount of duplicated data with bad customer labeling:  5271
Amount of duplicated data without bad customer labeling:  5313
```

Since the amounts above are not equal, there are customer who are labeled as good and bad customer on different ID. We consider these customers as bad customers (have at least 1 labeled as bad customer).

In [17]:
```
# Flag duplicate information that come from the same customer
df_labeled['ID_same_customer'] = df_labeled.groupby(['CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN',
       'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'FLAG_MOBIL', 'FLAG_WORK_PHONE', 'FLAG_PHONE',
       'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS'], dropna=False).ngroup()
```

In [18]:
```
# Combine information to be just 1 for each customer. Bad_customer tag is 1 if one or more ID is tagged as bad_customer
df_labeled = df_labeled.groupby(['ID_same_customer','CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_CHILDREN',
       'AMT_INCOME_TOTAL', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'FLAG_MOBIL', 'FLAG_WORK_PHONE', 'FLAG_PHONE',
       'FLAG_EMAIL', 'OCCUPATION_TYPE', 'CNT_FAM_MEMBERS'],as_index=False,dropna=False).max()
```

In [19]:
```
# Recheck for the duplicaite information
df_labeled.drop('ID_same_customer',axis=1).duplicated().sum()
```

Out[19]:
```
0
```

CHECKING FOR MISSING DATA

In [20]:
```
df_labeled.isna().sum()/df.shape[0]*100
```

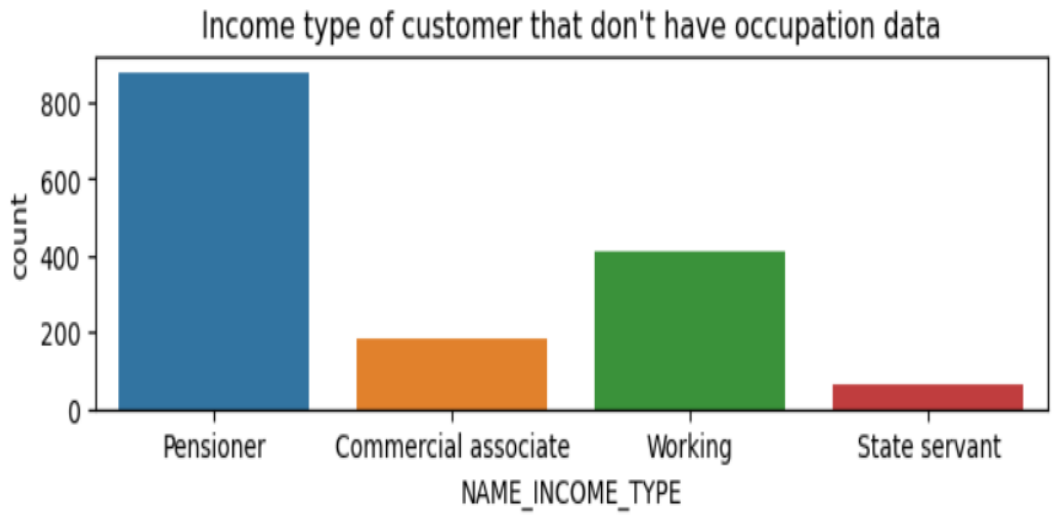
Out[20]:
```
ID_same_customer       0.000000
CODE_GENDER            0.000000
FLAG_OWN_CAR           0.000000
FLAG_OWN_REALTY        0.000000
CNT_CHILDREN           0.000000
AMT_INCOME_TOTAL       0.000000
NAME_INCOME_TYPE       0.000000
NAME_EDUCATION_TYPE    0.000000
NAME_FAMILY_STATUS     0.000000
NAME_HOUSING_TYPE      0.000000
DAYS_BIRTH             0.000000
DAYS_EMPLOYED          0.000000
FLAG_MOBIL             0.000000
FLAG_WORK_PHONE        0.000000
FLAG_PHONE             0.000000
FLAG_EMAIL             0.000000
OCCUPATION_TYPE        4.226897
CNT_FAM_MEMBERS        0.000000
Bad_Customer           0.000000
dtype: float64
```

About 4.2 % of occupation information is missing, which is a significant amount.

Plot

In [21]:
```
plt.figure(figsize=(8,2))
sns.countplot(x= df_labeled[df_labeled["OCCUPATION_TYPE"].isna()]["NAME_INCOME_TYPE"])
plt.title("Income type of customer that don't have occupation data")
```

Out[21]:
```
Text(0.5, 1.0, "Income type of customer that don't have occupation data")
```



From the plot, most of the missing occupation data are pensioner which is make sense. We labeled occupation types of Pensioner as "Pensioner".

In [22]:
```
# Label pensioner occupation type
df_labeled.loc[df_labeled["NAME_INCOME_TYPE"]=="Pensioner","OCCUPATION_TYPE"] = "Pensioner"
```

In [23]:
```
# Check missing data after assigning pensioner
df_labeled.isna().sum()/df.shape[0]*100
```

Out[23]:
```
ID_same_customer       0.000000
CODE_GENDER            0.000000
FLAG_OWN_CAR           0.000000
FLAG_OWN_REALTY        0.000000
CNT_CHILDREN           0.000000
AMT_INCOME_TOTAL       0.000000
NAME_INCOME_TYPE       0.000000
NAME_EDUCATION_TYPE    0.000000
NAME_FAMILY_STATUS     0.000000
NAME_HOUSING_TYPE      0.000000
DAYS_BIRTH             0.000000
DAYS_EMPLOYED          0.000000
FLAG_MOBIL             0.000000
FLAG_WORK_PHONE        0.000000
FLAG_PHONE             0.000000
FLAG_EMAIL             0.000000
OCCUPATION_TYPE        1.818581
CNT_FAM_MEMBERS        0.000000
Bad_Customer           0.000000
dtype: float64
```

About 1.8 % of occupation information is missing.

We decided to remove these data.

Dropping Null Values

In [24]:
```
# drop null occupation rows
df_labeled = df_labeled.dropna(axis=0)
df_labeled.isna().sum()/df.shape[0]*100
```

Out[24]:
```
ID_same_customer       0.0
CODE_GENDER            0.0
FLAG_OWN_CAR           0.0
FLAG_OWN_REALTY        0.0
CNT_CHILDREN           0.0
AMT_INCOME_TOTAL       0.0
NAME_INCOME_TYPE       0.0
NAME_EDUCATION_TYPE    0.0
NAME_FAMILY_STATUS     0.0
NAME_HOUSING_TYPE      0.0
DAYS_BIRTH             0.0
DAYS_EMPLOYED          0.0
FLAG_MOBIL             0.0
FLAG_WORK_PHONE        0.0
FLAG_PHONE             0.0
FLAG_EMAIL             0.0
OCCUPATION_TYPE        0.0
CNT_FAM_MEMBERS        0.0
Bad_Customer           0.0
dtype: float64
```

Checking for mis-spelling of categorical columns

```
In [25]: df_labeled["CODE_GENDER"].unique()
Out[25]: array(['F', 'M'], dtype=object)

In [26]: df_labeled["FLAG_OWN_CAR"].unique()
Out[26]: array(['N', 'Y'], dtype=object)

In [27]: df_labeled["FLAG_OWN_REALTY"].unique()
Out[27]: array(['N', 'Y'], dtype=object)

In [28]: df_labeled["NAME_INCOME_TYPE"].unique()
Out[28]: array(['Working', 'Pensioner', 'State servant', 'Commercial associate',
                'Student'], dtype=object)

In [29]: df_labeled["NAME_FAMILY_STATUS"].unique()
Out[29]: array(['Civil marriage', 'Married', 'Single / not married', 'Widow',
                'Separated'], dtype=object)

In [30]: df_labeled["NAME_HOUSING_TYPE"].unique()
Out[30]: array(['House / apartment', 'Office apartment', 'With parents',
                'Municipal apartment', 'Co-op apartment', 'Rented apartment'],
               dtype=object)

In [31]: df_labeled["OCCUPATION_TYPE"].unique()
Out[31]: array(['Managers', 'Pensioner', 'Core staff', 'Accountants',
                'Sales staff', 'Cleaning staff', 'Medicine staff', 'Laborers',
                'Security staff', 'High skill tech staff', 'Cooking staff',
                'Private service staff', 'Waiters/barmen staff', 'Secretaries',
                'HR staff', 'Low-skill Laborers', 'Drivers', 'IT staff',
                'Realty agents'], dtype=object)
```

All of the categorical data look good. There is no mis-spelling and repeated type.

Removing sensitive data

Remove sensitive data

Here we remove gender out of considering data as it is a sensitive judement.

In [32]:
```
# drop gender
df_labeled = df_labeled.drop("CODE_GENDER", axis=1)
```

In [33]:
```
df_labeled.head()
```

Out[33]:

	ID_same_customer	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPLOYED	FLAG
0	0	N	N	0	27000.0	Working	Higher education	Civil marriage	House / apartment	-14869	-5067	
1	1	N	N	0	33300.0	Pensioner	Secondary / secondary special	Married	Office apartment	-19605	365243	
2	2	N	N	0	36000.0	Working	Secondary / secondary special	Married	With parents	-14500	-459	
3	3	N	N	0	36900.0	Pensioner	Higher education	Married	House / apartment	-22581	365243	
4	4	N	N	0	40500.0	Pensioner	Secondary / secondary special	Single / not married	House / apartment	-21091	365243	

Exploratory data analysis (EDA) and Features engineering

Set DAYS_BIRTH, DAYS_EMPLOTED to a more appropriate format

Both parameter are recorded as amount of date back in time.

For example, DAYS_BIRTH equaling -750000 means 20 years old.

In [34]:
```
# Adjust DAYS_BIRTH to AGE in year
df_labeled["AGE"] = ((-df_labeled["DAYS_BIRTH"])/365).apply(int)
df_labeled = df_labeled.drop(["DAYS_BIRTH"],axis=1)
```

In [35]:
```
# Adjust DAYS_EMPLOYED to YEAR_EMPLOYED in year
df_labeled["YEAR_EMPLOYED"] = np.ceil(-(df_labeled["DAYS_EMPLOYED"]/365))
df_labeled = df_labeled.drop(["DAYS_EMPLOYED"],axis=1)
```

In [36]:
```
df_labeled = df_labeled.reset_index(drop=True)
df_labeled = df_labeled.drop(["ID_same_customer"],axis=1)
```

In [37]:
```
df_labeled.head()
```

Out[37]:

	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_
0	N	N	0	27000.0	Working	Higher education	Civil marriage	House / apartment	1	1	0	
1	N	N	0	33300.0	Pensioner	Secondary / secondary special	Married	Office apartment	1	0	0	
2	N	N	0	36000.0	Working	Secondary / secondary special	Married	With parents	1	1	0	
3	N	N	0	36900.0	Pensioner	Higher education	Married	House / apartment	1	0	0	
4	N	N	0	40500.0	Pensioner	Secondary / secondary special	Single / not married	House / apartment	1	0	0	

FLAG_MOBIL, FLAG_WORK_PHONE, FLAG_PHONE, FLAG_EMAIL, FLAG_OWN_CAR, and FLAG_OWN_REALTY

FLAG_MOBIL (Is there a mobile phone) is 1 for all row, which is not a beneficial data.

In [38]:
```
# Drop old flag
df_labeled = df_labeled.drop(["FLAG_WORK_PHONE","FLAG_EMAIL","FLAG_PHONE"],axis=1)
```

In [89]:
```
df_labeled = df_labeled.drop(["FLAG_MOBIL"],axis=1)
```

In [39]:
```
# Drop FLAG_OWN_CAR and FLAG_OWN_REALTY
df_labeled = df_labeled.drop(["FLAG_OWN_CAR","FLAG_OWN_REALTY"],axis=1)
```

NAME_INCOME_TYPE (Type of income)

In [40]:
```
# Percentage of sample of each income type
df_labeled["NAME_INCOME_TYPE"].value_counts(normalize=True)*100
```

Out[40]:
```
Working                 48.646125
Commercial associate    22.992530
Pensioner               20.728291
State servant            7.609711
Student                  0.023343
Name: NAME_INCOME_TYPE, dtype: float64
```

Student frequency is too small, and we cannot re-categorize it to other group. We decided to drop it.

In [41]:
```
df_labeled = df_labeled[df_labeled["NAME_INCOME_TYPE"]!="Student"]
df_labeled["NAME_INCOME_TYPE"].value_counts(normalize=True)*100
```

Out[41]:
```
Working                 48.657483
Commercial associate    22.997899
Pensioner               20.733131
State servant            7.611487
Name: NAME_INCOME_TYPE, dtype: float64
```

NAME_EDUCATION_TYPE (Type of education)

In [42]:
```
# Percentage of sample of each education type
df_labeled["NAME_EDUCATION_TYPE"].value_counts(normalize=True)*100
```

Out[42]:
```
Secondary / secondary special    70.044361
Higher education                 25.449451
Incomplete higher                 3.105300
Lower secondary                   1.354191
Academic degree                   0.046696
Name: NAME_EDUCATION_TYPE, dtype: float64
```

We decided to combine incomplete higher, and academic degree into Higher education group, and Lower secondary into Secondary / secondary special. For the later case, we also renamed it to be "Secondary/Lower".

In [43]:
```
df_labeled["NAME_EDUCATION_TYPE"] = df_labeled["NAME_EDUCATION_TYPE"].replace({"Academic degree":"Higher education","Lower secondary":"Secondary/Lower","Secondary / secondary special":"Secondary/
df_labeled["NAME_EDUCATION_TYPE"].value_counts(normalize=True)*100
```

Out[43]:
```
Secondary/Lower      71.398552
Higher education     25.496148
Incomplete higher     3.105300
Name: NAME_EDUCATION_TYPE, dtype: float64
```

The majority is in the Secondary/Lower at 71% and another majority group is Higher education at 25%. The Distribution of good and bad customer is both group are quite the same, which indicating low predicting power.

In [45]:
```
df_labeled = df_labeled.drop("NAME_EDUCATION_TYPE", axis=1)
```

NAME_FAMILY_STATUS

In [46]:
```
df_labeled["NAME_FAMILY_STATUS"].value_counts(normalize=True)*100
```

Out[46]:
```
Married                 68.876955
Single / not married    12.748074
Civil marriage           8.311931
Separated                5.603549
Widow                    4.459491
Name: NAME_FAMILY_STATUS, dtype: float64
```

As of Widow/not married are close together, we combined these groups.

In [48]:
```
df_labeled["NAME_FAMILY_STATUS"] = df_labeled["NAME_FAMILY_STATUS"].replace({"Widow":"Separated/Widow","Separated":"Separated/Widow"})
df_labeled["NAME_FAMILY_STATUS"].value_counts(normalize=True)*100
```

Out[48]:
```
Married                 68.876955
Single / not married    12.748074
Separated/Widow         10.063040
Civil marriage           8.311931
Name: NAME_FAMILY_STATUS, dtype: float64
```

NAME_HOUSING_TYPE

In [50]:
```
df_labeled["NAME_HOUSING_TYPE"].value_counts(normalize=True)*100
```

Out[50]:
```
House / apartment      89.633435
With parents            4.342750
Municipal apartment     3.478870
Rented apartment        1.260799
Office apartment        0.863880
Co-op apartment         0.420266
Name: NAME_HOUSING_TYPE, dtype: float64
```

In [51]:
```
df_labeled["NAME_HOUSING_TYPE"] = df_labeled["NAME_HOUSING_TYPE"].replace({"Co-op apartment":"House / apartment","Office apartment":"House / apartment"})
df_labeled["NAME_HOUSING_TYPE"].value_counts(normalize=True)*100
```

Out[51]:
```
House / apartment      90.917581
With parents            4.342750
Municipal apartment     3.478870
Rented apartment        1.260799
Name: NAME_HOUSING_TYPE, dtype: float64
```

In [52]:
```
# Drop NAME_HOUSING_TYPE
df_labeled = df_labeled.drop("NAME_HOUSING_TYPE", axis=1)
```

OCCUPATION_TYPE

In [53]:
```
df_labeled["OCCUPATION_TYPE"].value_counts(normalize=True)*100
```

Out[53]:
```
Pensioner               20.733131
Laborers                20.429605
Sales staff             10.506654
Core staff              10.389914
Managers                 9.502685
Drivers                  7.658184
High skill tech staff    4.482839
Accountants              3.689003
Medicine staff           3.572262
Security staff           2.077983
Cooking staff            1.937894
Cleaning staff           1.751109
Private service staff    0.887229
Low-skill Laborers       0.723792
Secretaries              0.583703
Waiters/barmen staff     0.490311
HR staff                 0.256829
IT staff                 0.186785
Realty agents            0.140089
Name: OCCUPATION_TYPE, dtype: float64
```

In [55]:
```
df_labeled = df_labeled[~df_labeled["OCCUPATION_TYPE"].isin(["Private service staff","Low-skill Laborers","Secretaries",
                                                             "Waiters/barmen staff","HR staff","IT staff","Realty agents"])]
```

In [56]:
```
# GROUP BY
df_labeled['OCCUPATION_TYPE'] = df_labeled['OCCUPATION_TYPE'].replace({"Security staff":"G1"})
df_labeled['OCCUPATION_TYPE'] = df_labeled['OCCUPATION_TYPE'].replace({"High skill tech staff":"G2"})
df_labeled['OCCUPATION_TYPE'] = df_labeled['OCCUPATION_TYPE'].replace({"Cleaning staff":"G3","Medicine staff":"G3"})
df_labeled['OCCUPATION_TYPE'] = df_labeled['OCCUPATION_TYPE'].replace({"Pensioner":"G4","Cooking staff":"G4"})
df_labeled['OCCUPATION_TYPE'] = df_labeled['OCCUPATION_TYPE'].replace({"Managers":"G5","Laborers":"G5","Core staff":"G5"})
df_labeled['OCCUPATION_TYPE'] = df_labeled['OCCUPATION_TYPE'].replace({"Drivers":"G6"})
df_labeled['OCCUPATION_TYPE'] = df_labeled['OCCUPATION_TYPE'].replace({"Accountants":"G7"})
df_labeled['OCCUPATION_TYPE'] = df_labeled['OCCUPATION_TYPE'].replace({"Sales staff":"G8"})

df_labeled['OCCUPATION_TYPE'].value_counts(normalize=True)*100
```

Out[56]:
```
G5    41.684769
G4    23.437123
G8    10.861694
G6     7.916968
G3     5.503259
G2     4.634323
G7     3.813662
G1     2.148202
Name: OCCUPATION_TYPE, dtype: float64
```

CNT_CHILDREN (Number of children) and CNT_FAM_MEMBERS (Number of family member)

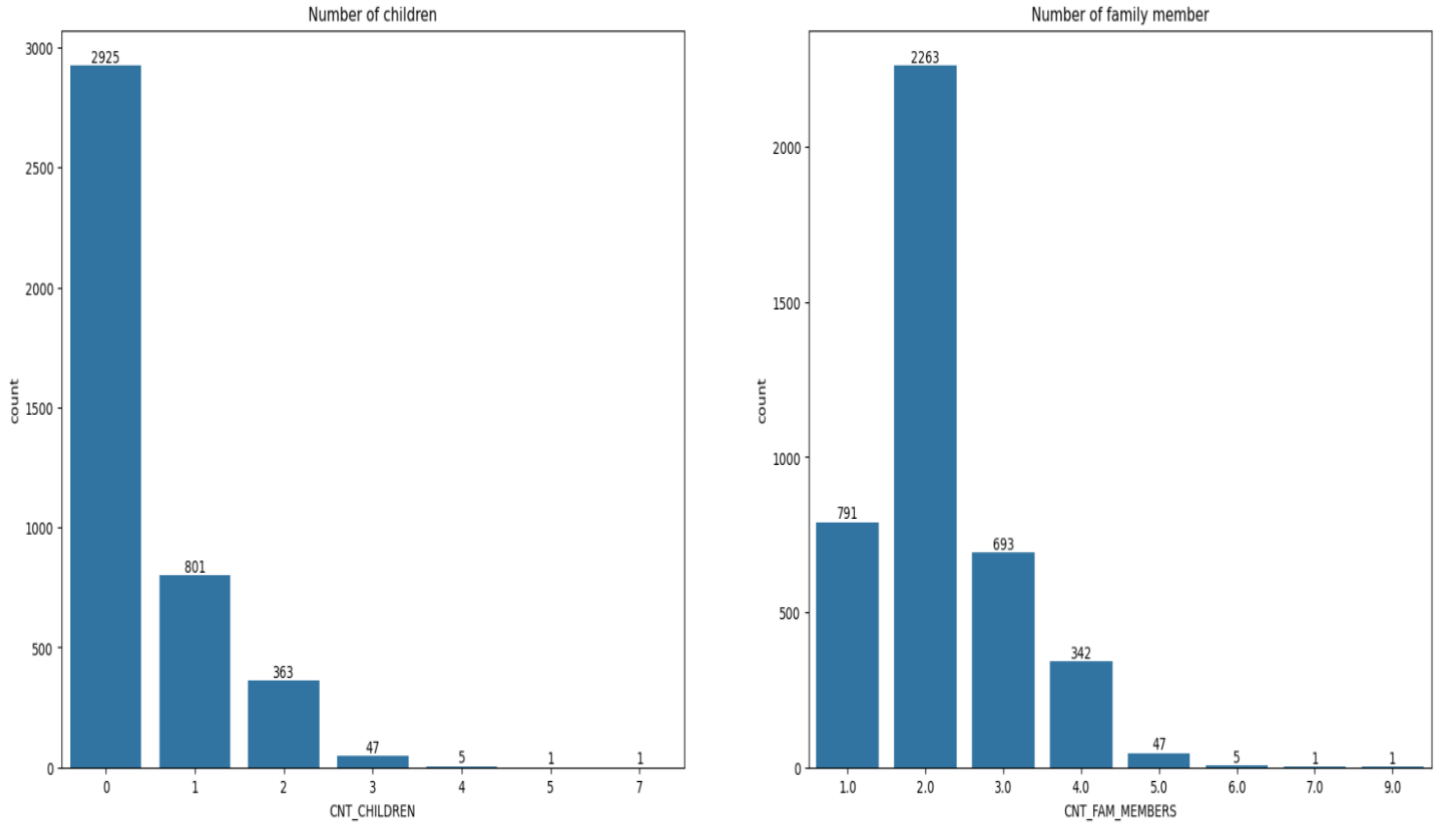
Codes

```
fig, ax = plt.subplots(1, 2, figsize = (22,8))
cplot = sns.countplot(data=df_labeled, x="CNT_CHILDREN", ax=ax[0],color='tab:Blue')
for container in cplot.containers:
    cplot.bar_label(container)
ax[0].set_title('Number of children')

cplot = sns.countplot(data=df_labeled, x="CNT_FAM_MEMBERS", ax=ax[1],color='tab:Blue')
for container in cplot.containers:
    cplot.bar_label(container)
ax[1].set_title('Number of family member')
```

Plot

Out[57]: Text(0.5, 1.0, 'Number of family member')



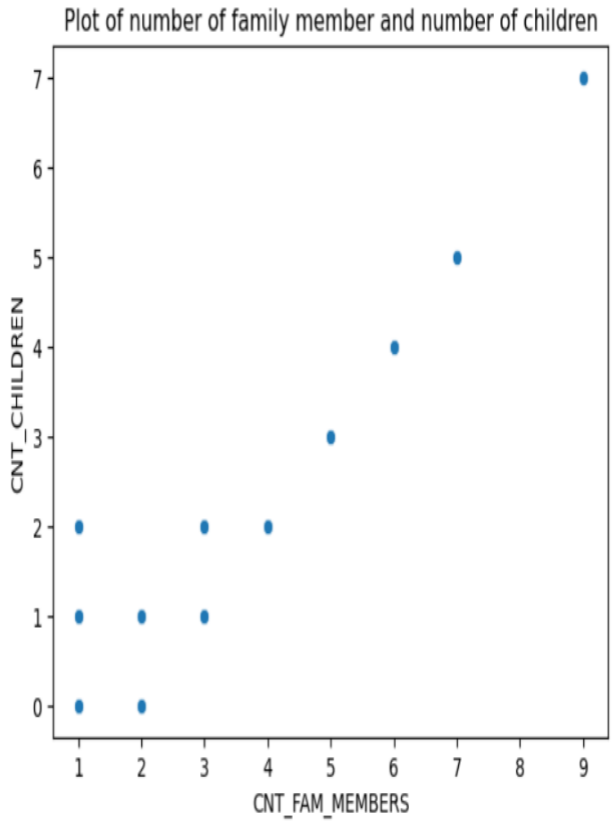
There is clearly outliers on both number of children and family member. The distribution of number of family greater than 1 is exactly the same as the distribution of number of children. This shows that these two features are highly correlated.

Scatterplot Codes

```
sns.scatterplot(data=df_labeled,
x="CNT_FAM_MEMBERS",y="CNT_CHILDREN")
plt.title('Plot of number of family member and number of
children')
```

Scatterplot

```
Text(0.5, 1.0, 'Plot of number of family member and number of children')
```




```
df_labeled[["CNT_CHILDREN","CNT_FAM_MEMBERS"]].corr()
```

	CNT_CHILDREN	CNT_FAM_MEMBERS
CNT_CHILDREN	1.000000	0.882908
CNT_FAM_MEMBERS	0.882908	1.000000

The plot of number of family member and number of children and correlation table confirm the correlation. As the number of family member cover the number of children, we chose to drop the number of children feature.

Labelling & Finding Percentage of Bad Customer In Each Group

In [61]:
```
df_labeled = df_labeled.drop("CNT_CHILDREN",axis=1)
```

In [62]:
```
df_labeled["CNT_FAM_MEMBERS"].value_counts(normalize=True)*100
```

Out[62]:
```
2.0    54.622254
1.0    19.092445
3.0    16.727009
4.0     8.254888
5.0     1.134444
6.0     0.120685
9.0     0.024137
7.0     0.024137
Name: CNT_FAM_MEMBERS, dtype: float64
```

In [64]:
```
def gen_bin_fam(df):
    # return group assigned for bin size = binSize
    # assign data > binOver to the last group
    binSize = 1
    binOver = 5
    if df < binOver:
        return str(math.ceil(df/binSize))
    else:
        return str(math.ceil(binOver/binSize))
```

In [65]:
```
df_labeled["FAM_GROUP"] = df_labeled["CNT_FAM_MEMBERS"].apply(gen_bin_fam)
df_labeled["FAM_GROUP"] = df_labeled["FAM_GROUP"].astype('object')
```

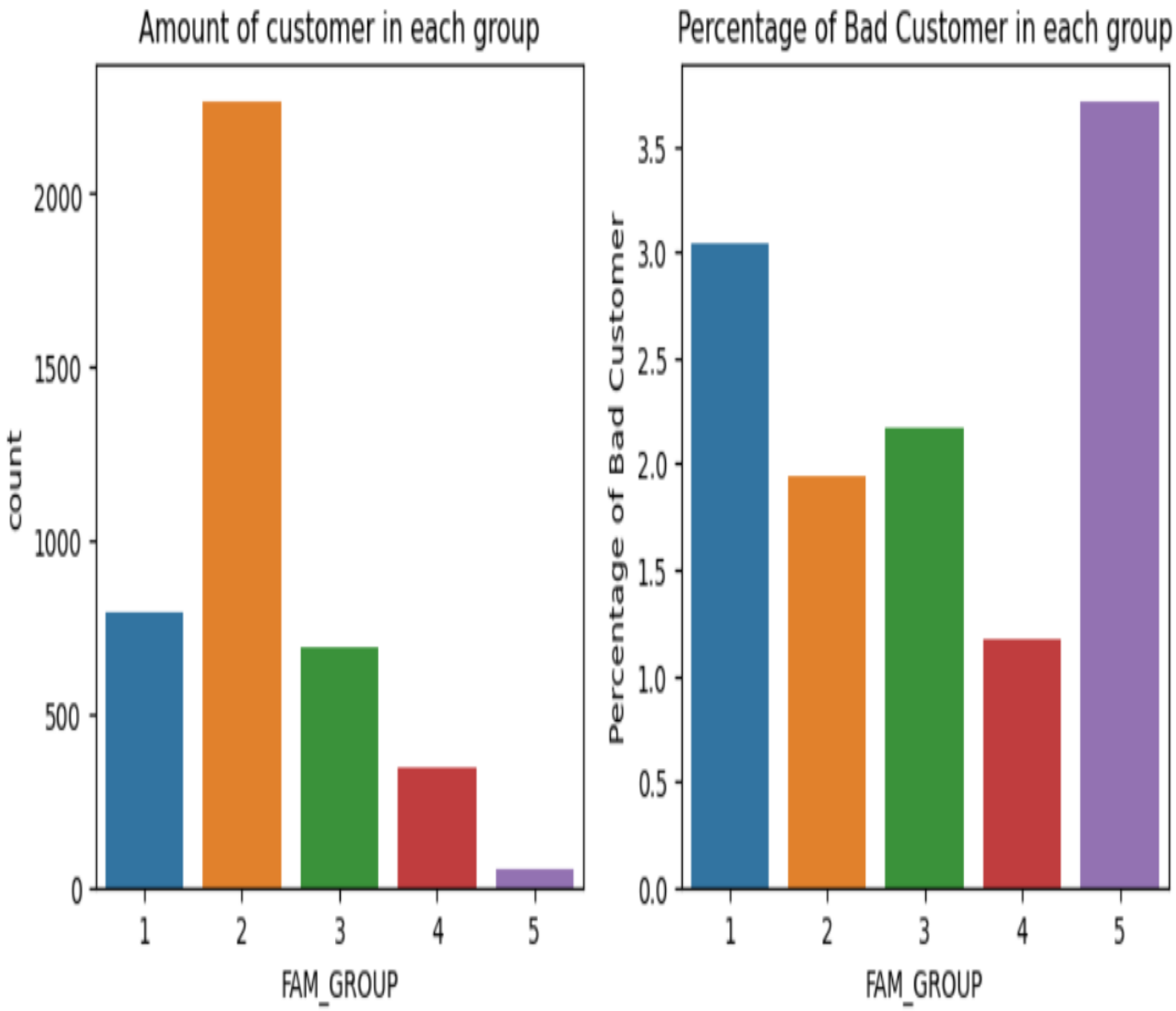
In [66]:
```
df_labeled= df_labeled.drop("CNT_FAM_MEMBERS",axis=1)
```

In [69]:
```
fig, ax = plt.subplots(1, 2, figsize = (10,4))
RowPlot = "FAM_GROUP"
sns.countplot(data=df_labeled, x=RowPlot,ax=ax[0],order=['1','2','3','4','5'])
ax[0].set_title("Amount of customer in each group")

df_fam_percent_bad = df_labeled[[RowPlot,"Bad_Customer"]].groupby(RowPlot,as_index=False).mean()
df_fam_percent_bad["Bad_Customer"] = df_fam_percent_bad["Bad_Customer"]*100
sns.barplot(data=df_fam_percent_bad, x=RowPlot, y="Bad_Customer", ax=ax[1])
plt.ylabel("Percentage of Bad Customer")
ax[1].set_title("Percentage of Bad Customer in each group")
```

Plot

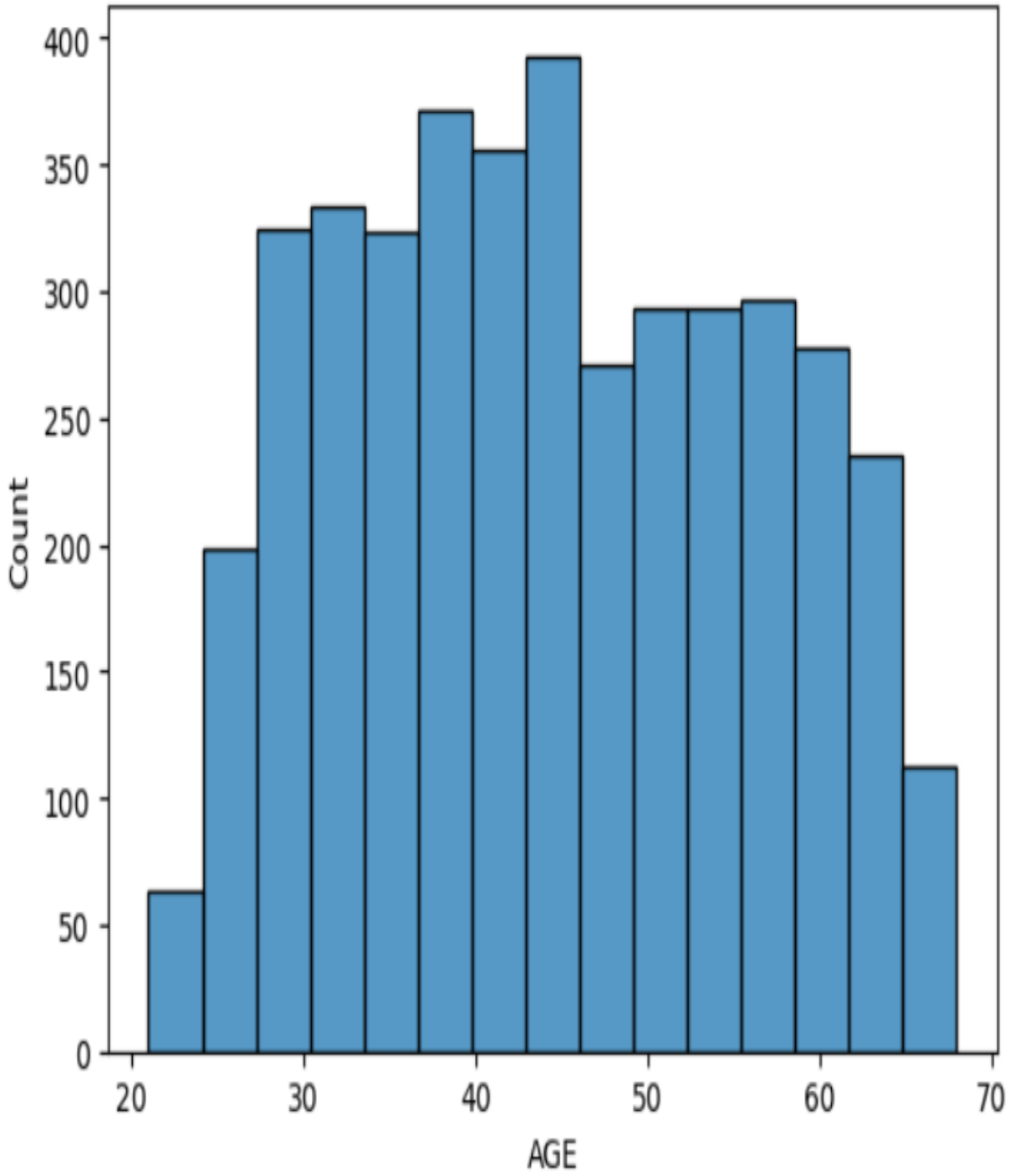
```
Text(0.5, 1.0, 'Percentage of Bad Customer in each group')
```



Age

```
In [70]: sns.histplot(data=df_labeled, x="AGE",bins=15)
```

```
Out[70]: <Axes: xlabel='AGE', ylabel='Count'>
```



Labelling & Finding Percentage of Bad Customer In Each Group

In [71]:
```
def gen_bin_age(df):
    # return group assigned for bin size = binSize
    # assign data > binOver to the last group
    binSize = 5
    binOver = 65
    binMin = 25
    if df <= binMin:
        return str(1)
    elif df <= binOver:
        return str(math.ceil((df-binMin)/binSize))
    else:
        return str(math.ceil((binOver-binMin)/binSize))
```

In [72]:
```
df_labeled["AGE_GROUP"] = df_labeled["AGE"].apply(gen_bin_age)
df_labeled["AGE_GROUP"] = df_labeled["AGE_GROUP"].astype('object')
```

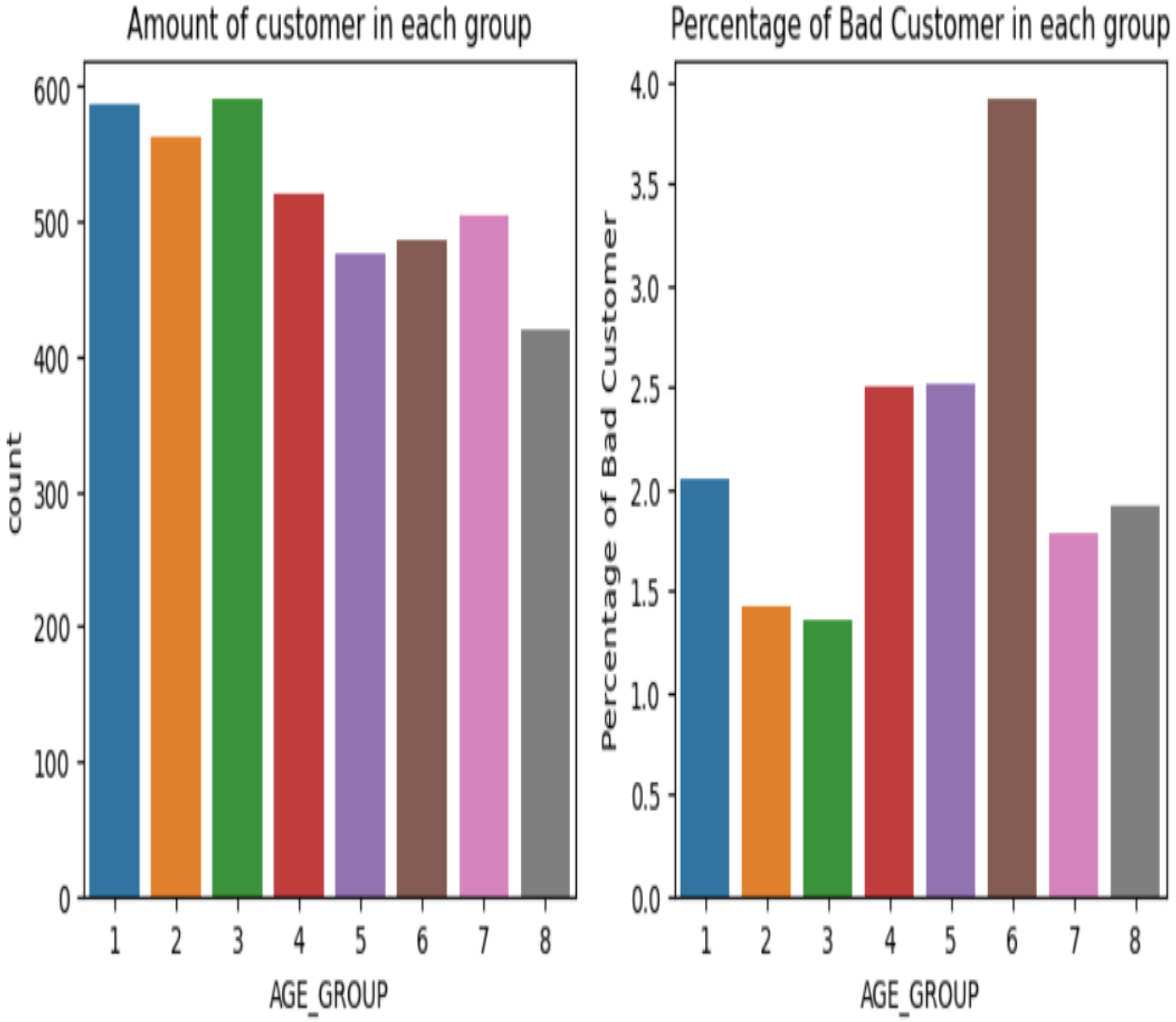
In [73]:
```
df_labeled= df_labeled.drop("AGE",axis=1)
```

In [74]:
```
fig, ax = plt.subplots(1, 2, figsize = (10,4))
RowPlot = "AGE_GROUP"
sns.countplot(data=df_labeled, x=RowPlot,ax=ax[0],order=map(str,range(1,9)))
ax[0].set_title("Amount of customer in each group")

df_fam_percent_bad = df_labeled[[RowPlot,"Bad_Customer"]].groupby(RowPlot,as_index=False).mean()
df_fam_percent_bad["Bad_Customer"] = df_fam_percent_bad["Bad_Customer"]*100
sns.barplot(data=df_fam_percent_bad, x=RowPlot, y="Bad_Customer", ax=ax[1],order=map(str,range(1,9)))
plt.ylabel("Percentage of Bad Customer")
ax[1].set_title("Percentage of Bad Customer in each group")
```

Plot

```
Text(0.5, 1.0, 'Percentage of Bad Customer in each group')
```

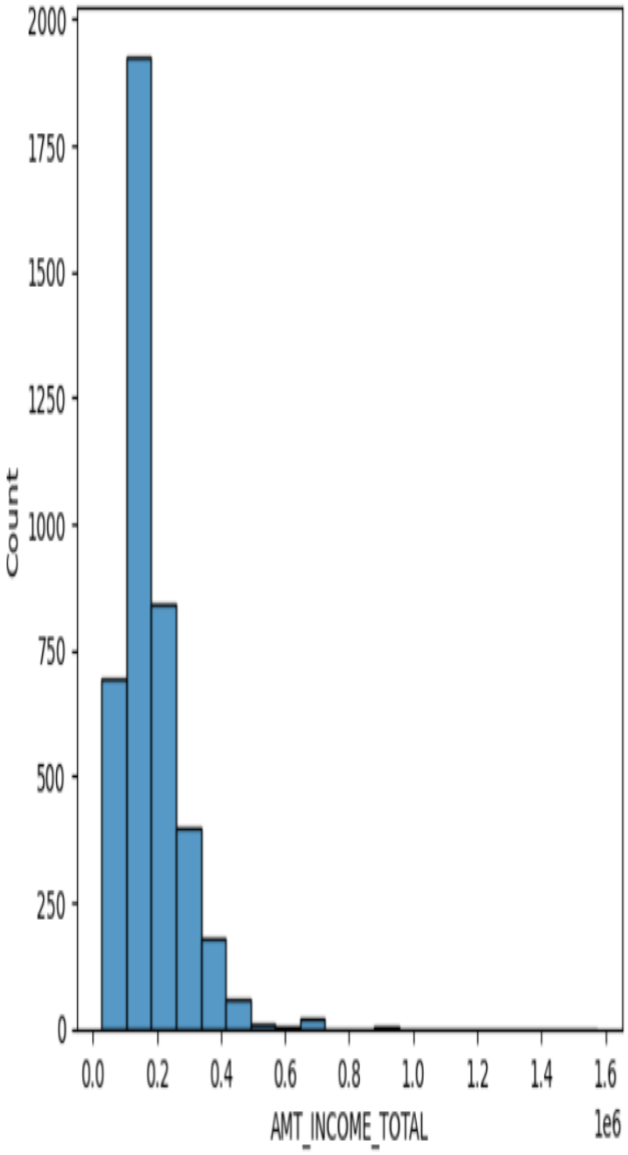


From the graph, customer age between 50-55 (group 6) has high percentage of bad customer.

AMT_INCOME_TOTAL

```
[6]: sns.histplot(data=df_labeled, x="AMT_INCOME_TOTAL",bins=20)
```

```
[6]: <Axes: xlabel='AMT_INCOME_TOTAL', ylabel='Count'>
```



We categorize this into categorical data instead. We grouped customer having income less than 80000 together, and greater than 320000 together. For the customer haveing income between 40000 and 320000 we group them with 40000 d bin spacing.

Labelling & Finding Percentage of Bad Customer In Each Group

```
7]: def gen_bin_income(df):
        # return group assigned for bin size = binSize
        # assign data > binOver to the last group
        binSize =40000
        binOver = 320000
        binMin = 80000
        if df <= binMin:
            return str(1)
        elif df <= binOver:
            return str(math.ceil((df-binMin)/binSize))
        else:
            return str(math.ceil((binOver-binMin)/binSize))
```

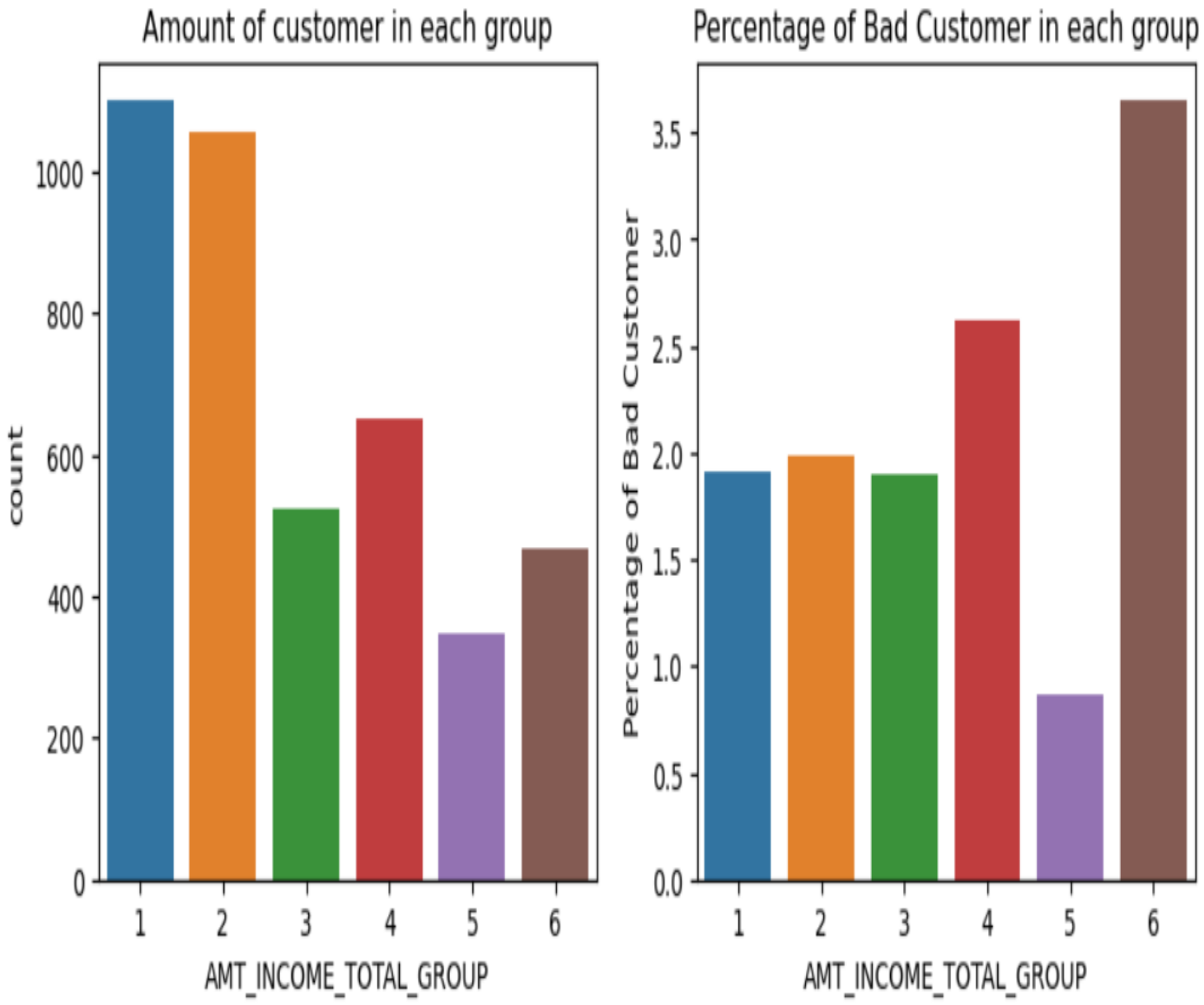
```
3]: df_labeled["AMT_INCOME_TOTAL_GROUP"] = df_labeled["AMT_INCOME_TOTAL"].apply(gen_bin_income)
    df_labeled["AMT_INCOME_TOTAL_GROUP"] = df_labeled["AMT_INCOME_TOTAL_GROUP"].astype('object')
    df_labeled= df_labeled.drop("AMT_INCOME_TOTAL",axis=1)
```

```
9]: fig, ax = plt.subplots(1, 2, figsize = (10,4))
    RowPlot = "AMT_INCOME_TOTAL_GROUP"
    sns.countplot(data=df_labeled, x=RowPlot,ax=ax[0],order=['1','2','3','4','5','6'])
    ax[0].set_title("Amount of customer in each group")

    df_fam_percent_bad = df_labeled[[RowPlot,"Bad_Customer"]].groupby(RowPlot,as_index=False).mean()
    df_fam_percent_bad["Bad_Customer"] = df_fam_percent_bad["Bad_Customer"]*100
    sns.barplot(data=df_fam_percent_bad, x=RowPlot, y="Bad_Customer", ax=ax[1])
    plt.ylabel("Percentage of Bad Customer")
    ax[1].set_title("Percentage of Bad Customer in each group")
```

Plot

```
Text(0.5, 1.0, 'Percentage of Bad Customer in each group')
```



Surprisingly, the group that has highest precentage of bad customer is group 6 or the customer who has income over 320000

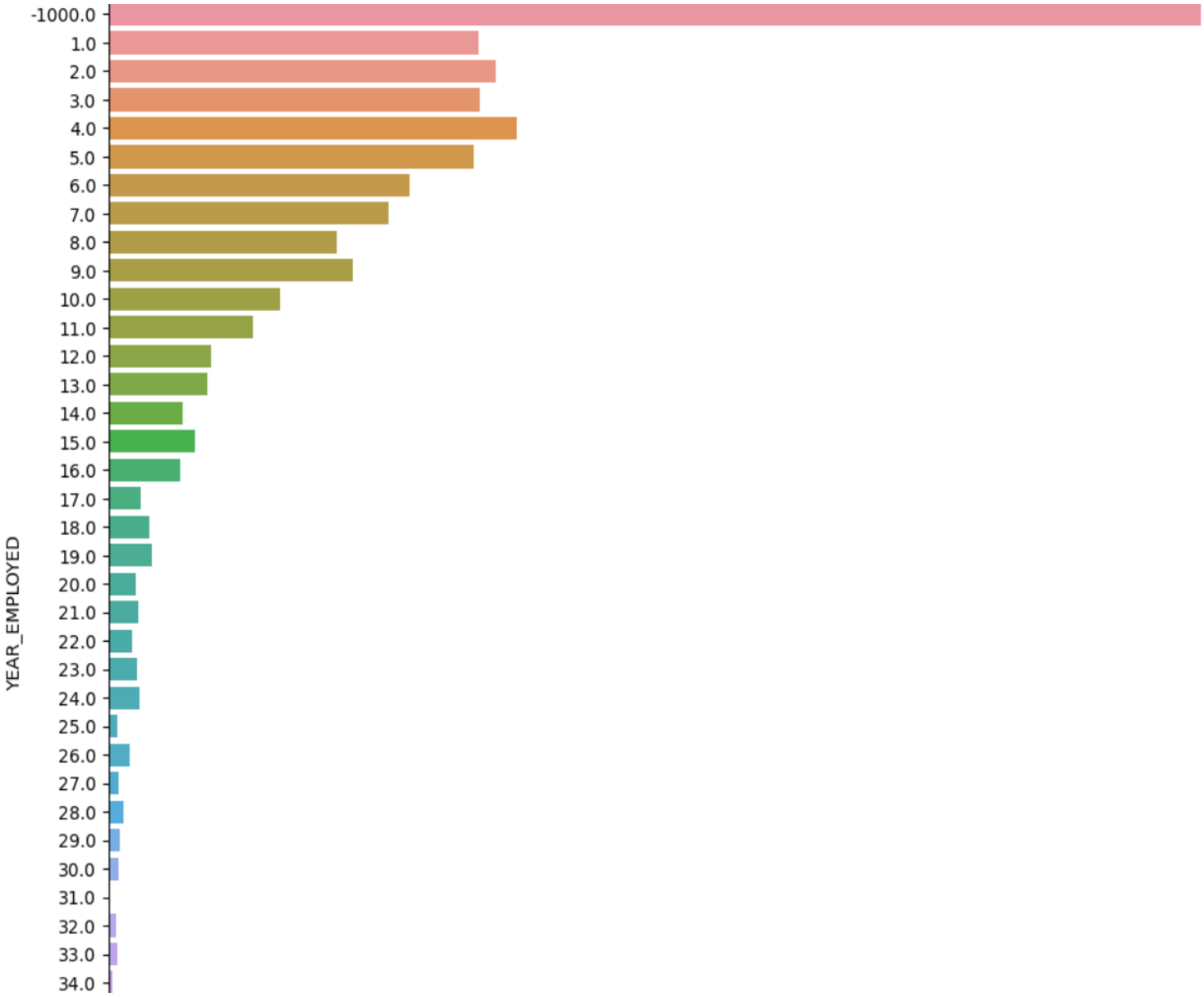
YEAR_EMPLOYED

For year employed, we could not continue with continuous data as year employed for pensioner is undefined. Hence we group year employed to be a categorical data.

```
# Show distribution of the data
fig, ax = plt.subplots(figsize=(12, 12))
sns.countplot(data=df_labeled,y="YEAR_EMPLOYED")
```

Plot

```
<Axes: xlabel='count', ylabel='YEAR_EMPLOYED'>
```



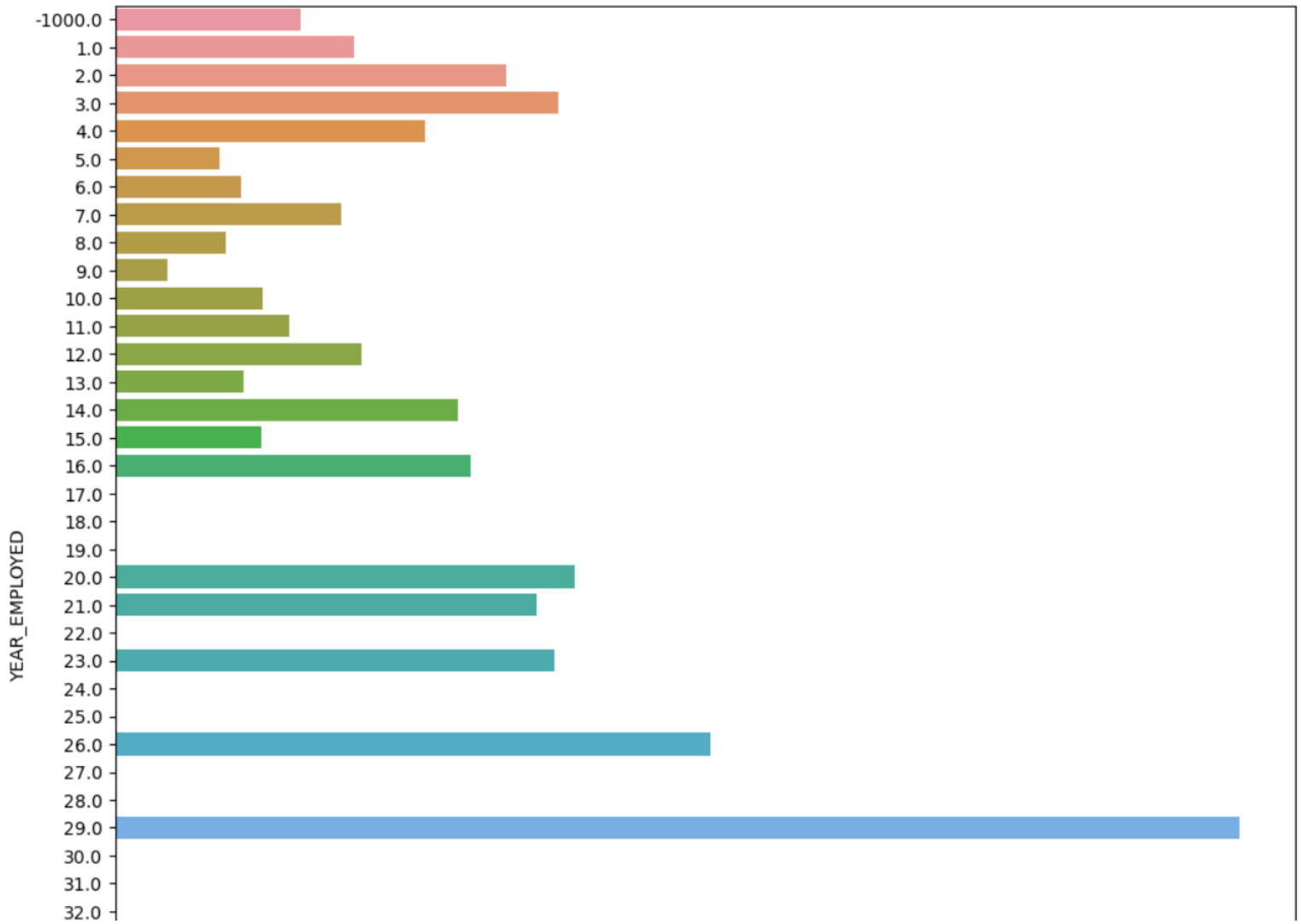
Percentage of bad customer for each year employed

Codes

```
fig, ax = plt.subplots(figsize=(12, 12))
df_year_percent_bad =
df_labeled[["YEAR_EMPLOYED","Bad_Customer"]].groupby(
"YEAR_EMPLOYED",as_index=False).mean()
df_year_percent_bad["Bad_Customer"] =
df_year_percent_bad["Bad_Customer"]*100
sns.barplot(data=df_year_percent_bad,
y="YEAR_EMPLOYED", x="Bad_Customer",orient='h')
```

Plot

```
<Axes: xlabel='Bad_Customer', ylabel='YEAR_EMPLOYED'>
```



We categorize this into categorical data instead. For pensioner we separately grouped them together. We grouped customer have been working greater than 16 years together. For the customers have been working between 2 and 16 we group them with 2 year bin spacing.

Percentage of Bad Customer in each group

In [82]:
```
def gen_bin_year(df):
    # return group assigned for bin size = binSize
    # assign data > binOver to the last group
    binSize = 2
    binOver = 16
    if df == -1000:
        return str(1)
    elif df <= binOver:
        return str(math.ceil(df/binSize) + 1)
    else:
        return str(math.ceil(binOver/binSize) + 1)
```

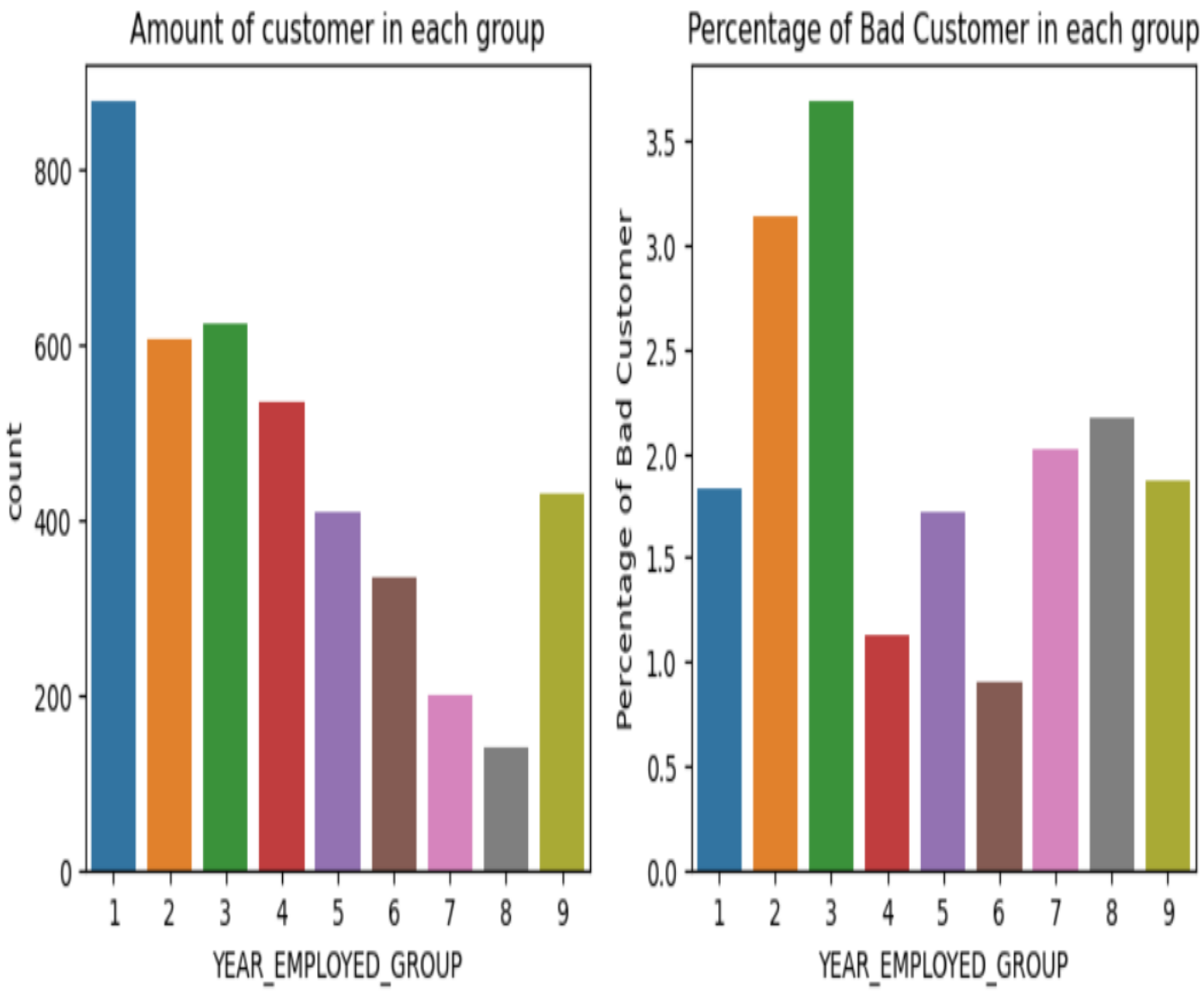
In [83]:
```
df_labeled["YEAR_EMPLOYED_GROUP"] = df_labeled["YEAR_EMPLOYED"].apply(gen_bin_year)
```

In [84]:
```
fig, ax = plt.subplots(1, 2, figsize = (10,4))
RowPlot = "YEAR_EMPLOYED_GROUP"
sns.countplot(data=df_labeled, x=RowPlot,ax=ax[0],order=map(str,range(1,10)))
ax[0].set_title("Amount of customer in each group")

df_fam_percent_bad = df_labeled[[RowPlot,"Bad_Customer"]].groupby(RowPlot,as_index=False).mean()
df_fam_percent_bad["Bad_Customer"] = df_fam_percent_bad["Bad_Customer"]*100
sns.barplot(data=df_fam_percent_bad, x=RowPlot, y="Bad_Customer", ax=ax[1])
plt.ylabel("Percentage of Bad Customer")
ax[1].set_title("Percentage of Bad Customer in each group")
```

Plot

84]:
```
Text(0.5, 1.0, 'Percentage of Bad Customer in each group')
```



It is clear that customer who have been working between 3-5 years (group 2 and 3) have the highest percentage of bad customers.

86]:
```
df_labeled= df_labeled.drop("YEAR_EMPLOYED",axis=1)
df_labeled["YEAR_EMPLOYED_GROUP"] = df_labeled["YEAR_EMPLOYED_GROUP"].astype('object')
```

```
df_labeled.head()
```

	NAME_INCOME_TYPE	NAME_FAMILY_STATUS	OCCUPATION_TYPE	Bad_Customer	FAM_GROUP	AGE_GROUP	AMT_INCOME_TOTAL_GROUP	YEAR_EMPLOYED_GROUP
0	Working	Civil marriage	G5	0	2	3	1	8
1	Pensioner	Married	G4	0	2	6	1	1
2	Working	Married	G5	0	2	3	1	2
3	Pensioner	Married	G4	0	2	8	1	1
4	Pensioner	Single / not married	G4	0	1	7	1	1

```
df_labeled["Bad_Customer"].value_counts(normalize=True)*100
```

```
0    97.851798
1     2.148202
Name: Bad_Customer, dtype: float64
```

Split data into train and test set.

Handle imbalance data by using SMOTE.

Encode data

- **df_LogReg_one = df_labeled**
- **x_LogReg = df_LogReg_one.drop("Bad_Customer",axis=1)**
- **y_LogReg = df_LogReg_one["Bad_Customer"]**
- **x_LogReg_train, x_LogReg_test, y_LogReg_train, y_LogReg_test = train_test_split(x_LogReg, y_LogReg, test_size=0.25, stratify=y_LogReg, random_state=42)**

In [93]:
```
x_LogReg_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3107 entries, 2861 to 1167
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   NAME_INCOME_TYPE        3107 non-null   object
 1   NAME_FAMILY_STATUS      3107 non-null   object
 2   OCCUPATION_TYPE         3107 non-null   object
 3   FAM_GROUP               3107 non-null   object
 4   AGE_GROUP               3107 non-null   object
 5   AMT_INCOME_TOTAL_GROUP  3107 non-null   object
 6   YEAR_EMPLOYED_GROUP     3107 non-null   object
dtypes: object(7)
memory usage: 194.2+ KB
```

In [94]:
```
x_train_balance,y_train_balance = SMOTEN(random_state=10).fit_resample(x_LogReg_train,y_LogReg_train)
x_LogReg_train_encoded = pd.get_dummies(x_train_balance)
x_LogReg_test_encoded = pd.get_dummies(x_LogReg_test)
```

In [95]:
```
x_LogReg_train_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6080 entries, 0 to 6079
Data columns (total 44 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   NAME_INCOME_TYPE_Commercial associate       6080 non-null   uint8
 1   NAME_INCOME_TYPE_Pensioner                  6080 non-null   uint8
 2   NAME_INCOME_TYPE_State servant              6080 non-null   uint8
 3   NAME_INCOME_TYPE_Working                    6080 non-null   uint8
 4   NAME_FAMILY_STATUS_Civil marriage           6080 non-null   uint8
 5   NAME_FAMILY_STATUS_Married                  6080 non-null   uint8
 6   NAME_FAMILY_STATUS_Separated/Widow          6080 non-null   uint8
 7   NAME_FAMILY_STATUS_Single / not married     6080 non-null   uint8
 8   OCCUPATION_TYPE_G1                          6080 non-null   uint8
 9   OCCUPATION_TYPE_G2                          6080 non-null   uint8
 10  OCCUPATION_TYPE_G3                          6080 non-null   uint8
 11  OCCUPATION_TYPE_G4                          6080 non-null   uint8
 12  OCCUPATION_TYPE_G5                          6080 non-null   uint8
 13  OCCUPATION_TYPE_G6                          6080 non-null   uint8
 14  OCCUPATION_TYPE_G7                          6080 non-null   uint8
 15  OCCUPATION_TYPE_G8                          6080 non-null   uint8
 16  FAM_GROUP_1                                 6080 non-null   uint8
 17  FAM_GROUP_2                                 6080 non-null   uint8
 18  FAM_GROUP_3                                 6080 non-null   uint8
 19  FAM_GROUP_4                                 6080 non-null   uint8
 20  FAM_GROUP_5                                 6080 non-null   uint8
 21  AGE_GROUP_1                                 6080 non-null   uint8
 22  AGE_GROUP_2                                 6080 non-null   uint8
 23  AGE_GROUP_3                                 6080 non-null   uint8
 24  AGE_GROUP_4                                 6080 non-null   uint8
 25  AGE_GROUP_5                                 6080 non-null   uint8
 26  AGE_GROUP_6                                 6080 non-null   uint8
 27  AGE_GROUP_7                                 6080 non-null   uint8
 28  AGE_GROUP_8                                 6080 non-null   uint8
 29  AMT_INCOME_TOTAL_GROUP_1                    6080 non-null   uint8
 30  AMT_INCOME_TOTAL_GROUP_2                    6080 non-null   uint8
 31  AMT_INCOME_TOTAL_GROUP_3                    6080 non-null   uint8
 32  AMT_INCOME_TOTAL_GROUP_4                    6080 non-null   uint8
 33  AMT_INCOME_TOTAL_GROUP_5                    6080 non-null   uint8
 34  AMT_INCOME_TOTAL_GROUP_6                    6080 non-null   uint8
 35  YEAR_EMPLOYED_GROUP_1                       6080 non-null   uint8
 36  YEAR_EMPLOYED_GROUP_2                       6080 non-null   uint8
 37  YEAR_EMPLOYED_GROUP_3                       6080 non-null   uint8
 38  YEAR_EMPLOYED_GROUP_4                       6080 non-null   uint8
```

Binary Logistic regression

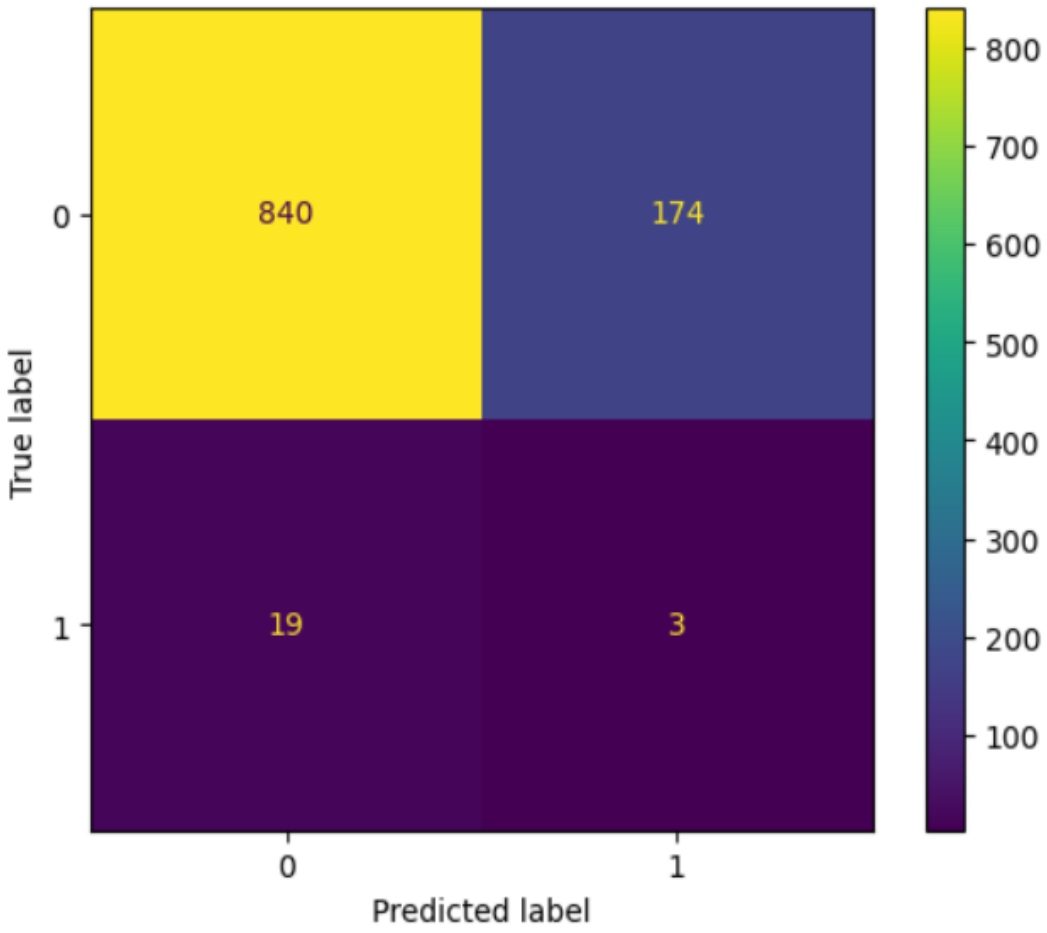
Codes

- log_clf = LogisticRegression(random_state=42, max_iter=2000).fit(x_LogReg_train_encoded, y_train_balance)
- y_predict= log_clf.predict(x_LogReg_test_encoded)
- log_cm = confusion_matrix(y_LogReg_test, y_predict, labels=log_clf.classes_)
- # Create display of confusion matrix
- log_disp = ConfusionMatrixDisplay(confusion_matrix=log_cm, display_labels=log_clf.classes_)

Plot confusion matrix

- log_disp.plot(values_format='')

Out[100]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x21ce4151490>



Display plot

```
plt.show()
print(classification_report(y_LogReg_test,y_predict))
y_pred_log = y_predict.copy()
```

```
              precision    recall  f1-score   support

           0       0.98      0.83      0.90      1014
           1       0.02      0.14      0.03        22

    accuracy                           0.81      1036
   macro avg       0.50      0.48      0.46      1036
weighted avg       0.96      0.81      0.88      1036
```

Random Forest

In [103…

```
cv_params = {'n_estimators' : [50,100,200],
             'max_depth' : [10,20,50],
             'min_samples_leaf':[1,5,10]}
rf = RandomForestClassifier(random_state=0)
rf_val = GridSearchCV(rf, cv_params, refit='f1')
rf_val.fit(x_LogReg_train_encoded, y_train_balance)
```

Out[103]:

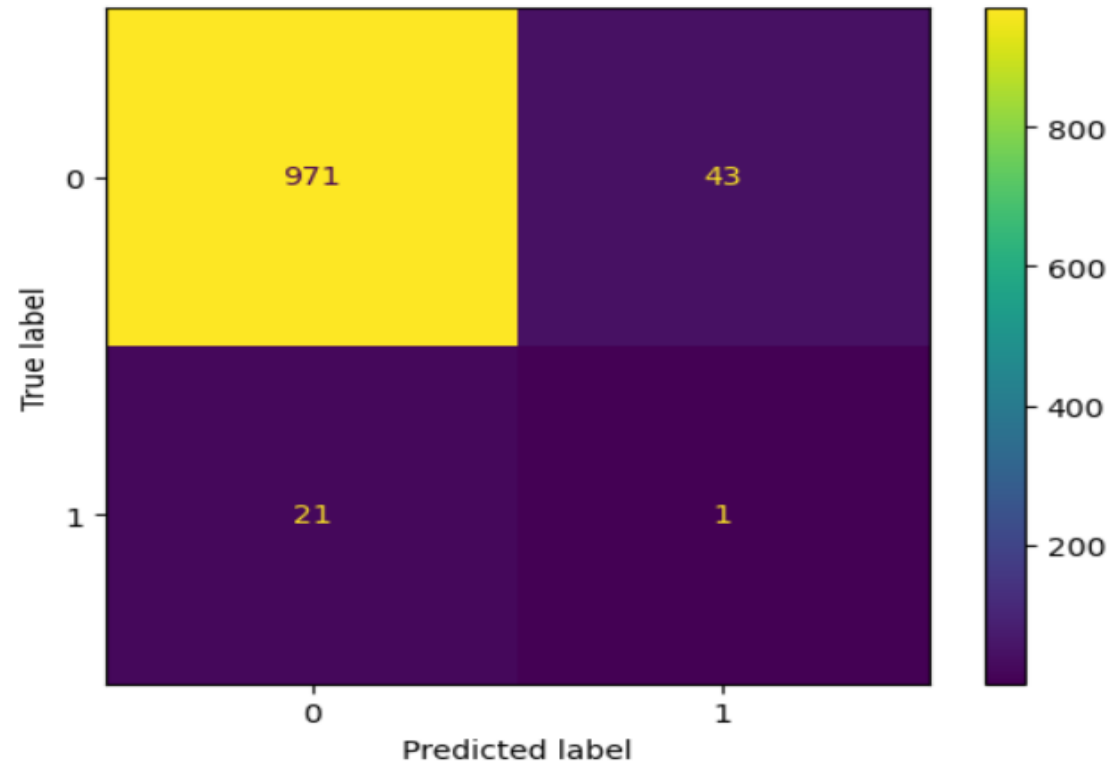
```
GridSearchCV(estimator=RandomForestClassifier(random_state=0),
             param_grid={'max_depth': [10, 20, 50],
                         'min_samples_leaf': [1, 5, 10],
                         'n_estimators': [50, 100, 200]},
             refit='f1')
  estimator: RandomForestClassifier
RandomForestClassifier(random_state=0)
    RandomForestClassifier
RandomForestClassifier(random_state=0)
```

Create display of confusion matrix

- **y_predict = rf_val.predict(x_LogReg_test_encoded)**
- **log_cm = confusion_matrix(y_LogReg_test, y_predict, labels=log_clf.classes_)**
- **log_disp = ConfusionMatrixDisplay(confusion_matrix=log_cm,**
- **display_labels=log_clf.classes_)**

Plot confusion matrix

- **log_disp.plot(values_format='')**
- **plt.show()**
- **print(classification_report(y_LogReg_test,y_predict))**
- **y_pred_rf = y_predict.copy()**



	precision	recall	f1-score	support
0	0.98	0.96	0.97	1014
1	0.02	0.05	0.03	22
accuracy			0.94	1036
macro avg	0.50	0.50	0.50	1036
weighted avg	0.96	0.94	0.95	1036

FINDINGS , RECOMMENDATIONS & CONCLUSION

Findings

Data Limitations and Imbalance

The findings of this project reveal crucial insights into the challenges faced during the development of the Credit Card Approval Prediction System. Two key findings have significantly influenced the project's outcomes.

1. Lack of Highly Relevant Data

One notable finding is the scarcity of highly relevant data, including factors such as debt, mortgage, and expense. These variables are typically pivotal in assessing creditworthiness. However, their absence in the dataset hindered the model's ability to predict bad customers accurately. This limitation emphasizes the importance of comprehensive and detailed data in credit assessment systems.

2. Imbalanced Data

Another critical discovery pertains to data balance. The dataset contained a disproportionately small number of instances representing bad customers, resulting in a highly imbalanced dataset. This skewed distribution adversely affected the model's learning process, causing it to be biased towards predicting good customers. Addressing data imbalance is essential for improving the model's performance and decision-making accuracy.

Recommendations

Enhancing Data Quality and Quantity

Based on the findings, several recommendations are proposed to enhance the Credit Card Approval Prediction System:

1. Data Enrichment: Efforts should be made to acquire additional data variables, particularly those related to an applicant's financial situation, such as debt, mortgage obligations, and monthly expenses. This enriched dataset would provide a more holistic view of an applicant's financial health.

2. Data Imbalance Mitigation: Strategies to address data imbalance are vital. Techniques such as oversampling minority class instances or using advanced algorithms like Synthetic Minority Over-sampling Technique (SMOTE) should be explored to create a more balanced training dataset.

3. External Data Sources: Consider incorporating external data sources, such as credit bureau data, to augment the current dataset. These sources can provide comprehensive financial histories and improve prediction accuracy.

4. Feature Engineering: Explore feature engineering techniques to create new relevant features that may compensate for missing data.

Conclusions

The models developed failed to predict bad customers. Though the process can be finely tuned, the performance of the model is not increasing significantly. The assumption we made are

1) The lack of highly relevant data,
e.g. debt, mortgage, and expense.

2) There is small example of bad customer (the data is highly imbalance).

Balancing Risk Assessment

In conclusion, this project highlights the challenges and limitations encountered when developing a Credit Card Approval Prediction System. Despite rigorous efforts, the models struggled to accurately predict bad customers, primarily due to the absence of highly relevant data and data imbalance.

This research serves as a reminder of the complexities involved in credit risk assessment and the importance of data quality and balance. While significant strides have been made, further refinement and expansion of the system are necessary to balance risk assessment and enhance decision-making.

The Path Forward

This project, though faced with limitations, offers valuable insights into credit risk modeling. Future endeavors should focus on the recommendations outlined to enhance data quality, quantity, and balance. By addressing these issues, financial institutions can significantly improve their credit card approval processes, reduce risk exposure, and make more informed lending decisions.

In essence, while this project encountered hurdles, it paves the way for ongoing research and innovation in credit risk assessment. It underscores the significance of evolving data science methodologies to meet the dynamic challenges of the financial industry.

Appendix

Datasets and Codes(html) are Submitted

Bibliography

Dataset Source :

https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction