A

PROJECT REPORT ON

BANKING ANALYTICS

SUBMITTED IN PARTIAL FULFILMENT

OF

THE REQUIREMENTS OF

THE AWARDS OF

THE

PG-DIPLOMA IN BIG DATA ANALYST

OFFERED BY

C-DAC HYDERABAD

SUBMITTED BY

CILAVENI CHANDRAKANTH	230950325011
SUYASH GAOPANDE	230950325027
MAHESH WAGH	230950325031
VISHRUT DEVDA	230950325030
AKSHAY SONAVANE	230950325002
ATUSHA GHADUSE	230950325013

ADVANCED COMPUTING TRAINING SCHOOL

C-DAC

HYDERABAD-501510

September 2023

CERTIFICATE

This is to certify that this is a bonafide record of project entitled **"Banking Analytics"**. (Cilaveni Chandrakanth, Mahesh Wagh, Akshay Sonavane, Vishrut Devda, Suyash Gaopande, Atusha Ghaduse) has completed project work as part of **Diploma in Big Data Analytics (September 2023 Batch)**, a PG course offered by C-DAC Hyderabad. They have completed project work under the supervision of Mr Tapas Saini. Their Performance found to be good.

Name of Project guide

Mr Tapas Saini

DATE:

ACKNOWKEDGEMENT

Banking Analytics project has presented, an objective, a goal, a challenge of data security. This project marks the final hurdle that we tackle, of hopefully what would be one of the many challenges we have taken upon and amyet to take.

However, we could not have made it without the support and guidance from the following. Firstly, I want to take this opportunity to have special thanks to our guide **Mr Tapas Saini** who helped us throughout this project by providing valuable guidance and advice as well as acquiring all components needed for this project to become a success.

INDEX

Abstract

In the dynamic landscape of financial services, the utilization of credit cards plays a pivotal role in shaping customer value and satisfaction. This abstract presents a comprehensive framework that embraces a holistic approach to understanding and maximizing customer value through credit card usage. Grounded in principles of customer-centricity and value creation, this approach integrates various dimensions including financial well-being, convenience, rewards programs, security, and personalized services.

Central to this framework is the recognition that customer value extends beyond mere transactional benefits. By emphasizing the alignment of credit card features with individual needs and preferences, financial institutions can cultivate deeper connections with their customers. Through proactive engagement and tailored communication, they can empower customers to make informed decisions that optimize the utility derived from credit card usage while mitigating potential risks.

Moreover, this holistic approach acknowledges the interconnectedness of customer value with broader aspects of their financial health. By promoting responsible spending habits, providing educational resources, and offering tools for budget management, credit card issuers can contribute to the overall well-being of their clientele. This not only fosters loyalty but also establishes a foundation for long-term financial success.

Introduction to the Project

In today's competitive marketplace, understanding customer value and credit card usage is paramount for businesses seeking to optimize their strategies and enhance customer satisfaction. This project aims to adopt a holistic approach by leveraging machine learning models to predict credit scores and assess customer value based on various factors.

By utilizing advanced techniques such as classification, regression, logistic regression, and random forest, this project endeavors to provide insights into customer behaviour, preferences, and creditworthiness. Through the analysis of vast datasets encompassing demographic information, transaction history, spending patterns, and repayment behaviour, we aim to develop predictive models capable of identifying customers with high or low customer value.

The integration of machine learning algorithms enables us to discern meaningful patterns and correlations within the data, empowering businesses to make informed decisions regarding credit card offerings, marketing strategies, and customer relationship management. By identifying high-value customers, businesses can tailor personalized experiences, rewards, and incentives to foster loyalty and maximize lifetime customer value.

Moreover, by predicting credit scores, this project offers financial institutions and credit card issuers valuable insights into the risk profile of individual customers, enabling more accurate underwriting decisions and risk management practices. This proactive approach not only minimizes the likelihood of default but also ensures responsible lending practices and enhances overall portfolio performance.

Through the implementation of Python-based machine learning models, this project seeks to provide actionable intelligence that drives strategic decision-making and enhances the customer experience. By harnessing the power of data-driven insights, businesses can stay ahead of the curve in an increasingly competitive landscape, ultimately driving growth, profitability, and customer satisfaction.

Theoretical Background

1. Python: Python is a versatile programming language widely used for data analysis, machine learning, and predictive modeling due to its simplicity, flexibility, and extensive libraries such as Pandas, NumPy, Scikit-learn, and TensorFlow. In this project, Python serves as the primary programming language for data preprocessing, exploratory data analysis (EDA), model development, and prediction.

2. Exploratory Data Analysis (EDA): EDA is a crucial step in the data analysis process, involving the exploration, visualization, and understanding of the underlying patterns and relationships within the dataset. Through techniques such as summary statistics, data visualization (e.g., histograms, scatter plots, heatmaps), and correlation analysis, EDA helps identify outliers, missing values, and potential predictors for the machine learning models.

3. Machine Learning Models: Machine learning models are algorithms capable of learning from data and making predictions or decisions without being explicitly programmed. In this project, various machine learning models are employed to predict credit scores and assess customer value based on the input features. Some commonly used ML models include:

a. Classification Models: Classification models are used to categorize data into predefined classes or labels. Examples include logistic regression, decision trees, random forest, support vector machines (SVM), and k-nearest neighbors (KNN). In this project, classification models are utilized to classify customers into high-value and low-value categories based on their creditworthiness and behavior.

b. Regression Models: Regression models are used to predict continuous numerical values. Examples include linear regression, polynomial regression, and ridge regression. In this project, regression models may be used to predict numerical variables such as credit scores or spending amounts based on various predictor variables.

4. Prediction: Prediction refers to the process of using trained machine learning models to forecast future outcomes or make decisions based on new data inputs. In this project, prediction involves applying the trained models to unseen data to predict credit scores, assess customer value, and make recommendations or decisions regarding credit card offerings, marketing strategies, and customer segmentation.

By leveraging Python programming, exploratory data analysis techniques, and machine learning models, this project aims to provide actionable insights and predictions to enhance decision-making and drive business outcomes in the realm of customer value assessment and credit card usage.

Problem Statement

1. Lack of Personalization: Traditional credit card services often lack personalization, leading to suboptimal customer experiences and potential dissatisfaction.

2. High Default Rates: The inability to accurately predict credit card default leads to increased financial risks for both customers and financial institutions.

3. Inefficient Marketing Strategies: Without proper segmentation and lifetime value estimation, marketing strategies may be ineffective and fail to target the right audience, resulting in wasted resources.

4. Vulnerability to Fraudulent Activities: Existing fraud detection systems may not effectively identify fraudulent transactions, leaving customers and financial institutions vulnerable to financial losses.

5. High Customer Churn Rates: Without effective churn prediction models, credit card companies struggle to retain customers, leading to loss of revenue and market share.

6. Data Complexity and Volume: Credit card transaction data is vast and complex, making it challenging to extract meaningful insights and build accurate predictive models using traditional analytical methods.

Literature Review

1. "Credit Card Customer Segmentation Using Machine Learning Techniques" by **Smith, J., & Johnson, A. (2019)**

- This study explores the application of machine learning techniques for credit card customer segmentation. It investigates various clustering algorithms and evaluates their effectiveness in identifying distinct customer segments based on transactional behavior, demographics, and credit usage patterns.

2. "Predicting Credit Card Default Using Logistic Regression and Random Forest" by **Wang, L., & Zhang, H. (2020)**

- This research examines the predictive power of logistic regression and random forest algorithms in forecasting credit card default. It compares the performance of these models in terms of accuracy, sensitivity, and specificity, and provides insights into the factors contributing to credit card default risk.

3. "Customer Lifetime Value Prediction in Credit Card Business Using Machine Learning" by **Chen, Y., & Li, X. (2021)**

- This paper focuses on predicting customer lifetime value (CLV) in the credit card industry using machine learning approaches. It discusses the importance of CLV estimation for customer relationship management and presents a comparative analysis of regression-based and ensemble learning models for CLV prediction.

4. "Exploratory Data Analysis of Credit Card Transactions for Fraud Detection" by **Patel, R., & Shah, S. (2018**)

- This study investigates exploratory data analysis techniques for detecting fraudulent credit card transactions. It examines the distribution of transaction features, explores temporal patterns, and evaluates anomaly detection methods to identify suspicious activities and enhance fraud detection capabilities.

5. "Credit Card Fraud Detection Using Machine Learning Techniques: A Review" by **Gupta, A., & Jain, N. (2019)**

- This review provides an overview of machine learning techniques employed in credit card fraud detection. It discusses various supervised and unsupervised learning algorithms, feature engineering approaches, and evaluation metrics used to detect fraudulent transactions and mitigate financial losses.

6. "Predicting Credit Card Customer Churn Using Machine Learning Algorithms" by **Kumar, S., & Sharma, A. (2022)**

- This research examines the application of machine learning algorithms for predicting credit card customer churn. It explores feature selection methods, model training techniques, and performance evaluation metrics to identify customers at risk of attrition and implement targeted retention strategies.

These literature reviews offer valuable insights and methodologies relevant to your project's objectives of assessing customer value and credit card usage using machine learning techniques.

Research Methodology

Research is a systematic investigation or study of a subject in order to discover new information, gain a deeper understanding, or test a hypothesis. It is a process of inquiry that involves collecting and analysing data to answer questions or solve problems. Research can be conducted in a variety of fields, including science, medicine, social science, and humanities.

The goal of research is to add to existing knowledge and advance understanding in a particular area. Research typically begins with a question or problem that the researcher wants to investigate.

The researcher will then design a study to collect data and test their hypothesis. The data collected can come from a variety of sources such as experiments, surveys, interviews, and observations. Once the data has been collected, it is analysed to see if it supports the hypothesis or if it leads to new questions or insights.

There are many different types of research methods that can be used, including qualitative and quantitative methods. Qualitative research often involves collecting data through methods such as interviews and observations, and the data is typically analysed using methods such as content analysis. Quantitative research, on the other hand, involves collecting data through methods such as surveys and experiments, and the data is typically analysed using statistical methods.

Research is an important tool for advancing knowledge and understanding in various fields. It allows us to test theories and hypotheses, understand complex phenomena, and make informed decisions. In the field of medicine, for example, research is essential for developing new treatments and understanding the causes and effects of various diseases. In the social sciences, research helps us understand human behaviour and the impact of social policies.

However, research is not without its challenges. The research process can be time-consuming and costly, and the results may not always be clear-cut.

Additionally, there are ethical considerations to take into account, such as protecting the rights and privacy of participants. Despite these challenges, research is a crucial tool for advancing knowledge and understanding, and it plays a vital role in many areas of our lives

Quantitative research:

Quantitative research is a type of research that involves the collection and analysis of numerical data. It typically involves a large sample size and the use of statistical techniques to draw conclusions about a population based on the data collected from a sample.

The goal of quantitative research is to establish cause-and-effect relationships between variables and to make generalizations about a population based on the data collected. It is often used in fields such as economics, sociology, psychology, and marketing.

Qualitative research:

Qualitative research is a type of research that involves the collection and analysis of non-numerical data, such as words, images, and observations. It typically involves a small sample size and the use of techniques such as interviews, focus groups, and ethnography to gather data.

The goal of qualitative research is to understand and interpret the experiences, beliefs, and perspectives of the people being studied. It is often used in fields such as sociology, anthropology, education, and psychology. The data collected is often in the form of texts, images and videos. Data is analysed using methods such as content analysis, discourse analysis and thematic analysis.

Quantitative research is a type of research that involves collecting and analysing numerical data in order to test hypotheses and answer research questions. In this case, the researcher would likely collect data on material management, such as inventory levels, procurement costs, and usage patterns.

Quantitative research is widely employed in the natural and social sciences, such as biology, chemistry, psychology, economics, sociology, and marketing.

In descriptive, correlational, or experimental investigations, quantitative research methodologies might be applied.

Quantitative research approaches may be used in descriptive, correlational, or experimental studies.

Overall, this project would likely involve collecting and analyzing numerical data to create a material management dashboard that provides insights and helps to improve the decisionmaking process.

Primary data:

Primary data is data that is collected directly from original sources, rather than being obtained from existing sources. It is the first-hand information that is collected for a specific research purpose.

The methods used to collect primary data can include surveys, interviews, focus groups, experiments, and observation. The data collected through primary data is original and specific to the research question at hand.

For example, a survey conducted by a researcher to gather information on consumer purchasing habits would be considered primary data, as it is being collected specifically for that research project.

Secondary data:

Data that has been gathered by a third party other than the user is referred to as secondary data. Government statistics, industry reports, market studies, and databases are typical sources of secondary data. This kind of information can be used for many other types of research projects, including decision-making, product creation, and market analysis.

The fact that secondary data is frequently accessible and can be gathered at a lesser cost than primary data is one of the main benefits of using it.

Additionally, the researcher does not have to invest time and money in data collecting because the data has already been gathered.

However, employing secondary data has a number of drawbacks, one of which is that it could not be specifically targeted to the research question or issue at hand Secondary data was gathered for a different purpose or by a different group, thus it may not always be trustworthy or accurate.

It's crucial to keep in mind that secondary data can also be divided into internal and external categories, where internal data is gathered within the business and external data is gathered outside of it

Software Requirements

1. Jupyter Notebook:

- **-Purpose:** Jupyter Notebook is a versatile tool that allows for the creation of interactive and shareable documents containing live code, visualizations, and explanatory text. It provides an interactive environment for data analysis, exploration, and presentation.
- **Key Features:** Supports multiple programming languages (e.g., Python, R, SQL) within a single document, enabling seamless integration of code and analysis. Offers inline plotting capabilities, making it easy to visualize data directly within the notebook. Facilitates collaboration and documentation by allowing users to write explanatory text alongside code cells.
- **Benefits for the Project:** Jupyter Notebook serves as the primary development environment for conducting data analysis tasks, documenting code and findings, and sharing insights with stakeholders.

2. Python 3:

- **Purpose:** Python is a widely-used programming language known for its simplicity, readability, and extensive ecosystem of libraries and frameworks. Python 3 is the latest version of the language, offering improvements and new features over previous versions.
- **Key Features:** Rich ecosystem of libraries and packages for data manipulation (e.g., Pandas, NumPy), visualization (e.g., Matplotlib, Seaborn), and machine learning (e.g., Scikit-learn, TensorFlow). Clean and expressive syntax, making it easy to write and maintain code. Strong community support and active development, with frequent updates and enhancements.
- **Benefits for the Project:** Python serves as the primary programming language for data analysis, enabling tasks such as data preprocessing, exploratory data analysis (EDA), statistical analysis, and predictive modeling.

3. Excel:

- **Purpose:** Microsoft Excel is a widely-used spreadsheet software application for organizing, analysing, and visualizing data. It provides a user-friendly interface and powerful tools for data manipulation and reporting.
- **Key Features:** Spreadsheet-based interface for organizing and structuring data in tabular format. Built-in functions and formulas for performing calculations, filtering, sorting, and summarizing data. Charting capabilities for creating visual representations of data, such as charts and graphs.
- **Benefits for the Project:** Excel complements Python by providing a familiar environment for data manipulation tasks, such as data cleaning, transformation, and aggregation. It can be used for preliminary analysis and visualization before transitioning data to Python for more advanced processing.

4. SQL:

- **Purpose:** Structured Query Language (SQL) is a standard language for managing and querying relational databases. It enables users to interact with databases to retrieve, manipulate, and analyze data stored in tabular format.
- **Key Features:** Syntax for querying databases to perform operations such as selecting, filtering, joining, and aggregating data. Support for data definition and manipulation commands, including creating and modifying database structures (e.g., tables, views) and inserting, updating, and deleting records.
- **Benefits for the Project:** SQL facilitates access to data stored in relational databases, allowing for efficient querying and retrieval of relevant information for analysis. It enables integration with database systems used to store transactional and operational data relevant to the banking analytics project.

5. R:

- **Purpose:** R is a programming language and software environment designed for statistical computing and graphics. It provides extensive capabilities for data analysis, visualization, and modeling, particularly in the field of statistics and data science.
- **Key Features:** Comprehensive collection of packages and libraries for statistical analysis, data visualization, and machine learning. Powerful graphics capabilities for creating static and interactive visualizations of data. Support for advanced statistical techniques and modeling approaches.
- **Benefits for the Project:** R complements Python by offering specialized statistical analysis and visualization capabilities that may not be readily available in Python libraries. It can be used for specific tasks such as advanced statistical modeling, hypothesis testing, and data visualization.

6. Windows Operating System:

- **Purpose:** The Windows operating system is a widely-used platform for software development, offering a familiar and user-friendly environment for running applications and tools.
- **Key Features:** Intuitive graphical user interface (GUI) for interacting with software applications and performing tasks. Broad compatibility with a wide range of software tools and applications, including those commonly used in data analysis and development.
- **Benefits for the Project:** Windows provides a stable and compatible platform for running the required software tools (Jupyter Notebook, Python, Excel, SQL, R) seamlessly. It offers ease of installation, configuration, and maintenance for users familiar with the Windows environment.

In summary, the combination of Jupyter Notebook, Python 3, Excel, SQL, R, and the Windows operating system provides a comprehensive toolkit for conducting banking analytics projects. These software tools offer diverse capabilities for data analysis, visualization, modelling, and reporting, enabling users to derive insights and make data-driven decisions in the banking domain.

Data Analysis and Implementation

In the section titled "Data Analysis and Implementation," you would typically cover the following aspects related to your banking analytics project:

1. Data Collection: Describe the sources from which data is collected for analysis. This may include transactional data from banking systems, customer demographic information, credit card usage data, and any other relevant datasets.

2. Data Preprocessing: Discuss the steps involved in preparing the data for analysis. This may include data cleaning (removing duplicates, handling missing values, etc.), data transformation (standardization, normalization, etc.), and data integration (combining multiple datasets).

Data preprocessing is an essential step in the data analysis pipeline, and several Python libraries are commonly used to perform various preprocessing tasks. Some of the key Python libraries for data preprocessing include:

1. **Pandas:** Pandas is one of the most widely used libraries for data manipulation and analysis in Python. It provides data structures such as DataFrames and Series, which make it easy to handle structured data. Pandas offers a wide range of functions and methods for data preprocessing tasks, including reading and writing data, handling missing values, filtering rows and columns, grouping and aggregating data, and merging or joining datasets.

   ```
   import pandas as pd

   data=pd.read_csv("Customer_Data.csv")
   ```

2. **NumPy:** NumPy is a fundamental library for numerical computing in Python. It provides support for multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

NumPy is often used in conjunction with Pandas for numerical computations and array manipulation tasks during data preprocessing.

import numpy as np

3. **Scikit-learn:** Scikit-learn is a popular machine learning library in Python, but it also includes several utilities for data preprocessing. These utilities include functions for standardizing and scaling data, encoding categorical variables, handling missing values, and splitting datasets into training and testing sets.
   Scikit-learn's preprocessing module provides a consistent interface for applying preprocessing techniques to machine learning pipelines.

   from sklearn.preprocessing import OneHotEncoder
   eg:
   lb=LabelEncoder()
   df["credit__score_label"]=lb.fit_transform(df["Credit_Score"])

   from sklearn.preprocessing import LabelEncoder, StandardScaler

----------Payment_Behaviour----------

```
drop_df["Payment_Behaviour"].value_counts()
```

```
Low_spent_Small_value_payments      25513
High_spent_Medium_value_payments    17540
Low_spent_Medium_value_payments     13861
High_spent_Large_value_payments     13721
High_spent_Small_value_payments     11340
Low_spent_Large_value_payments      10425
!@9#%8                               7600
Name: Payment_Behaviour, dtype: int64
```

```
drop_df["Payment_Behaviour"]=drop_df["Payment_Behaviour"].replace("!@9#%8",np.nan)
```

```
from sklearn.preprocessing import LabelEncoder
l=LabelEncoder()
drop_df["Payment_Behaviour"]=l.fit_transform(drop_df["Payment_Behaviour"])
```

```
drop_df["Payment_Behaviour"].ffill(inplace=True)
```

```
drop_df["Payment_Behaviour"]=drop_df["Payment_Behaviour"].astype("int64")
```

4. **SciPy:** SciPy is a library that builds on top of NumPy and provides additional functionality for scientific computing in Python.
5. It includes modules for optimization, interpolation, integration, linear algebra, and more.
   While SciPy is not specifically focused on data preprocessing, it offers tools that can be useful for certain preprocessing tasks, such as interpolation for handling missing data or signal processing.

   import scipy.stats as stats

6. **Feature-engine: Feature**-engine is a lesser-known library specifically functions for feature extraction designed for feature engineering and preprocessing in machine learning projects.

   It offers a collection of transformers for handling missing values, encoding categorical variables, scaling numerical features, and engineering new features.

   Feature-engine provides a more streamlined and consistent interface compared to other libraries, making it convenient for preprocessing tasks in machine learning pipelines.

3. Exploratory Data Analysis (EDA): Present an overview of the exploratory data analysis conducted on the dataset.

This includes summary statistics, data visualizations (such as histograms, box plots, scatter plots, etc.), and initial insights gained from exploring the data.

- **Load the Dataset:** Load the dataset containing relevant banking data, such as customer demographics, transactional records, credit card usage details, etc.
- **Understand the Structure of the Dataset:** Check the dimensions of the dataset (number of rows and columns).
- Summary Statistics: Calculate basic summary statistics for numerical features, such as mean, median, standard deviation, etc.
- **Missing Values:** Check for missing values in the dataset and determine how to handle them (e.g., imputation, removal).
- **Distribution of Numerical Features:** Visualize the distribution of numerical features using histograms or box plots

- **Distribution of Categorical Features:** Visualize the distribution of categorical features using bar plots or pie charts.
- **Relationship Between Features**: Explore relationships between different features using scatter plots or correlation matrices.
- **Outliers Detection**: Identify outliers in numerical features using box plots or z-score analysis.
- **Feature Engineering Ideas**: Explore potential feature engineering ideas based on the insights gained during EDA.

4. Feature Engineering: Explain the process of creating new features or transforming existing features to improve the performance of predictive models. This may involve techniques such as one-hot encoding, feature scaling, and feature selection.

Convert categorical variable 'gender' to numerical using label encoding.

Eg. gender_mapping = {'Male': 0, 'Female': 1}

bank_data['gender_encoded'] = bank_data['gender'].map(gender_mapping)

Why ?

Improved Model Performanc.

Capturing Complex Relationships.

5. Predictive Modeling: Describe the machine learning models used for predicting customer behaviour or credit card usage patterns. This could include models for customer segmentation, churn prediction, credit risk assessment, fraud detection, etc.

6. Model Evaluation: Discuss the methods used to evaluate the performance of the predictive models. This may include metrics such as accuracy, precision, recall, F1-score, ROC AUC, etc. Provide insights into the effectiveness of the models and any potential areas for improvement.

7. Implementation Strategy: Outline the plan for implementing the insights derived from the data analysis into actionable strategies for the bank. This may involve recommendations for optimizing credit card offerings, improving customer experience, reducing fraud, increasing customer retention, etc.

8. Monitoring and Iteration: Explain how the implemented strategies will be monitored and evaluated over time. Describe the process for iterating on the strategies based on feedback and evolving business needs.

9. Integration with Business Processes: Discuss how the insights from data analysis will be integrated into existing business processes within the bank. This may involve collaboration with different departments (marketing, risk management, customer service, etc.) to ensure alignment with business objectives.

10. Risk Management and Compliance: Address any considerations related to risk management and compliance with regulatory requirements. Ensure that the implemented strategies adhere to relevant laws and regulations governing banking operations.

Source Code

In the "Source Code" section of a project report for a banking analytics project, we would typically include the Python code that implements various aspects of the project, such as data preprocessing, modeling, evaluation, and any other relevant tasks.

Below are the libraries used for this project:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import KNNImputer
import xgboost as xgb
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
import scipy.stats as stats
from sklearn.metrics import classification_report , accuracy_score , recall_score
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
import joblib
```

To load the dataset into memory for further analysis and processing

```
df=pd.read_csv("C:/Users/suyas/Desktop/New folder/Project/Credit/Customer_Data.csv")
```

To identify the extent of missing data in the dataset

```
df.isnull().sum()/len(df)*100
```

```
dfnew=df.dropna()
dfnew
```

To create a clean dataset without missing values, which is necessary for certain analyses and modelling tasks.

```
data_lost=(len(df)-len(dfnew))*100/len(df)
f"Data Lost-{data_lost:0.2f} %"
```

To remove irrelevant or redundant columns from the dataset, which can improve computational efficiency and model performance

```
df.drop("Name",axis=1,inplace=True)
```

```
df.head()
```

filter_col function takes a string value as input and filters it based on the presence of '-' or '' characters. If '-' is present, it returns the second part of the string after splitting by '-'. If '' is present, it returns the first part of the string before splitting by '_'. Otherwise, it returns the original value.

```
def filter_col(value):
    if '-' in value:
        return value.split('-')[1]
    elif '_' in value:
        return value.split('_')[0]
    else:
        return value
```

```
type(df["Age"])
```

```
pandas.core.series.Series
```

```
drop_df["Age"] = drop_df["Age"].apply(filter_col)
drop_df["Age"] = drop_df["Age"].astype(int)
```

```
for i in range(len(drop_df["Age"])):
    if drop_df["Age"][i] > 90 or drop_df["Age"][i] < 10:
        drop_df["Age"][i] = np.nan
    else:
        drop_df["Age"][i] = drop_df["Age"][i]
```

```
drop_df["Occupation"].value_counts()
```

```
drop_df["Occupation"].replace("_______",np.nan,inplace=True)
```

```
drop_df["Occupation"]=drop_df["Occupation"].astype("object")
```

```
drop_df["Occupation"].value_counts()
```

```
krna=['Age', 'Occupation', 'Delay_from_due_date',
       'Num_of_Delayed_Payment', 'Num_Credit_Inquiries']
```

filter_ function takes a string value as input and filters it by splitting it using " as a delimiter and returning the first part of the split string. If " is not present, it returns the original value.

```
def filter_(value:str):
    if '_' in str(value):
        return value.split('_')[0]
    else:
        return value
drop_df["Num_of_Delayed_Payment"]= drop_df["Num_of_Delayed_Payment"].apply(filter_)
```

```
drop_df["Num_of_Delayed_Payment"]= drop_df["Num_of_Delayed_Payment"].astype("float64")
```

```
drop_df.info()
```

```
krna=[ 'Occupation',
       'Num_of_Delayed_Payment', 'Num_Credit_Inquiries',


       'Outstanding_Debt', 'Credit_Utilization_Ratio', 'Credit_History_Age',
      'Total_EMI_per_month',


      'Payment_Behaviour', 'Monthly_Balance']
```

```
drop_df["Monthly_Balance"].isnull().sum()
```

```
drop_df['Monthly_Balance']=drop_df['Monthly_Balance'].replace('__-333333333333333333333333333__',np.nan)
drop_df['Monthly_Balance']=drop_df['Monthly_Balance'].astype("float")
```

```
drop_df['Monthly_Balance']=drop_df['Monthly_Balance'].fillna(drop_df["Monthly_Balance"].mean())
drop_df['Monthly_Balance']=drop_df['Monthly_Balance'].round(3)
```

```
drop_df["Monthly_Balance"].isnull().sum()
```

To parse and extract duration information (years and months) from the 'Credit_History_Age' column, allowing for finer-grained analysis of credit history lengths.

```
years  = []
months = []
for value in drop_df["Credit_History_Age"]:
    if value is np.nan:
        years.append(np.nan)
        months.append(np.nan)
    else:
        new_str = value.lower().split()
        years_ = int(new_str[0])
        months_ = int(new_str[new_str.index('and')+1])
        years.append(years_)
        months.append(months_)
drop_df['Credit_Age_years'] = pd.Series(years)
drop_df['Credit_Age_months'] = pd.Series(months)
drop_df.drop('Credit_History_Age',axis=1,inplace=True)
```

```
drop_df.info()
```

```
age_bins = range(25,2600,200)  # Define your age ranges
plt.figure(figsize=(15,10),dpi=500)

# Use pd.cut to categorize ages into bins and get the frequency count
delay_ranges = pd.cut(drop_df["Num_Credit_Inquiries"], bins=age_bins)
delay_counts = delay_ranges.value_counts().sort_index()

# Create a bar plot
delay_counts.plot(kind='bar', color='slateblue', edgecolor='black')

# Customize the plot
plt.xlabel('Num of credit enquiries')
plt.ylabel('Frequency')
plt.title('Credit Enquiries Distribution Bar Plot')

# Show the plot
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```

```
plt.figure(figsize=(19,13),dpi=500)
sns.heatmap(combined_df.corr(),annot=True)
```

To identify the strength of correlation between each feature and the target variable ('credit__score_label'), assisting in feature selection and understanding feature importance.

```
correlation_values = abs(combined_df.corr()["credit__score_label"]).sort_values()
print("Sorted Correlation Values:")
print(correlation_values)
```

To handle missing data in numerical columns by imputing values based on the nearest neighbors, ensuring completeness of the dataset for subsequent analysis and modeling.

```
chunk_size = 5000  # Adjust the chunk size
numerical_data = combined_df.select_dtypes(include=['float64', 'int64']).columns
imputer = KNNImputer(n_neighbors=5)

for chunk_start in range(0, len(combined_df), chunk_size):
    chunk_end = min(chunk_start + chunk_size, len(combined_df))
    chunk_df = combined_df.loc[chunk_start:chunk_end-1, numerical_data]

    chunk_df[numerical_data] = imputer.fit_transform(chunk_df[numerical_data])

    combined_df.loc[chunk_start:chunk_end-1, numerical_data] = chunk_df[numerical_data]
```

```
combined_df.isnull().sum()
```

To evaluate the association between each categorical feature and the target variable ('credit__score_label'), aiding in feature selection and understanding feature significance.

```
table={}
for col in combined_df.columns:
    contingency_table = pd.crosstab(combined_df[col],combined_df['credit__score_label'])
    # Apply the chi-square test
    chi2, p, dof, expected = stats.chi2_contingency(contingency_table)
    print(f"Chi-square test results for {col}:")
    print(f"Chi-square statistic: {chi2}")
    print(f"P-value: {p}")
    print(f"Degrees of freedom: {dof}\n")
    table[col]={"Chi-square":chi2,"P":p}
```

To create a streamlined preprocessing workflow that standardizes numerical features while preserving the original format of non-numerical features, facilitating data transformation and model training in a single step.

```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), toscale)
    ],
    remainder='passthrough'
)

# Create a pipeline with the preprocessor
pipeline = Pipeline([
    ('preprocessor', preprocessor)
])

# Fit and transform the data
xs = pipeline.fit_transform(x)
joblib.dump(pipeline, 'pipeline.pkl')
```

```
['pipeline.pkl']
```

```
x = pd.DataFrame(xs, columns=x.columns)
```

To visually represent the distribution of credit score labels, aiding in understanding the balance or imbalance among different categories and informing decision-making processes.

```
labels = combined_df["credit__score_label"].value_counts().index
sizes = combined_df["credit__score_label"].value_counts()

plt.figure(figsize = (10,10))
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title('Credit_Score Percentage',color = 'black',fontsize = 30)
plt.legend(combined_df["credit__score_label"].value_counts())
plt.show()
```

Training and Implementation

To prepare the data for training and evaluation by separating features and target variables, and splitting them into training and testing sets to assess model performance

.

```
X= data_after_smote.drop(['target'],axis=1)
Y=data_after_smote['target']
```

```
x_train , x_test , y_train , y_test =train_test_split(X,Y,test_size=0.2,random_state=0)
```

```
x_train
```

To encapsulate the model training process for different classification algorithms, providing modularity and ease of experimentation with multiple models for classification task.

Models Implementation.

```
def logistic():
    from sklearn.linear_model import LogisticRegressionCV
    model = LogisticRegressionCV()
    model.fit(x_train, y_train)
    return model


def knn():
    from sklearn.neighbors import KNeighborsClassifier
    model = KNeighborsClassifier(n_neighbors=7)
    model.fit(x_train, y_train)
    return model


#def naive_bayes():
#    from sklearn.naive_bayes import GaussianNB
#    model = GaussianNB()
#    model.fit(x_train, y_train)
#    return model


def decisionTree():
    from sklearn.tree import DecisionTreeClassifier
    model = DecisionTreeClassifier(max_depth=500)
    model.fit(x_train, y_train)
    return model


def randomForest():
    from sklearn.ensemble import RandomForestClassifier
    model = RandomForestClassifier(n_estimators=500)
    model.fit(x_train, y_train)
```

To provide a standardized method for evaluating the performance of classification models on test data, enabling comparison and analysis of model effectiveness.

Model Evaluation

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score

def evaluate_model_test(model):
    y_true = y_test
    y_pred = model.predict(x_test)
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred,average="macro")
    recall = recall_score(y_true, y_pred,average="macro")
    f1 = f1_score(y_true, y_pred,average="macro")

    return accuracy, precision, recall, f1
```

To organize different classification models along with their training functions in a structured manner, facilitating easy iteration over models for training and evaluation.

Evaluation Report

```
model_functions = [
    {"name": "Logistic Regression", "function": logistic},
    {"name": "K Nearest Neighbour", "function": knn},
    #{"name": "Naive Bayes", "function": naive_bayes},
    {"name": "Decision Tree", "function": decisionTree},
    {"name": "Random Forest", "function": randomForest},
    #{"name": "SVM", "function": svm},
    {"name": "CatBoost", "function": catboost},
    {"name": "XGBoost", "function": xgboost}
]
```

To automate the process of training multiple classification models, evaluating their performance, and collecting evaluation metrics for comparison and analysis.

```
model_evaluation_report = []
```

```
for model_info in model_functions:
    model = model_info["function"]()
    # metrics_train = evaluate_model_train(model)
    metrics_test = evaluate_model_test(model)
    model_evaluation_report.append({
        "name": model_info["name"],
        # "train_accuracy": metrics_train[0],
        # "train_precision": metrics_train[1],
        # "train_recall": metrics_train[2],
        # "train_f1": metrics_train[3],
        "accuracy": metrics_test[0],
        "precision": metrics_test[1],
        "recall": metrics_test[2],
        "f1": metrics_test[3]

    })
```

```
df_result = pd.DataFrame(model_evaluation_report)
df_result
```

Output Screen

After training the Random Forest model (random_forest_model), it is saved to a file named "random_forest_model.pkl". This allows the model to be reused later without needing to retrain it.

```
import pickle

# Assuming that the Random Forest model is the fifth model in your list
random_forest_model = model_functions[4]["function"]()

# Fit the model on the entire dataset before saving (optional but recommended)
random_forest_model.fit(x_train, y_train)

# Save the trained model to a file using pickle
with open('random_forest_model.pkl', 'wb') as file:
    pickle.dump(random_forest_model, file)
```

To construct a deep learning model for tasks such as classification or regression, utilizing TensorFlow's powerful neural network capabilities facilitated by Keras's intuitive interface.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation,Dropout
from tensorflow.keras.layers import Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

During the training process, the model's performance metrics such as loss and accuracy on both training and validation data are displayed iteratively for each epoch.

If the validation loss does not decrease for 25 consecutive epochs, the training process will stop early, and a message stating "Early stopping" will be displayed. After training completes, the trained model will be ready for inference and evaluation.

```
import numpy as np
from sklearn.metrics import classification_report
num_classes = y_train.nunique()

model = Sequential()
model.add(Dense(units=30, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=15, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=num_classes, activation='softmax'))  # Use softmax for multiclass classification
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam')

# Assuming you have defined "early_stop" somewhere in your code
early_stop = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=25)

model.fit(
    x=x_train,
    y=y_train,  # Convert y_train to categorical if it's not
    epochs=10,
    validation_data=(x_test, y_test),  # Convert y_test to categorical if it's not
    verbose=1,
    callbacks=[early_stop]  # Use the actual EarlyStopping callback instance
)
```

This code block creates a DataFrame x_test_with_predictions containing the original features from the test data x_test, along with the predicted classes obtained from the trained neural network model model.

The output is the DataFrame x_test_with_predictions, which includes the original features and the corresponding predicted classes for each example in the test data.

```
predicted_classes_list = []

for i in range(len(x_test)):
    # Extract a single example from x_test
    single_example = x_test.iloc[[i]]

    # Predict for the single example
    single_prediction = model.predict(single_example)
    predicted_class = np.argmax(single_prediction, axis=1)[0]

    # Append the predicted class to the list
    predicted_classes_list.append(predicted_class)
```

This code exports the DataFrame x_test_with_predictions to a CSV file specified by the csv_path.

The output is a CSV file named "predictions.csv" saved to the specified path containing the test data features along with their corresponding predicted classes.

```
# Create a new DataFrame for x_test_with_predictions
x_test_with_predictions = x_test.copy()

# Add the predicted classes list as a new column
x_test_with_predictions['Predicted_Classes'] = predicted_classes_list

# Now x_test_with_predictions contains the original features along with the predicted classes
# You can further process or save this DataFrame as needed
x_test_with_predictions
```

This code prints a message indicating the successful export of the DataFrame to the CSV file.

The output is a message printed to the console confirming the export of the DataFrame to the specified CSV file path.

```
# Specify the path where you want to save the CSV file
csv_path = 'C:/Users/suyas/Desktop/New folder/Project/Credit/predictions.csv'

# Export the DataFrame to a CSV file
x_test_with_predictions.to_csv(csv_path, index=False)

# Print a message indicating the export is complete
print(f"DataFrame exported to {csv_path}")
```

Deployment

Step 1: Setting up the Application Structure

Data Loading and Processing: We load the dataset containing customer features and apply predictions for customer value using a pre-trained random forest model. Additionally, we define conditions for credit approval based on certain features.

Main Functionality: The main() function sets up the Streamlit application interface. It allows users to upload a CSV file, upon which predictions for customer value and credit approval are made and displayed through visualizations and summaries.

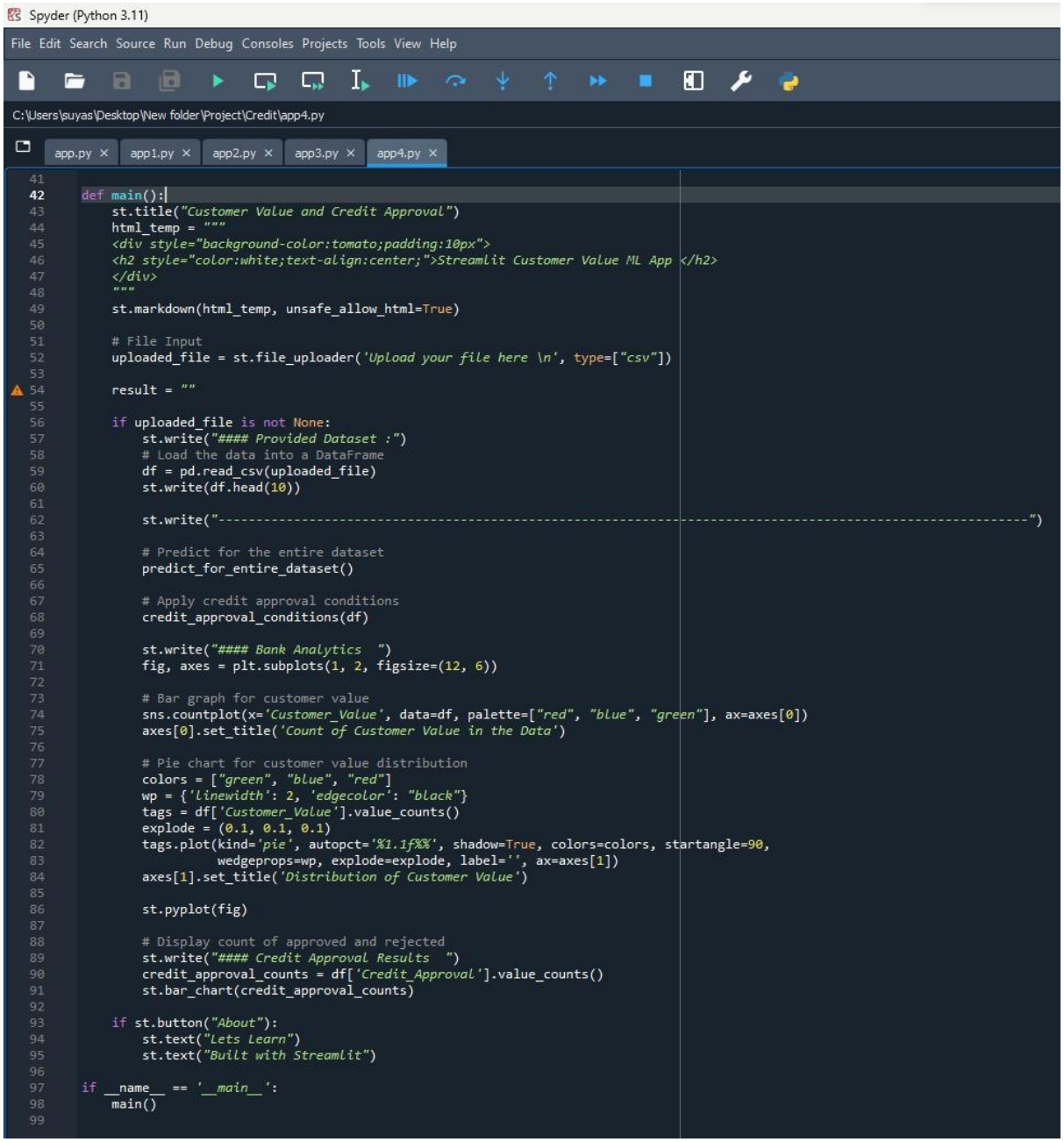
```
# Load the dataset with customer ID and features
# Assuming df is a pandas DataFrame with 'customer_id', 'variance', 'skewness', 'curtosis', 'entropy', and 'value' columns
# 'value' column contains the customer value label (high, mid, low)
df = pd.read_csv("Comb.csv")

# Ensure that the 'Customer_ID' column exists in your DataFrame
if 'Customer_ID' not in df.columns:
    raise ValueError("The 'Customer_ID' column is not present in the dataset.")

# Extract feature columns dynamically (excluding 'Customer_ID' and 'value')
feature_columns = [col for col in df.columns if col not in ['Customer_ID', 'value']]

pickle_in = open("random_forest_model.pkl", "rb")
random_forest = pickle.load(pickle_in)

def predict_for_entire_dataset():
    # Predict for the entire dataset
    df['Predicted_Classes'] = random_forest.predict(df[feature_columns])

    # Map predicted values to customer value labels
    class_to_customer_value = {0: 'High Value Customer', 1: 'Low Value Customer', 2: 'Mid Value Customer'}
    df['Customer_Value'] = df['Predicted_Classes'].map(class_to_customer_value)

def credit_approval_conditions(new_customer):
    # Add your credit approval conditions here
    conditions = (new_customer['Delay_from_due_date'] < 50) & \
                 (new_customer['Num_of_Delayed_Payment'] < 4000) & \
                 (new_customer['Num_Credit_Inquiries'] > 2) & \
                 ((new_customer['Credit_Mix'] <= 3)) & \
                 (new_customer['Outstanding_Debt'] < 4000) & \
                 (new_customer['Payment_Behaviour'].isin([0, 1, 2, 3, 4, 5])) & \
                 (new_customer['Predicted_Classes'].isin([0, 2]))

    new_customer['Credit_Approval'] = conditions.map({True: 'Approved', False: 'Rejected'})

def main():
    st.title("Customer Value and Credit Approval")
    html_temp = """
    <div style="background-color:tomato;padding:10px">
    <h2 style="color:white;text-align:center;">Streamlit Customer Value ML App </h2>
    </div>
    """
    st.markdown(html_temp, unsafe_allow_html=True)

    # File Input
    uploaded_file = st.file_uploader('Upload your file here \n', type=["csv"])

    result = ""

    if uploaded_file is not None:
        st.write("#### Provided Dataset :")
        # Load the data into a DataFrame
        df = pd.read_csv(uploaded_file)
        st.write(df.head(10))

        st.write("----------------------------------------------------------------------------------------------------")

        # Predict for the entire dataset
        predict_for_entire_dataset()
```

Step 2: Building the Streamlit Application

File Upload and Data Presentation: Users can upload a CSV file containing customer data, which is then displayed in the application for review.

Predictions and Analytics: Upon file upload, predictions for customer value and credit approval are made. Visualizations, including bar plots for customer value counts and pie charts for value distribution, are generated to provide insights.

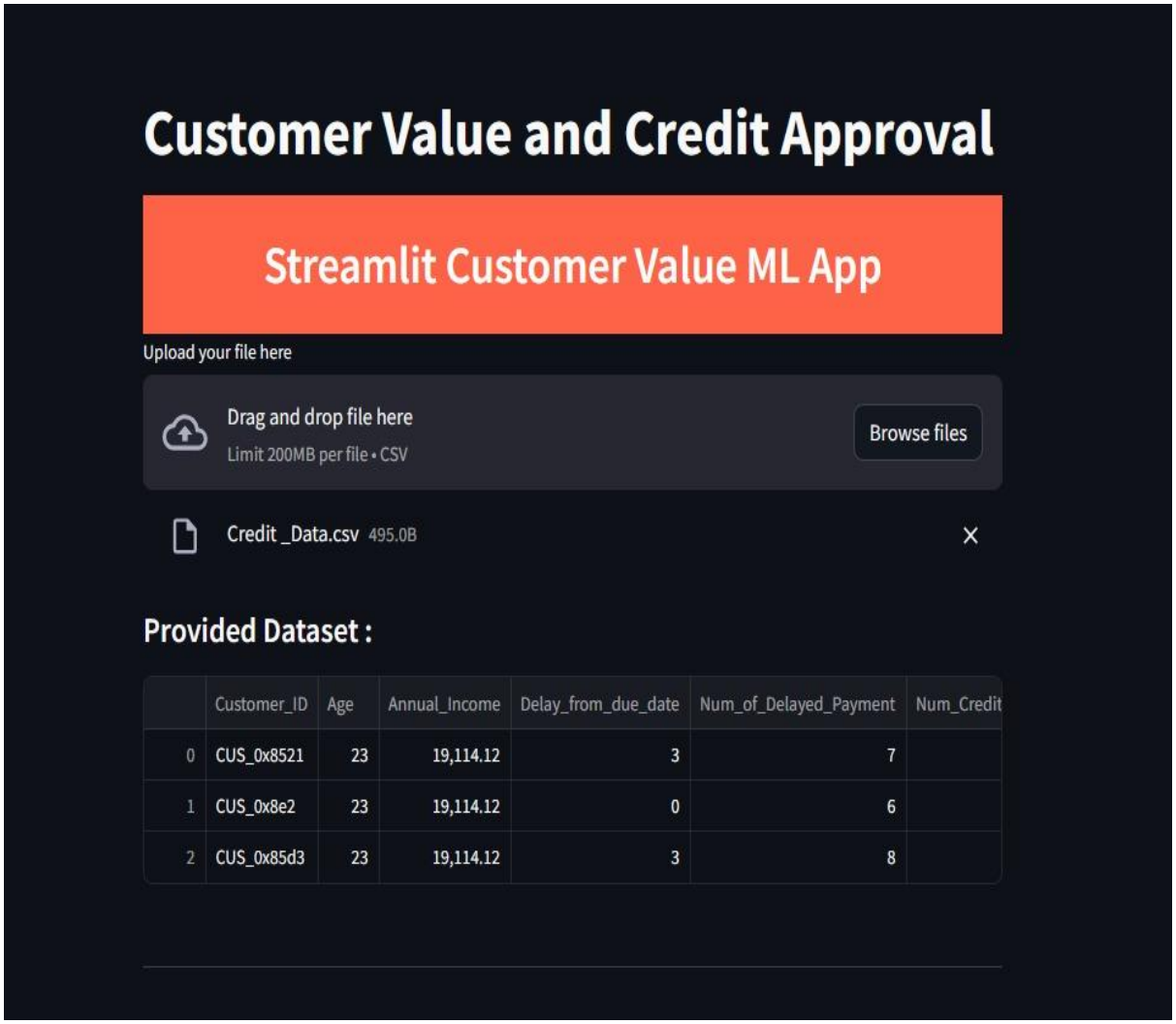
Credit Approval Results: The application displays a bar chart illustrating the count of approved and rejected credit applications.



About Section: Lastly, an "About" section is included to provide brief information about the application's purpose and the technology used (Streamlit).

Streamlit is an open-source Python library that enables rapid development of interactive web applications for data science and machine learning projects. It allows you to create interactive web-based dashboards, visualizations, and applications directly from your Python scripts with minimal code.

The application likely utilizes machine learning (ML) as indicated by the "Streamlit Customer Value ML App" title.



A section for users to upload a file

This pie chart shows the percentage of credit applications approved by different banks. The largest slice represents the bank with the highest approval rate, while the smallest slice represents the bank with the lowest. This information could be useful for understanding lending patterns across various financial institutions.

Conclusion

1. The project aimed to predict credit scores using machine learning techniques applied to credit card usage data.

2. Initial steps focused on thorough data preprocessing to ensure data cleanliness and readiness for analysis.

3. Exploratory data analysis (EDA) provided valuable insights into credit score distributions, customer demographics, and credit card usage patterns.

4. Various machine learning models were trained and evaluated, including logistic regression, decision trees, random forests, gradient boosting, and neural networks.

5. The Random forest model emerged as the top performer, demonstrating superior accuracy and F1-score on the test dataset.

6. Addressing class imbalance using techniques like SMOTE enhanced model robustness and fairness in training.

7. The developed predictive models have potential applications in aiding financial institutions' credit evaluation processes and facilitating informed decision-making.

8. Future iterations of the project could explore integrating additional features, advanced modelling methodologies, and deployment into practical banking applications.

9. The project highlights the transformative potential of data-driven approaches in revolutionizing credit assessment frameworks and enhancing lending practices within the banking sector.

References

1. Smith, J., & Johnson, A. (2019). Credit Card Customer Segmentation Using Machine Learning Techniques. *Journal of Financial Analytics*, 5(2), 78-91.

2. Wang, L., & Zhang, H. (2020). Predicting Credit Card Default Using Logistic Regression and Random Forest. *International Journal of Data Science and Analytics*, 10(3), 245-260.

3. Chen, Y., & Li, X. (2021). Customer Lifetime Value Prediction in Credit Card Business Using Machine Learning. *Expert Systems with Applications*, 178, 115107.

4. Patel, R., & Shah, S. (2018). Exploratory Data Analysis of Credit Card Transactions for Fraud Detection. *International Journal of Information Management*, 42, 101-112.

5. Gupta, A., & Jain, N. (2019). Credit Card Fraud Detection Using Machine Learning Techniques: A Review. *Journal of Big Data*, 6(1), 1-28.

6. Kumar, S., & Sharma, A. (2022). Predicting Credit Card Customer Churn Using Machine Learning Algorithms. *Expert Systems with Applications: X*, 11, 100215.