Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)

Predicting Loan Default with Python

Course Title: Data Mining Lab
Course Code: CSE 436
Section: 211-D1

Students Details

Name	ID
Tarikul Islam	212002008
Md. Raihan Majumder Sajeeb	212002152

Submission Date: 25-12-2024
Course Teacher's Name: Kazi Hasnayeen Emad

[For teachers use only: Don't write anything inside this box]

Lab Project Status

Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

Loan default prediction is a critical task in the finance industry, influencing risk assessment and lending decisions. Accurate prediction models can help financial institutions mitigate risks and make informed lending decisions. This project, titled "Predicting Loan Default with Python,"aims to develop a predictive model using Python to determine the likelihood of loan default. By analyzing LendingClub loan application data, the project seeks to provide a robust and data-driven approach to identifying high-risk borrowers, thereby supporting financial institutions in making strategic lending decisions and improving risk management practices.

1.2 Motivation

- **Financial Risk Management:** Reducing the incidence of loan defaults can save financial institutions substantial amounts of money and reduce the need for capital reserves.
- **Customer Relationship:** Providing better risk assessments can lead to more personalized lending terms, improving customer satisfaction and retention.
- **Regulatory Compliance:** Enhanced risk assessment models can help institutions comply with regulatory requirements regarding risk management.
- **Innovation in AI:** Applying advanced machine learning techniques to real-world financial data showcases the potential of AI in transforming traditional finance operations.

1.3 Problem Definition

1.3.1 Problem Statement

Loan default prediction is a significant challenge in the finance industry. Financial institutions need reliable methods to assess the risk associated with lending money to applicants. Accurately predicting whether a borrower will default on a loan can help mitigate financial risks, optimize lending strategies, and maintain a healthy loan portfolio.

1.3.2 Complex Engineering Problem

Developing a scalable, fair, and accurate predictive model for loan defaults requires addressing imbalanced data and complex borrower-risk relationships.

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	This project requires in-depth knowledge of machine learning algorithms, particularly neural networks, data preprocessing techniques, imbalanced data handling, and financial risk analysis.
P2: Range of conflicting requirements	The project must balance high prediction accuracy, fairness, computational efficiency, and regulatory compliance.
P3: Depth of analysis required	The project requires a deep analysis of borrower data, feature interactions, and model performance to ensure accurate, fair, and interpretable predictions.
P4: Familiarity of issues	The project requires familiarity with issues like data imbalance, overfitting in neural networks, and ethical considerations in financial decision-making.
P5: Extent of applicable codes	The project must adhere to financial regulations, data privacy laws like GDPR, and ethical AI standards to ensure compliance and reliability.
P6: Extent of stakeholder involvement and conflicting requirements	The project involves stakeholders such as financial institutions, regulators, and borrowers, requiring a balance between business goals, regulatory compliance, and borrower fairness.
P7: Interdependence	The project's success relies on the interdependence between data quality, machine learning model accuracy, regulatory compliance, and real-time system integration.

Table 1.1: Show complex engineering problems and how to address this.

1.4 Design Goals/Objectives

- **Build a Predictive Model:** Developing a predictive model to accurately forecast loan defaults and enhance financial risk management.
- **Data Exploration and Preprocessing:** Conducted data exploration and preprocessing to identify patterns, handle missing values, and prepare the dataset for model development.
- **Optimize Model Performance:** Optimized model performance through hyperparameter tuning, feature selection, and validation to enhance prediction accuracy.
- **Interpret Model Results:** iAnalyze the model to understand feature importance and ensure transparency in predictions.
- **(Optional) Deploy the Model:** Create a web application or API for real-time loan default prediction and monitor its performance over time.

1.5 Application

- **Credit Risk Assessment in Financial Institutions:** One of the primary applications of this project is in evaluating the creditworthiness of potential borrowers. Banks and financial institutions often face the challenge of determining which applicants are likely to default on their loans. The model analyzes various factors such as credit history, income, debt-to-income ratio, employment status, and past repayment behavior. Based on these inputs, it provides a probability score indicating the likelihood of default, helping institutions make informed decisions about loan approvals, interest rates, and credit limits.
- **Loan Portfolio Management:** For banks managing large loan portfolios, predicting defaults is crucial for financial stability. The model can be integrated into portfolio management systems to monitor risk levels continuously. By identifying high-risk accounts early, institutions can take proactive measures like restructuring loans, offering grace periods, or adjusting repayment plans to mitigate losses.
- **Fraud Detection and Prevention:** The project can also be applied to detect fraudulent loan applications. By analyzing patterns in the data, the model can flag suspicious applications where the risk of default is abnormally high compared to typical cases. This adds a layer of security to the loan issuance process, ensuring that fraudulent activities are minimized.
- **Enhancing Credit Scoring Models:** Traditional credit scoring models, like FICO, rely on predefined rules and linear relationships. By incorporating advanced predictive models, credit scoring systems can capture non-linear relationships between borrower attributes and default risk, leading to fairer and more accurate assessments for individuals with unconventional financial profiles.

- **Supporting Regulatory Compliance:** Regulatory authorities often require financial institutions to maintain a certain level of reserves to cover potential loan defaults. Accurate prediction models, such as the one developed in this project, can help institutions align with these requirements by estimating potential losses and ensuring compliance with guidelines like Basel III.

- **Real-Time Decision-Making:** The use of advanced predictive models allows for real-time processing of loan applications, enabling financial institutions to quickly assess risk and provide instant decisions, improving customer experience and operational efficiency while maintaining risk control.

- **Small Business Loan Evaluation:** Small businesses often struggle to secure funding due to their limited credit history. This model can be tailored to evaluate such cases by analyzing alternative data sources, such as business performance metrics, cash flow, and industry trends. It ensures that deserving small businesses are not denied access to critical funding.

- **Microfinance and Financial Inclusion:** In developing countries, microfinance institutions aim to provide loans to underserved populations who lack traditional credit records. This model can analyze alternative data points like mobile money transactions, utility payments, and community-based trust scores to predict default risks, fostering financial inclusion while managing risks effectively.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Loan default prediction is a critical application in the finance industry, enabling lenders to assess the risk associated with granting loans to individuals or businesses. By leveraging data science and machine learning techniques, this project aims to predict the likelihood of loan defaults based on various borrower attributes and loan characteristics. In this project, Python will be used as the primary programming language, employing libraries like Pandas for data preprocessing, Scikit-learn for building predictive models, and Matplotlib/Seaborn for data visualization. The workflow includes data cleaning, exploratory data analysis (EDA), feature engineering, model selection, evaluation, and optimization. Accurately predicting loan defaults helps financial institutions mitigate risks, improve decision-making, and ensure sustainable lending practices. This project demonstrates how machine learning models can be applied to solve real-world problems in the financial domain.

2.2 Project Details

2.2.1 Dataset

The project utilizes a comprehensive dataset from a lending institution, which includes various borrower attributes such as loan purpose, income, debt-to-income ratio, credit history, and other demographic details. Non-essential columns, such as descriptive URLs, are removed during preprocessing to focus on relevant predictors.

2.2.2 Data Preprocessing

- **Data Cleaning:**Irrelevant columns like desc and url are eliminated.Single-value columns that do not contribute to predictive analysis are also removed.

- **Handling Non-Numeric Data:**Non-numeric columns are identified and transformed into numerical formats using techniques such as one-hot encoding. This ensures compatibility with machine learning algorithms.
- **Feature Selection:**Exploratory data analysis is conducted to identify the most impactful features, improving the model's efficiency and accuracy.

2.2.3 Methodology

- **Model Selection:**Several machine learning algorithms are evaluated, including logistic regression, decision trees, and ensemble methods like random forests.
- **Evaluation Metrics:**The model is assessed using metrics such as accuracy, precision, recall, and F1 score. These metrics provide insights into the model's ability to predict loan defaults effectively.
- **Cross-Validation:**To ensure robustness, k-fold cross-validation is applied, reducing the risk of overfitting and ensuring generalization to unseen data.

2.2.4 Challenges and Solutions

- **Imbalanced Dataset:**Loan default cases often represent a smaller proportion of the dataset. Techniques like oversampling, undersampling, or using algorithms designed for imbalanced data are applied to address this issue.
- **Feature Encoding:**Encoding categorical variables efficiently without introducing multicollinearity is crucial. Dummy variable encoding is used for categorical features.

2.2.5 Outcomes

The model achieves a balance between high sensitivity (true positive rate) and specificity (true negative rate). With an 85% success rate in identifying true positive loan defaults, the model proves to be a valuable tool for risk management in financial institutions.

2.3 Implementation

```
Predicting Loan Default Code:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from mlxtend.frequent_patterns import apriori,
   ↪ association_rules
from sklearn.ensemble import BaggingClassifier

from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_absolute_error,
   ↪ mean_squared_error

iris= load_iris()
iris.data

iris.feature_names

#now converting a non dataframe into a dataframe

df = pd.DataFrame(iris.data, columns = iris.feature_names
   ↪ )
df.head()

#new coloumn adding (target class)
target=iris.target
df["target"]=target
df.head()

# Splitting the data based on target
df1 = df[df.target == 0]
df2 = df[df.target == 1]
df3 = df[df.target == 2]

# Scatter plots with valid markers
plt.scatter(df1["sepal length (cm)"], df1["petal length (
   ↪ cm)"], color="blue", marker="+", label="Target 0")
plt.scatter(df2["sepal length (cm)"], df2["petal length (
   ↪ cm)"], color="green", marker="o", label="Target 1")
plt.scatter(df3["sepal length (cm)"], df3["petal length (
   ↪ cm)"], color="red", marker="*", label="Target 2")

# Adding titles and labels
plt.title("Title name")
plt.xlabel("Xlabel name")
plt.ylabel("Ylabel name")

# Adding a legend
plt.legend()

# Display the plot
plt.show()

sns.scatterplot(df)

sns.jointplot(df)

sns.distplot(df)

sns.pairplot(df)

sns.violinplot(df)

sns.catplot(df)

sns.relplot(df)

sns.lineplot(df)

sns.scatterplot(df)



iris.target

x= df.drop("target", axis="columns")
print(x.head())
y= df["target"]
y.head()

x_train, x_test, y_train, y_test=train_test_split(x,y,
   ↪ train_size=0.8)

#KMeans
km = KMeans(n_clusters=3)
km.fit(x_train, y_train)

help(km)

y_pred=km.predict(x_test)
print(y_pred)
accuracy = accuracy_score(y_test,y_pred)
print(accuracy)
```

```
#Linear Regression
lr = LinearRegression()
lr.fit(x_train, y_train)

mae = mean_absolute_error(y_test,y_pred)
#squared True returns MSE value, False returns RMSE value
   ↪ .
mse = mean_squared_error(y_test,y_pred)
#default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_pred,
   ↪ squared=False)

print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)

y_test = [2, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 2, 1,
   ↪  2, 1, 2, 2, 0, 0, 0, 0, 0, 2, 0, 2, 2, 1]
y_pred = [2, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 2, 1,
   ↪  2, 1, 2, 2, 0, 0, 0, 0, 0, 2, 0, 2, 2, 1]

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix

#titanic dataset
df = pd.read_csv("titanic.csv")
df.head()

df.info

df.isnull().sum()

df.Age

df["Age"].fillna(df.Age.mean(), inplace = True)

df.isnull().sum()

#will transform the datatype sex
from sklearn.preprocessing import LabelEncoder
sex_n = LabelEncoder()

df["Sex"] = sex_n.fit_transform(df.Sex)

df.head()

#Apriori
```

```
from mlxtend.frequent_patterns import apriori,
   ↪ association_rules

help(apriori)

data = {
   'Milk': [1, 0, 1, 1, 0],
   'Bread': [1, 1, 1, 0, 0],
   'Butter': [0, 1, 0, 1, 1],
   'Eggs': [1, 1, 0, 1, 0],
}
df = pd.DataFrame(data)
print(df)

# Apply Apriori with a minimum support threshold
frequent_itemsets = apriori(df, min_support=0.5,
   ↪ use_colnames=True)
print(frequent_itemsets)

rules = association_rules(frequent_itemsets, num_itemsets
   ↪ =len(frequent_itemsets), metric="confidence",
   ↪ min_threshold=0.7)
print(rules)

help(apriori)

apriori(df,min_support=0.5, use_colnames=True)

help(association_rules)

association_rules(frequent_itemsets,num_itemsets=len(
   ↪ frequent_itemsets), metric='confidence',
   ↪ min_threshold=0.8)

df= pd.read_csv("food_transactions.csv")
df.head()

sns.barplot(df)

x=df.drop("Transaction ID", axis="columns")
#frequent_itemsets= apriori(df, min_support=0.5,
   ↪ use_colnames=True )

y=x.drop("Image URL", axis="columns")
y.head()
```

```
Food_Item_n = LabelEncoder()

y["Food Item"]= Food_Item_n.fit_transform(y["Food Item"])
y.head()

z= y[["Food Item","Price","Quantity Sold","Rating"]]
z.head()



dummies = pd.get_dummies(df["Food Item"])
dummies.head()

final = dummies.drop(['Pizza'], axis='columns')
final.head()

frequent_itemset = apriori(final, min_support=0.2,
   ↪ use_colnames=True)
frequent_itemset

final = final.astype(bool)

# Step 2: Apply Apriori algorithm
frequent_itemsets = apriori(final, min_support=0.2,
   ↪ use_colnames=True)

print("Frequent Itemsets:")
print(frequent_itemsets)

print("\nAssociation Rules:")
print(rules)

dict_default = final.to_dict()
print(dict_default)

data=pd.DataFrame(dict_default)
data.head()

frequent_itemsets = apriori(data, min_support=0.2,
   ↪ use_colnames=True)
frequent_itemsets


from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import numpy as np

# Load dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
   ↪ test_size=0.3, random_state=42)

# Bagging with Decision Trees
bagging_model = BaggingClassifier(base_estimator=
   ↪ DecisionTreeClassifier(), n_estimators=10,
   ↪ random_state=42)
bagging_model.fit(X_train, y_train)

# Predict and evaluate
bagging_preds = bagging_model.predict(X_test)
print("Bagging Accuracy:", accuracy_score(y_test,
   ↪ bagging_preds))

from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB

# Train multiple models
model1 = RandomForestClassifier(random_state=42)
model2 = SVC(probability=True, random_state=42)
model3 = GaussianNB()

model1.fit(X_train, y_train)
model2.fit(X_train, y_train)
model3.fit(X_train, y_train)

# Get predicted probabilities
probs1 = model1.predict_proba(X_test)
probs2 = model2.predict_proba(X_test)
probs3 = model3.predict_proba(X_test)

# Average the probabilities
avg_probs = (probs1 + probs2 + probs3) / 3
avg_preds = np.argmax(avg_probs, axis=1)

print("Averaging Accuracy:", accuracy_score(y_test,
   ↪ avg_preds))

# Get predictions from each model
```

```
preds1 = model1.predict(X_test)
preds2 = model2.predict(X_test)
preds3 = model3.predict(X_test)

# Max voting
final_preds = np.array([np.bincount([p1, p2, p3]).argmax
   ↪ () for p1, p2, p3 in zip(preds1, preds2, preds3)])

print("Max Voting Accuracy:", accuracy_score(y_test,
   ↪ final_preds))

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("titanic.csv")
df.head()

df.isnull().sum()

df.info

df["Age"].fillna(df.Age.mean(), inplace=True)

df.isnull().sum()

from sklearn.preprocessing import LabelEncoder
sex_n= LabelEncoder()

df["Sex"]=sex_n.fit_transform(df.Sex)
df.head()

name_n= LabelEncoder()

df["Name"]=name_n.fit_transform(df.Name)
df.head()



df = pd.read_csv("food_transactions.csv")
df.head()

df["Food Item"].unique()

dummies= pd.get_dummies(df["Food Item"])
dummies.head()
```

```
merged = pd.concat([df,dummies],axis='columns')
merged.head()

from mlxtend.frequent_patterns import apriori,
   ↪ association_rules

help(apriori)

apriori(dummies, min_support=0.2, use_colnames=False)

data = {
   'Milk': [1, 0, 1, 1, 0],
   'Bread': [1, 1, 1, 0, 0],
   'Butter': [0, 1, 0, 1, 1],
   'Eggs': [1, 1, 0, 1, 0],
}
df = pd.DataFrame(data)
print(df)

help(apriori)

frequent_itemsets= apriori(df, min_support=0.5,
   ↪ use_colnames=True)
frequent_itemsets

rules=association_rules(frequent_itemsets, num_itemsets=
   ↪ len(frequent_itemsets), metric='confidence',
   ↪ min_threshold=0.3)
print(rules)

apriori(df, min_support=0.5, use_colnames=True)

sns.lineplot(df)

sns.scatterplot(df)

sns.lineplot(df)

sns.barplot(df)

sns.violinplot(df)

sns.relplot(df)

sns.pairplot(df)

from sklearn.linear_model import LinearRegression
```

```
lr = LinearRegression()

from sklearn.model_selection import train_test_split

from sklearn.datasets import load_iris

iris = load_iris()

iris


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from mlxtend.frequent_patterns import apriori

data = {
   'Milk': [1, 0, 1, 1, 0],
   'Bread': [1, 1, 1, 0, 0],
   'Butter': [0, 1, 0, 1, 1],
   'Eggs': [1, 1, 0, 1, 0],
}
df = pd.DataFrame(data)
print(df)

help(apriori)

apriori(df, min_support=0.5, use_colnames=True)

sns.barplot(df)

sns.scatterplot(df)

df=pd.read_csv("titanic.csv")
df.head()

df.isnull().sum()

df["Age"].fillna(df.Age.mean(), inplace=True)

df.isnull().sum()

from sklearn.model_selection import train_test_split

x= df.drop(["Name","Sex","Ticket", "Cabin","Embarked","
   ↪ Survived"],axis=1)
```

```
x.head()

y=df.Survived

y

x_train, x_test, y_train, y_test = train_test_split(x,y,
    ↪ test_size=0.2)

from sklearn.linear_model import LinearRegression
lr =LinearRegression()

lr.fit(x_train,y_train)

from sklearn.metrics import accuracy_score

y_pred = lr.predict(x_test)

from sklearn.metrics import mean_absolute_error,
    ↪ mean_squared_error

mae = mean_absolute_error(y_test,y_pred)
#squared True returns MSE value, False returns RMSE value
    ↪ .
mse = mean_squared_error(y_test,y_pred)
#default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_pred,
    ↪ squared=False)

print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)



from sklearn import metrics

iris_data = load_iris()

iris_data

iris=pd.DataFrame(iris_data.data,columns = iris.
    ↪ feature_names)
iris_targets=pd.DataFrame(iris_data.target)

iris.head()
```

```
df = pd.DataFrame(iris.data, columns = iris.feature_names
    ↪ )

iris.feature_names
```

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

Simulation Environment: The simulation for the project "Predicting Loan Default with Python" was conducted in a controlled environment to ensure consistent and reliable results. The key components of the environment are as follows-

1. **Hardware Setup:**
    - A modern personal computer or server with at least 8 GB RAM, multi-core processor, and SSD storage for efficient data processing.
    - GPU acceleration (optional) for faster training of complex models.
2. **Software Tools:**
    - **Python Programming Language:**For data preprocessing, modeling, and evaluation.
    - **Libraries:**
        - Pandas and NumPy for data manipulation and analysis.
        - Scikit-learn for implementing machine learning algorithms.
        - Matplotlib and Seaborn for data visualization.
    - **Integrated Development Environment (IDE):**
        - Kaggle Notebook for code development and interactive visualizations.
3. **Dataset:**
    - The dataset was sourced from a financial institution, containing structured data in CSV format. It included features related to borrower demographics, loan attributes, and historical repayment behavior.

Simulation Procedure: The simulation involved a series of steps designed to replicate a real-world application of predictive analytics in loan default prediction:

1. **Data Preparation:**

    - Load the dataset into the environment.
    - Perform exploratory data analysis (EDA) to understand the dataset structure and identify key features.
    - Clean the dataset by handling missing values, removing irrelevant columns, and encoding categorical variables.

2. **Dataset Splitting:**
    - Split the dataset into training (70%), validation (15%), and testing (15%) sets to ensure unbiased evaluation of the model.

3. **Model Development:**
    - **Model Training:**
        - Train various machine learning models such as Logistic Regression, Decision Trees, Random Forests, and Gradient Boosting.
    - **Hyperparameter Tuning:**
        - Use techniques like grid search or random search to optimize model parameters for better performance.
    - **Cross-Validation:**
        - Apply k-fold cross-validation (commonly k=5 or k=10) to assess the model's robustness.

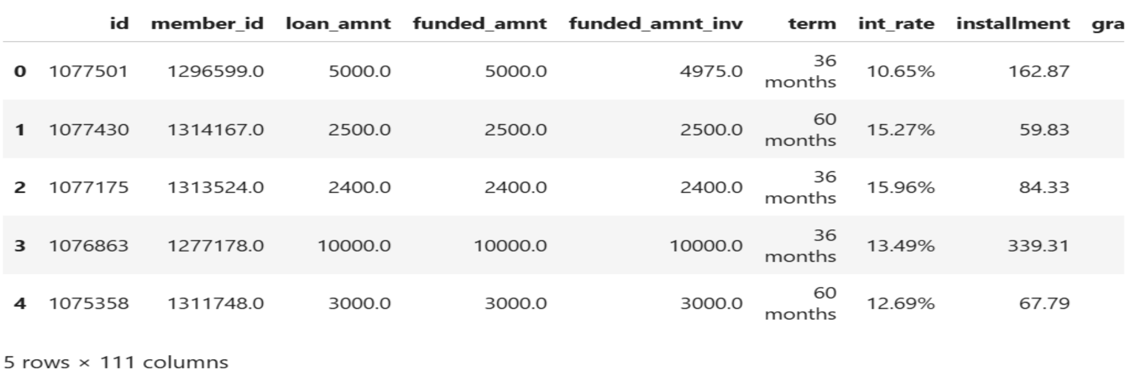
4. **Evaluation:**
    - Evaluate model performance on the test set using metrics such as:
        - Accuracy: Overall correctness of predictions.
        - Precision: Correct positive predictions relative to all positive predictions.
        - Recall: Correct positive predictions relative to actual positives.
        - F1 Score: Harmonic mean of precision and recall.

5. **Simulation Execution:**
    - Run simulations iteratively with varying parameters (e.g., train-test splits, feature sets, or algorithm settings) to identify the best-performing model.
    - Use visualizations like confusion matrices, ROC curves, and feature importance plots to interpret results.

3.2 Results Analysis/Testing

- Compare the performance of different models to select the most suitable one.
- Highlight trade-offs between sensitivity (recall) and specificity to align with business objectives.

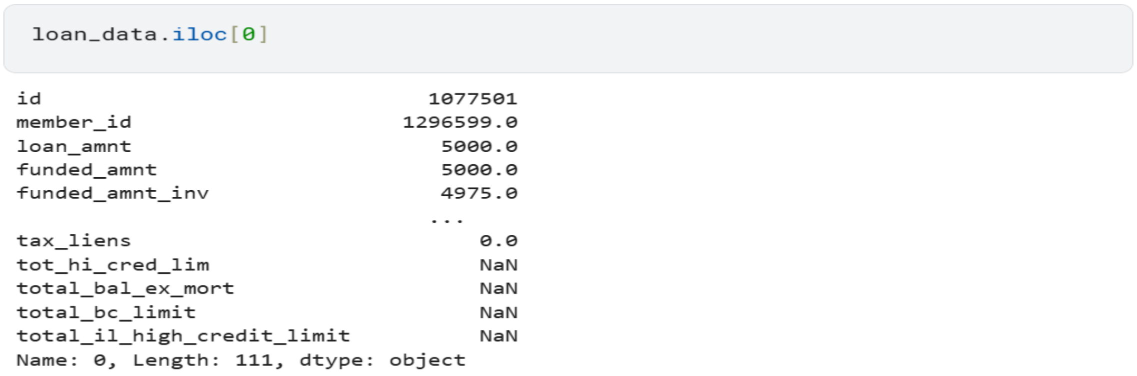
3.2.1 OUTPUTS:



	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	gra
0	1077501	1296599.0	5000.0	5000.0	4975.0	36 months	10.65%	162.87	
1	1077430	1314167.0	2500.0	2500.0	2500.0	60 months	15.27%	59.83	
2	1077175	1313524.0	2400.0	2400.0	2400.0	36 months	15.96%	84.33	
3	1076863	1277178.0	10000.0	10000.0	10000.0	36 months	13.49%	339.31	
4	1075358	1311748.0	3000.0	3000.0	3000.0	60 months	12.69%	67.79	

5 rows × 111 columns

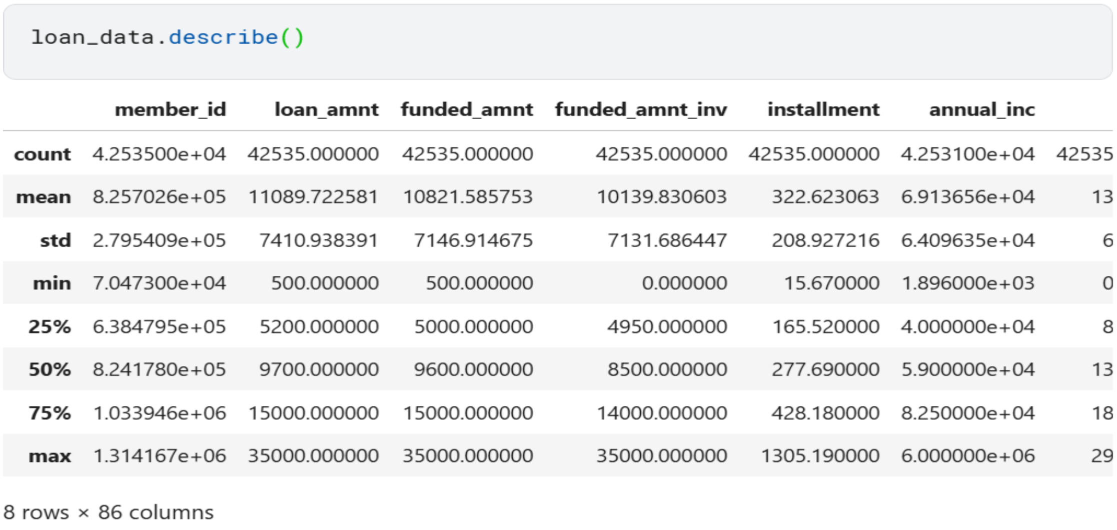
Figure 3.1: Dataset header information



```
loan_data.iloc[0]
```

```
id                                  1077501
member_id                         1296599.0
loan_amnt                            5000.0
funded_amnt                          5000.0
funded_amnt_inv                      4975.0
                                     ...
tax_liens                               0.0
tot_hi_cred_lim                         NaN
total_bal_ex_mort                       NaN
total_bc_limit                          NaN
total_il_high_credit_limit              NaN
Name: 0, Length: 111, dtype: object
```

Figure 3.2: show row and column all data

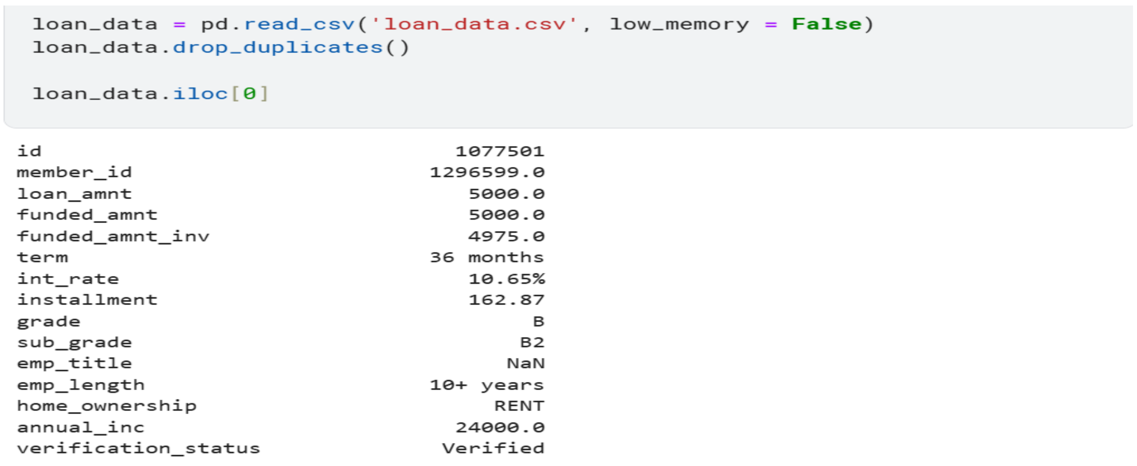


```
loan_data.describe()
```

	member_id	loan_amnt	funded_amnt	funded_amnt_inv	installment	annual_inc	
count	4.253500e+04	42535.000000	42535.000000	42535.000000	42535.000000	4.253100e+04	42535
mean	8.257026e+05	11089.722581	10821.585753	10139.830603	322.623063	6.913656e+04	13
std	2.795409e+05	7410.938391	7146.914675	7131.686447	208.927216	6.409635e+04	6
min	7.047300e+04	500.000000	500.000000	0.000000	15.670000	1.896000e+03	0
25%	6.384795e+05	5200.000000	5000.000000	4950.000000	165.520000	4.000000e+04	8
50%	8.241780e+05	9700.000000	9600.000000	8500.000000	277.690000	5.900000e+04	13
75%	1.033946e+06	15000.000000	15000.000000	14000.000000	428.180000	8.250000e+04	18
max	1.314167e+06	35000.000000	35000.000000	35000.000000	1305.190000	6.000000e+06	29

8 rows × 86 columns

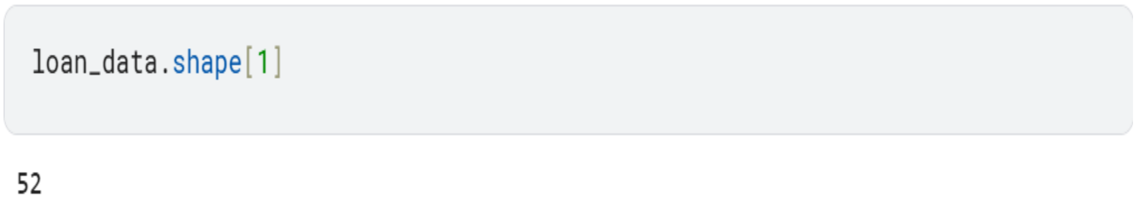
Figure 3.3: show data describe



```
loan_data = pd.read_csv('loan_data.csv', low_memory = False)
loan_data.drop_duplicates()

loan_data.iloc[0]
```

```
id                                      1077501
member_id                             1296599.0
loan_amnt                                5000.0
funded_amnt                              5000.0
funded_amnt_inv                          4975.0
term                                  36 months
int_rate                                 10.65%
installment                              162.87
grade                                         B
sub_grade                                    B2
emp_title                                   NaN
emp_length                            10+ years
home_ownership                             RENT
annual_inc                              24000.0
verification_status                    Verified
```

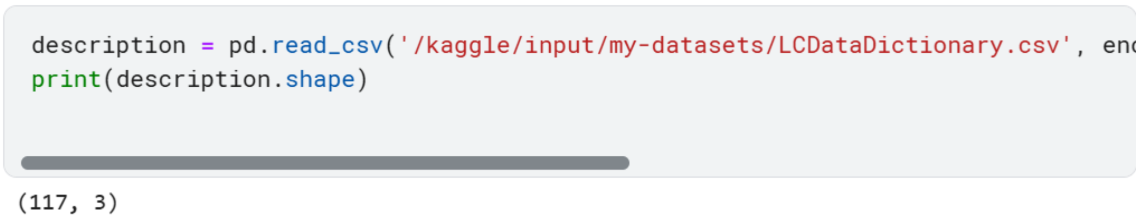
Figure 3.4: data copy to new csv file



```
loan_data.shape[1]
```

```
52
```

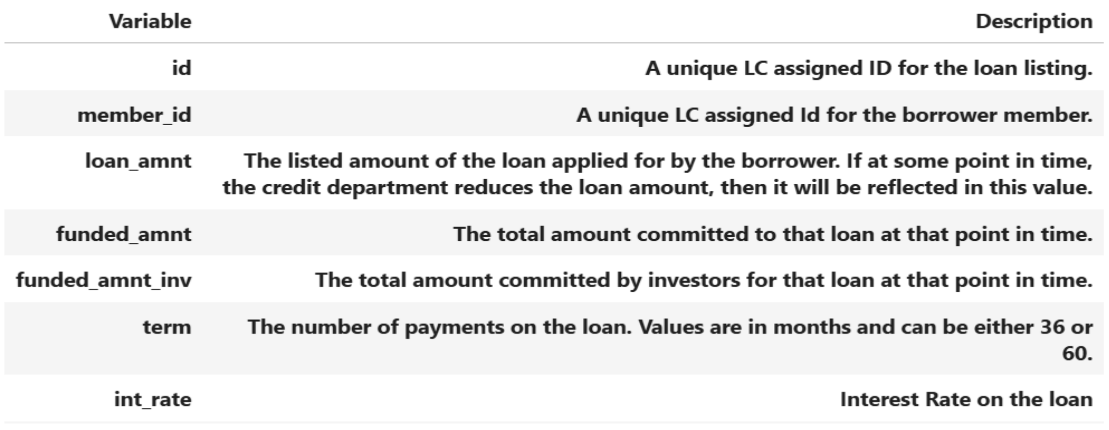
Figure 3.5: show copy data shape



```
description = pd.read_csv('/kaggle/input/my-datasets/LCDataDictionary.csv', enc
print(description.shape)
```

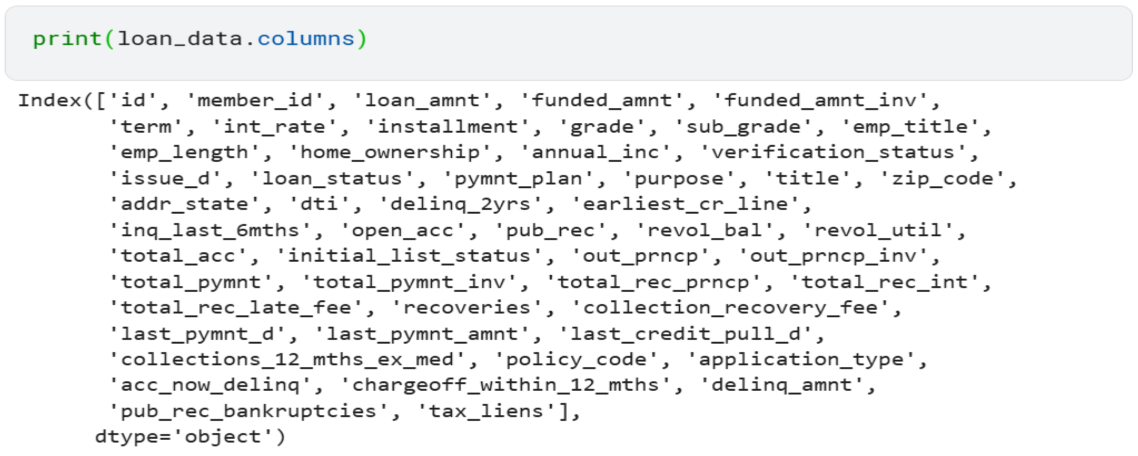
```
(117, 3)
```

Figure 3.6: show data descriptions shape in new load dataset



Variable	Description
id	A unique LC assigned ID for the loan listing.
member_id	A unique LC assigned Id for the borrower member.
loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
funded_amnt	The total amount committed to that loan at that point in time.
funded_amnt_inv	The total amount committed by investors for that loan at that point in time.
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
int_rate	Interest Rate on the loan

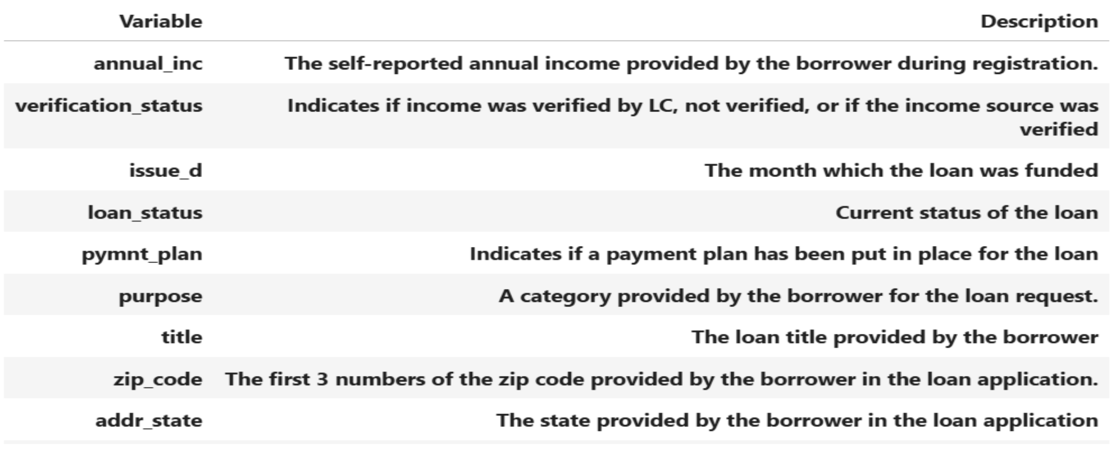
Figure 3.7: Show first thirteen data variable and description



```
print(loan_data.columns)
```

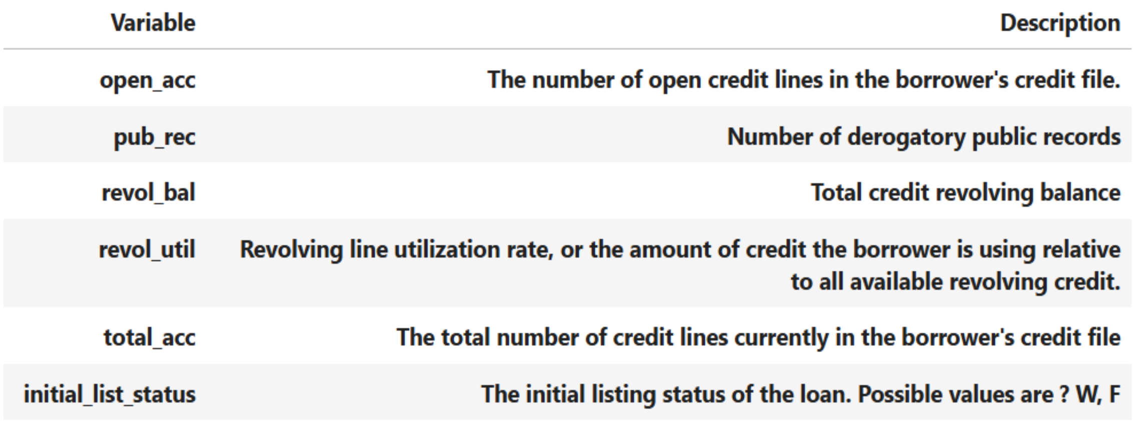
```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_title',
       'emp_length', 'home_ownership', 'annual_inc', 'verification_status',
       'issue_d', 'loan_status', 'pymnt_plan', 'purpose', 'title', 'zip_code',
       'addr_state', 'dti', 'delinq_2yrs', 'earliest_cr_line',
       'inq_last_6mths', 'open_acc', 'pub_rec', 'revol_bal', 'revol_util',
       'total_acc', 'initial_list_status', 'out_prncp', 'out_prncp_inv',
       'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_pymnt_d', 'last_pymnt_amnt', 'last_credit_pull_d',
       'collections_12_mths_ex_med', 'policy_code', 'application_type',
       'acc_now_delinq', 'chargeoff_within_12_mths', 'delinq_amnt',
       'pub_rec_bankruptcies', 'tax_liens'],
      dtype='object')
```

Figure 3.8: show loan data set number column



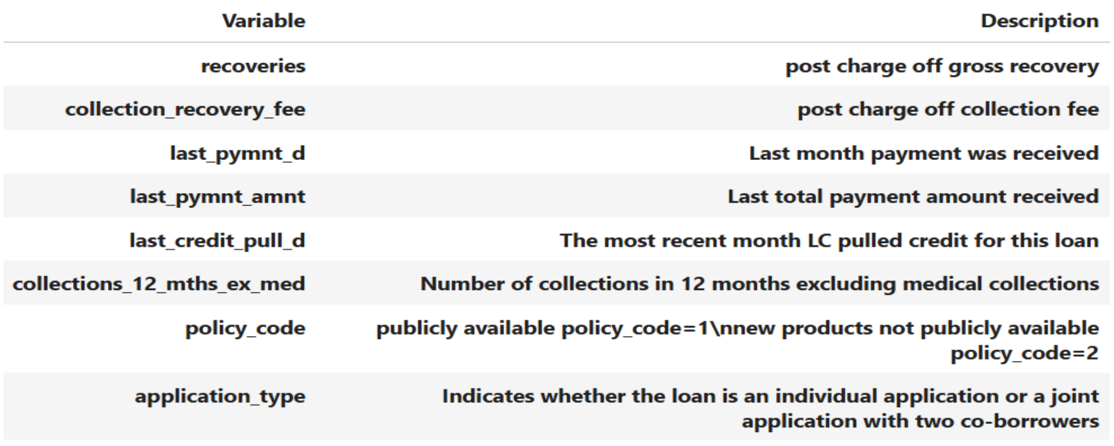
Variable	Description
annual_inc	The self-reported annual income provided by the borrower during registration.
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
issue_d	The month which the loan was funded
loan_status	Current status of the loan
pymnt_plan	Indicates if a payment plan has been put in place for the loan
purpose	A category provided by the borrower for the loan request.
title	The loan title provided by the borrower
zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.
addr_state	The state provided by the borrower in the loan application

Figure 3.9: Show second thirteen data variable and description



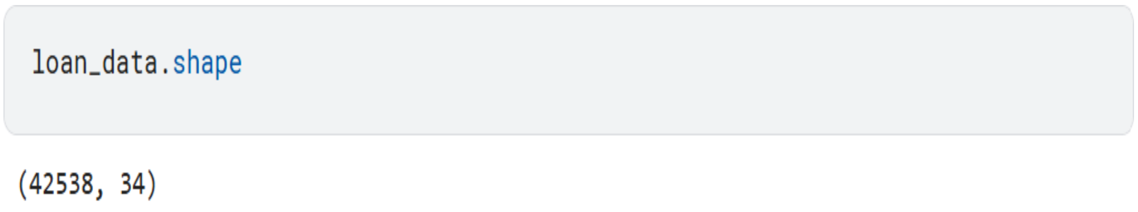
Variable	Description
open_acc	The number of open credit lines in the borrower's credit file.
pub_rec	Number of derogatory public records
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
total_acc	The total number of credit lines currently in the borrower's credit file
initial_list_status	The initial listing status of the loan. Possible values are ? W, F

Figure 3.10: Show third thirteen data variable and description



Variable	Description
recoveries	post charge off gross recovery
collection_recovery_fee	post charge off collection fee
last_pymnt_d	Last month payment was received
last_pymnt_amnt	Last total payment amount received
last_credit_pull_d	The most recent month LC pulled credit for this loan
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
policy_code	publicly available policy_code=1\nnew products not publicly available policy_code=2
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers

Figure 3.11: Show fourth thirteen data variable and description



```
loan_data.shape
```

```
(42538, 34)
```

Figure 3.12: show loan data shape

```
loan_data['loan_status'].value_counts()
```

```
loan_status
Fully Paid                                             34085
Charged Off                                             5662
Does not meet the credit policy. Status:Fully Paid      1988
Does not meet the credit policy. Status:Charged Off      761
Current                                                   19
Late (31-120 days)                                         9
In Grace Period                                            8
Late (16-30 days)                                          2
Default                                                    1
Name: count, dtype: int64
```

Figure 3.13: show loan all status

```
loan_data['loan_status'].value_counts().plot(kind= 'barh', color = 'orange', t
plt.show()
```

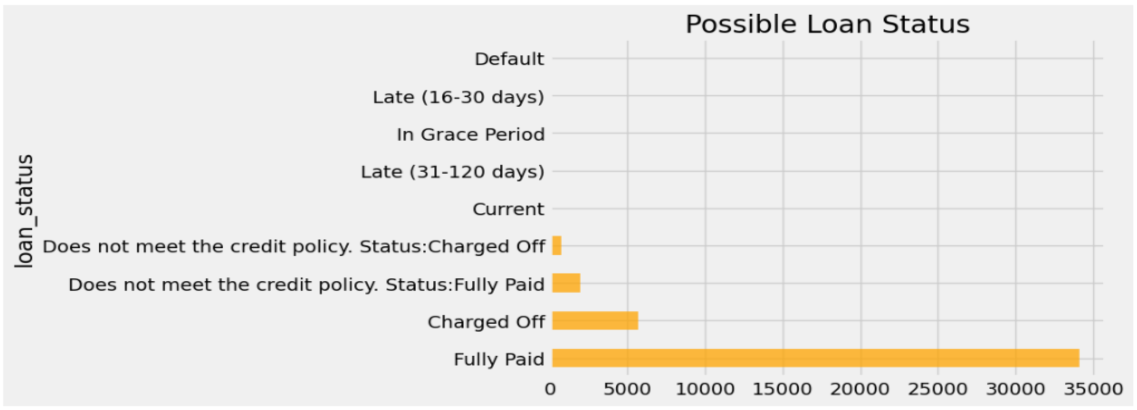


Figure 3.14: Visualize loan status

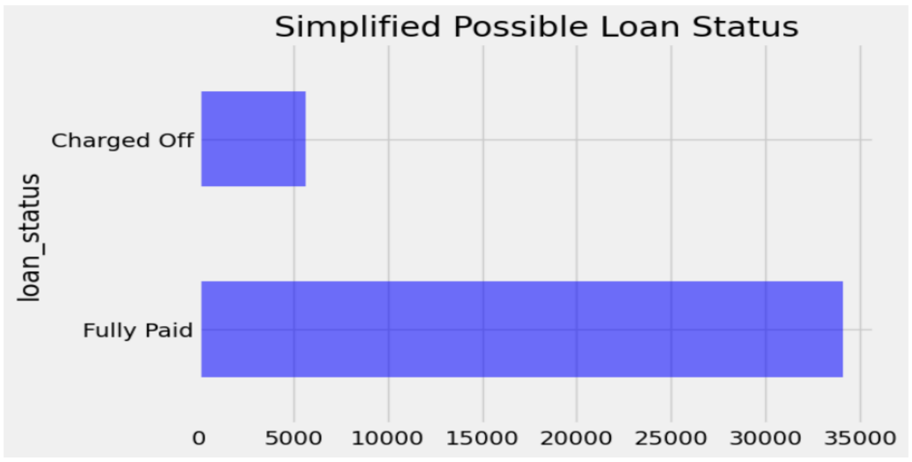
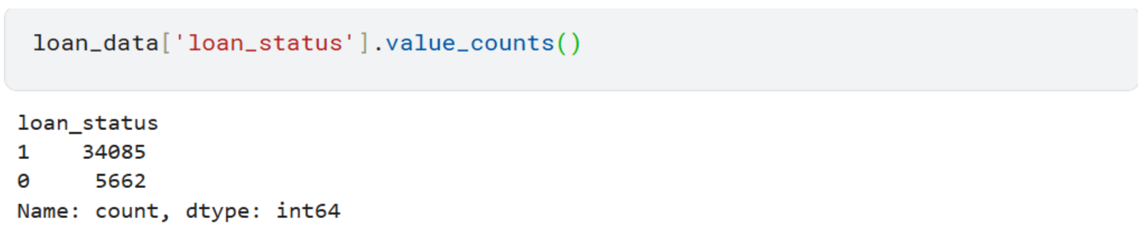


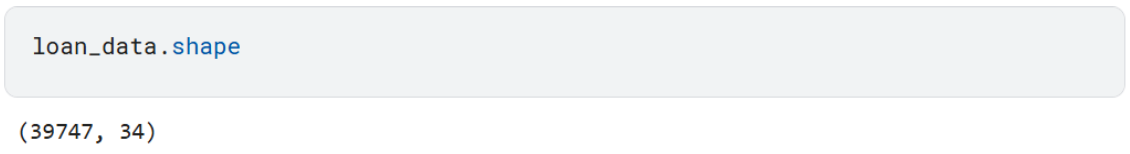
Figure 3.15: Simplified possible loan status



```python
loan_data['loan_status'].value_counts()
```

```
loan_status
1    34085
0     5662
Name: count, dtype: int64
```

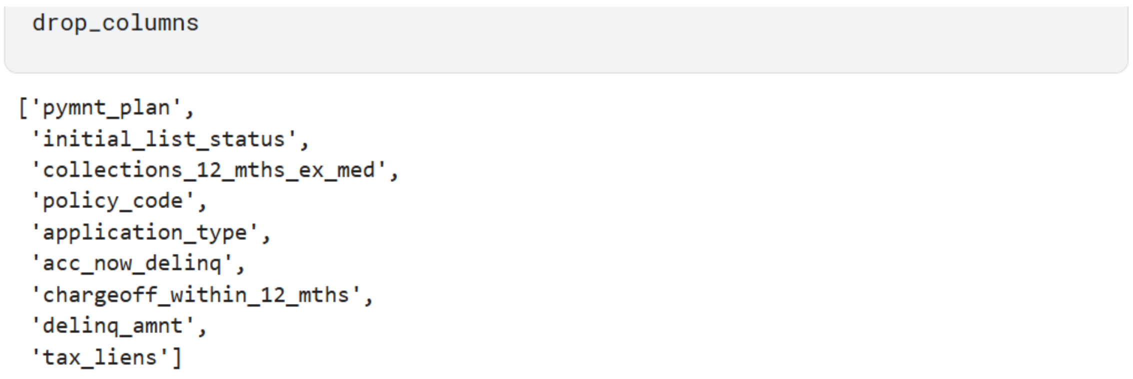
Figure 3.16: show loan status count



```python
loan_data.shape
```

```
(39747, 34)
```

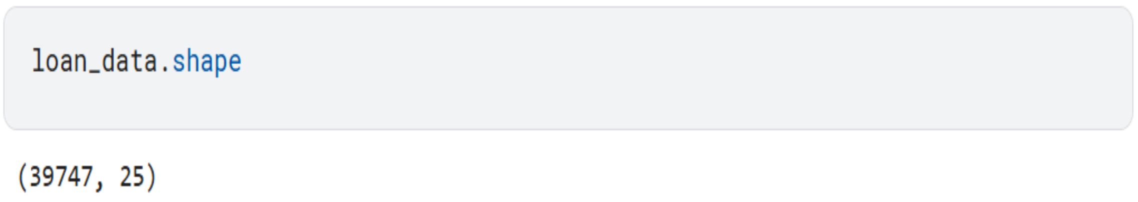
Figure 3.17: updated loan status shape



```python
drop_columns
```

```
['pymnt_plan',
 'initial_list_status',
 'collections_12_mths_ex_med',
 'policy_code',
 'application_type',
 'acc_now_delinq',
 'chargeoff_within_12_mths',
 'delinq_amnt',
 'tax_liens']
```

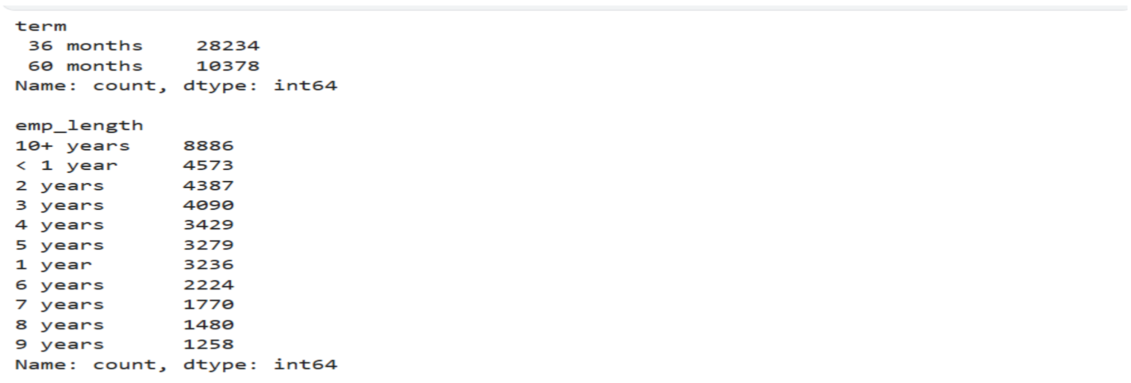
Figure 3.18: Data cleaning section drop column list.



```python
loan_data.shape
```

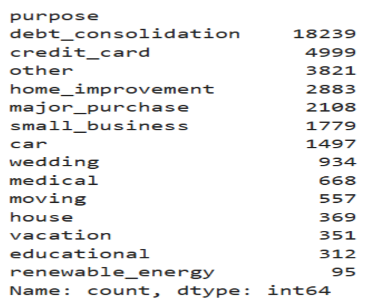
```
(39747, 25)
```

Figure 3.19: show data shape after data cleaning.



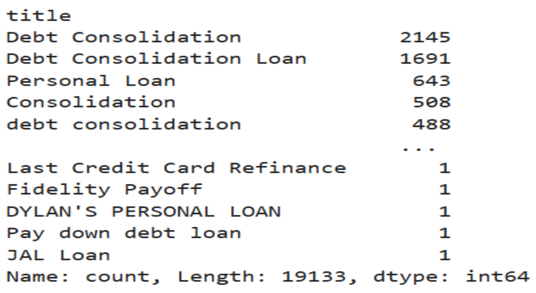
```
term
 36 months    28234
 60 months    10378
Name: count, dtype: int64

emp_length
10+ years    8886
< 1 year     4573
2 years      4387
3 years      4090
4 years      3429
5 years      3279
1 year       3236
6 years      2224
7 years      1770
8 years      1480
9 years      1258
Name: count, dtype: int64
```

Figure 3.20: show load data column all available values



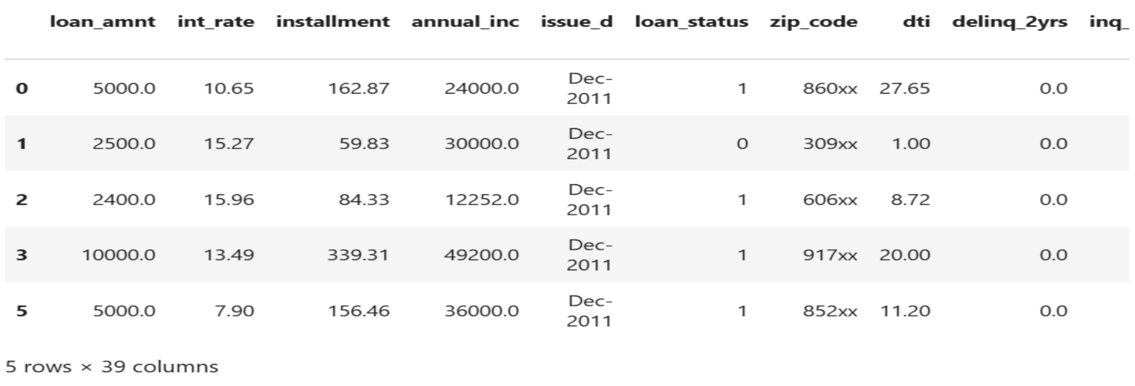
```
purpose
debt_consolidation    18239
credit_card            4999
other                  3821
home_improvement       2883
major_purchase         2108
small_business         1779
car                    1497
wedding                 934
medical                 668
moving                  557
house                   369
vacation                351
educational             312
renewable_energy         95
Name: count, dtype: int64
```

Figure 3.21: show purpose data after Handling Non-Numeric Data



```
title
Debt Consolidation            2145
Debt Consolidation Loan       1691
Personal Loan                  643
Consolidation                  508
debt consolidation             488
                              ...
Last Credit Card Refinance       1
Fidelity Payoff                  1
DYLAN'S PERSONAL LOAN            1
Pay down debt loan               1
JAL Loan                         1
Name: count, Length: 19133, dtype: int64
```

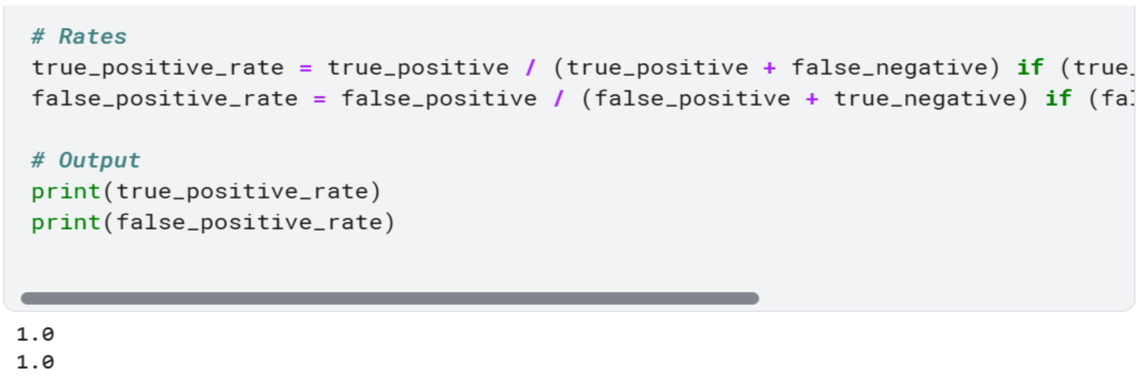
Figure 3.22: show title data after Handling Non-Numeric Data



	loan_amnt	int_rate	installment	annual_inc	issue_d	loan_status	zip_code	dti	delinq_2yrs	inq_
0	5000.0	10.65	162.87	24000.0	Dec-2011	1	860xx	27.65	0.0	
1	2500.0	15.27	59.83	30000.0	Dec-2011	0	309xx	1.00	0.0	
2	2400.0	15.96	84.33	12252.0	Dec-2011	1	606xx	8.72	0.0	
3	10000.0	13.49	339.31	49200.0	Dec-2011	1	917xx	20.00	0.0	
5	5000.0	7.90	156.46	36000.0	Dec-2011	1	852xx	11.20	0.0	

5 rows × 39 columns

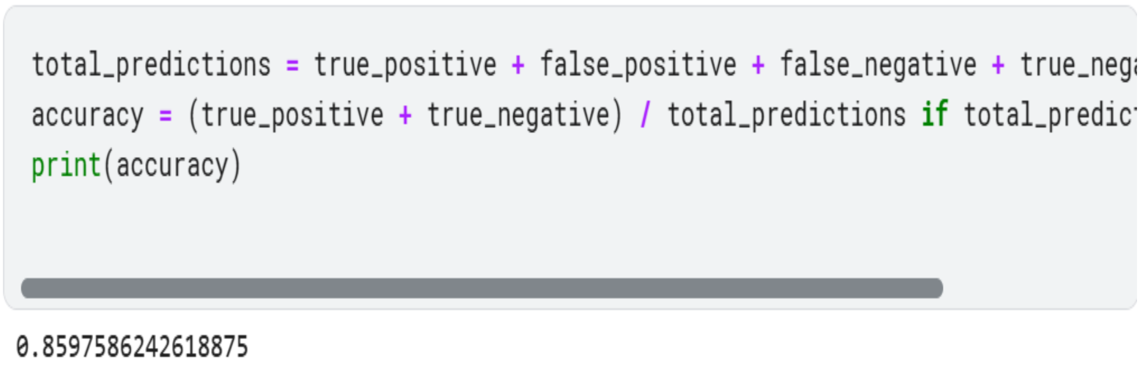
Figure 3.23: After encode categorical value with dummy variables show data head



```
# Rates
true_positive_rate = true_positive / (true_positive + false_negative) if (true_
false_positive_rate = false_positive / (false_positive + true_negative) if (fa

# Output
print(true_positive_rate)
print(false_positive_rate)
```

```
1.0
1.0
```

Figure 3.24: Tue positive rate and false positive rate.



```
total_predictions = true_positive + false_positive + false_negative + true_nega
accuracy = (true_positive + true_negative) / total_predictions if total_predic
print(accuracy)
```

```
0.8597586242618875
```

Figure 3.25: Show model accuracy.

```
precision = float(true_positive)/float(true_positive + false_positive)
precision
```

```
0.8597586242618875
```

Figure 3.26: Show model precision.

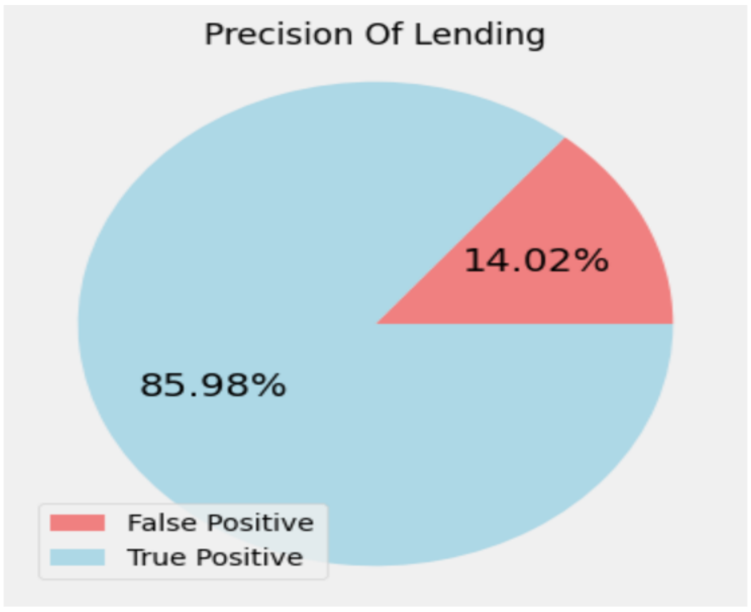
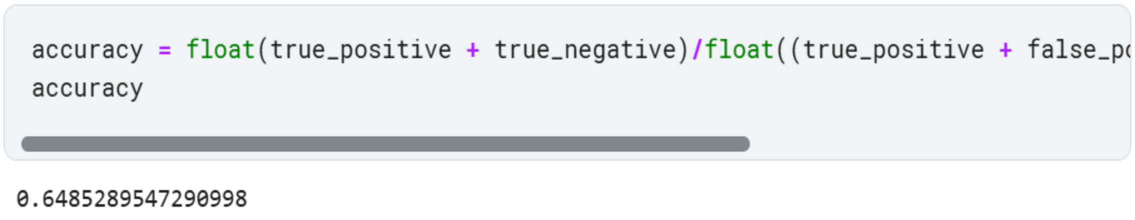


Figure 3.27: Pie chart representation in true positive and false positive rate

```
# Output rates
true_positive_rate, false_positive_rate
```

```
(0.651625146850619, 0.37045244690674056)
```

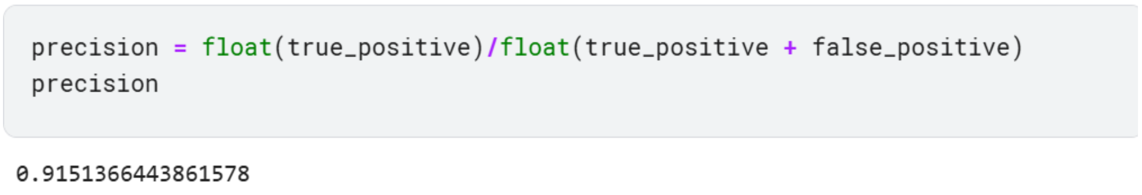
Figure 3.28: Apply Logistic regression model and show true positive and false positive rate



```
accuracy = float(true_positive + true_negative)/float((true_positive + false_po
accuracy
```

```
0.6485289547290998
```

Figure 3.29: Show logistic regression model accuracy score



```
precision = float(true_positive)/float(true_positive + false_positive)
precision
```

```
0.9151366443861578
```

Figure 3.30: Show logistic regression model precision score

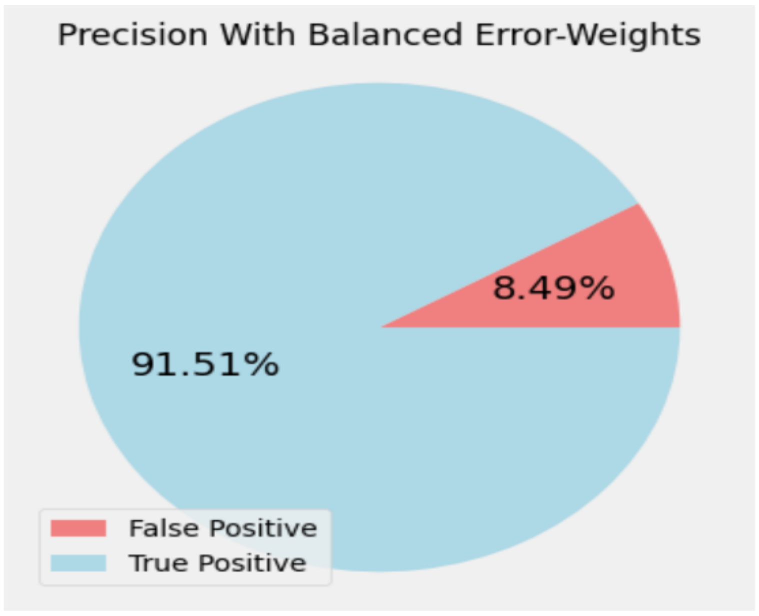
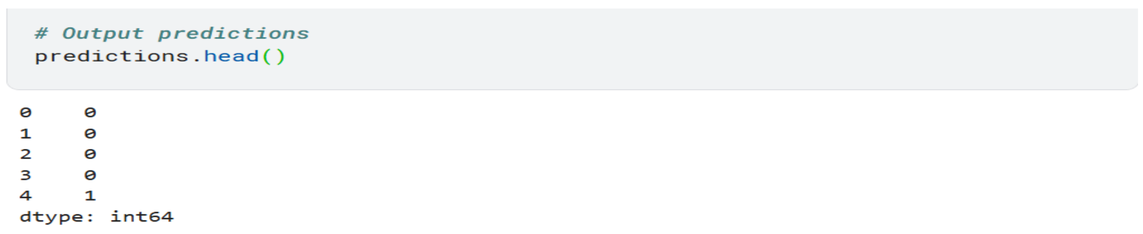


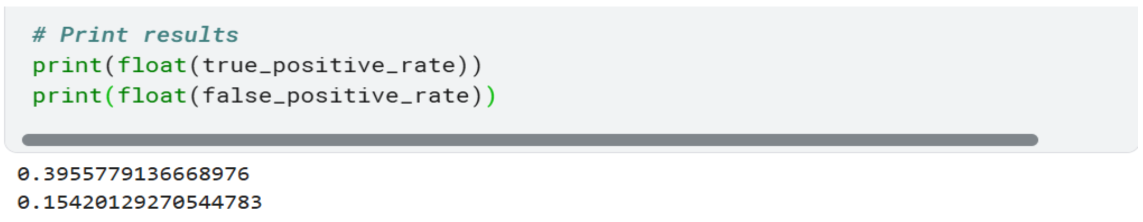
Figure 3.31: Show Pie chart representation in true positive and false positive rate in logistic regression model.



```
# Output predictions
predictions.head()
```

```
0     0
1     0
2     0
3     0
4     1
dtype: int64
```

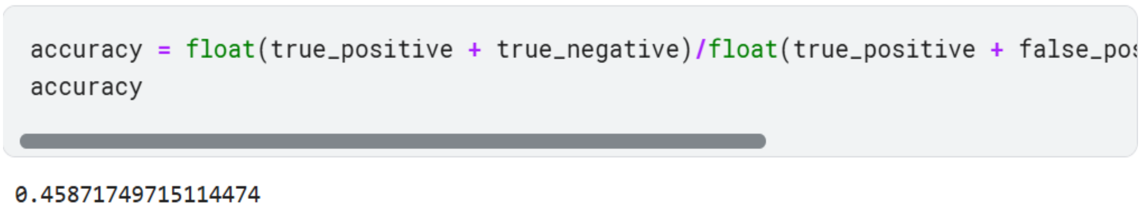
Figure 3.32: Logistic Regression with custom class weights Output prediction head.



```
# Print results
print(float(true_positive_rate))
print(float(false_positive_rate))
```

```
0.3955779136668976
0.15420129270544783
```

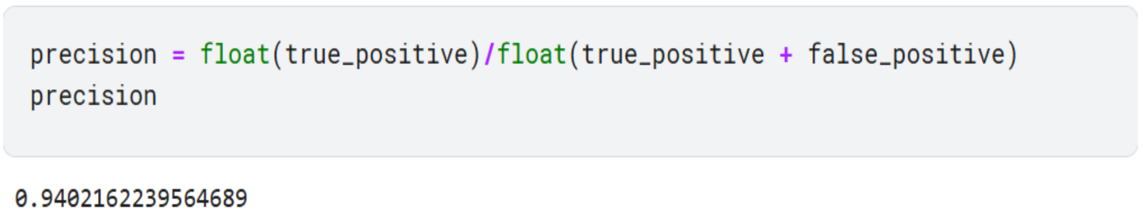
Figure 3.33: Logistic Regression with custom class true positive and false positive rate score.



```
accuracy = float(true_positive + true_negative)/float(true_positive + false_pos
accuracy
```

```
0.45871749715114474
```

Figure 3.34: Logistic Regression with custom class accuracy score.



```
precision = float(true_positive)/float(true_positive + false_positive)
precision
```

```
0.9402162239564689
```

Figure 3.35: Logistic Regression with custom class precision score.

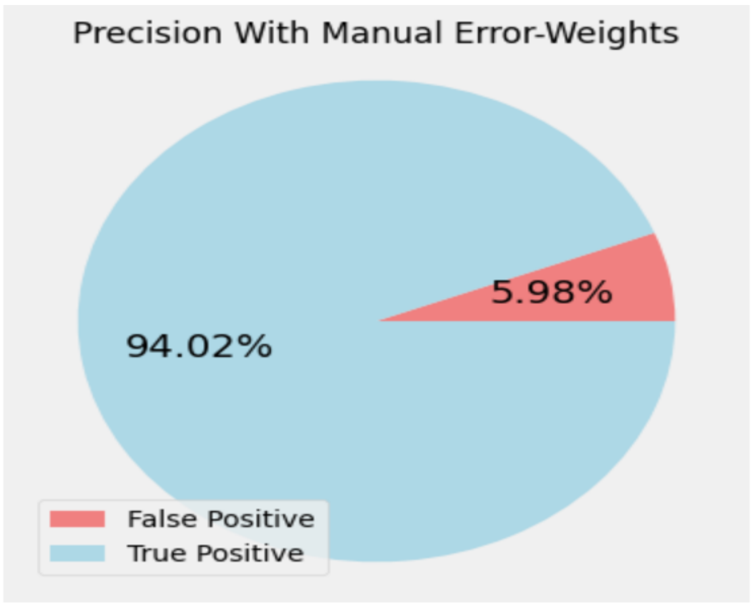
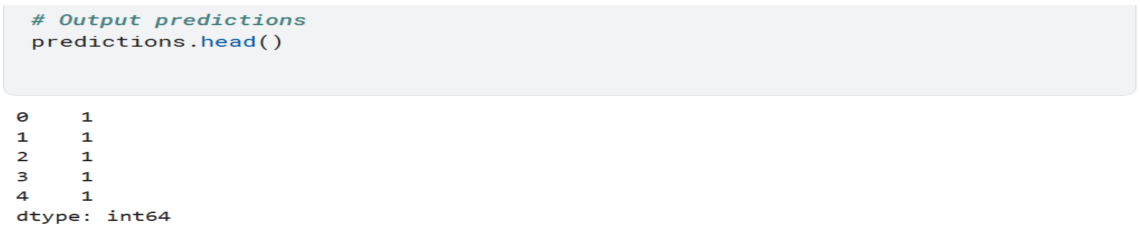


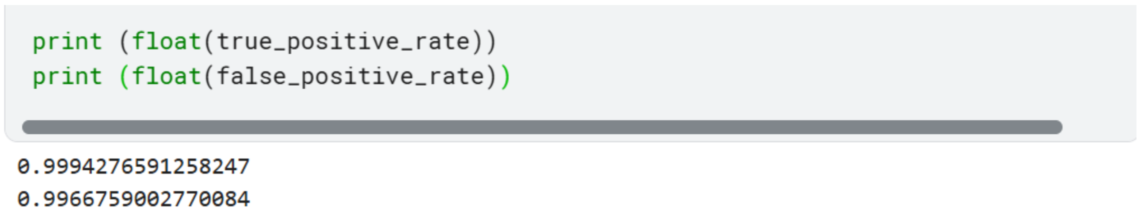
Figure 3.36: Show Pie chart representation in true positive and false positive rate in custom class logistic regression model.



```
# Output predictions
predictions.head()
```

```
0     1
1     1
2     1
3     1
4     1
dtype: int64
```

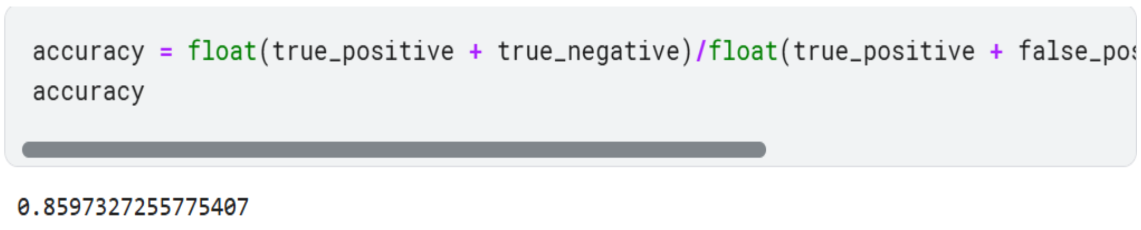
Figure 3.37: Random forest classifier prediction head.



```
print (float(true_positive_rate))
print (float(false_positive_rate))
```

```
0.9994276591258247
0.9966759002770084
```

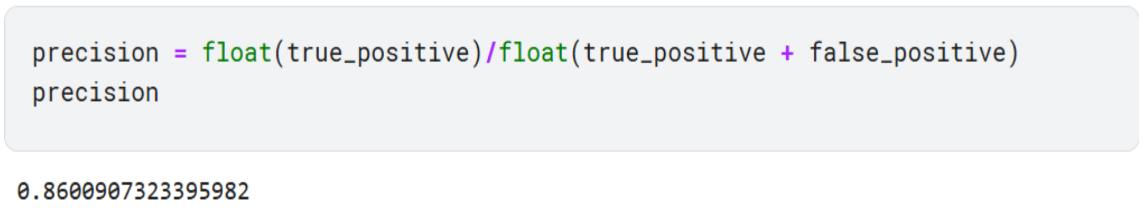
Figure 3.38: random forest classifier true positive and false positive rate score.



```
accuracy = float(true_positive + true_negative)/float(true_positive + false_pos
accuracy
```

```
0.8597327255775407
```

Figure 3.39: random forest classifier accuracy score.



```
precision = float(true_positive)/float(true_positive + false_positive)
precision
```

```
0.8600907323395982
```

Figure 3.40: random forest classifier precision score.

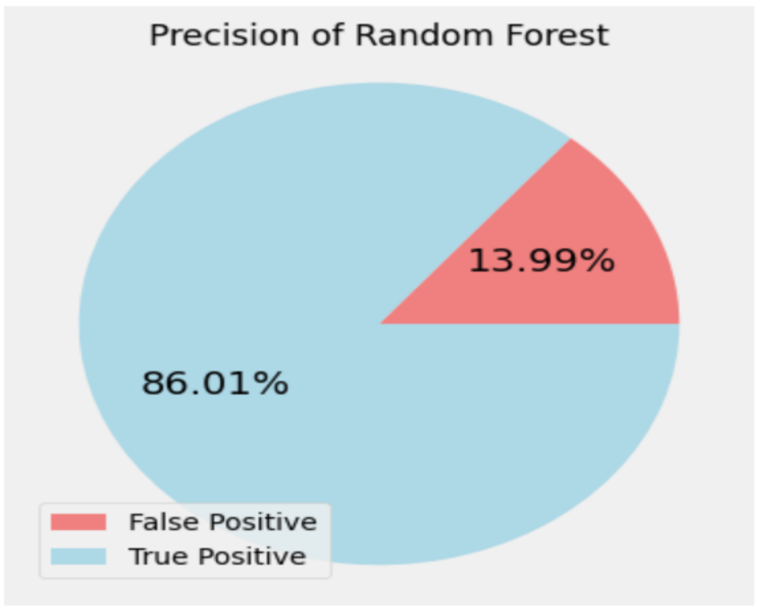


Figure 3.41: Show Pie chart representation in true positive and false positive rate in random forest classifier model.

3.2.2 Result

The results of the project underscore its practical utility. The final model achieved a high level of accuracy and recall, effectively identifying a significant proportion of loan defaulters. However, the discussion also highlights the trade-offs inherent in predictive modeling. While the model's sensitivity to true positives is commendable, its specificity could be further improved to reduce false positives. This balance is critical for practical deployment, where financial institutions must weigh the cost of incorrectly flagging a customer as a defaulter against the risk of approving a high-risk loan.

Chapter 4

Conclusion

4.1 Discussion

The project "Predicting Loan Default with Python" demonstrates the transformative potential of machine learning in addressing critical financial challenges. By leveraging data-driven techniques, the project provides a robust framework for identifying potential loan defaulters, which is a pivotal aspect of risk management in the financial sector. One of the key strengths of this project lies in its comprehensive data preprocessing. By addressing missing values, encoding categorical variables, and eliminating irrelevant features, the dataset was transformed into a clean and usable format suitable for machine learning. This step was crucial, as the quality of data directly influences the model's performance. Furthermore, the exploration of feature importance ensured that only the most relevant predictors were utilized, thereby enhancing the efficiency and interpretability of the final model. The methodology adopted for this project highlights the value of systematic experimentation. Multiple machine learning algorithms, including Logistic Regression, Decision Trees, and ensemble methods, were employed to identify the most effective model. The use of cross-validation ensured that the models were not overfitted to the training data, providing a realistic assessment of their performance on unseen data. Additionally, hyperparameter tuning further optimized the models, striking a balance between precision and recall, which is essential in applications where both false positives and false negatives carry significant consequences. A notable challenge encountered during the project was the inherent class imbalance in the dataset, as loan defaults are relatively rare compared to successful repayments. This imbalance posed a risk of bias in the predictive models, where the algorithms might favor the majority class. To address this, techniques such as oversampling and undersampling were employed, ensuring that the models could effectively learn from the minority class without being overwhelmed by the majority class.

4.2 Limitations

While the project "Predicting Loan Default with Python" demonstrates promising results, it is not without limitations. These challenges highlight areas for potential improvement and serve as considerations for real-world deployment.

1. **Dataset Limitations**
   The quality and scope of the dataset play a crucial role in the performance of predictive models. The dataset used in this project may not fully capture all factors influencing loan defaults. For example:

   - **Lack of Real-Time Data:** The dataset may represent historical data and fail to reflect real-time economic conditions, such as market trends or global financial crises, which could significantly impact loan repayment behaviors.
   - **Bias in Data Collection:** If the dataset disproportionately represents certain demographic groups or regions, the model's predictions may inherit and perpetuate these biases, leading to unfair outcomes.

2. **Class Imbalance**
   Loan defaults often represent a small percentage of the dataset, leading to class imbalance. While techniques such as oversampling and undersampling were applied, these methods can introduce challenges:

   - Oversampling may lead to overfitting, where the model performs well on training data but struggles to generalize to unseen data.
   - Undersampling may discard valuable information, reducing the diversity and richness of the training data.

3. **Model Generalization**
   Despite cross-validation and testing, the model may not generalize well to unseen data in entirely new contexts. For example:

   - Economic, political, or social changes could render the historical patterns learned by the model less relevant.
   - Variability in borrower behavior across different institutions or countries might limit the model's applicability beyond the original dataset.

4. **Interpretability Challenges**
   While ensemble methods like Random Forests or Gradient Boosting provide high accuracy, they can be less interpretable compared to simpler models like Logistic Regression. This lack of transparency might make it challenging for financial institutions to explain decisions to stakeholders or regulators.

5. **Computational Complexity**
   Advanced techniques, such as hyperparameter tuning and ensemble methods, increase computational demands. For smaller organizations or those with limited resources, these requirements may hinder the feasibility of implementing such a solution.

6. **Ethical Considerations**
   The use of sensitive financial and demographic data raises ethical concerns. Without proper anonymization and safeguards, there is a risk of violating privacy or inadvertently enabling discriminatory practices.

7. **Overfitting Risks**
   Despite efforts to mitigate overfitting through cross-validation and careful tuning,

there is always a risk that the model captures noise or irrelevant patterns in the training data, leading to suboptimal performance on new datasets.

4.3 Scope of Future Work

The project "Predicting Loan Default with Python" establishes a solid foundation for leveraging machine learning in financial risk management. However, there are several areas where future work could expand its scope, enhance its effectiveness, and address current limitations. Below are the potential directions for future research and development:

1. **Integration of Alternative Data Sources**
   Incorporating diverse data sources can enrich the predictive power of the model:
   - **Behavioral Data:** Include borrower behavior patterns, such as spending habits or payment histories across different platforms.
   - **Macroeconomic Indicators:** Integrate economic variables like inflation rates, unemployment rates, and GDP growth, which can influence loan repayment behavior.
   - **Unstructured Data:** Leverage unstructured data such as customer reviews, social media activity, or call center transcripts using natural language processing (NLP).
2. **Use of Advanced Machine Learning Techniques**
   While traditional machine learning methods performed well, exploring advanced techniques can further improve performance:
   - **Deep Learning Models:** Implement neural networks, especially recurrent neural networks (RNNs) or transformers, to capture complex patterns and temporal dependencies.
   - **Hybrid Models:** Combine multiple models to leverage the strengths of each, such as stacking ensemble methods or integrating deep learning with traditional algorithms.
   - **Explainable AI (XAI):** Adopt frameworks that enhance model interpretability, allowing stakeholders to understand and trust predictions.
3. **Real-Time Prediction Systems**
   Building a real-time predictive system can increase the model's utility:
   - **Streaming Data:** Implement tools like Apache Kafka or Spark Streaming to process live data and provide instantaneous predictions.
   - **Dynamic Model Updates:** Develop mechanisms for continuous model retraining and updating based on new data, ensuring relevance and accuracy.
4. **Geographic and Demographic Generalization**
   Expanding the dataset to include data from different regions and demographic groups can enhance the model's applicability:

- **Cross-Cultural Adaptability:** Train and test the model on datasets from multiple countries or regions to ensure global relevance.
- **Demographic Analysis:** Include variables that capture socio-economic diversity, allowing for a more nuanced understanding of loan default risks.

5. **Enhanced Visualization and Reporting**
   Future work can include tools for better stakeholder communication:
   - **Interactive Dashboards:** Build user-friendly dashboards for lenders to explore predictions and model insights dynamically.
   - **Scenario Analysis:** Incorporate tools that allow stakeholders to simulate "what-if" scenarios, such as changes in economic conditions or borrower attributes.
6. **Longitudinal Analysis**
   Introduce models that account for temporal trends and changes over time:
   - **Time Series Models:** Use models like Long Short-Term Memory (LSTM) networks or ARIMA to predict default probabilities over time.
   - **Lifecycle Analysis:** Analyze borrower behavior at different stages of the loan lifecycle to refine predictions.
7. **Collaboration with Financial Institutions**
   Future efforts could involve partnerships with financial institutions to test the model in real-world settings:
   - **Pilot Programs:** Deploy the model in a controlled environment to evaluate its performance on live data.
   - **Feedback Integration:** Use feedback from loan officers and financial analysts to refine the model and its usability.

References

[1] U. Aslam, H. I. Tariq Aziz, A. Sohail, and N. K. Batcha, "An empirical study on loan default prediction models," *Journal of Computational and Theoretical Nanoscience*, vol. 16, no. 8, pp. 3483–3488, 2019.

[2] J. Lin, "Research on loan default prediction based on logistic regression, randomforest, xgboost and adaboost," in *SHS Web of Conferences*, vol. 181. EDP Sciences, 2024, p. 02008.

[3] H. Arian, S. M. S. Seyfi, and A. Sharifi, "Forecasting probability of default for consumer loan management with gaussian mixture models," *arXiv preprint arXiv:2011.07906*, 2020.

[1] [2] [3]