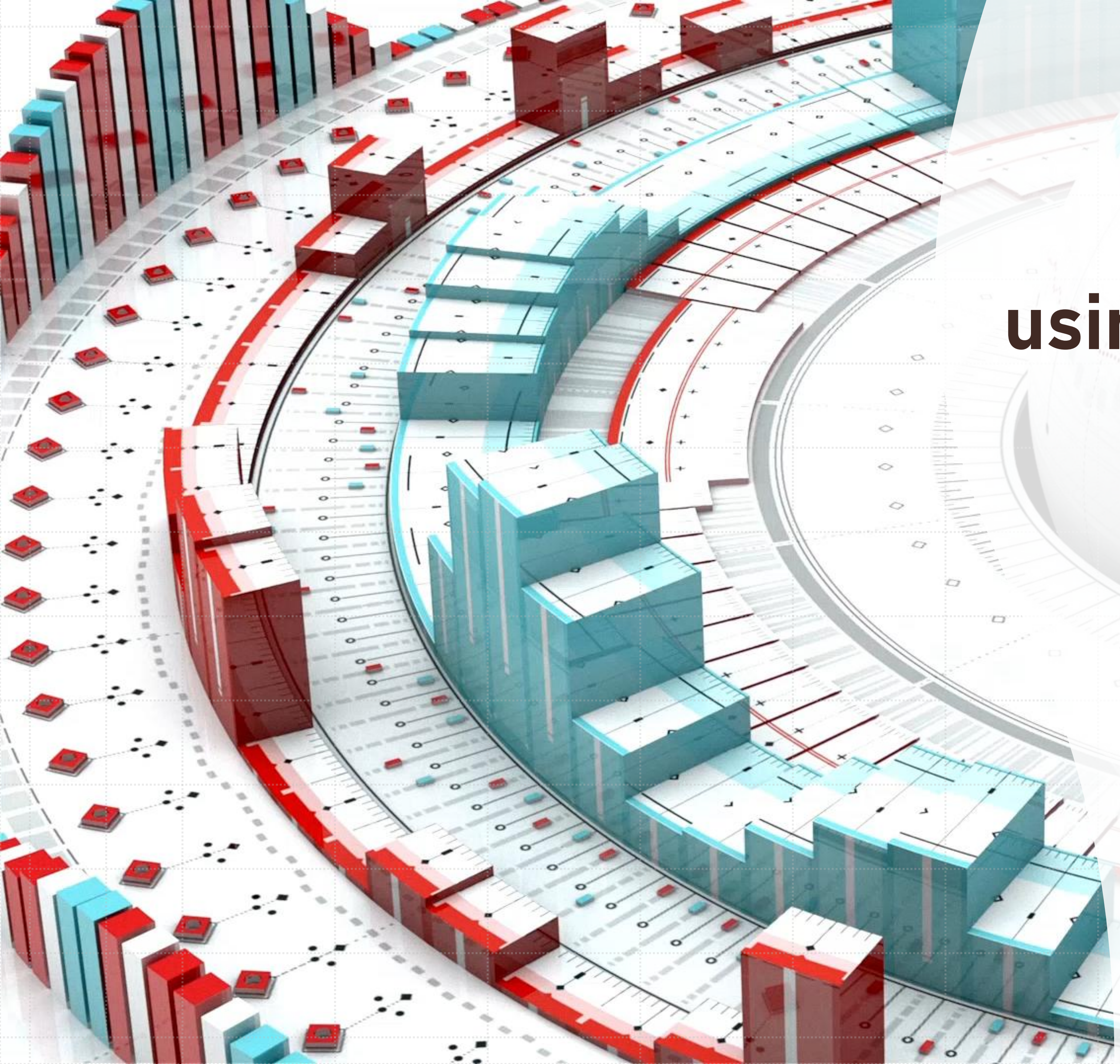


Loan Approval Prediction using Machine Learning

Data System Architecture

Project done by-

Rimi Mondal

INTRODUCTION

BACKGROUND

We developed a machine learning model to predict creditworthiness based on demographic and financial factors.

The dataset contains information on 614 loan applicants and was obtained from Kaggle.

Our project includes data cleaning, exploratory data analysis, and modeling techniques to create the final machine learning model.

The project's audiences include finance and banking professionals, data scientists and researchers, students and educators in data science and finance, and individuals interested in personal finance and loan eligibility.

DATA DESCRIPTION

The dataset was obtained from Kaggle and contains 13 variables, 12 independent and 1 dependent.

The dataset used in this project is called "Loan Prediction" and was uploaded to Kaggle by user Vikas U Kani.

The dataset includes information on 614 loan applicants, each represented by a row in the dataset.

INDEPENDENT VARIABLES

1.Gender,
2.Married,
3.Dependents,
4.Education,
5.Self_Employed,
6.ApplicantIncome,
7.CoapplicantIncome,
8.LoanAmount,
9.Loan_Amount_Term,
10.Credit_History,
11.Property_Area, and
12.Total_Income

DEPENDENT VARIABLE

1.Loan_Status

- It indicates whether a loan application was approved (labeled as 'Y') or not approved (labeled as 'N')

- The data cleaning and preprocessing steps involved identifying and handling missing values in the dataset.
- The following variables had missing values:
  - Gender,
  - Married,
  - Dependents,
  - Self-Employed,
  - LoanAmount,
  - Loan_Amount_Term, and
  - Credit_History.
- Missing categorical values were imputed using the mode, and missing numerical values were imputed using the median.

WORKFLOW

PROCESS

DATA COLLECTION

- Gather the dataset from a reliable source, such as Kaggle.

DATA PREPROCESSING

- Preprocess the data by removing duplicates and missing values, converting categorical data into numerical data, and standardizing numerical data.

DATA ANALYSIS

- Conduct exploratory data analysis to identify relationships between the variables and the loan status. Visualize the data using plots and charts to gain insights into the data.

FEATURE ENGINEERING

- Create new features from existing data that may be more predictive of the loan status.

MODEL SELECTION

- Select an appropriate model for the task of predicting loan eligibility. Consider using classification models like logistic regression, decision trees, or random forests.

MODEL TRAINING

- Train the selected model on the preprocessed data using appropriate training techniques.

MODEL EVALUATION

- Evaluate the performance of the model using evaluation metrics such as accuracy, precision, recall, and F1 score.

HYPERPARAMETER TUNING

- Tune the hyperparameters of the model using techniques like GridSearchCV to improve its performance.

MODEL DEPLOYMENT

- Deploy the trained and optimized model in a production environment where it can make accurate loan eligibility predictions.

MONITORING AND MAINTENANCE

- Monitor the model's performance in the production environment, retrain it periodically, and update it as necessary to maintain its accuracy and relevance.

DATA CLEANING AND PREPROCESSING

DETAILED STEPS

- The first step was importing necessary libraries for data analysis and machine learning, such as pandas, numpy, scikit-learn, and matplotlib.
- Null values, duplicates, and NAN values were dropped from the dataset to ensure clean and accurate data.
- The data was transposed to make it easier to work with and analyze.
- NAN values were checked and replaced if necessary, using methods like mean or median.
- Inconsistent values were identified by looking at unique values in each column and ensuring they were within the expected range.
- Outliers were removed from the test data using methods like z-score and the IQR method to ensure the accuracy of the model.
- Outliers were also removed from the train data using the same methods as for the test data.
- Once the data was cleaned, it was exported to a CSV file for further analysis.
- A random forest algorithm was used to build a machine learning model that could predict outcomes based on the data.
- The confusion matrix and error rate were calculated to evaluate the accuracy of the model and identify areas for improvement.

STEP 1 TO 6

Importing Libraries

In [88]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#import math
import plotly.express as px
from IPython.display import Image
```

In [2]:
```
#importing test csv
l_test = pd.read_csv('loan-test.csv')
```

In [59]:
```
#checking 5 rows from first
l_test.head()
```

Out[59]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Hist
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	360.0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	360.0	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	360.0	
3	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	360.0	
4	LP001054	Male	Yes	0	Not Graduate	Yes	2165	3422	152.0	360.0	

In [4]:
```
#importing train csv
l_train = pd.read_csv('loan-train.csv')
```

Cleaning Outliers from test data

Box plot

In [27]:
```
# Define the columns to be analyzed for outliers
columns = ['ApplicantIncome', 'LoanAmount']

# Create boxplots to visualize outliers in each column
for column in columns:
    plt.figure()
    plt.boxplot(l_test[column])
    plt.title(f"{column} Boxplot")
    plt.show()
```

Dropping null values, duplicates,NAN

In [8]:
```
l_test.dropna(inplace=True)
l_train.dropna(inplace=True)
```

In [9]:
```
print(l_test.shape)
print(l_train.shape)
```

```
(289, 12)
(480, 13)
```

In [10]:
```
print(l_test.info())
l_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 289 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            289 non-null    object
 1   Gender             289 non-null    object
 2   Married            289 non-null    object
 3   Dependents         289 non-null    object
 4   Education          289 non-null    object
 5   Self_Employed      289 non-null    object
 6   ApplicantIncome    289 non-null    int64
 7   CoapplicantIncome  289 non-null    int64
 8   LoanAmount         289 non-null    float64
 9   Loan_Amount_Term   289 non-null    float64
 10  Credit_History     289 non-null    float64
 11  Property_Area      289 non-null    object
```

STEP 7 TO 9

Using Z-score method

In [28]:
```
# Define the columns to be analyzed for outliers
columns = ['ApplicantIncome', 'LoanAmount']

# Remove outliers from each column using z-score method
for column in columns:
    col_mean = l_test[column].mean()
    col_std = l_test[column].std()
    l_test = l_test[(l_test[column] >= col_mean - 3*col_std) & (l_test[column] <= col_mean + 3*col_std)]

# Reset the index of the cleaned dataframe
l_test = l_test.reset_index(drop=True)
```

In [29]:
```
# # Define the columns to be analyzed for outliers
# columns = ['ApplicantIncome', 'LoanAmount']

# # Create boxplots to visualize outliers in each column
# for column in columns:
#     plt.figure()
#     plt.boxplot(l_test[column])
#     plt.title(f"{column} Boxplot")
#     plt.show()
```

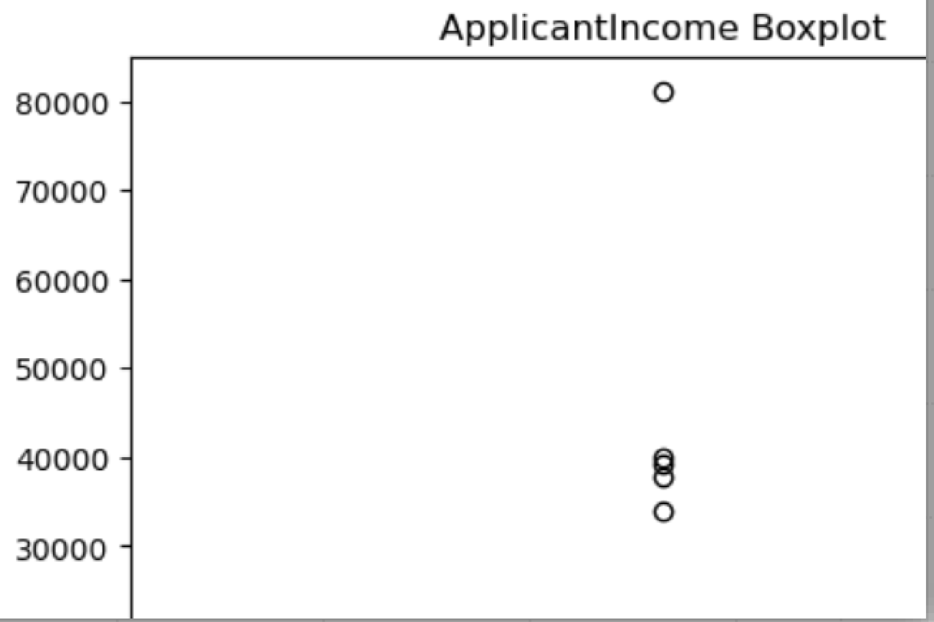
Removing Outliers using IQR method

In [31]:
```
# Define the columns to be analyzed for outliers
columns = ['ApplicantIncome', 'LoanAmount']

# Remove outliers from each column using IQR method
for column in columns:
    Q1 = l_test[column].quantile(0.25)
    Q3 = l_test[column].quantile(0.75)
    IQR = Q3 - Q1
    l_test = l_test[(l_test[column] >= Q1 - 1.5*IQR) & (l_test[column] <= Q3 + 1.5*IQR)]

# Reset the index of the cleaned dataframe
l_test = l_test.reset_index(drop=True)
```

In [32]:
```
# Create boxplots to visualize outliers in each column
for column in columns:
    plt.figure()
    plt.boxplot(l_test[column])
    plt.title(f"{column} Boxplot")
    plt.show()
```

Cleaning Outliers from train_data

In [34]:
```
# Define the columns to be analyzed for outliers
columns = ['ApplicantIncome', 'LoanAmount']

# Create boxplots to visualize outliers in each column
for column in columns:
    plt.figure()
    plt.boxplot(l_train[column])
    plt.title(f"{column} Boxplot")
    plt.show()
```



STEP 10 TO 12

Data to csv

In [38]:
```
#l_test.to_csv("updated_test.csv")
```

In [39]:
```
#l_train.to_csv("updated_train.csv")
```

In [43]:
```
l_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            263 non-null    object 
 1   Gender             263 non-null    object 
 2   Married            263 non-null    object 
 3   Dependents         263 non-null    object 
 4   Education          263 non-null    object 
 5   Self_Employed      263 non-null    object 
 6   ApplicantIncome    263 non-null    int64  
 7   CoapplicantIncome  263 non-null    int64  
 8   LoanAmount         263 non-null    float64
 9   Loan_Amount_Term   263 non-null    float64
 10  Credit_History     263 non-null    float64
 11  Property_Area      263 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 24.8+ KB
```

Using Random Forest

In [80]:
```
# importing rfc, trainingand test dataset with validation metrics too
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score
```

In [81]:
```
# Split the dataset into independent and dependent variables
X = l_train.drop(['Loan_ID', 'Loan_Status'], axis=1)
y = l_train['Loan_Status'].apply(lambda x: 1 if x=='Y' else 0)
```

In [82]:
```
# Convert categorical variables into dummy variables
X = pd.get_dummies(X)
```

In [83]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

test size is only 20%

In [84]:
```
# Create a Random Forest classifier with default hyperparameters
rfc = RandomForestClassifier()

# Fit the model to the training data
rfc.fit(X_train, y_train)

# Use the model to predict the values of the target variable
```

MODELLING

- Random forest algorithm was used with a test size of only 20%.
- The efficiency of the model was 74%, while accuracy and recall were also 74% and 89%, respectively.
- Both training and test datasets were the same.
- Other classification techniques such as logistic regression and decision trees were employed.
- GridSearchCV's adjustments were made to hyperparameters.
- Evaluation metrics such as F1 score, recall, accuracy, and precision were used.
- Random forest had the highest F1 score, accuracy, and precision.
- Confusion matrix showed 30 true positives and 54 true negatives with an error rate of 0.0.

PREDICTION

RAPID MINER

ExampleSet (Apply Feature Set on Complete Training)

Result History | ExampleSet (Log to Data)

Data | Statistics | Visualizations | Annotations

Open in Turbo Prep Auto Model

Row No.	Number of T...	Maximal De...	Learning Rate	Error Rate
1	30	2	0.001	0.368
2	90	2	0.001	0.368
3	150	2	0.001	0.368
4	30	4	0.001	0.368
5	90	4	0.001	0.368
6	150	4	0.001	0.305
7	30	7	0.001	0.368
8	90	7	0.001	0.368
9	150	7	0.001	0.305
10	30	2	0.010	0.305
11	90	2	0.010	0.358
12	150	2	0.010	0.358
13	30	4	0.010	0.305
14	90	4	0.010	0.316
15	150	4	0.010	0.337
16	30	7	0.010	0.305
17	90	7	0.010	0.316

ExampleSet (27 examples, 0 special attributes, 4 regular attributes)

DATA RESULTS – LOG DATA

Result History | ExampleSet (Apply Model) | ExampleSet (//Local Repository/updated_train)

Data | Statistics | Visualizations | Annotations

Open in Turbo Prep Auto Model

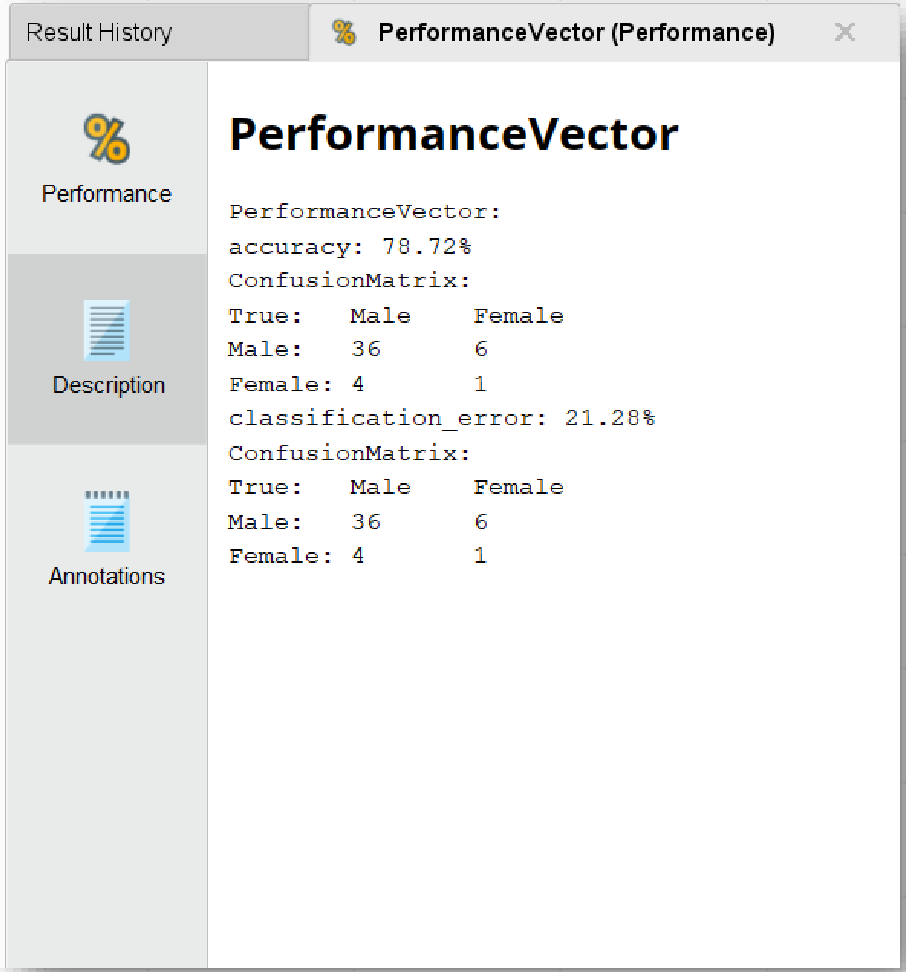
Filter (47 / 47 examples): all

Row No.	ApplicantInc...	prediction(A...	att1	Loan_ID	Gender	Married	Dependents	Education	Self_Employ...
1	3000	2866.500	1	LP001005	Male	Yes	0	Graduate	Yes
2	6000	4277.250	3	LP001008	Male	No	0	Graduate	No
3	2333	2012.750	4	LP001013	Male	Yes	0	Not Graduate	No
4	4006	3863	6	LP001018	Male	Yes	2	Graduate	No
5	3073	4716	8	LP001028	Male	Yes	2	Graduate	No
6	4950	4277.250	10	LP001032	Male	No	0	Graduate	No
7	3510	3337	11	LP001036	Female	No	0	Graduate	No
8	4887	2765.500	12	LP001038	Male	Yes	0	Not Graduate	No
9	2600	1426.500	14	LP001047	Male	Yes	0	Not Graduate	No
10	4692	4760	19	LP001097	Male	No	1	Graduate	Yes
11	3667	3614.600	21	LP001112	Female	Yes	0	Graduate	No
12	1800	2012.750	25	LP001120	Male	No	0	Graduate	No
13	5821	4277.250	28	LP001144	Male	Yes	0	Graduate	No
14	4000	2169	30	LP001151	Female	No	0	Graduate	No
15	3086	4127	32	LP001157	Female	No	0	Graduate	No
16	2708	2878	35	LP001194	Male	Yes	2	Graduate	No
17	8080	4716	38	LP001198	Male	Yes	1	Graduate	No

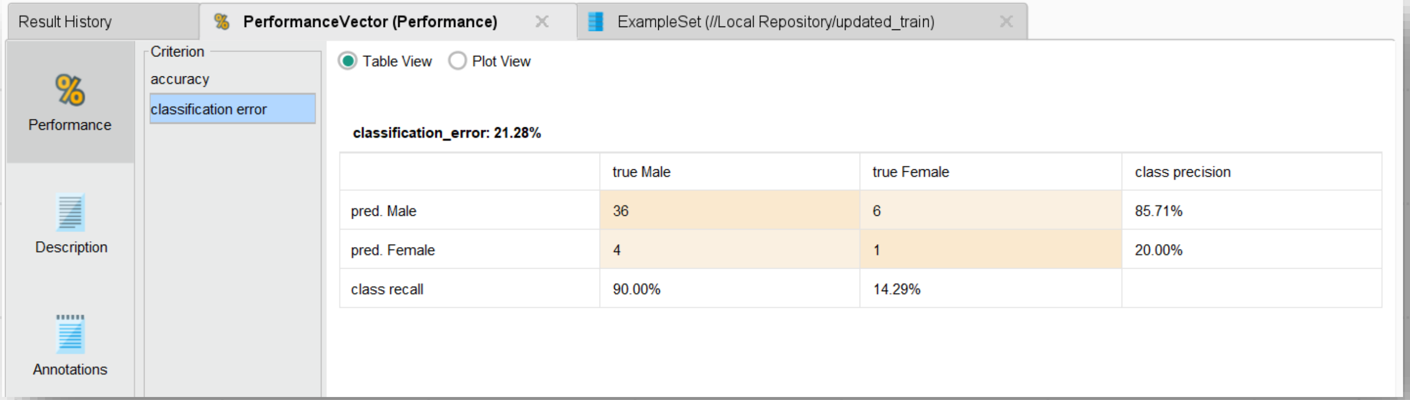
ExampleSet (47 examples, 2 special attributes, 13 regular attributes)

DATA RESULTS – APPLY MODEL

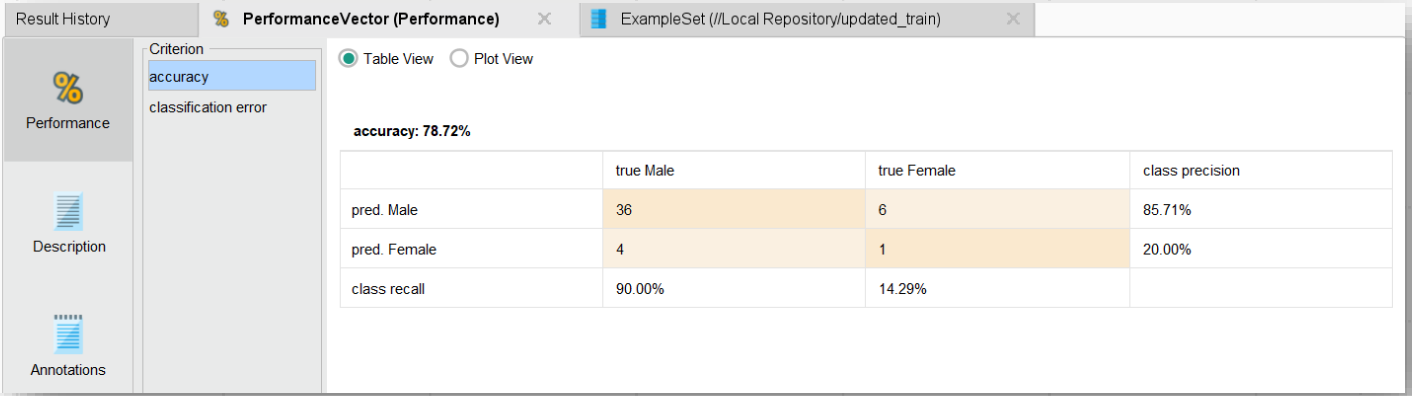
RAPID MINER



PERFORMANCE VECTOR

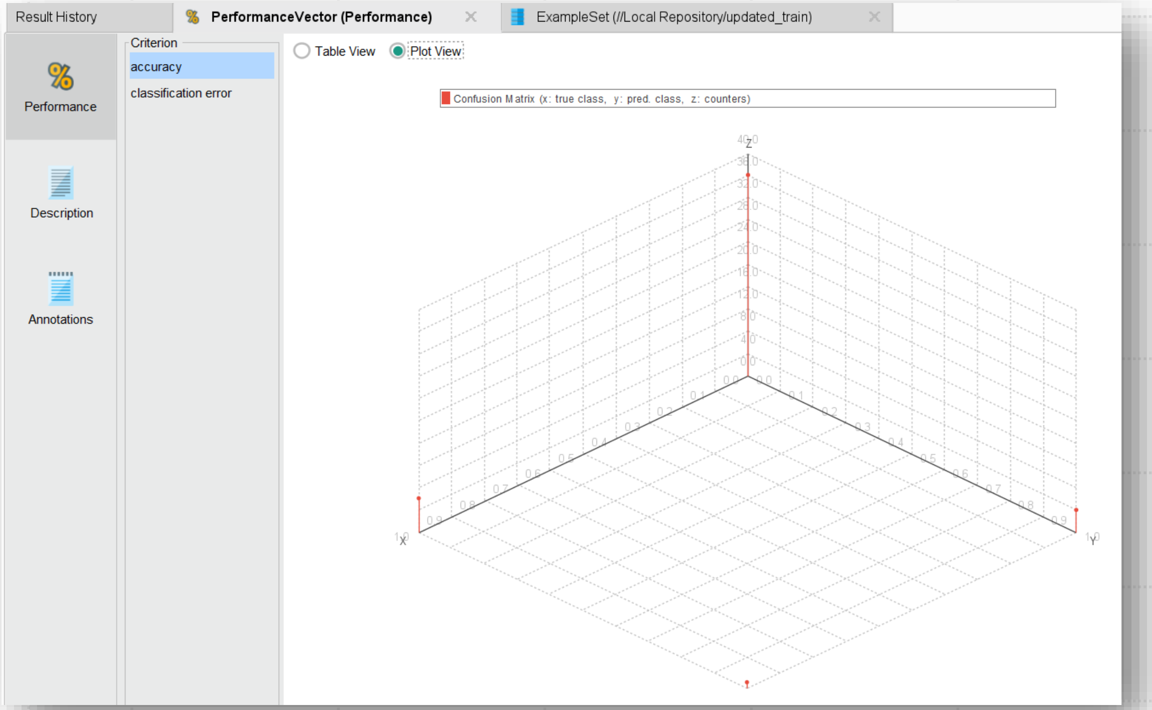


CLASSIFICATION ERROR

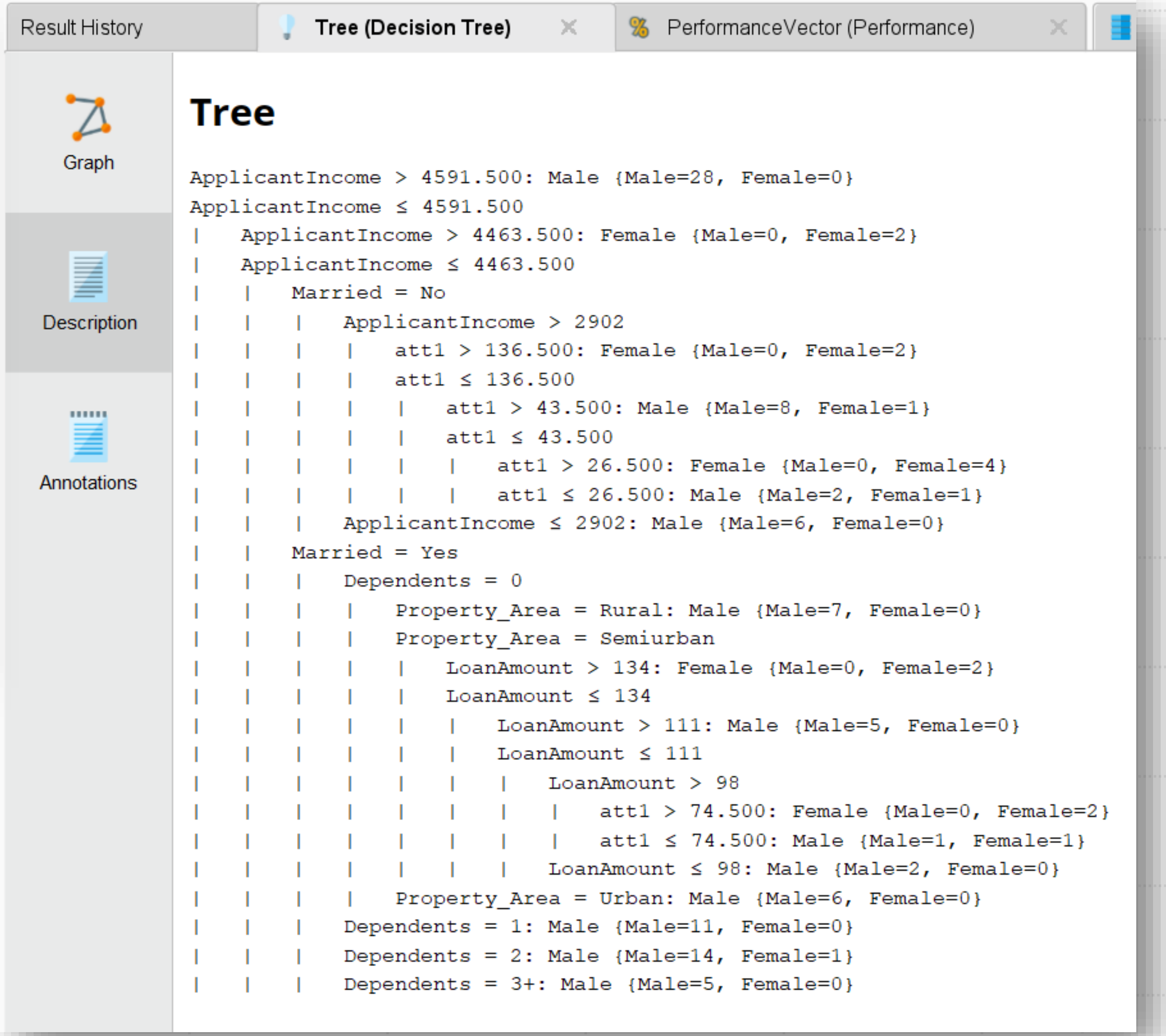


ACCURACY

RAPID MINER

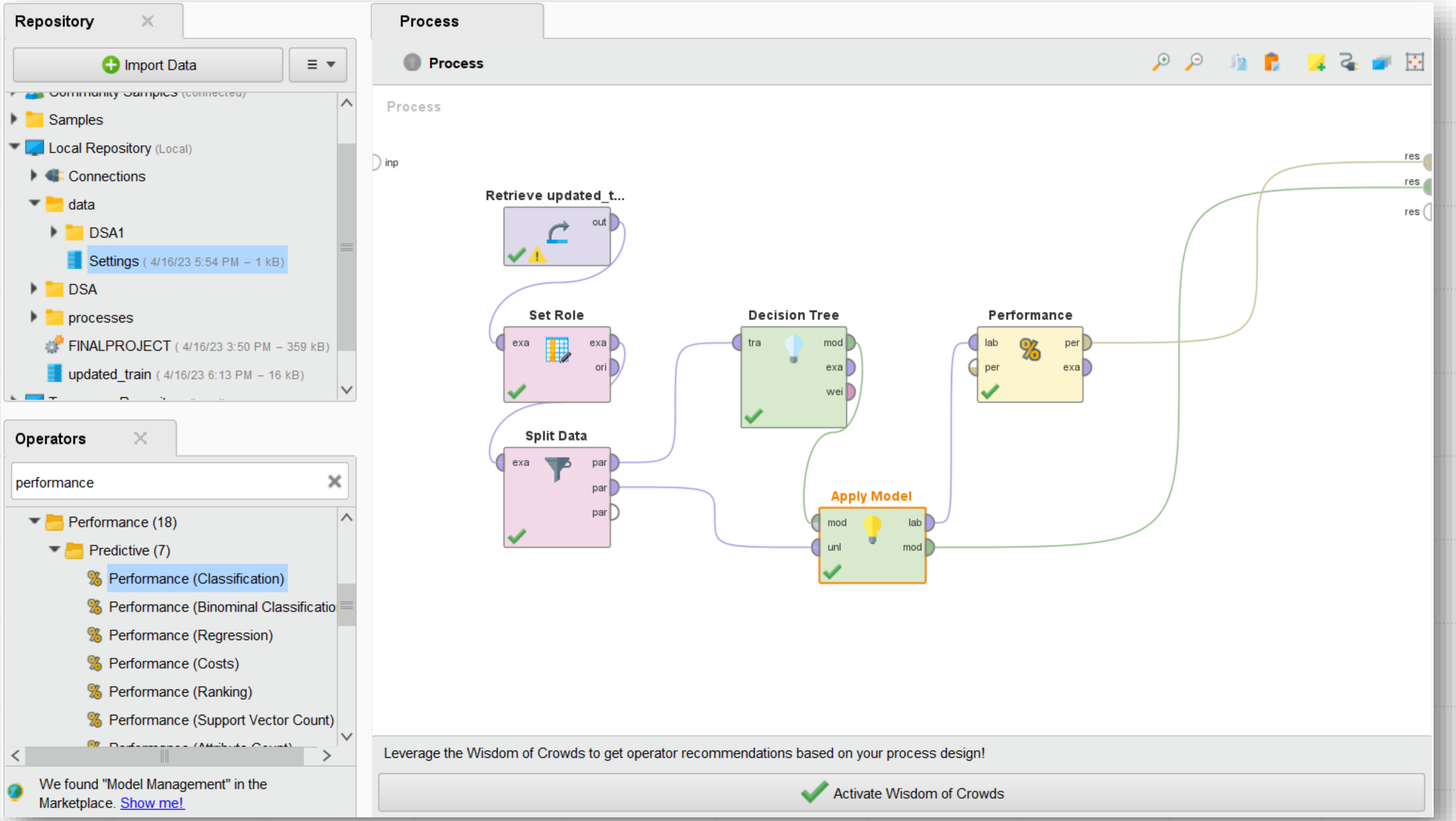


PLOT VIEW OF
PERFORMANCE ACCURACY



TREE DESCRIPTION

RAPID MINER



PROCESS MODEL

VISUALIZATION

EXPLORATORY DATA ANALYSIS

The data set contains information about customers, including their demographics and transaction history.

The data set is relatively large, with several variables that need to be explored.

The variables include both categorical and numerical data, which require different types of analysis.

The first step in EDA is to examine the distribution of the variables and identify any outliers or missing values.

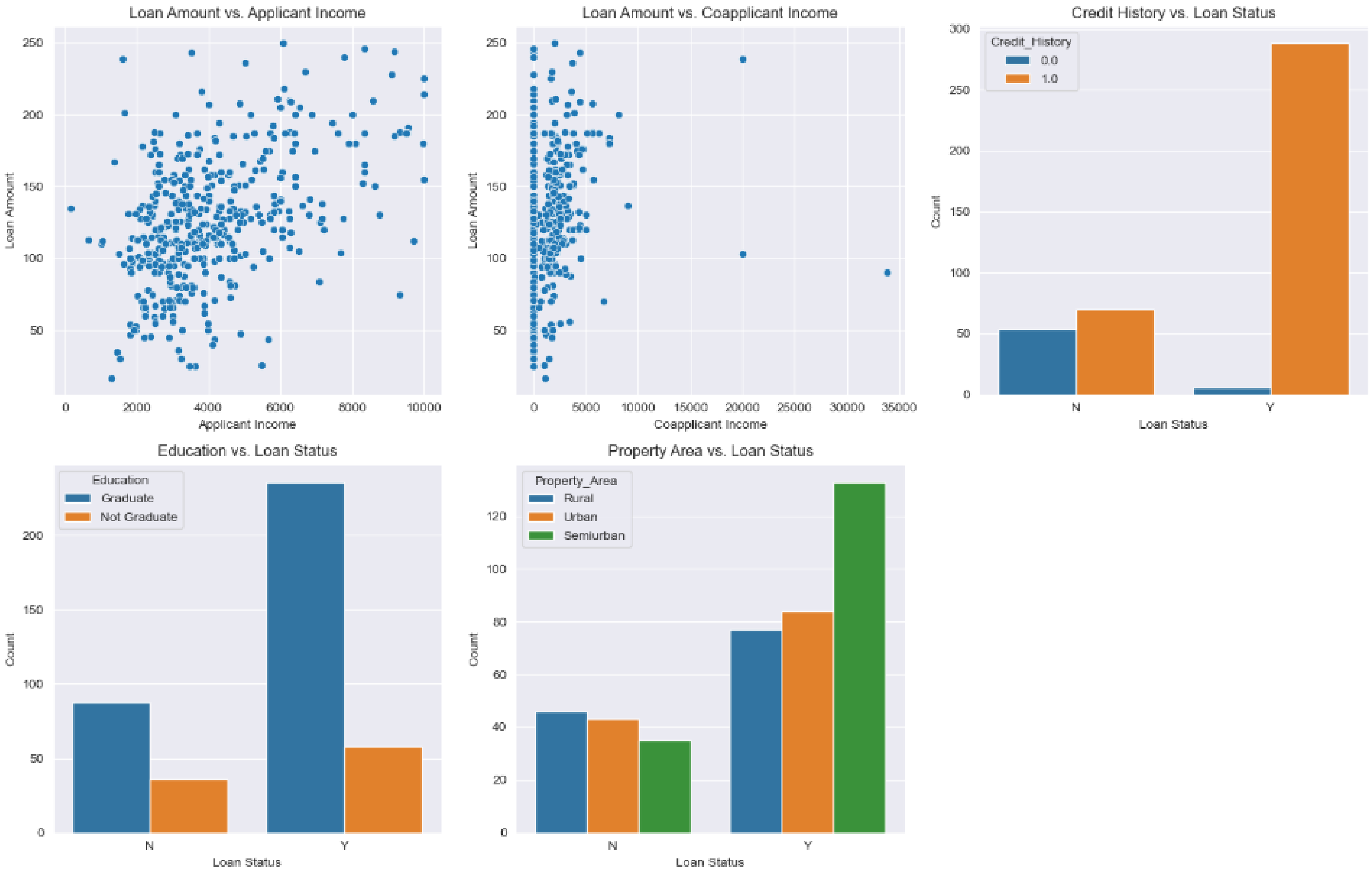
Histograms, box plots, and scatterplots are useful tools for visualizing the data and identifying patterns.

Correlation analysis can be used to identify relationships between variables.

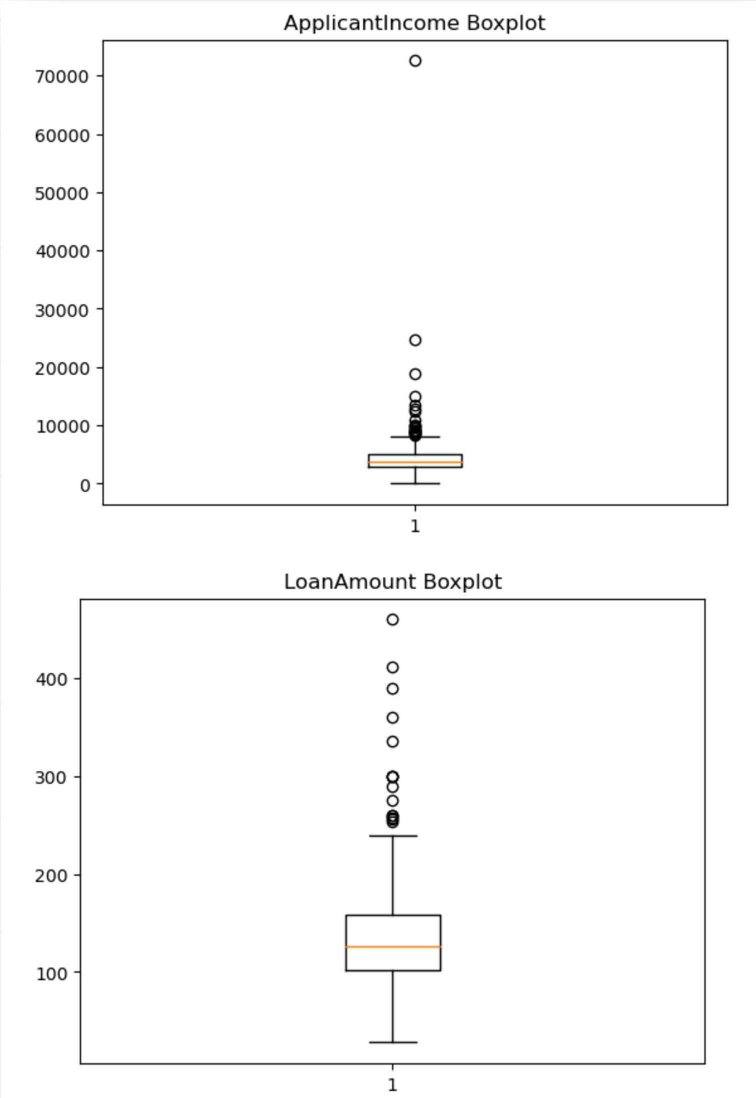
Grouping the data by different variables, such as age or gender, can reveal insights into customer behavior.

EDA can be used to identify variables that are most strongly correlated with the outcome variable and can be used for feature selection in machine learning models.

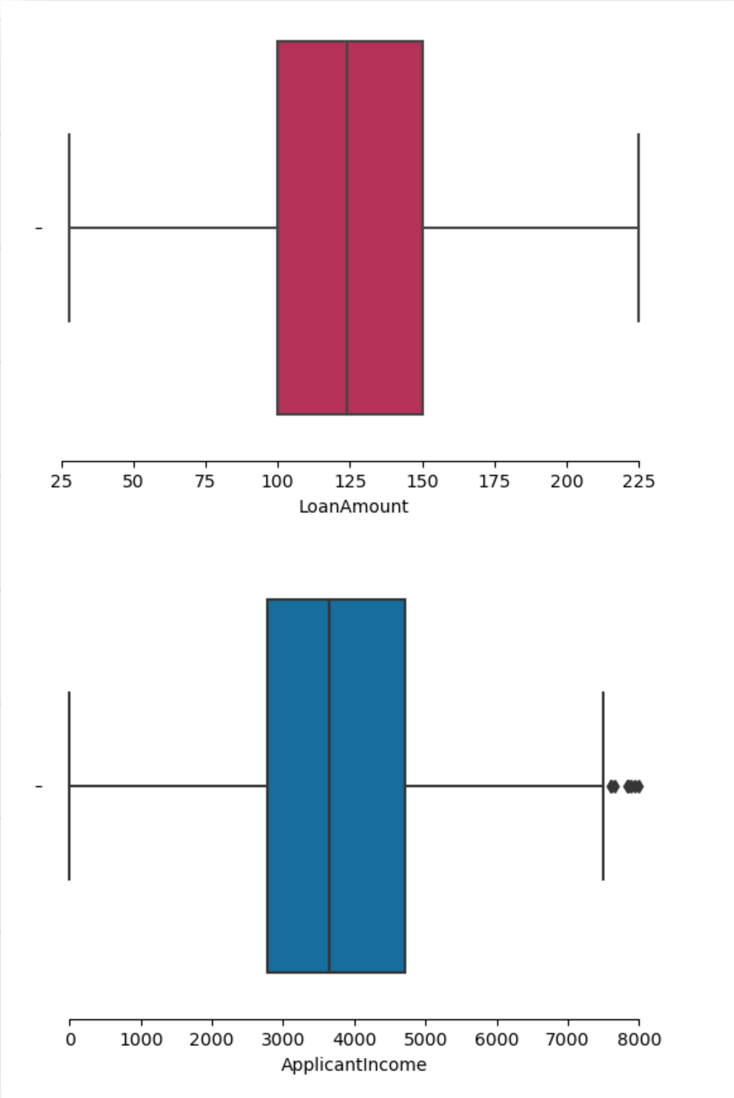
RELATIONSHIP CHARTS



BOXPLOT



BEFORE CLEANING



AFTER CLEANING

CONCLUSION

The Loan data initially had 614 rows, but after cleaning, removing duplicates, null values, characters, and outliers, it reduced to around 367.

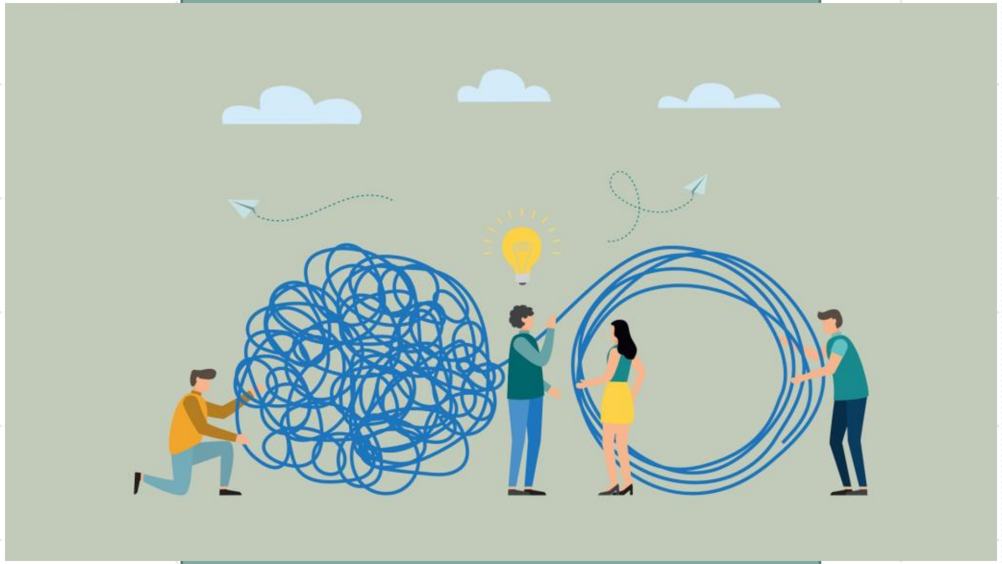
The accuracy of the model was 74%, with an error percentage of around 26%, indicating a probability of 1/4th error.

The True Positive rate or Recall rate was only 89%.

The main limitation of the data was its illogical nature in giving loans to individuals, with loans given to everyone regardless of their income.

Although the random forest model showed good performance, further exploration of other models and feature engineering techniques could potentially improve the accuracy and precision of the model.

- Additionally, it is important to note that the data used in this project was limited in scope and may not be representative of larger datasets or different populations. Further research and data collection may be necessary to improve the generalizability of the model.



THANK YOU