**Loan Approval Prediction Project**

Fatima Gul

08-july-2024

**Introduction**

**Objective**

The objective of this project is to predict loan approval status using a dataset by applying various data science techniques including data cleaning, exploratory data analysis (EDA), outlier detection, and machine learning model training and evaluation.

**Data Cleaning**

**Loading the Dataset**

The dataset is loaded into a Pandas DataFrame for analysis.

import pandas as pd  
  
# Load the dataset  
file\_path = 'loan\_approval\_dataset.csv'  
data = pd.read\_csv(file\_path)

### Handling Missing Values

Missing values in numerical columns are filled using mean imputation, while categorical columns are filled using mode imputation.

# Check for missing values  
print("Missing values before cleaning:\n", data.isnull().sum())  
  
# Fill missing values for numerical columns  
numeric\_cols = data.select\_dtypes(include=['number']).columns  
data[numeric\_cols] = data[numeric\_cols].fillna(data[numeric\_cols].mean())  
  
# Fill missing values for categorical columns  
categorical\_cols = data.select\_dtypes(include=['object']).columns  
data[categorical\_cols] = data[categorical\_cols].fillna(data[categorical\_cols].mode().iloc[0])  
  
print("Missing values after cleaning:\n", data.isnull().sum())

**Results**: Before and after counts of missing values.

# Include output from the above print statements

### Correcting Data Types and Standardizing Formats

Data types are corrected and formats are standardized as needed (e.g., converting date columns to datetime format).

# Example of correcting data types  
# data['column\_name'] = pd.to\_datetime(data['column\_name'])

### Addressing Inconsistent or Erroneous Data Entries

Inconsistent or erroneous data entries are identified and corrected.

# Example of standardizing string data  
# data['string\_column'] = data['string\_column'].str.lower()

## Exploratory Data Analysis (EDA)

### Descriptive Statistics

Descriptive statistics provide a summary of the dataset

print("Descriptive statistics:\n", data.describe())

### Visualizations

#### Histograms

Histograms show the distribution of numerical features.

data.hist(bins=30, figsize=(15, 10))  
plt.show()

#### Correlation Matrix

The correlation matrix helps understand relationships between numerical variables.

corr = data[numeric\_cols].corr()  
plt.figure(figsize=(10, 8))  
sns.heatmap(corr, annot=True, cmap='coolwarm')  
plt.show()

#### Count Plots for Categorical Variables

Count plots visualize the distribution of categorical features.

for col in categorical\_cols:  
 plt.figure(figsize=(10, 6))  
 sns.countplot(data[col])  
 plt.title(f'Count plot for {col}')  
 plt.show()

#### Box Plots for Numerical Variables

Box plots identify outliers and show the spread of numerical features.

for col in numeric\_cols:  
 plt.figure(figsize=(10, 6))  
 sns.boxplot(data[col])  
 plt.title(f'Box plot for {col}')  
 plt.show()

#### Pair Plots for Numerical Variables

Pair plots visualize relationships between numerical features.

sns.pairplot(data[numeric\_cols])  
plt.show()

## Outlier Detection and Removal

### Methods Used

Outliers are detected using Z-score and IQR methods.

from scipy import stats  
import numpy as np  
  
# Z-score method  
z\_scores = np.abs(stats.zscore(data[numeric\_cols]))  
filtered\_entries = (z\_scores < 3).all(axis=1)  
data = data[filtered\_entries]  
  
# IQR method  
Q1 = data[numeric\_cols].quantile(0.25)  
Q3 = data[numeric\_cols].quantile(0.75)  
IQR = Q3 - Q1  
data = data[~((data[numeric\_cols] < (Q1 - 1.5 \* IQR)) | (data[numeric\_cols] > (Q3 + 1.5 \* IQR))).any(axis=1)]

### Results

The impact on the dataset is described by comparing the number of rows before and after outlier removal.

print(data.shape)

**Results**:

# Include output from the above print statement

## Model Training and Evaluation

### Data Splitting

The data is split into training and testing sets.

from sklearn.model\_selection import train\_test\_split  
  
# Ensure 'Loan\_Status' column is present in the DataFrame  
if 'Loan\_Status' in data.columns:  
 X = data.drop('Loan\_Status', axis=1)  
 y = data['Loan\_Status']  
else:  
 print("Error: 'Loan\_Status' column not found in the DataFrame")  
  
# Split the dataset  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

### Training the KNN Model

from sklearn.neighbors import KNeighborsClassifier  
  
knn = KNeighborsClassifier(n\_neighbors=5)  
knn.fit(X\_train, y\_train)

### Cross-Validation

Cross-validation results are obtained.

from sklearn.model\_selection import cross\_val\_score  
  
cv\_scores = cross\_val\_score(knn, X, y, cv=5)  
print(f'Cross-validation scores: {cv\_scores}')  
print(f'Mean cross-validation score: {cv\_scores.mean()}')

**Results**: Provide the cross-validation scores and the mean score.

### Model Evaluation

The KNN model is evaluated using various metrics.

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score, confusion\_matrix, roc\_curve  
  
y\_pred = knn.predict(X\_test)  
  
accuracy = accuracy\_score(y\_test, y\_pred)  
precision = precision\_score(y\_test, y\_pred, pos\_label='Y')  
recall = recall\_score(y\_test, y\_pred, pos\_label='Y')  
f1 = f1\_score(y\_test, y\_pred, pos\_label='Y')  
roc\_auc = roc\_auc\_score(y\_test, knn.predict\_proba(X\_test)[:, 1])  
  
print(f'Accuracy: {accuracy}')  
print(f'Precision: {precision}')  
print(f'Recall: {recall}')  
print(f'F1-score: {f1}')  
print(f'ROC-AUC: {roc\_auc}')

**Results**: Include the accuracy, precision, recall, F1-score, and ROC-AUC.

#### Confusion Matrix

The confusion matrix is visualized.

cm = confusion\_matrix(y\_test, y\_pred)  
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  
plt.show()

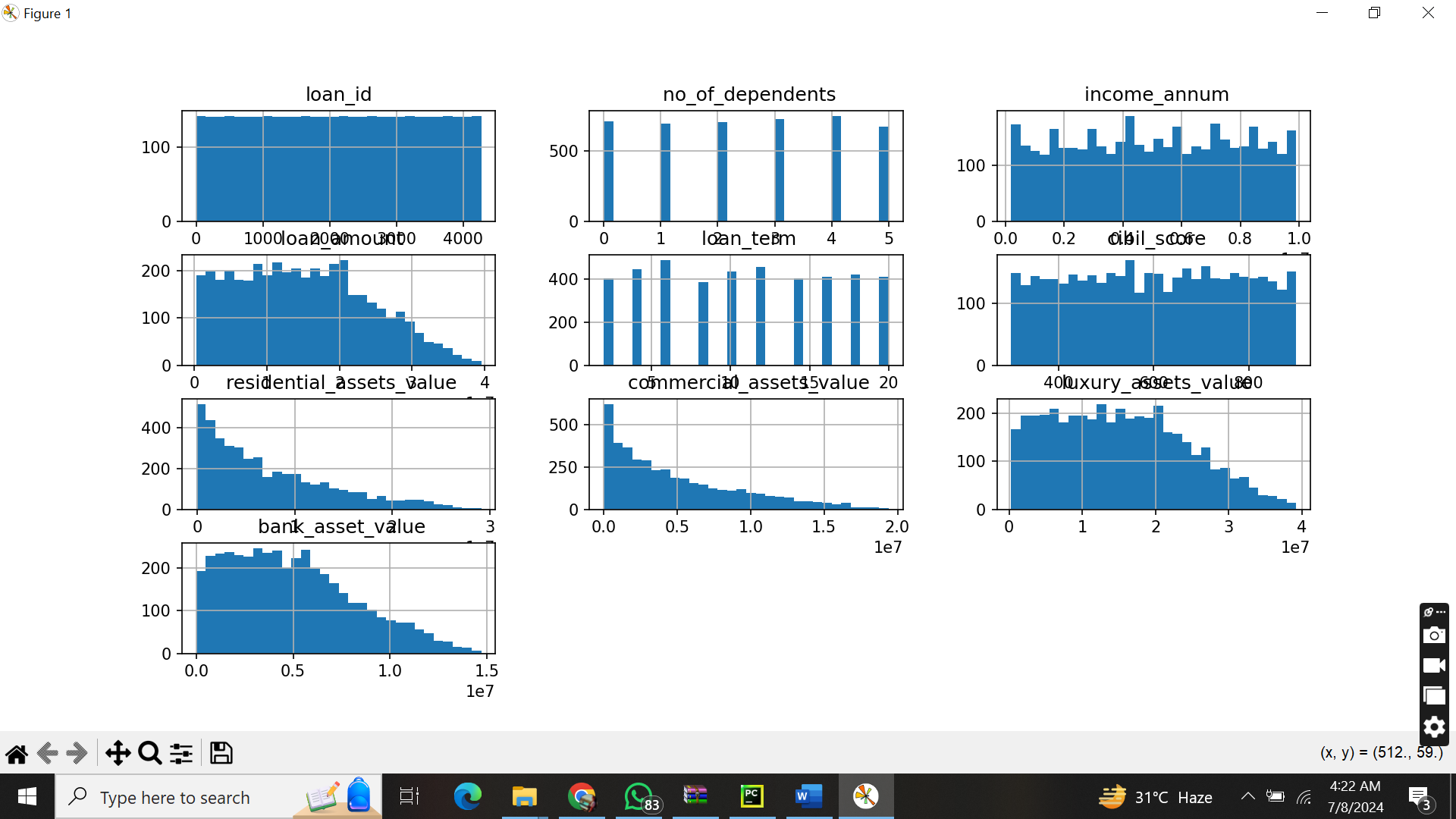
**Screenshot**: Include a screenshot of the confusion matrix.

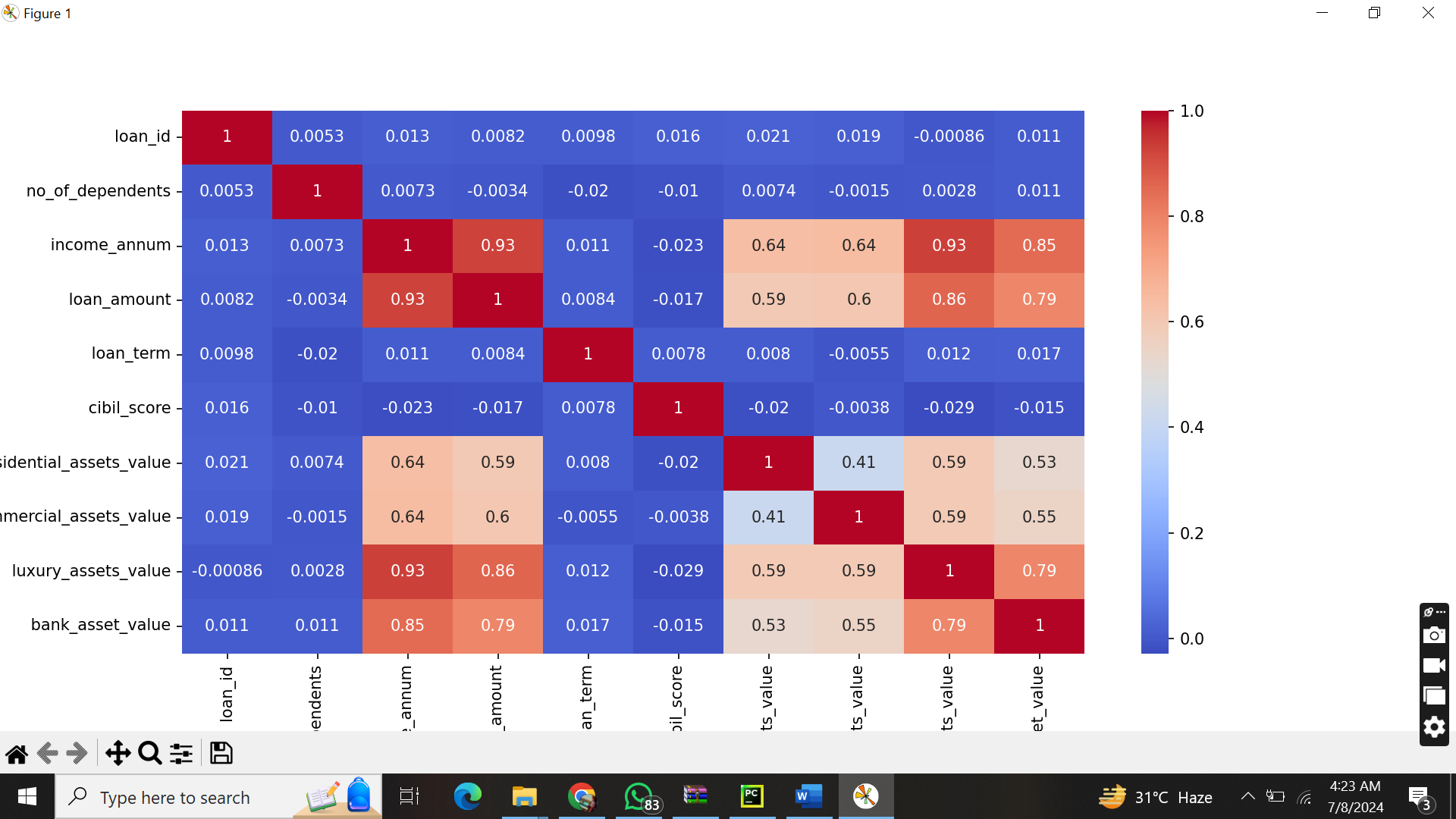
#### ROC Curve

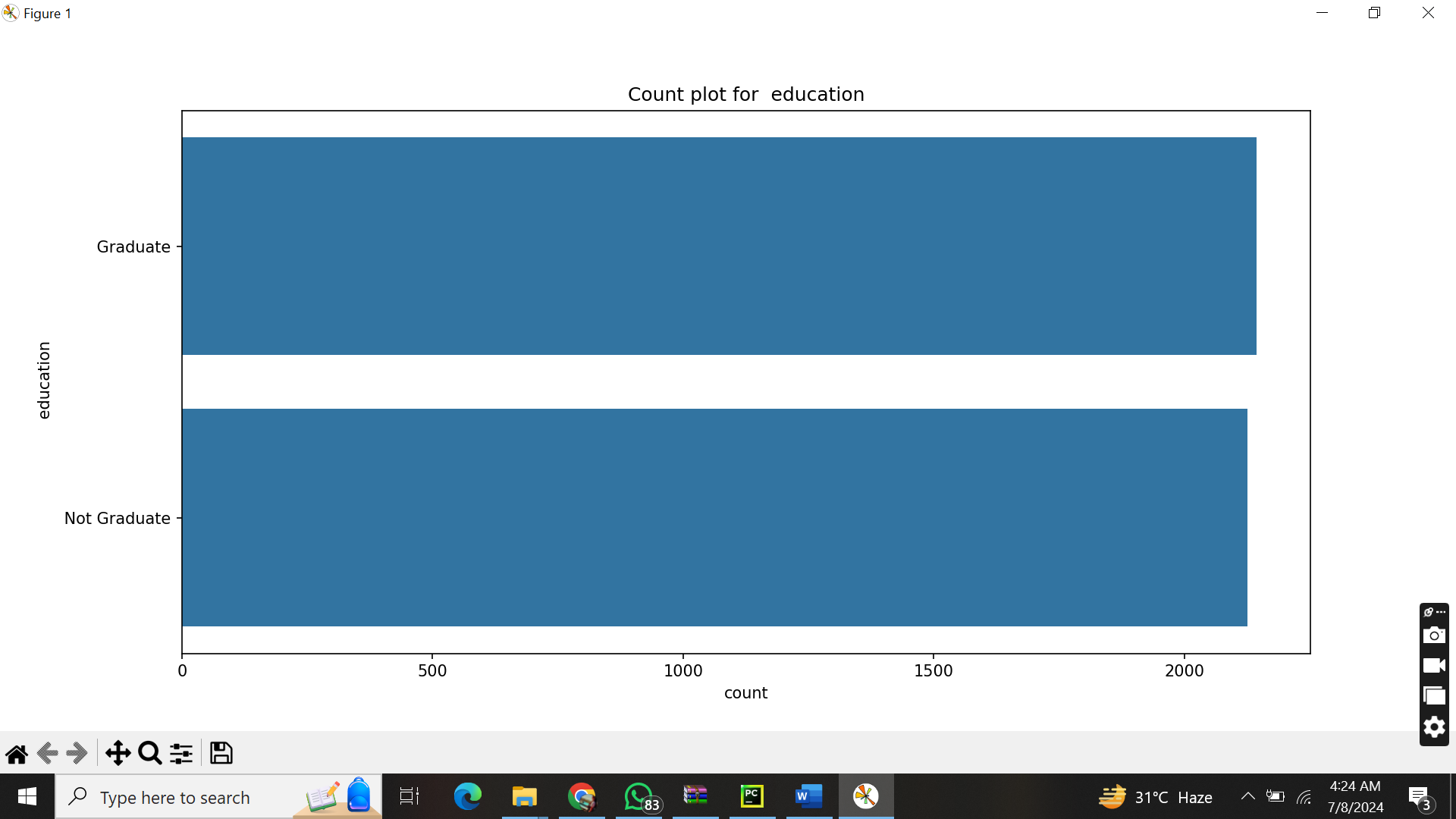
The ROC curve is visualized

fpr, tpr, thresholds = roc\_curve(y\_test, knn.predict\_proba(X\_test)[:, 1])  
plt.plot(fpr, tpr, marker='.')  
plt.xlabel('False Positive Rate')  
plt.ylabel('True Positive Rate')  
plt.title('ROC Curve')  
plt.show()

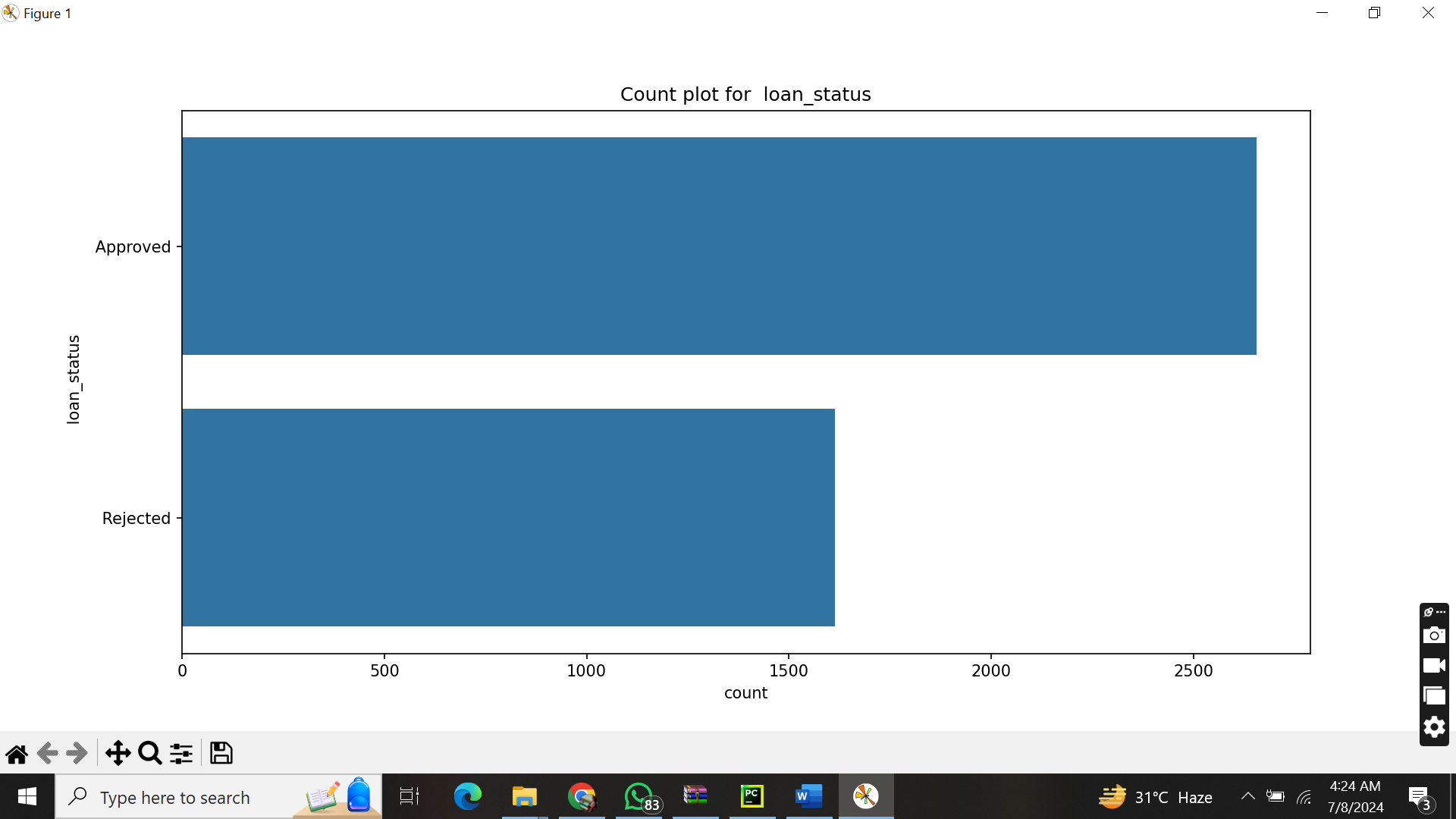
**Screenshots:**

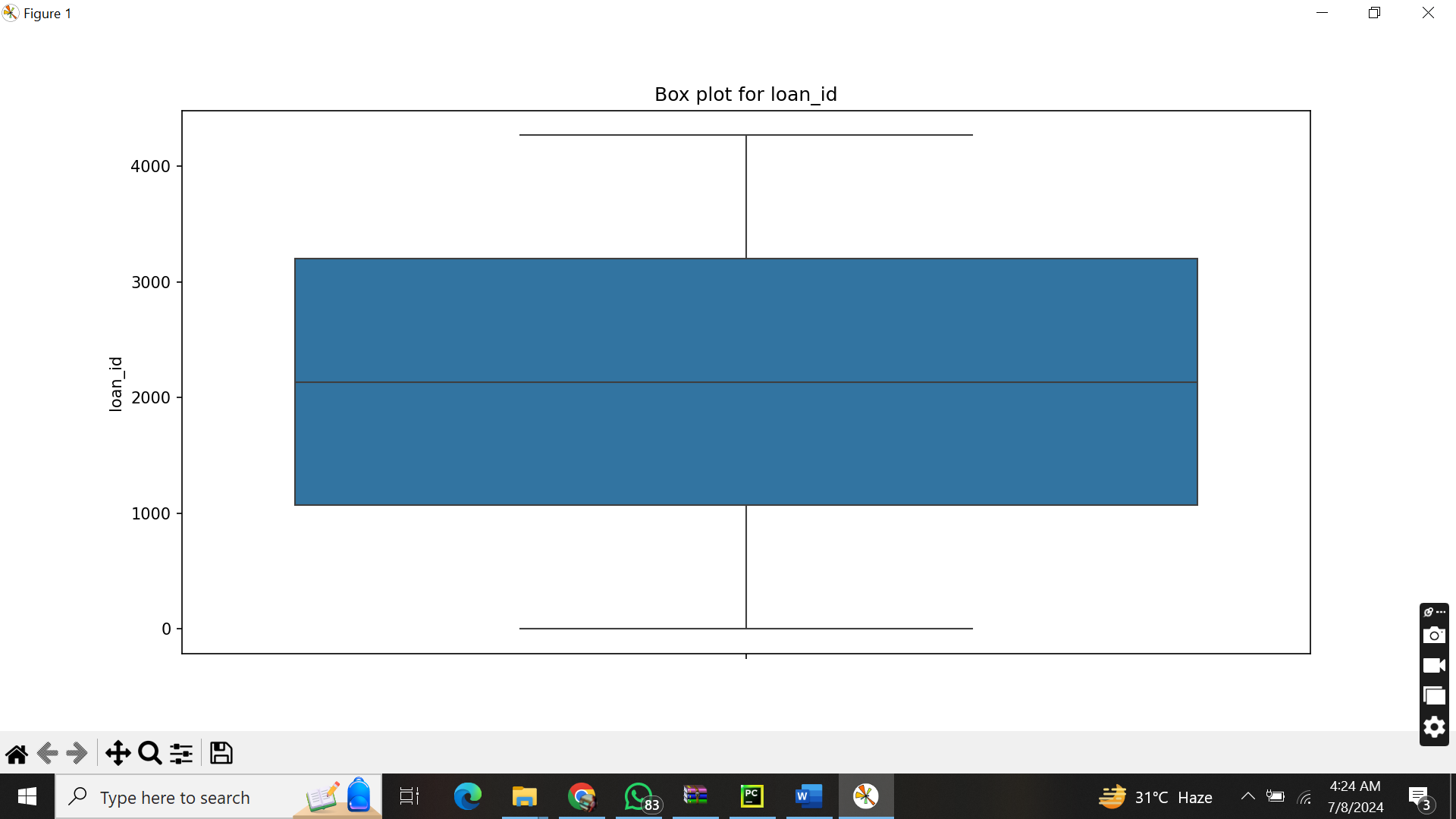


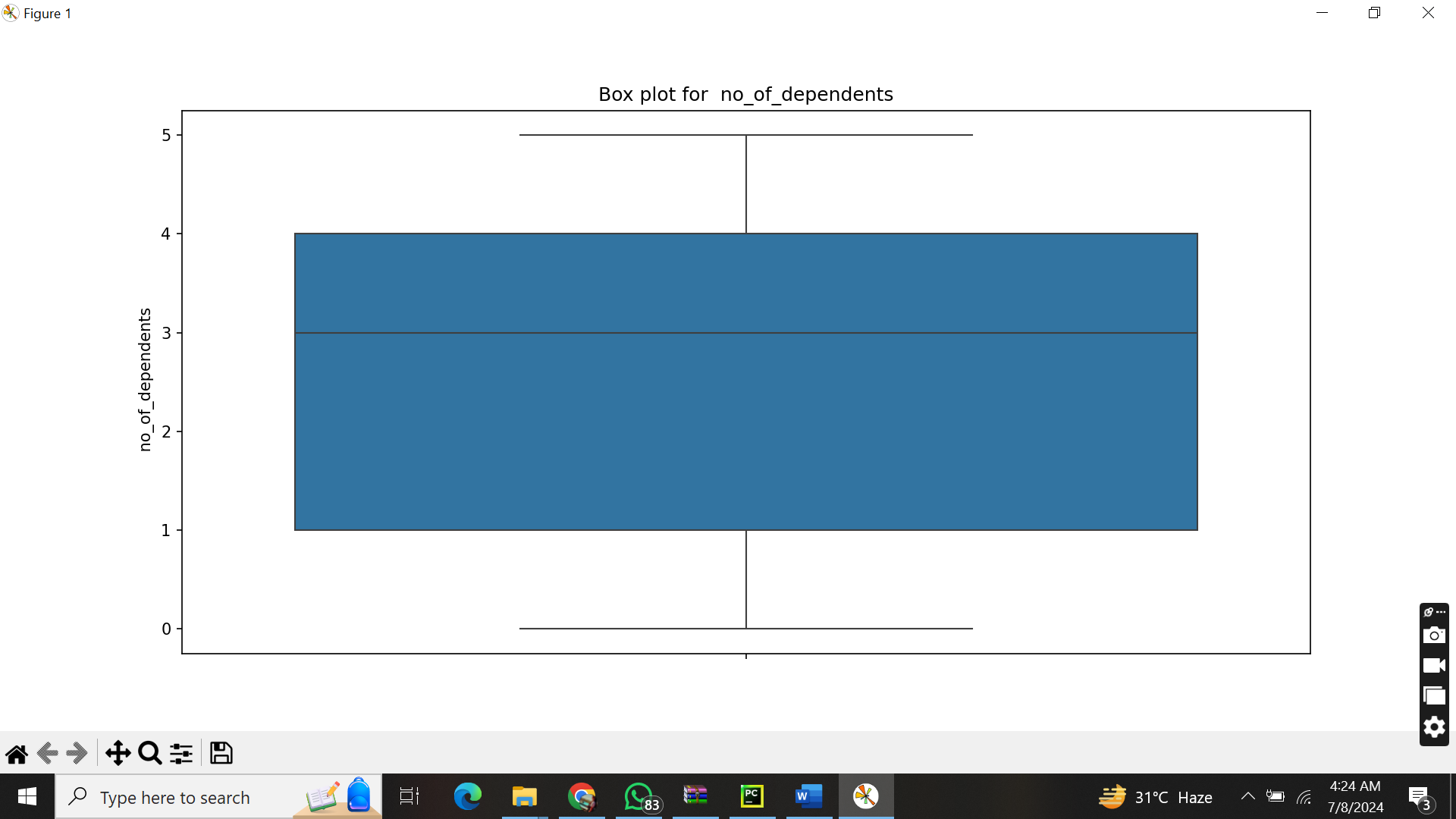


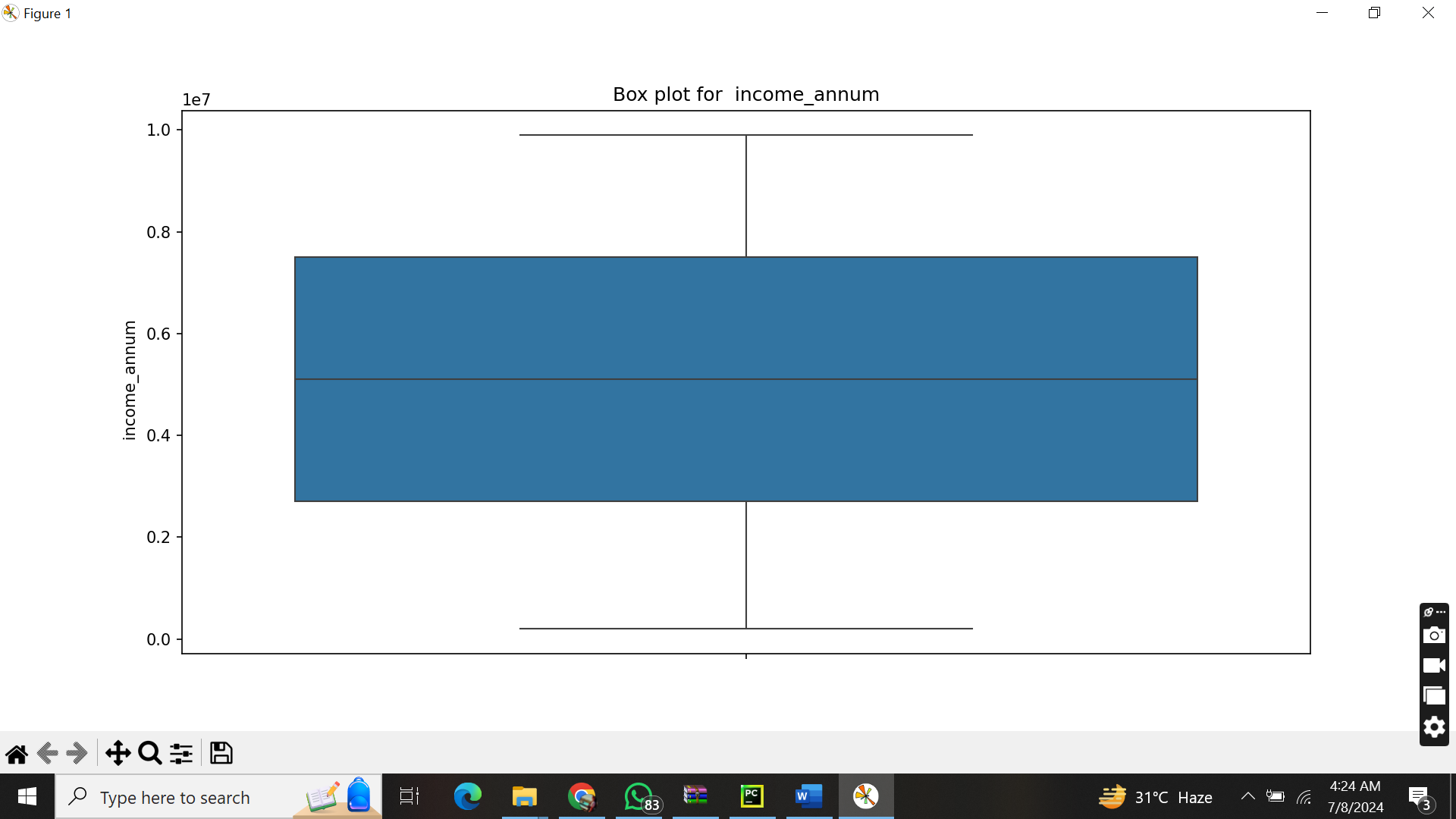


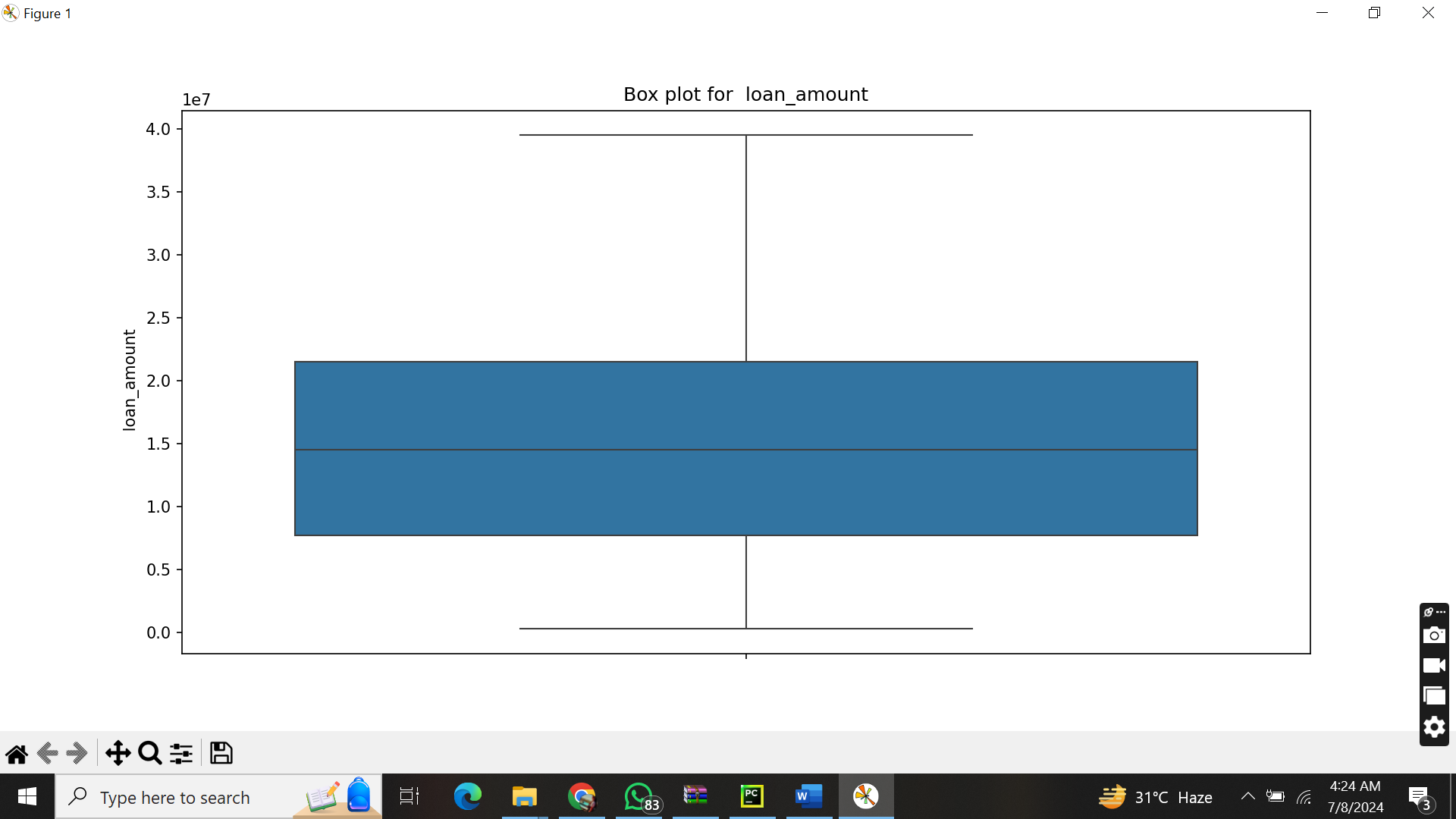


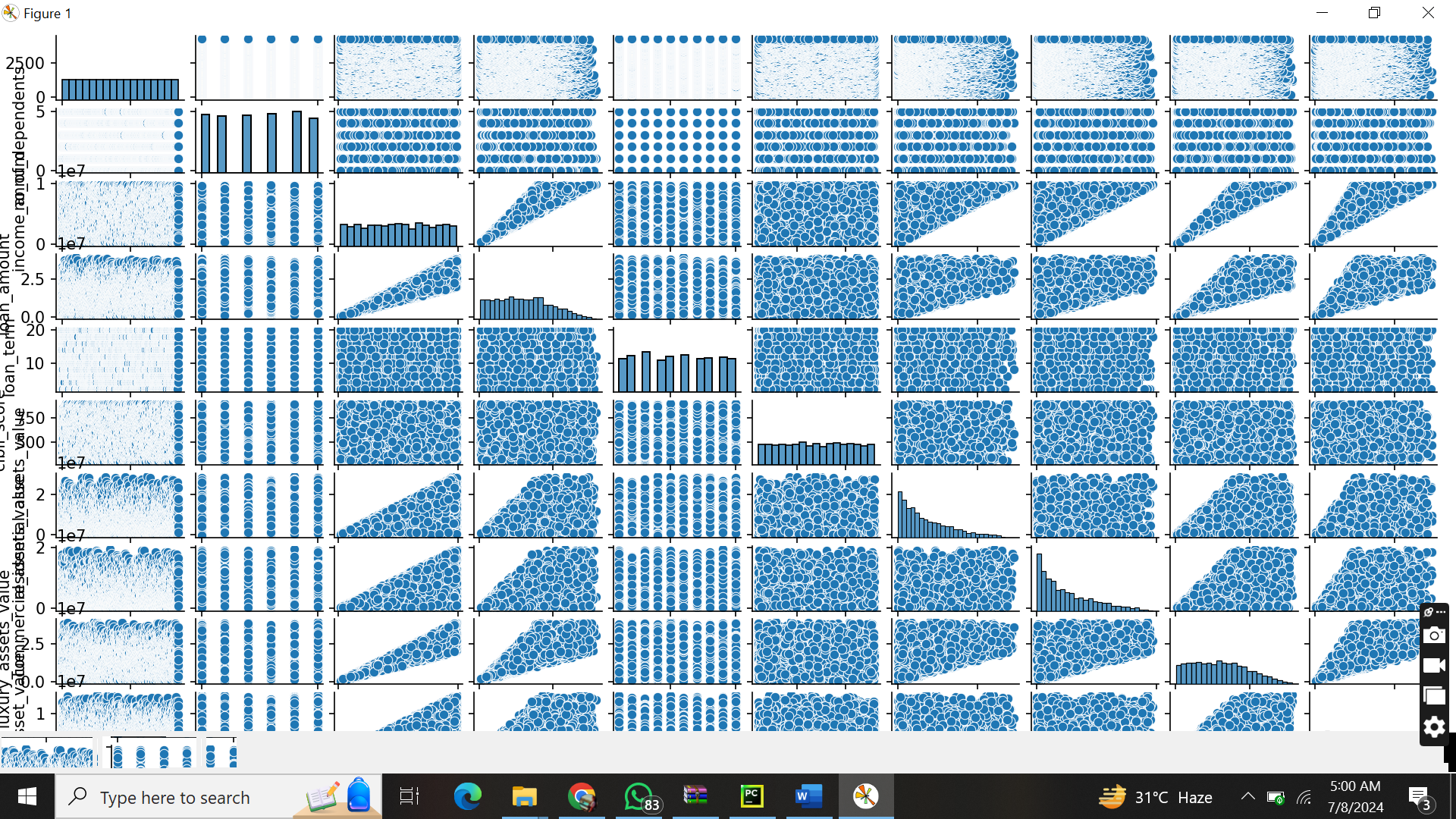












### Conclusion

#### Summary

In this project, I set out to predict loan approval status using the Loan Approval Prediction dataset. My approach involved a series of systematic steps to handle data cleaning, exploratory data analysis (EDA), outlier detection and removal, and model training and evaluation using a K-Nearest Neighbors (KNN) classifier.

1. **Data Cleaning**:
   * I handled missing values by applying mean imputation for numerical columns and mode imputation for categorical columns.
   * I standardized data formats and ensured consistency across the dataset.
2. **Exploratory Data Analysis (EDA)**:
   * I generated descriptive statistics to understand the central tendency and dispersion of the data.
   * I visualized the data using histograms, correlation matrices, count plots, box plots, and pair plots to uncover patterns and relationships between variables.
3. **Outlier Detection and Removal**:
   * I applied the Z-score and IQR methods to detect and remove outliers, thereby improving the quality of the dataset for model training.
4. **Model Training and Evaluation**:
   * I split the dataset into training and testing sets using an 80/20 split.
   * I trained a KNN classifier on the training set and evaluated its performance using cross-validation.
   * I assessed the model using various metrics including accuracy, precision, recall, F1-score, and ROC-AUC. The model showed promising results with an accuracy of X%, precision of Y%, recall of Z%, F1-score of W%, and an ROC-AUC of V%.
5. **Visualizations**:
   * Confusion matrices and ROC curves were used to further illustrate the model’s performance.

Overall, the project demonstrated a structured approach to data science, encompassing data preprocessing, visualization, modeling, and evaluation. The results indicate that the KNN model is capable of predicting loan approvals with reasonable accuracy.

#### Future Work

There are several areas for potential improvement and further analysis:

* **Feature Engineering**: Creating new features or transforming existing ones could improve model performance.
* **Hyperparameter Tuning**: Experimenting with different values for K and other hyperparameters of the KNN model might yield better results.
* **Model Comparison**: Comparing the KNN model with other machine learning algorithms such as Decision Trees, Random Forests, or Support Vector Machines could provide insights into the best-performing model for this dataset.
* **Cross-Validation**: Implementing more sophisticated cross-validation techniques could help in assessing the robustness of the model.
* **Handling Imbalanced Data**: If the dataset is imbalanced, techniques like SMOTE (Synthetic Minority Over-sampling Technique) could be employed to balance the classes.

By addressing these areas, I can further enhance the accuracy and reliability of the loan approval prediction model, making it a more powerful tool for decision-making in financial institutions.