**Loan Default Risk**

This project will be for the purpose of evaluating loan default risk from a dataset of borrowers. This dataset was taken from Kaggle and originated from Coursera's loan default prediction challenge. I will be looking at factors such as loan amount, DTI, income, and credit scores, and current debt to make the prediction of loan default risk.

1. **Data Collection and Preprocessing**

The first step was to import libraries and read the dataset. In my code, I imported all the needed libraries and imported the dataset and then read a sample of the data

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import plotly.express as px

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn import metrics

#Importing the dataset

df = pd.read\_csv("Loan\_default.csv")

#Reading the dataset to get a sample of the data.

df.head()

|  | LoanID | Age | Income | LoanAmount | CreditScore | MonthsEmployed | NumCreditLines | InterestRate | LoanTerm | DTIRatio | Education | EmploymentType | MaritalStatus | HasMortgage | HasDependents | LoanPurpose | HasCoSigner | Default |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | I38PQUQS96 | 56 | 85994 | 50587 | 520 | 80 | 4 | 15.23 | 36 | 0.44 | Bachelor's | Full-time | Divorced | Yes | Yes | Other | Yes | 0 |
| 1 | HPSK72WA7R | 69 | 50432 | 124440 | 458 | 15 | 1 | 4.81 | 60 | 0.68 | Master's | Full-time | Married | No | No | Other | Yes | 0 |
| 2 | C1OZ6DPJ8Y | 46 | 84208 | 129188 | 451 | 26 | 3 | 21.17 | 24 | 0.31 | Master's | Unemployed | Divorced | Yes | Yes | Auto | No | 1 |
| 3 | V2KKSFM3UN | 32 | 31713 | 44799 | 743 | 0 | 3 | 7.07 | 24 | 0.23 | High School | Full-time | Married | No | No | Business | No | 0 |
| 4 | EY08JDHTZP | 60 | 20437 | 9139 | 633 | 8 | 4 | 6.51 | 48 | 0.73 | Bachelor's | Unemployed | Divorced | No | Yes | Auto | No | 0 |

Next I read the datatypes because I need to know if there are any that I have to convert in order to make the classification model work. The models like numerical numbers.

#Checking all of our datatypes.

df.dtypes

LoanID object

Age int64

Income int64

LoanAmount int64

CreditScore int64

MonthsEmployed int64

NumCreditLines int64

InterestRate float64

LoanTerm int64

DTIRatio float64

Education object

EmploymentType object

MaritalStatus object

HasMortgage object

HasDependents object

LoanPurpose object

HasCoSigner object

Default int64

dtype: object

I see that I do have to convert some of the categorical features to numerical. I convert them using sklearn and then check the results

#Converting categorical to numerical

from sklearn.preprocessing import LabelEncoder

var\_mod =['LoanID','Education','EmploymentType', 'MaritalStatus','HasMortgage', 'HasDependents', 'LoanPurpose', 'HasCoSigner']

var\_mod = [col for col in var\_mod if col in df.columns]

le = LabelEncoder()

for i in var\_mod:

    df[i] = le.fit\_transform(df[i])

df.dtypes

LoanID int32

Age int64

Income int64

LoanAmount int64

CreditScore int64

MonthsEmployed int64

NumCreditLines int64

InterestRate float64

LoanTerm int64

DTIRatio float64

Education int32

EmploymentType int32

MaritalStatus int32

HasMortgage int32

HasDependents int32

LoanPurpose int32

HasCoSigner int32

Default int64

dtype: object

I now want to check for missing values that can skew the data in our models. If there are any missing values I would want to drop those columns. Here you can see that when I checked for missing values, we had none so there was no need for that step.

#Checking for missing values that we may not need to use

df.isnull().sum()

LoanID 0

Age 0

Income 0

LoanAmount 0

CreditScore 0

MonthsEmployed 0

NumCreditLines 0

InterestRate 0

LoanTerm 0

DTIRatio 0

Education 0

EmploymentType 0

MaritalStatus 0

HasMortgage 0

HasDependents 0

LoanPurpose 0

HasCoSigner 0

Default 0

dtype: int64

Here I was checking for any duplicate data that could skew the numbers as well

#Checking for duplicates

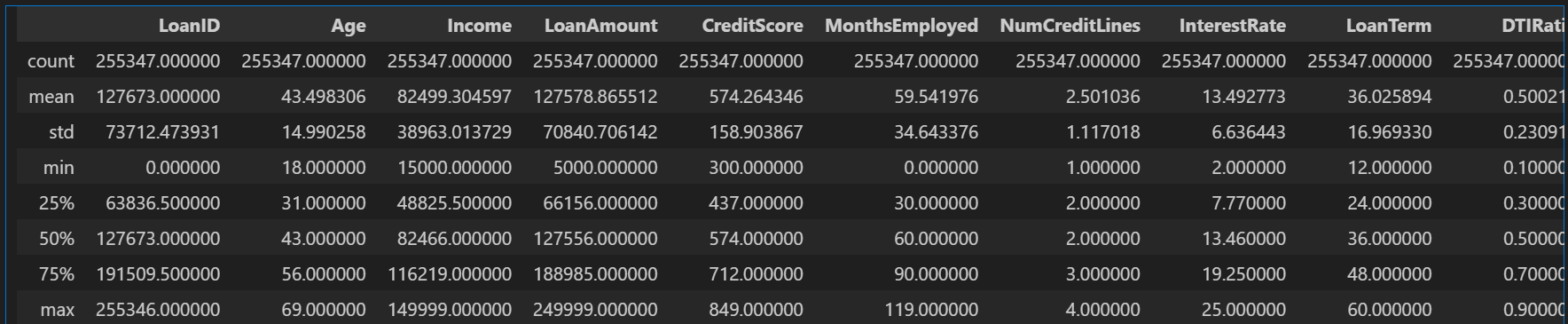
df.duplicated().sum()

**2. EDA and Analysis**

I want to display some stats about the data. Here I run the code to be able to see a sample.

#Looking at some statistics about the data

df.describe()



Here I’m looking for in the summary statistics is potential outliers. The differences between the mean and max on most of these aren’t bad. Loan amount is the only one that seems to have the biggest difference but that might be due to the loan purpose. Let's explore that a bit more.

This code helps me visualize if thee are any real outliers with the loan amount

#Visualizing the outliers

variables=['LoanAmount']

plt.figure(figsize=(20, 10))

ind=1

for  variable in (variables):

    plt.subplot(3, 3, ind)

    sns.boxplot(x=df[variable])

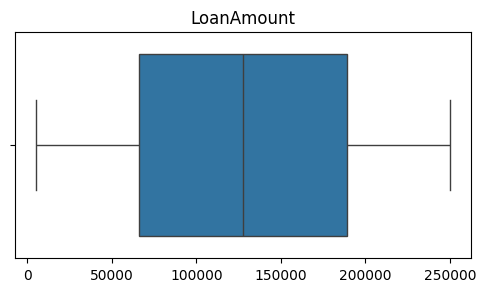
    plt.title(variable)

    ind+=1

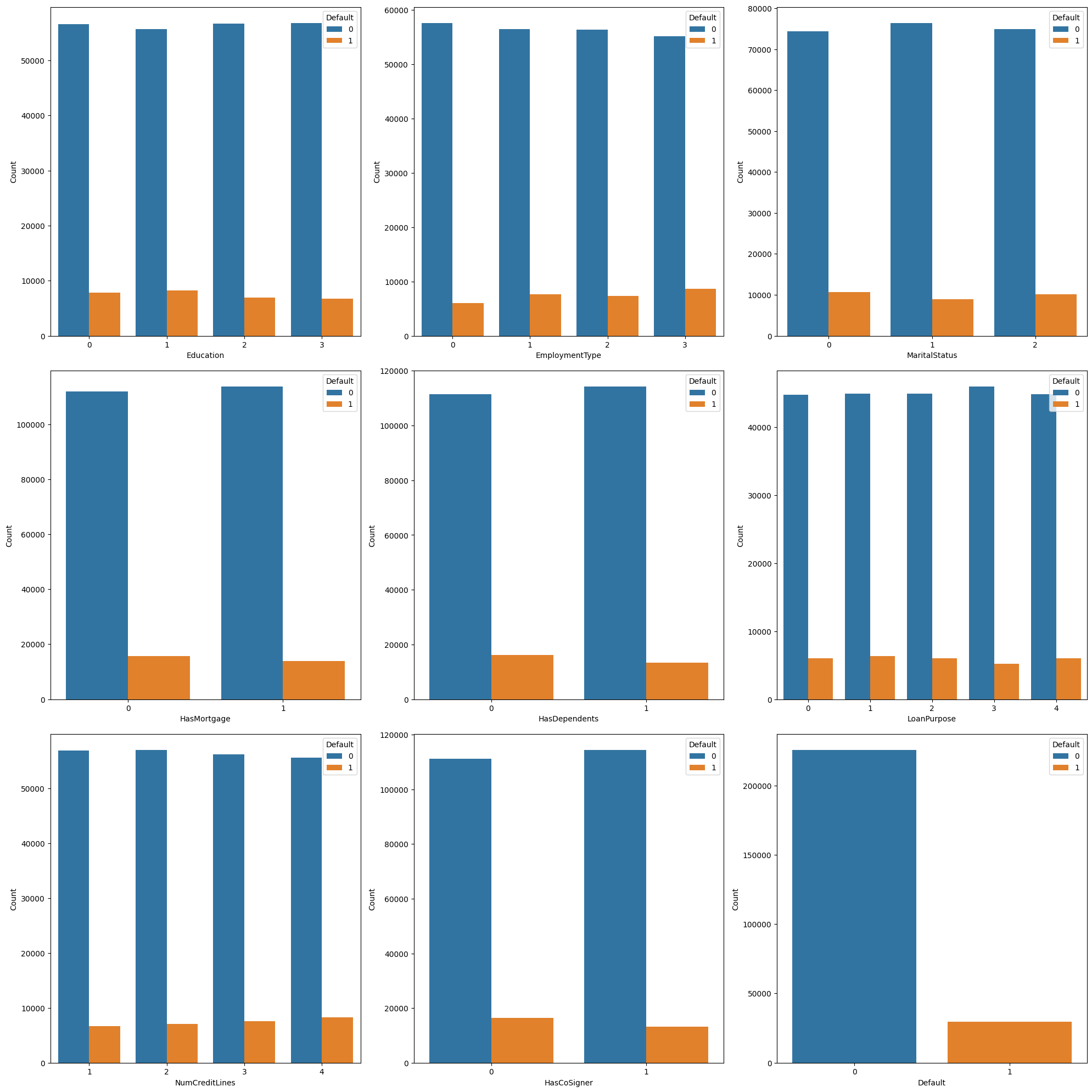
    plt.xlabel('')

plt.show()

My output suggests there isn’t anything off balance.



Next, I want to visualize how our target feature ‘Default’ looks against all the other features. I left out ‘LoanID’ because it was irrelevant to interpreting the data

 This is interesting. There’s only one currently in default and when you look at the comparison, you can see that this one had a higher number of credit lines, didn’t have a co-signer, and unemployed. That would make sense in why there could have been a default.

**3. Creating and Training the Model**

First I will be splitting the data into training and testing sets. This includes importing other libraries that I need to split them, create the decision tree model and post the accuracy scores.

#Splitting data into training and test sets

#Importing libraries

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier, export\_text

from sklearn.metrics import accuracy\_score, classification\_report

# Separate features (X) and target variable (y)

X = df.drop('Default', axis=1)

y = df['Default']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

Creating the decision tree classifier

#Create decision tree classifier

dt\_classifier = DecisionTreeClassifier(random\_state=42)

Training the model

# Training the model

dt\_classifier.fit(X\_train, y\_train)

Making predictions on the test set

# Make predictions on the test data

y\_pred = dt\_classifier.predict(X\_test)

**4. Model Interpretation**

Now I am going to evaluate the accuracy of my decision tree. I also need to import the sklearn metrics library to help with this.

#Evaluate the accuracy of the model

from sklearn import metrics

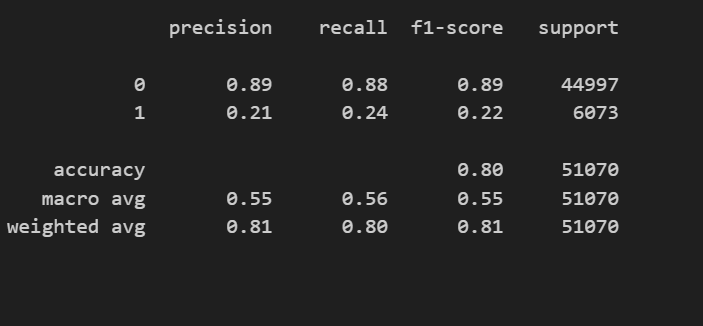
print(metrics.accuracy\_score(y\_test, y\_pred))

0.8014685725474838

Not too bad of a score. Let’s see more details using precision, recall, and F1 scores

#Evaluate using classification report to show precision, recall and F1 score)

print(metrics.classification\_report(y\_test, y\_pred))



I now want to know what the model saw as the most important features in determining loan default risk.

feature\_imp = pd.Series(dt\_classifier.feature\_importances\_, index=df.columns[:17]).sort\_values(ascending=False)

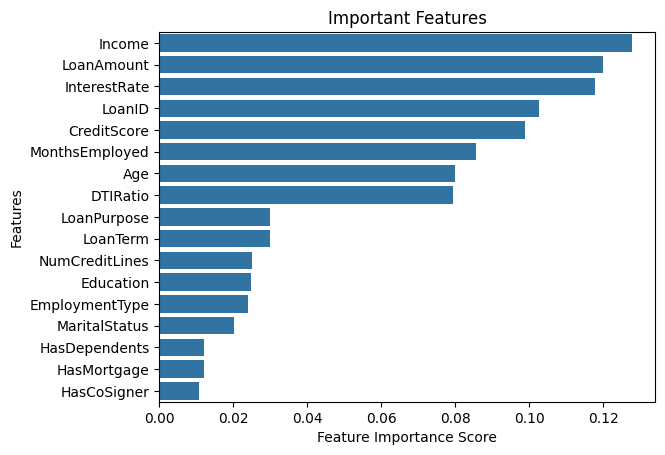
sns.barplot(x=feature\_imp, y=feature\_imp.index)

plt.xlabel('Feature Importance Score')

plt.ylabel('Features')

plt.title("Important Features")

plt.show()



This plot shows that income, loan amount and interest rate are the key factors in predicting loan default. Using this information, I want to next plot out how they all correlate together. I will use a correlation matrix

#Let's visuaize using correlation matrix

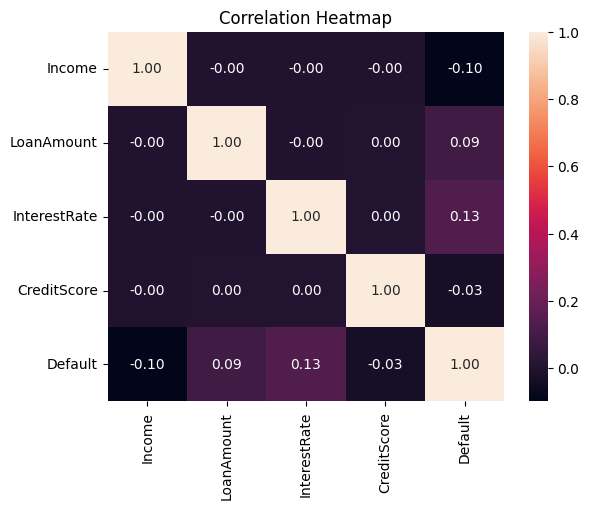
correlation\_matrix = df[['Income', 'LoanAmount', 'InterestRate', 'CreditScore','Default']].corr()

plt.figure()

sns.heatmap(correlation\_matrix, annot=True, fmt=".2f")

plt.title('Correlation Heatmap')

plt.show()



My observations on this are:

\*Default and interest rate positively correlated. Meaning if interest rate goes up so does default risk. Interest rate had the highest correlation with .13

\*Default and income negatively correlated. This means if income is lower, default risk is higher. This is the 2nd highest correlation.

\*Default and loan amount positively correlated. If loan amount goes up, so does default risk.

It’s interesting that credit score didn’t have a higher correlation.

Now let’s look at a confusion matrix. This will help us visualize how well the model predicted. I had to import another library with sklearn to create this.

from sklearn.metrics import confusion\_matrix

y\_pred=dt\_classifier.predict(X\_test)

y\_true=y\_test

cm=confusion\_matrix(y\_true,y\_pred)

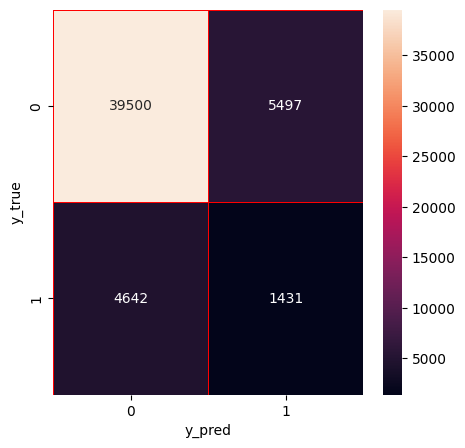
f, ax=plt.subplots(figsize=(5,5))

sns.heatmap(cm,annot=True,linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)

plt.xlabel("y\_pred")

plt.ylabel("y\_true")

plt.show()



My observations are:

Top left quadrant = True Positives = 39,500

Bottom right quadrant = True Negatives = 1431

Top right quadrant = False Positives = 5497

Bottom left quadrant = False Negatives = 4642

This is telling me that the model performed the right answer 40,931 times and predicted wrong only 10,139. Good but I think we need to check and see how a random forest classifier will work on this data and see if it can improve the accuracy

**5. Evaluation and Validation**

Here I am going to create the random forest classifier and compare the results of our decision tree classifier

#Let's see how random forest performs

from sklearn.ensemble import RandomForestClassifier

rf\_classifier = RandomForestClassifier(n\_estimators = 100)

# Training the model

rf\_classifier.fit(X\_train, y\_train)

# Training the model

rf\_classifier.fit(X\_train, y\_train)

# Make predictions on the test data

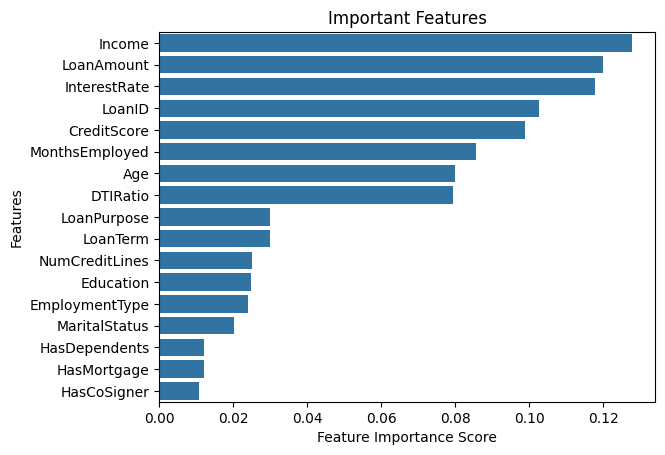
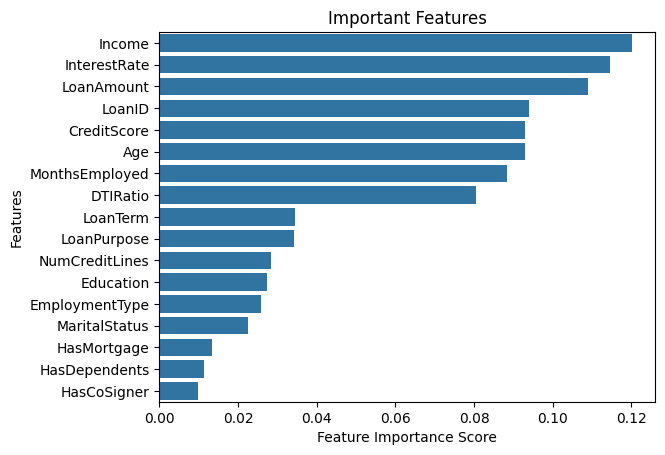
y\_pred = rf\_classifier.predict(X\_test)

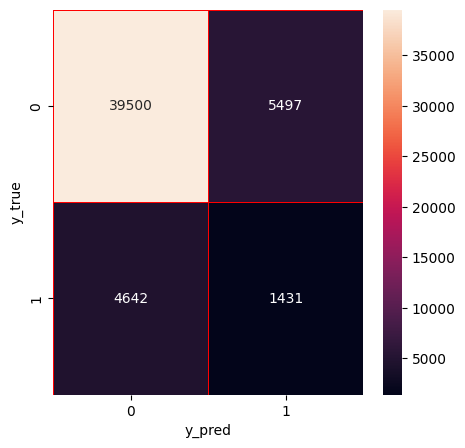
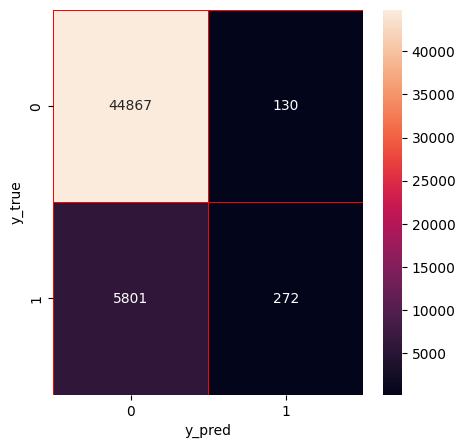
print(metrics.accuracy\_score(y\_test, y\_pred))

0.8838652829449775

Already we can see that random forest performed better. I am going to duplicate plotting feature importance, and confusion matrix of random forest and display them side by side to see the differences.

Decision Tree Random Forest

Key findings here are that random forest put more weight into interest rate than loan amount. Random forest made correct decisions 44,997 times vs decision tree at 40,931 times. The wrong predictions in random forest were also better, only being incorrect 6073 times vs 10,139.

**6. Conclusion**

The benefit of using these predictions for lending companies is to mitigate their risk of lending to those that have a higher risk of default. For example, if interest rates are higher which is a key factor in potential default, they can use this information to limit loan amounts based on income. Using this prediction will help in lessoning their burden of default and save substantial amount of money that could come with trying to recover the borrowed money. In turn, I believe that using these models to lessen their risk also benefits the customer by helping prevent them from getting in over their head.