COURSERA

Welcome to the Data Science Coding Challange!

Test your skills in a real-world coding challenge. Coding Challenges provide CS & DS Coding Competitions with Prizes and achievement badges!

CS & DS learners want to be challenged as a way to evaluate if they're job ready. So, why not create fun challenges and give winners something truly valuable such as complimentary access to select Data Science courses, or the ability to receive an achievement badge on their Coursera Skills Profile - highlighting their performance to recruiters.

Introduction

In this challenge, you'll get the opportunity to tackle one of the most industry-relevant machine learning problems with a unique dataset that will put your modeling skills to the test. Financial loan services are leveraged by companies across many industries, from big banks to financial institutions to government loans. One of the primary objectives of companies with financial loan services is to decrease payment defaults and ensure that individuals are paying back their loans as expected. In order to do this efficiently and systematically, many companies employ machine learning to predict which individuals are at the highest risk of defaulting on their loans, so that proper interventions can be effectively deployed to the right audience.

In this challenge, we will be tackling the loan default prediction problem on a very unique and interesting group of individuals who have taken financial loans.

Imagine that you are a new data scientist at a major financial institution and you are tasked with building a model that can predict which individuals will default on their loan payments. We have provided a dataset that is a sample of individuals who received loans in 2021.

This financial institution has a vested interest in understanding the likelihood of each individual to default on their loan payments so that resources can be allocated appropriately to support these borrowers. In this challenge, you will use your machine learning toolkit to do just that!

Understanding the Datasets

Train vs. Test

In this competition, you'll gain access to two datasets that are samples of past borrowers of a financial institution that contain information about the individual and the specific loan. One dataset is titled `train.csv` and the other is titled `test.csv`.

`train.csv` contains 70% of the overall sample (255,347 borrowers to be exact) and importantly, will reveal whether or not the borrower has defaulted on their loan payments (the "ground truth").

The `test.csv` dataset contains the exact same information about the remaining segment of the overall sample (109,435 borrowers to be exact), but does not disclose the "ground truth" for each borrower. It's your job to predict this outcome!

Using the patterns you find in the `train.csv` data, predict whether the borrowers in `test.csv` will default on their loan payments, or not.

Dataset descriptions

Both `train.csv` and `test.csv` contain one row for each unique Loan. For each Loan, a single observation (`LoanID`) is included during which the loan was active.

In addition to this identifier column, the `train.csv` dataset also contains the target label for the task, a binary column `Default` which indicates if a borrower has defaulted on payments.

Besides that column, both datasets have an identical set of features that can be used to train your model to make predictions. Below you can see descriptions of each feature. Familiarize yourself with them so that you can harness them most effectively for this machine learning task!

```python
import pandas as pd
data_descriptions = pd.read_csv('data_descriptions.csv')
pd.set_option('display.max_colwidth', None)
data_descriptions
```

```
       Column_name Column_type Data_type  \
0           LoanID  Identifier    string
1              Age     Feature   integer
2           Income     Feature   integer
3       LoanAmount     Feature   integer
4      CreditScore     Feature   integer
5   MonthsEmployed     Feature   integer
6   NumCreditLines     Feature   integer
7     InterestRate     Feature     float
8         LoanTerm     Feature   integer
9         DTIRatio     Feature     float
10       Education     Feature    string
11  EmploymentType     Feature    string
12   MaritalStatus     Feature    string
13     HasMortgage     Feature    string
14   HasDependents     Feature    string
15     LoanPurpose     Feature    string
16     HasCoSigner     Feature    string
17         Default      Target   integer


Description
0                                                                  A
```

```
unique identifier for each loan.
1
The age of the borrower.
2                                                                  The
annual income of the borrower.
3                                                                 The
amount of money being borrowed.
4                                The credit score of the borrower,
indicating their creditworthiness.
5                                                 The number of months
the borrower has been employed.
6                                                    The number of
credit lines the borrower has open.
7
The interest rate for the loan.
8                                                               The
term length of the loan in months.
9                  The Debt-to-Income ratio, indicating the borrower's
debt compared to their income.
10  The highest level of education attained by the borrower (PhD,
Master's, Bachelor's, High School).
11   The type of employment status of the borrower (Full-time, Part-
time, Self-employed, Unemployed).
12                                   The marital status of the
borrower (Single, Married, Divorced).
13                                                    Whether the
borrower has a mortgage (Yes or No).
14                                                    Whether the
borrower has dependents (Yes or No).
15                                 The purpose of the loan (Home,
Auto, Education, Business, Other).
16                                                       Whether the
loan has a co-signer (Yes or No).
17                   The binary target variable indicating whether the
loan defaulted (1) or not (0).
```

How to Submit your Predictions to Coursera

Submission Format:

In this notebook you should follow the steps below to explore the data, train a model using the data in `train.csv`, and then score your model using the data in `test.csv`. Your final submission should be a dataframe (call it `prediction_df` with two columns and exactly 109,435 rows (plus a header row). The first column should be `LoanID` so that we know which prediction belongs to which observation. The second column should be called `predicted_probability` and should be a numeric column representing the **likelihood that the borrower will default**.

Your submission will show an error if you have extra columns (beyond `LoanID` and `predicted_probability`) or extra rows. The order of the rows does not matter.

The naming convention of the dataframe and columns are critical for our autograding, so please make sure to use the exact naming conventions of `prediction_df` with column names `LoanID` and `predicted_probability`!

To determine your final score, we will compare your `predicted_probability` predictions to the source of truth labels for the observations in `test.csv` and calculate the ROC AUC. We choose this metric because we not only want to be able to predict which loans will default, but also want a well-calibrated likelihood score that can be used to target interventions and support most accurately.

Import Python Modules

First, import the primary modules that will be used in this project. Remember as this is an open-ended project please feel free to make use of any of your favorite libraries that you feel may be useful for this challenge. For example some of the following popular packages may be useful:

- pandas
- numpy
- Scipy
- Scikit-learn
- keras
- maplotlib
- seaborn
- etc, etc

```
# Import required packages

# Data packages
import pandas as pd
import numpy as np

# Machine Learning / Classification packages
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.dummy import DummyClassifier

# Visualization Packages
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
```

Load the Data

Let's start by loading the dataset `train.csv` into a dataframe `train_df`, and `test.csv` into a dataframe `test_df` and display the shape of the dataframes.

```
train_df=pd.read_csv("train.csv")
print("train_df Shape:", train_df.shape)
train_df.head()
```

```
train_df Shape: (255347, 18)
```

```
       LoanID  Age  Income  LoanAmount  CreditScore  MonthsEmployed  \
0  I38PQUQS96   56   85994       50587          520              80
1  HPSK72WA7R   69   50432      124440          458              15
2  C1OZ6DPJ8Y   46   84208      129188          451              26
3  V2KKSFM3UN   32   31713       44799          743               0
4  EY08JDHTZP   60   20437        9139          633               8

   NumCreditLines  InterestRate  LoanTerm  DTIRatio    Education  \
0               4         15.23        36      0.44   Bachelor's
1               1          4.81        60      0.68     Master's
2               3         21.17        24      0.31     Master's
3               3          7.07        24      0.23  High School
4               4          6.51        48      0.73   Bachelor's

  EmploymentType MaritalStatus HasMortgage HasDependents
LoanPurpose  \
0      Full-time      Divorced         Yes           Yes       Other

1      Full-time       Married          No            No       Other

2     Unemployed      Divorced         Yes           Yes        Auto

3      Full-time       Married          No            No    Business

4     Unemployed      Divorced          No           Yes        Auto


  HasCoSigner  Default
0         Yes        0
1         Yes        0
2          No        1
3          No        0
4          No        0
```

```
test_df=pd.read_csv("test.csv")
print("test_df Shape:", test_df.shape)
test_df.head()
```

```
test_df Shape: (109435, 17)
```

```
       LoanID  Age  Income  LoanAmount  CreditScore  MonthsEmployed  \
0  7RYZGMKJIR   32  131645       43797          802              23
1  JDL5RH07AM   61  134312       18402          369              87
2  STAL716Y79   55  115809      151774          563               3
3  SO0KKJ3IQB   58   94970       55789          337              24
4  T99CWTYDCP   63   71727      189798          451              52

   NumCreditLines  InterestRate  LoanTerm  DTIRatio    Education  \
0               2          6.10        24      0.13  High School
1               2         12.99        60      0.59  High School
2               3          5.51        48      0.82   Bachelor's
3               1         23.93        36      0.77   Bachelor's
4               3         22.05        48      0.44          PhD

  EmploymentType MaritalStatus HasMortgage HasDependents
LoanPurpose  \
0      Full-time      Divorced         Yes            No       Other

1  Self-employed        Single          No            No    Business

2      Full-time        Single         Yes           Yes       Other

3     Unemployed      Divorced          No            No    Business

4     Unemployed        Single         Yes            No        Auto


  HasCoSigner
0          No
1         Yes
2         Yes
3          No
4          No
```

Explore, Clean, Validate, and Visualize the Data (optional)

Feel free to explore, clean, validate, and visualize the data however you see fit for this competition to help determine or optimize your predictive model. Please note - the final autograding will only be on the accuracy of the `prediction_df` predictions.

Exploratory Data Analysis (EDA):

Before building the model, it's essential to understand data. We will perform exploratory data analysis to gain insights into the dataset. This includes checking for missing values, data types, and basic statistics.

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 255347 entries, 0 to 255346
Data columns (total 18 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   LoanID          255347 non-null  object 
 1   Age             255347 non-null  int64  
 2   Income          255347 non-null  int64  
 3   LoanAmount      255347 non-null  int64  
 4   CreditScore     255347 non-null  int64  
 5   MonthsEmployed  255347 non-null  int64  
 6   NumCreditLines  255347 non-null  int64  
 7   InterestRate    255347 non-null  float64
 8   LoanTerm        255347 non-null  int64  
 9   DTIRatio        255347 non-null  float64
 10  Education       255347 non-null  object 
 11  EmploymentType  255347 non-null  object 
 12  MaritalStatus   255347 non-null  object 
 13  HasMortgage     255347 non-null  object 
 14  HasDependents   255347 non-null  object 
 15  LoanPurpose     255347 non-null  object 
 16  HasCoSigner     255347 non-null  object 
 17  Default         255347 non-null  int64  
dtypes: float64(2), int64(8), object(8)
memory usage: 35.1+ MB
```

```python
train_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
```

```python
test_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
dtype: int64
```

Both `train.csv` and `test.csv` have no missing values and datatypes are identical

Data Preprocessing:

Prepare the data for training by handling missing values, encoding categorical variables, and splitting the training data into features (X) and the target (y).

```
# Convert categorical columns to numeric using Label Encoding
categorical_columns=['Education', 'EmploymentType', 'MaritalStatus',
'HasMortgage', 'HasDependents', 'LoanPurpose', 'HasCoSigner']
label_encoders={}

for column in categorical_columns:
    label_encoder=LabelEncoder()
    train_df[column]=label_encoder.fit_transform(train_df[column])
    test_df[column]=label_encoder.fit_transform(test_df[column])
    label_encoders[column]=label_encoder

train_df.head()
```

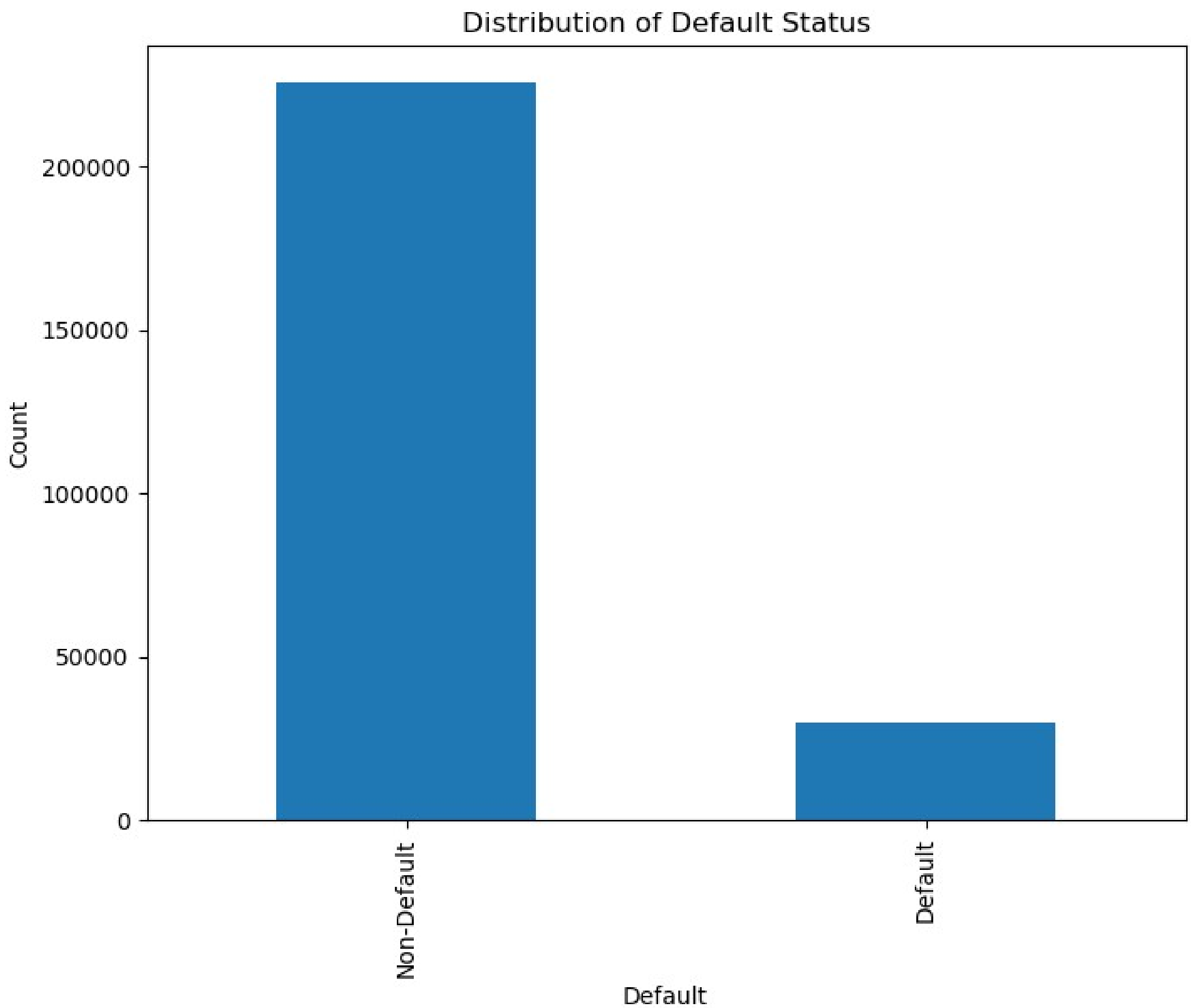
```
       LoanID  Age  Income  LoanAmount  CreditScore  MonthsEmployed  \
0  I38PQUQS96   56   85994       50587          520              80   
1  HPSK72WA7R   69   50432      124440          458              15   
2  C1OZ6DPJ8Y   46   84208      129188          451              26   
3  V2KKSFM3UN   32   31713       44799          743               0   
4  EY08JDHTZP   60   20437        9139          633               8   

   NumCreditLines  InterestRate  LoanTerm  DTIRatio  Education  \
0               4         15.23        36      0.44          0   
```

```
1                  1           4.81          60       0.68              2
2                  3          21.17          24       0.31              2
3                  3           7.07          24       0.23              1
4                  4           6.51          48       0.73              0

   EmploymentType  MaritalStatus  HasMortgage  HasDependents
LoanPurpose  \
0               0              0            1              1
4
1               0              1            0              0
4
2               3              0            1              1
0
3               0              1            0              0
1
4               3              0            0              1
0

   HasCoSigner  Default
0            1        0
1            1        0
2            0        1
3            0        0
4            0        0
```

Histogram of Default Status

This histogram will show the distribution of default and non-default loans in the training data.

```python
# Plot a histogram of Default status
plt.figure(figsize=(8, 6))
train_df['Default'].value_counts().plot(kind='bar')
plt.title('Distribution of Default Status')
plt.xlabel('Default')
plt.ylabel('Count')
plt.xticks([0, 1], ['Non-Default', 'Default'])
plt.show();
```



Correlation Matrix:

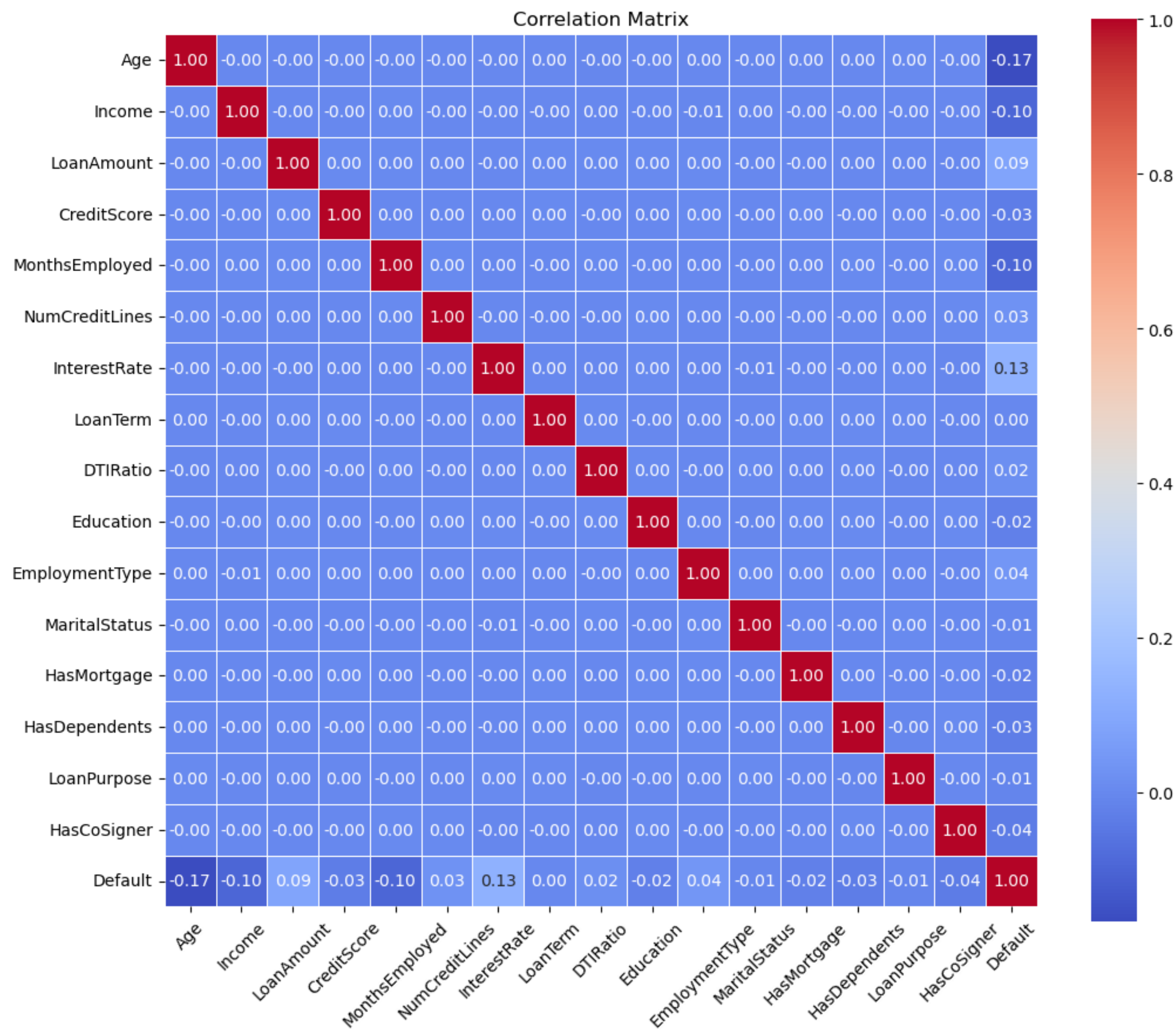
A correlation matrix will help us understand the relationships between different features

```
# Calculation of the correlation matrix
correlation_matrix=train_df.corr()
correlation_matrix

                     Age    Income  LoanAmount  CreditScore  
MonthsEmployed  \
Age             1.000000 -0.001244   -0.002213    -0.000548        -
0.000341   
Income         -0.001244  1.000000   -0.000865    -0.001430        
0.002675   
LoanAmount     -0.002213 -0.000865    1.000000     0.001261        
0.002817   
CreditScore    -0.000548 -0.001430    0.001261     1.000000        
0.000613   
```

```
MonthsEmployed -0.000341  0.002675    0.002817     0.000613
1.000000
NumCreditLines -0.000890 -0.002016    0.000794     0.000016
0.001267
InterestRate   -0.001127 -0.002303   -0.002291     0.000436
0.000096
LoanTerm        0.000263 -0.000998    0.002538     0.001130        -
0.001166
DTIRatio       -0.004689  0.000205    0.001122    -0.001039
0.001765
Education      -0.000882 -0.000965    0.002551     0.000214        -
0.001304
EmploymentType  0.000787 -0.005146    0.003060     0.003503
0.000564
MaritalStatus  -0.002187  0.000637   -0.000771    -0.003218        -
0.000095
HasMortgage     0.000035 -0.000945   -0.000801     0.001728
0.000210
HasDependents   0.000710 -0.001570    0.000139    -0.003018
0.001450
LoanPurpose     0.002264 -0.002092    0.000057     0.000596        -
0.002579
HasCoSigner    -0.002918 -0.003524   -0.001848    -0.002755
0.001045
Default        -0.167783 -0.099119    0.086659    -0.034166        -
0.097374

                NumCreditLines  InterestRate  LoanTerm  DTIRatio
Education  \
Age                  -0.000890     -0.001127  0.000263 -0.004689  -
0.000882
Income               -0.002016     -0.002303 -0.000998  0.000205  -
0.000965
LoanAmount            0.000794     -0.002291  0.002538  0.001122
0.002551
CreditScore           0.000016      0.000436  0.001130 -0.001039
0.000214
MonthsEmployed        0.001267      0.000096 -0.001166  0.001765  -
0.001304
NumCreditLines        1.000000     -0.000297 -0.000226 -0.000586
0.002691
InterestRate         -0.000297      1.000000  0.000892  0.000575
0.002879
LoanTerm             -0.000226      0.000892  1.000000  0.002273  -
0.002999
DTIRatio             -0.000586      0.000575  0.002273  1.000000
0.001789
Education             0.002691      0.002879 -0.002999  0.001789
1.000000
```

```
EmploymentType        0.000219      0.000525  0.000779 -0.000578
0.000236
MaritalStatus        -0.000664     -0.005079 -0.001042  0.004492  -
0.004717
HasMortgage          -0.001744     -0.000424  0.001775  0.000231
0.001167
HasDependents        -0.001895     -0.000243  0.002417  0.001492
0.001048
LoanPurpose           0.000340      0.001472  0.002856 -0.003819  -
0.003271
HasCoSigner           0.002105     -0.003991 -0.001166  0.000373
0.001707
Default               0.028330      0.131273  0.000545  0.019236  -
0.022835

                EmploymentType  MaritalStatus  HasMortgage
HasDependents  \
Age                   0.000787      -0.002187     0.000035
0.000710
Income               -0.005146       0.000637    -0.000945       -
0.001570
LoanAmount            0.003060      -0.000771    -0.000801
0.000139
CreditScore           0.003503      -0.003218     0.001728       -
0.003018
MonthsEmployed        0.000564      -0.000095     0.000210
0.001450
NumCreditLines        0.000219      -0.000664    -0.001744       -
0.001895
InterestRate          0.000525      -0.005079    -0.000424       -
0.000243
LoanTerm              0.000779      -0.001042     0.001775
0.002417
DTIRatio             -0.000578       0.004492     0.000231
0.001492
Education             0.000236      -0.004717     0.001167
0.001048
EmploymentType        1.000000       0.002768     0.001193
0.002480
MaritalStatus         0.002768       1.000000    -0.000408       -
0.000437
HasMortgage           0.001193      -0.000408     1.000000
0.000067
HasDependents         0.002480      -0.000437     0.000067
1.000000
LoanPurpose           0.000734       0.001434    -0.002157       -
0.003759
HasCoSigner          -0.000033      -0.000888    -0.003529
0.001602
```

```
Default                      0.041010      -0.007902    -0.022856       -
0.034678

                LoanPurpose  HasCoSigner   Default
Age                0.002264    -0.002918 -0.167783
Income            -0.002092    -0.003524 -0.099119
LoanAmount         0.000057    -0.001848  0.086659
CreditScore        0.000596    -0.002755 -0.034166
MonthsEmployed    -0.002579     0.001045 -0.097374
NumCreditLines     0.000340     0.002105  0.028330
InterestRate       0.001472    -0.003991  0.131273
LoanTerm           0.002856    -0.001166  0.000545
DTIRatio          -0.003819     0.000373  0.019236
Education         -0.003271     0.001707 -0.022835
EmploymentType     0.000734    -0.000033  0.041010
MaritalStatus      0.001434    -0.000888 -0.007902
HasMortgage       -0.002157    -0.003529 -0.022856
HasDependents     -0.003759     0.001602 -0.034678
LoanPurpose        1.000000    -0.001935 -0.010096
HasCoSigner       -0.001935     1.000000 -0.039109
Default           -0.010096    -0.039109  1.000000

# Create a heatmap of the correlation matrix
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f",
cmap='coolwarm', linewidths=0.5, square=True)

# Adjust font size and rotation for the labels
plt.xticks(fontsize=10, rotation=45)
plt.yticks(fontsize=10)
plt.title('Correlation Matrix', fontsize=12)
plt.show()
```



Train a machine learning model

```
# Define features and target variable
X=train_df.drop(['LoanID', 'Default'], axis=1)
y=train_df['Default']

# Split the data into a training set and a validation set
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)

# Initialize and train a Random Forest Classifier
model=RandomForestClassifier()
model.fit(X, y)


RandomForestClassifier()
```

```
model.score(X_train, y_train)
```

```
0.9999804187451353
```

```
# Predict Probabilities
predicted_probabilities=model.predict_proba(X_val)[:, 1]
```

Calculate ROC AUC

To determine your final score, we will compare your predicted_probability predictions to the source of truth labels for the observations

```
roc_auc=roc_auc_score(y_val, predicted_probabilities)

print(f'Validation ROC AUC Score: {roc_auc}')
```

```
Validation ROC AUC Score: 1.0
```

```
# Now, we can use the trained model to make predictions on the test data
X_test=test_df.drop(columns=['LoanID'])

# Predict probabilities for the test data
predicted_probabilities=model.predict_proba(X_test)[:, 1]
```

Create the submission dataframe with LoanID and predicted_probability and save it.

```
# Predicted_probabilities probabilities should be 0 and 1

threshold = 0.5
binary_predictions=(predicted_probabilities>=threshold).astype(int)

prediction_df=pd.DataFrame({
    'LoanID': test_df[['LoanID']].values[:, 0],
    'predicted_probability': binary_predictions
})

prediction_df.head(10)
```

	LoanID	predicted_probability
0	7RYZGMKJIR	0
1	JDL5RH07AM	0
2	STAL716Y79	0
3	SO0KKJ3IQB	0
4	T99CWTYDCP	0
5	0SNHFWV4UP	0
6	S6ITP6LGYS	0
7	A6I7U12IRJ	0

```
8  8W6KY50JU4                      0
9  THFQ08OLMU                      0
```

```
prediction_df.shape
```

```
(109435, 2)
```

```
prediction_df.predicted_probability.value_counts()
```

```
0    108299
1      1136
Name: predicted_probability, dtype: int64
```

Final Tests - **IMPORTANT** - the cells below must be run prior to submission

Below are some tests to ensure your submission is in the correct format for autograding. The autograding process accepts a csv `prediction_submission.csv` which we will generate from our `prediction_df` below. Please run the tests below an ensure no assertion errors are thrown.

```
# FINAL TEST CELLS - please make sure all of your code is above these
test cells

# Writing to csv for autograding purposes
prediction_df.to_csv("prediction_submission.csv", index=False)
submission = pd.read_csv("prediction_submission.csv")

assert isinstance(submission, pd.DataFrame), 'You should have a
dataframe named prediction_df.'

# FINAL TEST CELLS - please make sure all of your code is above these
test cells

assert submission.columns[0] == 'LoanID', 'The first column name
should be CustomerID.'
assert submission.columns[1] == 'predicted_probability', 'The second
column name should be predicted_probability.'

# FINAL TEST CELLS - please make sure all of your code is above these
test cells

assert submission.shape[0] == 109435, 'The dataframe prediction_df
should have 109435 rows.'

# FINAL TEST CELLS - please make sure all of your code is above these
test cells

assert submission.shape[1] == 2, 'The dataframe prediction_df should
have 2 columns.'
```

```python
roc_auc=roc_auc_score(y_val, predicted_probabilities)

print(f'Validation ROC AUC Score: {roc_auc}')
```

```
Validation ROC AUC Score: 1.0
```