LoanDefaultPrediction

November 7, 2023

1 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.

1.1 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!

1.2 Understanding the Datasets

1.2.1 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.

1.2.2 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!

[1]:
```
import pandas as pd
data_descriptions = pd.read_csv('data_descriptions.csv')
pd.set_option('display.max_colwidth', None)
data_descriptions
```

[1]:

	Column_name	Column_type	Data_type	Description
0	LoanID	Identifier	string	A unique identifier for each loan.
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	

```
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).
```

1.3 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.

1.4 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

[2]:
```
# 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.dummy import DummyClassifier

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

[3]:
```
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder, StandardScaler,␣
 ↪PolynomialFeatures, KBinsDiscretizer,FunctionTransformer, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import scipy.stats as stats
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.svm import SVC
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import seaborn as sns
from sklearn.base import BaseEstimator, TransformerMixin
```

1.5 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.

[4]:
```
train_df = pd.read_csv("train.csv")
print('train_df Shape:', train_df.shape)
train_df.head()
```

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

[4]:

	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

[5]:
```
test_df = pd.read_csv("test.csv")
print('test_df Shape:', test_df.shape)
test_df.head()
```

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

[5]:

	LoanID	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Education	EmploymentType	MaritalStatus	HasMortgage	HasDependents	LoanPurpose	HasCoSigner
0	7RYZGMKJIR	32	131645	43797	802	23	2	6.10	24	0.13	High School	Full-time	Divorced	Yes	No	Other	No
1	JDL5RH07AM	61	134312	18402	369	87	2	12.99	60	0.59	High School	Self-employed	Single	No	No	Business	Yes
2	STAL716Y79	55	115809	151774	563	3	3	5.51	48	0.82	Bachelor's	Full-time	Single	Yes	Yes	Other	Yes
3	SOOKKJ3IQB	58	94970	55789	337	24	1	23.93	36	0.77	Bachelor's	Unemployed	Divorced	No	No	Business	No
4	T99CWTYDCP	63	71727	189798	451	52	3	22.05	48	0.44	PhD	Unemployed	Single	Yes	No	Auto	No

1.6 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.

[6]:
```
list(train_df.columns.values)
```

[6]:
```
['LoanID',
 'Age',
 'Income',
 'LoanAmount',
 'CreditScore',
 'MonthsEmployed',
```

```
 'NumCreditLines',
 'InterestRate',
 'LoanTerm',
 'DTIRatio',
 'Education',
 'EmploymentType',
 'MaritalStatus',
 'HasMortgage',
 'HasDependents',
 'LoanPurpose',
 'HasCoSigner',
 'Default']
```

1.7 Make predictions (required)

Remember you should create a dataframe named `prediction_df` with exactly 109,435 entries plus a header row attempting to predict the likelihood of borrowers to default on their loans in `test_df`. Your submission will throw an error if you have extra columns (beyond `LoanID` and `predicted_probaility`) or extra rows.

The file should have exactly 2 columns: `LoanID` (sorted in any order) `predicted_probability` (contains your numeric predicted probabilities between 0 and 1, e.g. from `estimator.predict_proba(X, y)[:, 1]`)

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`!

1.7.1 Example prediction submission:

The code below is a very naive prediction method that simply predicts loan defaults using a Dummy Classifier. This is used as just an example showing the submission format required. Please change/alter/delete this code below and create your own improved prediction methods for generating `prediction_df`.

PLEASE CHANGE CODE BELOW TO IMPLEMENT YOUR OWN PREDICTIONS

[7]:
```
X = train_df.drop(columns=['LoanID', 'Default', 'LoanTerm'])  # Exclude␣
↪'LoanTerm'
y = train_df['Default']

# Define numerical, ordinal, and nominal features (excluding 'DTIRatio' and␣
↪'LoanTerm')
numerical_features = ['Age', 'LoanAmount', 'CreditScore', 'MonthsEmployed',␣
↪'NumCreditLines', 'InterestRate', 'DTIRatio']
ordinal_features = ['Education', 'EmploymentType']
nominal_features = ['MaritalStatus', 'HasMortgage', 'HasDependents',␣
↪'LoanPurpose', 'HasCoSigner']
```

```
# Define ordinal encoding mapping
education_order = ['High School', "Bachelor's", "Master's", 'PhD']
employment_order = ['Unemployed', 'Part-time', 'Self-employed', 'Full-time']

# Manually bin 'LoanAmount'
X['LoanAmount'] = pd.cut(X['LoanAmount'], bins=50, labels=False)

# Add a square term for 'Age'
X['AgeSquared'] = X['Age'] ** 2

# Manually bin 'Income' and apply logarithm transformation
income_bins = [0, 15000, 17000, 20000, 25000, 35000, 50000, 75000, 150000,␣
 ↪float('inf')]  # Define income bins
X['IncomeBins'] = pd.cut(X['Income'], bins=income_bins, labels=False)
X['LogIncome'] = np.log(X['IncomeBins']+1)

# Create 'LoanAmount/Income' ratio feature
X['LoanAmountToIncome'] = X['LoanAmount'] / (X['Income'] + 1)

X = X.drop(columns=['Income'])

# Create transformers for preprocessing
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])
ordinal_transformer = Pipeline(steps=[('ordinal',␣
 ↪OrdinalEncoder(categories=[education_order, employment_order]))])
nominal_transformer = Pipeline(steps=[('onehot', OneHotEncoder(drop='first'))])

# Combine transformers using ColumnTransformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_features + ['AgeSquared',␣
 ↪'LogIncome', 'LoanAmountToIncome']),  # Include log('IncomeBins')
        ('ord', ordinal_transformer, ordinal_features),
        ('nom', nominal_transformer, nominal_features)
    ])

# Create a logistic regression model
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('classifier', LogisticRegression())])

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Fit the model on the training data
```

```
model.fit(X_train, y_train)

# Predict probabilities on both train and test sets
y_train_pred = model.predict_proba(X_train)[:, 1]
y_test_pred = model.predict_proba(X_test)[:, 1]

# Calculate ROC AUC score for train and test sets
roc_auc_train = roc_auc_score(y_train, y_train_pred)
roc_auc_test = roc_auc_score(y_test, y_test_pred)

print(f"ROC AUC Train Score: {roc_auc_train}")
print(f"ROC AUC Test Score: {roc_auc_test}")
```

```
ROC AUC Train Score: 0.7561101473990344
ROC AUC Test Score: 0.7608207262207181
```

[9]:
```
Xt = test_df.copy()

# Drop unnecessary columns
Xt = Xt.drop(columns=['LoanID', 'LoanTerm'])

# Manually bin 'LoanAmount'
Xt['LoanAmount'] = pd.cut(Xt['LoanAmount'], bins=50, labels=False)

# Add a square term for 'Age'
Xt['AgeSquared'] = Xt['Age'] ** 2

# Manually bin 'Income' and apply logarithm transformation
Xt['IncomeBins'] = pd.cut(Xt['Income'], bins=income_bins, labels=False)
Xt['LogIncome'] = np.log(Xt['IncomeBins'] + 1)

# Create 'LoanAmount/Income' ratio feature
Xt['LoanAmountToIncome'] = Xt['LoanAmount'] / (Xt['Income'] + 1)

# Drop the 'Income' column
Xt = Xt.drop(columns=['Income'])

# Make predictions on the test data
predicted_probability = model.predict_proba(Xt)[:, 1]
```

[10]:
```
### PLEASE CHANGE THIS CODE TO IMPLEMENT YOUR OWN PREDICTIONS

# Combine predictions with label column into a dataframe
prediction_df = pd.DataFrame({'LoanID': test_df[['LoanID']].values[:, 0],
                              'predicted_probability': predicted_probability})
```

[11]:
```
### PLEASE CHANGE THIS CODE TO IMPLEMENT YOUR OWN PREDICTIONS

# View our 'prediction_df' dataframe as required for submission.
# Ensure it should contain 104,480 rows and 2 columns 'CustomerID' and␣
 ↪'predicted_probaility'
print(prediction_df.shape)
prediction_df.head(10)
```

```
(109435, 2)
```

[11]:

	LoanID	predicted_probability
0	7RYZGMKJIR	0.070862
1	JDL5RH07AM	0.030580
2	STAL716Y79	0.033927
3	S0OKKJ3IQB	0.207608
4	T99CWTYDCP	0.126099
5	OSNHFWV4UP	0.087341
6	S6ITP6LGYS	0.038490
7	A6I7U12IRJ	0.061565
8	8W6KY50JU4	0.081906
9	THFQ08OLMU	0.092887

PLEASE CHANGE CODE ABOVE TO IMPLEMENT YOUR OWN PREDICTIONS

1.8 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.

[12]:
```
# 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.'
```

[13]:
```
# 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.'
```

[14]:
```
# 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.'
```

[15]:
```
# 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.'
```

[16]:
```
# FINAL TEST CELLS - please make sure all of your code is above these test cells

## This cell calculates the auc score and is hidden. Submit Assignment to see␣
 ↪AUC score.
```

1.9 SUBMIT YOUR WORK!

Once we are happy with our `prediction_df` and `prediction_submission.csv` we can now submit for autograding! Submit by using the blue **Submit Assignment** at the top of your notebook. Don't worry if your initial submission isn't perfect as you have multiple submission attempts and will obtain some feedback after each submission!