Hanuma Pavani P

6-CSE-B

1PE17CS117

**TensorFlow:**

* It is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers.
* **TensorFlow** is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation. The core open source library to help you develop and train ML models.
* The basic building block of a neural network is the layer. Think of this layer as unstacking rows of pixels in the image and lining them up.

**Keras:**

* Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow.
* Keras is high Level API(Application Programmable Interface) that allow to build the Deep learning or Machine Learning models easily.
* The **Keras** library provides a convenient **wrapper** for deep **learning** models to be used as classification or regression estimators in **scikit**-**learn**.

**Keras API Project Exercise- LendingClub**

## The Data

We will be using a subset of the LendingClub DataSet obtained from Kaggle: <https://www.kaggle.com/wordsforthewise/lending-club>

LendingClub is a US peer-to-peer lending company, and is the world's largest peer-to-peer lending platform.

### Our Goal

Given historical data on loans given out with information on whether or not the borrower defaulted (charge-off), can we build a model that can predict weather or nor a borrower will pay back their loan? This way in the future when we get a new potential customer we can assess whether or not they are likely to pay back the loan. Keep in mind classification metrics when evaluating the performance of your model!

# Section 1: Exploratory Data Analysis

**OVERALL GOAL: Get an understanding for which variables are important, view summary statistics, and visualize the data**

* Since we will be attempting to predict loan status, create a countplot.
* **Create a histogram of the loan\_amnt column.**
* **Explore correlation between the continuous feature variables. Calculate the correlation between all continuous numeric variables using .corr() method.**
* Visualize this using a heatmap
* Explore almost perfect correlation with the "installment" feature. Explore this feature further. Print out their descriptions and perform a scatterplot between them. Does this relationship make sense to you? Do you think there is duplicate information here?
* Create a boxplot showing the relationship between the loan\_status and the Loan Amount.
* **Calculate the summary statistics for the loan amount, grouped by the loan\_status.**
* **Explore the Grade and SubGrade columns that LendingClub attributes to the loans. What are the unique possible grades and subgrades?**
* **Display a count plot per subgrade.**
* **It looks like F and G subgrades don't get paid back that often. Isloate those and recreate the countplot just for those subgrades.**
* **Create a new column called 'loan\_repaid' which will contain a 1 if the loan status was "Fully Paid" and a 0 if it was "Charged Off".**
* **Create a new column called 'loan\_repaid' which will contain a 1 if the loan status was "Fully Paid" and a 0 if it was "Charged Off".**

**Section 2: Data Pre-processing**

Section Goals: Remove or fill any missing data. Remove unnecessary or repetitive features. Convert categorical string features to dummy variables.

**Missing Data**

* Explore this missing data columns.
* What is the length of the dataframe?
* Create a Series that displays the total count of missing values per column.
* **Convert this Series to be in term of percentage of the total DataFrame.**
* **Examine emp\_title and emp\_length to see whether it will be okay to drop them. Print out their feature information using the feat\_info() function.**
  + The job title supplied by the Borrower when applying for the loan.\*
  + Employment length in years. Possible values are between 0 and 1 where 0 means less than

10 years and 1 means ten or more years.

* **How many unique employment job titles are there?**
* Realistically there are too many unique job titles to try to convert this to a dummy variable feature. Let's remove that emp\_title column.​
* Create a count plot of the emp\_length feature column. Sort the order of the values.
* Plot out the countplot with a hue separating Fully Paid vs Charged Off
* This still doesn't really inform us if there is a strong relationship between employment length and being charged off, what we want is the percentage of charge offs per category. Essentially informing us what percent of people per employment category didn't pay back their loan. There are a multitude of ways to create this Series. Once you've created it, see if visualize it with a bar plot.
* Charge off rates are extremely similar across all employment lengths. Go ahead and drop the emp\_length column.​
* TASK: Revisit the DataFrame to see what feature columns still have missing data.
* Review the title column vs the purpose column. Is this repeated information?
* The title column is simply a string subcategory/description of the purpose column. Go ahead and drop the title column.
* Find out what the mort\_acc feature represents. Create a value\_counts of the mort\_acc column.
* There are many ways we could deal with this missing data. We could attempt to build a simple model to fill it in, such as a linear model, we could just fill it in based on the mean of the other columns, or you could even bin the columns into categories and then set NaN as its own category.
* **Looks like the total\_acc feature correlates with the mort\_acc, this makes sense! Let's try this fillna() approach. We will group the dataframe by the total\_acc and calculate the mean value for the mort\_acc per total\_acc entry.**
* **Fill in the missing mort\_acc values based on their total\_acc value. If the mort\_acc is missing, then we will fill in that missing value with the mean value corresponding to its total\_acc value from the Series we created above. This involves using an .apply() method with two columns.**
* **revol\_util and the pub\_rec\_bankruptcies have missing data points, but they account for less than 0.5% of the total data. Go ahead and remove the rows that are missing those values in those columns with dropna().**

## **Categorical Variables and Dummy Variables**

**We're done working with the missing data. Now we just need to deal with the string values due to the categorical columns.**

* **List all the columns that are currently non-numeric.**
* **Let's now go through all the string features to see what we should do with**

**Term feature**

* **Convert the term feature into either a 36 or 60 integer numeric data type using .apply() or .map().**

### **Grade feature**

* **We already know grade is part of sub\_grade, so just drop the grade feature.**
* **Convert the subgrade into dummy variables. Then concatenate these new columns to the original dataframe. Remember to drop the original subgrade column and to add drop\_first=True to your get\_dummies call.**

### **verification\_status, application\_type, initial\_list\_status, purpose**

* **Convert these columns: ['verification\_status', 'application\_type','initial\_list\_status','purpose'] into dummy variables and concatenate them with the original dataframe. Remember to set drop\_first=True and to drop the original columns.**
* ​home\_ownership
* **Review the value\_counts for the home\_ownership column.**
* **Convert these to dummy variables, but**[replace](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.replace.html)**NONE and ANY with OTHER, so that we end up with just 4 categories, MORTGAGE, RENT, OWN, OTHER. Then concatenate them with the original dataframe. Remember to set drop\_first=True and to drop the original columns.**

### **Address**

* **Let's feature engineer a zip code column from the address in the data set. Create a column called 'zip\_code' that extracts the zip code from the address column.**
* **Now make this zip\_code column into dummy variables using pandas. Concatenate the result and drop the original zip\_code column along with dropping the address column.**

### **issue\_d**

* **This would be data leakage, we wouldn't know beforehand whether or not a loan would be issued when using our model, so in theory we wouldn't have an issue\_date, drop this feature.**​

### **earliest\_cr\_line**

* **This appears to be a historical time stamp feature. Extract the year from this feature using a .apply function, then convert it to a numeric feature. Set this new data to a feature column called 'earliest\_cr\_year'.Then drop the earliest\_cr\_line feature.**

## **Train Test Split**

* **Import train\_test\_split from sklearn.**

​

* **Drop the load\_status column we created earlier, since it’s a duplicate of the loan\_repaid column. We'll use the loan\_repaid column since its already in 0s and 1s.**

​

* **Set X and y variables to the .values of the features and label.**

## **Grabbing a Sample for Training Time**

* **Perform a train/test split with test\_size=0.2 and a random\_state of 101.**

**​**

## **Normalizing the Data**

* **Use a MinMaxScaler to normalize the feature data X\_train and X\_test. Recall we don't want data leakge from the test set so we only fit on the X\_train data.**

# Creating the Model

* **Run the cell below to import the necessary Keras functions.**
* Build a sequential model to will be trained on the data. You have unlimited options here, but here is what the solution uses: a model that goes 78 --> 39 --> 19--> 1 output neuron. Explore adding [Dropout layers](https://keras.io/layers/core/) [1](https://en.wikipedia.org/wiki/Dropout_(neural_networks)) [2](https://towardsdatascience.com/machine-learning-part-20-dropout-keras-layers-explained-8c9f6dc4c9ab)

​

* Fit the model to the training data for at least 25 epochs. Also add in the validation data for later plotting. Add in a batch\_size of 256.

# Section 3: Evaluating Model Performance.

* **TASK: Plot out the validation loss versus the training loss.**
* **Create predictions from the X\_test set and display a classification report and confusion matrix for the X\_test set.**
* **Given the customer below, would you offer this person a loan?**
* **Now check, did this person actually end up paying back their loan?**