**Lending Club Report**

**By**

**Nisha Muthukumaran**

## Introduction

LendingClub is a US peer-to-peer lending company, headquartered in San Francisco, California. It was the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission (SEC), and to offer loan trading on a secondary market. Lending Club operates an online lending platform that enables borrowers to obtain a loan, and investors to purchase notes backed by payments made on loans. Lending Club is the world's largest peer-to-peer lending platform

Lending Club enables borrowers to create loan listings on its website by supplying details about themselves and the loans that they would like to request. All loans are unsecured personal loans and can be between $1,000 - $40,000. On the basis of the borrower's credit score, credit history, desired loan amount and the borrower's debt-to-income ratio, Lending Club determines whether the borrower is credit worthy and assigns to its approved loans a credit grade that determines payable interest rate and fees. The standard loan period is three years; a five-year period is available at a higher interest rate and additional fees. The loans can be repaid at any time without penalty. Only investors in 39 US states are eligible to purchase notes on the Lending Club Platform. However, eligibility differs when purchasing notes on the secondary market, FolioFN. Borrowers from all but 2 US states are eligible to apply for a loan.

## DATA CLEANING AND EXPLORATORY ANALYSIS

There are numerous columns as a part of the data set. `150` columns and `1646801` are the number of rows. There are two parts to the problem.

1. Classify loan status – the classification problem
2. Interest rate – the regression problem

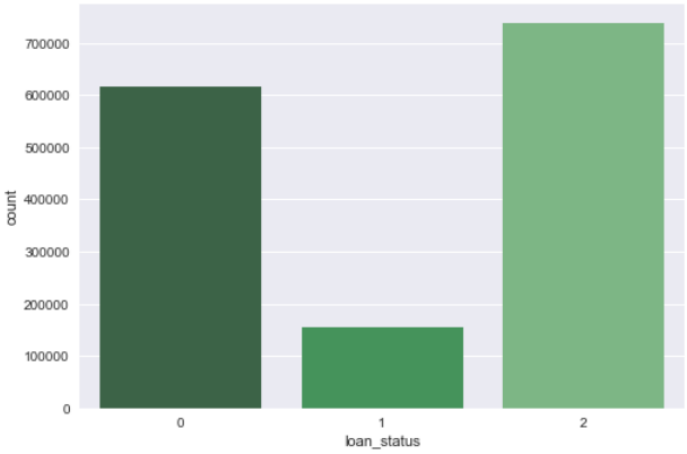
Data cleaning was done separately for both the problems because when predicting the interest rates, the features that need to be considered are those that will not be present already, for example the last payment on the loan will not be expected to predict the interest rates rather that feature will be more useful to predict the loan type. But the data cleaning explanation given here will be a summarized path that was taken across solving both the problems. The dataset requires only the loans that fall under the status of Current, Fully Paid and Charged Off.

**Distribution of the remaining data frame**

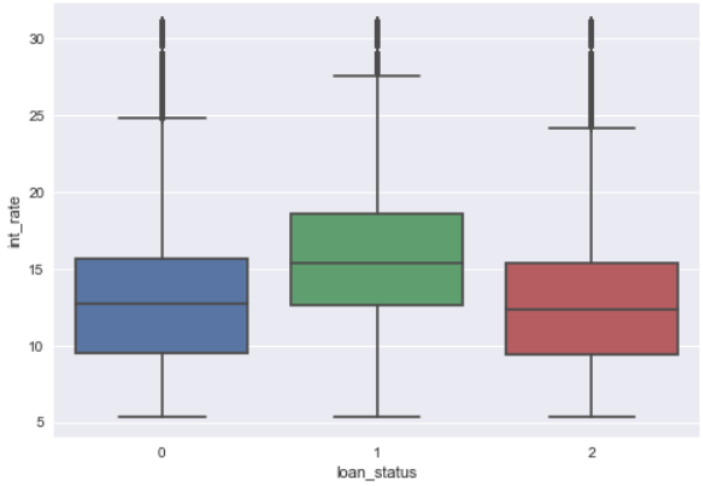
0 – Fully paid loans

1 – Charged-off loans

2 – Current loans

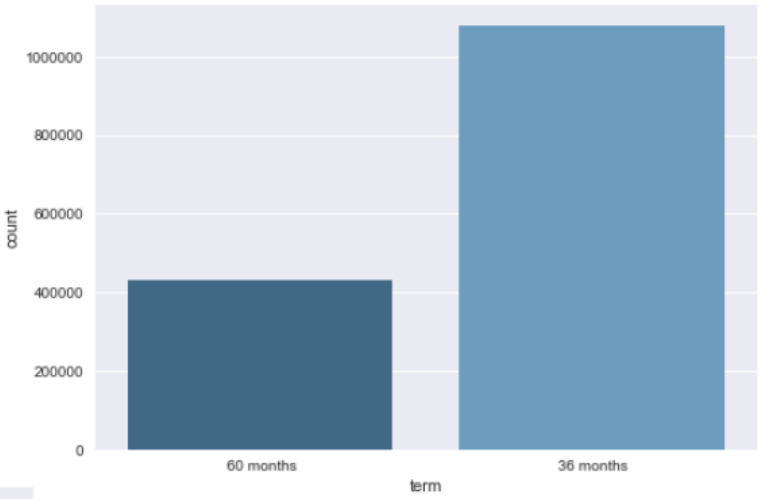


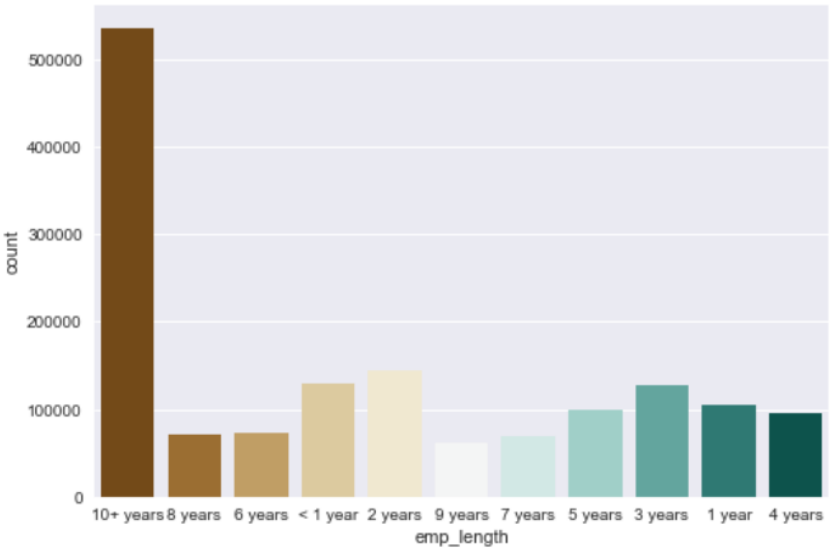
The first approach is to understand what these columns mean and how well they might contribute to the final model. Beginning the data cleaning by listing out the data types of all the columns in the data frame. Unique columns like IDs are the first ones that need to be removed from the data frame. The **loan_amnt** is the column that shows how much the borrower has asked for, this might be able to judge the loan status. **Id** and **member_id** can be dropped as they have no real predictive power. The next column is the **funded_amnt** which tells how much a borrower asked for but how much he/she was exactly given. There are `2088` cases where the two (**loan_amnt** and **funded_amnt**) are different. And there are `129557` cases where **funded_amnt** and **funded_amnt_inv** are different. All three columns are not needed to represent how much loan a person received, instead funded_amt can be retained. **Int_rate** is important for the second part of the problem



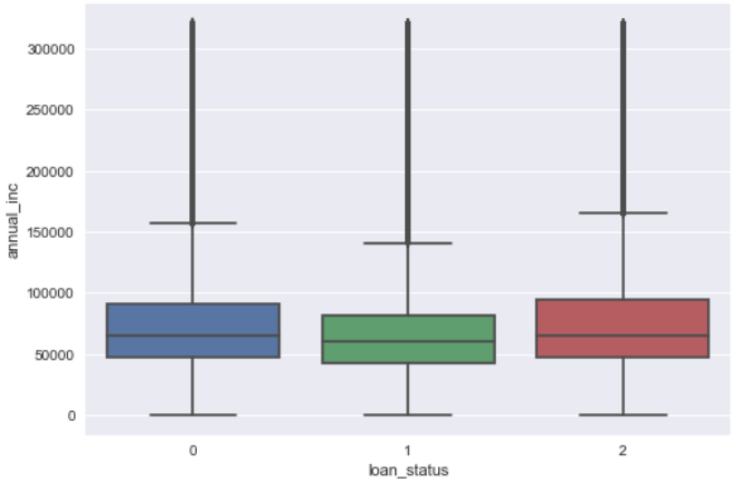

This graph shows that the interest rates for charged off loans are relatively higher when compared to the other two types of loans. It can be inferred from this that higher the interest rates, more are the defaulters.

Next, the **term**, **int_rate** and **installment** columns. **Term** has two values, one being a 36-month term and the other as a 60-month term. This can give the algorithm the idea of whether shorter loans are better or longer ones.



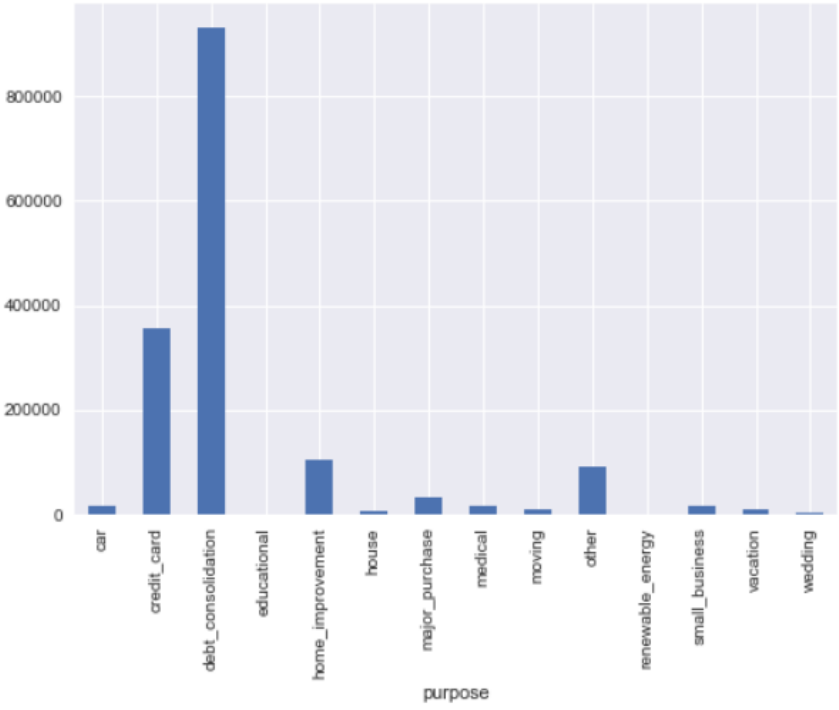




**Grade** and **sub_grade** are columns that might directly influence the final response, so for now it is removed. **Emp_title** contains `434231` unique values which might not be relevant to the model. **Emp_length** might be a factor which can contribute to the final model as it determines how long a person has funding to support payment of the loan. This can be converted to a categorical data as there are fewer categories. **Home_ownership** may also be a key determinant as a reason to pay-off a loan, so this predictor is retained.




**Annual_income** is a continuous column which is also a good variable to retain as it gives more insight to a person's incoming salary to pay off loans. As per this graph the loan owners who end up getting the loan charged off are primarily the ones with lower annual income when compared to the other two categories. This is an interesting insight.

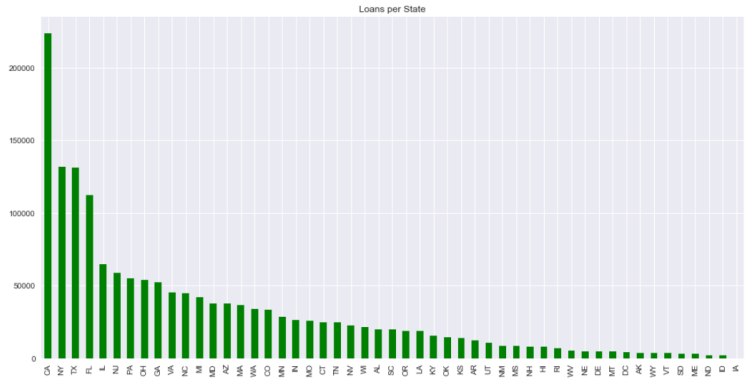
**Verification_Status** tells whether the person taking the loan has a verified source of income or not which is helpful for predicting. **Issue_d** talks about the month that the loan was funded in which may not really be important because it varies for different people and different reasons that the loan was taken**. Loan_status** is our main concern for the first part of the problem, as it represents the status of the loan. The focus is only on the loans that belong to either of the three statuses: Current, fully paid and charged off. These should be converted to categorical values and used as the response variable. **Pymnt_plan** has a very weak distribution amongst the yes and no response that is hence, so this can be dropped.

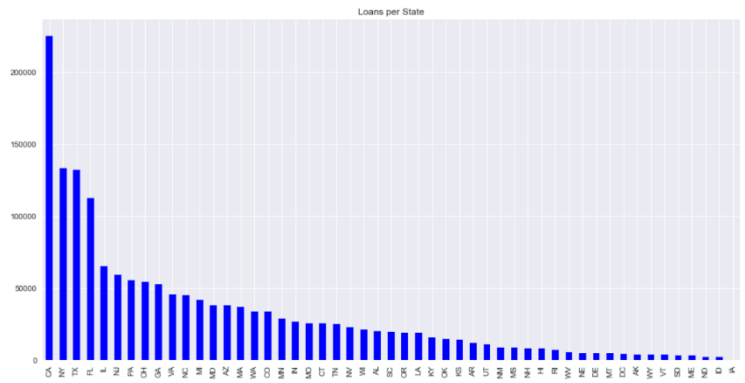



**Purpose** is an important column which describes for what reason the loan was taken which may help in determining which loans are more likely to default. This shows that many loans are taken for debt consolidations and to pay off credit cards.

**Title** should be dropped and so should **desc** as there are far too many unique values for the algorithm to consider. **Zip_code** needs to be removed as it contains junk values and **addr_state** can be retained to show patterns among where defaulters originate more from which state. California seems to have the highest number of defaulters but this is probably because California has the highest population of loan borrowers.

The left side graph shows the total number of loans taken per state and the right side graph shows the loans that were charged off across all the states.




**Dti** is very import as it helps in determining the debt to income ratio of the individual. Based on the distribution of the **delinq_2yrs**, it has only few unique values and hence can be considered. **Earliest_cr_line** talks about everyone's beginning time of the credit lines for the loan so this is an information that would come after the loan has been issued so it can be neglected. **FICO** score need to be retained as it is a form of a credit score, higher it is means that the person is more reliable and hence lesser their probability of defaulting. **Inq_last_6mths** may not be of importance. **Revol_util** talks about amount of credit the borrower is using out of what he was given so it may be important. **Mths_since_last_delinq** tells us whether this person has committed delinquency or not so it could be feature engineered into a yes or no column but has too many nulls to be considered. **pub_rec** can be made into a yes or no column as it shows behaviors of the individual. **mths_since_last_record** can be removed as it doesn't give important information. **Initial_list_status** has a fair distribution of fractional and whole loans and hence should be retained. **Total_acc** could be used for the second part of the problem to determine the interest rates. **Application_type** has very few uneven distributions of unique values so it can be removed. The remaining columns contain dates for each person on specifics on when they used the loan or they contain too many nulls in the majority hence they were removed.

**FEATURE ENGINEERING**

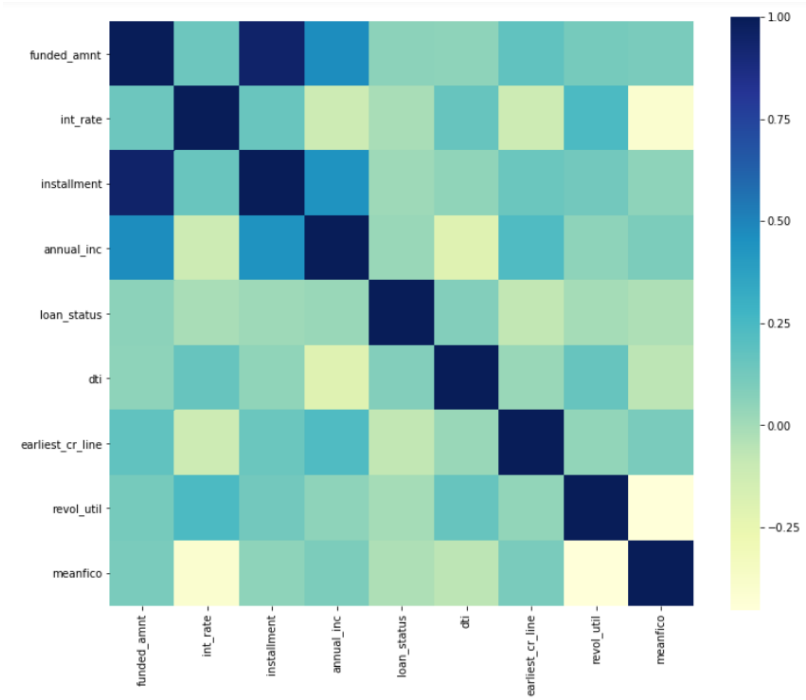
The dataframe is reduced to `1595158` rows, `150` columns by getting rid of **loan_status** that aren't fully_paid, charged_off or current. Columns that have NA's greater than 40% of the column length are dropped mercilessly as they do not add value to the business problem. The columns are reduced to 92.

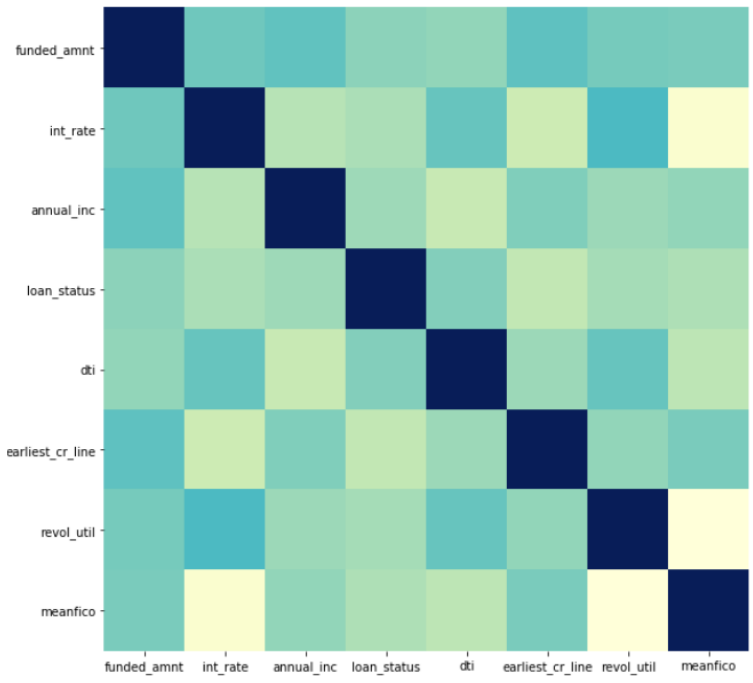
**Earliest_cr_line** tells us how long the person has been paying off the loan for, so larger the value higher is their probability of paying off the loan. So, the year is extracted from this column and calculated up to today. **FICO** value may not be helpful on their own so a mean range could be calculated from averaging the two **fico_range_high** and **fico_range_low.**

Remaining rows containing NA's are removed. After evaluating the distribution of unique values in the remaining columns distributions, a few columns are removed namely '**pymnt_plan', 'hardship_flag', 'disbursement_method', 'debt_settlement_flag'.** This is because the model may get biased due the distribution of majority value in these columns.

**CORRELATIONS**

On viewing the correlations amongst the remaining numerical columns, the below right graph is drawn with the existing data and it shows that installments are heavily correlated to the funded amount which means it should be taken care off or this correlation might interfere with the final model by doubling the effect of the variable on the response and giving more weightage to it. The left graph is after all possible correlations have been removed. This is the correlation matrix for the classification problem.



**ONE HOT ENCODING**

The remaining columns are converted from categorical to numerical values. This step is done to put categories that have smaller distributions into a single group rather than accounting for it separately. These are further one-hot encoded as the model doesn't support string values. The below columns are columns that contain categorical values.

| | |
|---|---|
| term | 60 months |
| emp_length | 10+ years |
| home_ownership | RENT |
| verification_status | Source Verified |
| purpose | debt_consolidation |
| addr_state | VA |
| initial_list_status | w |
| application_type | Individual |

These columns are further one hot encoded and then dummy variables are created out of it.

**TRAIN, TEST AND VALIDATE**

All the entries that have a loan status of current are made into the validation set and the remaining entries (charged-off and fully paid) are further split into train and test. This is done so that a value for accuracy score can be obtained.

The data is also standardized so that all the predictors are on one scale. The y response of loan status is also one hot encoded so that there are two columns, on for fully paid loans and the other for charged off loans.

**MACHINE LEARNING**

K-mean clustering was performed on the train and test set to see if there were any groupings in the data while predicting whether the loan would default or not. This prediction was then compared to the actual response for the test and train set and the accuracy was at 32% which clearly proves that K-means clustering with K=2 was not the way to go.

Logistic Regression was also performed to predict the loan classifications but this also did not perform well. The whole point being that Neural Networks tend to outperform any other form of advanced classification as it accounts for all the non-linearities and hidden patterns.

A simple multiple linear regression was also thrown on this data to which it did not react well when it came to predicting interest rates.

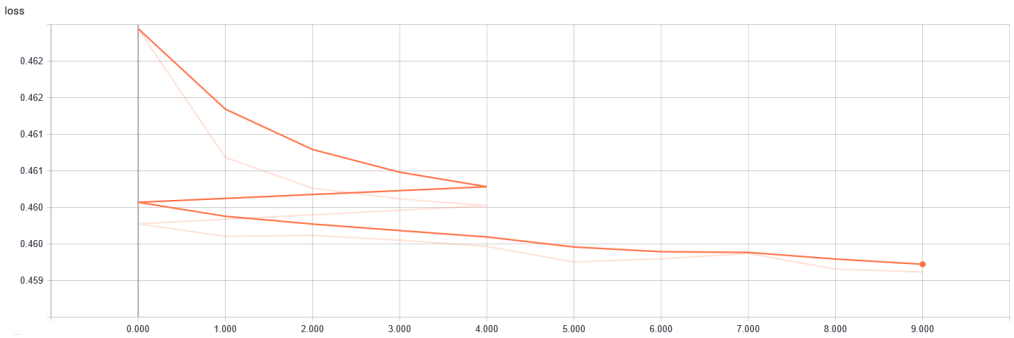
**MODEL BUILDING**

1. **For classification –**

Since the model is going to validate the data based only on loan statuses that are fully paid and charged-off, the model should not overfit this data because the validation set has to be tested on it. There are only two classes 0 – full paid and 1 – charged off loans so it is considered as a binary classification problem. And neural networks consider only numbers so they are already one hot encoded by this phase.

- The Keras sequential model is a stack of layers and this is set up by model = Sequential()
- The model needs to know what is the shape of the data coming into it and has to be prepared for it. This is defined as the shape of the X_train matrix. The layers should be dense which means that these are fully connected layers.
- The first layer or the input layer has the activation function as relu and the input shape is the number of columns in X_train. And it has a drop-out rate of 0.2. Drop out regularization is a way of preventing the model from overfitting. Dropout is a technique where randomly selected neurons are ignored during training. They are "dropped-out" randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.
- Input layer has a dimension of accepting an input (97, *) and outputs in the form of (*, 20).
- The next hidden layer accepts the previous layers input with dimensions of (20,*) and also outputs with a dimension of (*,20) and with the activation function as RelU
- The final output layer accepts the input from the previous layer and outputs only (*,2) with the final layer using a softmax activation function so that the output is a probability. This means that this will result in a score between 0 and 1, indicating how likely the sample is to have the target "1", or how likely the loan will be fully paid in this case.
- The optimizer used in this case is the stochastic gradient descent which means the gradient descent moves in random directions with a hope to attain the global minima.
- The learning rate is defined as 0.1, it could have been 0.01 also to slow down the rate at which the global minima is achieved.
- The loss function is also very gradual over 5 epochs that it was run for


- The loss function is very gradual but random after 5 epochs, over 10 epochs




- Accuracy on test data: **0.79**

```
Epoch 1/10
322852/322852 [==============================] - 27s 84us/step - loss: 0.4689 - acc: 0.7976 - val_loss: 0.4604 - val_acc: 0.796
6
Epoch 2/10
322852/322852 [==============================] - 27s 84us/step - loss: 0.4609 - acc: 0.7983 - val_loss: 0.4604 - val_acc: 0.796
3
Epoch 3/10
322852/322852 [==============================] - 27s 84us/step - loss: 0.4601 - acc: 0.7988 - val_loss: 0.4597 - val_acc: 0.796
2
Epoch 4/10
322852/322852 [==============================] - 27s 84us/step - loss: 0.4597 - acc: 0.7987 - val_loss: 0.4594 - val_acc: 0.797
0
Epoch 5/10
322852/322852 [==============================] - 27s 83us/step - loss: 0.4594 - acc: 0.7990 - val_loss: 0.4593 - val_acc: 0.796
9
Epoch 6/10
322852/322852 [==============================] - 27s 84us/step - loss: 0.4591 - acc: 0.7991 - val_loss: 0.4595 - val_acc: 0.796
5
Epoch 7/10
322852/322852 [==============================] - 27s 84us/step - loss: 0.4590 - acc: 0.7990 - val_loss: 0.4587 - val_acc: 0.796
5
Epoch 8/10
322852/322852 [==============================] - 27s 83us/step - loss: 0.4585 - acc: 0.7987 - val_loss: 0.4590 - val_acc: 0.797
6
Epoch 9/10
322852/322852 [==============================] - 27s 84us/step - loss: 0.4582 - acc: 0.7993 - val_loss: 0.4581 - val_acc: 0.798
9
Epoch 10/10
322852/322852 [==============================] - 27s 84us/step - loss: 0.4582 - acc: 0.7995 - val_loss: 0.4591 - val_acc: 0.796
```

## 2. For Regression –

The interest rate is being predicted in this case.

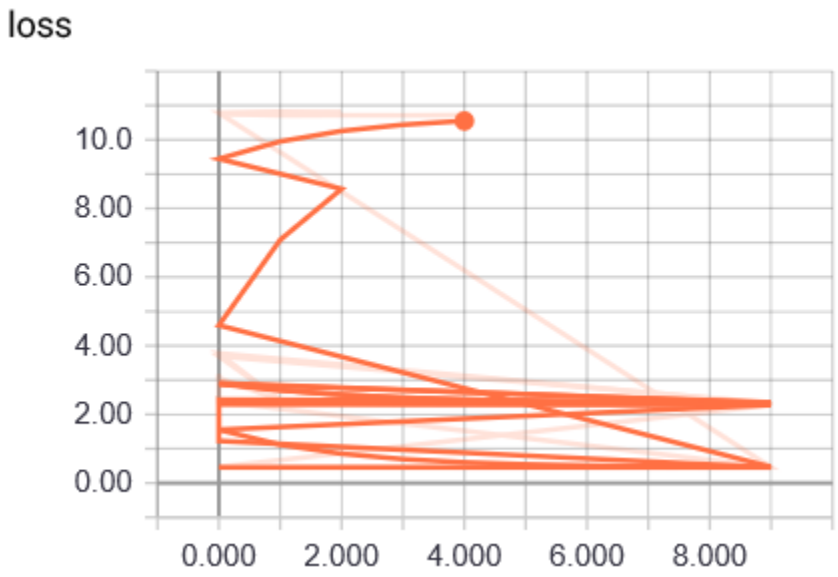
- The input layer has 124 inputs
- It also has 20 neurons in the next hidden layer and another 20 neurons in the hidden layer after it.
- Each hidden layer utilizes the relu activation function
- The output layer outputs just one output as this is a regression problem so it uses the linear activation function. Whatever is coming out of the last hidden layer is expected to be the output.
- The optimizer used for this is Adam. Adam optimizer has the ability to judge the learning rate at which the loss decreases.

```
Epoch 1/10
1051983/1051983 [==============================] - 67s 63us/step - loss: 12.4929
Epoch 2/10
   1440/1051983 [..............................] - ETA: 2:08 - loss: 11.1372

C:\Anaconda\lib\site-packages\keras\callbacks.py:526: RuntimeWarning: Early stoppi
not available. Available metrics are: loss
  (self.monitor, ','.join(list(logs.keys()))), RuntimeWarning

1051983/1051983 [==============================] - 63s 60us/step - loss: 11.1540
Epoch 3/10
1051983/1051983 [==============================] - 62s 59us/step - loss: 11.0160
Epoch 4/10
1051983/1051983 [==============================] - 62s 59us/step - loss: 10.9702
Epoch 5/10
1051983/1051983 [==============================] - 63s 60us/step - loss: 10.9069
Epoch 6/10
1051983/1051983 [==============================] - 61s 58us/step - loss: 10.8491
Epoch 7/10
1051983/1051983 [==============================] - 60s 57us/step - loss: 10.8474
Epoch 8/10
1051983/1051983 [==============================] - 61s 58us/step - loss: 10.8444
Epoch 9/10
1051983/1051983 [==============================] - 61s 58us/step - loss: 10.7986
Epoch 10/10
1051983/1051983 [==============================] - 61s 58us/step - loss: 10.8184
```

- This model has an R square value of 0.5213
- The tensorflow diagram was got after the above model was run




Although the model found its least loss, it looks like it fluctuated a bit as the epochs were pushing it to perform. This could have been reduced to lesser epochs to avoid it.