Risk Prediction Modeling for Credit Defaulters

Machine Learning Engineer Capstone Project
By Nishant Sharma
August 27, 2017

I. Definition

- **Project Overview:**

Credit risk refers to the risk that a current or potential customer may not repay back its dues on time. This kind of a customer is generally regarded as a credit defaulter and rising credit default rates remains a major issue with most banks and lending institutions everywhere.Accoridng to the S&P Dow jones indices ,credit default rates are at all-time high this year and are expected to be this way throughout the year of 2017.My Capstone project addresses this very real and current problem by building a robust credit risk prediction model which classifies potential defaulting customers. In order to do this, I will be using UCI Machine learning repository's dataset of credit card defaulters, which was donated to them by ranchers from Tamkang University in Taiwan.

This dataset has 30000 total instances and 24 relevant client information features including their age, gender, due payments, marital status, initial borrowed amount, past due's etc. The ultimate goal here is to train and compare multiple predictive models using supervised learning techniques and finally selecting an optimal model that best classifies defaulters and non-defaulters in the defaulter's column accurately.

- ## Problem Statement:

Making better credit lending decisions for its current and potential customers is highly important to a bank; the problem here is to classify which customer would default on its credit card payments with respect to the customer information the bank has already collected. My project will attempt to solve this problem by training multiple supervised classification models, which would use customer information as input features to predict whether they will default or not. The model, which best classifies the defaulters with respect to the chosen evaluation metrics (mentioned below) will be my pick for this project.

In order to achieve these goals, the following higher-level steps will be followed:

i. Download the dataset for UCI repository.
ii. Perform some exploratory data analysis and visualization of relevant features.
iii. Data preprocessing and Transformations.
iv. Train multiple supervised classifiers and feed forward deep neural network on the dataset to predict the default payment feature.
v. Evaluate and select an optimal model for credit risk prediction.

- ## Metrics:

In the model evaluation phase, I will have to compare the newly trained models to my benchmark model (logistic regression).In order to do that; I will be comparing each model with respect to accuracy first. Here accuracy is defined as ratio of correct predictions over total predictions. This is generally regarded ideal for binary classification tasks.

However, since our dataset here has imbalanced prediction variables (there are more non-defaulters than defaulters), this may lead a problem of accuracy paradox. Thus in order to avoid this issue we will also use Precision, recall and f1 score’s to evaluate and select an optimal model for this project. Precision here is defined as a measure of how many positive predictions are actually positive predictions; recall is defined as the portion of all real positive predictions that are correct and f1 score is defined as the harmonic mean of precision and recall.

Metrics formulas→

Accuracy =(True Positves+True Negatives)/Total Predictions

Precision=True Positives/(True Positives+False Positives)

Recall=True Positves/(True Positives+False Negatives)

F1 Score=2*(Precision *Recall)/(Precision+Recall).

II. Analysis

- **Data Exploration:**

This dataset can be downloaded from this UCI ML Repository; it consists of 30000 total instances and 25 features including-

- ID: Numerical unique id for each customer
- Limit_Bal:Credit amount given to each customer in NT dollars
- Sex: Gender of the customer represented as:(1=Male, 2=Female).
- Education: Education level of the customer; levels include :(1=Graduate School, 2=University, 3=High School and 4=Others).
- Marital status: Represents marital status of the customer: (1=married, 2=single, 3=others).
- Age: age of the customers in years
- Pay_0: Represents past history of repayments in September 2005.
- Pay_2: Represents past history of repayments in August 2005.
- Pay_3: Represents past history of repayments in July 2005.
- Pay_4: Represents past history of repayments in June 2005.
- Pay_5: Represents past history of repayments in May 2005.
- Pay_6: Represents past history of repayments in April 2005.

  Here each column from Pay_0 to Pay 6 contains the following values which represent duly and delayed payments: Pay duly for one month=-1, Pay duly for two months=-2,payment delay for one month=1, payment delay for two months=2, payment delay for nine months and above=9

  Note-Due to faulty numbering, Pay_1 is not present here, so I will have to simply rename Pay_0 to Pay_1.

- Bill_AMT1 to Bill_AMT 6: Represents bill statement amount in NT dollars starting from September 2005(Bill_AMT1) and going 6 months back to April 2005(Bill_AMT 6).

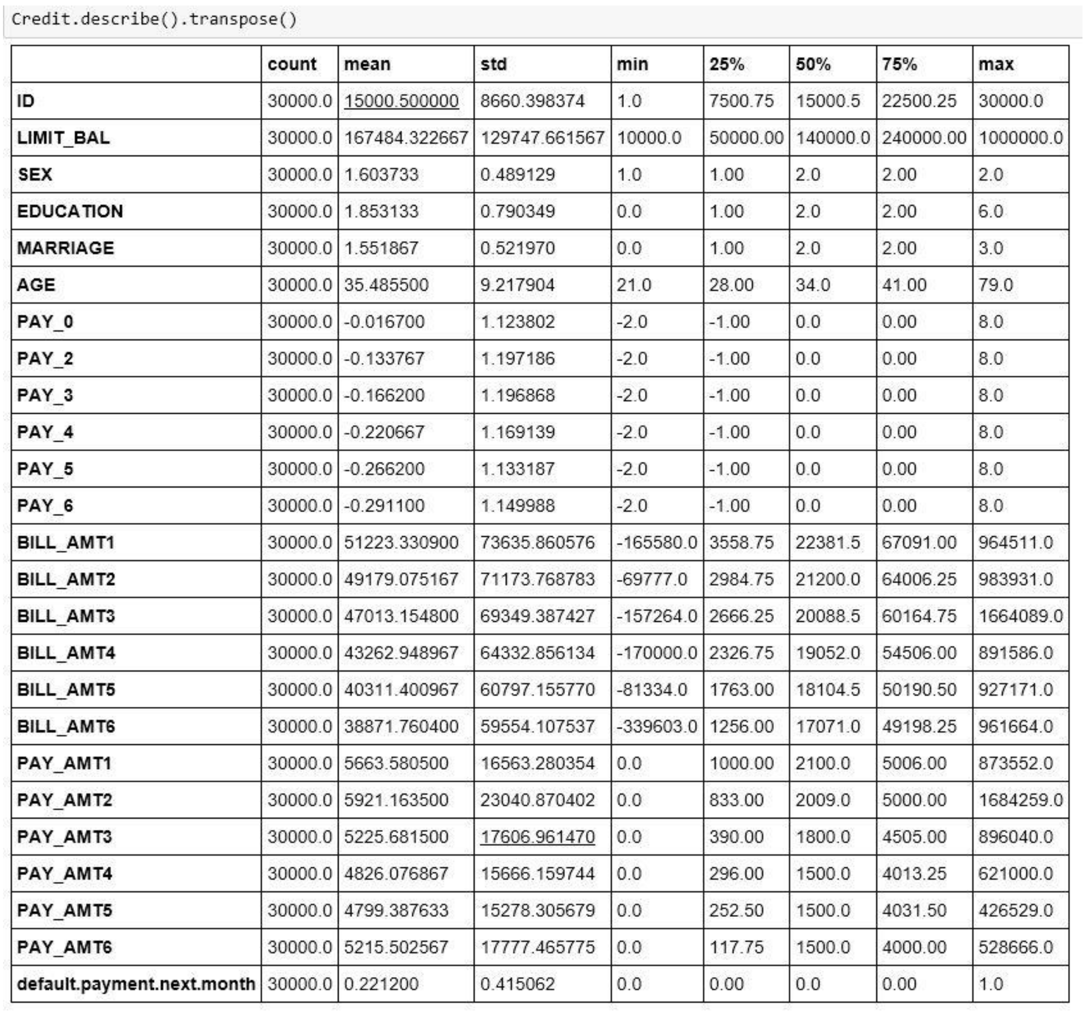
- PAY_AMT1 to PAY_AMT 6: Represents amount of previous payments in NT dollars starting from September 2005(Pay_AMT1) and again going 6 months back to April 2005(PAY_AMT6).

- Default.payment.next.month-This represents whether a particular customer has defaulted on its credit payments next month or not :(Defaulter=1, Non-defaulter=0).

➢ **Some Exploratory statistics:**

i. First we look at overall stats of all the features:

Table 1: Overall stats



```
Credit.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
ID	30000.0	15000.500000	8660.398374	1.0	7500.75	15000.5	22500.25	30000.0
LIMIT_BAL	30000.0	167484.322667	129747.661567	10000.0	50000.00	140000.0	240000.00	1000000.0
SEX	30000.0	1.603733	0.489129	1.0	1.00	2.0	2.00	2.0
EDUCATION	30000.0	1.853133	0.790349	0.0	1.00	2.0	2.00	6.0
MARRIAGE	30000.0	1.551867	0.521970	0.0	1.00	2.0	2.00	3.0
AGE	30000.0	35.485500	9.217904	21.0	28.00	34.0	41.00	79.0
PAY_0	30000.0	-0.016700	1.123802	-2.0	-1.00	0.0	0.00	8.0
PAY_2	30000.0	-0.133767	1.197186	-2.0	-1.00	0.0	0.00	8.0
PAY_3	30000.0	-0.166200	1.196868	-2.0	-1.00	0.0	0.00	8.0
PAY_4	30000.0	-0.220667	1.169139	-2.0	-1.00	0.0	0.00	8.0
PAY_5	30000.0	-0.266200	1.133187	-2.0	-1.00	0.0	0.00	8.0
PAY_6	30000.0	-0.291100	1.149988	-2.0	-1.00	0.0	0.00	8.0
BILL_AMT1	30000.0	51223.330900	73635.860576	-165580.0	3558.75	22381.5	67091.00	964511.0
BILL_AMT2	30000.0	49179.075167	71173.768783	-69777.0	2984.75	21200.0	64006.25	983931.0
BILL_AMT3	30000.0	47013.154800	69349.387427	-157264.0	2666.25	20088.5	60164.75	1664089.0
BILL_AMT4	30000.0	43262.948967	64332.856134	-170000.0	2326.75	19052.0	54506.00	891586.0
BILL_AMT5	30000.0	40311.400967	60797.155770	-81334.0	1763.00	18104.5	50190.50	927171.0
BILL_AMT6	30000.0	38871.760400	59554.107537	-339603.0	1256.00	17071.0	49198.25	961664.0
PAY_AMT1	30000.0	5663.580500	16563.280354	0.0	1000.00	2100.0	5006.00	873552.0
PAY_AMT2	30000.0	5921.163500	23040.870402	0.0	833.00	2009.0	5000.00	1684259.0
PAY_AMT3	30000.0	5225.681500	17606.961470	0.0	390.00	1800.0	4505.00	896040.0
PAY_AMT4	30000.0	4826.076867	15666.159744	0.0	296.00	1500.0	4013.25	621000.0
PAY_AMT5	30000.0	4799.387633	15278.305679	0.0	252.50	1500.0	4031.50	426529.0
PAY_AMT6	30000.0	5215.502567	17777.465775	0.0	117.75	1500.0	4000.00	528666.0
default.payment.next.month	30000.0	0.221200	0.415062	0.0	0.00	0.0	0.00	1.0

Above result inference-

Note-Categorical features are best explained by charts in next section.

a) Here by looking at the first "count" column we can see that all features have total 30000 count, which means that we have no missing values.

b) About 50% of customers get a credit (Limit_Bal) of about 14000 NT dollars, with highest being in the range of 240,000 to 1,000,000 dollars.

c) Average age of customers is about 34-35 and going all the way up to 75years.

d) Average credit bill to be repaid (Bill_AMT) is in range of about 17,000 to 22,000 across all 6 months. Here the negative values (min column) show that credit has been repaid.

e) Average payments (Pay_AMT) is in range of about 15,000 to 21,000 dollars across all 6months.

- **Exploratory Visualization:**

Since our focus here is to predict the defaulter's column, we will first see the distribution of defaulters/non defaulters-

Chart 1: Here we can see that non defaulters (0) are much higher than defaulters (1), which is realistic as for any profitable bank/financial institution non defaulters are generally higher than defaulters. However, this may affect our prediction model later.

```
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x1cba1684ef0>
```

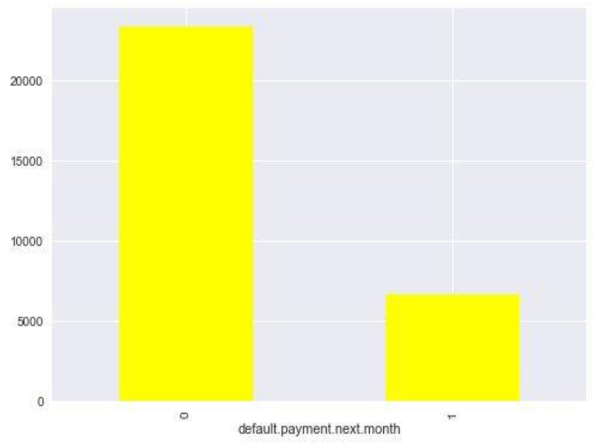


Chart 1: Default payment categorical distribution

Some Demographic/categorical variables visual analysis-

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e1bbbdb080>
```

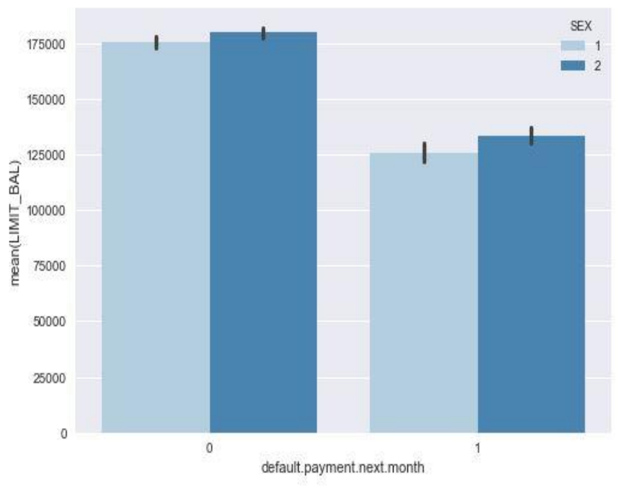


Chart 2: =LIMIT_BAL VS Default payment Barplot, segregated w.r.t Sex

Chart 2: Here we can see that non defaulters(0) generally have high average credit amount (LIMIT_BAL) and both females and Males get almost same credit and have similar default/non default counts.

Chart 3:Most of the customers here are University students followed by Grad and high school students.Thus,proportionately we have high default count among University students followed by the rest in same order, signifying little to no affect on credit defaults by education levels.

Additionally, we can also see a presence on unwanted categorical levels like 0, 5 and 6 which need to be dealt with.

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cba309e5c0>
```

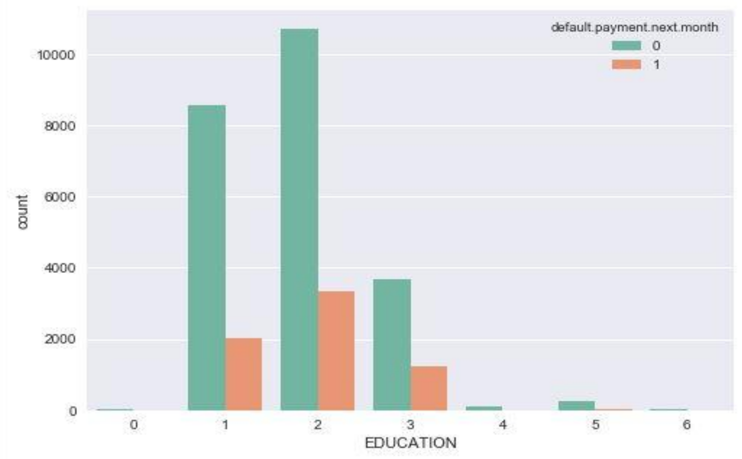


Chart 3: Educational distribution w.r.t default payments.

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cba3162390>
```

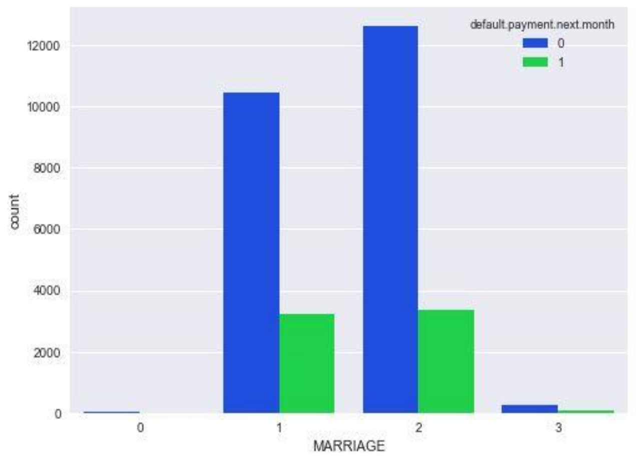


Chart 4: Single Customers (2) are more in count in comparison to married customers. Default Rates are almost the same for both single and married customers. Additionally, again we have some unwanted Marriage level(0).This needs to be dealt with later.

Chart 4: Marriage levels vs default payments

Chart 6: Past history of repayments distribution (Pay_6)

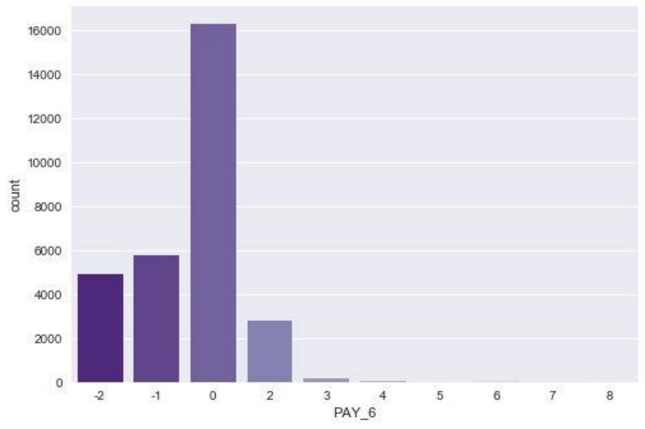


Chart 6: Here we compare past history of repayments distribution (Pay_6) for April 2005,from the chart here, we can see the most payments were made upfront(0),followed by duly payments for 1month and payment delays for 2months(2).Most of the other repayment for months May to September(Pay_5 to Pay_0),follow this similar pattern.

- **Algorithms and Techniques:**

Credit risk modeling is a binary classification problem, which can can solved using classical supervised learning techniques and advanced classifiers like ensemble methods, and feed forward neural networks can be used.

In this project, I will be training a basic benchmark model (Log Regression) and comparing its performance with the following classifiers:

i. **Random Forests-**Random Forest is a type of supervised learning algorithm and falls under ensemble methods. Here we have a collection of simple (weak) decision trees, and the final predictions are derived from aggregating predictions of each tree.
For example, in our dataset, a random forest classifier will train multiple small decision trees on different subsets of the training set, each tree will try to classify defaulters, and non-defaulters (1 or 0) .The final prediction will be based on aggregating vote of predictions made by each individual tree.

   Random Forests generally has a high average accuracy across different datasets and are less prone to overfitting, hence its ideal candidate to beat logistic regression (benchmark model).

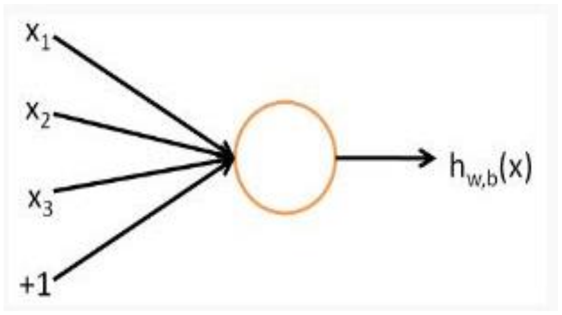
ii. **Gradient Boosting:** This is also a supervised learner and falls under ensemble methods. Just like random forests, it uses a combination of weak decision trees to arrive at its final prediction. However, instead of simply using aggregated voting, here each decision tree is built iteratively and its loss (misclassifications) is calculated. Based on a loss value it builds another small decision tree, which tries to reduce the errors made by the previous tree by changing some parameters.
This process keeps going iteratively until we have reached a certain threshold of acceptable error rate. This results in a more expressive, less prone to overfitting and an optimal classifier, again making it a perfect candidate for risk predictions.

iii. **Adaboosting**: This is also an adaptive boosting ensemble method, which is very similar to gradient boosting, and is regarded as a special case of gradient boosting. Here instead of calculating the loss via loss function, the classifier simply assigns more weights to the misclassified predictions/observations, and works on improving or boosting these predictions. Adaboost is known to achieve a good accuracy score for a wide variety of datasets and it may be an optimal model for credit risk predictions.

iv. **Weighted Voting Classifier:** This is a simple ensemble classifier, which allows us to combine three different classifiers for prediction. Here we can assign specific weights to each classifier, and a weighted average of the predicted class probabilities of each individual classifier is calculated and this becomes the final prediction.

The goal here is to first train the above-mentioned three classifiers and then add the three to this voting classifier, and assign specific weights to each classifier with respect its individual performance earlier. This may help in balancing out individual error rates of each classifer, potentially giving us a better accuracy/precision score.

v. **Feed Forward deep neural network:** These are also called as multilayer neural networks. In order further understand multilayer neural networks, I will first describe a “neuron”:



This is basically a computational unit, that takes in this case,x1,x2 and x3 and an intercept term(+1).

This outputs-> f(W^T *x).

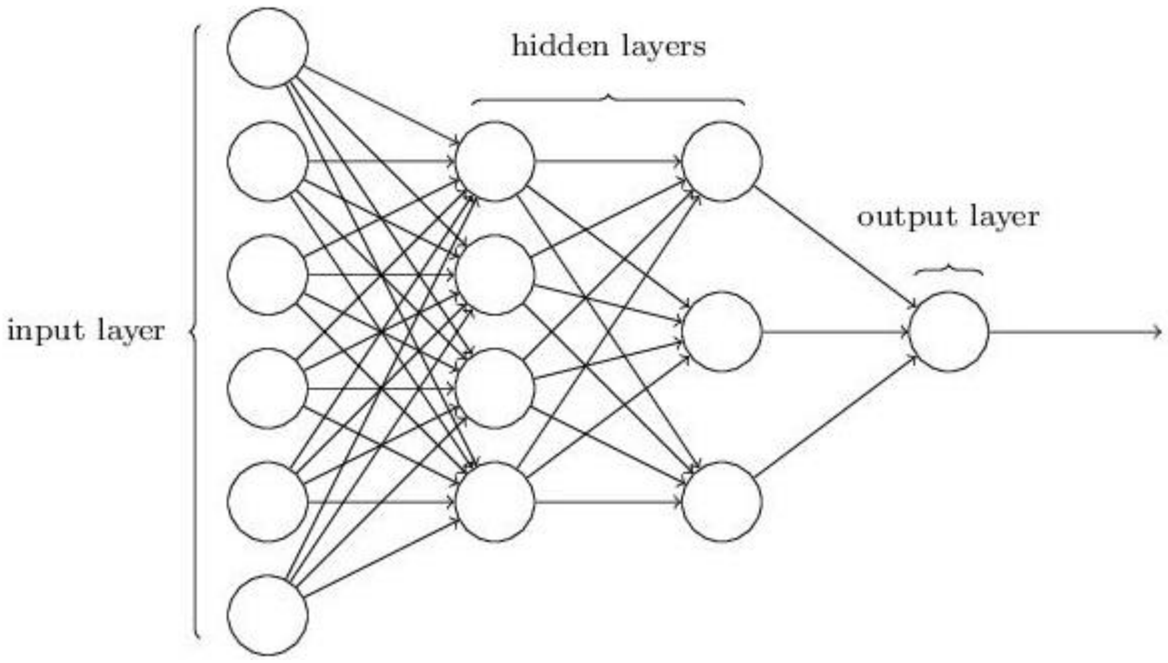
Here (W^T*x) is the weighted aggregated of all inputs plus bias. And function(f) is called the activation function.

There are many activation functions, which can be used like for example-

Sigmoid function works exactly like logistic regression.

$$f(x) = sigmoid(x) = \frac{1}{1+e^{-x}}$$

A Multilayer neural network consist of large number of these neurons, organized in layers like this-



Here initial learning involves sending a pattern observed in the dataset from the input layer through multiple hidden layers until it reaches the output layer. The output here is compared to the correct target class, and based on this comparison an error term is calculated for each node, which is then used to adjust the weights in the hidden layers, so that in the next cycle/iteration the output node value corresponds a little bit more to the correct target class values (in our case 1 or 0).The goal here is to perform several of these iterations (epochs) and get an optimal accuracy.

Additionally, we would also need to specify an optimization function which basically helps (in our case) minimize the error term or loss function. One of the most popular optimization algorithm is called Gradient descent, which updates the weights after each backpropagation by calculating the gradient of error function with respect to weights and updates the weights in the opposite direction of this gradient lost function with respect to the models parameters.

Feed Forward neural networks are known to achieve a higher accuracy with increasing layers and other changes in network topologies. In this project, I will attempt to do the same by using a SGD (Stochastic gradient descent) optimizer, which tries to minimize the loss function iteratively and is generally much quicker than the normal gradient descent optimizer.

Additionally, I will be using Relu(Rectified Linear Units**)** for the initial layers (according to research these are known to achieve an higher accuracy rate and the final layer will be sigmoid to classify 0 or 1(non defaulters or defaulters).

Relu formula-

$$f(x) = max(x, 0)$$

▪ Benchmark:

Logistic regression as mentioned above is essentially a sigmoid function, where we give input values(y) to get binary output(0 or 1),this makes it perfect for the task at hand here.

More importantly, Logistic regression is generally regarded as one of the best classification algorithms for these kind of tasks, and it is widely used for loan default modeling research. It has also achieved good accuracy scores about 70 to 80% for credit risk modeling on other datasets, as mentioned here.

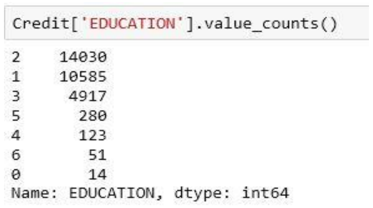
In this project I will be training a basic logistic regression model(benchmark model) for this dataset and the accuracy (or other evaluation metrics mentioned above) obtained after training and testing will be used as benchmark scores (threshold values) which the other classifiers need to beat to be considered as an optimal risk prediction model.

III. Methodology

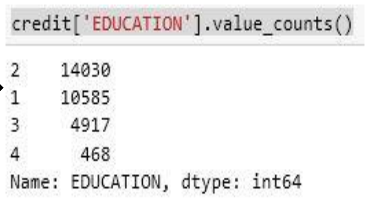
- **Data Preprocessing:**

As pointed out in the exploratory analysis, the following issues were dealt with:

- ID column was dropped as its unnecessary for our modeling.
- Pay_0 column name was changed to Pay_1.
- Education category levels corrected by changing 0, 5, 6 to 4(others).

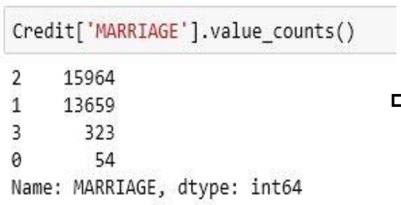


```
Credit['EDUCATION'].value_counts()

2    14030
1    10585
3     4917
5      280
4      123
6       51
0       14
Name: EDUCATION, dtype: int64
```

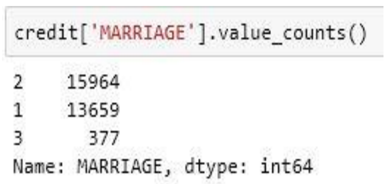


```
credit['EDUCATION'].value_counts()

2    14030
1    10585
3     4917
4      468
Name: EDUCATION, dtype: int64
```

- Marriage category levels corrected by changing 0 to 3



```
Credit['MARRIAGE'].value_counts()

2    15964
1    13659
3      323
0       54
Name: MARRIAGE, dtype: int64
```



```
credit['MARRIAGE'].value_counts()

2    15964
1    13659
3      377
Name: MARRIAGE, dtype: int64
```

Scales maybe in different ranges, hence feature scaling is necessary. In order to scale this uniformly I have used MinMaxScaler (from sklearn package) on the dataset, which scales all features to a range between 0 and 1.

Lastly, further preprocessing is not necessary, because this like any other financial dataset is mostly clean, and all categorical variables encoded, and as shown in further sections the accuracy/precision scores achieved with this are quite satisfactory.

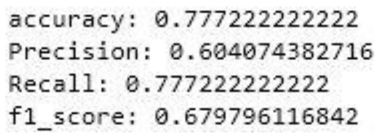
In addition, the issue of imbalanced classes as addressed earlier in chart 1 of visualization will not affect my model evaluation as my final evaluations will consist of stratified kfold validation and comparing performance based on weighted precision, recall and f1 score alongside with accuracy. This will ultimately give us an optimal model.

- **Implementation:**

The following steps were followed:

- Splitting the data into train and test set using sklearn's train_test_split: Here we will be keeping 30% as testing data, and 70% as training data, target variable will be our default payment column(0 or 1) and the rest will be used as input features.

- <u>Training Logistic regression (benchmark model) and note down its accuracy, precision, recall and f1 score on test set:</u>



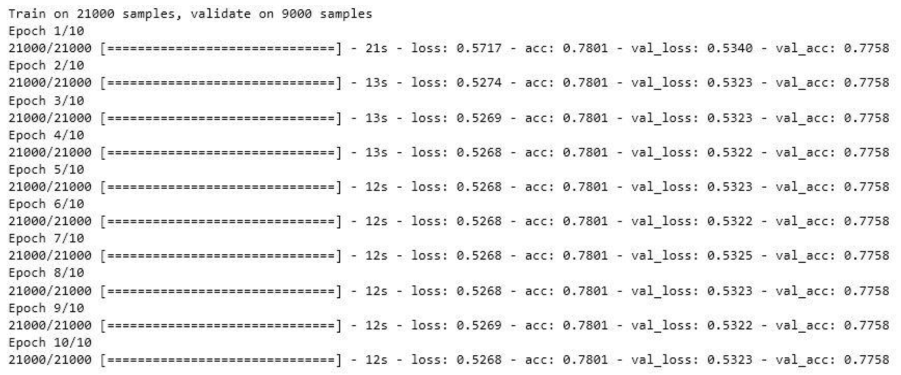
```
accuracy: 0.777222222222
Precision: 0.604074382716
Recall: 0.777222222222
f1_score: 0.679796116842
```

- <u>Training a Feed forward deep neural network using SGD optimizer on the same dataset and also calculating the accuracy on validation set(test_set)</u>:

    i. Using Keras (tensor flow backend) ,I trained a feed forward neural network on the Train set with test set as my validation set.

    ii. Topology of the Network includes:1 input layer(with 128 neurons,input_dimensions set to no. of features and "relu" as activation function);7 hidden layers with decreasing number of neurons and activation function as "relu" were used;1 Final output layer used sigmoid function for activation because we have a binary output predictor (1 or 0).

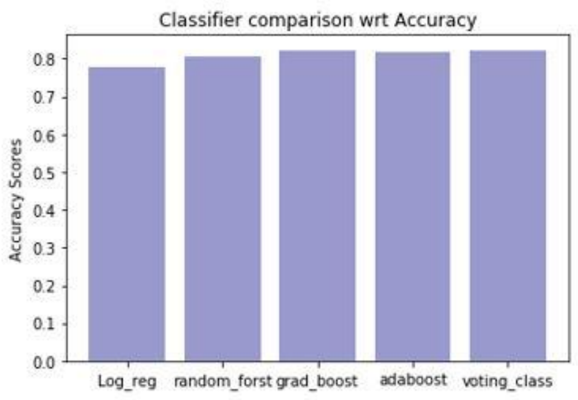
    iii. SGD optimizer was used with a low learning rate of 0.01 and decay of 1e-6.
    iv. After multiple trial and error, batch size of 100 and epoch of 10 was selected.
    v. Final Accuracy on Train test was 78% and test set was about 77.5%.

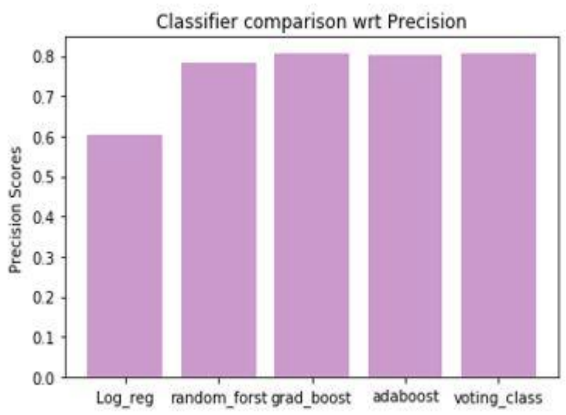


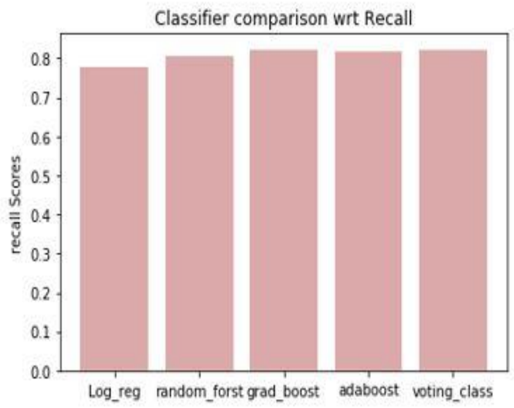
```
Train on 21000 samples, validate on 9000 samples
Epoch 1/10
21000/21000 [==============================] - 21s - loss: 0.5717 - acc: 0.7801 - val_loss: 0.5340 - val_acc: 0.7758
Epoch 2/10
21000/21000 [==============================] - 13s - loss: 0.5274 - acc: 0.7801 - val_loss: 0.5323 - val_acc: 0.7758
Epoch 3/10
21000/21000 [==============================] - 13s - loss: 0.5269 - acc: 0.7801 - val_loss: 0.5323 - val_acc: 0.7758
Epoch 4/10
21000/21000 [==============================] - 13s - loss: 0.5268 - acc: 0.7801 - val_loss: 0.5322 - val_acc: 0.7758
Epoch 5/10
21000/21000 [==============================] - 12s - loss: 0.5268 - acc: 0.7801 - val_loss: 0.5323 - val_acc: 0.7758
Epoch 6/10
21000/21000 [==============================] - 12s - loss: 0.5268 - acc: 0.7801 - val_loss: 0.5322 - val_acc: 0.7758
Epoch 7/10
21000/21000 [==============================] - 12s - loss: 0.5268 - acc: 0.7801 - val_loss: 0.5325 - val_acc: 0.7758
Epoch 8/10
21000/21000 [==============================] - 12s - loss: 0.5268 - acc: 0.7801 - val_loss: 0.5323 - val_acc: 0.7758
Epoch 9/10
21000/21000 [==============================] - 12s - loss: 0.5269 - acc: 0.7801 - val_loss: 0.5322 - val_acc: 0.7758
Epoch 10/10
21000/21000 [==============================] - 12s - loss: 0.5268 - acc: 0.7801 - val_loss: 0.5323 - val_acc: 0.7758
```

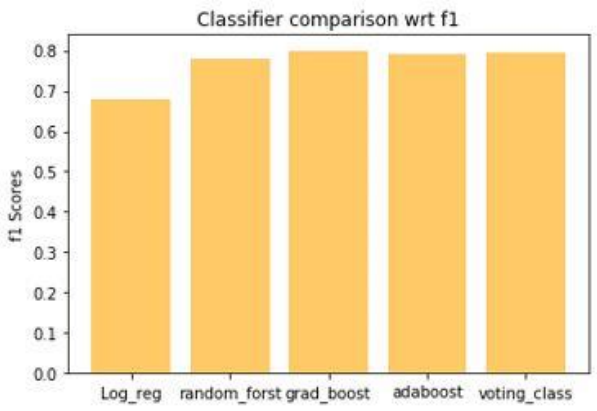
    vi. Precision score-0.60, recall-0.77 and f1 score of 0.67 was obtained.

- <u>Training Ensemble methods-adaboost, gradient boosting,randomforest and a voting classifier and checking their results on test set:</u>
  Note-Voting Classifier uses (Adaboost, Gradient Boost and Random Forest)







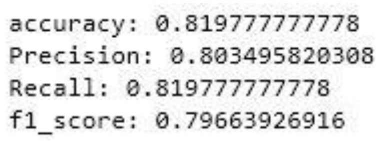


All ensemble methods are quite robust and perform better than log regression (benchmark model) and my earlier neural network with respect to Accuracy, precision, recall and F1score.

- <u>Choosing the final model for further refinement</u>:

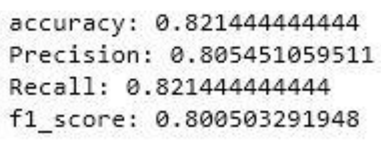
  Here I choose Gradient Boosting and Voting Classifier and for further refinement as they have very similar scores and are the highest among other classifiers above. Also intuition is that since Voting Classifier already uses all other three ensemble methods, I may get better scores if weighted and calibrated properly.

Voting Classifier:



```
accuracy: 0.819777777778
Precision: 0.803495820308
Recall: 0.819777777778
f1_score: 0.79663926916
```

Gradient Boosting:



```
accuracy: 0.821444444444
Precision: 0.805451059511
Recall: 0.821444444444
f1_score: 0.800503291948
```

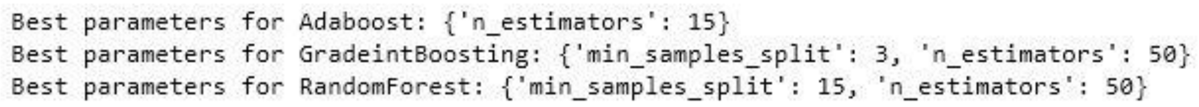
- **Refinement:**

  Now in order to get good results out of my voting classifier I have to first tune the parameters for each ensemble methods, which will be later, used in the voting classifier and to train gradient boosting separately.

  I will be using Grid Search Cross Validation technique with 10 fold cross validation. This basically, takes in a set of parameter values and exhaustively searches each fold to finally give us an optimal set of values.

  In order to maintain consistency, I just used the same set of possible parameter value sets for all classifiers, as they are all variants of decision trees.
  Parameters selected include:

- n_estimators(no. of sequential trees modeled):Higher number here gives better performance results as no. of trees increases.
- Min_sample_splits: Minimum number of samples required in a node for splitting, I have used this additionally to control overfitting, used only for Random Forest and Gradient Boosting.
  Final Values:

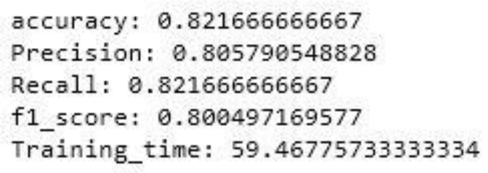


```
Best parameters for Adaboost: {'n_estimators': 15}
Best parameters for GradeintBoosting: {'min_samples_split': 3, 'n_estimators': 50}
Best parameters for RandomForest: {'min_samples_split': 15, 'n_estimators': 50}
```

CalibratedClassifierCV from sklearn (across 10 folds) was used to calibrate all three classifiers for my voting classifier and since Gradient Boosting, has a relatively high score in comparison to other models, I allocated it 2-weight value and other two have 1 weight value (1,1,2).

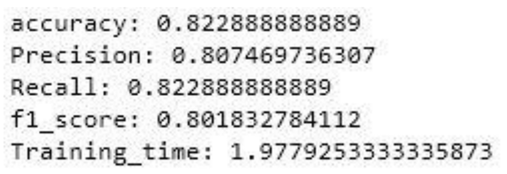
Additionally, I also used the above tuned parameters for Gradient Boosting and trained it separately.

Results from the above tuned models were calculated across our evaluation metrics and training time and we see improvements in both.



```
accuracy: 0.821666666667
Precision: 0.805790548828
Recall: 0.821666666667
f1_score: 0.800497169577
Training_time: 59.46775733333334
```

(Weighted Voting classifier)



```
accuracy: 0.822888888889
Precision: 0.807469736307
Recall: 0.822888888889
f1_score: 0.801832784112
Training_time: 1.9779253333335873
```

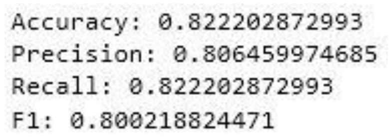
(Tuned Gradient Boost)

IV. Results

- **Model Evaluation :**

From the above results, we can see that weighted voting classifier and Gradient boost has almost the same scores with very small 0.1 to 0.01 difference in metric values. However, for my final model I will choose Gradient Boosting as its training time is very low in comparison to the voting classifier, which makes it relatively a better and more practical option here.

Additionally, in order to further validate robustness of my selected model I trained my tuned gradient boosting classifier again using 10 fold stratified cross validation:



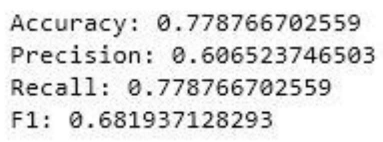
```
Accuracy: 0.822202872993
Precision: 0.806459974685
Recall: 0.822202872993
F1: 0.800218824471
```

Here we can see that our scores are very much the same as above (non-cross validated scores), Hence making this model and its parameters optimal for this problem.

- **Justification :**

After training log regression and after trying some parameter tweaks like adjusting C value (0.8), my scores had little to no effect on the final metrics.

Finally, I trained the base log regression using 10-fold cross validation just like the above weighted classifier and got much lower scores for all metrics:



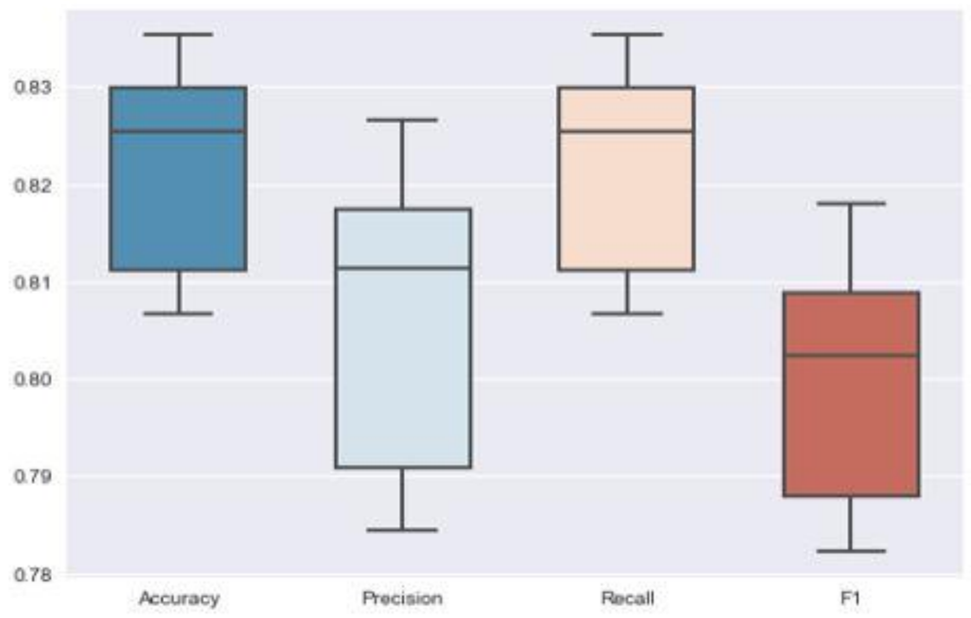
```
Accuracy: 0.778766702559
Precision: 0.606523746503
Recall: 0.778766702559
F1: 0.681937128293
```

Hence making my chosen much model better than this benchmark model.

V. Conclusion

- **Freeform Visualization :**
  The Ultimate goal of my capstone was to come with an optimal model, which does better across all metric values and not only accuracy.

  The following boxplot shows the distribution of metric values including (precision, accuracy, recall and f1) across 10 (cross-validated) folds of the entire dataset.



  We can see above that we are getting us amazing average scores (above 80%) with very little variation across all metrics, thus justifying the robustness of my model.

- **Reflection :**

  Having worked in the finance industry in the past, I was keen on using my newly developed machine learning knowledge on some financial datasets. After a lot of research I finally nailed my interest to credit risk modelling as this is an important problem to deal with and it affects all major financial institutions including insurance companies , banks etc.

Additionally, this particular dataset gave me opportunity to play around with different machine learning algorithms like feed forward deep neural nets, ensemble methods, parameter tuning of each algorithms etc. Which is exactly what I wanted in my capstone project is to have a comprehensive comparison of different classification techniques.

Following brief steps were followed in the project:

- **Downloading dataset and some visualizations of important features**: Here I noticed that my target class variables are actually imbalanced which is a problem as now I couldn't rely on just accuracy to evaluate my model, in order to get optimal results I had to use other evaluation metrics like precision, recall and f1 scores along with cross validation to make sure we have the correct model.
- **Some preprocessing including scaling of feathers and changing categorical scales.**
- **Training our benchmark model, feed forward neural network, ensemble methods:** Here I was curious to know that if we add more layers to a base feed forward neural network, will it actually perform better than log regression, so I played around with different layers and topologies and couldn't get the accuracy score higher than 77%which is basically the same as my base model.Also,to my surprise ensemble methods proved to be quite robust and I also played around with the idea of using a voting classifier of ensemble methods to check to see if it performs better ,but due to ridiculously high training times, I finally settled for Gradient boosting as its scores were slightly higher with very low training time.
- **Finally, I cross validated my final model and compared with the results of cross-validated log regression (benchmark model):** This finally proved that gradient boosting with tuned parameters does far better than base log regression.

Lastly, by doing this project solo, I am now confident of my skills in machine learning and ready to apply it in real world scenarios.

▪ Improvement :

Even though I feel I was quite thorough with my analysis, there are a few things that can be improved here including:

- Using Xgboost instead of Gradient boost may give use better results,as it's a robust algorithm and many kaggle completions are won using this.
- Using Under sampling techniques to balance out imbalanced classes may help improving accuracy slightly.
- Using an even deeper feed forward neural network with batch normalization and a higher momentum value (to make sure it's not stuck at local minima) may also improve the accuracy to a certain extend.

- **References used in the project :**

http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

https://medium.com/towards-data-science/decision-trees-and-random-forests-df0c3123f991

http://www.fon.hum.uva.nl/praat/manual/Feedforward_neural_networks_1_1__The_learning_phase.html

http://vinhkhuc.github.io/2015/03/01/how-many-folds-for-cross-validation.html

http://fastml.com/classifier-calibration-with-platts-scaling-and-isotonic-regression/

http://scikit-learn.org/0.15/auto_examples/grid_search_digits.html

https://svds.com/learning-imbalanced-classes/

https://stats.stackexchange.com/questions/117643/why-use-stratified-cross-validation-why-does-this-not-damage-variance-related-b

https://keras.io/optimizers/

https://keras.io/models/sequential/

http://seaborn.pydata.org/generated/seaborn.boxplot.html#seaborn.boxplot

https://jmetzen.github.io/2015-04-14/calibration.html