Facebook Comment Volume

I have done the assignment in **Python** for the below two datasets

1. Facebook comments count prediction
2. Credit fraud

Packages used for Neural Networks and K-Nearest Neighbour

ANN – TensorFlow using Keras KNN – KNNClassifier

Dataset Overview

In this project, we are trying to classify whether 10 or more comments will be posted. After analysing dataset, I found 10 best feature which can be used for classification problem. The detailed analysis can be found below:

Data Pre-processing

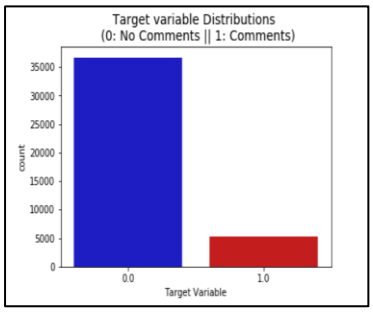
Our features are heavily skewed, so we need to normalize our dataset. Further Analysis shows how imbalanced is our original dataset! Most of the posts don't receive comments. If we use this DataFrame as the base for our predictive models and analysis, we might get a lot of errors and our algorithms will probably overfit since it will "assume" that most posts don't receive comments. But we don't want our model to assume, we want our model to detect patterns that give signs of comments.

There are two techniques –

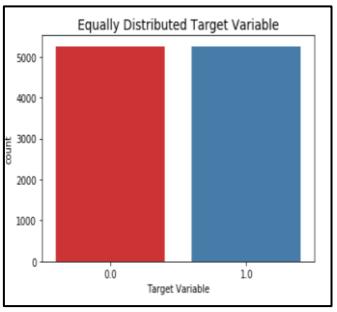
- ✓ SMOTE (over sampling) which understands the distribution and create more sample.
- ✓ Under sampling which instead of increasing sample size, creates sub-sample where we have equally distributed Target Variable.

I chose Under sampling because this dataset contains 50k datapoints and training model after oversampling will consume lot of time. So, to save time, we chose this technique

Before Under Sampling



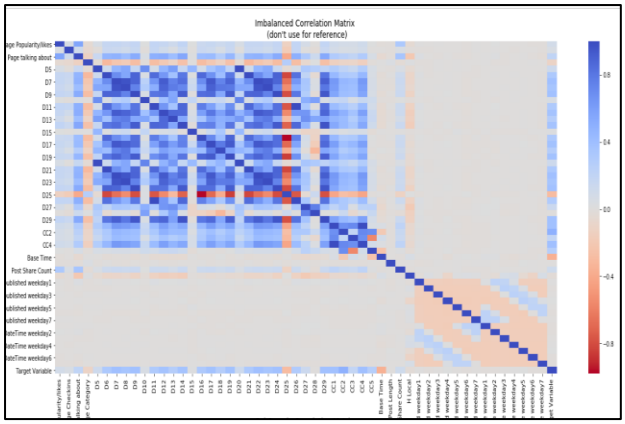
After Under sampling



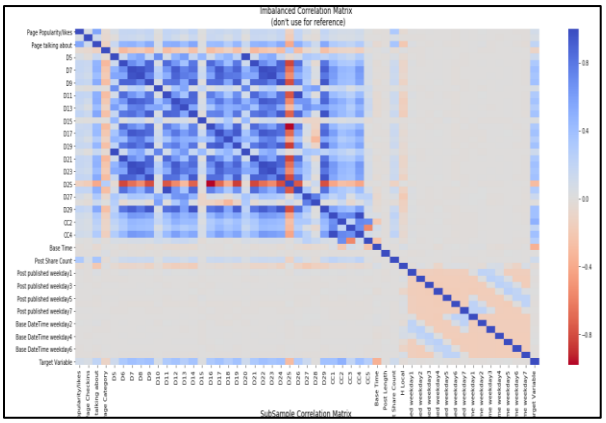
Correlation matrices are the essence of understanding our data. We want to know if there are features that influence heavily in whether a specific target variable is a comment. Below correlation matrix shows that which is dataframe is better with regards to comment posts.

We can cearly observe below after Under Sampling, We have equally distributed Target Variable.

Imbalanced correlation matrix



Balanced correlation matrix



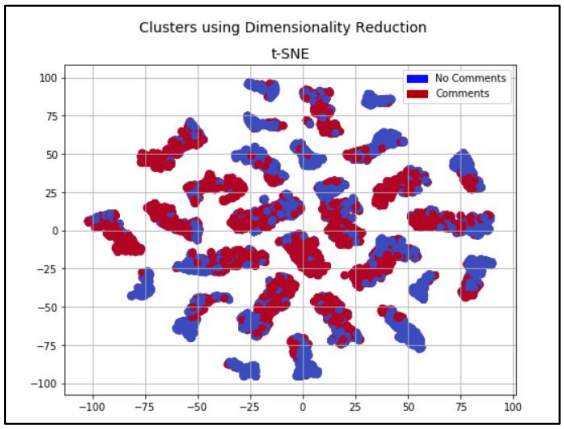
I have already checked the correlation between Target Variable and other features,

Feature Selection

After using correlation box plot between independent and target variables, step function and RFE model, the chosen independent features which best represents Target Variable are as follows:

D7, D9, D17, D22, CC1, CC2, CC3, CC4, CC5, Base Time

Dimensionality Reduction: - clustering of comments and No-comments



Support Vector Machine

Train Dataset

SVM Type	Linear SVM	RBF SVM	Poly SVM
Accuracy	86.5%	89.9%	91%
Computation Time	2.5 s	2.5 s	2.6 s

Test Dataset

SVM Type	Linear SVM	RBF SVM	Poly SVM
Accuracy	86.7%	89.8%	90.4%
Computation Time	0.74 s	0.92 s	0.84s

As we can see above, polynomial SVM with 3 degree best provides the accuracy, hence we can conclude that Poly SVM is better than linear and RBF model.

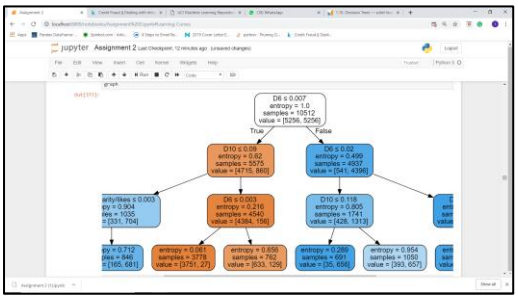
Decision Tree

To perform Decision tree, I have used DecisionTreeClassifier and grid search to find best parameters which best describe the Target Variable.

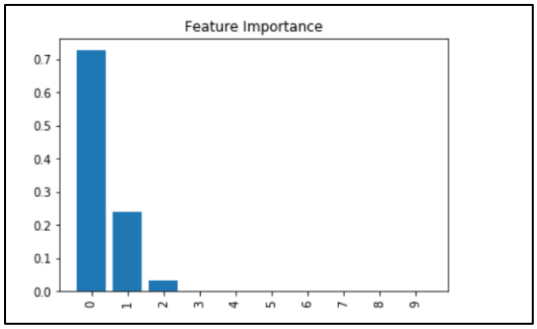
Train Dataset

Accuracy	Computation Time
91.58%	0.57 s

Decision Tree plot



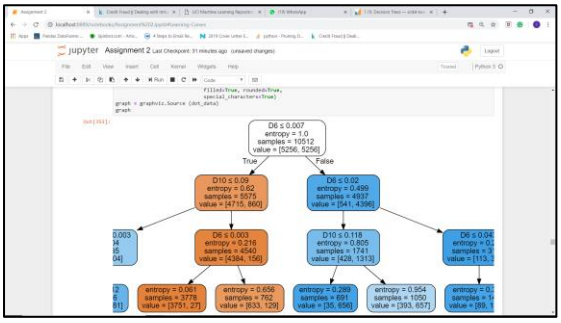
Features Importance by GINI as criteria



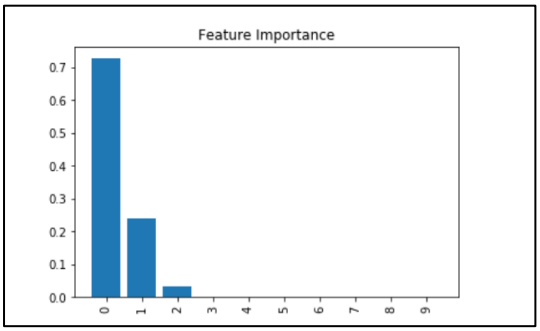
Test Dataset

Accuracy	Computation Time
90.68%	0.57 s

Decision Tree plot



Features Importance by GINI as criteria



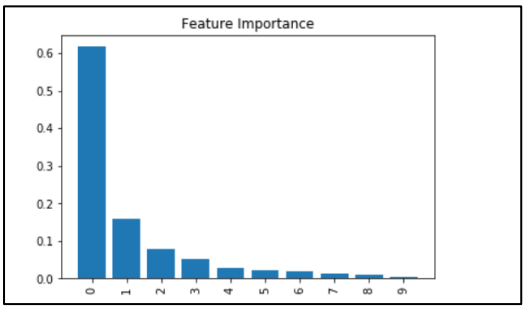
XGboost Boosted Decision Tree

To perform Decision tree, I have used XGBClassifier and grid search to find best parameters which best describe the Target Variable

Train Dataset

Accuracy	Computation Time
93.2%	0.57 s

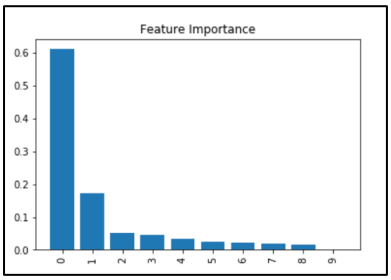
Features Importance by XGboost Algorithm



Test Dataset

Accuracy	Computation Time
93.16%	5.2 s

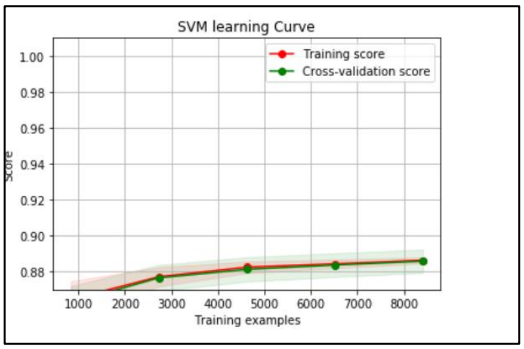
Features Importance by XGboost Algorithm



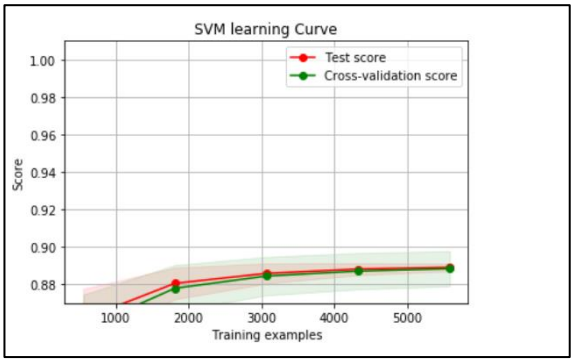
Learning Curves

SVM with best parameters vs Training examples

Train Dataset



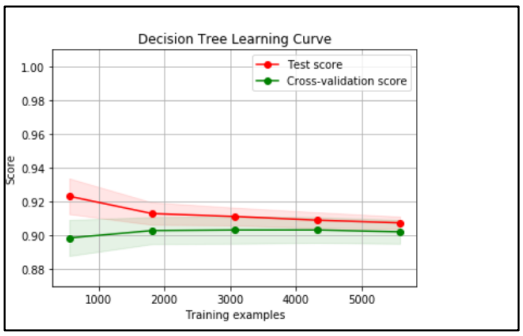
Test Dataset



Decision Tree with best parameters vs Training examples

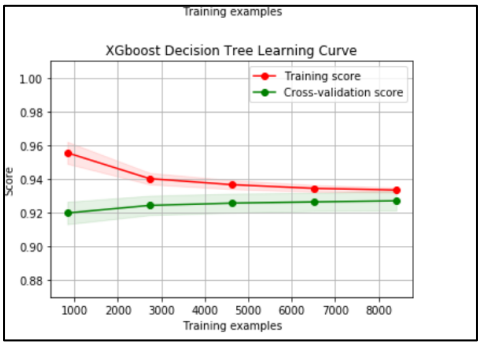
Train Dataset

Test Dataset

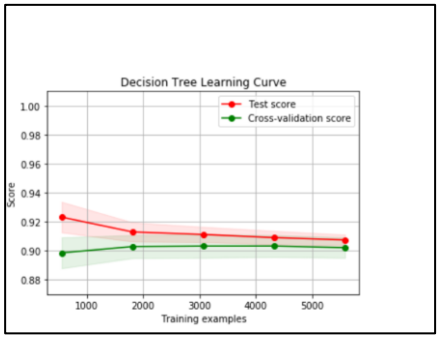


XGboost Decision Tree with best parameters vs Training examples

Train Dataset



Test Dataset



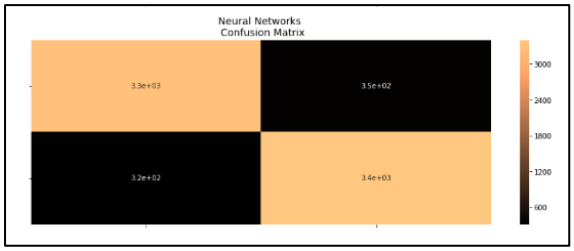
Neural Networks

Neural Networks works great for classification problem. The model run 6 times for different layer sizes and no of neurons to best parameters.

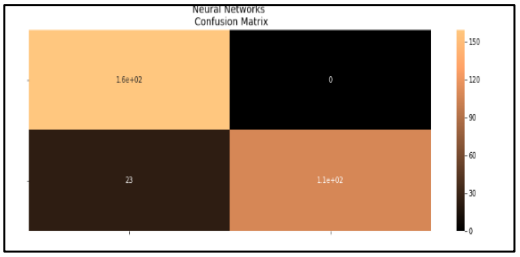
`Optimized ANN Accuracy on training dataset = 90.89% and 89.22% for test dataset.`

Confusion Matrix

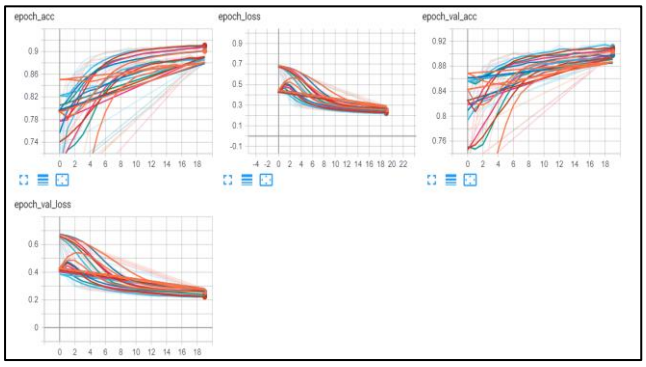
Training Dataset

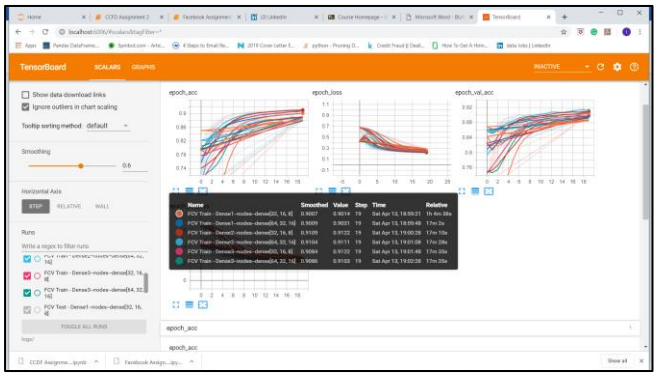


Test Dataset



Below are the learning curves of accuracy vs epochs and loss vs epochs for both training and validation dataset.





Inferences from Neural Network figures:

- Epoch vs accuracy plot shows that as the number of propagations increase the accuracy is changing. In this case epoch of 8 is good after which curve is flattening (with little increase in accuracy or decrease in loss).
- Second figure shows that ANN with 2 and 3 dense layers with 32, 16 and 8 neurons has the highest accuracy of 91.22%. It means that 2 dense layers with 32 and 16 neurons are more than enough to classify Target Variable.

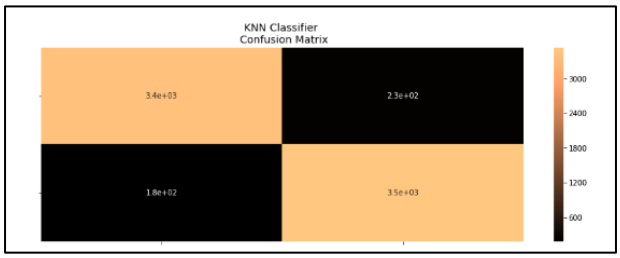
KNN

- For KNN I have used KNeighborsClassifier. The classifier took 1s to run. (The time is less because I have run KNN code multiple times and Jupyter Notebook saves the result, so, when we run model next time, it takes less time than before)
- Also, I have used grid search to find the best KNN parameters and used it for prediction

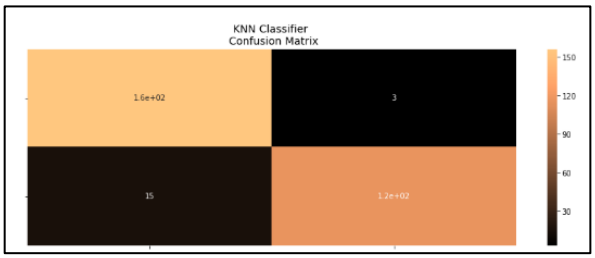
`KNN Accuracy on training dataset = 94.4% and 90.17% for test dataset.`

Confusion Matrix

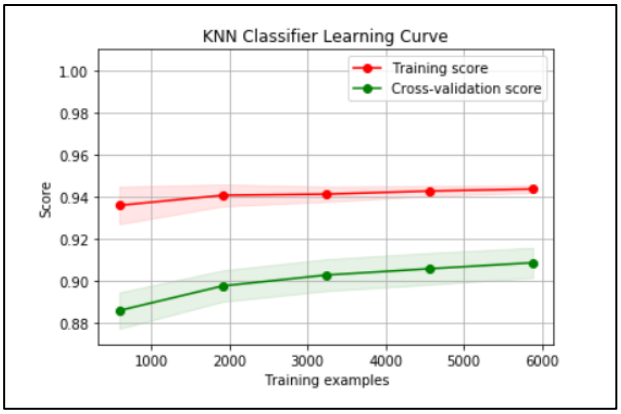
Training Dataset



Test Dataset



KNN Learning Curve of Accuracy vs Training examples for Training and Validation dataset.

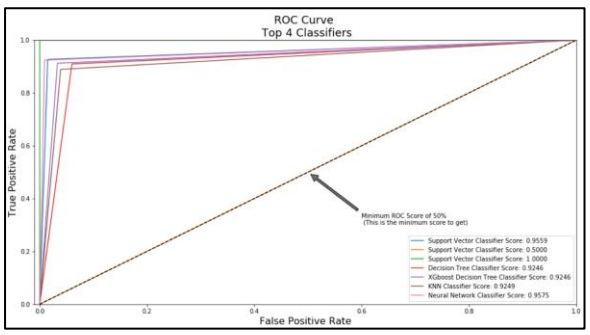


From the plot we can see that there is a gap between training and cross-validation hinting high variance.

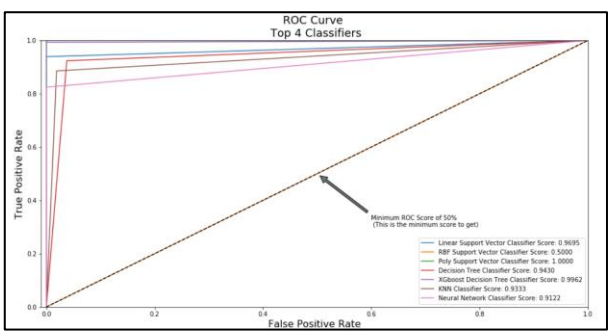
Comparisons between various models

ROC Curve

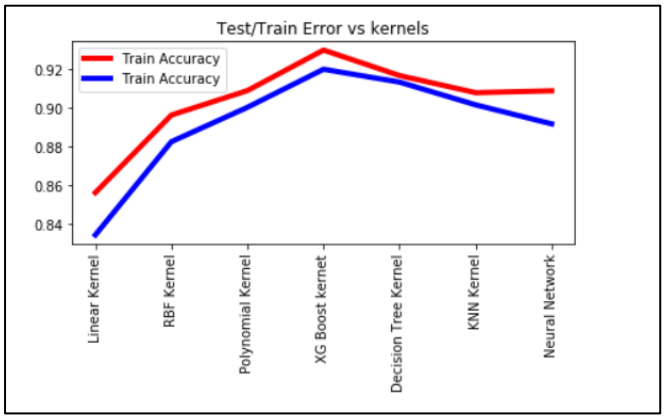
Train Dataset



Test Dataset



- Looking at the ROC curve, we can notice that for Facebook dataset Polynomial support vector machine is the best classifier.



- At last, I have plotted the training and testing error for different algorithms. Above figures shows that XGboost Decision tree is best classifier.

From above analysis, we can conclude that Target variable does not have linear trend and polynomial with degree 3 and XGboost decision tree are best at classifying comments.

Instructions to run above code:

- TensorFlow package needs to be installed to model Neural Network
- Please Visit https://www.kaggle.com/mlg-ulb/creditcardfraud site to download the dataset.
- To view Learning curves for this one is displayed using tensorboard, go to anaconda prompt, change directory to logs folder then type tensorboard –logdir=logs/. Once done, open your browser and type localhost:6006.