INTERIM REPORT – TEMPLATE

Your interim report is also the milestone-1. It should Summarize the 1st 3 deliverables mentioned in the project document. The report structure and brief requirements of each section is listed below.

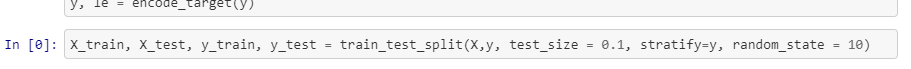
1. Summary of problem statement, data and findings Every good abstract describes briefly what was intended at the outset, and summarizes findings and implications.

2. Summary of the Approach to EDA and Pre-processing Include any insightful visualization you have teased out of the data. If you’ve identified particularly meaningful features, interactions or summary data, share them and explain what you noticed. Visual displays are powerful when used well, so think carefully about what information the display conveys.

3. Deciding Models and Model Building Based on the nature of the problem, decide what algorithms will be suitable and why? Experiment with different algorithms and get the performance of each algorithm.

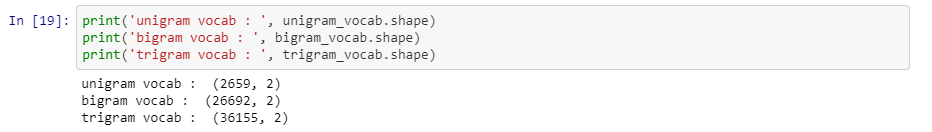
After cleansing the data, we tried different machine learning models namely, Naïve Bayes, Logistic Regression, SVM Classifier, Random Forest, XGBoost and KNN.

For modeling we have taken 90% train and 10% test data for modeling.

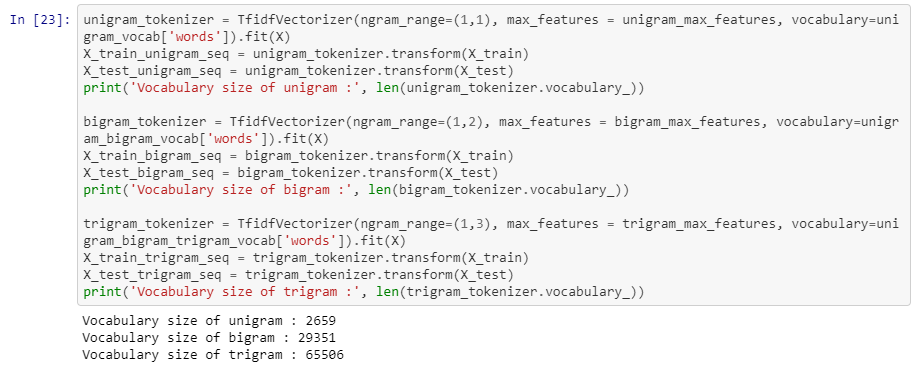


Since it is a classification problem we have used the aforementioned algorithms.

Modeling was done on uni-gram, bi-gram and tri-grams. Below is the vocab size for all the three.



Then we calculated the Tf-Idf Vectorizer for Uni-gram, bi-gram and tri-gram.



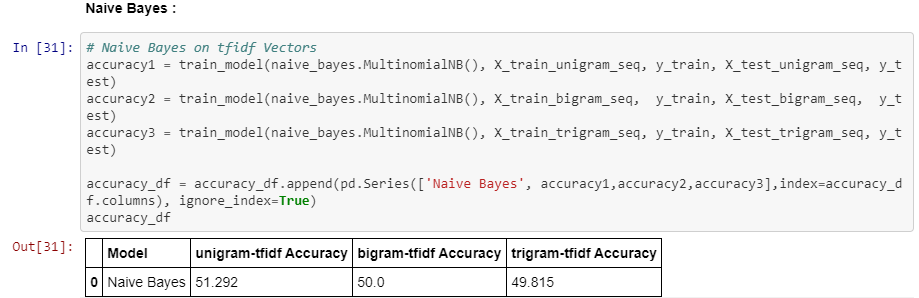
We then modeled the uni-gram, bi-grams and tri-grams vectors.

Below are the models used and their corresponding accuracy scores.

**Naïve Bayes**

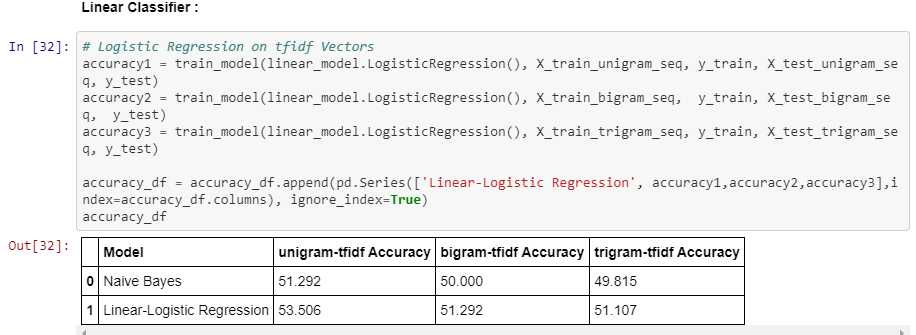
Naïve Bayes is a fast algorithm for classification problem. This algorithm is a good fit for real-time prediction, multi-class prediction, recommendation system, text classification, and sentiment analysis use cases. Naive Bayes Algorithm can be built using Gaussian, Multinomial and Bernoulli distribution. This algorithm is scalable and easy to implement for the large data set.

Due to its better performance with multi-class problems and its independence rule, Naive Bayes algorithm perform better or have a higher success rate in text classification, Therefore, it is used in [Sentiment Analysis](https://www.educba.com/sentiment-analysis-social-media/) and Spam filtering.



**Logistic Regression**

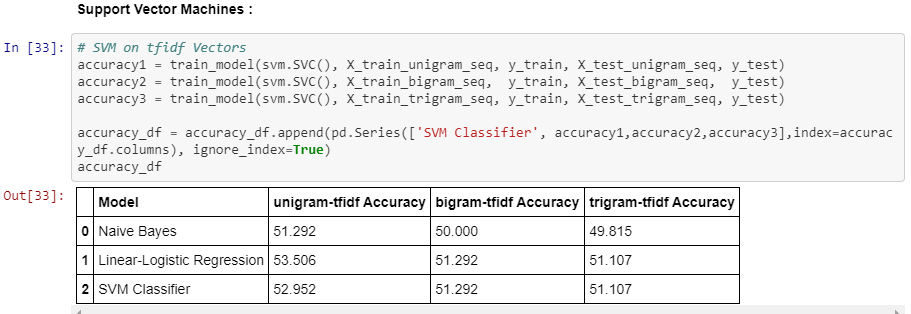
Logistic Regression is widely used technique because it is very efficient, does not require too many computational resources, it’s highly interpretable, it doesn’t require input features to be scaled, it doesn’t require any tuning, it’s easy to regularize, and it outputs well-calibrated predicted probabilities. Another advantage of Logistic Regression is that it is incredibly easy to implement and very efficient to train.



**SVM Classifier**

SVM separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points. The benefit is that it can capture much more complex relationships between data points without having to perform difficult transformations on your own.

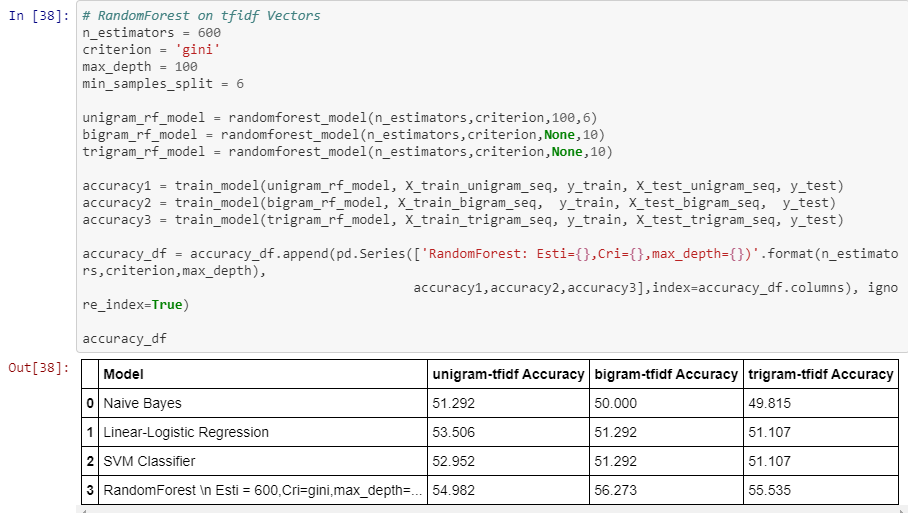
SVM's can model non-linear decision boundaries, and there are many kernels to choose from. They are also fairly robust against overfitting, especially in high-dimensional space.



**Random Forest**

Random forest is considered as a highly accurate and robust method because of the number of decision trees participating in the process. It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.

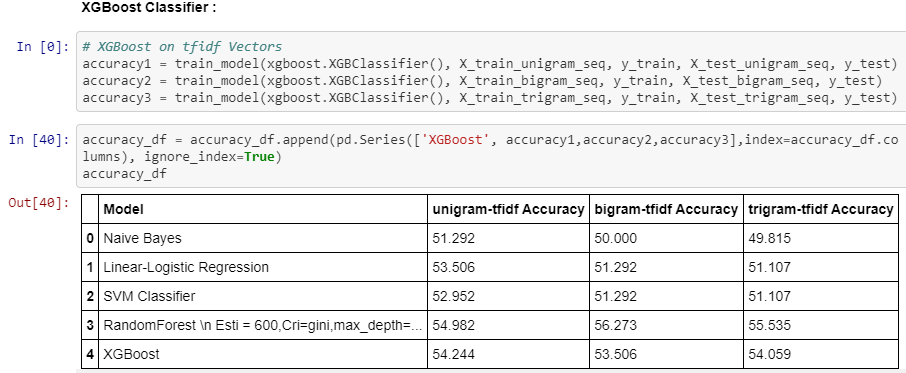
Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.



**XGBoost**

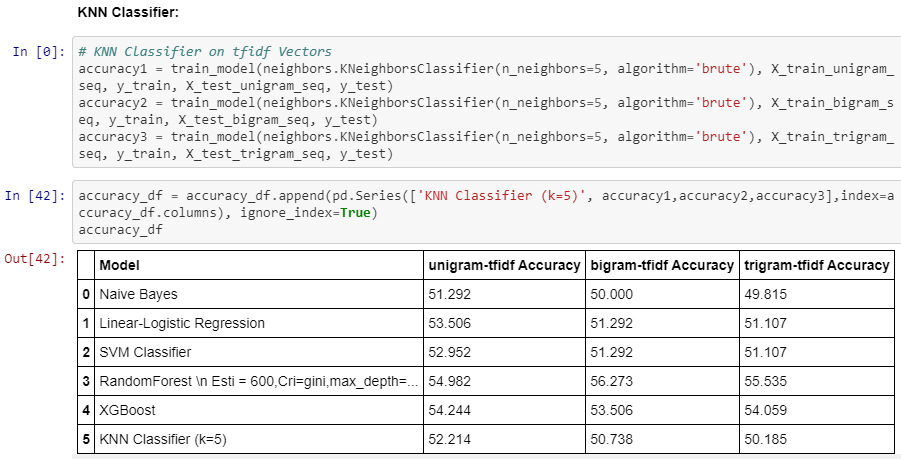
The XGBoost library implements the [gradient boosting decision tree algorithm](https://en.wikipedia.org/wiki/Gradient_boosting). XGBoost goes by lots of different names such as gradient boosting, multiple additive regression trees, stochastic gradient boosting or gradient boosting machines.

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. Generally, XGBoost is fast when compared to other implementations of gradient boosting. XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems.



**KNN**

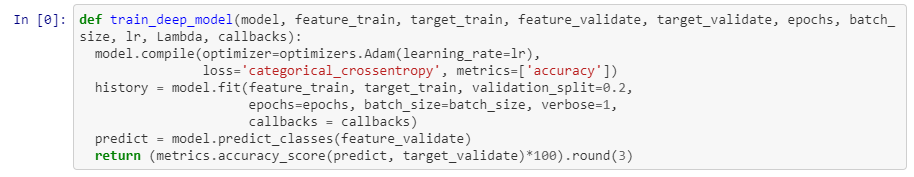
KNN uses lazy training which means all computation is deferred till prediction. This works very well if we have good training data. Naïve Bayes is a quick classifier and K-NN should be preferred when the data-set is relatively small. For this particular data set the KNN was not performing well.

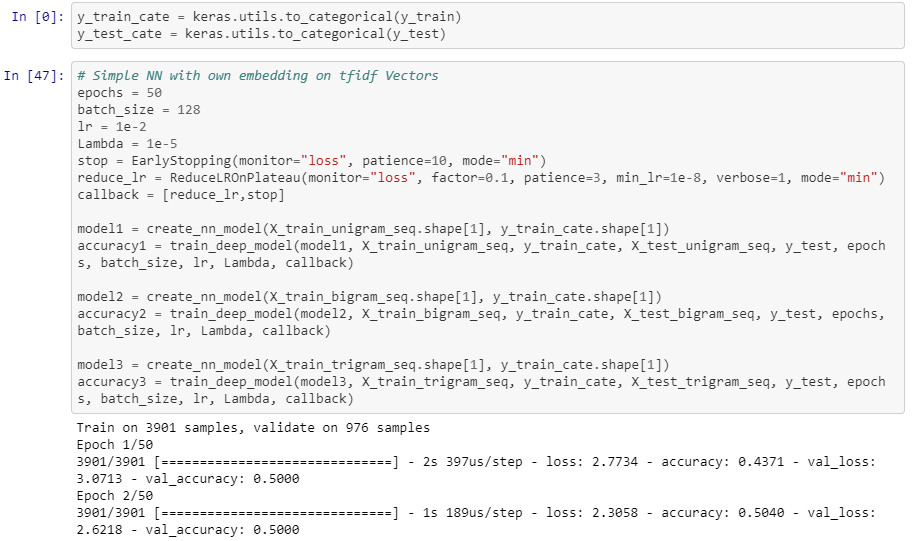


Naïve Bayes performed the worst Random Forest performed the best amongst all classifier.

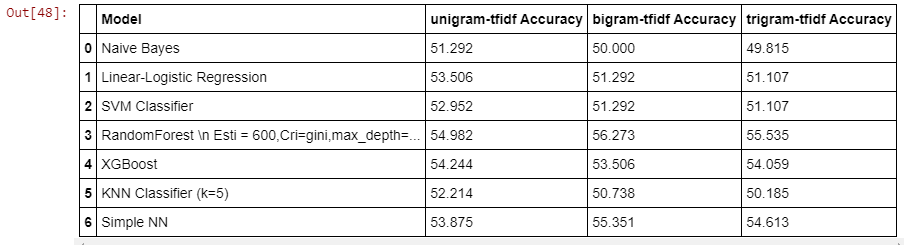
After Machine learning models, we tried deep learning models. Initially we tried simple neural network.







The final accuracy score of the models are below:

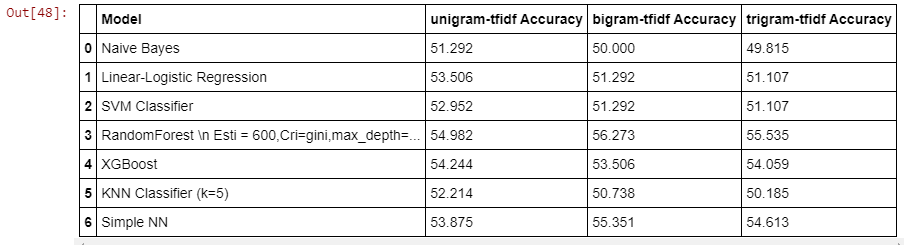


Random Forest performed the best amongst the basic Deep Learning and Machine Learning models.

4. How to improve your model performance? What are the approaches you can take to improve your model? Can you do some feature selection, data manipulation and model improvements? Provide your code and as much as visualizations you can share to describe what you have done so far.

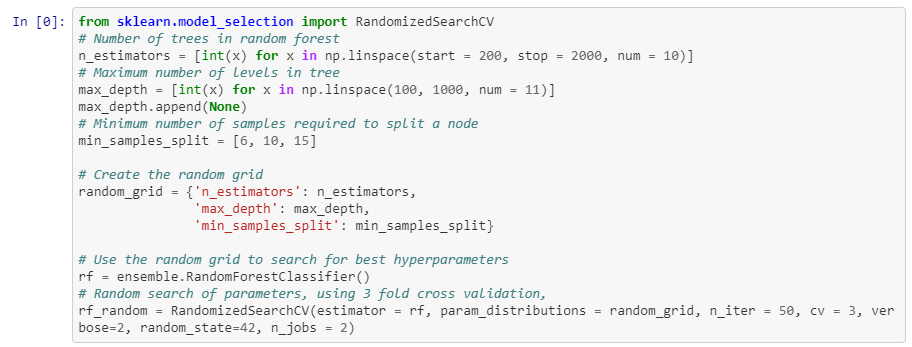
We dropped columns “caller” as it was not helpful for modeling. As mentioned in above section, we removed all the stop words and cleansed the data and combined the rows where there were few data sets and then did modeling on that. However, the results were not great. The accuracy score was ~54%.

For better performance of the model we tested for uni-gram, bi-gram and tri-gram vectors.



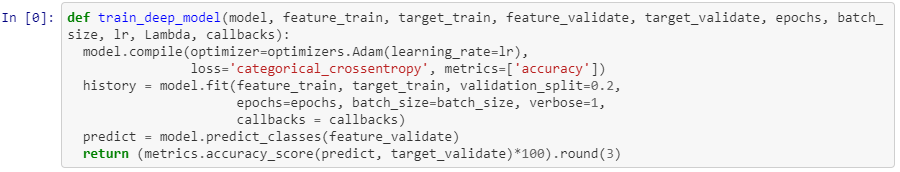
Random Forest and Simple Neural Network has better accuracy score in bi-gram and tri-grams. However, the others performed well in uni-grams.

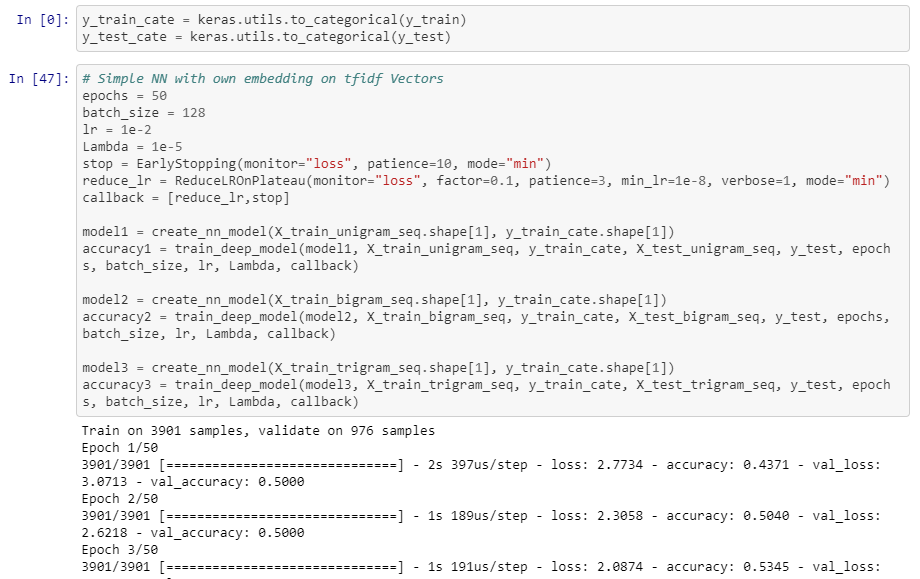
We performed hyper-parameter tuning for Random Forest to find best parameters.



Initially the neural network score was 50.526, 52.105 and 53.333 for uni-gram, bi-gram and tri-grams. To improve the model score we added more Dense Layer, performed batch normalization, added Dropout Layer, used callback and optimizer regularization loss.







We ran the model for 50 epochs. Below are the accuracy score after performing the above mentioned steps.

