Foundations of Intelligent Systems

Project 2

Exploring Discourse in Reddit

Prepared By:

Tejas Arya (ta2763)

Karshit Shah (ks6675)

Introduction

As any aspiring data scientist would start their venture into the professional world, we discuss our experience, learning, process and the progress of working on text data classification. The text data undertaken for this project work is data taken from three subreddits from www.reddit.com.

The subreddits we chose for this project are:

1) r/cricket
2) r/nba
3) r/science

The problem requires us to acquire posts from different subreddits in JSON format. The data is balanced as 600 posts are acquired from each thread.

We process the data to convert it into lowercase and remove the punctuations. Furthermore, we convert text and titles of the posts into feature vectors and attach labels to them. After preprocessing the data is stored in pandas data from and converted into feature vectors using scikit-learn's TfIdfVectorizer() methods.

After the vectorization of data, we split the posts from each subreddit evenly in the ratio of 50:25:25 into training, validation, and testing datasets such that there are 300 posts from each subreddit available for training our models, 150 posts from each subreddit available for validation and 150 posts from each subreddit available for testing our models' accuracy.

We then initialize our classifiers which we want to train our data on, train our model by passing our training data and their target values and then predict the values for test data. The models we used in this project are:

1) Support Vector Machine
2) Random Forest
3) Long Short-Term Memory (LSTM)
4) Feed Forward Neural Network

We, then compare our predictions with target values for test data to evaluate performance of our models and generate different plots such as learning curve, confusion matrix and ROC curve. We evaluate following measures to evaluate our learning approaches:

1) Precision
2) Recall
3) F1-Score

We also generated Word Cloud for text in posts including title and the main text. Furthermore, to visualize how close different posts are related to each other in meaning and text frequency, we use gensim library for topic modeling.

Methods

- Data Extraction: We used Praw library to extract data from reddit. Particularly, we use Praw.reddit.subreddit('subreddit name').hot(n) method to extract 600 hottest (a category to filter posts in reddit) posts in each subreddit.

- Creating Data Frames: Pandas library was used to create dataframes of posts so that the data could be vectorized. We used pandas.DataFrame({'text': posts.text, 'titles': posts.titles}) to create the data frames.

- Creating Feature Vectors: We imported sklearn.feature_extraction.text.TfidfVectorizer library and used its TfidfVectorizer().fit_transform(dataframe) method to learn term frequency and returns term-document matrix in vectorized form.

- Classifiers:
    - SVM : We used scikit-learn's SVC().fit() to train our SVM model by passing the training data, SVC().predict() to make class prediction on passed test data.

    - Random Forest: We imported RandomForestClassifier from sklearn.ensemble and used RandomForestClassifier().fit() to train our model and RandomForestClassifier.predict() to predict the classes.

    - LSTM: For LSTM we have used keras library's keras.models.Sequential() to create a Recurrent Neural Network Classifier.

- Performance Evaluation:
    - Learning Curve: Learning curve gives a plot that shows experience on the x-axis and learning or accuracy on the y-axis. It shows the rate at which the model is learning.
    - Confusion Matrix: We used sklearn.metrics.confusion_matrix() to generate confusion matrix and seaborn.heatmap() to visualize its values and distribution of predicted values versus actual values. This gives the number of false positives and false negatives while comparing the true results with the predicted results.
    - Precision: We used sklearn.metrics.classification_report() to generate precision values. This gives information about how accurate our model based on the predicted positive values.
    - Recall: We used sklearn.metrics.classification_report() to generate recall values
    - F1-Score: We used sklearn.metrics.classification_report() to generate F1-Score values. This is the weighted average of precision and recall scores.
    - ROC Curve: We used sklearn.metrics.roc_curve() to create an ROC curve which gives the plot of True Positive Rate and False Positive Rate.

- WordCloud: wordcloud.WordCloud().generate() method was used to create a word cloud from the text stored in the dataframe of our posts.

- LDA Topic Modeling: For extracting texts about frequent and hot topics, and generating visualizations for the magnitude of offering a words has towards topics, we tokenize our data using nltk.word_tokenize() and remove stop words using

nltk.corpus.stopwords.words('english') method. Then we create a dictionary and corpus using gensim.corpora library and then create a LDA model using gensim.models.LdaModel(corpus, dictionary) method.

Results

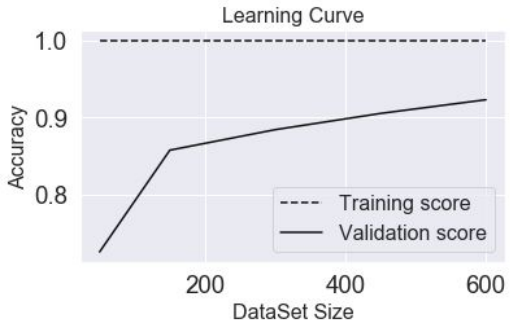
Binary Classification

We used two of the three subreddits for binary classification of subreddits. The two subreddits are

1. r/nba
2. r/cricket

We used different models on this subreddits to implement binary classification.

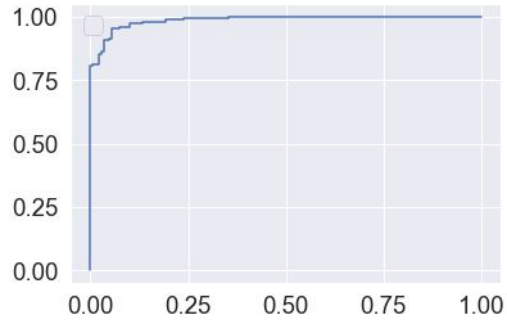
- **SVM**



Learning Curve

Confusion Matrix

	precision	recall	f1-score	support
0	0.95	0.89	0.92	150
1	0.89	0.95	0.92	150
micro avg	0.92	0.92	0.92	300
macro avg	0.92	0.92	0.92	300
weighted avg	0.92	0.92	0.92	300



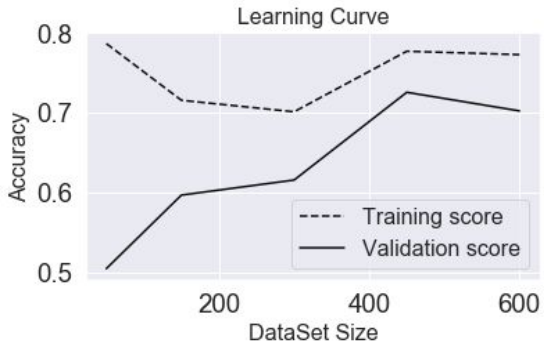
ROC Curve

For binary classification, there is a steep increase in accuracy till data size of approximately 100. There is a steady increase in accuracy for data set of size 100 to 600. We are able to achieve pretty good accuracy for dataset of size 600. Then at some point the accuracy remains steady which suggests that the accuracy won't increase by much even if we use more data.

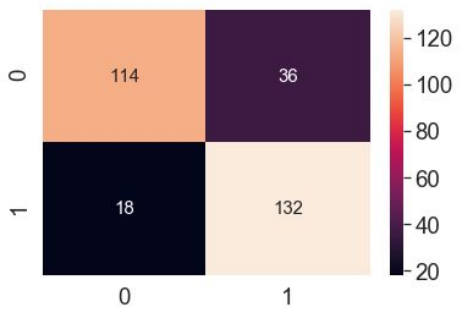
Also, precision is good for both class labels which indicates which indicates low false positive rate.

Recall rate is also above 0.89 which indicates that 89% of the test data was correctly classified.

- **Random Forest**

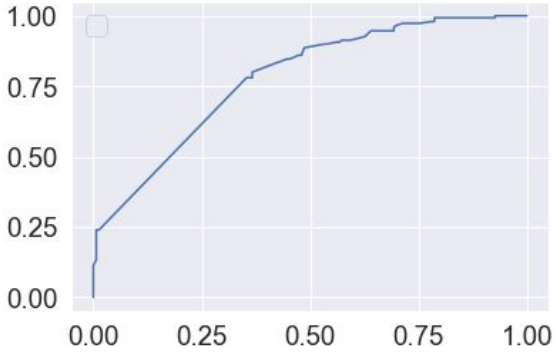


Learning Curve



Confusion Matrix

	precision	recall	f1-score	support
0	0.86	0.76	0.81	150
1	0.79	0.88	0.83	150
micro avg	0.82	0.82	0.82	300
macro avg	0.82	0.82	0.82	300
weighted avg	0.82	0.82	0.82	300

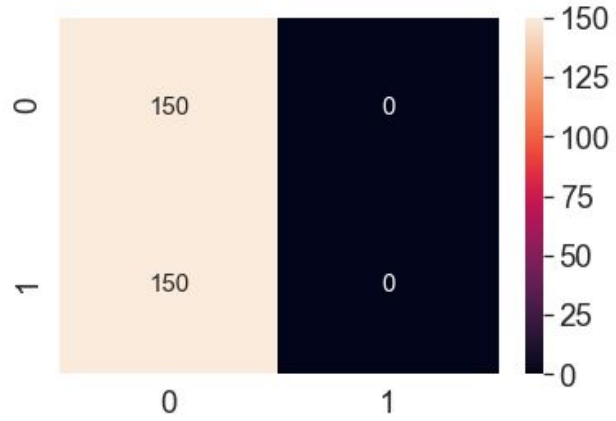


ROC Curve

Random Forest for binary classification performs slightly less accurately than SVM for binary classification.

Values of performance measures for Random Forest for the same data is slightly less than values achieved by fitting training data on SVM Classifier.

- **LSTM**



Confusion Matrix

We could not achieve desired results through implementation of LSTM classifier. We can see through confusion matrix generated that our model is predicting just one class i.e. r/nba for all posts. Our implementation of LSTM clearly did not work as implemented. We got to know how LSTM networks work, their architecture and how to use keras library to create a LSTM network and train and predict values for the data fitted on LSTM network.

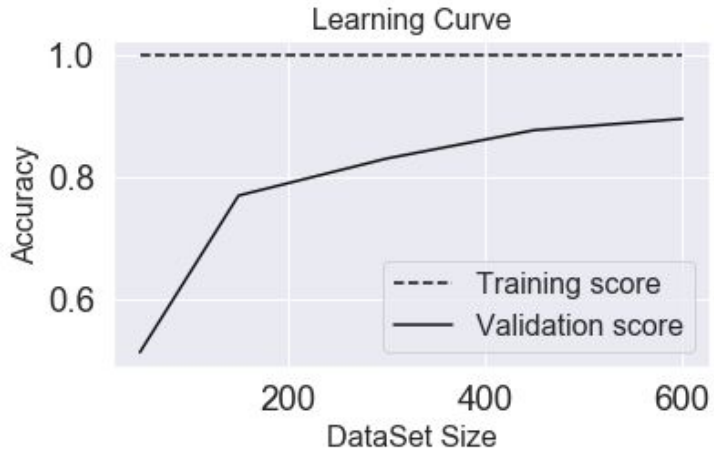
In order to be able to successfully implement a LSTM network, we need to work on text datasets that solely involves working on LSTM network without using the same dataset for various tasks so that the full attention could be given to understand the ropes of this task. We plan to work on this model in the future to ensure that it gives proper predictions

Multiclass Classification

We used three subreddits for binary classification of subreddits. The two subreddits are

1. r/nba
2. r/cricket
3. r/science

- **SVM**

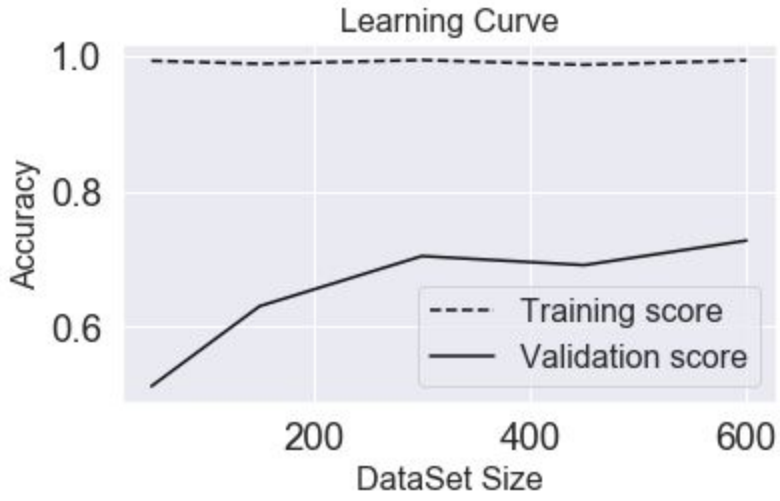


	precision	recall	f1-score	support
0	0.96	0.87	0.91	150
1	0.89	0.94	0.91	150
2	0.94	0.98	0.96	150
micro avg	0.93	0.93	0.93	450
macro avg	0.93	0.93	0.93	450
weighted avg	0.93	0.93	0.93	450

SVM for multiclass classification performs well in terms of performance measures achieving average precision of 0.93, average recall of 0.93, and average f1-score of 0.926.

It performs better in terms of performance measures in comparison to SVM for binary classification as well.

- **Random Forest**



	precision	recall	f1-score	support
0	0.79	0.82	0.81	150
1	0.68	0.74	0.71	150
2	0.86	0.75	0.80	150
micro avg	0.77	0.77	0.77	450
macro avg	0.78	0.77	0.77	450
weighted avg	0.78	0.77	0.77	450

Random Forest for multiclass classification has the worst performance in terms of precision, recall and f1-scores.

- **Feed Forward Neural Network**

	precision	recall	f1-score	support
0	0.98	0.89	0.93	150
1	0.86	0.99	0.92	150
2	0.99	0.93	0.96	150
micro avg	0.93	0.93	0.93	450
macro avg	0.94	0.93	0.93	450
weighted avg	0.94	0.93	0.93	450

Feed-Forward Artificial Neural Networks performs the best as compared to all the models we implemented achieving precision of value 0.98, recall of value 0.99, and f1-score of value 0.96.

Conclusion

In this project, we accomplished extracting data from an online resource in a desired format, processing it to convert it to a format that enables us to better handle the data without losing any important information, vectorize data, split data in training, validation and test sets.

We used libraries available in python to fit our data to various machine learning algorithms viz. SVM, Random Forest, Feed-Forward Artificial Neural Network.

We were able to achieve maximum precision of 0.98, maximum recall of 0.99 and maximum f1-score of 0.96.

We also created word cloud of text in the top 600 reddit posts from r/cricket, r/nba and r/science. Then we created an interactive LDA Model using gensim and visualized it.

References

1. https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
2. https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a
3. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
4. https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/
5. https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
6. https://amueller.github.io/word_cloud/generated/wordcloud.WordCloud.html
7. https://en.wikipedia.org/wiki/Long_short-term_memory
8. https://skymind.ai/wiki/lstm
9. https://scikit-learn.org/0.15/modules/generated/sklearn.cross_validation.train_test_split.html#sklearn.cross_validation.train_test_split