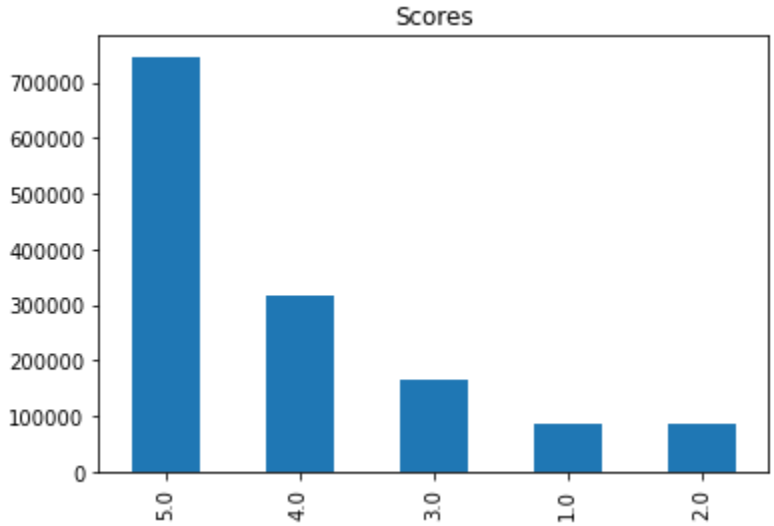
CS-506 Mid Term Report

Name: Krishna Adithya Venkatesh
BUID: U31711133

Amazon movie review Sentiment Analysis

1. Data Exploration:

1. Plot the Score Values and we understand that the number 5 star rating is pretty high compared to other scores.



2. Plotting the word cloud of 'Text' and]summary', shows us that the summary is very different from the Text and there is no correlation between the two.

3. Also going through some of the summary values, I felt like they had no correlation to the Score as well and hence decided not to use it.
4. Analysing 'Text', 'Text seems to have the highest correlation to the 'Score' Hence I chose to use just 'Text' to predict the 'Score'.

2. Feature Extraction:

1. Based on the dataset analysis 'Text' is the only column used to predict 'Score', 'Text' is a string(review), preprocessing is to remove the noise in the string.

2. The preprocessing steps:

a. Removal of punctuation marks
b. Converting all the strings to lower case, because the lower case and upper case words would be interpreted differently.
c. Removed stop words, experiments included two stop lists as follows.
    i. **Stop 1:** ['no', 'nor', 'not', 'ain', 'aren', "aren't", 'couldn', 'what', 'which', 'who','whom','why', 'how', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't",'hasn',"hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn'"mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',"wasn't",'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't", 'don', "don't"]
    ii. **Stop 2:** ['a','an','the','and','this','that']

3. TF-IDF Vectoriser from sklearn library was used to convert the string to a matrix of TF-IDF features.

3. Dataset:

1. Training Data had x rows and Test data had 300000 rows, test data does not have scores, so for **local validation,** Training Data was split into train and test datasets.
2. Performance on the local test set will give us a sense of how the model will perform on the unknown data
3. After preprocessing I saved the preprocessed results to a CSV file so that I don't have to repeat the process if the kernel resets or if the Kaggle notebook times out.
4. Since Random forest is very memory intensive, I need to subsample the dataset for a few experiments, I created a subsample ensuring there is an equal number of samples in all classes to **avoid data imbalance.**

4. Models, Hyperparameter and Training:

1. Models that were mainly used in the experiment were Support Vector Machine and Random Forest.
    a. Why SVM?: SVM is very effective in high dimensional space, it also uses a subset of training points in the decision function, so it is memory efficient. Since we have a very large training dataset, SVM seemed the most memory-efficient option.
    b. Why Random Forest ?: is a very good classification model and wanted to experiment with that.
2. To optimize the model's hyperparameters I experimented a bit with GridSearch, but since the dataset is too large, a portion of it was used but I ended up using hyperparameters based on intuition.

3. While training the model I used Sklearn's **Pipeline** function. Since the dataset is large fitting TF-IDF is very time-consuming before training a model every single time, using Pipeline makes to process much faster and streamlines it.
4. Saved model as a pickle file to use whenever needed.

5. Experiments:

My best model was an ensemble of predictions of 6 models: 3 SVM and 3 Random forest model.

1. SVM:
    a. Experiment 1: Removed the stop words from the stop 2 list (2.2.c) and processed the string using TF-IDF. Training on the split and achieved a decent accuracy so trained it on the entire training dataset.
    b. Experiment 2: Removed the stop words from the stop 1 list (2.2.c) and processed the string using TF-IDF.
    c. Experiment 2: Removed the stop words from the stop 2 list and processed the string using first using Vector counter and then transformed it using TF-IDF.

All three experiments gave good results and yielded RMSE close to 0.98.

2. Random Forest: Random forest was very memory intensive so had to use a sample dataset (3.4)
    a. Experiment 1: Removed the stop words from the stop 2 list (2.2.c) and processed the string using TF-IDF. Trained the model on the split training data (40,000 rows).
    b. Experiment 2: Removed the stop words from the stop 2 list (2.2.c) and processed the string using TF-IDF. Trained the model on the split training data (50,000 rows).
    c. Experiment 2: Removed the stop words from the stop 2 list (2.2.c) and processed the string using TF-IDF. Trained the model on the entire dataset by tuning the hyperparameter (Max Depth =2)

All the experiments gave very poor results and yielded RMSE close to 1.4.

Ensemble: At first I did not want to use Randomforest results and just created an ensemble (Mean prediction) of the SVM results and it improved the accuracy, But while experimenting a bit with creating an ensemble of all 6 experiments with weighted voting method my accuracy improved. Giving the SVM models higher weightage and Random Forest models low weightage based on their performance in the test set.

0.25*svm_1+0.2*svm_2+0.25*svm_3+0.1*rf_1+0.1*rf_2+0.1*rf_3