PROJECT SUMMARY

PROBLEM STATEMENT

The goal is to build a predictive model to predict sentiment ratings for hotel reviews. There are 5 sentiment ratings for the reviews.

EXPLORATORY DATA ANALYSIS

1. Train data contained one bad entry in the rating, that row was dropped.
2. In the train dataset, rating values are of type string i.e. ['1','2','3','4','5'] whereas in dev data set , sentiment rating is of type integer. So the datatype of rating in dev dataset is changed to be the same as that of train sentiment labels.
3. Analysed the distribution of reviews in the training dataset for all label categories. distribution is approximately balanced which is good for data modelling for classification.

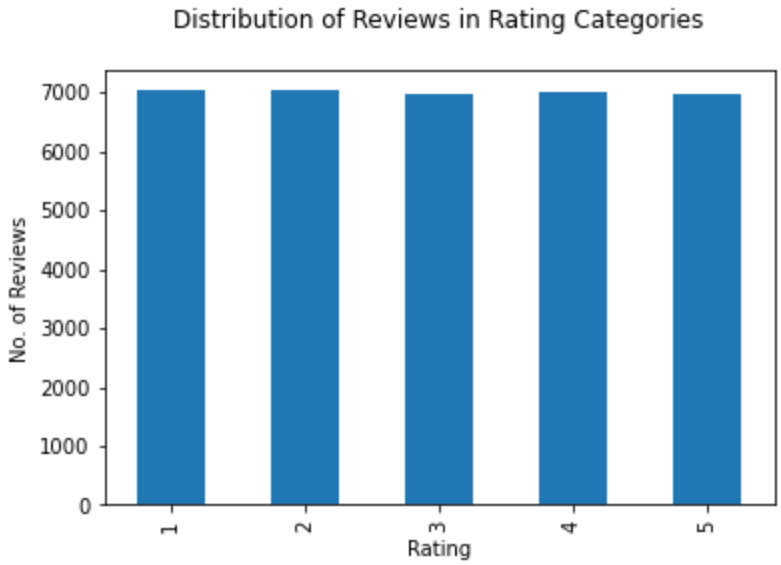

Figure 1

4. Analysed review length to check if data contains any anomalies in terms of review length. Analysis showed that there are some very long reviews associated with rating 1 and 2 in comparison to other categories.

MODELS AND FEATURE GENERATION

Baseline Model

I preferred to start with simple linear modes and I had two popular choices in mind. First Logistic Regression and second Naive Bayes. Logistic regression is a discriminative classifier while naive Bayes is a generative classifier. Naive Bayes also assumes that the features are conditionally independent. Real data sets are never perfectly independent but they can be close. After doing some literature survey and reading the chapter 5 on Speech and Language Processing by Daniel Jurafsky & James H. Martin [1], I was assured that Logistic regression is much more robust to correlated features and therefore, I decided to choose the Logistic Regression (Multinomial) model as the baseline model.

- **Text Feature Generation-**Text reviews are required to be converted into numeric features in order to be used as input to a model for training. In order to generate the feature vectors, the following steps were performed.

    1. **Data cleaning:**

       Undesired symbols such as spaces, punctuation, numbers, special characters and HTML tags were removed from the reviews.

    2. **Text preprocessing :**

       Remove Stop Words - Words such as "the" "a" and "it" are considered as English stop words in NLTK corpus [2]. Such words add dimension to the word count in review and do not effectively contribute towards the sentiment of a text.

       Stemming - Stemming is basically removing the suffix from a word and reducing it to its root word. Therefore words such as "fly" and "flying" would both be represented as "fly". Thus, allowing in the reduction of dimension for word features and also helps algorithms to more accurately find trends in the meaning of sentences (e.g "the bird is flying" and "the birds flies" will more accurately be associated with the same meaning.

       Count vectorizer - Convert a collection of text documents to a matrix of token counts based on ngram range.

       TF-IDF (Term Frequency Inverse Document Frequency) vectors - It transforms a count matrix to a normalized tf-idf representation. It scales down the impact of tokens that occur very frequently in a given corpus that are hence, empirically less informative than features that occur in a small fraction of the training corpus [3].

       TF-IDF feature vectors are used as input features for our baseline model. The Logistic Regression model was investigated with three different sets of parameters.

State of Art Model

The text sequences can vary in length having a very large vocabulary of input symbols and may require the model to learn the long-term context or dependencies between symbols in the input sequence. In such a scenario, the **Long Short Term Memory (LSTM)** model is more suitable. Therefore it is selected as a State of Art model for the text classification tasks.

- ### Text Feature Generation

    1. **Data Cleaning:**

        Undesired symbols such as spaces, punctuation, numbers, special characters and HTML tags were removed from the reviews.

    2. **Data Representation:**

        Tokenizer - Keras has a tokenizer that tokenizes (splits) a text into tokens and then to tokenized sequences while keeping only the words that occur the most in the text corpus.

        Pad Sequence - Our model requires that each text sequence is of the same length (the same number of words/tokens). pad_sequence function in Keras transforms the tokenizes sequences into fixed-length sequences of length controlled by max_len. Shorter sequences are padded to be equal to max_len.

- ### LSTM Model Architecture

    Word embedding - Word embedding is a technique where words are encoded as real-valued vectors in a high dimensional space. Keras provides an easy way to convert positive integer representations of words into a word embedding by an Embedding layer.

    Architecture of the LSTM model used

    1. The first layer is the Embedded layer that uses fixed-length vectors to represent each word.
    2. The next layer is the LSTM layer with fixed memory units
    3. Finally, a dense output layer with 5 neurons and a softmax activation function for classification in 5 sentiment categories.

Two variations of parameters in the model were used for training. The Model with each set of the network parameters was trained for 5 epochs and weights for the best performing model were only saved during training respectively.

RESULTS

The Logistic Regression model was investigated with different variations of ngram range and regularization strength. Results are as below.

Index	Logistic Regression Model (with different parameters)	Accuracy score	
		Train dataset	Dev dataset
1	ngram range(1,1) and C = 5	.92	.76
2	**ngram range(1,1) and C = 1 (selected)**	**.85**	**.76**
3	ngram range(1,2) and C = 1	.79	.63

* Where C=inverse of regularization strength

Observations-

- ★ Result 2 of logistic regression model n grams (1,1),C=1 gives better and generalised results in terms of accuracy and classification results. Therefore, it is selected for generating prediction on test data.
- ★ Model with ngram range (1,1), C=5 captures more variance from the train data but it seems to be overfitting as there is no improvement in dev accuracy score. Result 3 is not better either.

Classification report of dev dataset for selected Logistic Regression model parameters(result 2)

Dev Classification Report				
Class prediction/ parameters	**precision**	**recall**	**f1-score**	**support**
1	0.79	0.84	0.81	1523
2	0.71	0.70	0.71	1507
3	0.73	0.75	0.74	1483
4	0.75	0.74	0.74	1500
5	0.84	0.79	0.81	1486
Accuracy			0.76	7499
Macro Avg	0.76	0.76	0.76	7499
Weighted Avg	0.76	0.76	0.76	7499

LSTM Model was also investigated for variation in embedding dimension and LSTM memory units. Results are as

Index	LSTM Model (with different parameters)	Accuracy score	
		Train dataset	Dev dataset
1	Embedding dim=64, LSTM units=200	.92	.74
2	**Embedding dim=32, LSTM units=100 (selected)**	**.89**	**.74**

Accuracy graphs

LSTM (Embedding dim=64, LSTM units=200)

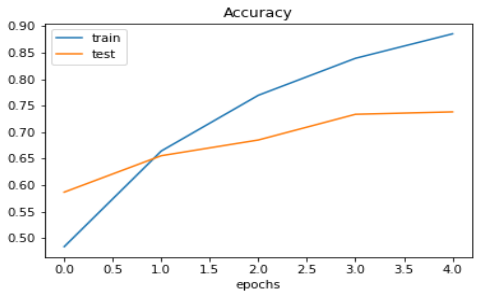

Figure 2

LSTM (Embedding dim=32, LSTM units=100)

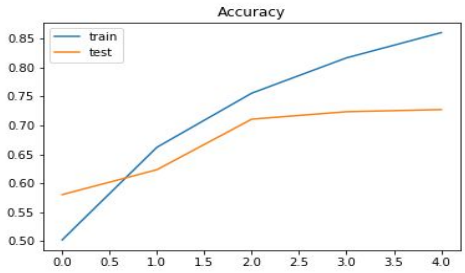

Figure 3

Observations-

- ★ Model result with parameters Embedding dim=32, LSTM units=100 is less overfitting and therefore, it is selected for generating prediction on test data

Dev Classification Report				
Class prediction/ parameters	**precision**	**recall**	**f1-score**	**support**
1	0.78	0.80	0.79	1523
2	0.66	0.67	0.66	1507
3	0.67	0.77	0.72	1483
4	0.74	0.68	0.71	1500
5	0.86	0.76	0.81	1486
Accuracy			0.74	7499
Macro Avg	0.74	0.74	0.74	7499
Weighted Avg	0.74	0.74	0.74	7499

COMPARISON OF SELECTED BASELINE AND STATE OF ART MODEL

Index	Model	Accuracy score	
		Train dataset	Dev dataset
1	Logistic regression (ngram range(1,1),C=1	.85	.76
2	LSTM model (embedding dim=32, LSTM units=100)	.89	.74

Observation

- ★ It is observed from the above table that both models performed approximately in a similar fashion. Logistic regression scores a bit higher in the dev dataset.
- ★ However, LSTM model has a scope of improvement by training and is more likely to perform better with training on more iterations, variation in architecture and large train dataset.

References:

1. https://web.stanford.edu/~jurafsky/slp3/5.pdf
2. http://www.nltk.org/howto/corpus.html
3. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html
4. https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/
5. https://lionbridge.ai/articles/using-deep-learning-for-end-to-end-multiclass-text-classification/