Sentiment Analysis On Tripadvisor Reviews (London Hotels)

IOD: CAPSTONE

Wong Kar Mun

CONTENTS

Background

Tripadvisor is a huge platform for travelers looking for hotel reviews and to connect with hotels.

884 million reviews on Tripadvisor as of 2020

83% customers say reviews help them pick the right hotel.

80% read at least 6 to 12 reviews prior to booking

53% won't commit to a booking until they read reviews

PhoCusWright

Problem Statement

Why are hotel reviews important?

- Booker: The credibility of a hotel
- Hotel: Highlights the things that can be improved for more booking

Problem Statement

Problem Statement

How do we efficiently and quickly scan through the reviews to get a positive or negative sentiments from huge number of hotel reviews

Problem Statement

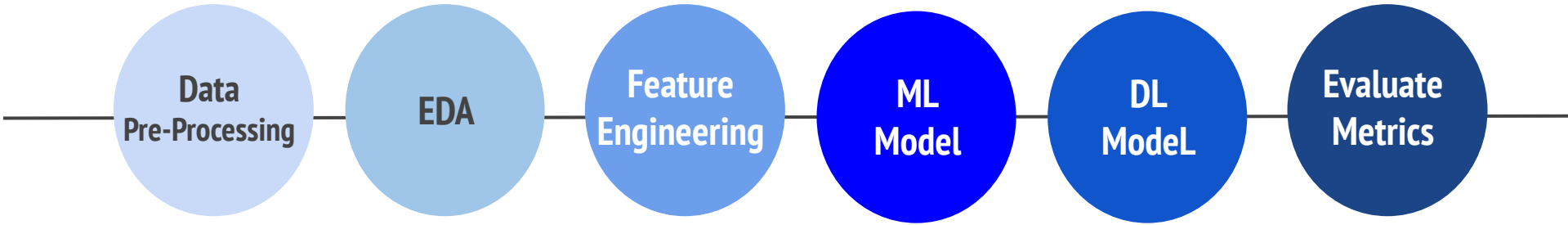
Objective

Implement machine learning and deep learning models to predict the rating of hotel reviews.

Dataset:

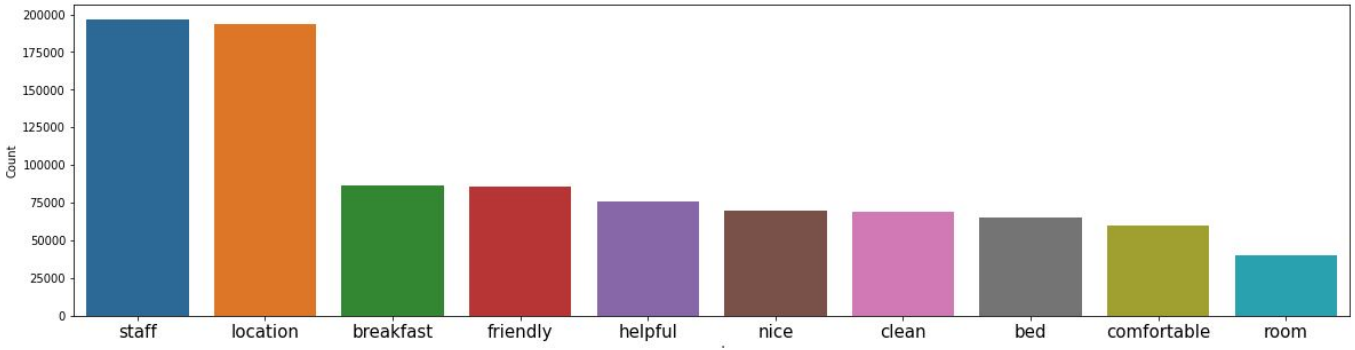
Kaggle: 515K Hotel Reviews Data in London

Framework

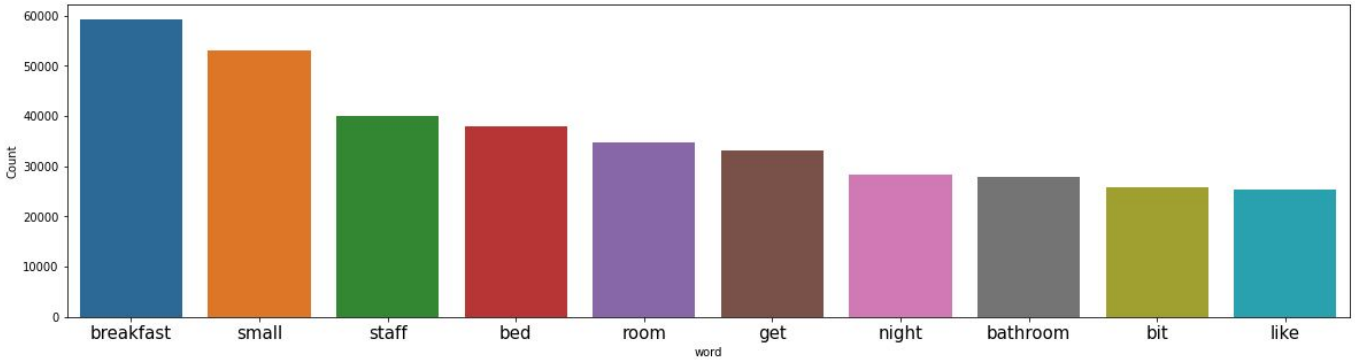

Data
Pre-Processing
EDA
Feature
Engineering
ML
Model
DL
ModeL
Evaluate
Metrics

EDA: TOP 10 Words

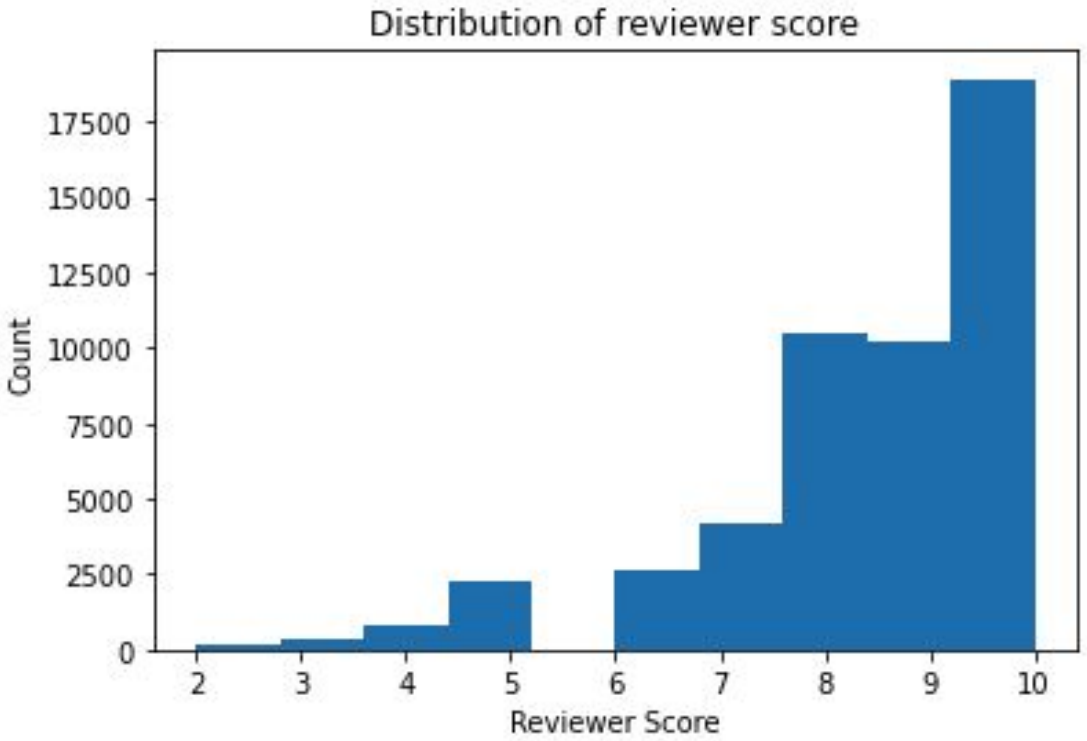
Positive



Negative



EDA: Distribution of Sentiments



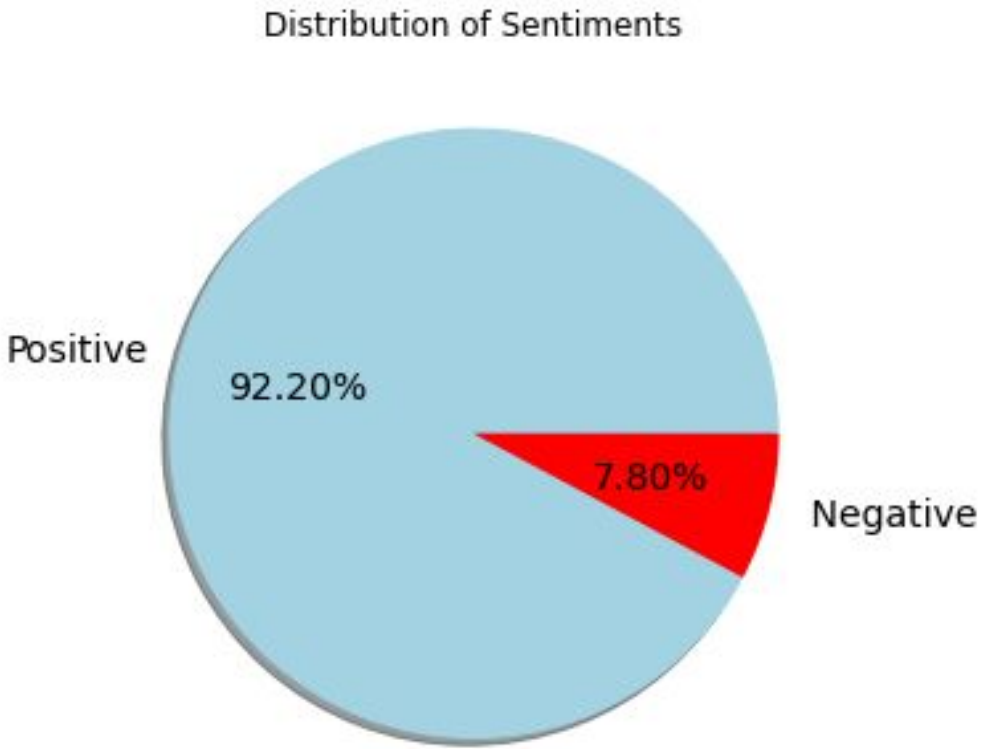
EDA: Distribution of Sentiments

Convert Ratings to Binary Class Target Variable of 0,1

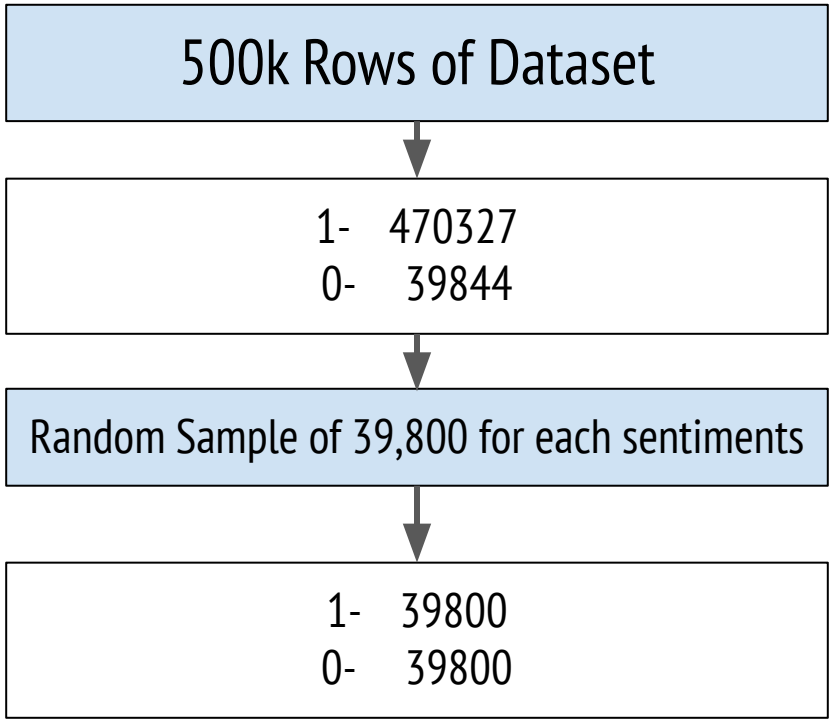
Negative Sentiments
Target Variable: 0
Score: 1 to 5

Positive Sentiments
Target Variable: 1
Score: 6 to 10

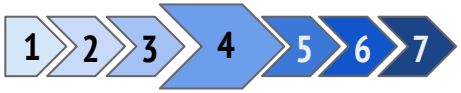
EDA: Distribution of Sentiments



Resampling Dataset



NLP Methods



1. Count Vectorizer
2. TF-IDF Vectorizer (Word)
3. TF-IDF Vectorizer (Bi & Tri-Gram)
4. TF-IDF Vectorizer (Char Level)

Classification Models

ACCURACY SCORE

Training 70%, Test 30%

	Count Vectors	WordLevel TF-IDF	N-Gram Vectors	CharLevel Vectors
Naïve Bayes	0.801675	0.803936	0.758375	0.780737
Logistic Regression	0.809129	0.818007	0.764154	0.812688
Support Vector Machine	0.799874	0.810846	0.758333	0.813861
Random Forest	0.798827	0.797236	0.734464	0.780193
Gradient Boosting	0.769514	0.770142	0.630193	0.781449
Stacking	0.811558	0.819514	0.767211	0.816248

Deep Learning (KERAS)

FRAMEWORK

1. Neurons: 32 vs 64 Nodes
2. Hidden Dense layer (1 vs 2)
3. Epoch 10, 20
4. DL Model - CNN
5. Hyperparameter Tuning
6. Regularize Model
7. Dropout

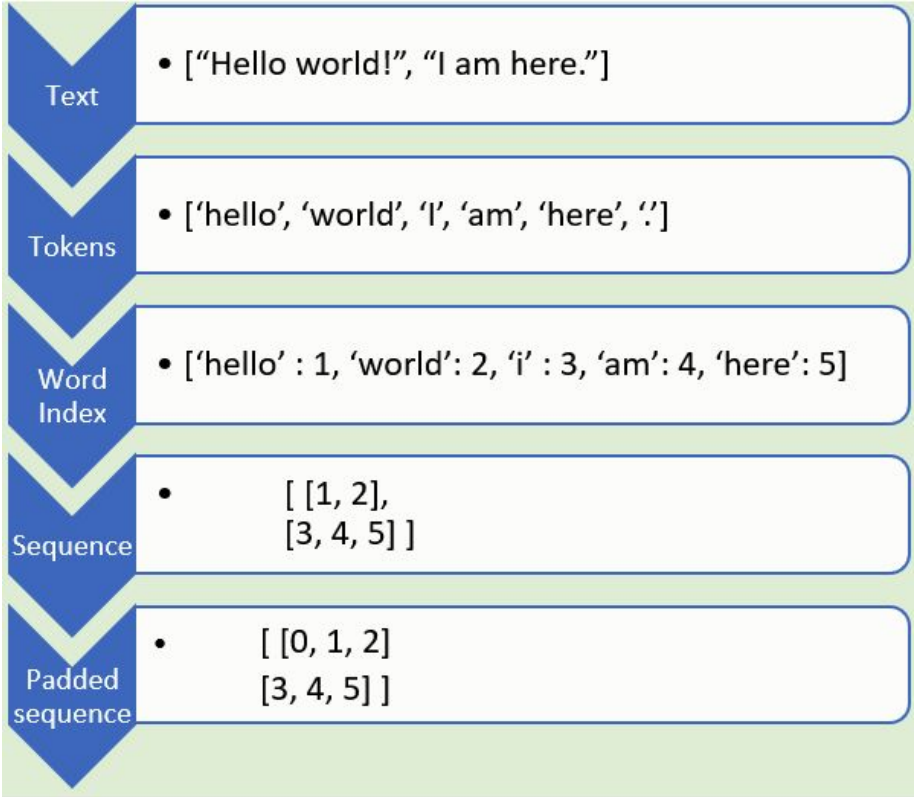
Deep Learning (KERAS)

TEXT PRE-PROCESSING

1. Tokenizer - Text to Sequence

↓

2. Pad Sequence



Deep Learning (KERAS)

32 nodes VS 64 nodes

Training 70%, Test 30%,
Epoch = 20, 2 Hidden Layers

	32 Nodes		64 Nodes	
	Train	Test	Train	Test
Accuracy	0.9860	0.7626	0.9821	0.7746
Precision	0.9850	0.7746	0.9745	0.7648
Recall	0.9871	0.7407	0.9900	0.7930
ROC AUC	0.9860	0.7626	0.9821	0.7746

Deep Learning (KERAS)

Hidden Layer: 1 VS 2

Training 70%, Test 30%,
Epoch = 20, Nodes = 64 each layer

	1 Hidden Layer		2 Hidden Layer	
	Train	Test	Train	Test
Accuracy	0.9844	0.7673	0.9841	0.7751
Precision	0.9801	0.7690	0.9762	0.7702
Recall	0.9888	0.7642	0.9924	0.7842
ROC AUC	0.9844	0.7673	0.9841	0.7751

Deep Learning (KERAS)

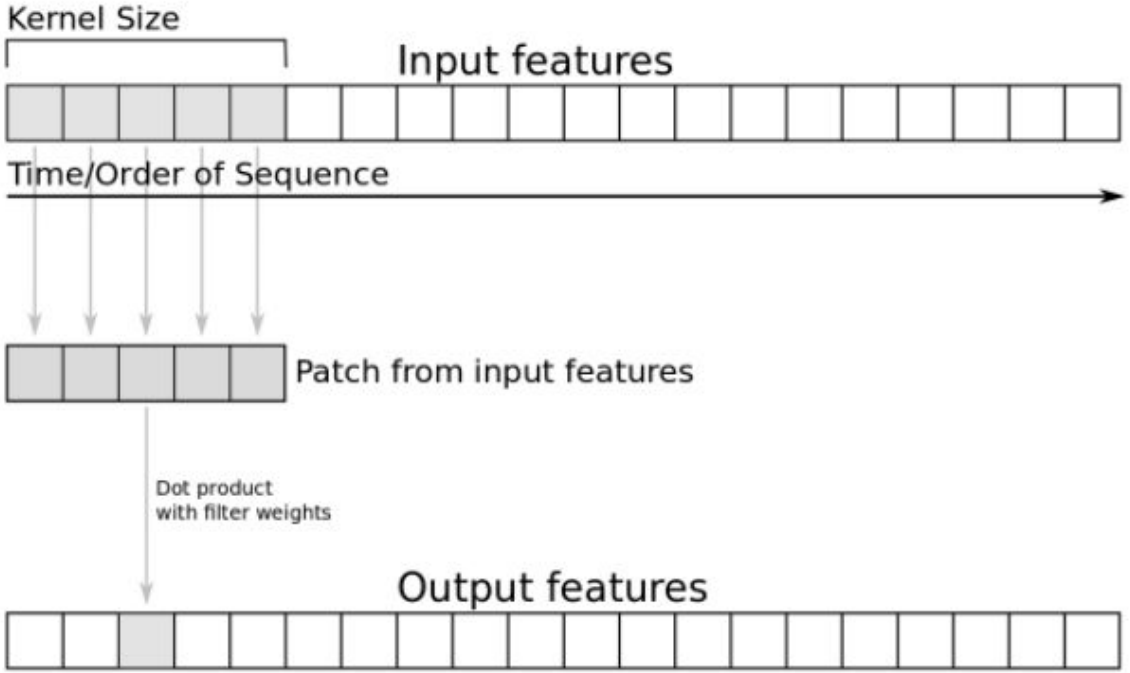
Epoch: 10 VS 20

Training 70%, Test 30%,
Nodes = 64, 2 Hidden Layer

	10		20	
	Train	Test	Train	Test
Accuracy	0.9748	0.7773	0.9830	0.7699
Precision	0.9677	0.7755	0.9816	0.7769
Recall	0.9825	0.7806	0.9845	0.7572
ROC AUC	0.9748	0.7773	0.9830	0.7699

Deep Learning (KERAS)

Convolutional Neural Networks (CNN)



Deep Learning (KERAS)

Convolutional Neural Networks (CNN)

Training 70%, Test 30%,
Epoch = 10, Nodes = 64, 2 Hidden Layer

MODEL OVERFITTING!

	Base		CNN	
	Train	Test	Train	Test
Accuracy	0.9738	0.7782	0.9747	0.7809
Precision	0.9693	0.7828	0.9710	0.7834
Recall	0.9786	0.7701	0.9787	0.7765
ROC AUC	0.9738	0.7782	0.9747	0.7809

```
Best Parameters: 'num_filters': 32,
                 'kernel_size': 5
```

Deep Learning (KERAS)

MODEL OVERFITTING!

Regularizing & Dropout

Training 70%, Test 30%,
Epoch = 10, Nodes = 32, 1 Hidden Layer

+0.0027

+0.0061

MODEL	CNN		+ Regularized (0.001)		+ Dropout (0.5)	
	Train	Test	Train	Test	Train	Test
Accuracy	0.9747	0.7809	0.9767	0.7836	0.9710	0.7897
Precision	0.9710	0.7834	0.9746	0.7954	0.9697	0.8168
Recall	0.9787	0.7765	0.9788	0.7637	0.9724	0.7470
ROC AUC	0.9747	0.7809	0.9767	0.7836	0.9710	0.7897

EVALUATE METRICS

CNN vs STACKING(TFIDF WORD):

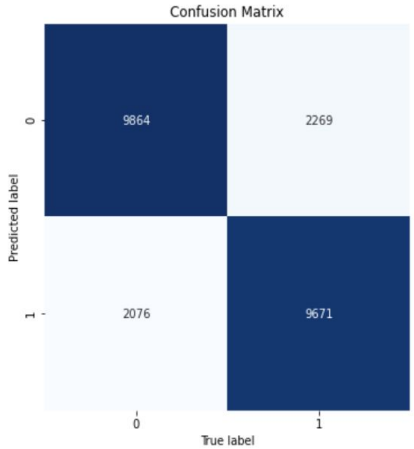
★ CHOSEN MODEL

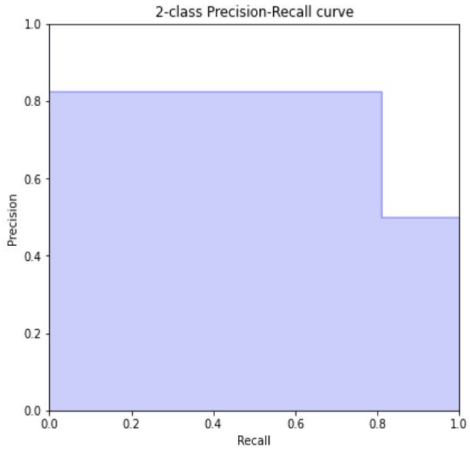
	CNN		STACKING	
METRICS	Train	Test	Train	Test
Accuracy	0.9710	0.7897	0.8968	0.8180
Precision	0.9697	0.8168	0.9017	0.8233
Recall	0.9724	0.7470	0.8906	0.8100
ROC AUC	0.9710	0.7897	0.8968	0.8180

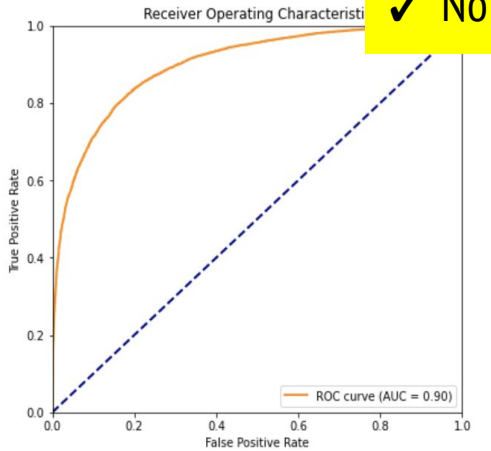
CONCLUSION

CHOSEN MODEL : STACKING (TFIDF WORD), Max_features = 5000

```
***************
* TF-IDF Word *
***************
Accuracy : 0.8180 [TP / N] Proportion of predicted labels that match the true labels. Best: 1, worst: 0
Precision: 0.8233 [TP / (TP + FP)] Not to label a negative sample as positive.        Best:
Recall   : 0.8100 [TP / (TP + FN)] Find all the positive samples.                     Best:
ROC AUC  : 0.8180                                                                     Best:
------------------------------------------------------------------------------------------
TP: True Positives, FP: False Positives, TN: True Negatives, FN: False Negatives, N: Number of sample
```







✓ Best classifier that can successfully predict the rating of reviews

✓ Highest Accuracy Score

✓ No Overfitting

CONCLUSION

Limitations:

- Balancing of dataset causes it to reduce significantly from 500k to 76.8k
- Dataset is only for hotels in London, would be good try the model on a bigger and balanced dataset across a bigger region like Europe.

Future Scope:

- Multi-class classification
- LSTM Model
- Text pre-processing with Word2Vec / GloVe

END

Thank You!

Questions?