

## **Capstone Project Report**

**By:** Sripriya Prabhala

**Mentor:** Dr. Rajiv Shah

# **Use of Machine Learning and Deep Learning Models in Classifying Sentiments Based on Hotel Reviews Dataset.**

## **Introduction**

In today's era, tonnes of information is available online, but in an unorganized form. A lot of investigation and research is going on organizing the data so that users can better utilize information for their needs. In recent times, focus has been shifted on to classifying the sentiments of reviews (whether positive or negative) from review websites (movie reviews, store reviews, hotel reviews, product/service reviews and so on). Classification based on sentiments will provide an overall summary to a user/consumer helping them make better choices. In this report, we present 4 machine learning models and a deep learning model we have used in order to classify the sentiments of hotel reviews for our client Trip Advisor.

## **Dataset**



The dataset (Hotel Reviews) used for this project has been obtained from the Datafiniti database on the website "data.world". The dataset is a sample of 1000 hotels in the United States and their reviews posted by consumers online. The dataset consists of the following categories:

1. Address
2. Categories (whether hotel or motels or resorts)
3. City
4. Country
5. Latitude
6. Longitude
7. Name (of the hotels)
8. Postal Code
9. Province
10. Reviews Date
11. Reviews Date Added
12. Reviews Do Recommend
13. Reviews ID
14. Reviews Rating (on a scale of 1 – 10)
15. Reviews Text
16. Reviews Title
17. Reviews User City
18. Reviews User Name
19. Review User Province

The following images show the various columns within the dataset.

|   | address                | categories | city     | country | latitude  | longitude | name               | postalCode | province |
|---|------------------------|------------|----------|---------|-----------|-----------|--------------------|------------|----------|
| 0 | Riviera San Nicol 11/a | Hotels     | Mableton | US      | 45.421611 | 12.376187 | Hotel Russo Palace | 30126      | GA       |
| 1 | Riviera San Nicol 11/a | Hotels     | Mableton | US      | 45.421611 | 12.376187 | Hotel Russo Palace | 30126      | GA       |
| 2 | Riviera San Nicol 11/a | Hotels     | Mableton | US      | 45.421611 | 12.376187 | Hotel Russo Palace | 30126      | GA       |

| reviews.date         | reviews.dateAdded    | reviews.doRecommend | reviews.id | reviews.rating | reviews.text                                      |
|----------------------|----------------------|---------------------|------------|----------------|---|
| 2013-09-22T00:00:00Z | 2016-10-24T00:00:25Z | NaN                 | NaN        | 4.0            | Pleasant 10 min walk along the sea front to th... |
| 2015-04-03T00:00:00Z | 2016-10-24T00:00:25Z | NaN                 | NaN        | 5.0            | Really lovely hotel. Stayed on the very top fl... |
| 2014-05-13T00:00:00Z | 2016-10-24T00:00:25Z | NaN                 | NaN        | 5.0            | Ett mycket bra hotell. Det som drog ner betyge... |

| reviews.title  | reviews.userCity | reviews.username | reviews.userProvince |
|--|------------------|------------------|----------------------|
| Good location away from the crouds   | NaN              | Russ (kent)      | NaN                  |
| Great hotel with Jacuzzi bath!   | NaN              | A Traveler       | NaN                  |
| Lugnt   ge | NaN              | Maud             | NaN                  |

### Methods used for classification

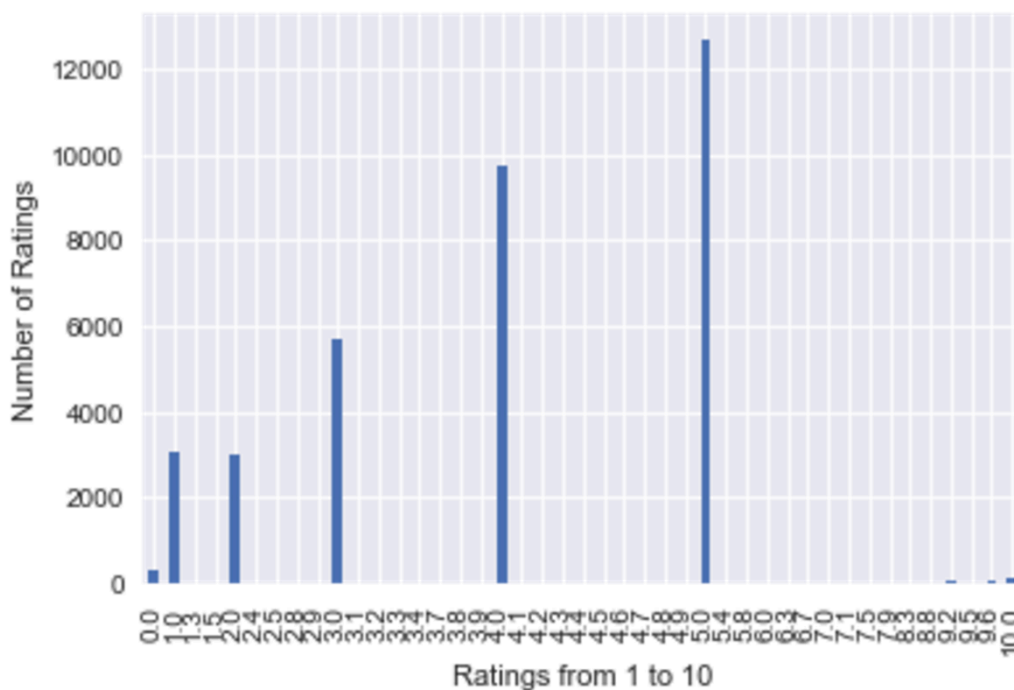
The methods used for analyzing the data and classifying the sentiments of hotel reviews include:

1. Exploratory Data Analysis
2. Machine Learning Models; and
3. Deep Learning Models (building an artificial neural network).

### Exploratory Data Analysis (EDA):

As the term indicates, EDA is performed in order to explore the data. It helps us in understanding and summarizing the main characteristics of the dataset. Exploring the data helps in framing further questions leading to collection of newer data and performing further experimentation. For the purpose of this project, EDA was performed to understand the patterns and relationships between different variables/categories. For example, from the data, patterns such as what hotels were visited the most; from which provinces/cities did users visit the hotels most; what time of the year were the most reviews given; what were the ratings given by the users for the most visited hotels and so on were observed.

This project intends to classify the sentiments of hotel reviews. For this purpose, two columns – “reviews.ratings” and “reviews.text” were chosen as the target columns for further analysis. Ratings of the reviews (given by the column reviews.ratings) were given on a scale of 1 to 10. The following graph shows a distribution of the scores.



In order to simplify the analysis in this project, we have encoded the ratings into 2 categories – 1 and 0. Instead of using numbers from 1 to 10, we have considered scores below 5 (negative) to be a 0 and scores equal to or above 5 (positive) as 1. The following graph shows the distribution of ratings after converting values to 1 and 0.



## Machine Learning Models

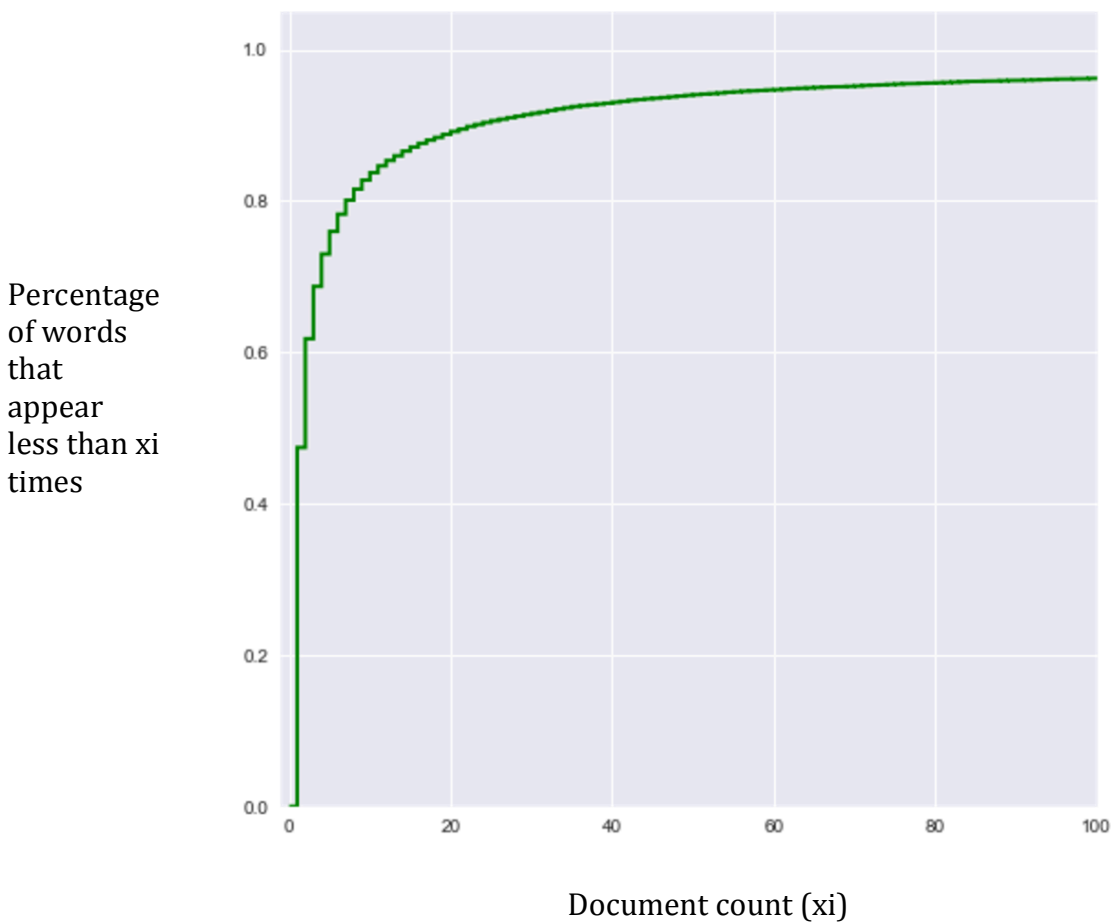
In this project, we apply 4 corpus based machine-learning models in order to classify sentiments based on reviews given by users on hotels they stayed at. These include: Multinomial Naïve Bayes Model, Logistic Regression, Decision Trees and Random Forests.

Before applying any of the above models, we first prepare and process the data to obtain a clean corpus to be fed to train the model. These preprocessing techniques include:

- Filling in missing values under the reviews.ratings column with '0';
- Drop any missing text in the reviews.text column using dropna function;
- Removing stop words – words such as “a, and, are, by, for, from, of, the”, etc. have very little value in predicting what a user is looking for. Such words are called stop words and are excluded from the data as a process of cleaning.
- Tokenization – using this function, we chop the document into small pieces (words) called token. This technique throws away unnecessary characters such as punctuation marks and leaves the words to be used for classification.
- Lemmatization (and stemming) – this technique is used to convert different forms of one word into its base word for easier classification. For instance, “girl, girls, girl’s, girls” is converted into its base word ‘girl’.
- Removal of numbers – numbers don’t play a part in classifying sentiments and therefore are excluded from the data as a process of cleaning.

### 1. Multinomial Naïve Bayes model

We use Multinomial Naïve Bayes (MNB), which is another variation of Naïve Bayes since we will be modeling word counts. The hyper-parameter used for MNB model is alpha ( $\alpha$ ). For tuning the parameters of a model, a combination of grid search and cross-validation is used. It is also essential to know what words are needed to be included in the vocabulary. For that, we first find a value for min\_df (minimum document frequency) for the count vectorizer. The min\_df value is the minimum number of documents a word should appear in, for that document to be included in the vocabulary. The following plot was constructed using the Term Frequency-Inverse Document Frequency (tf-idf) method to find the min\_df value, which is around 5.



A small value is chosen for the parameter  $\alpha$ . We use K-fold cross-validation method to choose its value. In this method, we split the data into n non-overlapping parts. We train the model using n-1 parts and perform our testing on the remaining parts. We have chosen our  $\alpha$  value as 0.01 after having performed a search over a grid of parameters. We then use the roc\_auc, precision, recall and F1-Scores as the scoring functions.

**Results of the MNB model:**

Precision score: 0.75  
Recall score: 0.75  
F1 score (harmonic mean of precision and recall scores): 0.75  
Accuracy of test data: 74.54%  
Accuracy of training data: 79.59%  
Area Under Curve (AUC) score: 0.7182

## **2. Logistic Regression model**

Logistic Regression classifier classifies an observation into either one of the two given classes. In logistic regression model, we first choose a class, which is used to compute the probability for each class. Then, the class with the maximum probability is chosen. The most important hyper-parameter to be tuned for a logistic regression model is the regularization parameter, C. C is used to control for very high regression coefficients that are usually unlikely (helps in avoiding over-fitting). It can also be used sometimes as a feature selection method, especially when the data is sparse. In this project, the C value obtained after performing grid search was 10, which was used to train the model.

### **Results of the Logistic Regression model**

Precision score: 0.75  
Recall score: 0.74  
F1 score (harmonic mean of precision and recall scores): 0.75  
Accuracy of test data: 74.29%  
Accuracy of training data: 84.53%  
Area Under Curve (AUC) score: 0.7138

## **3. Decision Tree classifier**

A decision tree classifier is widely used for different classifications. Basically, a decision tree has three types of nodes: a root node, an internal (or splitting) node and a terminal, also called as a leaf node. The root node contains all the data that needs to be classified; the internal node assigns the incoming data to a specific subgroup and the terminal node produces the final output (or the decision). There are many regularization methods for decision trees. However, we use the maximum depth hyper-parameter to tune our classifier. Maximum depth generally limits the maximum depth of the decision tree (meaning, the maximum number of nodes that any tree can hold). In this project, we have chosen the best maximum depth of the tree to be 100 for effective performance.

### **Results of the Decision Tree classifier**

Precision score: 0.69  
Recall score: 0.68  
F1 score (harmonic mean of precision and recall scores): 0.68  
Accuracy of test data: 67.84%

Accuracy of training data: 94.54%  
Area Under Curve (AUC) score: 0.6416

#### **4. Random Forest classifier**

This type of classifier is a meta estimator. It operates by constructing several decision trees during training on different smaller sub-samples from the dataset. The classifier uses averaging to reduce over-fitting and improve accuracy. The hyper-parameter for tuning random forest classifier is the number of estimators, which represents the number of trees in the forest. We chose 100 as the number of estimators for this project for maximum performance.

##### **Results of the Random Forest classifier**

Precision score: 0.77

Recall score: 0.74

F1 score (harmonic mean of precision and recall scores): 0.75

Accuracy of test data: 74.26%

Accuracy of training data: 98.17%

Area Under Curve (AUC) score: 0.6964

Looking at the 4 machine learning models above, we see that Multinomial Naïve Bayes model; Logistic Regression model and Random Forest Classifier have higher F1 scores than the Decision Tree Classifier (the higher the F1 score, the better the performance of the model).

#### **Deep Learning Model**

In this project, we have built an artificial neural network (ANN) for classifying text reviews from the hotel reviews dataset. ANNs are inspired and modeled after the neuronal structure of a human brain. ANNs are processing devices, which consist of hundreds or thousands of interconnected processing elements, responsible for processing information and producing a response or output to external inputs. Neural networks generally have several layers, which consist of a number of interconnected nodes. These nodes are responsible for an activation function. In general, neural networks can be explained as having three layers: an input layer; hidden layers; and an output layer. The input layer communicates (or passes on) information to the hidden layers, which is where all the processing of the data is done and finally sent to the output layer, where the response is produced and displayed.

##### **Data preprocessing before building a neural network**

Before building an ANN, the data has to be processed and cleaned for the model to perform effectively. The methods for filling in missing values, dropping missing review texts, lemmatization, tokenization and removal of stop words has been done

in the exact same way as the data has been processed for the machine learning models. In addition to the above methods of preprocessing, one more method called padding has been done to the data for building neural networks. Padding helps in creating a matrix with consistent dimension by cutting the sentences equal to the length of the longest sentence within the dataset. This processed data is now fed into the model directly. Most ANNs are capable of learning themselves, meaning that the weights of the connections are modified according to the inputs given to the model.

## **How an Artificial Neural Network is built**

Once the data is processed, it becomes ready to be fed to the model. We use the Multilayer Perceptron model to build our neural network. Google's TensorFlow, which is an open-source library for machine learning is used for building the network. The following are steps required for building an ANN:

First, we feed our data (weight the input data) and send it to hidden layer 1. Inside hidden layer 1, the data will get activated and the neuron will then decide if the data needs to be processed more and should be sent to hidden layer 2 or send it directly to the output layer. In our project, we have chosen 4 such hidden layers (which is why it is called a deep neural network). In case of a multilayer perceptron, we compare the output to an intended output, using a cost function, which is also known as the loss function (to see how much of a difference is there between the two). We use an activation function, called the Adam Optimizer to minimize the cost. In order to quickly lower the cost, the learning rate should be optimized. The lower the learning rate, the better the model learns and trains and gives better results.

To build our neural network, we have directly fed our data to the model, which is called forward feed. If we adjust the weights backward, it is called back propagation. We do forward feed and back propagation as many times as we wish and each cycle is called an epoch. For this project, we chose our value of epochs to be a 10.

For the model to be built, we also need to specify the number of nodes each hidden layer should contain, the number of classes and the number of batches. We chose 1000 as the number of nodes for this project. The number of classes our dataset should have is 2 (because we are classifying data as either positive or negative). The number of batches (or batch size) is 10. These are the placeholders. We are now ready to build the actual network.

We begin to define weights and biases (in order to determine the shape of the hidden layers' matrices). We input these values into the first layer (the raw input data is multiplied by its weight and then to this value, the bias is added). This process is repeated for all of the 4 hidden layers to the end (output). We now build a function that is going to run and train the neural network with TensorFlow. In order to do so, we first create a cost variable and optimize it using the Adam Optimizer. The learning rate for our optimizer has been set at 0.001. All of the required variables have now been defined and optimized. The final step in the process is to



train the actual neural network using the `train_neural_network` function. The following image is the result of the function.

```
train_neural_network(x)

Epoch 1 completed out of 10 loss: 6933294531.61
Epoch 2 completed out of 10 loss: 3124813523.2
Epoch 3 completed out of 10 loss: 1951108654.24
Epoch 4 completed out of 10 loss: 1523103915.7
Epoch 5 completed out of 10 loss: 1115793422.58
Epoch 6 completed out of 10 loss: 950260494.256
Epoch 7 completed out of 10 loss: 788498452.348
Epoch 8 completed out of 10 loss: 572315606.182
Epoch 9 completed out of 10 loss: 498729406.929
Epoch 10 completed out of 10 loss: 534235830.825
Accuracy: 0.985486
```

The accuracy above is very high (98.54%) compared to the machine learning models we have seen before, which shows that there is the problem of over-fitting. One of the methods for improving network generalization is to use larger networks in order to get a good fit.

## Conclusion

Sentiment classification is the extraction of a reviewer's opinion from texts. It is highly essential in today's world for improving businesses, conducting effective campaigning, increasing revenues, so on and so forth majorly on platforms where people depend on public opinions. In this report, we present results of 4 machine learning models and a deep learning model in sentiment classification of a hotels review dataset. We have observed that of the 4 machine-learning models, the Multinomial Naïve Bayes model, the Random Forest Classifier and the Logistic Regression model have done well in comparison to the Decision Tree Classifier. With the neural network model, we have observed the problem of over-fitting (with an accuracy of 98.54%), which will be controlled in further studies by applying specific regularization methods.