Project 5 - Sentimental Analysis Report

Katerina Tsilingiri
2806
tsaikaterini@uth.gr

Chrysa Noli
2780
nochrysoula@uth.gr

Panos Petropoulos
2610
papetropoulos@uth.gr

I. Introduction

The main goal of this project was to implement Sentiment Analysis on two different datasets and predict if the review is positive or negative. The first dataset consists of 50.000 reviews from IMDB and the second dataset consists of 271.874 reivews from the Airbnb website (for the area of Crete). The models we have used to predict the reviews are Logistic Regression, Naive Bayes Classifier, Random Forest, AdamBoost, SVM Classifier and a Deep Learning Neural Network.

II. Theory

A. Logistic Regression

Logistic Regression is a method that uses probabilistic methods in order to predict binary variables. The main function used in this method is the sigmoid :

$$\sigma(z) = \frac{e^z}{1+e^z}$$

. Regarding the evaluation of the Logistic Regression Model, a way to measure it is using the confusion matrix (as we have done later in our analysis).

Confusion Matrix is table that is widely used in order to evaluate the accuracy of a classification model. It is mainly a visual way to verify that your model predicts correctly the output and has a small error (small amount of guesses/predictions are wrong).

B. Naive Bayes Classifier

Naive Bayes is a classification method to predict the class of a datapoint from the dataset. It uses the Bayes theorem:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

where:

- $P(c|x)$: posterior probability of class
- $P(c)$: prior probability of class
- $P(x|c)$: likelihood and
- $P(x)$: prior probability of predictor

This model is very useful when the dataset contains categorical values, hence why in our analysis did not perform as well as the other models (there is a table with all the measurements in the end of this report for comparison).

C. Random Forest & AdamBoost

This model is an ensemble learning method that is used mainly in classification. Ensemble learning method means that this model uses more than one model to make predictions. The two types of models are Bagging and Boosting. In Random Forest, a new decision tee is created for every sample and each one of them will predict an output.

AdamBoost or AdaBoost belongs into the models that use the Boosting Ensemble Model that we mentioned above. The procedure that it follows is that after building a model from the training data, it creates another model that has as main goal to reduce/correct the errors of the first model. This is continued until there are no errors. Therefore, we can understand that the data we give to this classifier must be in their best form, since this algorithm learns in every iteration.

D. SVM Classifier

A support vector machine (SVM) is a supervised model that uses classification algorithms for two-labels classification problems.

More specifically, SVM Classifier succeed this by producing decision boundary (hyperplane) that best separates the groups. The optimal hyperplane we choose, it's the one that maximizes the margins from both labels or, alternatively, the one whose distance to the nearest point of each group is the largest.

It is important to highlight that the decision boundary when we are in 2 dimensions and we have linearly separable data, is a line. If we have nonlinear data, we can classify them by mapping our space to a higher dimension (kernel trick).

E. Deep Learning Neural Network

In general, a neural network, like human brain, consists of neurons that each neuron is responsible for solving a small part of a problem. They pass on what they know and have learned to the other neurons in the network, until the interconnected nodes are able to solve the problem and give an output. The key in this process is the trial and the errors that help nodes to learn better.

Parts of a Neural Network:

- Neurons: each neuron is a function that takes the output from the layer ahead of it, and spits out a number between 1 and 0.
- The input layer and input neurons
- Hidden layers: these are full of many neurons and a neural network can have many hidden layers inside

- Output layer: this is where the result comes after the information is segmented through all the hidden layers.

Deep Learning Neural Networks are these who have a lot of hidden layers between the input and the output layers. They can be used for classification, clustering and predictive analytics.

III. Sentiment Analysis on IMDB Reviews dataset

The first part of project 5, deals with the machine learning problem of predicting the number of positive and negative IMDB reviews based on sentiments, by using different classification models. This classification is achieved through the Sentiment Analysis we apply in our dataset, which help us recognise remarkable opinions contained in a movie review in order to discern whether the writer has a good or negative attitude/sentiment toward a movie.

A. Data Extraction & Exploration

The IMDB dataset we use for training and testing our application can be downloaded from the provided link. It contains in total 50000 (50K) highly polar movie reviews, equally divided into 25000 positive and 25000 negative ones. Positive are considered the reviews of them who rated the movie with up to 7 stars in IMDB and respectively negative reviews these with under 5 star rating. There are not neutral reviews in our dataset. So, the structure of our dataset:

- id
- review
- sentiment (positive or negative)

B. Data Pre-Processing

Sentiment analysis combines machine learning and natural language processing (NLP) as it is a technique for analysing a text to determine the sentiment contained within it. For this reason, it needs to preprocess and clean up the text, to highlight more easily the remarkable points/words in text which give us the information we want.

So, like every NLP problem, the data preprocess phase is a commonplace:

- text's noise removal: We expunge the noise the review may have by removing the htlm strips and the square brackets that remained from data extraction (with the support of Beautiful Soup library).

- special characters removal: Special characters don't give us information and add noise in text so we remove them.

- text stemming: Text stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. It help us analyze the meaning behind a word (with the support of PorterStemmer() of nltk library).

- stopwords removal: It's a commonplace to remove stopwords in a text as they are very common words that don't offer any meaning for our purpose. In our case, we remove english stopwords because the reviews are written in english language (with the support of nltk library).

- text tokenization: With tokenization prodecure we break the raw text into tokens of words for better management (with the support of word_tokenize of nltk library).

- text vectorization: Text Vectorization is the process of converting text into numerical representation, as the model can't understand characters. We use a Term Frequency Inverse Document Frequency (TFIDF) vectorizer which works by proportionally increasing the number of times a word appears in the document but is counterbalanced by the number of documents in which it is present (with the support of TfidfVectorizer() of sklearn library).

- label sentiment data: In our dataset, the label is a string: "positive" or "negative". It needs to transform them into numbers in order to load them to the model. The transformed labeled data:
  - 0 for negative
  - 1 for positive

  (with the support of LabelBinarizer() of sklearn library)

C. Split & Modelling the dataset

We split our entire IMDB Dataset of 50K Movie Reviews for training and testing with ratio 80:20. So, we have 40000 movie reviews for training and 10000 for testing our models.

We asked to use different classification models to evaluate and compare their performance and after all to choose which fits better in our problem. So,

1) Logistic Regression:

We first tried to train our dataset with Logistic Regression model. With the support of LogisticRegression() and its functions from sklearn library, we defined our model and then we fitted the train dataset in this. After this we used the predict() function in order to predict the output the model give us for the testing movie reviews (X_test).

Furthermore, when we evaluated our model's performance by comparing the actual output with the predicted output, we came in front of a pretty good accuracy.

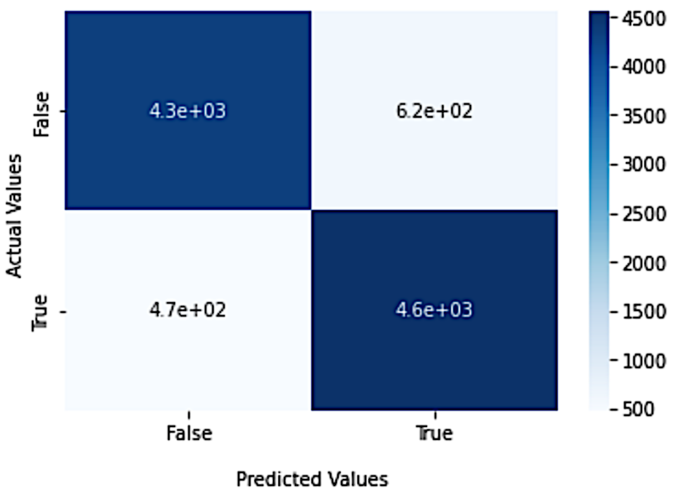
- Accuracy: 89.79%

For a better understanding of what this accuracy percent means, we plotted the confusion matrix we calculated (with the support of seaborn). As we can observe from the plot below:

- we have predicted correctly the positive reviews 4600 times (4.6e+03 = 4600)
- we have predicted wrongly the positive reviews 440 times
- we have predicted wrongly the negative reviews 580 times and
- we have predicted correctly the negative reviews 4400 times (4.4e+03 = 4400)

From the above, we can understand that our precision is very high, since the correct predictions are greater than the false.

Fig. 1. Confusion Matrix for Logistic Regression



2) *Naive Bayes Classifier:*

Similar with the Logistic Regression model, we follow the same procedure for the Naive Bayes Classifier with the support of MultinomialNB() of sklearn library. We again define our model, fit our train data in it and predict the outcome our model returns for them.

Respectively, we evaluate the performance of Naive Bayes Classifier by calculating the accuracy_score from sklearn.metrics library and we get the following result:

- Accuracy: 86.72%

Also, we generate and plot the confusion matrix that give us a more clear picture of the accuracy. We observe that:

- we have predicted correctly the positive reviews 4300 times (4.3e+03 = 4300)
- we have predicted wrongly the positive reviews 750 times
- we have predicted wrongly the negative reviews 580 times and
- we have predicted correctly the negative reviews 4400 times (4.4e+03 = 4400)

3) *Random Forest:*

For the model of Random Forest Classifier, we import RandomForestClassifier() with max_depth=10 from sklearn library. The procedure of fitting and predicting our dataset in the model is the same as above and give us a good accuracy of:

- Accuracy: 82.81%

We again use the confusion matrix for deeper understanding. The confusion matrix produced that:

- we have predicted correctly the positive reviews 4300 times (4.3e+03 = 4300)
- we have predicted wrongly the positive reviews 690 times
- we have predicted wrongly the negative reviews 1000 times and
- we have predicted correctly the negative reviews 3900 times (3.9e+03 = 3900)

Fig. 2. Confusion Matrix for Naive Bayes

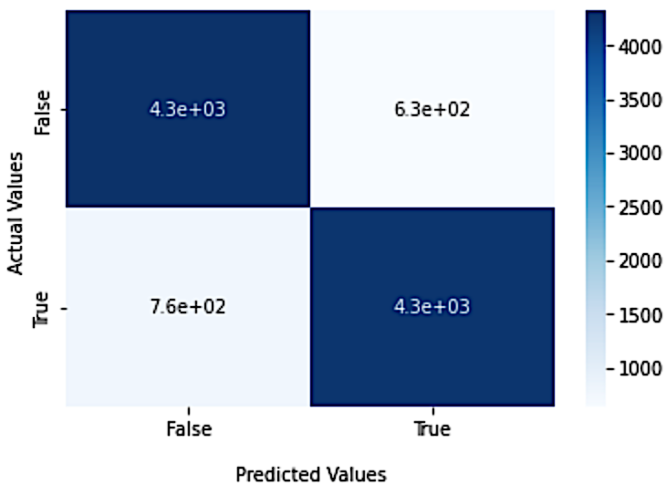
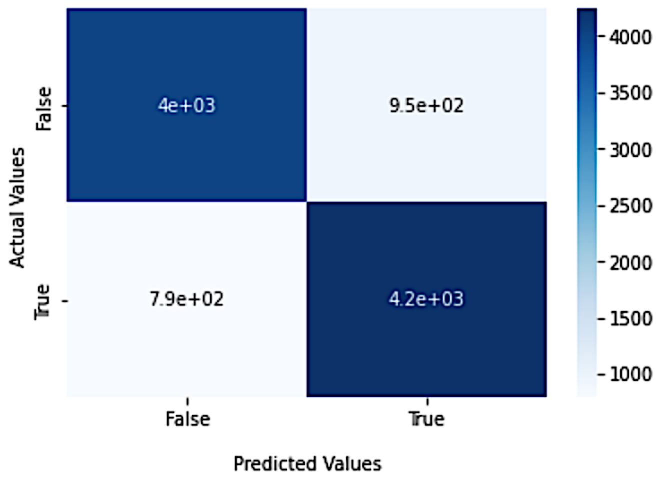


Fig. 3. Confusion Matrix for Random Forest



4) *AdamBoost:*

The next model we use is the AdamBoost model with the support of AdaBoostClassifier() with 10 estimators of sklearn library. After the fit of our train data, we predict the test reviews and we take an accuracy not so good enough:

- Accuracy: 73.09%

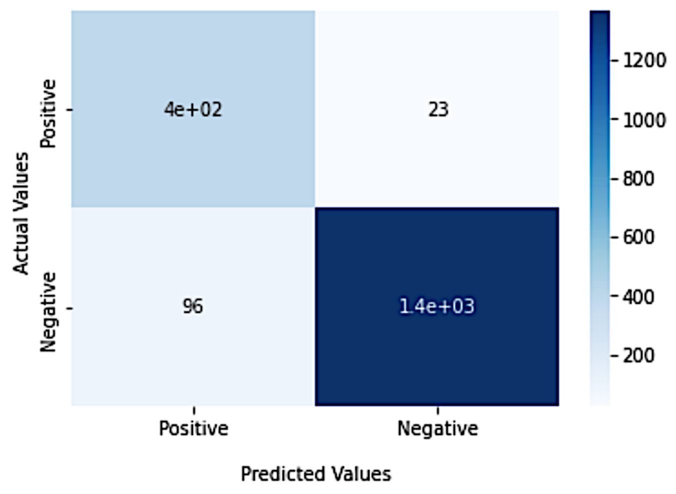
The confusion matrix we plot, confirms the above accuracy as:

- we have predicted correctly the positive reviews 4200 times (4.2e+03 = 4200)
- we have predicted wrongly the positive reviews 790 times
- we have predicted wrongly the negative reviews 1900 times and
- we have predicted correctly the negative reviews 3100 times (3.1e+03 = 3100)

5) *SVM Classifier:*

To implement this model we use SGDClassifier() from sklearn library with the hinge loss as loss function parameter

Fig. 4. Confusion Matrix for AdamBoost Classifier



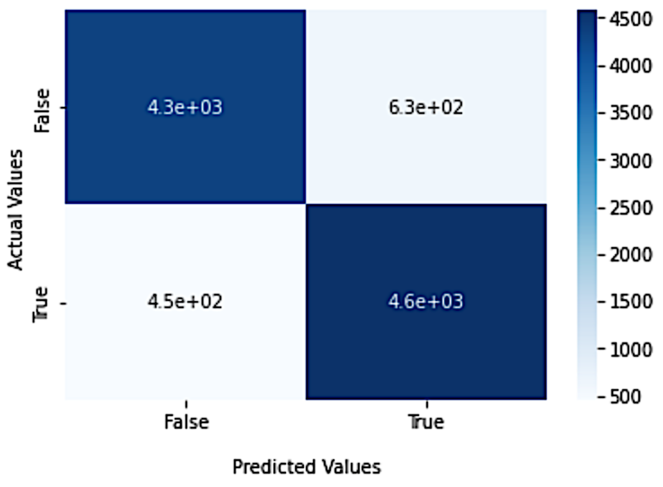
and 500 max iterations. Aftre the definition of our model, we fit our data for training and then we predict the output sentiment of our testing reviews with very high accuracy:

- Accuracy: 89.56%

This very good accuracy is confirmed by the confusion matrix we generate and plot as it gives us that:

- we have predicted correctly the positive reviews 4600 times (4.6e+03 = 4600)
- we have predicted wrongly the positive reviews 430 times
- we have predicted wrongly the negative reviews 620 times and
- we have predicted correctly the negative reviews 4300 times (4.3e+03 = 4300)

Fig. 5. Confusion Matrix for SVM



6) Neural Network:

Also, we asked to model our IMDB dataset with a deep learning neural network. We implement our Neural Network with the support of Sequential() from sklearn library. We used 3 hidden dense layers with activation functions:

1) ReLu
2) ReLu
3) Sigmoid

We compiled our neural network binary_crossentropy as loss function and with adam optimizer. Then, we fitted our train data in it, with batch size 10 and 10 epochs.

When we evaluate the performance of our neural network's implementationon testing data we get the following accuracy:

- Accuracy: 87.41%

IV. Sentiment analysis of Airbnb reviews for Greek regions

In the second part of project 5, we have to analyze the reviews of the Airbnb, in the area of Crete. More specifically, we have to study the provided comments of the visitors and separate them according to their opinion (negative or positive). Then, using different classification models, we try to predict whether the people were satisfied by the Airbnb or not.

A. Separation of the Reviews into Positive and Negative

Firstly, we have loaded the dataset from the provided link. Our dataset contains the following columns:

- listing_id,
- id,
- date,
- reviewer_id and
- comments

In our analysis, we study mainly the section of comments. Then, after having checked if there are any null values and having treated them accordingly, we have decided to keep only 10.000 samples for our analysis from 271874 samples (otherwise the analysis would take a lot of time). Our next step was to inspect the reviews to identify the various languages. We have identified Greek, English, French, Spanish, Italian and German. Therefore, we have obtained all the stop words from these languages, in order to remove them from the reviews. Since we are using a large number of reviews, we have decided to find the top 20 most common words.

Also, as we have mentioned, the reviews are in many different languages. Thus, it would be difficult to identify if the reviews are positive or negative, using only the standard dictionary (English). For this reason, we have tried to find the language of each review, generate its dictionary and then using the polarity_score from the SentimentIntensityAnalyzer(), we have created a new dataframe that contains the following columns:

- comments (in their initial form),
- compound metric,
- positivity metric,
- negativity metric and
- neutrality metric

In our analysis, we have used only the compound, which is the sum of positive, negative neutral scores and is then normalized between -1 (more negative) and +1 (more positive). Lastly, we have decided to categorize better the reviews, creating another column ('class') in our dataset, which can take only two values:

- 0 (indicating that the review is negative) and
- 1 (indicating that the review is positive)

We have separated the reviews according to the value of the compound column.

B. Data Pre-Processing

Before applying the classification models, we have to prepare our dataset. Firstly, we have decided to use the TF-IDF (Term Frequency - Inverse Document Frequency) algorithm on the comments section (our X) because of the huge amount of words. In this way, we can understand how important/relevant is a word in our reviews by giving it a value (weight). Then, we have used the train_test_split() method to train our data with ratio 80:20 (our Y is the 'class' column we have created in the Pre-Process procedure). Also, we have encoded the output y_train and y_test of the train_test_split() with LabelEncoder(). Then we have used the desired models to do the prediction.

C. Logistic Regression

For this method, we have imported the Logistic Regression module from sklearn. Firstly, we have fitted our model on the train set using the fit() function and then we have performed prediction on the test and train set using the predict() function.

Also, when we evaluated our model, we figured out that our accuracy and precision were very good.

- Accuracy : 93.67%
- Precision : 98.34%

Lastly, we have decided to also use the confusion matrix, in order to visualize better the accuracy of our model. As we can observe from the plot below:

- we have predicted correctly the positive reviews 400 times (4e+02 = 400)
- we have predicted wrongly the positive reviews 23 times
- we have predicted wrongly the negative reviews 96 times and
- we have predicted correctly the negative reviews 14000 times (1.4e+03 = 1400)

From the above, we can understand that our precision is very high, since the correct predictions are greater than the false.

D. Naive Bayes Classifier

For this method, we have followed the same procedure as in the Logistic Regression, using the sklearn. We have used the function fit() to fit our model and we have done the prediction with the corresponding function (predict()). After evaluating our model, we have gotten the following values:

- Accuracy : 89.47%
- Precision : 94.14%

Similarly, we have used the confusion matrix for better visualization.

Fig. 6. Confusion Matrix for Logistic Regression

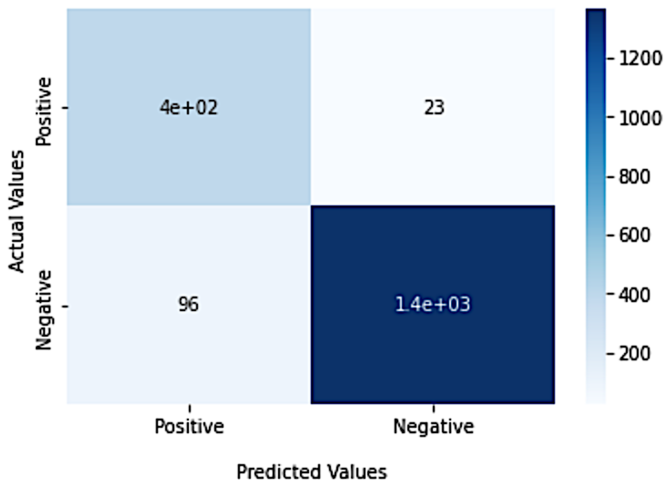
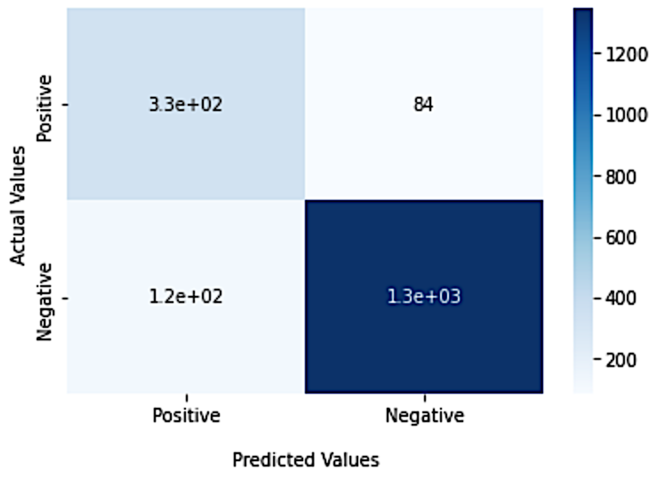


Fig. 7. Confusion Matrix for Naive Bayes Classifier



From the plot, we get that:

- we have predicted correctly the positive reviews 330 times
- we have predicted wrongly the positive reviews 84 times
- we have predicted wrongly the negative reviews 120 times and
- we have predicted correctly the negative reviews 1300 times

E. Random Forest

For the next classifier, Random Forest, we have created a Gaussian Classifier using again the sklearn. We have fitted our model and we have done the prediction with the functions that were mentioned in the above methods. Then, we have checked our accuracy, which was very good :

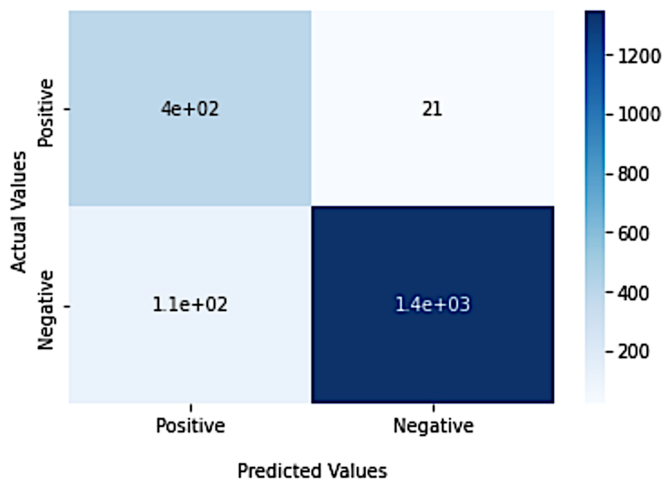
- Accuracy : 92.82%

Also, we have again used the confusion matrix, in order to get how many times we have predicted wrongly the type of the comment.

From the below figure we get that:

- we have predicted correctly the positive reviews 400 times
- we have predicted wrongly the positive reviews 21 times
- we have predicted wrongly the negative reviews 110 times and
- we have predicted correctly the negative reviews 1400 times

Fig. 8. Confusion Matrix for Random Forest Classifier



F. AdamBoost

Next, we have chosen the AdamBoost model, using the AdaBoostClassifier(). We have used 50 estimators and learning rate 1, as parameters. After fitting our train data and predicting we get the following:

- Accuracy : 95.85%

From the plot of the confusion matrix we get that using AdamBoost Classifier :

- we have predicted correctly the positive reviews 390 times
- we have predicted wrongly the positive reviews 26 times
- we have predicted wrongly the negative reviews 52 times and
- we have predicted correctly the negative reviews 140 times

G. SVM Classifier

For this method, we have used the SGDClassifier() from sklearn and we have defined the loss function using the hinge function. We have fitted our train data and then we have predicted our values. Below is shown our accuracy:

- Accuracy : 87.99%

From the plot of the confusion matrix we get that using AdamBoost Classifier :

Fig. 9. Confusion Matrix for Adam Boost Classifier

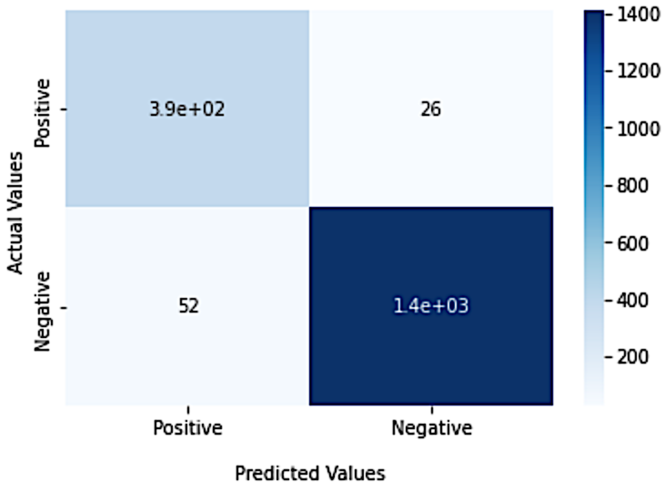
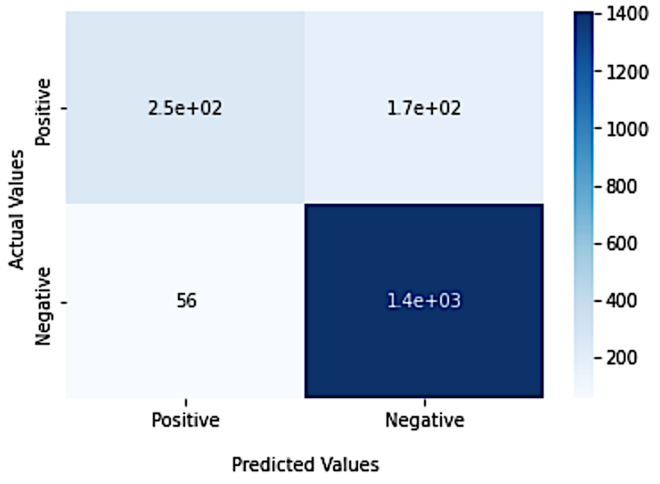


Fig. 10. Confusion Matrix for SVM Classifier



- we have predicted correctly the positive reviews 250 times
- we have predicted wrongly the positive reviews 170 times
- we have predicted wrongly the negative reviews 56 times and
- we have predicted correctly the negative reviews 140 times

H. Neural Network

Finally, for our last method, we have used a deep learning neural network. More specifically, we have used the Sequential() from the sklearn library in order to create our NN. Also, we have used 3 dense hidden layers (like in the imdb analysis) with the following activation functions:

- ReLu
- ReLu
- Sigmoid

After compiling our NN with binary_crossentropy as the loss function and adam as the optimizer, we have fitted our data with 150 epochs and 10 as batch size. With the evaluation of the deep learning neural network we get the following accuracy :

- Accuracy : 98.85%

V. Conclusion

In conclusion, we have gathered all the results regarding the accuracy of every model for both parts (IMDB and Airbnb) in the table below:

Accuracy Score		
	IMDB	Airbnb
Logistic Regression	89.79%	93.67%
Naive Bayes	86.72%	89.74%
Random Forest	82.81%	92.82%
AdamBoost	73.09%	95.85%
SVM	89.56%	87.99%
Deep Learning NN	87.41%	98.85%

References

[1] Logistic Regression- the Theory and Code *https://medium.com/@rajwrita/logistic-regression-the-the-e8ed646e6a29*,

[2] Confusion Matrix *https://en.wikipedia.org/wiki/Confusion_matrix*,

[3] 6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R *https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/*,

[4] Understanding Random Forest *https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/*,

[5] Boosting and AdaBoost for Machine Learning *https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/*

[6] Support Vector Machines (SVM) Algorithm Explained *https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/*

[7] Neural networks and deep learning explained. *https://www.wgu.edu/blog/neural-networks-deep-learning-explained2003.htmlclose*

[8] What is Deep Learning and How Does It Works [Explained] *https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-learning*

[9] Lecture Notes from our course Machine Learning