Sentiment Analysis for Amazon Fine Foods Reviews

Group 4

April, 2019

1 Introduction

Amazon and similar e-commerce websites are used vastly for online shopping purposes and these websites allow their users to write reviews about the products or services they received. These reviews have significant influence on the other users while deciding to buy a product or not. Therefore, it is valuable information to know the essence of a specific product's reviews. Furthermore, a classification made by the information gathered from these reviews can be applied to services such as product summary and product recommendation system. In this project, we intend to classify the usefulness of each product by studying their reviews using different learning models. We have implemented Linear Regression 3.3, Logistic Regression 3.4, Naive Bayes 3.5 (Gaussian 3.5.1, Multinomial 3.5.2 and Bernoulli 3.5.3), k-NN 3.6, SVM 3.7, Decision Tree 3.8, Random Forest 3.9, Neural Networks 3.10 and AdaBoost 3.11. To implement the algorithms, Python scikit-learn [1] library is used.

2 Problem Description

The main purpose of this project is to clarify the usefulness of Amazon Fine Foods products through their customer reviews. In doing that, the classifications of reviews are needed. This project aims to match the written opinions of people with their review points. As an example, one may not completely express his arguments. It may occur that a reviewer is satisfied with the product but cannot explain it clearly in his review. This project will clarify the sentiment regarding the usefulness of each product.

3 Methods & Results

3.1 Preprocessing

The comment texts in the dataset [2] have been preprocessed. The texts have been changed to lowercase and removed non alphabatic characters and stop words. Also, words br, product, just, coffee, food, tea, amazon, really, time, ve and use are also removed from the dataset (See Figure 1) becuase they don't have an effect on the distinction of positive and negative classes.
The positive and negative classes are generated by the ratings which are given by the users. We have used rating 4 and 5 as positive and 1, 2, 3 as negatives. There are 443,777 positives and 124,677 negatives in the dataset. This imbalance situation has caused a problem for prediction approximately every comment as positive. Thus, we have randomly selected 124,677 positive comments to combine with negatives. The dataset is under sampled and the imbalance problem has solved.

The dataset has randomly split into 2 pieces for train and test. %75 of the data is used for train and remaining %25 is used for test.

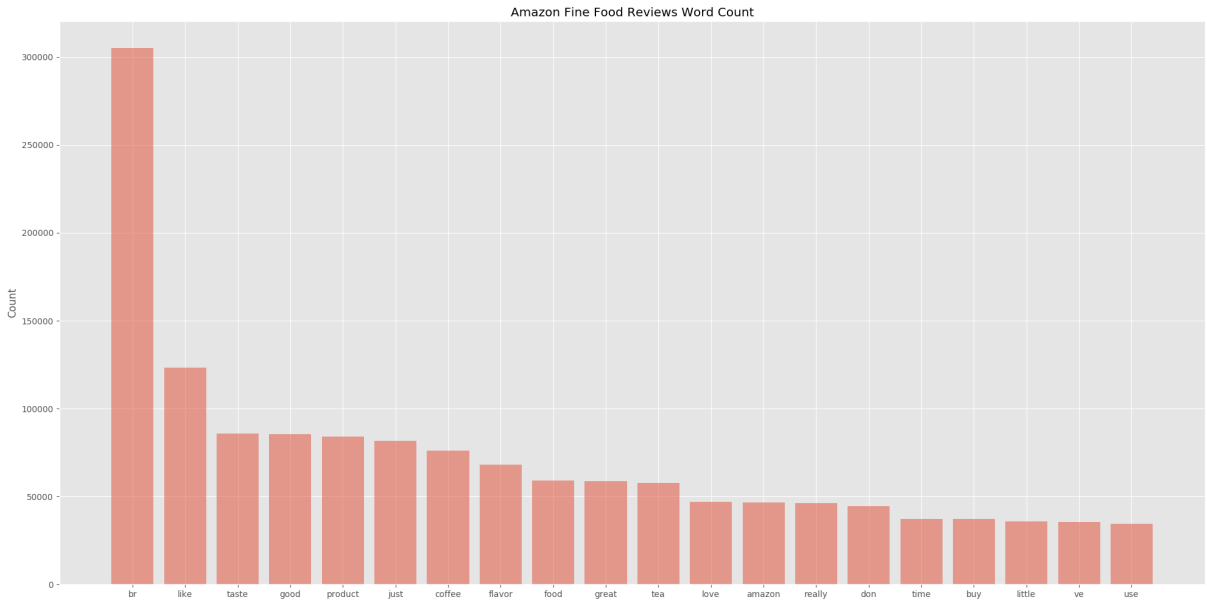

Figure 1: Most 20 words in the dataset

3.2 Metrics

TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{f1-score} = 2 \times \frac{precision \times recall}{precision + recall}$$

$$\text{accuracy} = \frac{TP + TN}{all}$$

Receiver Operating Characteristic(ROC) curve is the plot of "Spesificity" on the x-axis and "Sensitivity" on the y-axis, true positive rate and true negative rate respectively.

Precision-Recall curve is the plot of "Precision"on the y-axis and "Recall" on the x-axis.

3.3 Linear Regression

Linear regression is used to find linear relation between predictors and target. Linear Regression is not suitable for binary classification because the predicted value is continuous, not probabilistic, this problem can be seen in Figure 2.

Confusion Matrix	
TPs: 25066	FPs: 5992
FNs: 5122	TNs: 25749

	precision	recall	f1-score	support
positive	0.81	0.83	0.82	31052
negative	0.83	0.80	0.82	31287

Accuracy = 0.8151
Mean Squared Error = 0.19
Variance score: 0.26

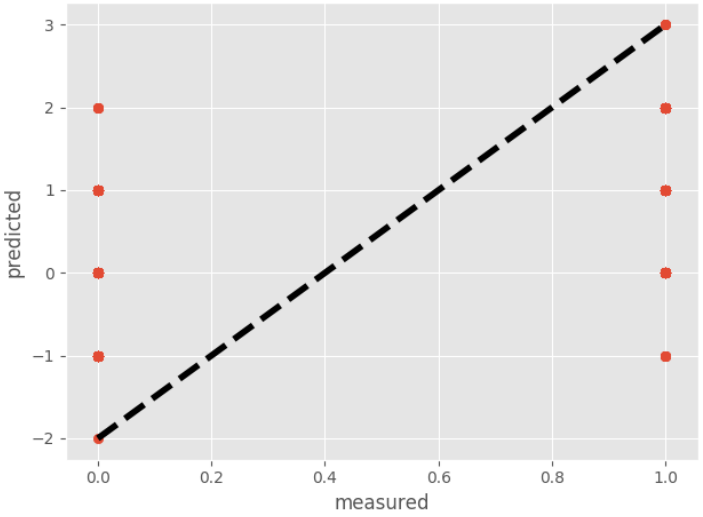

Figure 2: Linear Regression

3.4 Logistic Regression

Logistic regression is a statistical method to analyze a dataset where there are one or more independent variables that determines a categorical outcome. We have used SAGA [3] algorithm to solve the optimization problem of logistic regression.

Confusion Matrix	
TPs: 25494	FPs: 5793
FNs: 5199	TNs: 25853

	precision	recall	f1-score	support
positive	0.82	0.83	0.82	31052
negative	0.83	0.81	0.82	31287

Accuracy = 0.8236

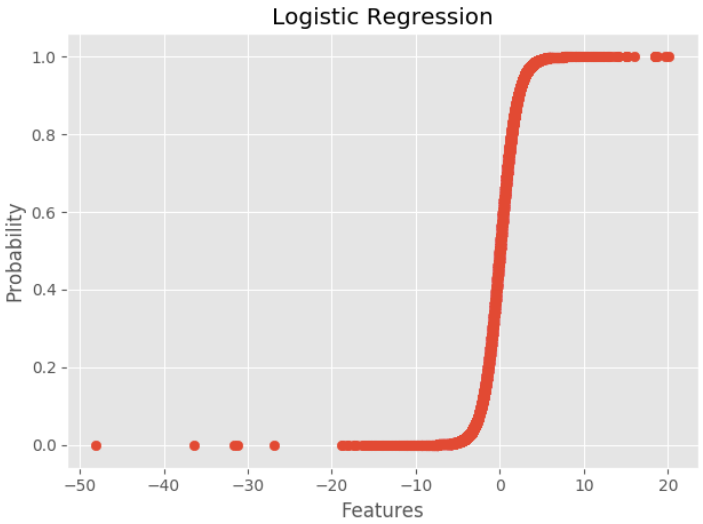

Figure 3: Logistic Regression

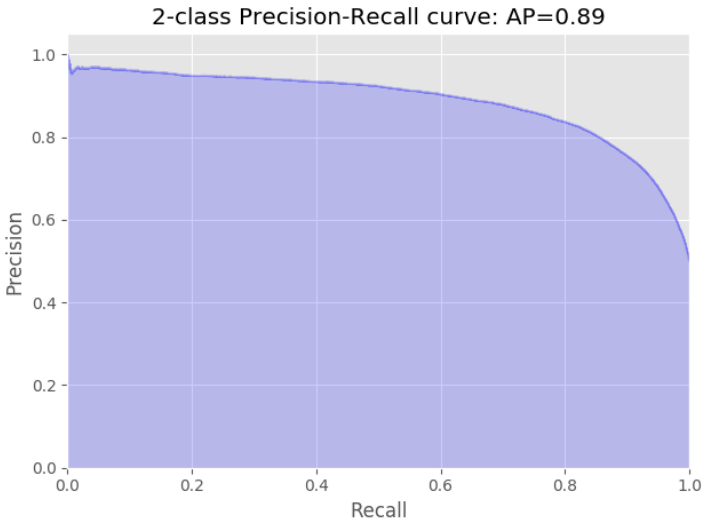

Figure 4: Logistic Regression Precision Recall Curve

3.5 Naive Bayes

Naive Bayes classifiers are a set of probabilistic classifiers that applies Bayes' Theorem with naive independence assumptions among features. 3 different Naive Bayes classifiers are used to make predictions.

3.5.1 Gaussian

When the data is continuous, a simple assumption is that continuous values are distributed according to a Gaussian probabilistic density function.

Confusion Matrix	
TPs: 17480	FPs: 13807
FNs: 4709	TNs: 26343

	precision	recall	f1-score	support
positive	0.66	0.85	0.74	31052
negative	0.79	0.56	0.65	31287

Accuracy = 0.7029

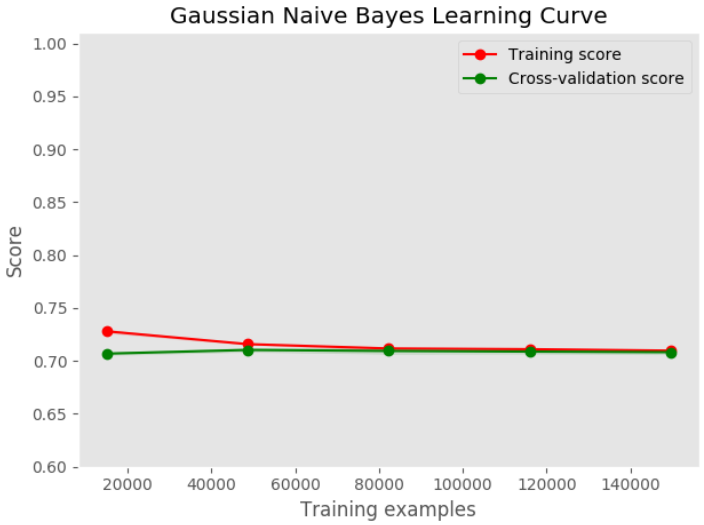

Figure 5: Gaussian Naive Bayes learning curve

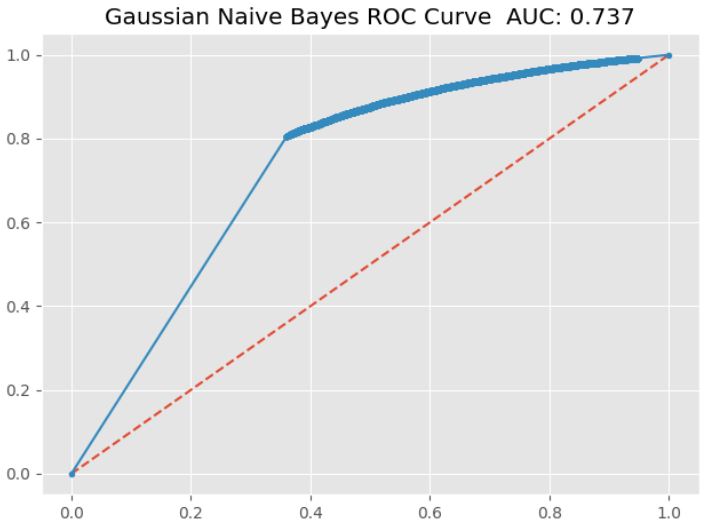

Figure 6: Gaussian Naive Bayes ROC curve

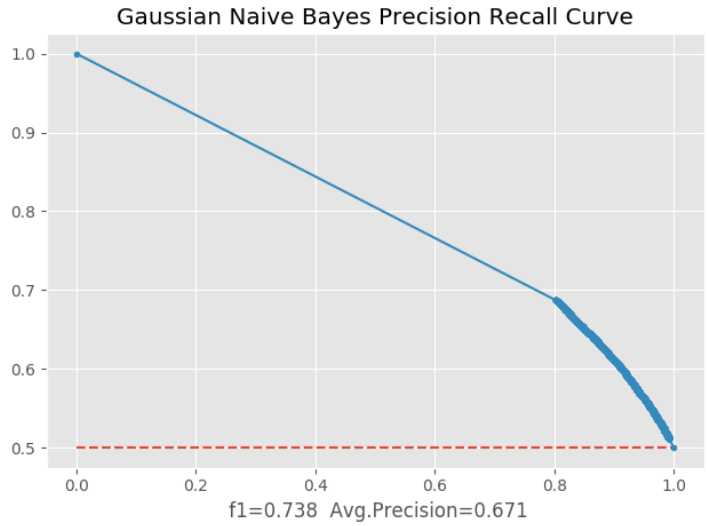

Figure 7: Gaussian Naive Bayes Precision Recall curve

3.5.2 Multinomial

In a multinomial model, features represent the frequencies which certain events have been generated by a multinomial model. The training data has been transformed to TF-IDF (Term Frequency–Inverse Document Frequency).

$$TF(t) = \frac{\text{Number of times term t appears in a review}}{\text{Total number of terms in the document}}$$

$$IDF(t) = ln\frac{\text{Total number of comments}}{\text{Number of comments with term t in it}}$$

Confusion Matrix	
TPs: 24916	FPs: 6371
FNs: 5595	TNs: 25457

	precision	recall	f1-score	support
positive	0.80	0.82	0.81	31052
negative	0.82	0.80	0.81	31287

Accuracy = 0.8080

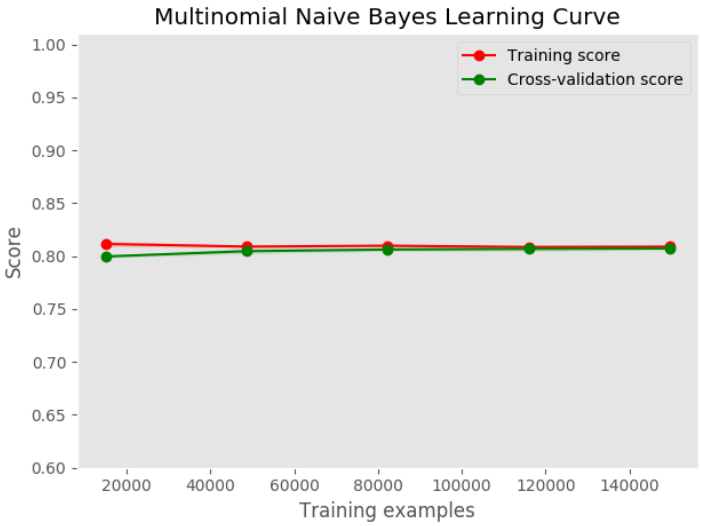

Figure 8: Multinomial Naive Bayes learning curve

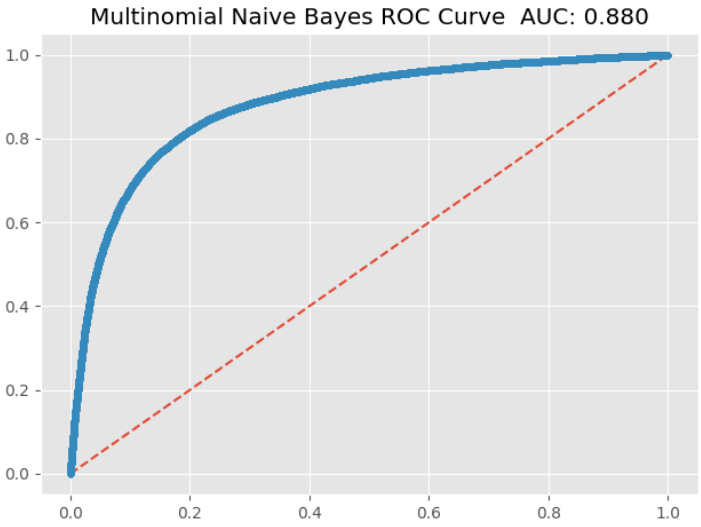

Figure 9: Multinomial Naive Bayes ROC curve

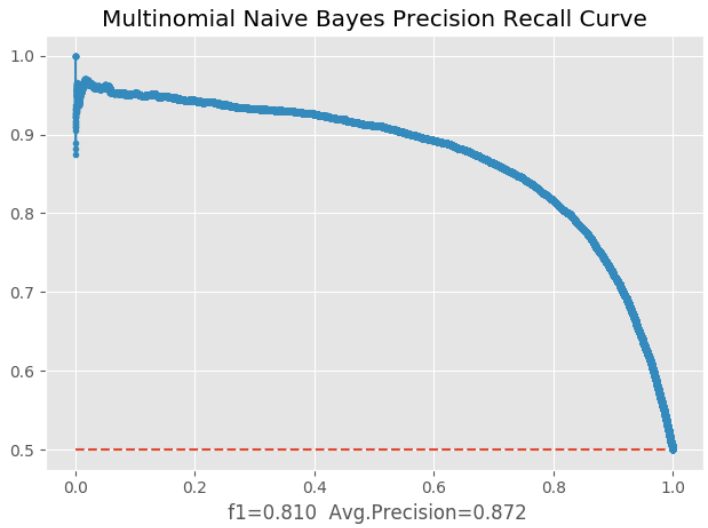

Figure 10: Multinomial Naive Bayes Precision Recall curve

3.5.3 Bernoulli

In a Bernoulli model, features are independent binary variables describing inputs. Similar to the multinomial model, it is mostly used for text classification, where binary occurrence features are used instead of frequencies.

Confusion Matrix	
TPs: 22171	FPs: 9116
FNs: 5263	TNs: 25789

	precision	recall	f1-score	support
positive	0.74	0.83	0.78	31052
negative	0.81	0.71	0.76	31287

Accuracy = 0.7693

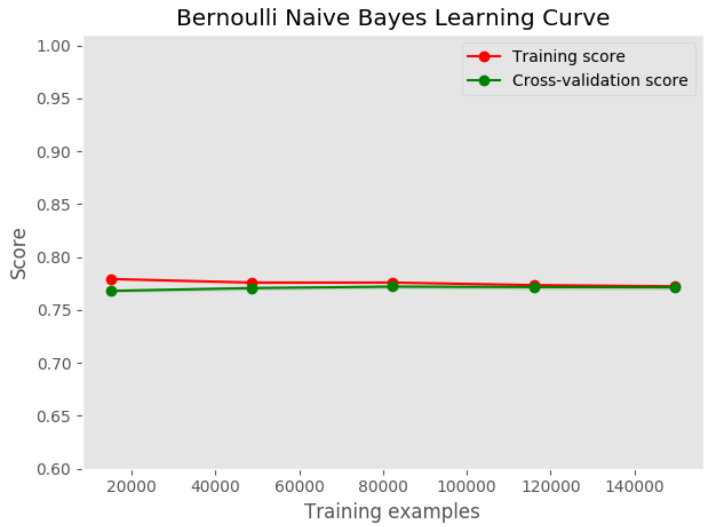

Figure 11: Bernoulli Naive Bayes learning curve

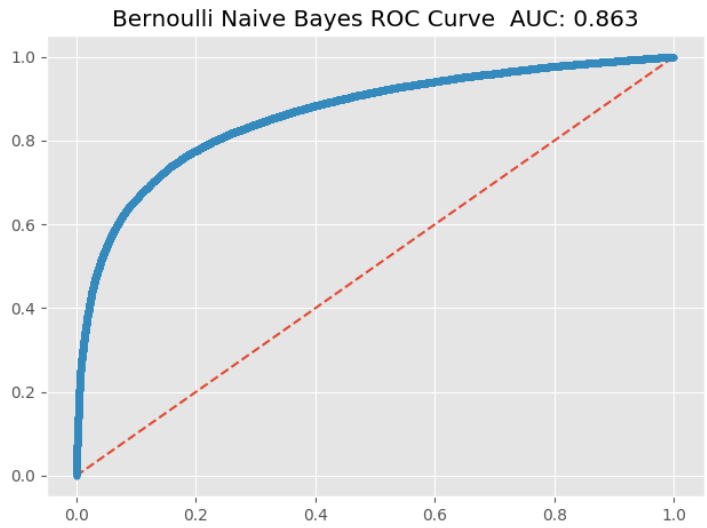

Figure 12: Bernoulli Naive Bayes ROC curve

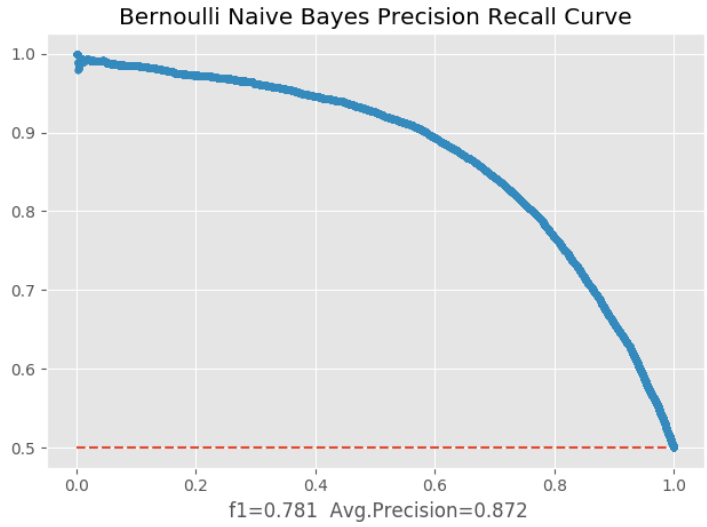

Figure 13: Bernoulli Naive Bayes Precision Recall curve

3.6 k-NN

kNN algorithm is a simple lazy learning algorithm that predicts a sample by looking at its "k" neighbours' classifications. In our case, kNN algorithm first takes our train data then predicts our new test data. For this, it takes the distance between each train data entry and the new test data. Then it looks at the closest "k" data's classifications and predict our new data as the majority of these classifications. In our project, we wanted to experiment and find what value of "k" would be the perfect choice for us. As it can seen from the Figure 14, we looked at k=2 to 9. This operation took almost 15 hours. In the end, we concluded that k=3 is the best choice. The confusion matrix and accuracy scores are based on k=3.

Confusion Matrix	
TPs: 22171	FPs: 9116
FNs: 5263	TNs: 25789

	precision	recall	f1-score	support
positive	0.73	0.74	0.72	31052
negative	0.71	0.74	0.73	31287

Accuracy = 0.7328

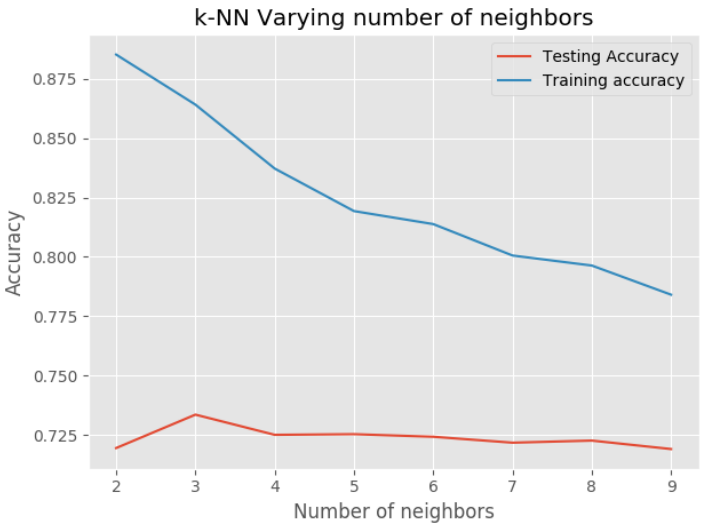

Figure 14: k-nn finding number of neighbours

3.7 SVM

The aim of the Support Vector Machine (SVM) algorithm is a discriminative classifier by a separating hyperplane in an N-dimensional space (N being the number of features) that categorizes data points.

Confusion Matrix	
TPs: 25399	FPs: 5888
FNs: 5133	TNs: 25919

	precision	recall	f1-score	support
positive	0.81	0.83	0.82	31052
negative	0.83	0.81	0.82	31287

Accuracy = 0.8232

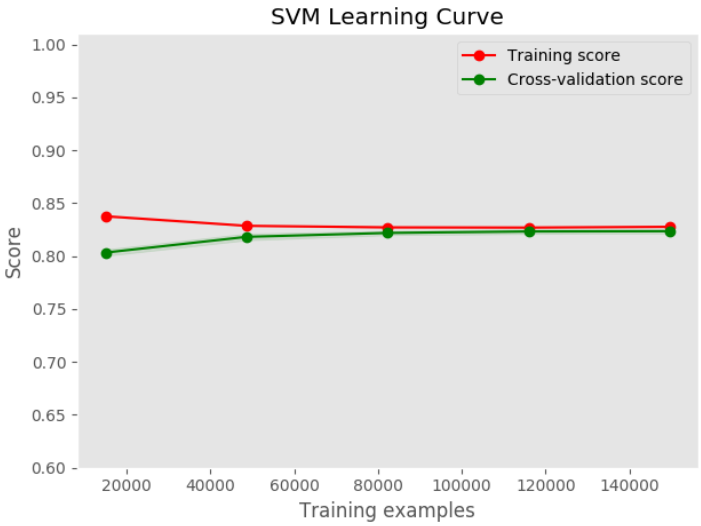

Figure 15: SVM Learning Curve

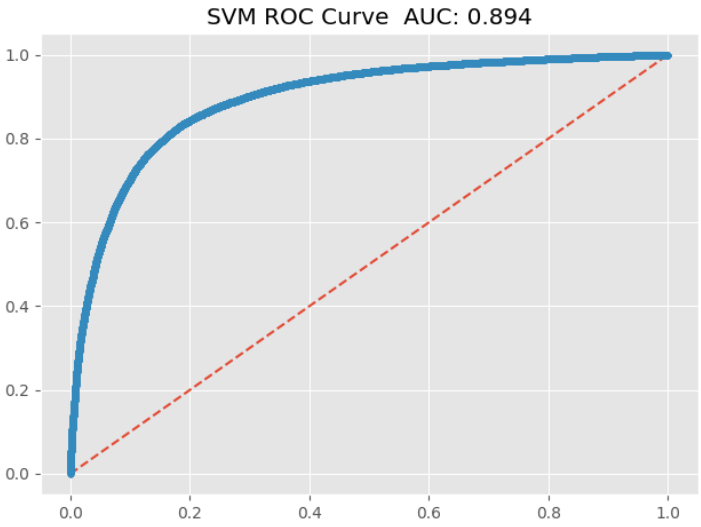

Figure 16: SVM ROC Curve

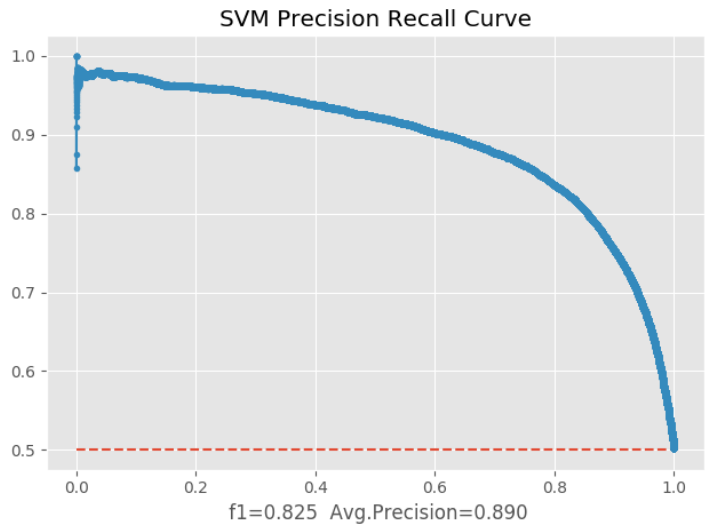

Figure 17: SVM Precision Recall Curve

3.8 Decision Tree

Decision Tree algorithm, another member of supervised learning algorithms, tries a solution by using "tree representation". Each internal node of the tree corresponds to an attribute while each leaf node corresponds to a label.
To indicate how deep the tree should be we tried the depths ranging from 1 to 32 (See Figure 21).
To indicate the minimum number of samples required to split an internal node we changed the *min_samples_split*. When we increase this parameter, the decision tree became more constrained because it has to consider more samples at each node. We changed *min_samples_split* from 10% to 100% of the samples. We can see that when we consider 100% of the samples at each node, the model cannot learn enough about the data. This is an underfitting case (See Figure 22).
min_samples_leaf is the parameter to select minimum number of samples required to be at a leaf node. This parameter is similar to *min_samples_splits*, however, *min_samples_leaf* describe the minimum number of samples of samples at the leafs. Same conclusion as *min_samples_splits*. Increasing this value caused underfitting (See Figure 23).

Confusion Matrix	
TPs: 24995	FPs: 6292
FNs: 6340	TNs: 24712

	precision	recall	f1-score	support
positive	0.80	0.80	0.80	31052
negative	0.80	0.80	0.80	31287

Accuracy = 0.7973

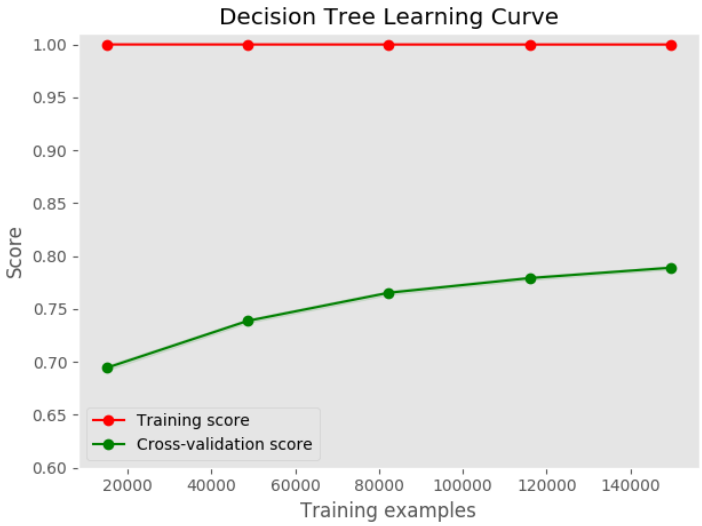

Figure 18: Decision Tree Learning Curve

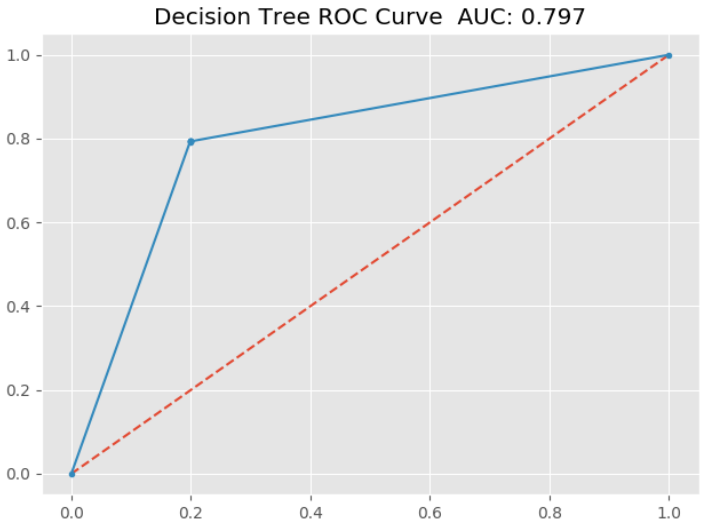

Figure 19: Decision Tree ROC Curve

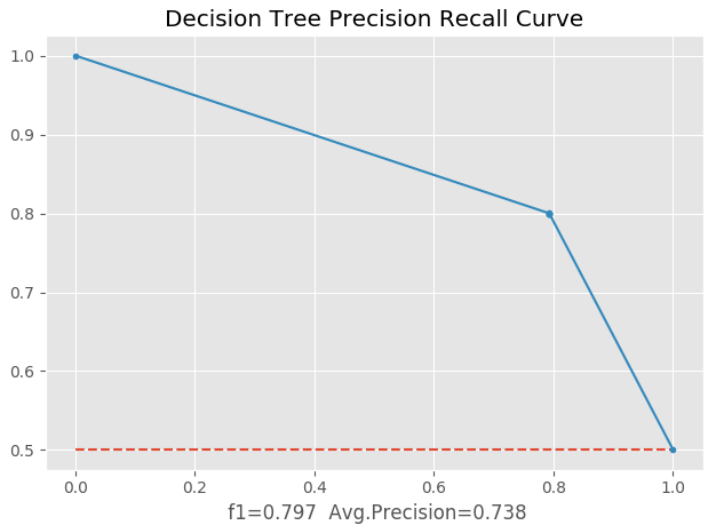

Figure 20: Decision Tree Precision Recall Curve

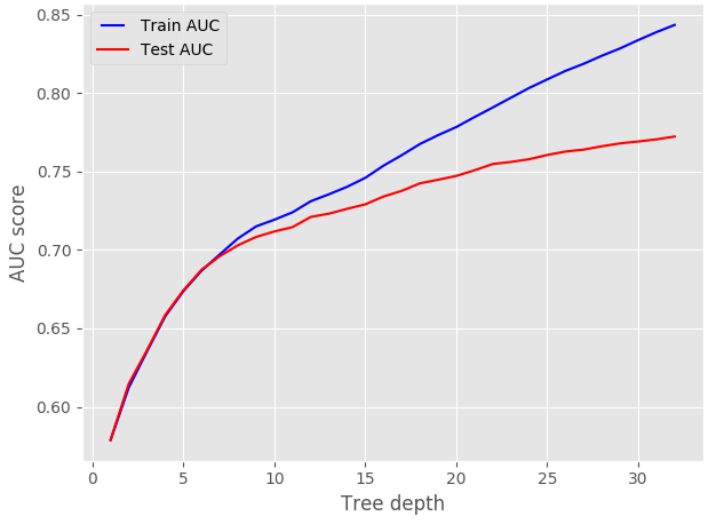

Figure 21: Decision Tree Max Depth

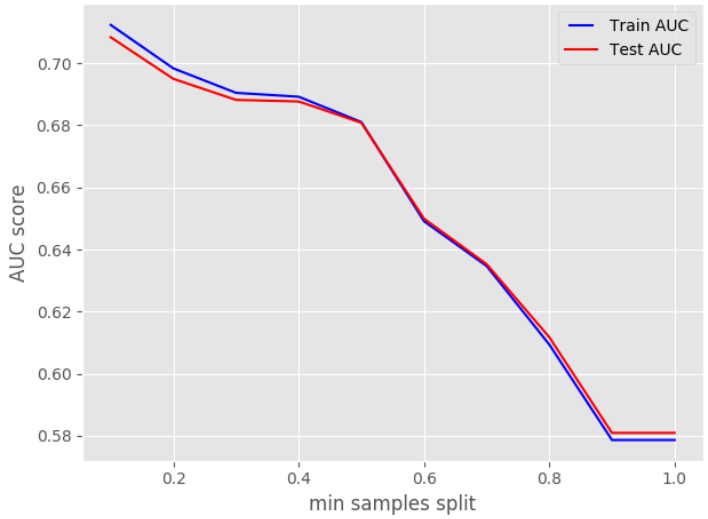

Figure 22: Decision Tree Min Samples Split

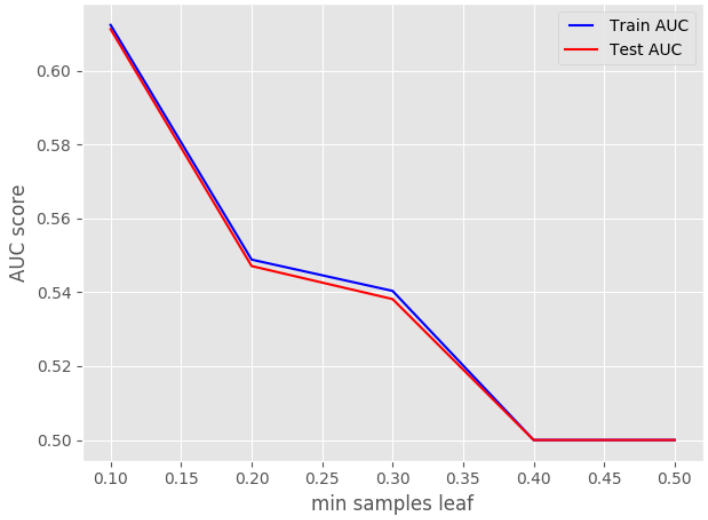

Figure 23: Decision Tree Min Samples Leaf

3.9 Random Forest

Random forest is a supervised learning algorithm which builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Confusion Matrix	
TPs: 27292	FPs: 3995
FNs: 5919	TNs: 25133

	precision	recall	f1-score	support
positive	0.86	0.81	0.84	31052
negative	0.82	0.87	0.85	31287

Accuracy = 0.8409

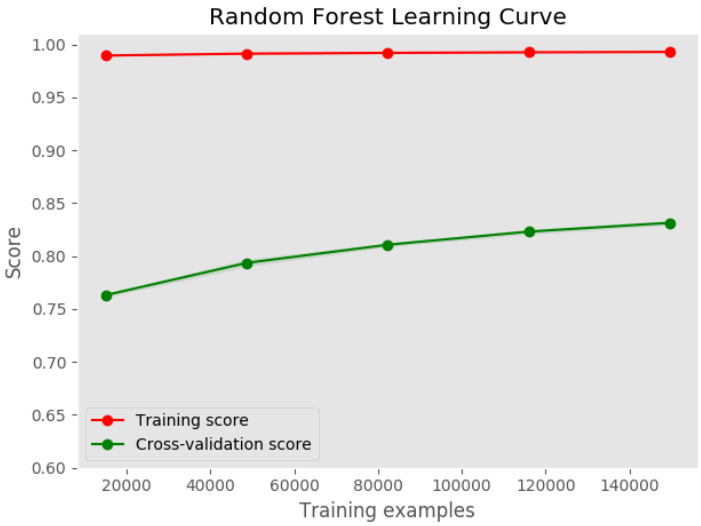

Figure 24: Random Forest Learning Curve

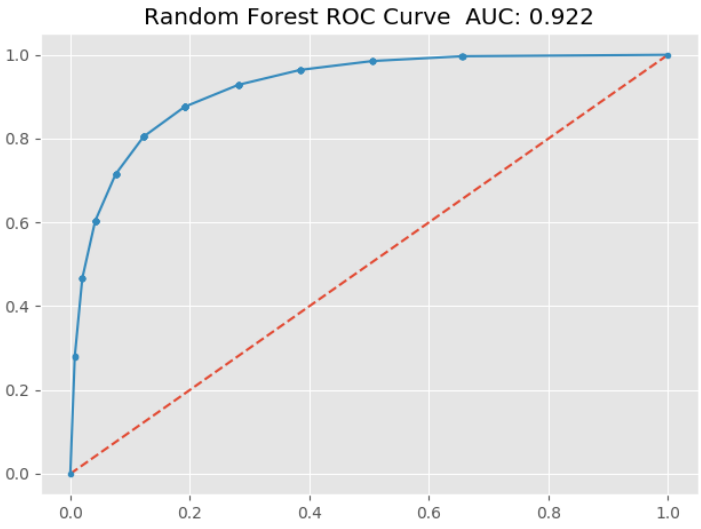

Figure 25: Random Forest ROC Curve

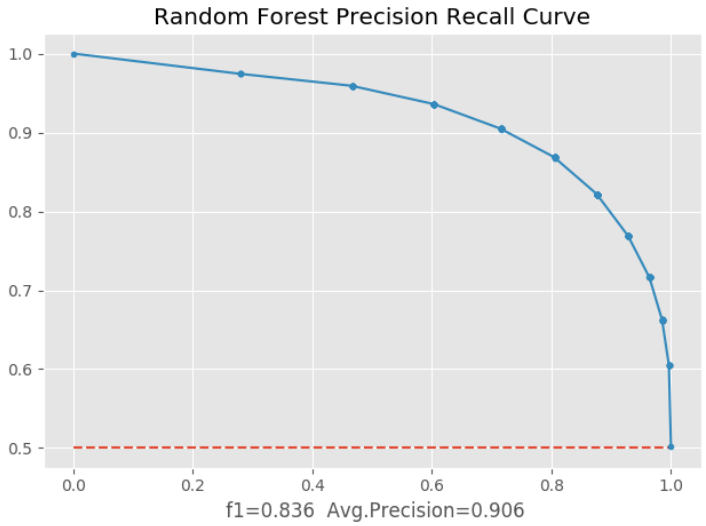

Figure 26: Random Forest Precision Recall Curve

3.10 Neural Networks

Neural networks are a group of algorithms that are designed for recognition of patterns. The purpose is to group unlabeled data, according to similarities among inputs.

Confusion Matrix	
TPs: 26645	FPs: 4642
FNs: 4428	TNs: 26624

	precision	recall	f1-score	support
positive	0.85	0.86	0.85	31052
negative	0.86	0.85	0.85	31287

Accuracy = 0.8545

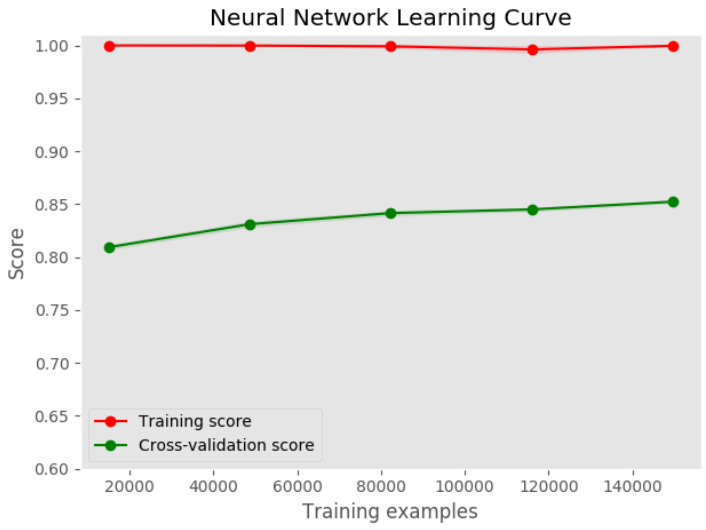

Figure 27: Neural Networks Learning Curve

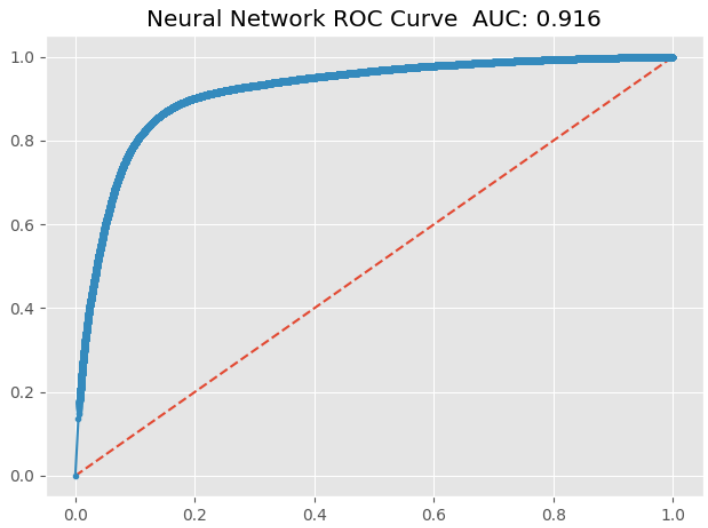

Figure 28: Neural Networks ROC Curve

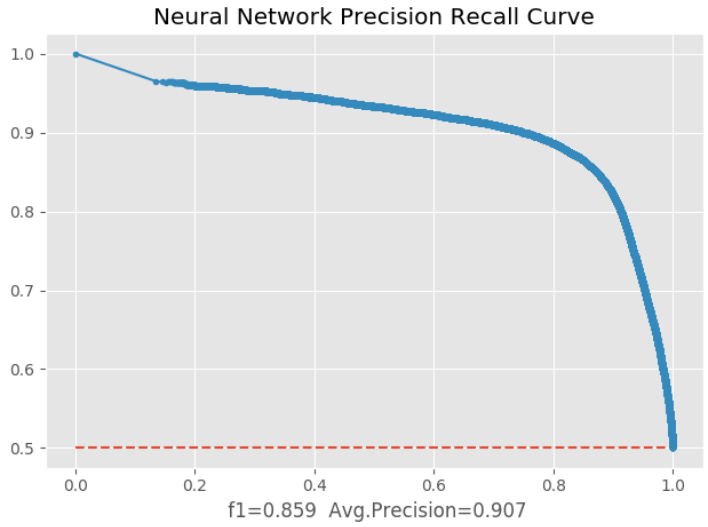

Figure 29: Neural Networks Precision Recall Curve

3.11 AdaBoost

AdaBoost (Adaptive Boosting) is a practical boosting algorithm which focuses on classification problems and aims to convert weak classifiers into a strong one. In this project we selected Decision Trees as a weak classifiers.

Confusion Matrix	
TPs: 24645	FPs: 6642
FNs: 5428	TNs: 25624

	precision	recall	f1-score	support
positive	0.81	0.82	0.81	31052
negative	0.82	0.81	0.81	31287

Accuracy = 0.8156

4 Discussion & Conclusion

Considering all of the work, coding and machine learning algorithms that was done, it was a good practice to learn the concepts of this class. According to the results of each of the algorithms, it can be said that every one of them were successful, but there are differences and preferences about the accuracy and process times. Briefly, k-NN was the slowest one (nearly 15 hours) among all while Multinomial Naive Bayes being the fastest. The most accurate results were obtained via Neural Networks, even though the remaining ones also gave similar and high accuracy values. Any future work might be improving the accuracy and decreasing process time for every algorithm.

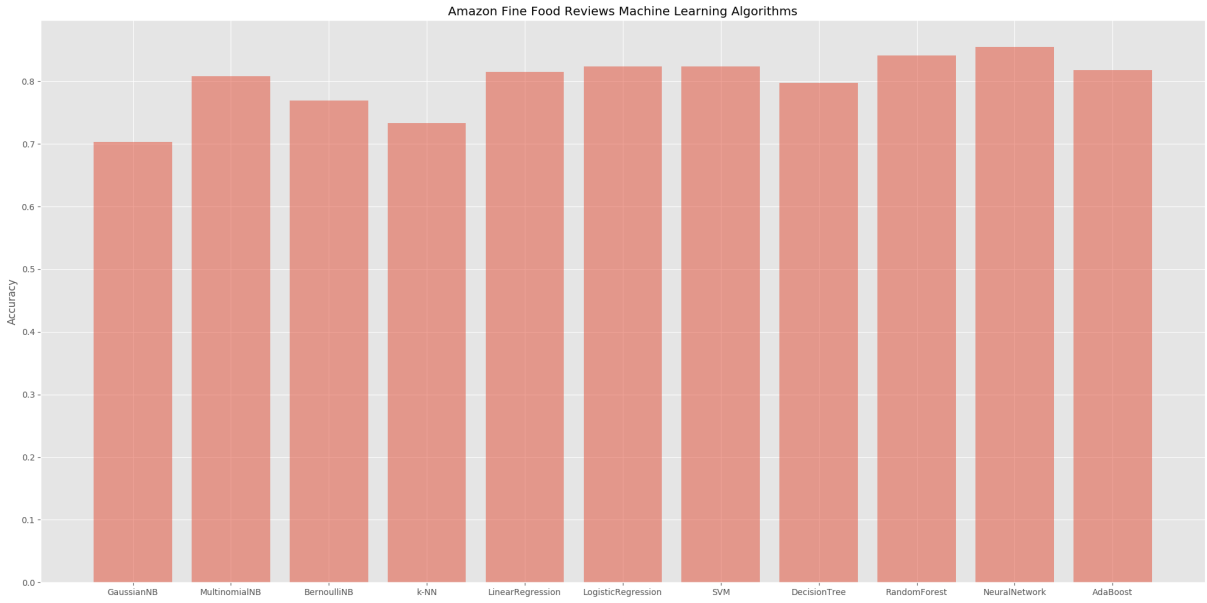

Figure 30: Amazon Fine Food Reviews Machine Learning Algorithms

5 Work Division

- Boran Yildirim: preprocessing, Linear Regression, Logistic Regression, Neural Networks, Decision Tree
- Deniz Evrensel: Gaussian Naive Bayes, SVM
- Batihan Akca: Multinomial Naive Bayes, AdaBoost
- Furkan Arif Bozdag: Bernoulli Naive Bayes, AdaBoost
- Sekip Kaan Ekin: k-NN, Random Forest

References

[1] scikit-learn Machine Learning in Python https://sklearn.org

[2] Dataset used in this project https://snap.stanford.edu/data/web-FineFoods.html

[3] SAGA – Defazio, A., Bach F. & Lacoste-Julien S. (2014). SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives https://arxiv.org/abs/1407.0202