# MLDM Coursework: *Oppenheimer*

**Anthony Awobasivwe** AA02350@SURREY.AC.UK

## Abstract

Machine learning algorithms have long been used as a powerful tool to solve different types of problems. In this project we will be leveraging this power to train two real-world datasets obtained from Kaggle on various supervised learning approaches and evaluating performance, by doing so we aim to get a better understanding of model performances across different kinds of data and investigate factors that affect how models perform.

## 1. Project Definition

There are two tasks we aim to accomplish both within the supervised learning paradigm with datasets obtained from Kaggle.

**Dataset 1 – Telco Customer Churn**

Dataset 1 is made up of 7043 rows and contains 20 features and a target label called Churn which indicates if a customer canceled their telephone contract or not. The major question to address from the dataset is how accurately customer churn predictions are based on factors such as their tenure period, usage, billing information, etc. Exploratory data analysis would help companies identify what factors influence customers who choose to churn and model predicting for this dataset would help companies better prepare for potential churners. The link to this dataset can be found below: https://www.kaggle.com/datasets/blastchar/telco-customer-churn

For dataset 1 an assumption is made that the continuous feature data follows a normal distribution hence we can also hypothesize that we expect the gaussian naïve bayes model to perform better than multinomial naive bayes

***Dataset 2 – IMDB Movie Reviews***

Dataset 2 is made of 50k rows and contains one feature called text and a target label called sentiment . The aim of this dataset is to predict if a given movie review text holds a positive or negative sentiment . Performing exploratory data analysis and modeling on this dataset could help show overall audience sentiment to various movies and potentially identify patterns which contribute to movie success.

The link to this dataset can be found below:

https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

For dataset 2 an assumption is made that categorical features are independent of each other . We also assume all text features are provided in English language.

In terms of modeling for both datasets 1 and 2, I will be conducting experiments on four distinct machine-learning approaches. The first of which is tree-based vs ensemble learning. In this domain, I will be comparing the performance of tree-based models i.e., decision trees to ensemble approaches such as random forests and adaptive boosting. The second experiment involves conducting experiments to evaluate model performance between a single-layer perceptron and a multi-layer perceptron across both datasets. The third experiment involves using a distance-based learning approach i.e., k nearest neighbors. In this experiment, I will explore different methods for finding the optimal k value to be used for modeling. The final experiment involves exploring two naive models within the Bayesian learning domain i.e., Gaussian and Multinomial Naive Bayes .

For each machine learning approach used, the models will be compared to the others within its domain. The best performing models from each experiment are then selected and compared with each other.

For both datasets evaluation metrics such as accuracy, precision, recall, and f1 score , AUC were considered. For datasets with a more imbalanced target label, more focus was placed on the AUC when considering overall performance.

**Hypothesis**

1. Gaussian Naive Bayes to outperform Multinomial Naive for dataset 1 . Multinomial Naive Bayes to outperform Gaussian Naive for dataset 2 .
2. Ensemble learning approach to outperform single tree-based approach across both datasets.
3. Optimal k value less than 25 for both datasets
4. Multi-Layer Perceptron to outperform single layer perceptron across both datasets.
5. Out of all the best-performing models within each domain, no model will perform significantly better than the other. Significance = +10 difference in chosen evaluation metric

## 2. Data Preparation

### 2.1 Data cleaning / data integration

For both datasets, we measure high quality of data based on factors such as ensuring no missing data and no

extremely unexplainable imbalance in class labels. For dataset 1 specifically, as this is continuous data, we also ensure there are no unexplained outliers. For dataset 2 as this is text data, we measure high-quality data based on factors such as making sure the data is tokenized, performing lemmatization, removal of stop words, etc. for focus to be placed on the content rather than metadata.

Dataset 1 came precleaned however there were still a few checks to ensure good data quality. The first check was checking for missing values in which a few were found. As there were only a few rows with missing values about 10 of those rows were removed completely. The next check was checking for outliers. By visualizing the continuous features through boxplots and following the IQR outlier rules, a few outliers were found. I decided to leave these outliers as they seemed reasonable e.g., a customer with extremely high monthly charges. The last check done was checking the distribution of class labels. The distribution of class labels(Churn vs no Churn ) showed about a 3:1 majority split for the no-churn class. This was an interesting observation however it is sensible that any company won't have most of their customers leaving so this wasn't further explored through methods such as over and under-sampling. However, our models do not have this domain knowledge so to combat any biased results caused by this imbalance I made use of stratified k-fold cross validation which ensured each fold has a representative distribution of samples from each class label. Dataset 1 also had a relatively normal number of features so no dimensionality reduction was done . A new feature called contract type which is based on the monthly charge was also created .

Dataset 2 also came precleaned so there was no missing data and no possibility of outliers as the data is text-based. From the label distribution, we see almost equal distribution between the positive and negative sentiment labels was balanced so there was no need for over/under-sampling on the dataset. As this is a text-based dataset, several more text preprocessing techniques such as tokenization, lemmatization, stop word removal, punctuation removal, etc. were done to get the data ready to be passed into a model. Finally, a count vectorizer was used as the chosen method for text representation.

### 2.2 Data exploration / data visualization

**Dataset 1**

Various boxplots were explored such as gender types of vs tenure period in relation to churn status however these in most cases didn't provide any interesting observances. One interesting observation was the tenure feature in relation to churn. By creating a churn rate feature and comparing this relation to tenure periods via a scatterplot we see most customers who decide to churn do so within the first year or 2 with telco making the tenure period an extremely important feature. This was further highlighted in the correlation plot later done. The charts for this are displayed below.

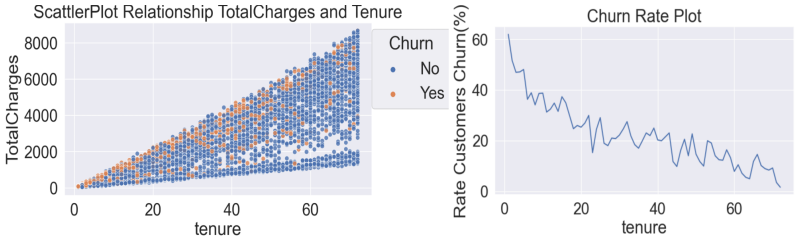



*Figure 1*. Scatterplot total charges vs tenure in relation to churn (Left) and churn rate by tenure period (Right)

**Dataset 2**

Interestingly after exploring character and word distribution between both labels this was also eerily similar. I then explored the most common unigrams in the text as seen in figure (2) for which most were stop words hence these were removed .

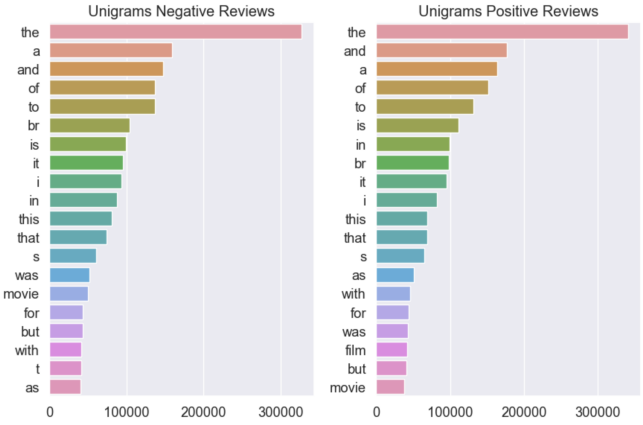



*Figure 2*. Unigram analysis for most common words

## 3. Model development

### 3.1 Tree Based / Ensemble Algorithms

Tree-based and ensemble learning are supervised learning approaches that address classification or regression problems by constructing tree-like structures when modeling which are then used to make predictions. As mentioned earlier three subsets of this were considered. For each model, two hyperparameters were selected for tuning via a grid search to maximize performance. This was set to use 2 hyperparameters as excessing training time and model complexity is also considered.

1. **Decision Trees**

Decision Trees are a non-parametric supervised learning method that can be used for classification problems and is built using one tree only. The decision tree predicts the target variable based on a set of decision rules inferred from the data features after building the tree. The tree is made up of three types of nodes, the root node, which is the top of

the three, the decision nodes where decisions are made for further splits, and finally, the terminal node which is the endpoint of the tree where predictions are output [5]. The decision tree algorithm was implemented using the DecisionTreeClassifier provided by sklearn.tree module from sci-kit learn. The parameters tuned were:

A. Max Depth – The maximum depth of a tree controls how deep the tree is allowed to grow. Tuning this parameter in a sense pre prunes the tree by stopping it from growing to its fullest which helps prevent overfitting from an overgrown tree

B. Criterion – The criterion parameter is used to measure the quality of a split at each node and hence is a very important parameter. Two criteria have been considered firstly gini impurity works by checking the degree of impurity in each set of samples and minimizing the impurity when choosing splits. The second criterion is entropy which measures the amount of information or uncertainty in each set of samples. The chosen split is then made by maximizing the information gain leading to purer subsets.

**2. Random Forest**

Random Forest is an ensemble learning method which works by combining a multitude of decision trees for training and hence its name ensemble . Each tree in the random forest gives its own class prediction and the final output is determined by aggregating the votes from all the trees. The class possessing the majority votes is then selected as the final prediction [5] . In order to make classifications the forest was built using the RandomForestClassifier provided by the sklearn.ensemble module from sci-kit learn . The parameters tuned were:

A. n_estimators – This parameter is used to determine the number of decision trees present in the Random Forest. It is important to tune as too high or little number of trees in the forest could lead to over or underfitting.
B. Max Depth – This parameter in Random Forest serves a similar purpose to its use in individual decision trees, as mentioned earlier with the maximum depth now applied to each tree in the forest of trees which adds another layer of control and adaptability to the ensemble.

3. **Adaptive Boosting**

Adaptive boosting is an ensemble learning approach that works by training so-called weak learners on weighted versions of the training data. Higher weights are assigned to misclassified instances from the previous boosting round for each training iteration. The final prediction is then determined by combining the predictions of each weak learner using a weighted majority vote [5]. To make classifications the adaptive boost model was built using the AdaBoostClassifier provided by the sklearn.ensemble module from sci-kit learn.

A. n_estimators: This represents the number of weak learners to be used in the ensemble. A too-high value for this parameter could potentially increase accuracy however this would also increase computational cost , hence this parameter is important to tune to find a good balance between computational cost and model performance as well as reducing overfitting.

B. learning rate: The learning rate parameter controls the contributions of individual weak learners to the final outputted prediction by scaling the weights of said, weak learners . This is important to tune as getting an optimal learning rate tends to reduce overfitting and improve model generalization

## 3.2 Neural Networks

Neural networks are Artificial intelligence-based models consisting of interconnected nodes which are organized into layers. The input layer receives the data, the hidden layer processes the information, and the output layer produces the results [3] in our case the predictions. I will be exploring two types of perceptron's firstly a single layer and then a multi-layer perceptron.

1. **Perceptron**

The perceptron is a single-layer neural network meaning it only has one layer of weights and no hidden layers between its input and output layers. The perceptron works by taking multiple input features with each having an associated weight and computing its sum. This weighted sum of inputs is then sent to the activation function which produces a binary output based on a threshold value set [3] . In order to make classifications the perceptron model was built using the Perceptron classifier provided by the sklearn.linear_model module from sci-kit learn .

**2. Multi-Layer Perceptron**

The multi-layer perceptron builds on the original perceptron model as it has one or more hidden layers

between its input and output layers [3] . This allows the model to handle more complex tasks than the single-layer perceptron as well as nonlinear data. In each layer, there are neurons interconnected where each of these connections has their associated weights adjusted during training so new patterns in the training data can be learned while increasing the model's overall ability to learn abstract features from the input. The activation function used in the hidden and output layers introduces non-linearity which enables the model to approximate functions of high complexity. Predictions are then made using forward propagation and backpropagation to update the weights based on prediction errors, using techniques like gradient descent to minimize the error. To make classifications the multi-layer perceptron was built using the MLPClassifier provided by the sklearn.neural_network module from sci-kit learn.

For both perceptron models the eta and max iteration parameters were tuned . In addition to this for the multi-layer perceptron the hidden layer sizes parameter was tuned. Below we will be explaining each hyper parameter used and reasons behind its selection.

A. eta– The eta also known as learning rate is used to determine the step size for which model weights will be updated during training. To speed up model convergence and optimize model performance this parameter was tuned.
B. Max Iterations - The maximum iterations specify the maximum number of epochs that can be trained during the learning phase hence it's important to find an optimal number for better model convergence. This is also important to tune as too little or too many epochs during training could lead to underfitting or overfitting.
C. Hidden Layers - The hidden layer specifies the number of neurons in each hidden layer. Tuning this potentially influences the model's ability to learn intricate patterns from the data.

### 3.3 Instance Based Learning (KNN)

K-Nearest Neighbor is an instance-based learning approach that is nonparametric in nature. The K value represents the number of nearest neighbors which are to be considered when making classifications. Given a new data point, the algorithm identifies the k-nearest data points in the training set and assigns a class label based on the majority class among these neighbors [4]. To make classifications, the knn algorithm I made used the KneighborsClassifier provided by the sklearn.neighbors module from sci-kit learn. To maximize performance for both datasets 2 key hyperparameters were tuned.

A. K value - The k value as previously mentioned represents the number of neighbors considered for classifications. A too-low k value could result in underfitting, and a too-high k value could result in overfitting hence it is important to tune this parameter to get optimal performance.

B. Metric - The metric parameter specifies what distance metric to be used when finding the k closest data points. The chosen metrics to be tuned are Euclidean distance (the distance between two points in Euclidean space) , Manhattan distance (the sum of horizontal and vertical distances between points on a grid) and Minkowski Distance which is the generalized form of Euclidean and Manhattan distance.

### 3.4 Bayesian Learning

Bayesian learning is a statistical approach based on Bayes' theorem which describes the probability of an event A given prior knowledge of another event B [1] . Hence Bayesian models can predict the probability of a target variable giving a set of features by updating the prior probabilities about a hypothesis or model based on observed data to obtain the posterior probabilities. From the Bayesian learning paradigm for this experiment two Naive Bayesian models were used, naive meaning these models assume independence amongst features.

1. Gaussian Naïve Bayes

Gaussian Naive Bayes(GNB) is a subset of the Naive Bayes classifier which assumes that features follow a normal distribution. When training the mean and variance for each feature are estimated within each class. When predicting classes, the probability of an instance belonging to each class is calculated through Bayes' theorem and the Gaussian probability density function. Finally, the class with the highest probability is then assigned to the input instance[1].

2. Multinomial Naïve Bayes

Multinomial Naive Bayes (MNB) is a subset of the Naive Bayes classifier for discrete features, commonly used in text classification tasks. The probabilities of each feature occurring in each class are calculated based on frequency counts during training. During prediction, these probabilities are then used to compute the likelihood of an instance belonging to each class. The class with the highest likelihood is then assigned to the input instance[1].

For both Naive models it was decided that no hyper parameter tuning would be done . This was due to naive Bayes being a simple algorithm with a limited amount of hyperparameters . To make classifications, the naive models were trained using the GaussianNB and

MultinomialNB classifiers provided by the sklearn.naive_bayes module from sci-kit learn.

## 4. Model evaluation / Experiments

For each experiment the chosen evaluation metric in deciding the best-performing model was the accuracy. The best-performing model from each experiment was then evaluated in hypothesis 5 by checking both their accuracies and Auc score and visualizing through a roc curve. The final best-performing model overall is then selected, and a final evaluation is done to gain possible insights from the model performance by checking recall, precision, f1 score, and visualizing using a confusion matrix. Below I will talk briefly about each of the metrics to be used and the reasons for their selection.

A. Confusion Matrix - The confusion matrix is a visualization table comprising 4 values (True Positives, True Negatives, False Positives, and False Negatives). True positives show the model's ability at classifying correct predictions. True negatives showcase the model's ability at classifying negative predictions. False positives are an indication of how many times the model incorrectly predicted the positive class while false negatives are an indication of how many times the model incorrectly predicted the negative class.
B. Accuracy - The accuracy tells us how well the model performed at classifying both true positives and True Negatives . This is an important metric to choose as a high model accuracy is directly related to high model performance in most cases .
C. Recall - The recall showcases the model's ability to correctly identify all data samples belonging to a class of interest for example in dataset 1 this would mean for cases where people choose to churn what percentage of this was correctly predicted . A good recall rate means the model is effective at capturing positive cases . In dataset 1 for example a low recall could give false information to the company when devising strategies to improve customer retention.
D. Precision - This precision metric tells the proportion of positive predictions that belong to the positive class. This is important to consider as high precision minimizes false negatives.
E. F1 Score - The f1 score works by combining the precision and recall and then taking their harmonic mean hence a high f1 score tends to indicate optimal values for both precision and recall which makes it an important metric to consider in cases such as dataset 1 where there is an imbalance in the dataset.
F. Roc Curve / Auc - The ROC(Receiver Operating Characteristic) showcases the predictive power of a model by visualizing its true positive rate against its false positive rate with curves closer to the top left corner being indicative of better model performance. The AUC (Area Under the Roc Curve) showcases how well the model performed at distinguishing between classes with a high Auc score tending to mean the model generally performed well. For example, in dataset two a high AUC would mean the model performed well in distinguishing between positive and negative sentiments.

### Material & Methods Overview

For both datasets a 70:30 train test split was used . A grid search was then used with k fold cross validation set to 5 for dataset 1 to reduce the bias of results as this dataset was imbalanced . A normal cross validation was done for dataset 2 with cv set to 2 due to this dataset being substantially larger and time consuming to train. For evaluating performance, I used different metrics from the sklearn.metrics module provided by sci-kit learn . To maintain consistency of results the same hyper parameter spaces were used across both datasets.

## 4.1 Experiment 1

### 4.1.1 Null Hypothesis 1

Gaussian Naive Bayes to outperform Multinomial Naive for dataset 1 . Multinomial Naive Bayes to outperform Gaussian Naive for dataset 2 .

### 4.1.2 Material & Methods 1

No hyperparameters were used in this experiment. The data was not scaled as naive models aren't affected by scaling.

### 4.1.3 Results & Discussion 1

*Figure 3*. Box Plot Showing Accuracies of Bayesian Algorithms for Dataset 1 (Left) and Dataset 2 (Right)

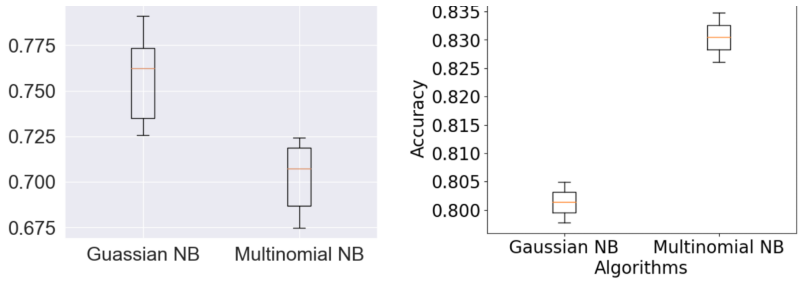



*Table 1*. Predictive accuracies for Bayesian Algorithms for dataset 1 (Telco Churn) and dataset 2 (IMDB Movie Reviews)

| Algorithms | Dataset 1 Accuracy | Dataset 2 Accuracy | Dataset 1 Auc | Dataset 2 Auc |
|---|---|---|---|---|
| GNB | 0.757 | 0.801 | 0.810 | 0.871 |
| MNB | 0.702 | 0.830 | 0.775 | 0.905 |

## 4.2 Experiment 2

### 4.2.1 Null Hypothesis 2

Ensemble learning approach to outperform single tree-based approach across both datasets.

### 4.2.2 Material & Methods 2

No scaling was necessary for this experiment. When selecting parameters to be used in the parameter space for maximum depth I opted to use [None, 5,10, and 20] for the trees to train on both shallow and large tree depths with the same logic applied to the number of estimators parameter space for random forests. In terms of learning rates, I used a diverse parameter space of [0.05, 0.1, 0.5, 1] to fully maximize the Adaboost performance.

### 4.2.3 Results & Discussion 2

*Figure 4*. ROC Curves Showing AUC of Tree vs Ensemble Learning Algorithms for Dataset 1 (Left) and Dataset 2 (Right)

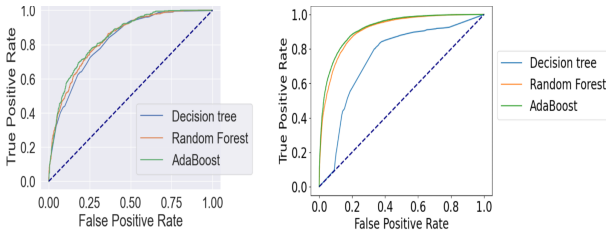



*Table 2*. Predictive accuracies for Tree vs Ensemble Learning Algorithms for dataset 1 (Telco Churn) and dataset 2 (IMDB Movie Reviews)

| Algorithms | Dataset 1 Accuracy | Dataset 2 Accuracy | Dataset 1 Auc | Dataset 2 Auc |
|---|---|---|---|---|
| Decision Tree | 0.782 | 0.722 | 0.820 | 0.746 |
| Random Forest | 0.800 | 0.830 | 0.834 | 0.910 |
| Adaptive Boosting | 0.804 | 0.834 | 0.843 | 0.921 |

## 4.3 Experiment 3

### 4.3.1 Null Hypothesis 3

Optimal k value less than 25 for both datasets

### 4.3.2 Material & Methods 4

The data was scaled as KNN is a distanced-based algorithm. When determining the optimal grid search, I defined a parameter space of 1 to 50 neighbors. To get the optimal k I first used a grid search and then visualized the results using an elbow method plot.

### 4.3.3 Results & Discussion 4

*Figure 5*. Elbow Plot Showing Error rates across different k values for Dataset 1 (Left) and Dataset 2 (Right)

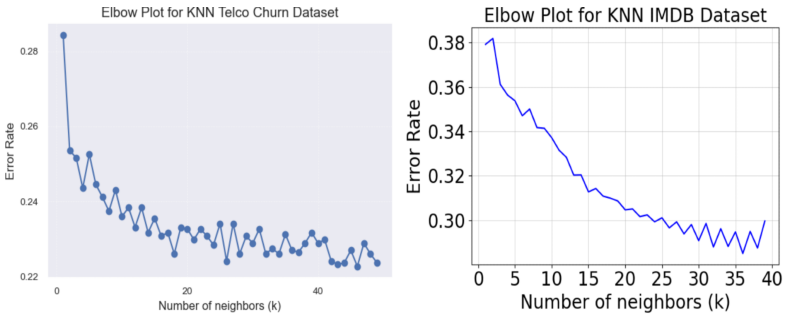



*Table 3*. Predictive accuracies for KNN optimal k value for dataset 1 (Telco Churn) and dataset 2 (IMDB Movie Reviews)

| Algorithms | Dataset 1 Accuracy | Dataset 2 Accuracy | Dataset 1 Auc | Dataset 2 Auc |
|---|---|---|---|---|
| KNN | 0.78 | 0.675 | 0.812 | 0.78 |

## 4.4 Experiment 4

### 4.4.1 Null Hypothesis 4

Multi-Layer Perceptron to outperform single layer perceptron across both datasets.

### 4.4.2 Material & Methods 4

Scaling was done for the experiment for the model to converge faster. When selecting parameters to be used in the parameter space a wide range of values were set for maximum iterations for both MLP and Perceptron with the same logic applied to the hidden layer parameter space for MLP . The solvers used for MLP were Adam , sgd and lgfgs.

### 4.4.3 Results & Discussion 4

*Figure 6*. Box Plot Showing Accuracies of Neural Network Algorithms for Dataset 1 (Left) and Dataset 2 (Right)

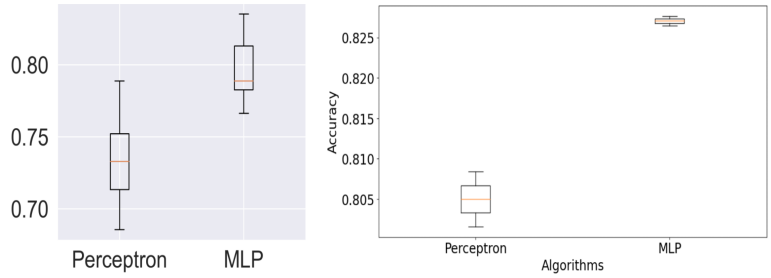


*Table 4*. Predictive accuracies for Neural Network Algorithms for dataset 1 (Telco Churn) and dataset 2 (IMDB Movie Reviews)

| Algorithms | Dataset 1 Accuracy | Dataset 2 Accuracy | Dataset 1 Auc | Dataset 2 Auc |
|---|---|---|---|---|
| Perceptron | 0.734 | 0.805 | 0.797 | 0.914 |
| MLP | 0.797 | 0.827 | 0.823 | 0.902 |

## 4.5 Experiment 5

### 4.5.1 Null Hypothesis 5

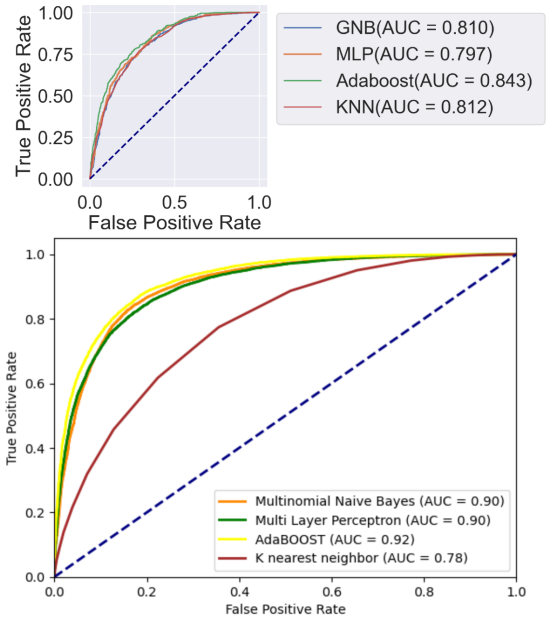
Out of all the best-performing models within each domain, no model will perform significantly better than the other. Significance = +10 difference in chosen evaluation metric

### 4.5.2 Material & Methods 5

After getting each of the best-performing algorithms from previous experiments I decided on using AUC as the chosen metric to check my hypothesis given factors such as the imbalance in dataset 1, the AUC score evaluates the model's ability for ranking examples from both positive and negative classes accordingly regardless of the class distribution.

### 4.5.3 Results & Discussion 5

*Figure 7*. ROC Curves Showing AUC of Best performing algorithms from each experiment . Dataset 1 (Top) and Dataset 2 (Bottom)




## 5. Discussion of the results, interpretation and critical assessment

### Experiment 1

From experiment 1 the best-performing model for dataset 1 was Gaussian naive Bayes and the best-performing model for dataset 2 was multinomial naive Bayes. Hence, we accept the null hypothesis previously stated for this experiment. This was also expected considering Gaussian naive Bayes works best with continuous data such as dataset 1 and multinomial Bayes works better with text data such as dataset 2 [1]. From the table see the gaussian bayes model for dataset 1 have an accuracy of 0.757 compared to the GNB accuracy of 0.801 in dataset 2. This is similar for MNB although the difference between performances across both datasets is quite larger (approx. 13 percent) and further reflected in the respective AUC scores. Although both accuracies would be deemed acceptable this shows the MNB model was able to generalize better to different types of data than the GNB model

### Experiment 2

From table 2 we see the Adaptive Boost was the best-performing model from experiment 2 for both datasets .For dataset 1 the best parameters were a learning rate of 1 and 200 estimators. For dataset 2 the best parameters were a learning rate of 0.5 and 200 estimators. This shows for both datasets a relatively high learning rate and number of estimators give the best performance. Interestingly the model could still perform better had the parameter space been increased further considering 200 is the maximum number of estimators in the parameter space presently. The AdaBoost model performance is followed closely by random forest which both had a huge gap in performance to the worst model decision tree which had accuracy scores in the 70s for both datasets. Hence, we accept the null hypothesis of ensemble techniques given better performance than single trees. Looking at the AUC scores we see the AdaBoost model performed the best in this regard however, it was followed very closely by random forest as seen in table 2 with both model's AUC in the ’90s for dataset 2 and in the ’80s for dataset1 which shows the models ability to generalize to unseen data. A higher performance in ensemble techniques is expected given ensemble techniques generalize better to unseen data given they can combine different weak learners in the case of AdaBoost [2] and different trees in the case of random forest to capture complex patterns and provide more robust predictions[5].

### Experiment 3

After experimenting with different k values across both datasets we see the optimal k values are quite similar (40 nearest neighbors for dataset 1 using Manhattan distance and 30 nearest neighbors using Minkowski distance metric for dataset 2). K values in both datasets are higher than our threshold of 25, hence we can reject the null hypothesis of

k values < 25 for both datasets. The high number of neighbors in dataset 1 could be due to the dataset being imbalanced while in a dataset this could be due to the data being sparse given it has been count vectorized before training [4] . From Table 3 we see the overall accuracies and AUC is higher in dataset 1 compared to dataset 2 which is expected considering the KNN algorithm works best with continuous data [4] and hence the low accuracy of 0.675 on dataset 2 compared to 0.78 in dataset 1. Another point to point from the elbow plot we see the optimal point is at k =40 for dataset 1 however it would be an idea to stop training at k =20 given the difference in accuracy isn't much especially as this model took the longest to train and hence was the most computationally expensive. The same thought process can be applied to dataset 2.

### Experiment 4

From experiment 4 we tested the performance of two neural networks. From Table 4 we see the multilayer performed better than the single-layer perceptron hence the null hypothesis holds. The better performance with MLP is likely due to its hidden which makes it better at handling nonlinear data [3] such as dataset 2 and its efficiency in training through methods like backpropagation. The MLP model performed better on Dataset 2 compared to Dataset 1 although the difference wasn't too drastic. An interesting observation was Dataset 2s AUC value for single-layer perceptron being slightly higher than the MLP even though the MLP had overall higher accuracy. This could potentially be because of the classifiers' capabilities to handle complex data given we are working with sparse text data[6]

### Experiment 5

From experiment 5 we selected each of the best performing models from previous experiments and compared them to each other using AUC as the evaluation metric. We see the best and worst performing models for dataset 1 are Adaboost and MLDM respectively. It would be quite difficult to give a concrete reason for the AdaBoost slightly giving better performance compared to MLP for both datasets given this this could be down to things such as random states or parameter spaces that were selected for the grid. From dataset 2 we see the best and worst performing models for dataset 2 are Adaboost and KNN respectively. Understandably Adaboost given its ensemble learning nature previously discussed would drastically outperform the KNN model which isn't suitable for text data. The difference in AUC between Adaboost and KNN is a huge 14 percent. This difference means we can reject the null hypothesis that no best-performing model will perform significantly better than the other across both datasets. It's also worth noting although the Adaboost had the best performance it also took a considerably long time to train compared to the MNB model .

## 6. Conclusions

The best-performing model across both datasets interestingly was the adaptive boosting model. This shows that AdaBoost is good at generalizing to different types of data due to its ensemble learning approach of combining weak learners. I went on to do a classification report and confusion matrix for both datasets using the AdaBoost model. For dataset 1 the precision-recall and f1 scores were all in the high 80s to 90s for class 0 however for class 1 this dropped to the 60s and was as low as a 55 percent precision for class 1 and hence giving this class a high number of false negatives. This shows despite using stratified K fold the models still struggled to handle the imbalance in dataset 1. This was further backed up by the performance in Dataset 2 which was almost even in every category across both classes with precision-recall and f1 scores all in the 80s. In general, models performed better on dataset 2 compared to dataset 1 . Upon further investigation I found that dataset 1 is made up of fictional data which is a possible explanation for the difference in performance. In conclusion, the Adaboost model did perform best overall however in a real-world setting with larger amounts of data where computation cost and training time are bigger factors other models such as the naive models would be a better overall choice for this particular problem .

## References


[1] VanderPlas, J. (n.d.). In Depth: Naive Bayes Classification. Python Data Science Handbook.

https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html

[2] Nádia Silva, Estevam Hruschka, and Eduardo Hruschka. 2014. Biocom Usp: Tweet Sentiment Analysis with Adaptive Boosting Ensemble. In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 129–134, Dublin, Ireland. Association for Computational Linguistics.

[3] Zhou, H. (n.d.). Neural Networks (Part I) http://hua-zhou.github.io/teaching/biostatm280-2019winter/slides/15-nn/nn1.html

[4] Analytics Vidhya. (2018, March 26). K-Nearest Neighbors Algorithm in Python and Scikit-Learn. https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

[5] Analytics Vidhya. (2021, April). Distinguish Between Tree-Based Machine Learning Algorithms. Retrieved https://www.analyticsvidhya.com/blog/2021/04/distinguish-between-tree-based-machine-learning-algorithms/

[6] Michael Collins and Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 111–118, Barcelona, Spain.