Brain Stroke Prediction

Saad Patel
Department of Computer Science
University of Western Ontario
London, Canada
spate696@uwo.ca

Sahil Rehman
Department of Computer Science
University of Western Ontario
London, Canada
srehma9@uwo.ca

Venkata Harshavardhan Arepalli
Department of Computer Science
University of Western Ontario
London, Canada
varepall@uwo.ca

Rutvik Patel
Department of Computer Science
University of Western Ontario
London, Canada
rpate492@uwo.ca

Abstract

Every year many deaths are caused due to brain stroke. Early prediction of whether a patient would get a stroke or not can be very useful in taking early precautions and help save many lives. The main objective of our project is to develop ML models that will help in predicting if the patient is likely to have a brain stroke or not based on features related to the patient. The dataset that is obtained from Kaggle was split into the training set, validation set and test set. We used five supervised learning algorithms - Logistic regression, Random Forest, Neural Networks (Multi-Layer Perceptron), Adaboost and Naive Bayes, to train our models. Then we assessed their performance on the validation set to figure out which of them was performing better and we used our top performers on the test dataset. But the significant challenge here is imbalanced data where the number of samples that are non-stroke is very high in the dataset. To overcome this problem, we did oversampling with a technique called smote.
Keywords: Stroke, ML algorithms, Logistic regression, Random Forest, Neural Networks (Multi-Layer Perceptron), Adaboost and Naive Bayes.

Introduction

According to the World Health Organization, brain stroke is the second most common cause of death in the world. About 11% of all deaths come under brain stroke every year. A stroke is a medical emergency that needs immediate medical attention. It is a condition that occurs in the brain when the cells in the brain do not get the required nutrients. [1] This can happen in two ways and hence we have two kinds of strokes. The first one is an ischemic stroke - in this case, blood clots in the veins that are connected in the brain obstruct the proper flow of blood which leads to oxygen deprivation to the cells in the brain. The second type of stroke is hemorrhagic - in this case, the blood from an artery leaks into the brain after exploding as the vessels are extremely weak.

If we can predict or estimate if a patient is likely to have a stroke based on various medical attributes of that patient it could be very helpful in preventing permanent damage and in many cases death. Also, the patient can potentially adopt a better lifestyle knowing they are at risk of having a stroke. This can be done with the help of machine learning algorithms that we will be applying to the patients' medical records. These algorithms will find the pattern among various attributes such as hypertension, body mass index level, heart disease, average glucose level, smoking status, previous stroke, age, etc, and the likelihood of getting a stroke.

There are many previous works and many experiments conducted by researchers in the domain of predicting strokes. Amini et al [2] in their research collected records from 807 patients that are

both healthy and unhealthy and came up with 50 risk factors associated with stroke. Out of all the techniques they used, the decision tree algorithm and K-nearest neighbor algorithms yielded the best results which are 95% and 94% accuracy respectively. Singh et al. [3] conducted their experiments on data related to cardiovascular stroke. They used the decision tree algorithm to extract features and used it for Principal component analysis. However, their best performing algorithm was the neural network classifier which gave them an accuracy of 97%. Adam et al. [4] performed various machine learning algorithms on the ischemic stroke dataset to classify them. They found that k-nearest neighbor and decision tree algorithms were performing well. However, they recommended that the decision tree algorithm would be better suited for the specialist to assist them in classifying the strokes.

Kansadub et al. [5] used algorithms like Naive Bayes, neural network and decision tree algorithms on the stroke prediction dataset. They used accuracy and area under curve (AUC) as their metric to evaluate which of them is working best. They came up with naive bayes as the best-performing model. Cheon et al. [6] used 15099 patient records to identify the occurrence of stroke. They used principal component analysis and neural network algorithms to predict if a patient is likely to get a stroke or not. They concluded that PCA was best performing with an area under curve value of 83%. Sung et al. [7] collected the data of 3,577 patients with acute ischemic stroke. With this, their goal was to create a stroke severity index. To predict they used many data mining techniques and linear models. Their best-performing model was the KNN algorithm with an accuracy of 95%.

Methodology

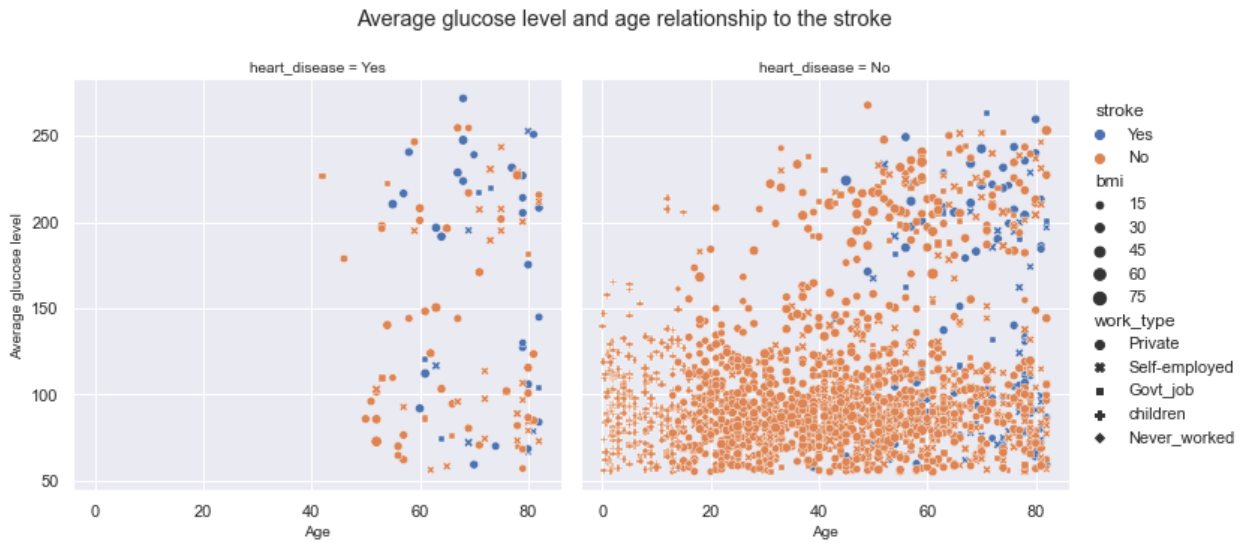
Data

The dataset we used in this paper is obtained from Kaggle as it is hosted there. The data is collected from a medical clinic in Bangladesh. This dataset contains 5110 observations where each row in the data provided relevant information about the patient. The dataset has 12 attributes related to each patient and a target column that tells whether the sample has a stroke or not. The 12 attributes are as follows -

id: This attribute is a unique identifier and comprises numerical data.
age: This attribute tells a person's age and comprises numerical data.
gender: This attribute gives a person's gender. and comprises categorical data.
hypertension: This attribute means that this person is hypertensive or not and comprises numerical data.
work type: This attribute gives info about the kind of work done by the patient and comprises categorical data.
residence type: This attribute tells what residence type the patient is living in and comprises categorical data.
heart disease: This attribute means whether this person has heart disease or not and comprises numerical data.
avg glucose level: This attribute means what was the level of a person's glucose condition and comprises numerical data.
bmi: This attribute means the body mass index of a person and comprises numerical data.

ever married: This attribute represents a person's married status and comprises categorical data.
smoking Status: This attribute means a person's smoking condition and comprises categorical data. The target column is
stroke: This attribute means a person previously had a stroke or not and comprises numerical data.

Data Visualisation



The following figure gives us a brief overview of the first 2000 samples of the dataset. This plot compares data based on the average glucose level and age relationship to stroke while categorizing them into two; heart disease yes/no. It can be noted that the majority of the patients that were diagnosed with a stroke were in the age range of 40 to 80, i.e in both cases of having heart disease and not having a heart disease. Whilst it can also be pointed out that, in comparison, the stroke patients had a relatively high BMI. Additionally, the glucose level ranges at all levels for both categories of stroke patients. There also seems to be a class imbalance as only a few samples are found to have a stroke.

Data Preprocessing

The data preprocessing and analyses were done in python. As mentioned earlier, the dataset contains a few categorical data which had to be converted into numerical data using pandas data manipulator "get_dummies", where all the categorical data were converted into indicator variables (binary vector representation). This is essential before feeding the data into our machine learning models. This lets the machine learning algorithms leverage all of the information from the categorical features without the confusion caused by ordinality. The features that were manipulated using were; gender, work_type and smoking_status.

This dataset was fairly clean with minimal null values, except for BMI which contained 201 null values. Missing data values were filled with median values using pandas fillna(df.median()).

StandardScaler from sklearn was used to standardize the features by subtracting the mean and then scaling to unit variance. Most of the algorithms used work better and converge faster when scaled similarly.

SMOTE Technique was used due to class imbalance. The dataset needed upsampling and this was done using SMOTE as this technique synthesizes new examples for the minority class. Since the outcome class was imbalanced, it was necessary for the model to effectively learn the decision boundary. SMOTE creates duplicating samples of the minority class to synthesize new samples of the minority class. This can directly affect the overall performance of the models.

The dataset was split into a training; 66% for modeling, validation set; 14% to compare performance and a test set of 20% for testing.

Algorithms used

A. Logistic Regression

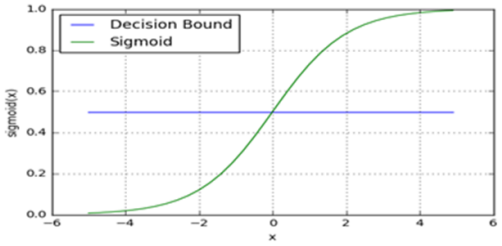
Logistic regression is a supervised classification algorithm. It works efficiently for binary classification problems. It is used when the dependent variable(target value) is categorical and it's easier to implement and interpret. It makes use of the Sigmoid Function and makes predictions using a learned logistic regression model. To have a probabilistic view of the classification, conditional probability is used which is given as:

$$p(y_i|x_i; w) = \begin{cases} \frac{1}{1+e^{-w^\top x_i}}, & \text{if } y_i = 1 \\ 1 - \frac{1}{1+e^{-w^\top x_i}}, & \text{if } y_i = 0 \end{cases}$$

where xi is the number of features (feature dimension). The sigmoid function is used to model conditional probability. The Sum of squared error can be used as an error function but it's not the best choice to opt for finding the error. So, it's better to use the log-likelihood function given as below:

$$\log L(w) = \sum_{i=1}^{m} (y_i \log t_i + (1 - y_i) \log(1 - t_i)), \text{ where } t_i = \sigma(h_w(x_i)).$$

Logistic regression uses the sigmoid function which is represented as shown in the graph:



B. Random Forest

At its core, random forest is composed of decision trees and the final result is an ensemble of these decision trees. The trees in the forest are generated by a technique known as bagging. From the dataset, random sets of data are taken and a tree is created out of it. This process of creating trees is done based on the total number of trees we want/specify in our random forest algorithm. It deals well with the problem of overfitting when compared to a decision tree algorithm. The reason is a single tree starts to develop rules for the minority class and in the process, overfitting occurs because the tree tries to create specific rules for the minority class (these rules are not generalized). However, in a random forest algorithm, this problem is avoided because of the vote that every tree gets which in turn would generalize the final result. In every tree that is to be created, splitting is one of the crucial steps. For that split to be of good quality there are two methods that are widely used which are Entropy and Gini impurity. Entropy is nothing but a measure of how pure the split is going to be. It is given by the formula:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

pi is the probability/ frequency of class 'i'. If there were two classes positive and negative then c would be 2 and we would have a summation of the two terms. At each stage, entropy is calculated by looking at the subset of the data and its positive and negative classes and entropy is calculated based on them. Gini impurity is similar to entropy and it measures the impurity of the split and the best value of Gini impurity is 0 which means that the split is perfect. The formula of Gini impurity. Here p is the probability of each class at a split.

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2$$

When it comes to computing power, the Gini impurity method is more efficient than entropy. Therefore in our implementation of the random forest, we are using Gini impurity.

C. Naive Bayes

Naive Bayes is a probabilistic machine learning algorithm used for both classifications and for regression. The fundamental assumption of Na¨ıve Bayes is that each feature makes an independent and equal contribution to the outcome. Bayes Theorem finds the probability of an event occurring given that probability of another event has already occurred. Bayes theorem was mathematically given by:

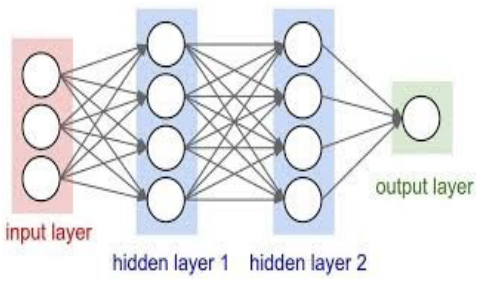
$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

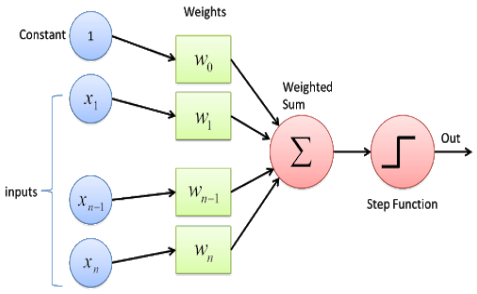
There are three variants of na¨ıve bayes; Gaussian, Multinomial and Bernoulli. Gaussian Naive Bayes assumes that the features follow the normal distribution. Multinomial is used when there is a multinomial distribution and is used mainly for text documents. Bernoulli is useful when the feature vectors are binary. In this paper, we have used Gaussian Na¨ıve Bayes for experiments. The mathematical representation of Gaussian Naive Bayes is

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

D. MLP(Neural Network)

A neural network is a collection of nodes or so-called neural units connected to each other. Artificial neural networks were created to mimic the behavior of the human brain/central nervous system. It was later discovered that the human neuron works very differently and the operation is very complex and cannot be replaced by a simple mathematical calculation. In general, an ANN consists of 2 or more layers. Namely, the input layer, hidden layer(s) or the output layer. The input layer receives input data and the output layer is used to provide outputs to the user and is partly responsible for learning features. The hidden layer contributes the most to the performance of the model and is mainly responsible for learning features. Each layer has a bias and an activation function like ReLu, Sigmoid, Softmax, etc.





The given equation gives the output y for each layer for input x from the previous layer and weights w for that particular layer, and b is the bias for that particular neural unit.

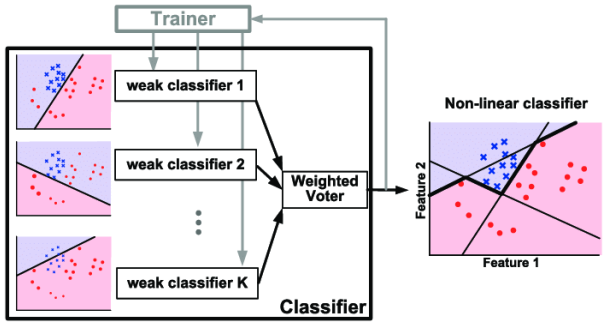
$$y = \varphi(\sum_{i=1}^{n} w_i x_i + b) = \varphi(\mathbf{w}^T \mathbf{x} + b)$$

E. AdaBoost Classifier

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

Adaboost involves the following steps:

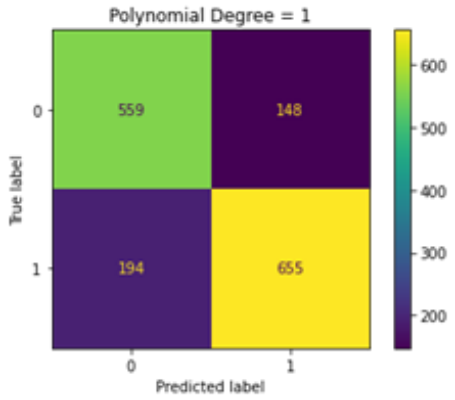
1) Initially, all samples have the same weight adding to 1.
2) For the first stump, determine the importance of features and set the most important feature at each split → the weight of this stump in the final prediction is based on its total prediction error. This comes from adding the incorrect sample weights.
3) The sample weight for any previous incorrect sample(s) increases; all others decrease.
4) Re-sample the re-weighted data (with replacement). This new data is used for the next stump.



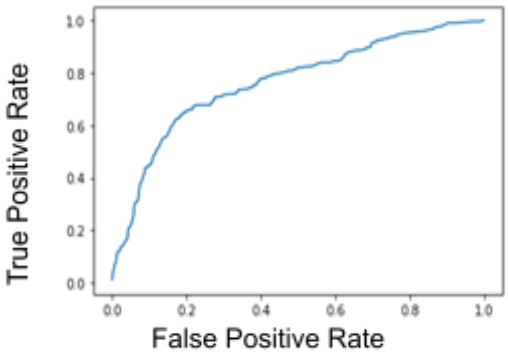
Results

A. Logistic Regression

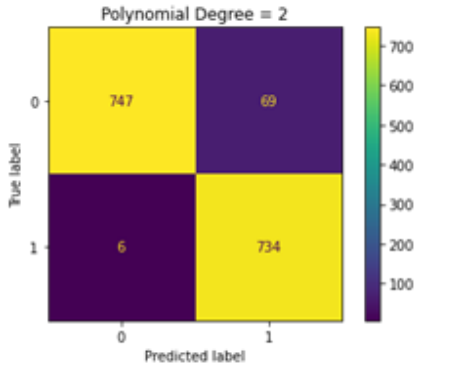
Confusion Matrix



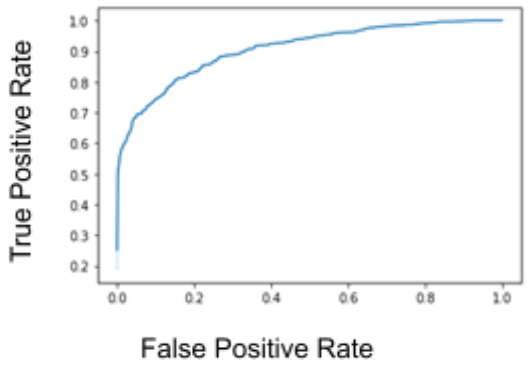
ROC Curve



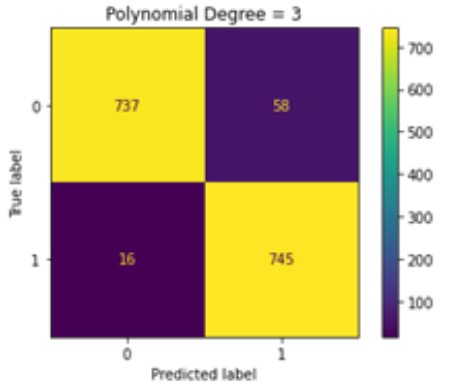
Confusion Matrix



ROC Curve



Confusion Matrix



ROC Curve

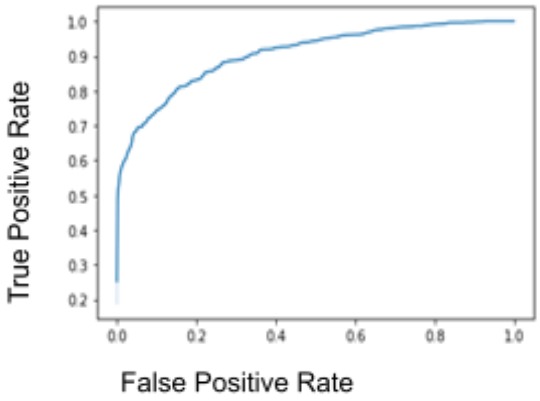
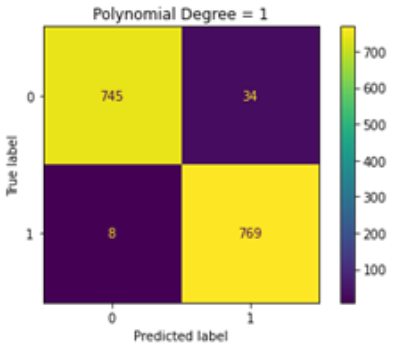


Table 1:

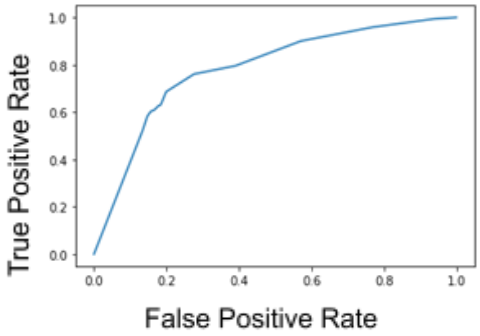
	Degree 1	Degree 2	Degree 3
Accuracy	0.7802056555269923	0.9517994858611826	0.9524421593830334
AUC	0.7636684478358877	0.9070740698476333	0.8751974253256795

B. Random Forest

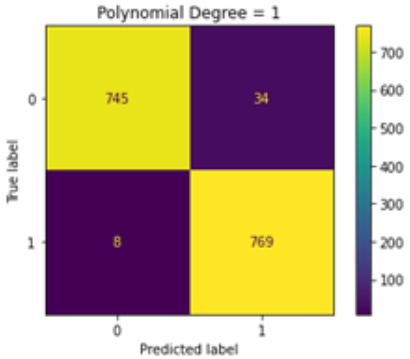
Confusion Matrix



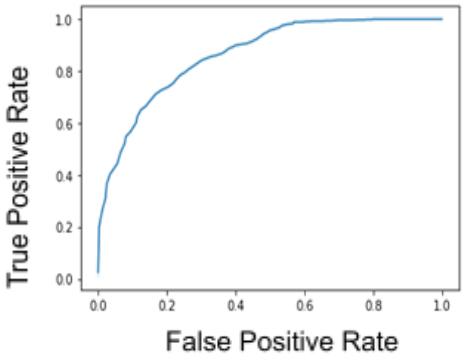
ROC Curve



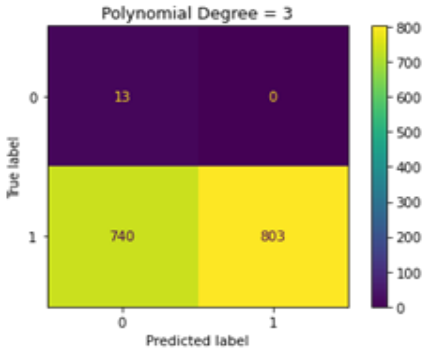
Confusion Matrix



ROC Curve



Confusion Matrix



ROC Curve

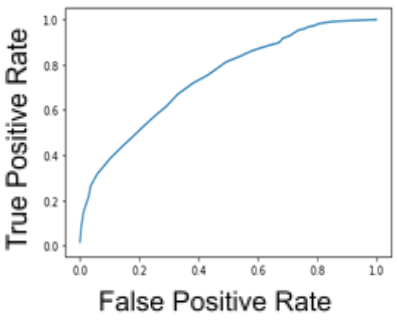
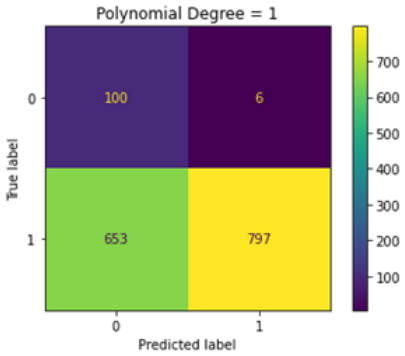


Table 2:

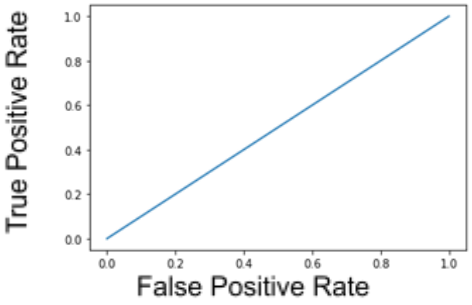
	Degree 1	Degree 2	Degree 3
Accuracy	0.9730077120822622	0.6844473007712082	0.5244215938303342
AUC	0.7832894242870776	0.8645302558962986	0.7423688392961983

C. Naive Bayes

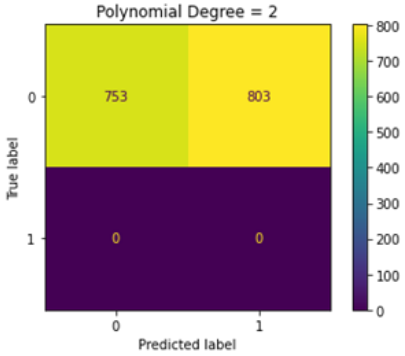
Confusion Matrix



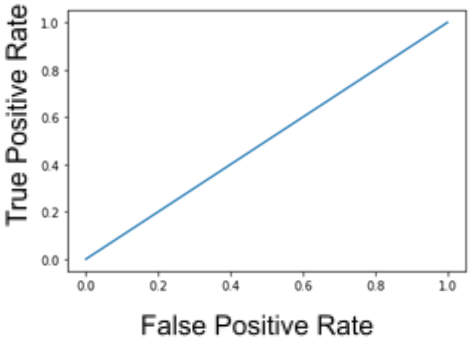
ROC Curve



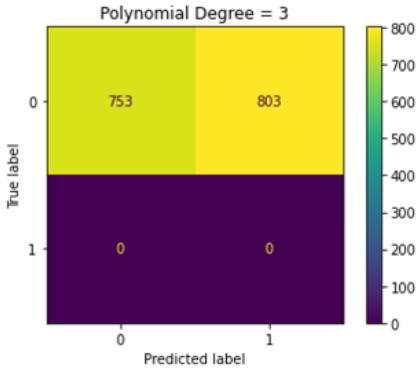
Confusion Matrix



ROC Curve



Confusion Matrix



ROC Curve

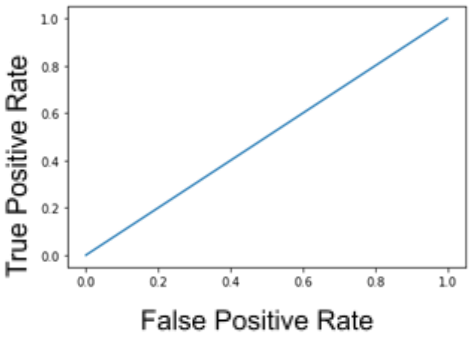
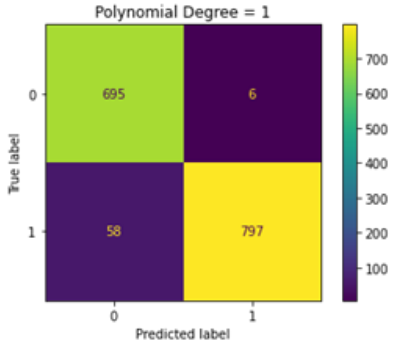


Table 3:

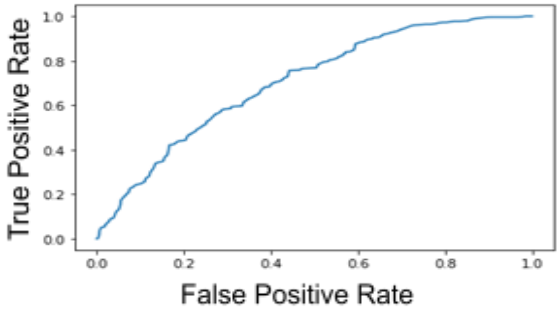
	Degree 1	Degree 2	Degree 3
Accuracy	0.5794344473007712	0.5002570694087404	0.5002570694087404
AUC	0.5	0.5	0.5

D. MLP(Neural Network)

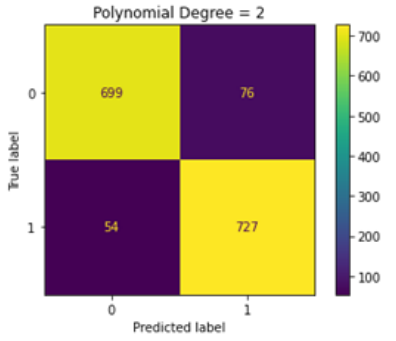
Confusion Matrix



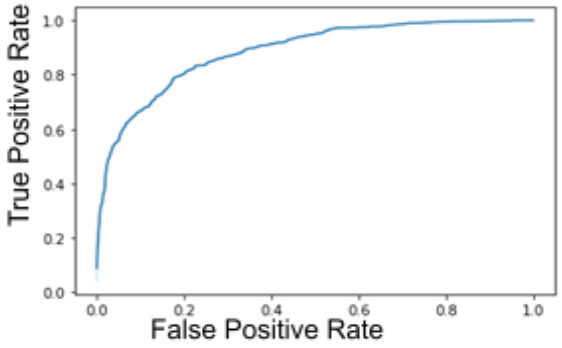
ROC Curve



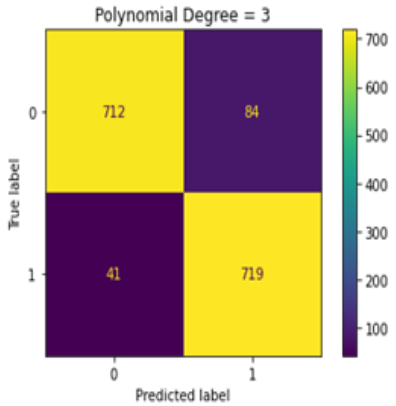
Confusion Matrix



ROC Curve



Confusion Matrix



ROC Curve

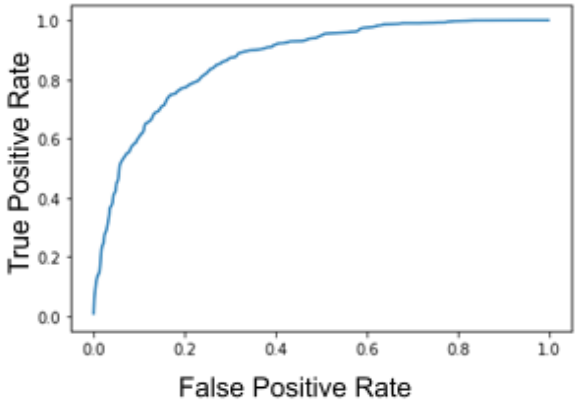
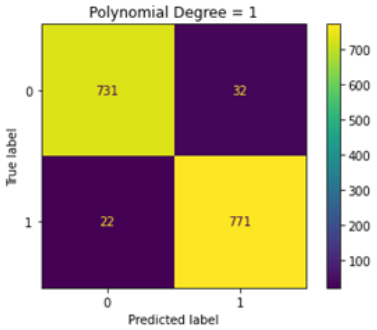


Table 4:

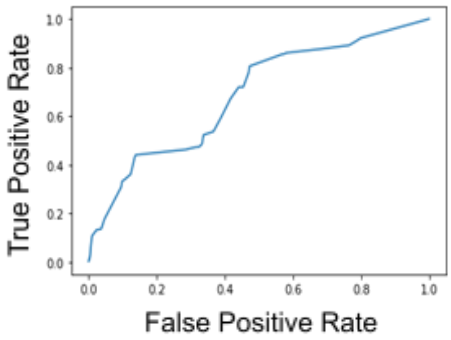
	Degree 1	Degree 2	Degree 3
Accuracy	0.9588688946015425	0.916452442159383	0.9196658097686375
AUC	0.7055629702030402	0.8866890263768505	0.8699829159906658

E. AdaBoost Classifier

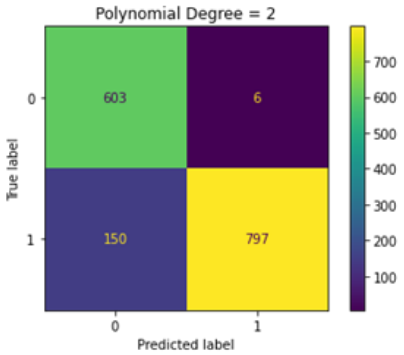
Confusion Matrix



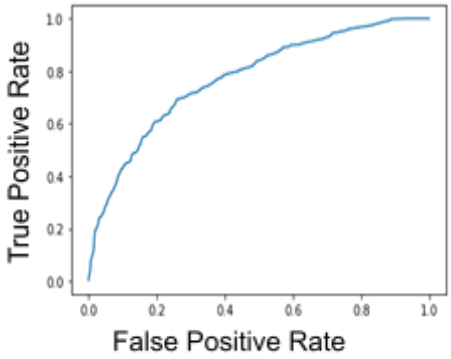
ROC Curve



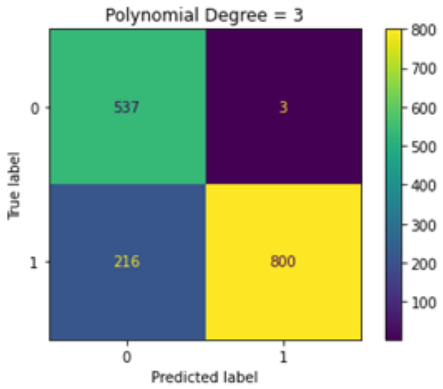
Confusion Matrix



ROC Curve



Confusion Matrix



ROC Curve

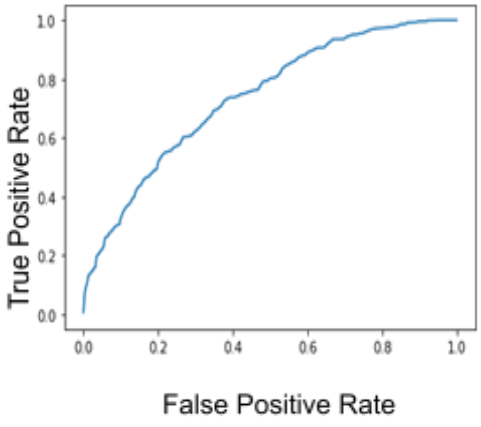


Table 5:

	Degree 1	Degree 2	Degree 3
Accuracy	0.9652956298200515	0.8997429305912596	0.859254498714653
AUC	0.6854425386870946	0.7727702721699338	0.7372295789858416

Considering the models with a Validation Accuracy of more than 95%, here are the shortlisted models:

Model Name	Polynomial Degree	Validation Accuracy	AUC
Random Forest	1	0.9730	0.7833
AdaBoost	1	0.9653	0.6854
Neural Network	1	0.9588	0.7055
Logistic Regression	3	0.9524	0.8752
Logistic Regression	2	0.9518	0.9070

Considering the top 3 models based on accuracy, we tested them on our test data, and here are the test accuracies.

Model Name	Polynomial Degree	Test Accuracy
Random Forest	1	0.9676
AdaBoost	1	0.9681
Neural Network	1	0.9465

Conclusion and Future Work

Looking at the test accuracy, we can see that all the selected models give similar accuracy and without any polynomial features. This concludes that polynomial features did not have any significant effect on this particular dataset.

We can try different machine learning techniques like Gradient boosting(XGBoost), KNN, Voting Classifier, and Support Vector Machines. We can also try fine-tuning hyperparameters for the algorithms used in this project. In the future, we would like to use different datasets which are similar to real-time scenarios.

Acknowledgment

We would like to take this opportunity to express our sincere gratitude and deep regards to Professor Jacob Morra for his guidance, monitoring, and constant encouragement throughout the course which helped us in doing research about machine learning techniques. We are really thankful to him.

References

[1] M. U. Emon, M. S. Keya, T. I. Meghla, M. M. Rahman, M. S. A. Mamun and M. S. Kaiser, "Performance Analysis of Machine Learning Approaches in Stroke Prediction," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2020.

[2] L. Amini, R. Azarpazhouh, M. T. Farzadfar, S. A. Mousavi, F. Jazaieri, F. Khorvash, R. Norouzi, and N. Toghianfar, “Prediction and control of stroke by data mining,” International Journal of Preventive Medicine, vol. 4, no. Suppl 2, pp. S245–249, May 2013.

[3] M. S. Singh and P. Choudhary, “Stroke prediction using artificial intelligence,” in 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), Aug. 2017, pp. 158–161.

[4] S. Y. Adam, A. Yousif, and M. B. Bashir, “Classification of Ischemic Stroke using Machine Learning Algorithms,” International Journal of Computer Applications, vol. 149, no. 10, pp. 26–31, Sep. 2016.

[5] T. Kansadub, S. Thammaboosadee, S. Kiattisin, and C. Jalayondeja, “Stroke risk prediction model based on demographic data,” in 2015 8th Biomedical Engineering International Conference (BMEiCON), Nov. 2015, pp. 1–3.

[6] S. Cheon, J. Kim, and J. Lim, “The Use of Deep Learning to Predict Stroke Patient Mortality,” International Journal of Environmental Research and Public Health, vol. 16, no. 11, 2019.

[7] S.-F. Sung, C.-Y. Hsieh, Y.-H. Kao Yang, H.-J. Lin, C.-H. Chen, Y.- W. Chen, and Y.-H. Hu, “Developing a stroke severity index based on administrative data was feasible using data mining techniques,” Journal of Clinical Epidemiology, vol. 68, no. 11, pp. 1292–1300, Nov. 2015.