CSE422 Project Report

Project Topic: Brain Stroke Detection

Section: 06

Group Number: 7

Group Members:

1. **Name: Anika Ahmed, ID: 21101029**
2. **Name: Moinul Haque, ID: 21101186**
3. **Name: Nafis Chowdhury, ID: 21101034**
4. **Name: Mohammed Adib Ahnaf Hamim, ID: 21101054**

Table of Contents

1. **Introduction:**

   Brain stroke is a very common health issue that many people face worldwide. Our project takes in a person's data as input and analyzes them to give an output that tells us whether the person has a risk of getting a stroke. In this way, people will be able to know and ask for help early if they are at a risk of having a brain stroke.

2. **Dataset Description:**

   - **Source: Kaggle**

     Link: Brain Stroke Dataset

   - **Dataset Description**

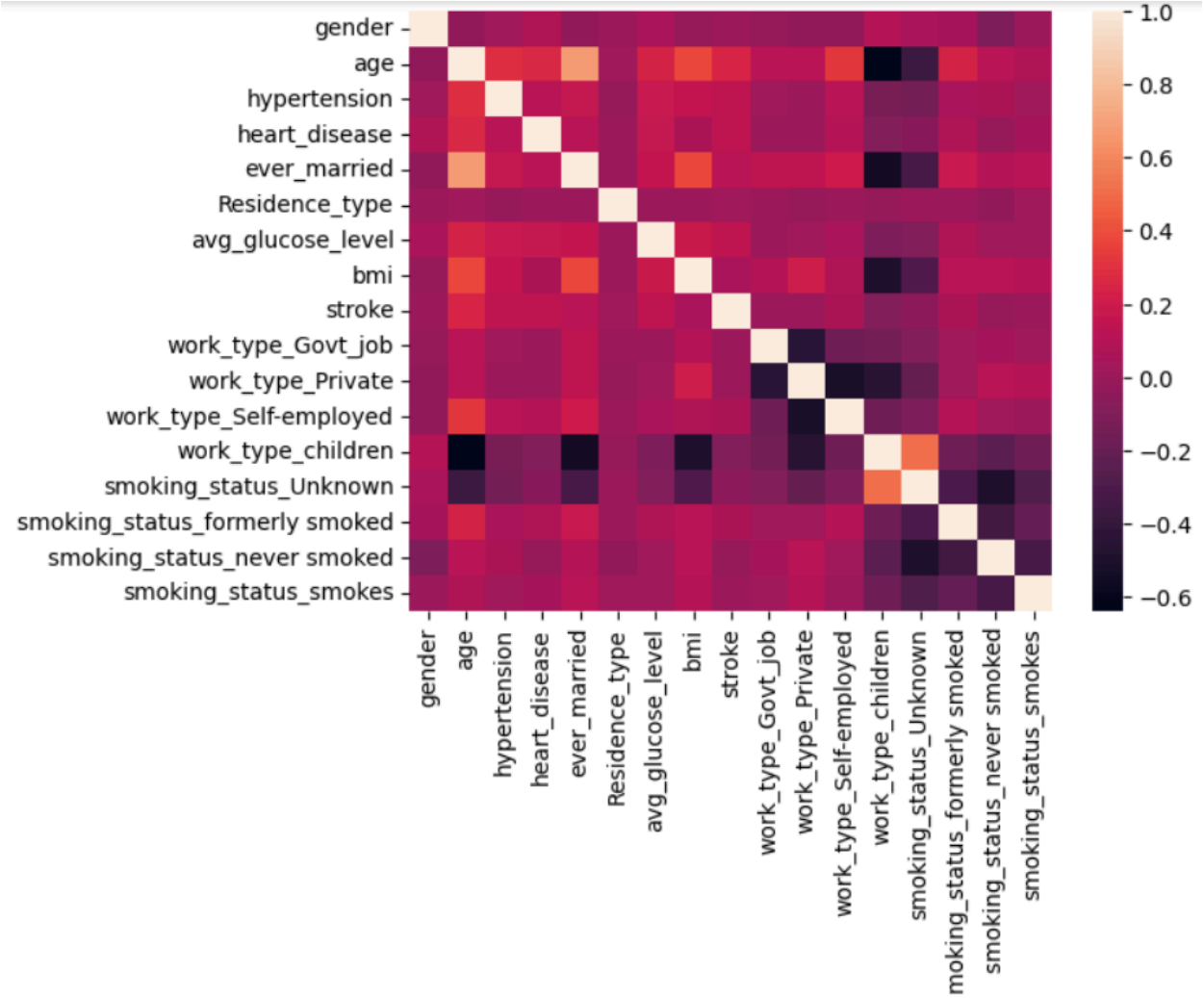
     There are 11 features in our dataset.

     It is a classification problem, because there are only two possible outcomes: the person has a risk of having a brain stroke or not. No other outcome is possible.

     Initially, there were 4981 data points in the dataset. However, after removing the rows containing null values, we are left with 4976 data points to work with.

     There are both quantitative and categorical features in our dataset. Some examples of quantitative features are 'average glucose level' and 'bmi', whereas 'work type' and 'residence type' are categorical features.

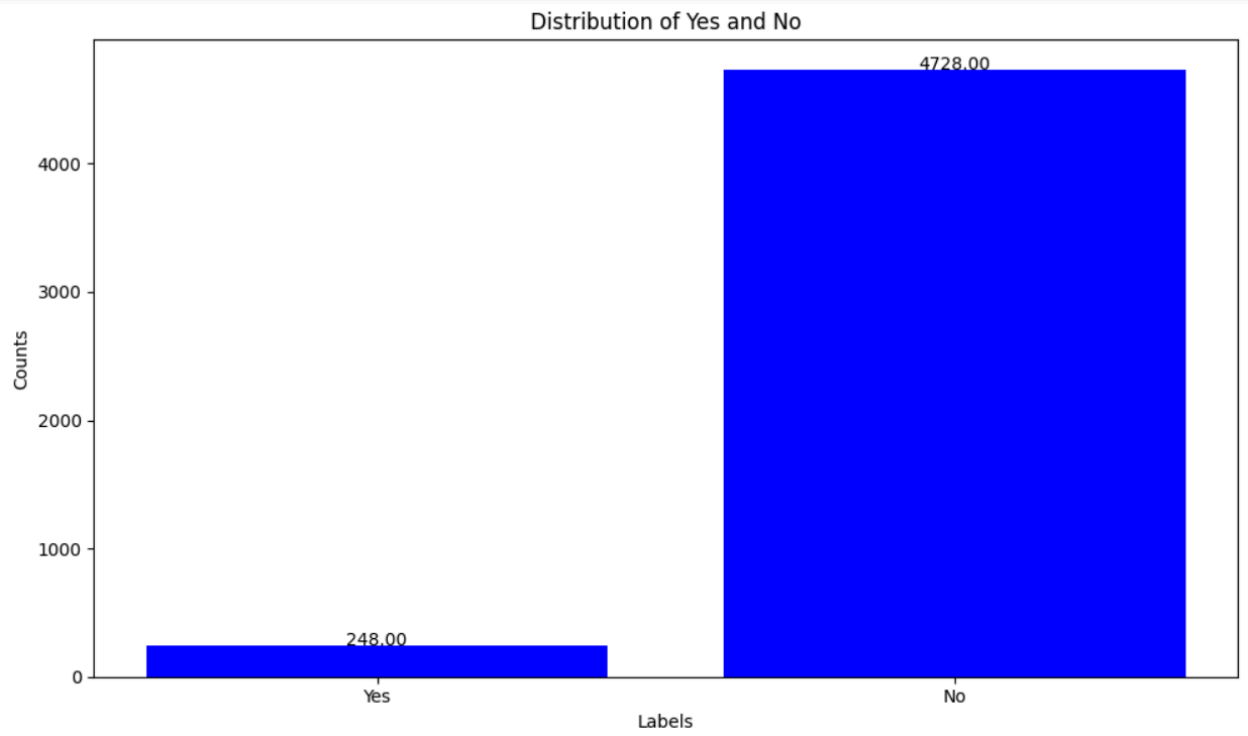
Correlation (Using Heatmap):



From the heatmap above, we can get an idea about the correlation between our features. We can see that work_type_children and smoking_status_unknown hold similar data. Also, the data for the people who never smoked and those who smoke are similar.

- **Imbalanced Dataset**

  We can get information about our dataset by plotting a bar chart containing the numbers of data points present for each type of output that our dataset contains.



From the bar chart above, we can see that our dataset is highly imbalanced. The number of data points that give ‘No’ as an output is much higher than those that give ‘Yes’ as an output. In order to resolve this problem, we have to use the stratify method while splitting the dataset for training and testing.

3. **Dataset pre-processing:**

- **Faults and Solutions:**

  - NULL Values:

    There were no null values in the original dataset. However, we put in some null values in order to show the pre-processing of null values.
    Null values need to be removed from the dataset before calculation because these cause errors in the model's learning process.
    So, we removed the rows that contained null values in any of the columns.

  - Categorical values:

    The categorical values in our dataset are represented by strings, which cannot be a part of the calculation process. We used **label encoders** and **one hot encoders** to encode these values.

    We used label encoders when the categorical variable had only two possible values. However, if the categorical variable had more than two possible values, we used one hot encoding. This is to ensure that all the categorical values are given equal importance in our calculation, which will in turn affect the accuracy of our models.

For example, for the ‘ever married’ feature, there are only two possible values: ‘Yes’ and ‘No’. These can be represented by 1 and 0 respectively, using a label encoder.

However, the ‘work type’ feature has 4 unique values, which are ‘Private’, ‘Self-employed’, ‘Govt-job’ and ‘Children’. If a label encoder was used here, these values would take the numbers 0, 1, 2 and 3, which would cause the value encoded as 3 to get a higher priority in calculations, compared to the other possible values. In order to avoid such bias, one hot encoding was used.

4. Feature Scaling:

Some features in our data, for example ‘age’, ‘average glucose level’ and ‘bmi’ had much larger numbers compared to the features on which we used label encoder or one hot encoding. To remove any kind of bias due to the presence of larger numbers in some features, feature scaling was used.

5. Dataset Splitting:

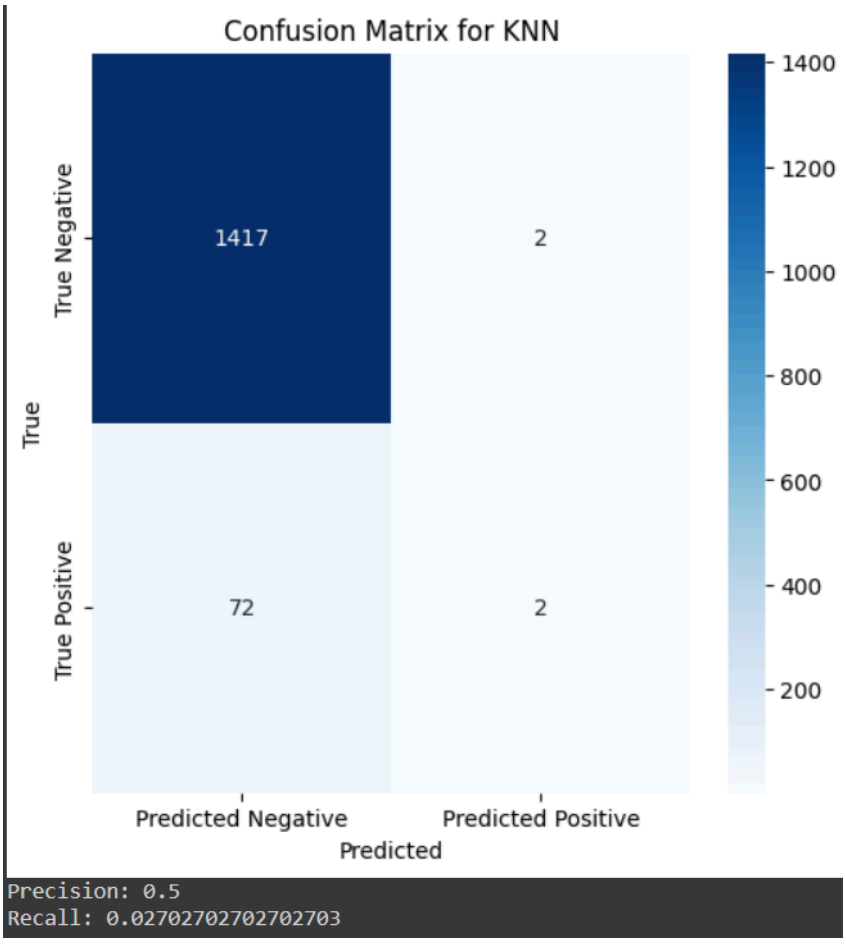
From the bar chart above, we can see that in our dataset, the number of data for ‘No’ output is much higher than the number of data for ‘Yes’ output. This might create a bias in our models. In order to avoid this bias, we used the stratified method to split our data.

Our dataset was split by keeping 70% of our data as our training data, and the remaining 30% was used as our testing data.

6. Model training and testing:

- **KNN**

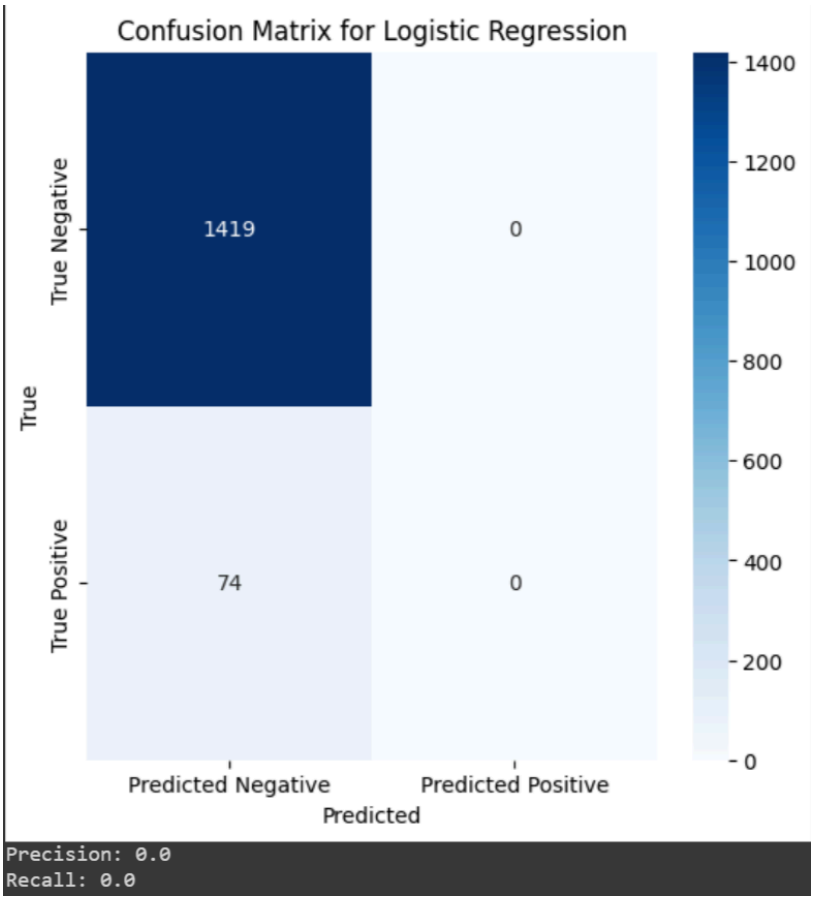
  K - Nearest Neighbours (KNN) model works by initially plotting all the training data, and then plotting the testing data in the same place. It measures the distance between the previously learned data points and the new data received, and gives the output the K nearest neighbors suggest. We set the value of k as 5, and got an accuracy of 95%.



As our dataset was highly imbalanced, and the data points that gave ‘Yes’ as an output were very less, our model does not work well for positive outcomes.

- **Logistic Regression**

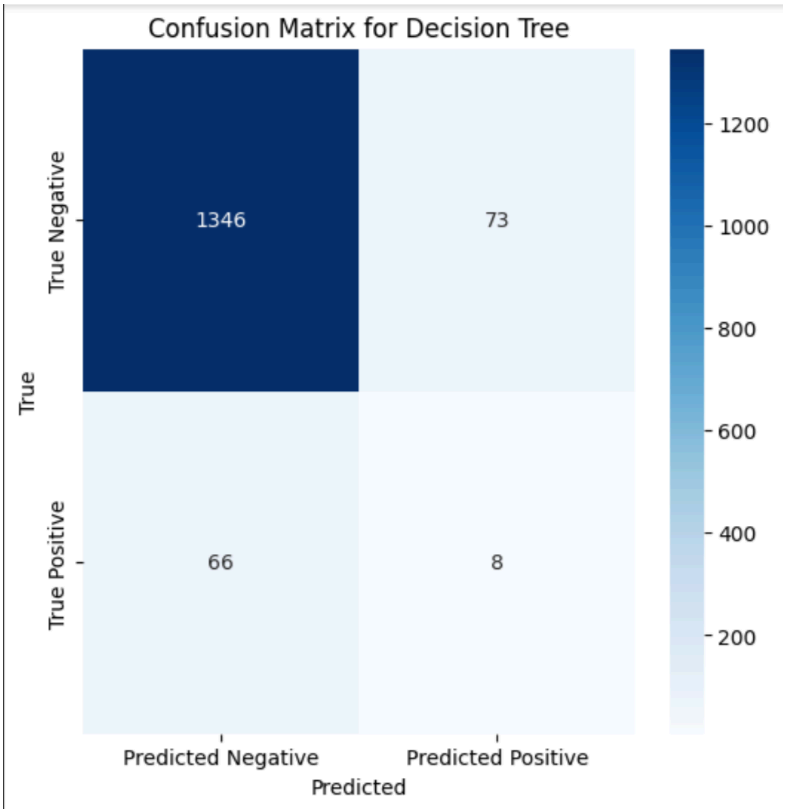
  Logistic regression predicts the probability of an instance of belonging to a given class or not. It uses a sigmoid function to estimate the probability for the given class. It maps training data within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the “S” form. In terms of testing, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tend to be 1, and a value below the threshold value tends to be 0. This model of ours gave an accuracy of 95.04%.



As the points plotted are very concentrated around ‘0’ output, our model does not work at all for ‘1’ outputs. Precision and recall of this model are 0, which means that for our dataset, Logistic Regression model’s predictions are irrelevant.

- **Decision Tree**

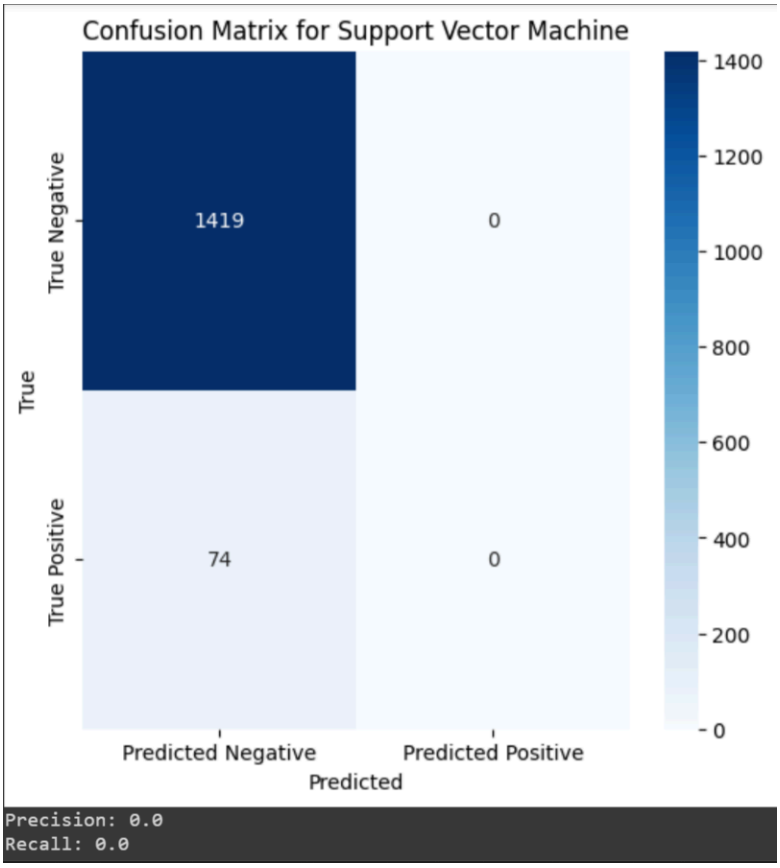
  The decision tree selects the best features to split the data based on certain criteria (like information gain for classification tasks) to create branches in the tree. It recursively repeats this process, creating more branches until all the features are assigned labels. Each leaf node in the tree is assigned a class label for classification. During testing, new, unseen data is passed down the decision tree. The test data follows the appropriate branch based on the comparison moving to the next node. This process continues until the test data reaches the leaf node and then assigns it a label. This model here, gives an accuracy of 90.69%.



As our dataset was highly imbalanced, and the data points that gave ‘Yes’ as an output were very less, our model does not work well for positive outcomes.

- **Support Vector Machine**

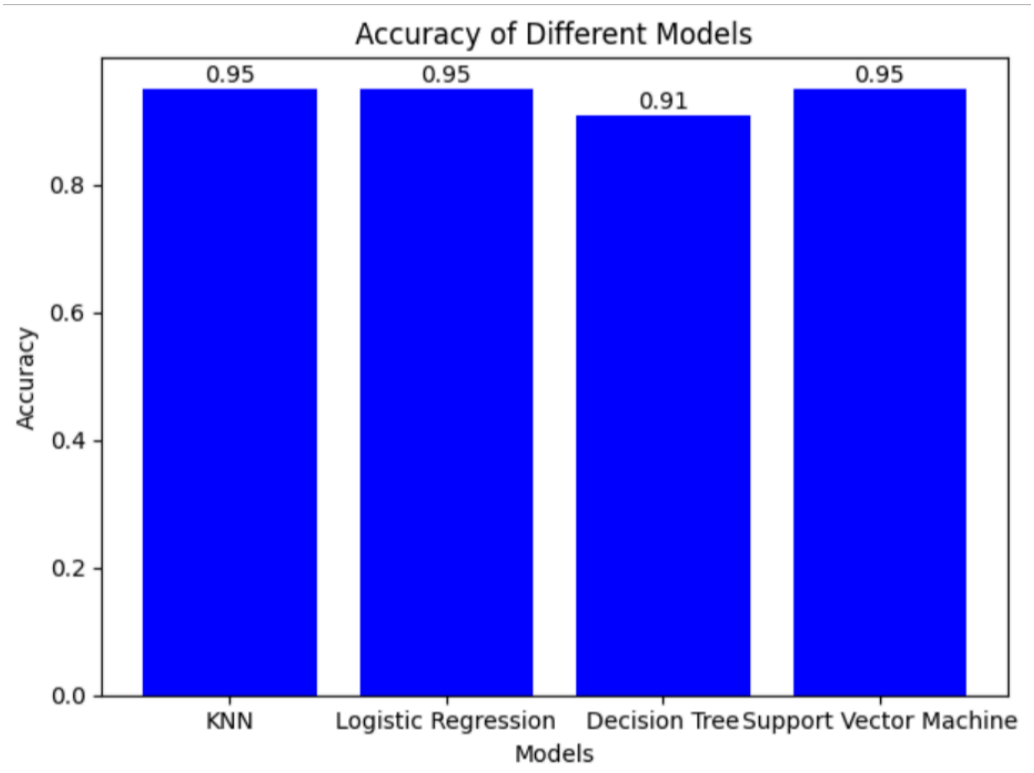
  Support Vector Machine takes the data and plots them, after that an optimal decision boundary is generated which is known as hyperplane. Hyperplane is generated by the support vectors(closest opposite classifier points). Hyperplane is off the equal distance from the Support vectors. In the testing period, the decision is made by comparing the point's position with the hyperplane's position and assigned a label to it. Our Support Vector Machine gave us an accuracy of 95.04%.



Similar to Logistic Regression, as the points plotted are very concentrated around ‘0’ output, our model does not work at all for ‘1’ outputs. This causes the precision and recall of this model to be 0, which means that for our dataset, Support Vector Machine predictions are irrelevant.

7. Model Selection/Comparison Analysis:

We can compare the accuracy of our models using a bar chart.



As the bar chart represents, KNN, Logistic Regression and Support Vector Machine give us the same level of accuracy, whereas the accuracy given by the Decision Tree model is slightly lesser. However, from the above information, we can see that precision and

recall values for Logistic Regression and Support Vector Machine models are 0, so these models can be considered irrelevant for our dataset. Precision values for KNN and Decision Tree models are 0.5 and 0.0988 respectively. This means that the KNN model gives us more relevant outputs. On the other hand, recall values for KNN and Decision Tree models are 0.027 and 0.108 respectively. This means that the percentage of total relevant results is better in the Decision Tree model. The recall value for Decision Tree is just slightly higher for KNN than that of Decision Tree, but the precision of KNN is much better than the precision of Decision Tree model. Therefore, we can conclude that KNN is the best model for our dataset out of these four models.

8. Conclusion:

Our models have good accuracy while testing the data, but from the Confusion Matrix we can see that all of these models are not working great for positive outcomes, this happened because the dataset is mostly filled with negative output data. As a result our all the models are working better for negative outputs. Our models have very low precision but among these KNN has the highest precision (0.5) and it also has highest accuracy (95%). Therefore, KNN is the most suitable model for this dataset. So according to precision data we found that KNN and Decision Tree are the best fittest models because they were able to identify some positive outcomes. On the other hand, we are claiming both SVM and Logistic Regression as the worst fitting model even though both of them gave accuracy around 95% because they were unable to detect positive outcomes.