**Pattern Recognition**

* **Shrey Modi**

1. **Model Prediction**

**1.1 Importing Libraries**

The first step is importing the required libraries for different models that are going to be used to get the results.

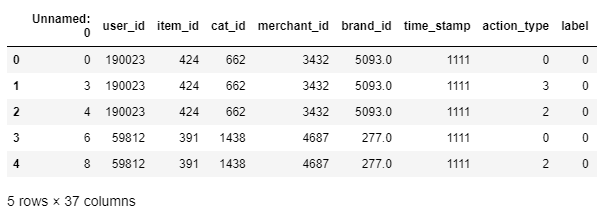
Some of the libraries that are included are:

* Sklearn
* Keras
* Scipy
* Matplotlib
* imblearn

**1.2 Dataframe**

The next step involves the data that has been obtained from the feature engineering, which has enhanced and unique columns for better training and testing purpose.

After the feature engineering the columns increased to 37 and the head of the dataframe can be seen in the figure below.



**1.3 Replacing The Null Values**

Before proceeding forward, it is always good to check if there are null values in the data or not for better training results. If null values are found then it can have a significant impact on the results of the feature selection process. Some feature selection algorithms may not be able to handle null values and may require imputation of the missing values before the feature selection process.

**1.4 What happens when the model is directly run on the dataframe?**

There is no harm in running the model without having a proper analytics on the dataset and many times the data are clean so the model gives a proper accuracy.

Also PCA on features has not been performed which can result in the less accuracy and may be a major reason for this.

After running 25 epochs we get an accuracy of 84 percent and about the same as that for the validation set for the training data which can be seen in the figure

The authors might have though that it is a proper accuracy as the hyperparameter tuning was done and still the accuracy was the same.

The different hyper parameters which were tried by the authors are:

Activation function: ‘relu’, ‘leakyrelu’ , ‘sigmoid’ ,’ softmax’

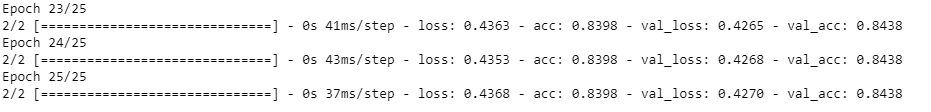
Loss function: ‘mean squared error(mse)’, ‘ binary\_crossentropy’, ‘categorical\_crossentropy’

Epochs: ‘5’ , ‘10’ , ‘25’ , ‘50’ ,’100’

Batch\_size: ‘’1’, ‘16’, ‘32’ , ‘64’ , ‘128’

But when we go to the evaluation of the performed model we get the results in figure

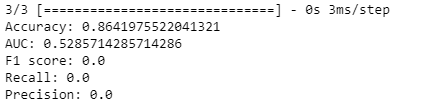
Which is about the AUC score , F1 score, Recall and Precision. This was a big blow for the authors as the F1 score , recall and precision were 0 which states that the data is imbalanced and due to which the values were 0.



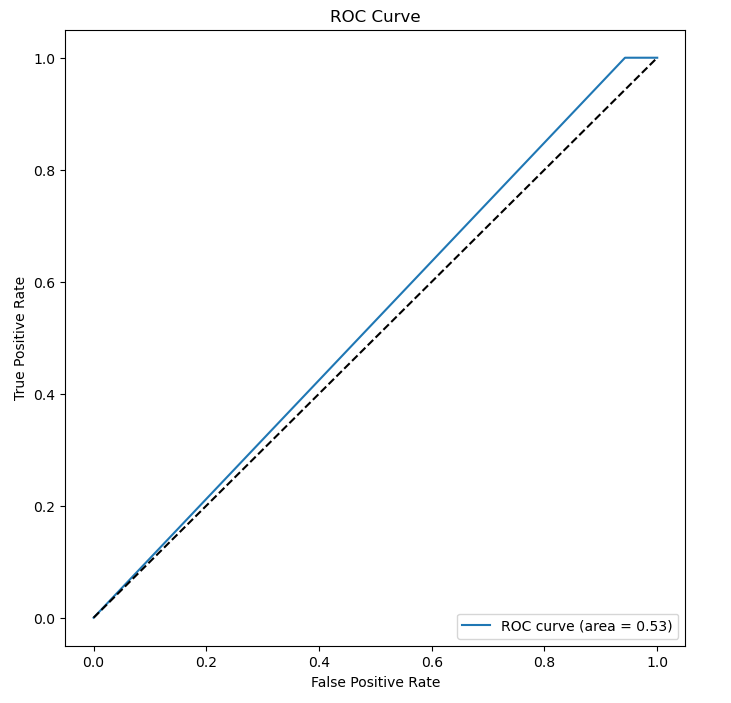
Figure



Figure

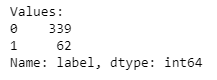


Figure

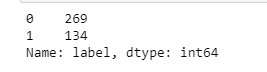


**1.5 Handling Data Imbalance**

When the target value is segregated we get to see that the label 0 has about 339 data and the value 1 has about 62 data which can be seen in figure . To cater this up oversampling and undersampling is done for the minority class which is ‘1’ and for the majority class which is ‘2’. The technique which is used to do this is SMOTE(give referencing) which helps to make the data balanced for the model. In figure we can see the data has been balanced and it is in the ratio of 1:2.



Figure



Figure

**1.6 Scaling**

Scaling is done to the features to bring them to a common scale or range. It is necessary because different features can have different units or scales. For example, in our case the scale for the feature average\_days\_between\_purchases which has float values while the item\_id feature is in integer so for this case scaling is very much necessary. It can be seen in figure

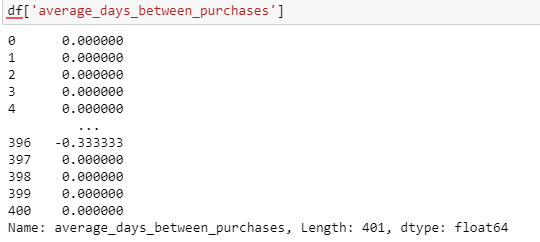


Figure:

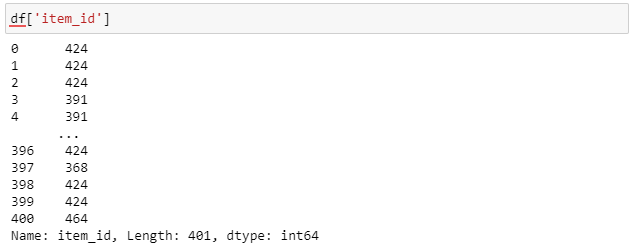


Figure:

**1.7 PCA on Features**

PCA can help in this problem statement by reducing the dimensionality of the feature set, which can improve the performance of neural network models by reducing the risk of overfitting and improving generalization. In this particular case, PCA can help by identifying the most important features that contribute to a customer being a repeat buyer, allowing for a more efficient use of resources and promotional efforts towards potential loyal customers. It can also help in identifying any patterns or clusters within the data, which can aid in understanding the behavior of customers and inform future marketing strategies. To implement PCA in this problem statement, one could first preprocess the data by transforming the categorical variables into numerical features, and then scaling the data to ensure that each feature has equal importance. Then, PCA could be applied to the feature set to identify the most important components that explain the majority of the variance in the data. The resulting components could then be used as input features in a neural network model to predict the probability of a customer being a repeat buyer.

**1.8 Neural Network Results after data balancing**

After the data balancing was performed, the neural network results increased from 86 percent to 95 percent and the f1 score, AUC, recall and precision also increased from 0, and we can tell that there is no problem with the dataset and performs well on the model. The metrics can be seen in figure and the roc curve and the confusion matrix in figure respectively.

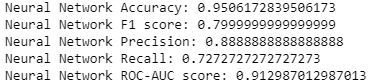


Figure:

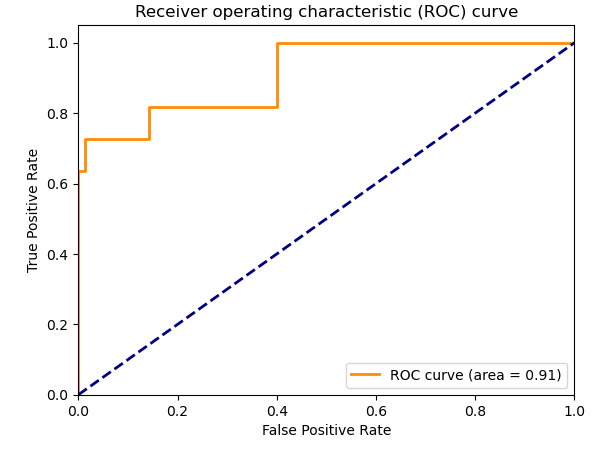


FIgure:

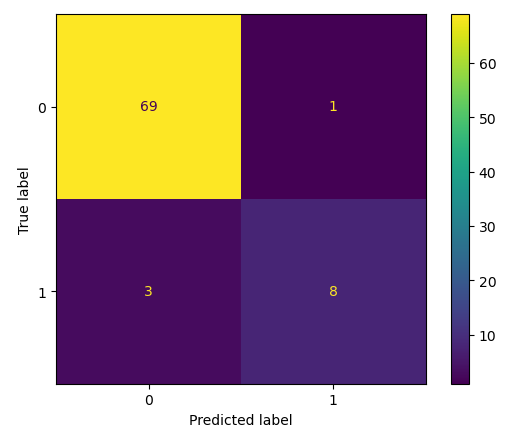


Figure:

**1.9 Removing the correlated features**

As the amount of features is a lot in numbers, the correlation needs to be taken out between the features and for which the authors have taken out a correlation matrix which shows how the features are correlated with each other in figure .

We eliminate correlated features from our model because they may cause multicollinearity issues. When two or more features in a dataset are highly correlated with one another, this is called multicollinearity. This can cause the model to double-count the effect of these features, which can lead to estimates of the model's parameters that are unstable and unreliable. The model's complexity can be reduced, its interpretability can be improved, and its performance could possibly be improved by removing correlated features. Additionally, the risk of overfitting the model to the training data can be mitigated by removing correlated features.

For this approach RFECV is used which Recursive Feature Elimination with Cross-Validation. It is a procedure utilized for highlight determination by choosing the ideal number of elements for a model. It works by removing features from the model in a recursive manner and cross-validating its performance until the optimal number of features is reached.

The algorithm applies a subset of the remaining features to the model during each iteration, and cross-validation is used to assess the model's performance. Based on the estimator's importance scores, the least important features are removed to reduce the number of features. This cycle go on until the ideal number of elements is, not entirely set in stone by the cross-approval score.

By selecting only the most essential features, reducing the model's complexity, and enhancing its interpretability, RFECV is useful for preventing overfitting and improving a model's accuracy.

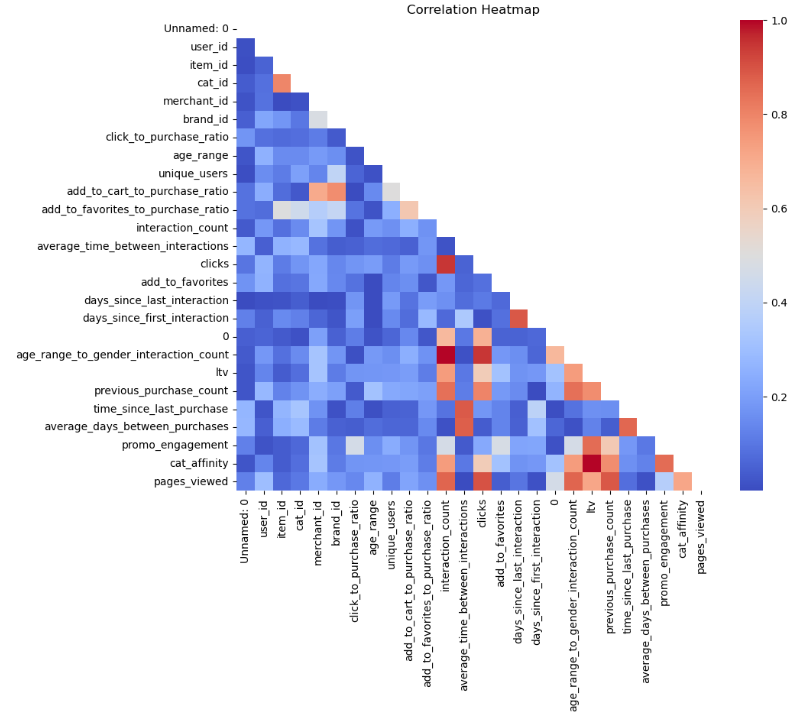


Figure:

**1.10 What happens to the neural network after the correlated features are removed?**

Many would think that the accuracy would have shot up as the model would be trained better on the data, but that isn’t the case as the accuracy of the model when trained on the same hyperparameters and fine-tuned it to see if there is any change, but there isn’t any. We can see the accuracy in figure and the graphs in figure respectively.

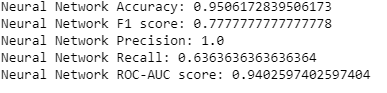


Figure:

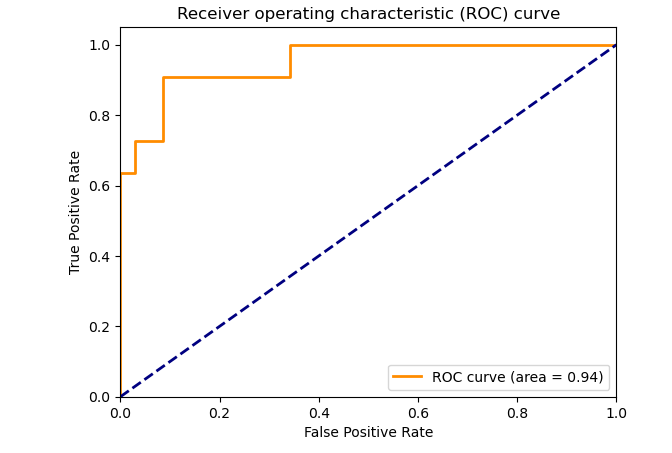


Figure:

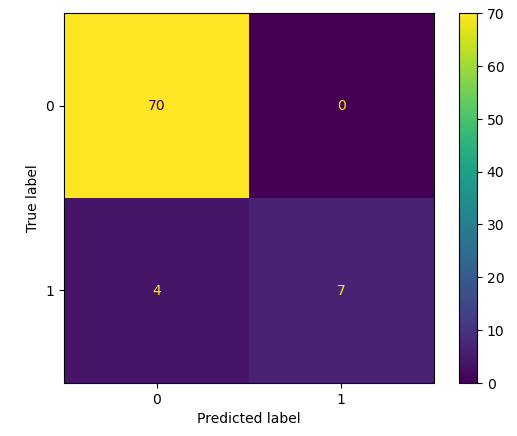


Figure:

**1.11 Why Neural network has been used?**

In the case of predicting customer behavior, neural networks can be used to learn patterns and relationships in the data that may be difficult or impossible for humans to detect. By training on historical data, the neural network can learn to recognize common features or characteristics of customers who are likely to become repeat buyers. Once the network has been trained, it can be used to make predictions on new data, allowing businesses to target their marketing efforts more effectively. One of the strengths of neural networks is their ability to learn complex, non-linear relationships between inputs and outputs. This can be particularly useful in predicting customer behavior, as there may be many different factors that contribute to whether or not a customer becomes a repeat buyer. By using a neural network, we can capture these complex relationships and use them to make more accurate predictions.

**1.12 Visual Evaluation of Non-Parametric techniques:**

The ROC curve and the confusion matrix for the non parametric approaches such as naive bayes, logistic regression, LDP and XGBoost are given in the following figure below.

The main reason for getting the ROC curve and the other metrics such as the F! Score , precision , recall and the AUC is to judge which mode performs better and how.

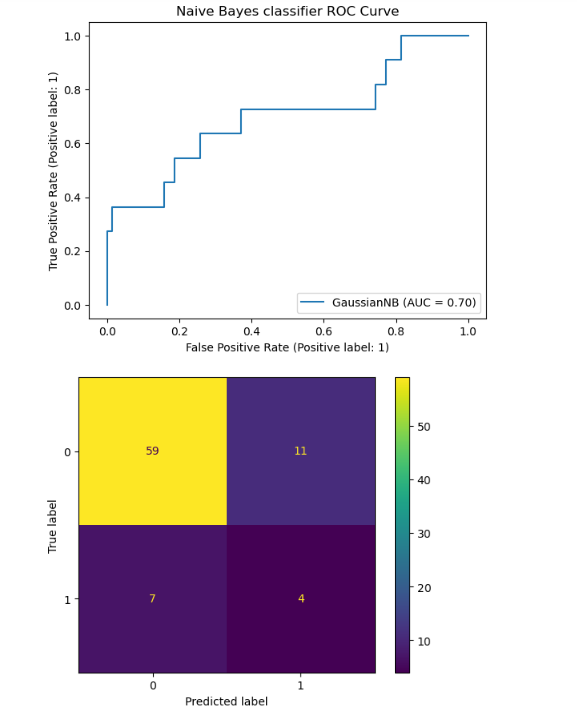
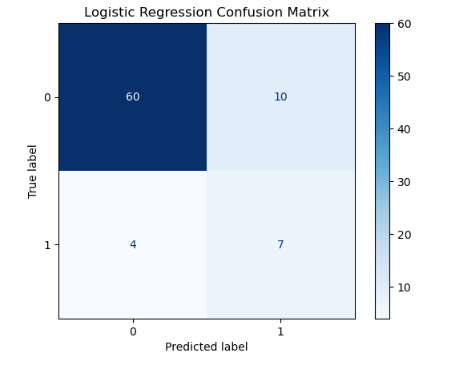


Figure:



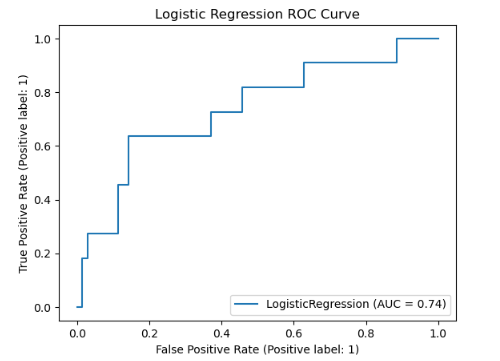


Figure:

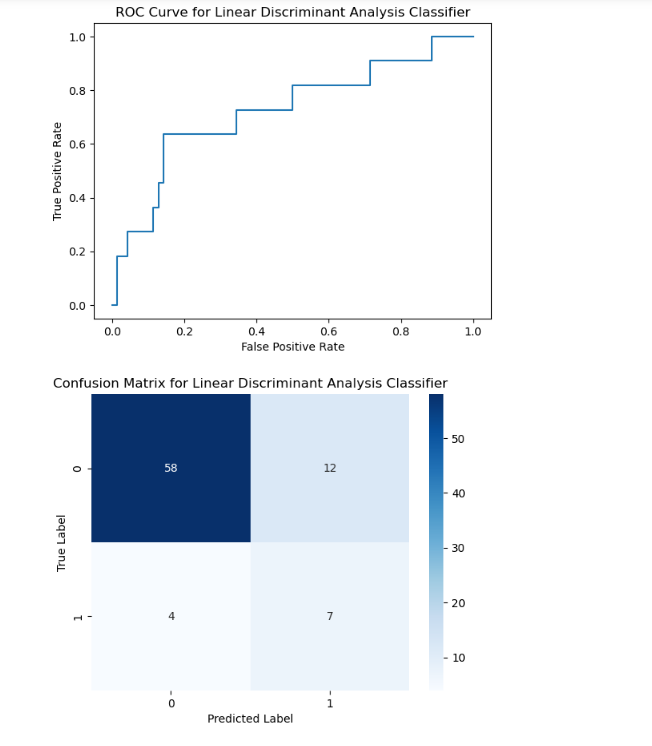
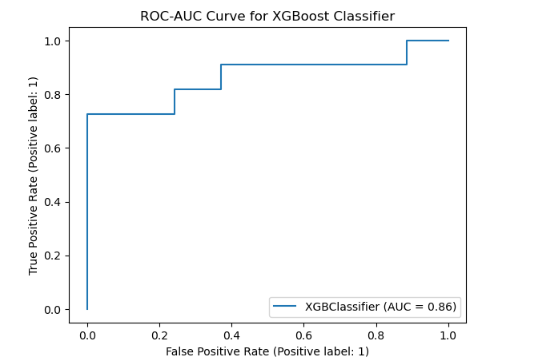


Figure:



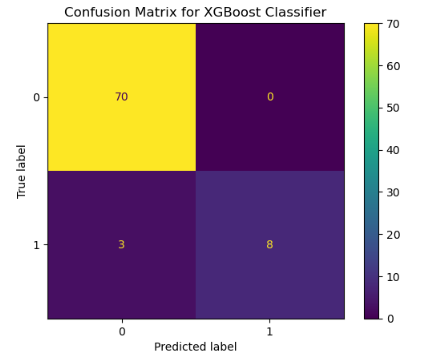


Figure:

**1.13 Recommendation System**

The system needs to be made to recommend the user of different item ID based on the purchases he has made. For this, a simple method needs to be followed:

* Taking the user ID and finding the unique users in the dataset: If the customer has just purchased items less than 2 then there are no items to recommend for them because it is early to judge what they like, and it would lead to randomness.
* Remove duplicate item id’s
* Create a list of item IDs that the customer has not purchased
* Calculate the probabilities of the customer buying each unpurchased item
* Sort the unpurchased items in decreasing order of probability
* Add the top 4 items to the recommendations for the customer

Results can be seen in figure



Figure:

**1.14 Models strengths and weakeness**

| **Models** | Strengths | Weakness |
| --- | --- | --- |
| Neural Network | * Can model highly complex relationships between input and output variables. * Can work with various types of input data, including images, audio, text, and time series. * Can learn from large datasets without overfitting. | * Can be computationally expensive and time-consuming to train, especially for large datasets and complex architectures. * Can suffer from overfitting if the network is too large or the training data is limited. * Requires a large amount of training data to achieve high accuracy. |
| Naive Bayes Classifer | * Simple and easy to implement. Can be trained with small datasets. * Performs well in high-dimensional spaces. * Fast to train and classify. | * Assumes that features are independent of each other, which may not always hold true in real-world scenarios. * Can suffer from the "zero-frequency" problem, where the model assigns zero probability to unseen feature values. * May not perform well when the distribution of the training data is significantly different from the test data. |
| Parzen Window | * Can handle arbitrary data distributions, including non-parametric and multimodal distributions. * Does not make any assumptions about the underlying data distribution. * Simple and easy to implement. | * Can be computationally expensive and slow, especially for high-dimensional data. * The choice of kernel Function can significantly affect the performance of the model. * Can suffer from the curse of dimensionality, where the number of dimensions in the input space is large compared to the number of training examples. |
| XGBoost | * Performs well on a wide range of structured data problems. * Fast and scalable, even for large datasets. * Can handle missing data and categorical features. * Provides built-in feature importance ranking. | * Requires careful tuning of hyperparameters to avoid overfitting. * Can be sensitive to outliers in the training data. * Can be challenging to interpret the inner workings of the model. |
| LDF | * Simple and easy to implement. * Performs well on linearly separable datasets. * Provides a clear separation boundary between classes. | * Assumes that the covariance matrix of each class is the same, which may not always hold true in real-world scenarios. * May not perform well on non-linearly separable datasets. * Cannot handle missing values or categorical data. |
| KNN | * Non-parametric, meaning it does not make any assumptions about the underlying data distribution. * Simple and easy to implement. * Performs well with small datasets and high-dimensional spaces. | * Can be computationally expensive and slow, especially for large datasets. * Requires a lot of memory to store the entire dataset. * Can be sensitive to the choice of distance metric. * Cannot handle missing values or categorical data. |

**1.15 Why parzen window performed the best?**

The Parzen Window classifier is a non-parametric density estimator that can handle arbitrary data distributions, including non-parametric and multimodal distributions. It does not make any assumptions about the underlying data distribution, which is an advantage over other classifiers that have to make assumptions about the distribution, such as the Naive Bayes Classifier or Linear Discriminant Analysis. In addition, the Parzen Window classifier does not require any training, which can be an advantage over other classifiers that require a large amount of training data to achieve high accuracy. This can be particularly useful in situations where the amount of training data is limited or where the cost of acquiring training data is high. Moreover, the Parzen Window classifier can handle multi-class classification problems and can be used with any distance metric, making it very flexible such as in our case by handling various features . However, the choice of kernel function can significantly affect the performance of the model. By selecting an appropriate kernel function and tuning the parameters of the classifier, the Parzen Window classifier can achieve high accuracy and consistent performance across different datasets. Overall, the Parzen Window classifier's ability to handle arbitrary data distributions, flexibility, and lack of training requirements make it a strong candidate for classification tasks where the underlying data distribution is unknown and the amount of training data is limited. Hence, it can has the highest and most consistent metrics amongst all the methods and can be considered the best classifier for this data.

While the neural network approach almost performed similar to the parzen window technique in terms of the accuracy metrics but didn’t perform well on the other metrics just like the parzen window and hence we can say that the parzen window is the best technique for this problem.