Bank Marketing Campaign Analysis

Group 7

Sai Revanth Reddy Boda
Bharat Raju Palla
Akhila Reddy Bokka
Krishna Teja Samudrala

boda.s@northeastern.edu
palla.b@northeastern.edu
bokka.a@northeastern.edu
samudrala.k@northeastern.edu

Percentage of Contribution by Student 1: 25%

Percentage of Contribution by Student 2: 25%

Percentage of Contribution by Student 3: 25%

Percentage of Contribution by Student 4: 25%

Submission Date: 06-28-2023

Table of Contents

1.Introduction:

Background of the domain: In today's interconnected world, driven by the expansion of media and technology, businesses face intense competition in marketing their products. This is particularly relevant for banks promoting fixed-term deposit products to their clients. To succeed in this competitive landscape, banks must analyze various factors, including the target audience, campaign strategies, and market conditions. By leveraging machine learning models, banks can predict customer subscriptions and implement effective strategies to boost the subscription rate.

2. Problem Definition:

The study uses data from a Portuguese banking institution's Bank Marketing Dataset (BMD) to examine and predict the success of bank telemarketing. There are 18 characteristics and 41,188 observations in the dataset. We aim to identify the key factors that clients consider while deciding whether to sign up for a term deposit using a number of machine learning techniques and methods. The dataset, the problem definition, the data resources, the data description, and the independent and dependent features used in the analysis are all thoroughly covered in this report. Using data from a Portuguese financial institution's Bank Marketing Dataset (BMD), the study analyzes and forecasts the success of bank telemarketing.

3. Data Source:

UCI Machine Learning Repository, Center for Machine Learning and Intelligent Systems (https://archive. ics.uci.edu/ml/datasets/Bank+Marketing#)

4. Data Description:

The dataset contains 18 attributes, which consists of 41,188 observations, include information on the clients, including their demographics, communication preferences, campaign information, and economic factors. The final objective is to forecast whether a customer will sign up for a bank term deposit, which is represented by the target variable "y" with values of "yes" or "no."

Feature	Description
y	desired target.t subscribed a term deposit? (no, yes)
Age	The age of the client in years.
Job	This is a categorical variable that could be indicative of a client's income level or their occupation.
Martial	The marital status of the client. It is indicative of a client's financial status or their family situation.
Education	The educational level of the client. It depicts client's financial literacy or their earning potential
Default	Whether the client has a history of defaulting on a loan ('yes' or 'no'). This is a binary variable is indicative of a client's creditworthiness.
Housing	Whether the client has a housing loan ('yes' or 'no').
Loan	Whether the client has a personal loan ('yes' or 'no'). This is a binary variable that could be indicative of a client's financial status or their debt load
Contact	How the client was contacted ('cellular', 'telephone', or 'unknown'). This is a categorical variable that could be indicative of a client's accessibility or their preference for communication.
Month	Month in which the client was contacted ('jan', 'feb', ..., 'nov', or 'dec').
Day	Day of the month in which the client was contacted. This is a categorical variable that could be indicative of a client's availability or their mood
Duration	Duration of the phone call in seconds.

Campaign	Number of contacts made to the client during the campaign
pdays	Number of days that passed since the client was last contacted before the current campaign.
previous	Number of contacts made to the client before the current campaign.
poutcome	Outcome of the previous marketing campaign ('success', 'failure', or 'unknown').
nr.employed	numeric. number of employees - quarterly indicator
emp.var.rate	numeric. employment variation rate - quarterly indicator
cons.price.idx	numeric. consumer price index - monthly indicator
cons.conf.idx	numeric. consumer confidence index - monthly indicator
euribor3m	numeric. euribor 3 month rate - daily indicator

5. Exploratory Data Analysis

The dataset was loaded into a dataframe using the pandas tool for simple manipulation and analysis. Due to the possibility of data leaking, the 'duration' feature was removed. This function calculates how long the consumer spoke on the phone with the bank's marketing representative. Incorporating this time into a predictive model would not produce accurate results because it cannot be known until after the call has concluded (when the customer's outcome is already known).

The following phase involved investigating and cleaning up categorical factors such "job type," "marital status," "education," etc.

Plots were created for each of them, examining both their relative frequency and normalized relative frequency. These graphs were generated in Python using the Seaborn library.
To handle this missing data is the next issue since many of these features have undefined values. Simply removing these rows would drastically reduce the amount of data and have a significant impact on the outcomes. As an alternative, these missing values are calculated using different independent variables. Even though it's not a guarantee, most of the lost data will be retrieved as a result. For instance, the assumption that a person's employment will be influenced by their education led to the introduction of cross-tabulation between "job" and "education."

Therefore, one can predict someone's education level based on their employment. For this cross-tabulation process, The 'home ownership' and 'loan status' features followed a similar cross-tabulation process. It's important to remember the care that was given when developing these claims to make sure the relationships made sense in practice. Otherwise, the values weren't changed. Dealing with missing data among the numerical features is the next step.

It should be noted that even though these values were present only in the 'pdays' (number of days since that consumer was contacted from the prior campaign) column, they made up the majority of the data for this feature. In other words, this column contained more information than it lacked. So we choose to move forward without changing it.

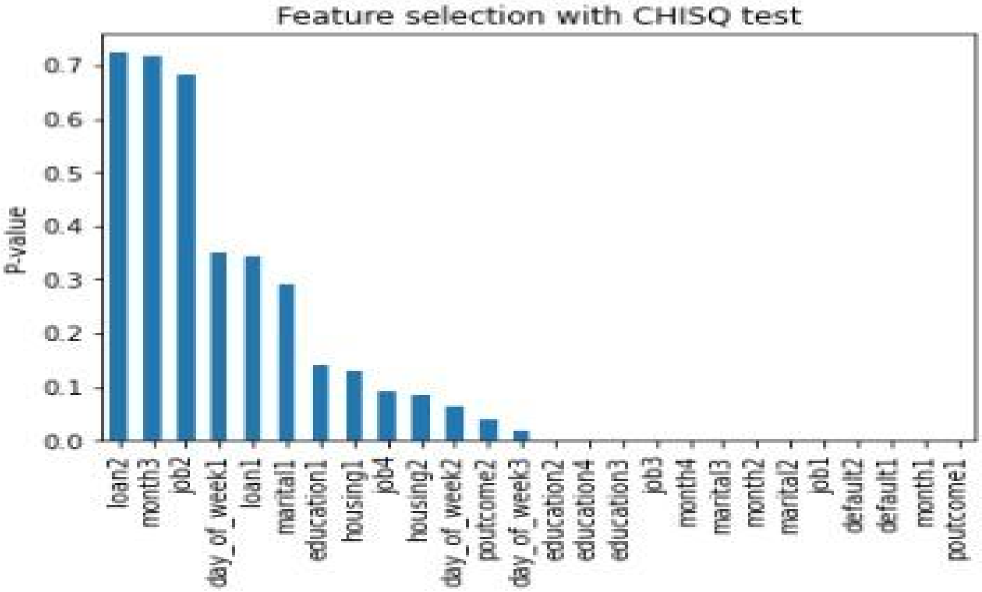
In order to determine whether there is a significant association between the any independent variables, a heatmap was produced. The heatmap is produced using Pearson correlation, which reduces the impact of outliers by measuring how closely each variable's ranks and actual values align. Those factors are anticipated to be significant during the modeling step once this has been measured.

A bank marketing dataset with 41000 data points, features comprising age, income, education, marital status, and others. To predict whether a customer would sign up for a bank term deposit, you should construct a model.

By identifying a set of new features that are a linear combination of the original features, PCA can be used to reduce the dimensionality of the dataset. The brand-new characteristics are referred to as principal components, and they are arranged in and their importance is shown by the order. The principal component that explains the most variation in the data is the first one, then the second, and so on.

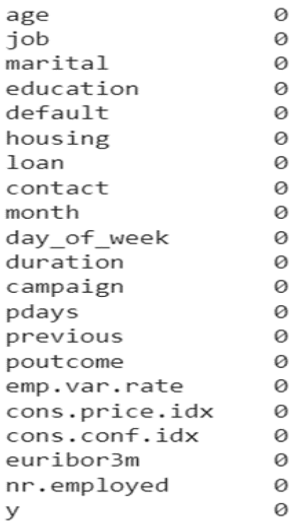
We select the top n eigenvectors after sorting the eigenvectors and project the data onto those eigenvectors. When the greatest variance is obtained for the first n components of the singular values decomposition, the following formula is used:
first n eigenvalues of n
M is the total eigenvalues.
The steps to perform PCA are as follows: -
1. Set the dataset's values to be centered around their mean.
2. Determine the centered dataset's covariance matrix.
3. Determine the covariance matrix's eigenvalues and eigenvectors.
4. Arrange the eigenvectors according to the eigenvalues that relate to each one.
5. Based on the desired number of retained principal components, select the top n eigenvectors.
6. To create a new k-dimensional feature space, project the original data onto the chosen eigenvectors.

To choose the features that are most important to the classification problem, apply the Chi-Square test. The Chi-Square test determines if two variables are independent. If the customer subscribed to a bank term deposit, then you would be testing the independence between each feature and the objective variable.



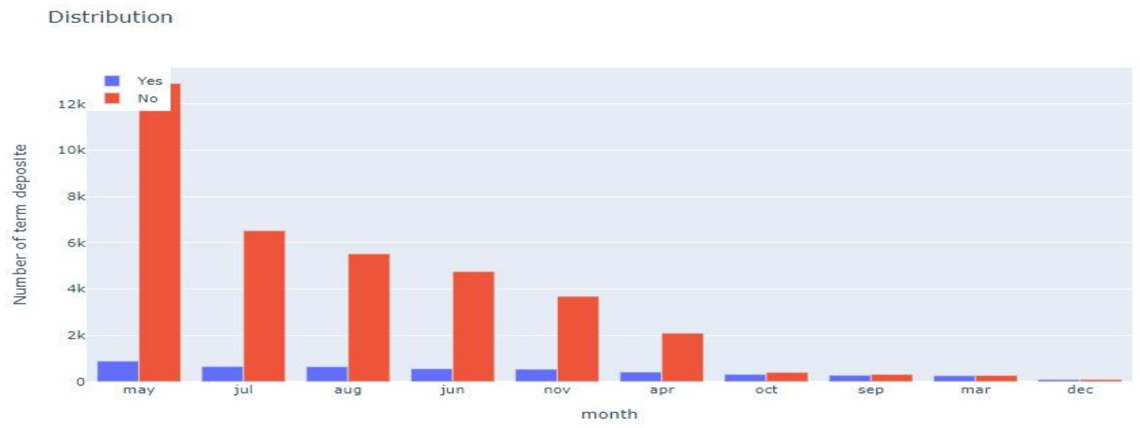
You can safely exclude the features with a high p-value (greater than 0.05) from the dataset because they are considered to be independent of the target variable. Low p-value (less than 0.05) characteristics are thought to be dependent on the target variable

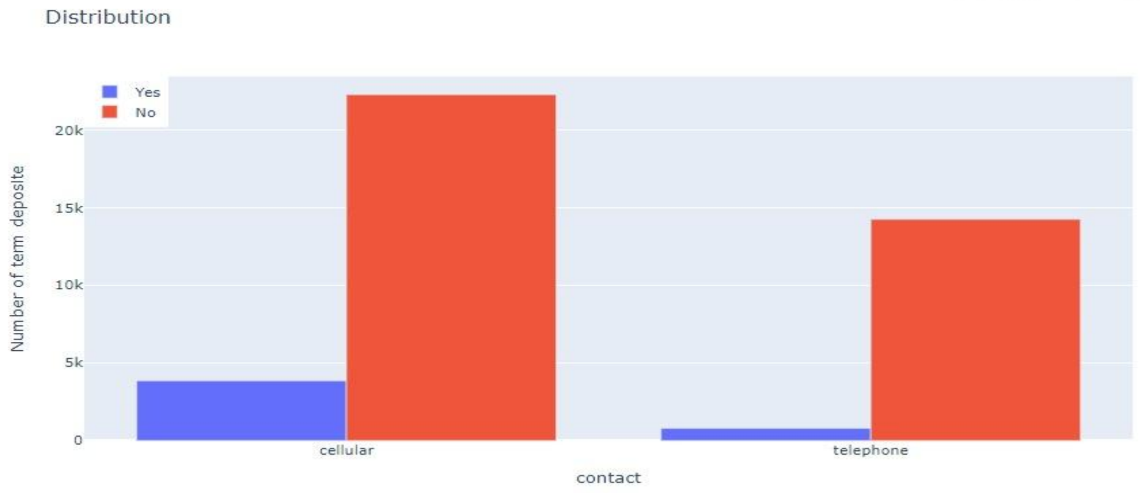
a. This tells us about the null values in the dataset. And then we know the description of the dataset to get an overview about the data and we got to know that there are no null values in our dataset.

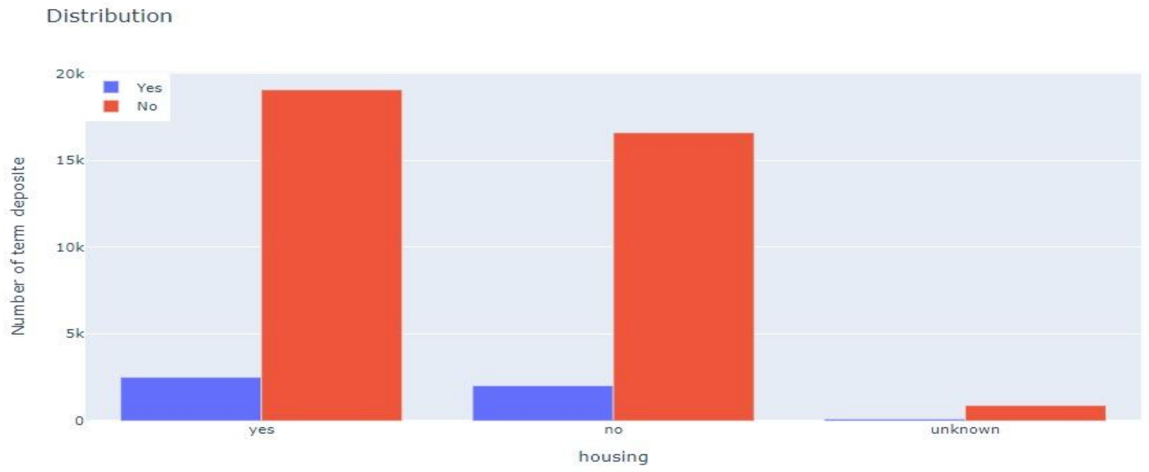


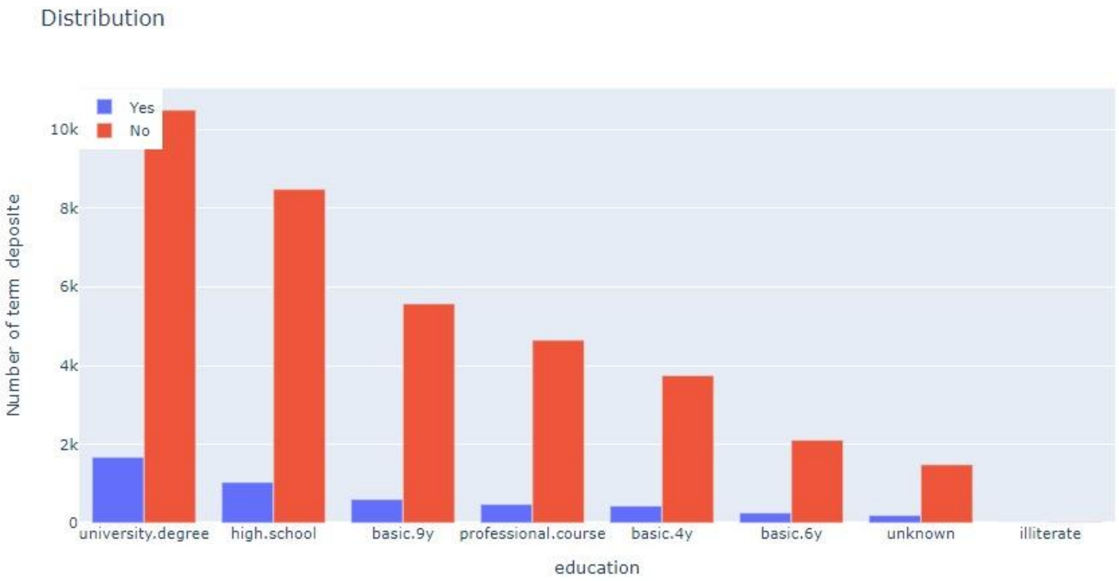
```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
```

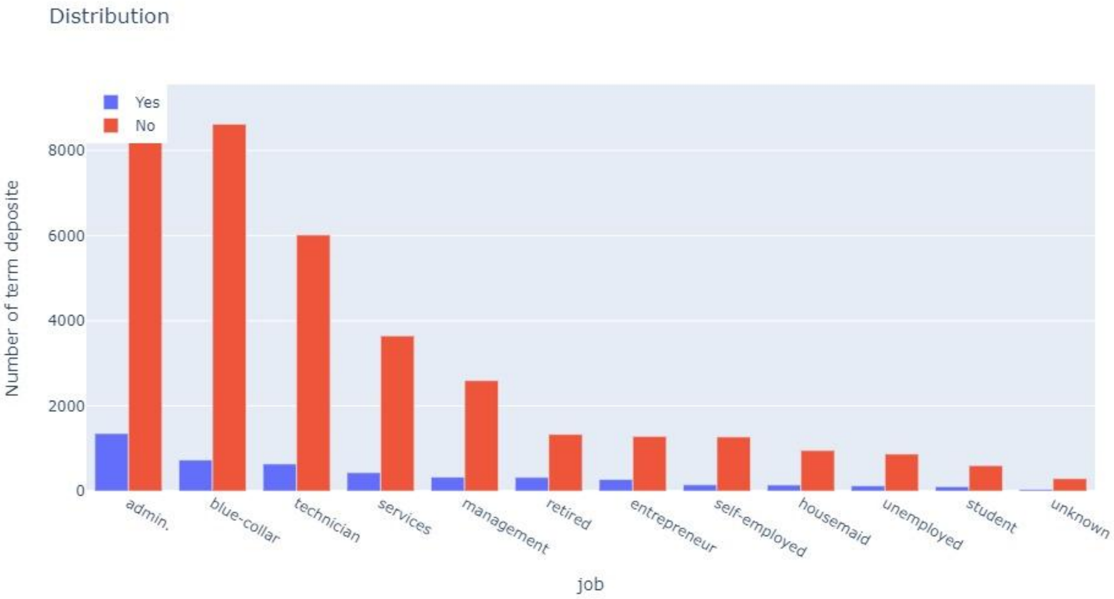
b. Then we have done the categorical and numerical column analysis to know the effect of each column on the target variable using the below graphs



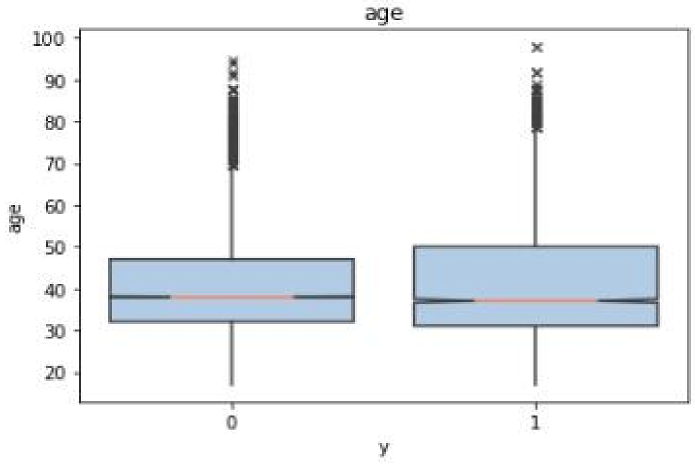


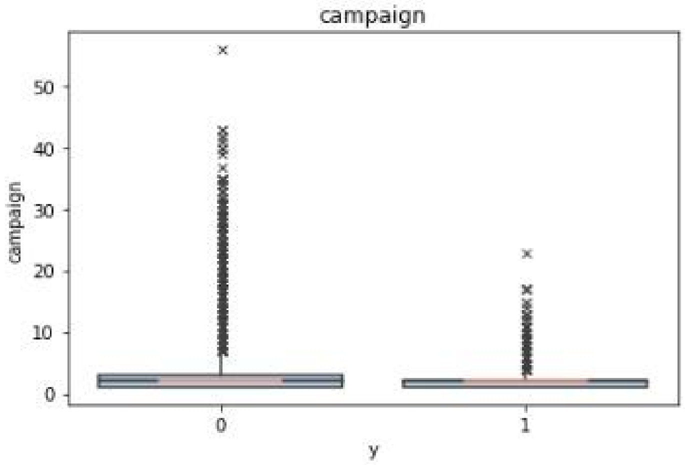


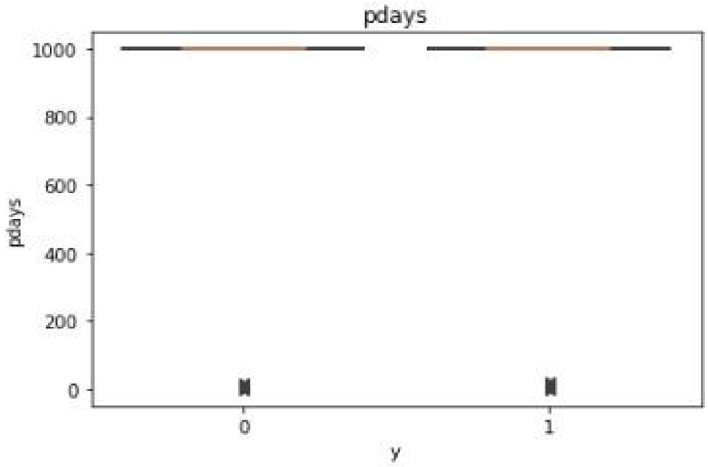

Distribution
Yes
No
Number of term deposite
10k
8k
6k
4k
2k
0
university.degree
high.school
basic.9y
professional.course
basic.4y
basic.6y
unknown
illiterate
education

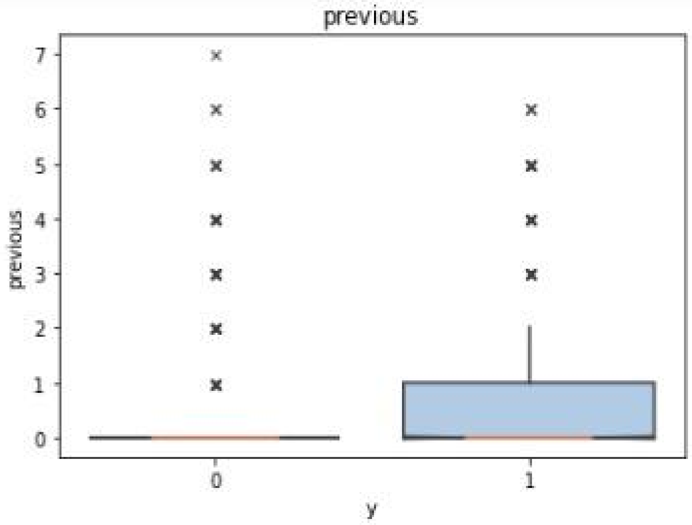

Distribution
Yes
No
Number of term deposite
8000
6000
4000
2000
0
admin.
blue-collar
technician
services
management
retired
entrepreneur
self-employed
housemaid
unemployed
student
unknown
job

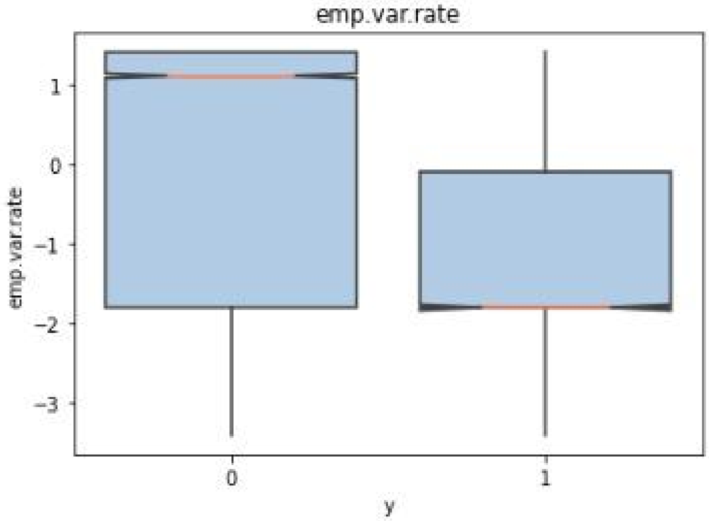
c. From the below boxplot, although there are many outliers. Removing the outliers could cause a lot of data loss so we not did not remove the outliers

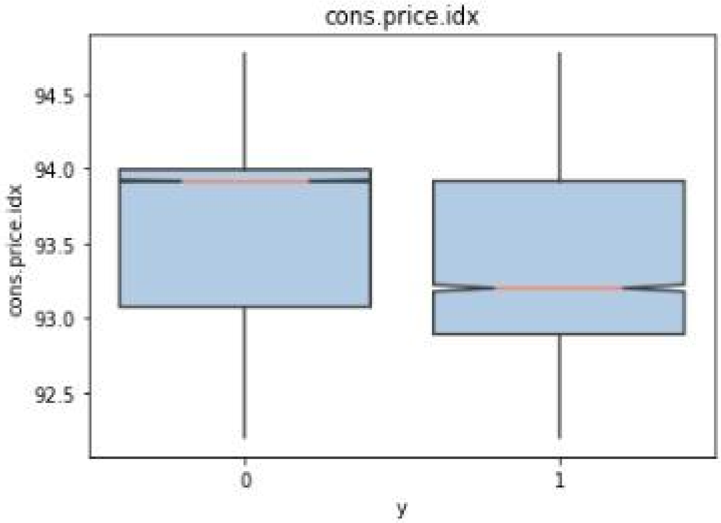


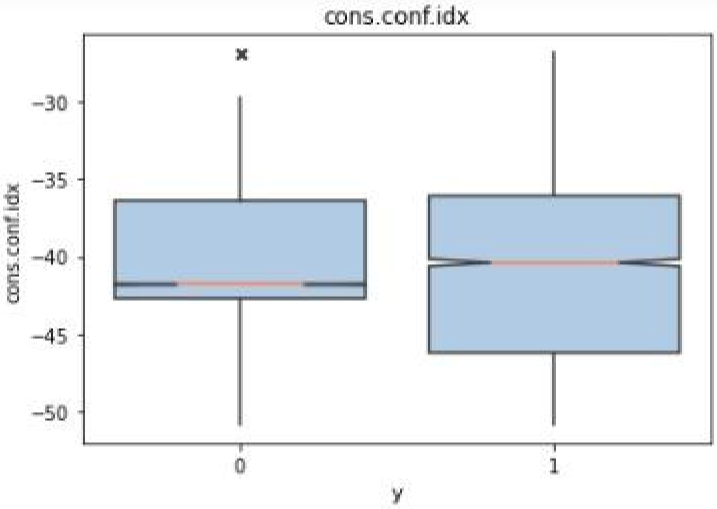


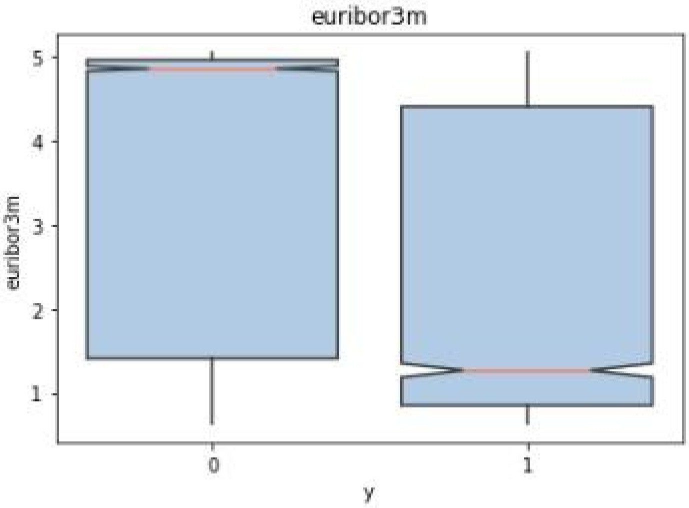


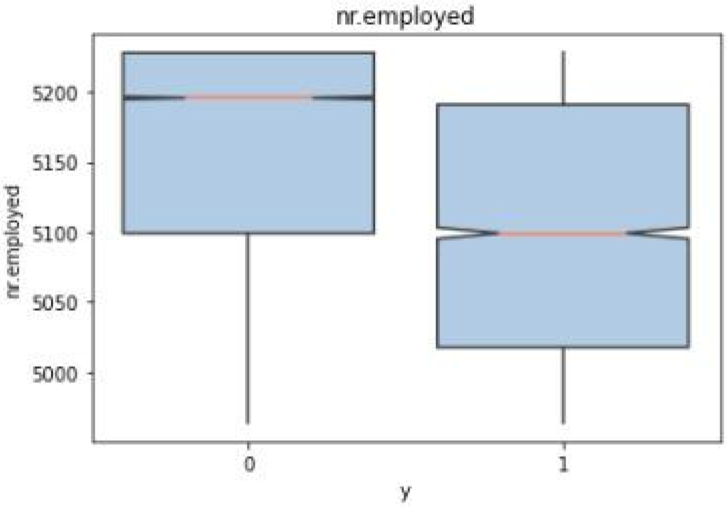

previous
7
6
5
4
3
2
1
0
previous
0
1
y


emp.var.rate
1
0
−1
−2
−3
emp.var.rate
0
1
y

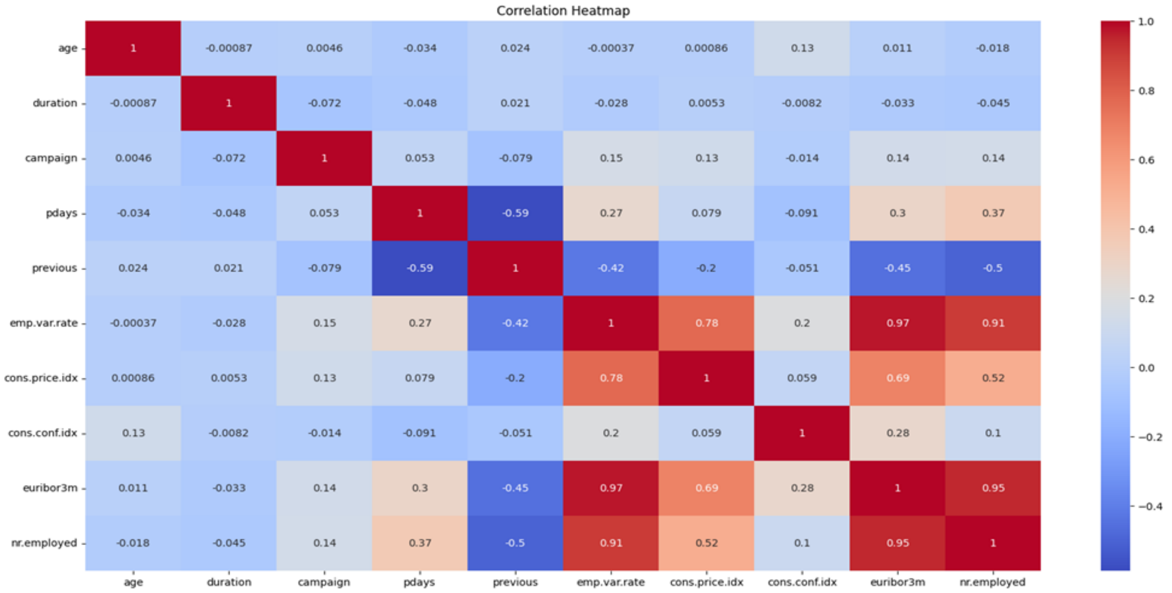

cons.price.idx
94.5
94.0
93.5
93.0
92.5
cons.price.idx
0
1
y

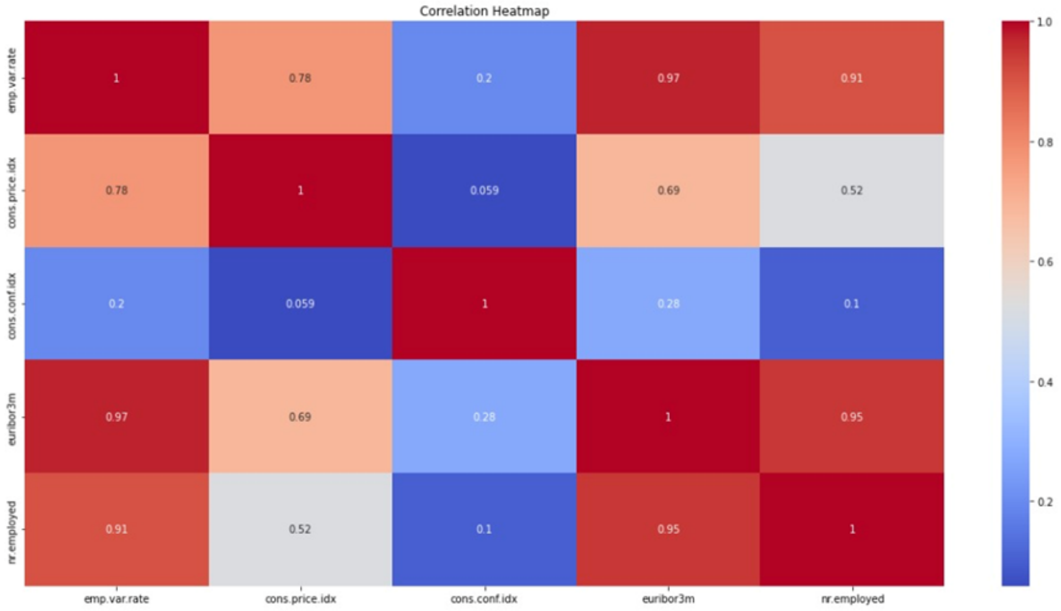

cons.conf.idx
cons.conf.idx
−30
−35
−40
−45
−50
0
1
y


euribor3m
euribor3m
5
4
3
2
1
0
1
y

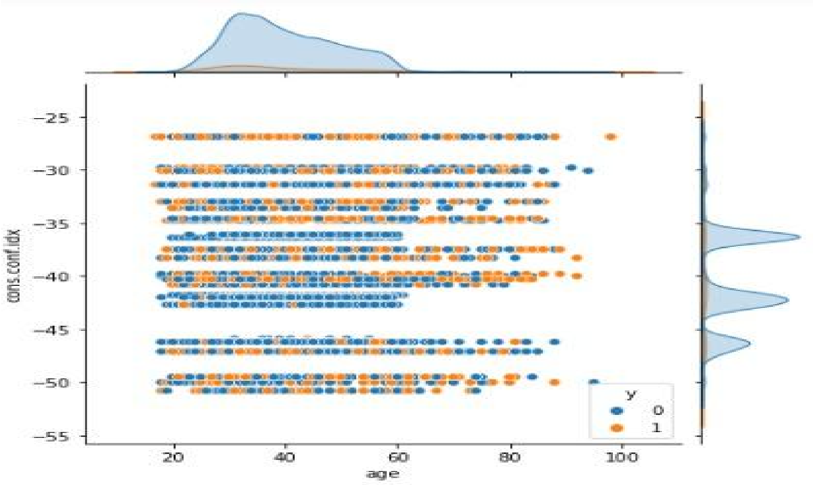

nr.employed
nr.employed
5200
5150
5100
5050
5000
0
1
y

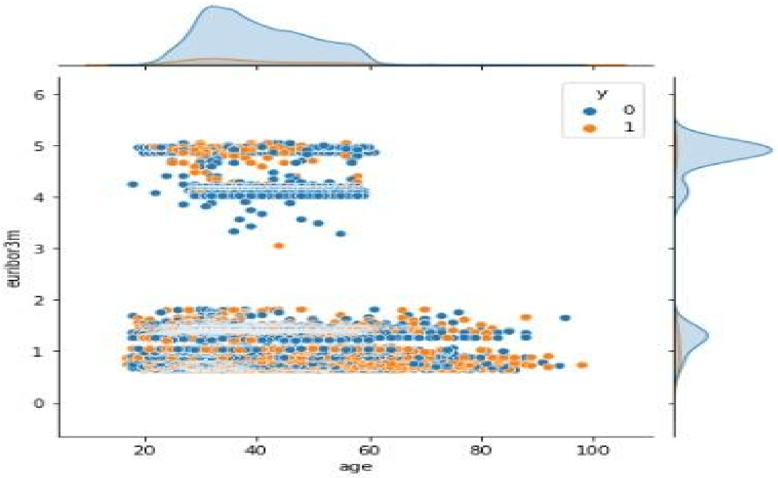
d. Heatmap or correlation matrix to show the correlations between pairs of numerical variables: This plot shows how each feature in the dataset is correlated with every other feature. It can help identify which features are most important for predicting the target variable.

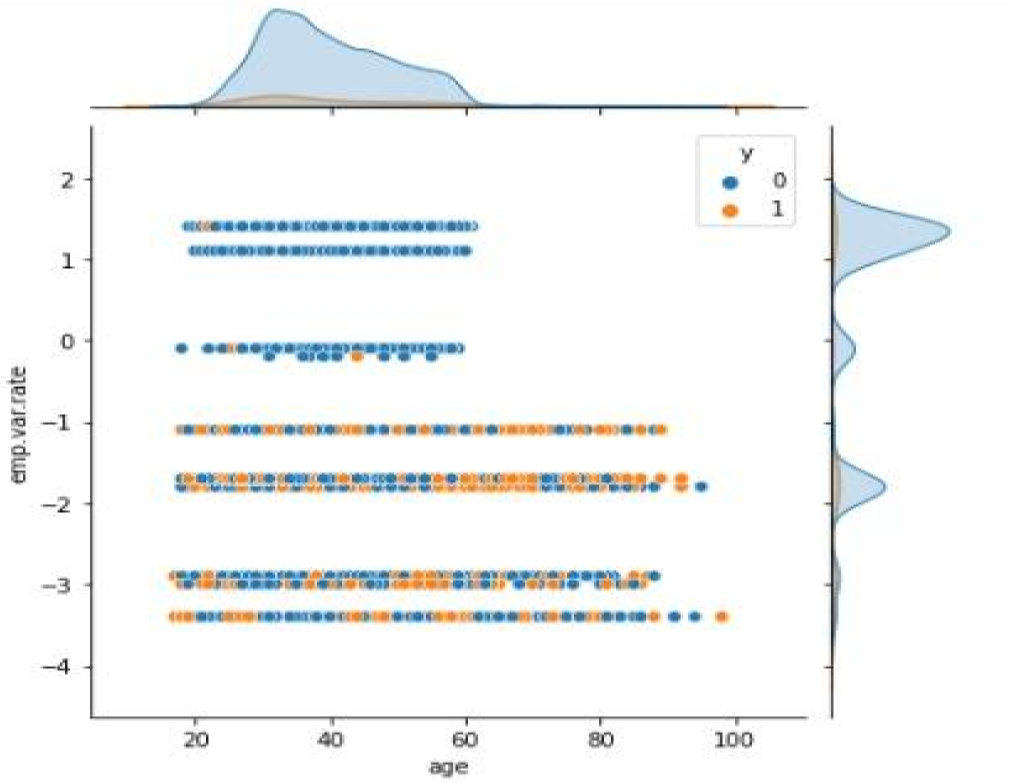


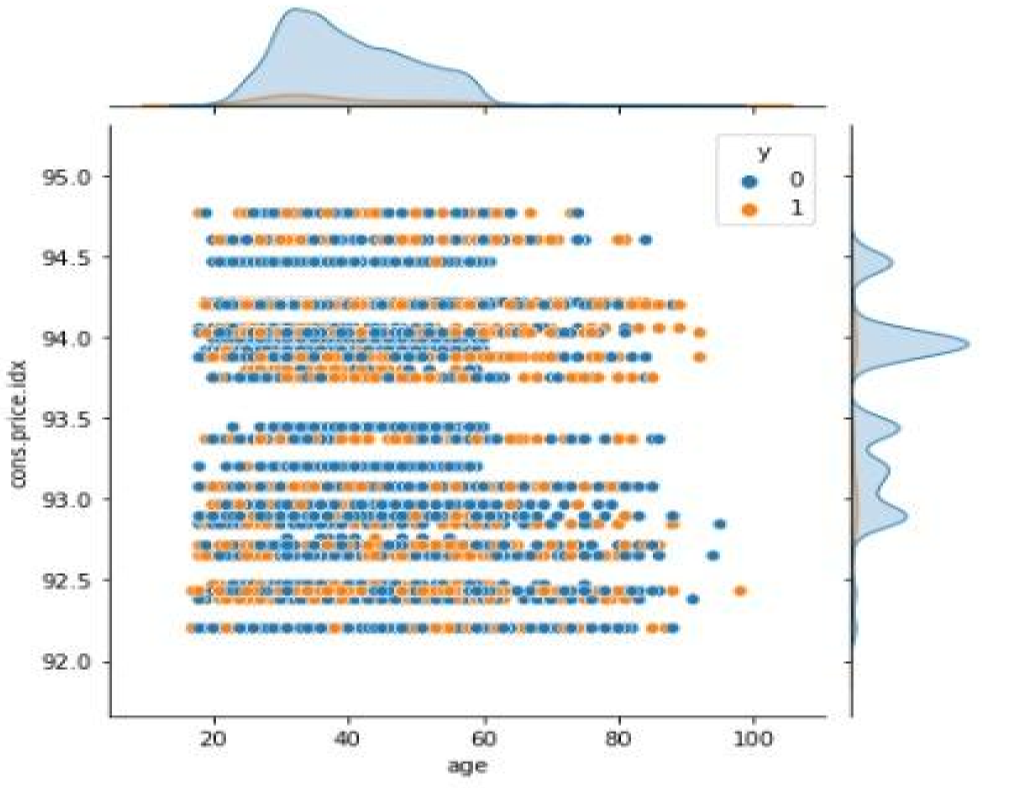


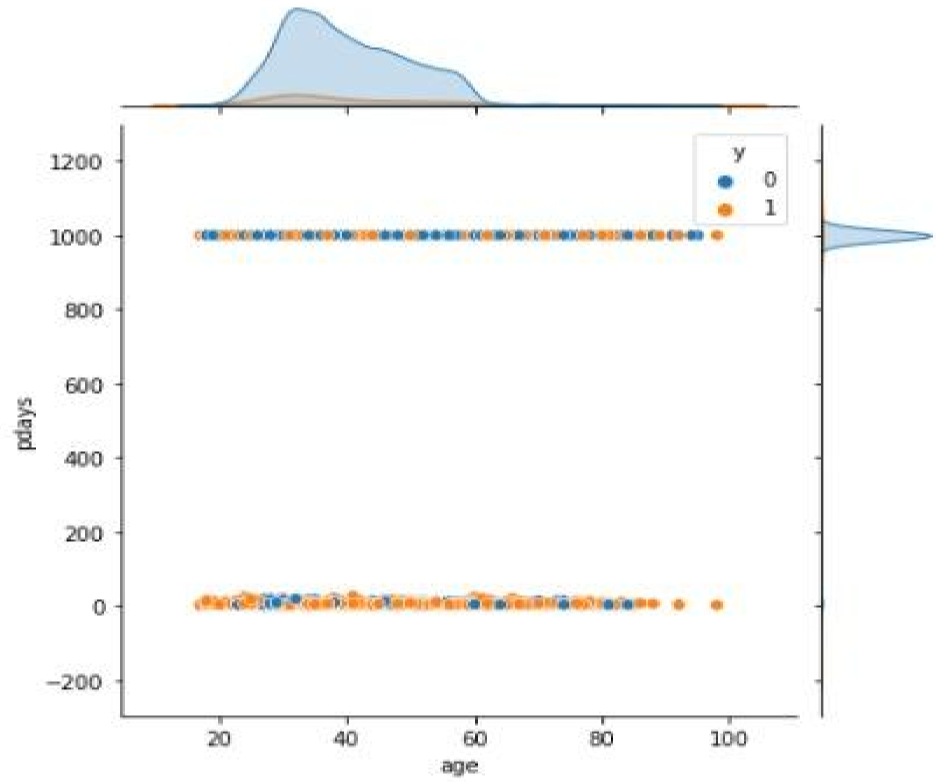
It was observed that the cons.conf.idx column is not correlated after plotting a correlation heatmap for numerical columns, we have to apply PCA on rest of the above columns or we can drop every column keeping one of highly correlated columns.

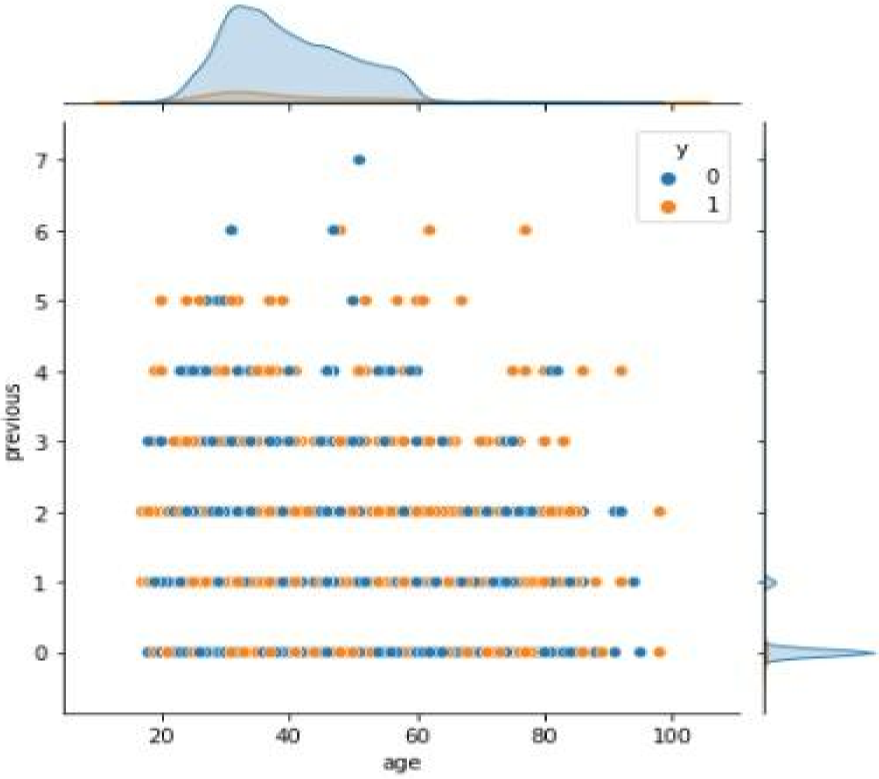


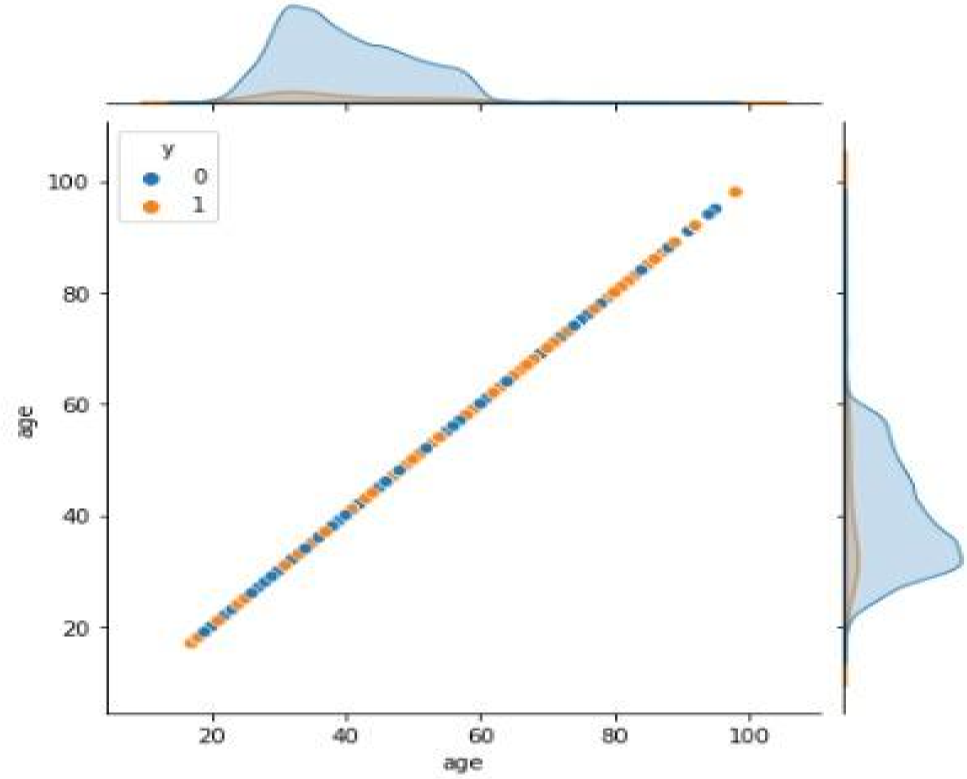


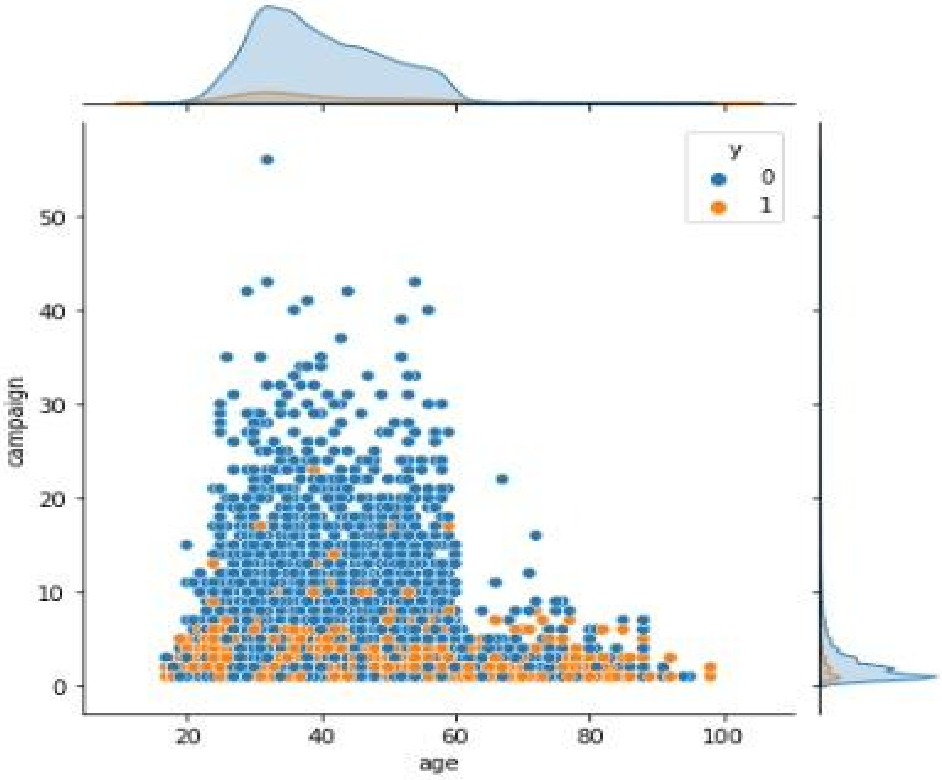

y
0
1
2
1
0
−1
−2
−3
−4
emp.var.rate
20
40
60
80
100
age


y
0
1
95.0
94.5
94.0
93.5
93.0
92.5
92.0
cons.price.idx
20
40
60
80
100
age

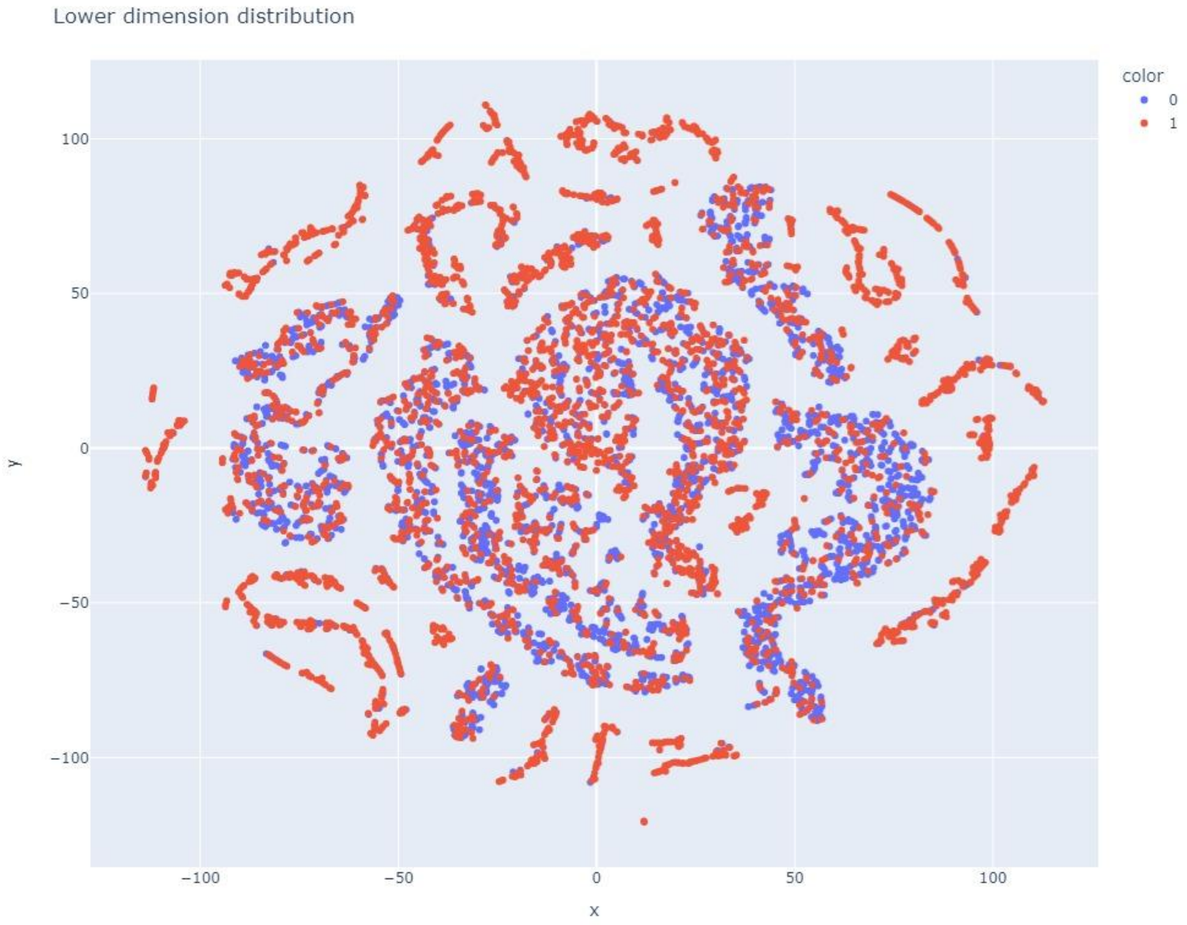

y
0
1
1200
1000
800
600
400
200
0
−200
pdays
20
40
60
80
100
age


y
0
1
7
6
5
4
3
2
1
0
previous
20
40
60
80
100
age


y
0
1
age
age
20
40
60
80
100


y
0
1
campaign
age
0
10
20
30
40
50
20
40
60
80
100

TSNE



From the above plots we can see that there are no seperability in a lower dimensional space across the classes, from which we can hypothesise that most of the classical Machine learning model may not perform weel but we can still have a hope models like Neural networks.

Splitting the dataset: Here, the dataset is splitted into training, validation and testing datasets using the train_test_split() function from the sklearn.model_selection library. The drop() function is used to drop the Revenue column from the features dataset X and assign it to the target variable y. A test size of 0.3, and the rest will be used for training.

Encode categorical variables: Machine learning models cannot work with categorical data in their raw form, so it is needed to encode them into numerical values. There are different methods for encoding categorical variables, and the choice of method depends on the nature of the data and the machine learning algorithm being used. In this dataset, One-hot encoding is used to convert these variables into numerical values. LabelEncoder assigns a unique integer to each category in a categorical variable. This can create an ordering of categories that might not make sense in certain situations. For example, in the Month column, January is encoded as 5, while December is encoded as 2. This can be misleading, as there is no inherent ordering of months. Hence, in this case One-hot encoding is more appropriate to use.In addition, we have used a method called Hashing to make the encoding process efficiency.

Split the data into training, validation and testing sets: To evaluate the performance of our machine learning model, it is needed to split the dataset into training, validation and testing sets. The training set is used to train the model, validation set is used evaluate the performance of the model during training, and the testing set is used to evaluate its performance on unseen data. The 'train_test_split' function from scikit-learn to split the data into training, validation and testing sets. The function takes four parameters: X, y, test_size, and random_state. The X parameter contains the final dataset with all features encoded and concatenated, y contains the target variable, 'test_size' specifies the proportion of the data to use for testing (in this case 20%), and random_state is used to ensure that the split is reproducible.

Balance the dataset: Undersampling is a technique commonly used to tackle imbalanced datasets, where there is a substantial disparity between the number of instances in the majority class compared to the minority class. In the context of bank marketing dataset, imbalanced data often arises as the majority of customers do not make a deposit. With undersampling, the number of instances in the majority class is randomly reduced, creating a more balanced dataset. This approach helps alleviate bias in models and enhances prediction accuracy by ensuring equal representation from all classes during training. Through the implementation of undersampling, we successfully addressed the data imbalance issue, leading to improved model performance and reduced prediction bias.

Exploration of Machine Learning Models

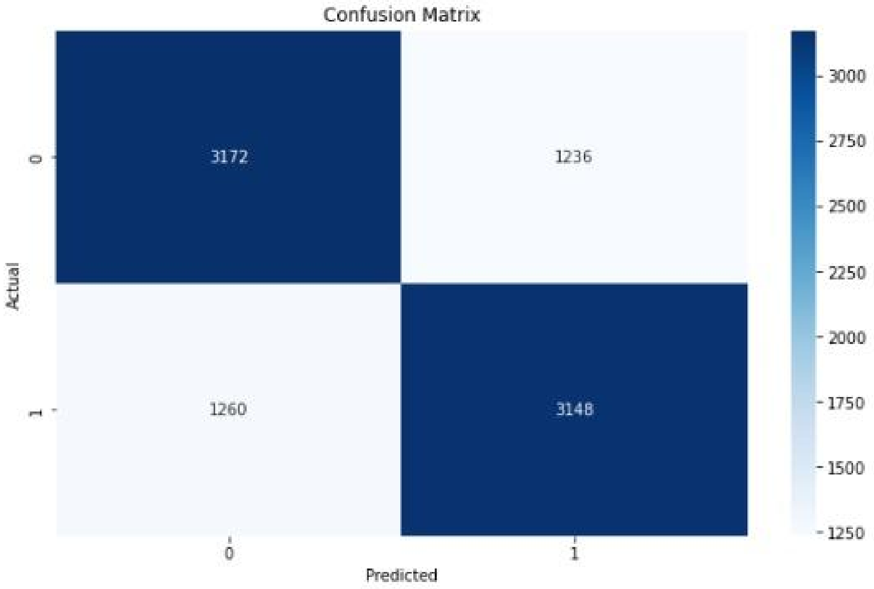
SUPPORT VECTOR MACHINES

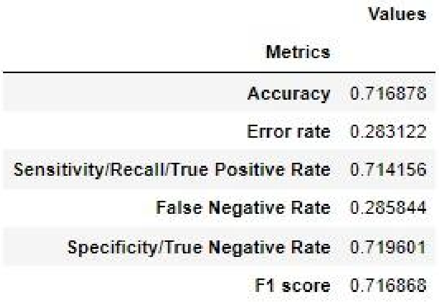
A well-liked supervised learning approach for classification and regression analysis is the support vector machine (SVM). By creating a decision boundary that maximizes the margins between the two categories, SVMs are used to sort data into one of two categories. When using a soft margin, the SVM algorithm permits some data points to be incorrectly classified in order to obtain a larger margin. SVMs are frequently employed in the one-vs-all (OVA) strategy for multi-class classification. Multiple binary SVM classifiers are trained as part of the OVA technique, and each one is taught to differentiate one class from all others. The class that has the highest rating from any of the binary classifiers is chosen as the predicted class during prediction.

SVMs are strong and adaptable machine learning algorithms that may be applied to a range of tasks, including fraud detection, natural language processing, and picture categorization. The specific problem and dataset at hand, as well as the required balance between accuracy and complexity, determine the SVM version and approach to be used.

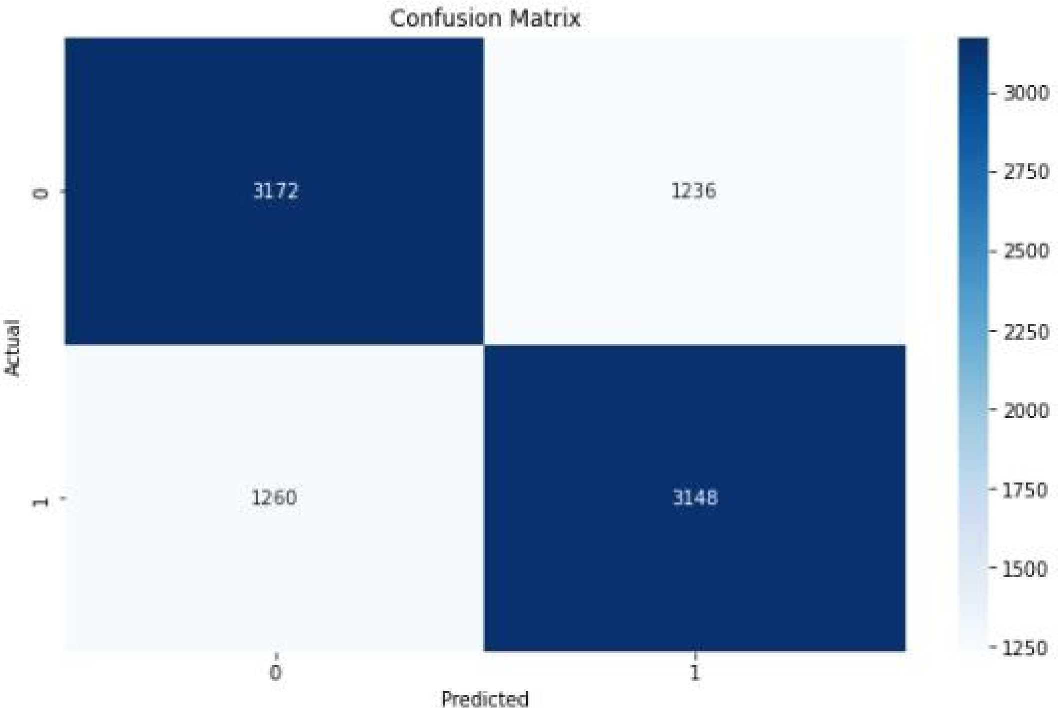
1) Using a hyperparameter C, the soft margin support vector machine (SVM) allows for the misclassification of some data points. The SVM transforms into a hard margin classifier, which seeks to accurately classify every data point, when C has a high value. However, if C is low, the SVM transforms into a soft margin classifier, which permits some misclassification. Alpha values can be maintained within a certain range by using complementary slackness, the fourth of the Karush-Kuhn-Tucker (KKT) requirements for SVM optimization, can be used to keep alpha values within a certain range. We can guarantee that alpha values remain inside a certain range and use the SVM successfully for classification tasks by meeting these requirements.

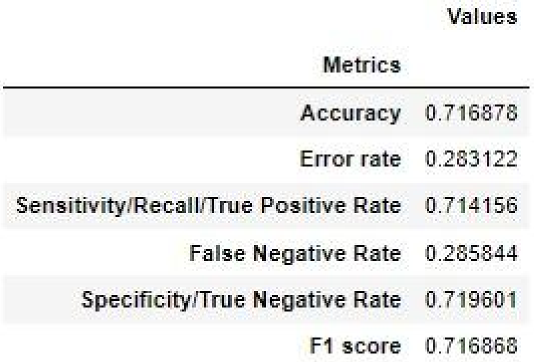
Results



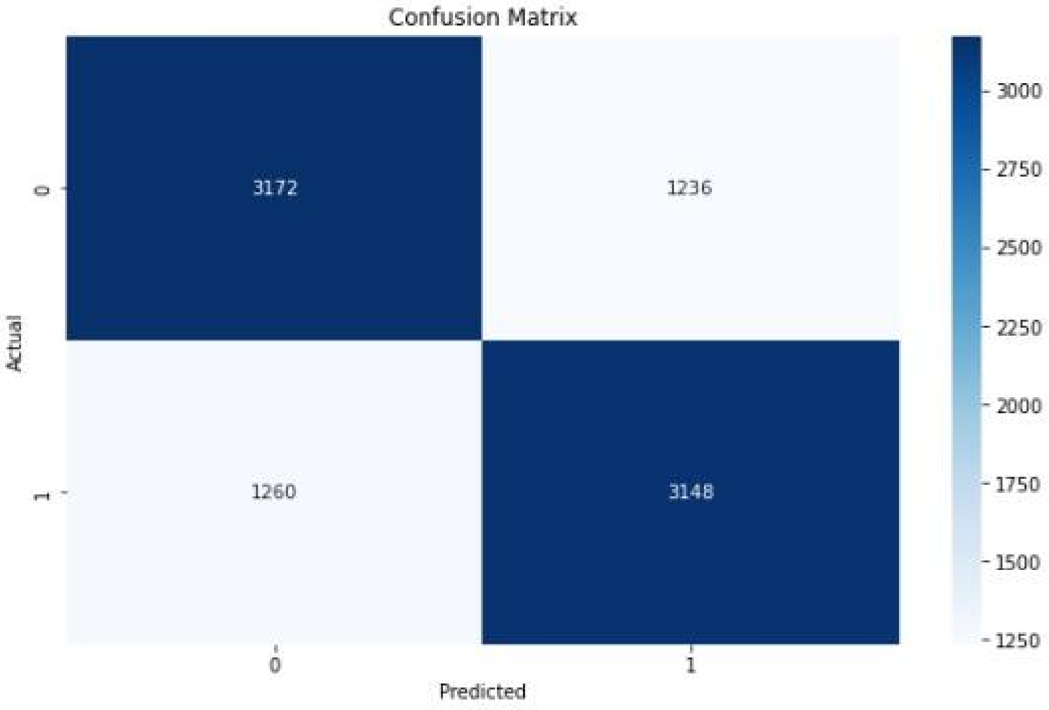


	Values
Metrics	
Accuracy	0.716878
Error rate	0.283122
Sensitivity/Recall/True Positive Rate	0.714156
False Negative Rate	0.285844
Specificity/True Negative Rate	0.719601
F1 score	0.716868





Metrics	Values
Accuracy	0.716878
Error rate	0.283122
Sensitivity/Recall/True Positive Rate	0.714156
False Negative Rate	0.285844
Specificity/True Negative Rate	0.719601
F1 score	0.716868



NEURAL NETWORKS: -

We looked at various methods for prioritizing speed while optimizing neural networks. In order to accomplish this, we have chosen to streamline matrix multiplication by employing small batches of size 64. The training and testing times are listed in table nn1.

1. In order to accelerate the convergence of gradient descent, we are also applying various optimization methods, such as Adam and RMSprop. To modify the learning rate, RMSprop combines gradient descent with momentum and exponentially weighted averages. Gradient descent optimization uses the momentum technique to hasten convergence. The gradient descent method benefits from moving more quickly in the important directions and less quickly in the unimportant ones. Making a weighted average of the prior gradients yields the momentum term.

2) In essence, this approach helps to make the update steps more consistent by using RMSprop to implement the same smoothing process.

3) According to the original research recommendation, we set the value of β to 0.99 and α as the learning rate as usual. To make the update slower in the vertical direction, we divide the update of b by a relatively larger number compared to W because the slope is larger in the b

direction. Sometimes, we also add a small value of ε in the denominator under the square root to ensure numerical stability in case the denominator is 0.

4. Adam Optimization (Adaptive moment estimation): Adam performs both momentum and RMSprop and combines them as follows. In a standard method such like Adam, a bias correction is applied prior to performing a gradient update. This algorithm, which combines gradient descent with momentum and gradient descent with RMSprop, is a commonly used approach for faster convergence in many types of neural networks. There are several hyperparameters in this algorithm. The most commonly used parameter values include:

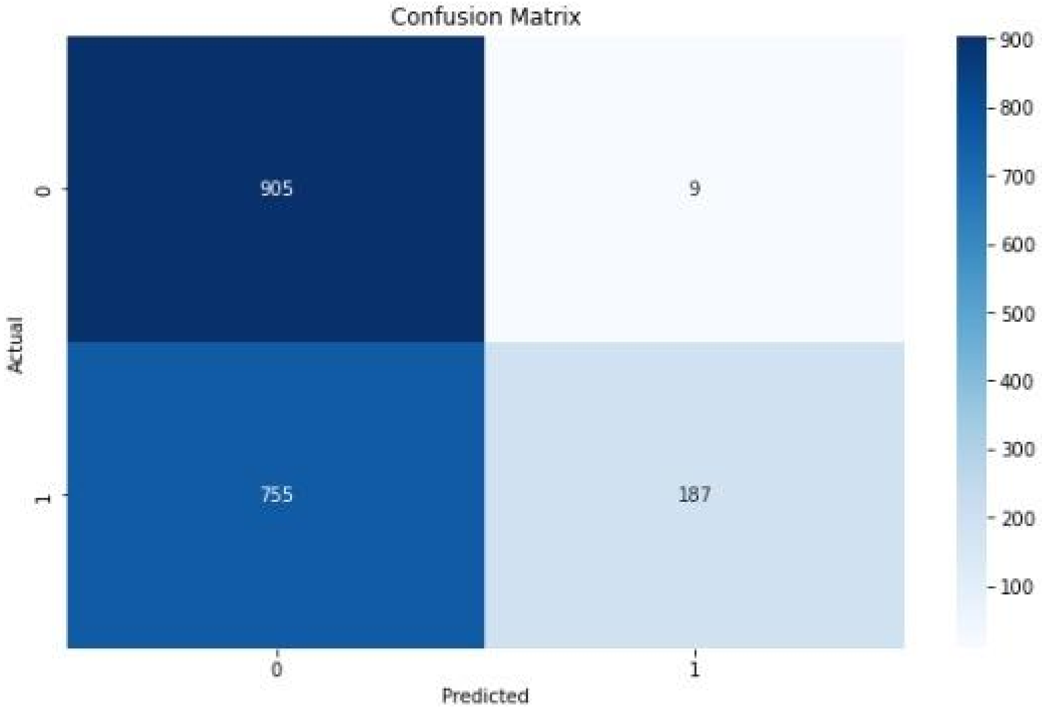
● α = needs to be tuned

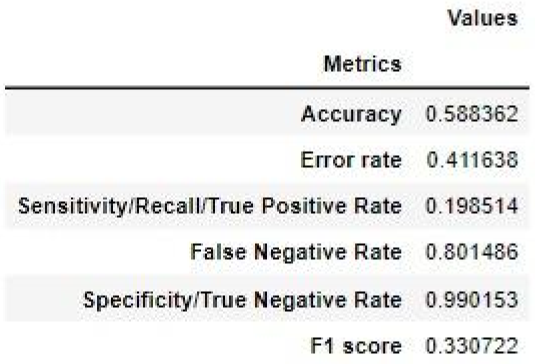
● β (Moving averages for dw, computes past 10 values averaged) 1 = 0. 9

●β (Moving averages of dw2) 2 = 0. 999

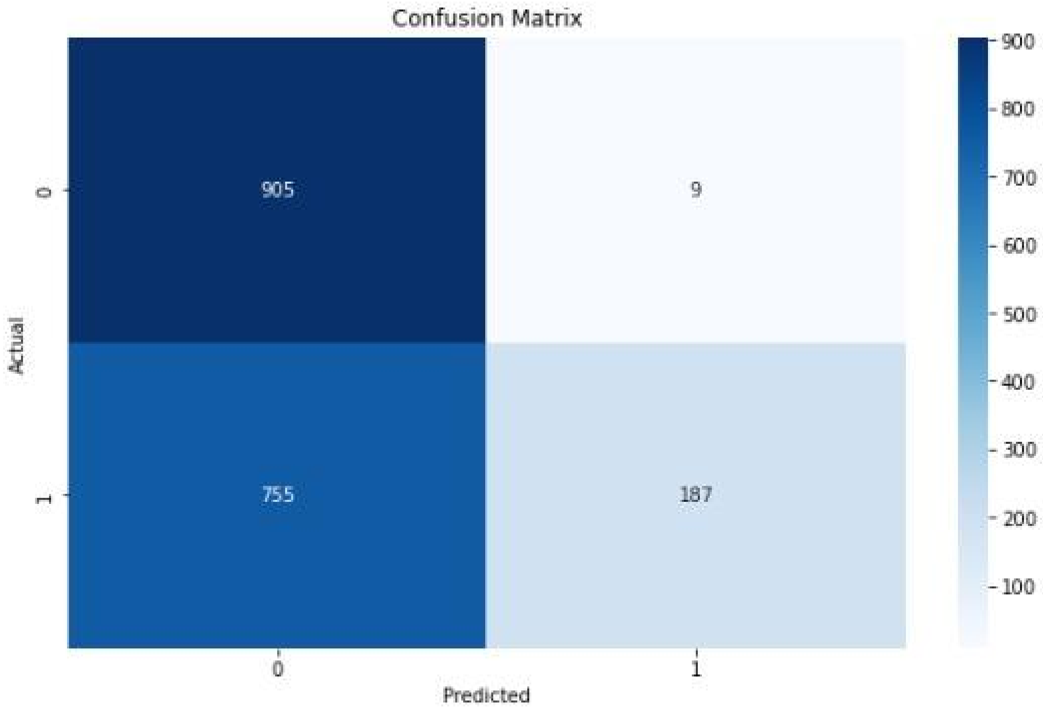
● ε = 10 (Usually not required to use)

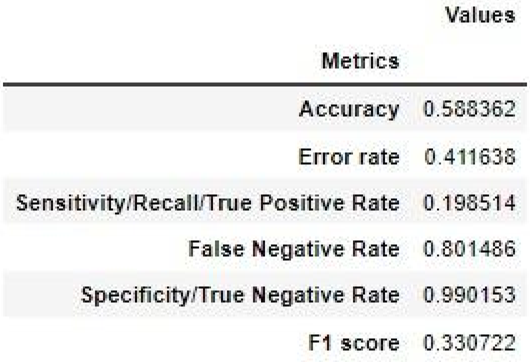
To accelerate the learning process of algorithms, one effective approach is to implement learning rate decay. This involves gradually decreasing the learning rate over time as the number of minibatches and iterations increase. The reasoning behind this method is that at the start of the learning process, taking larger steps can be beneficial, but as more iterations are completed, it becomes more advantageous to slow down the learning process to approach the optimal solution more accurately. Here, we have implemented neural networks from scratch using NumPy and optimized various parameters to achieve the maximum accuracy on the test results. In addition to the abovementioned hyperparameters and methods, we also utilized techniques to avoid getting stuck at the saddle point, which is one of the local optima during the learning process. To prevent this, we used Adam optimizer, which helps to slip off the saddle point during learning. We also finetuned the decay rate hyperparameter to optimize the learning rate decay, which gradually reduces the learning rate as the number of mini-batches and iterations increases. Overall, our approach helped to speed up the learning process and avoid getting stuck at local optima.



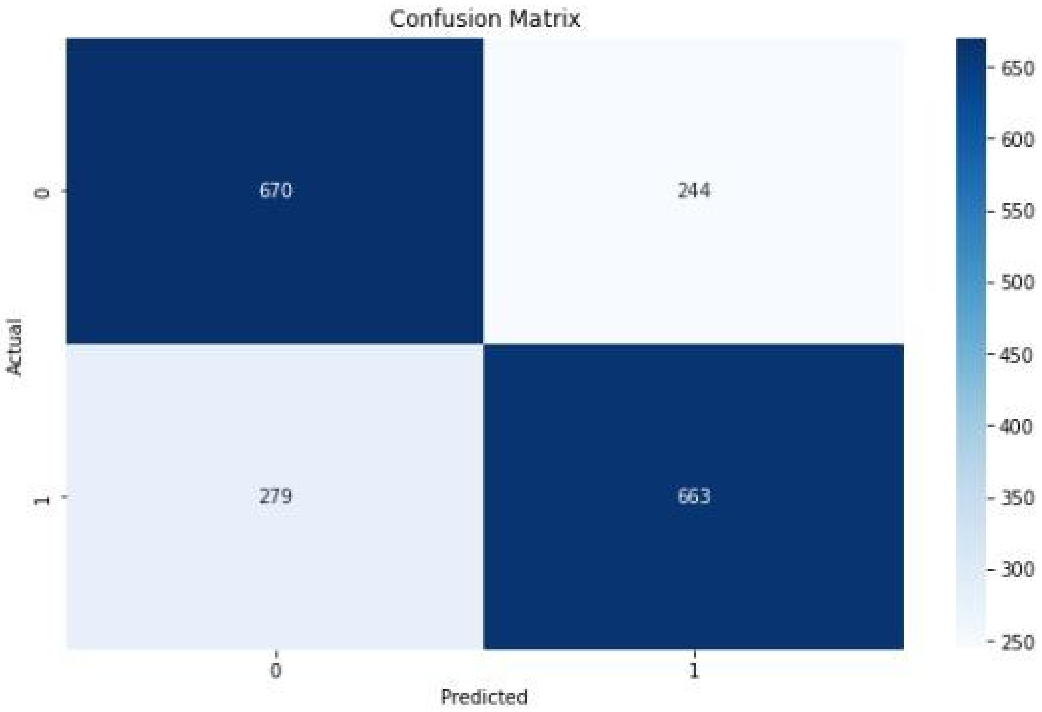


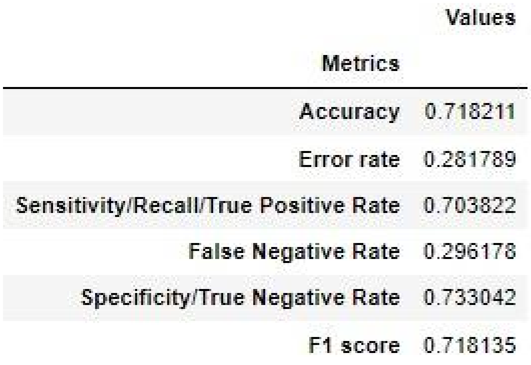
	Values
Metrics	
Accuracy	0.588362
Error rate	0.411638
Sensitivity/Recall/True Positive Rate	0.198514
False Negative Rate	0.801486
Specificity/True Negative Rate	0.990153
F1 score	0.330722



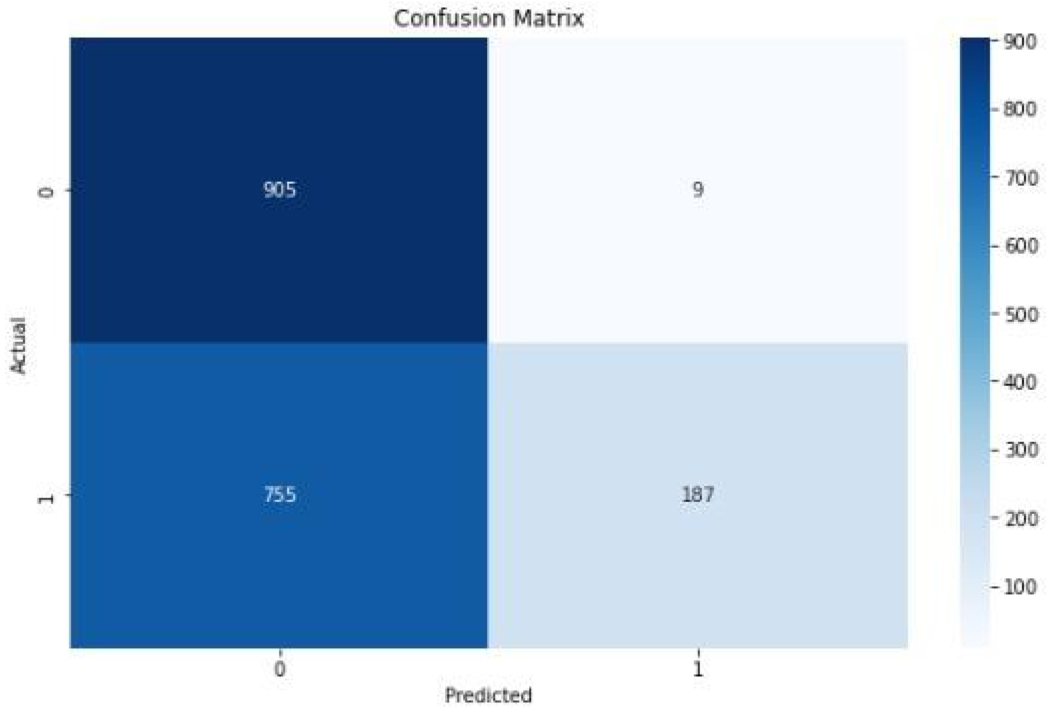


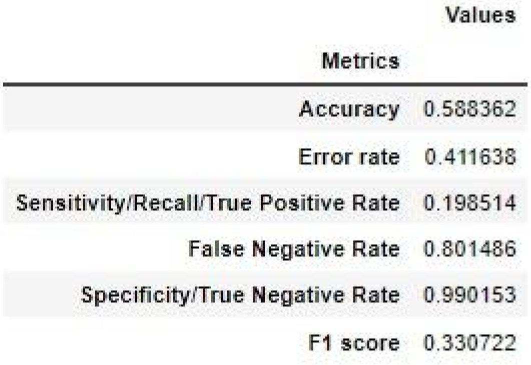
	Values
Metrics	
Accuracy	0.588362
Error rate	0.411638
Sensitivity/Recall/True Positive Rate	0.198514
False Negative Rate	0.801486
Specificity/True Negative Rate	0.990153
F1 score	0.330722



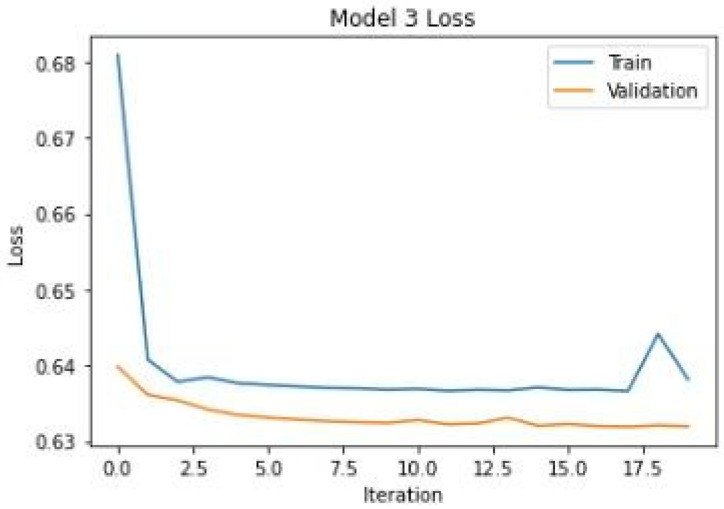


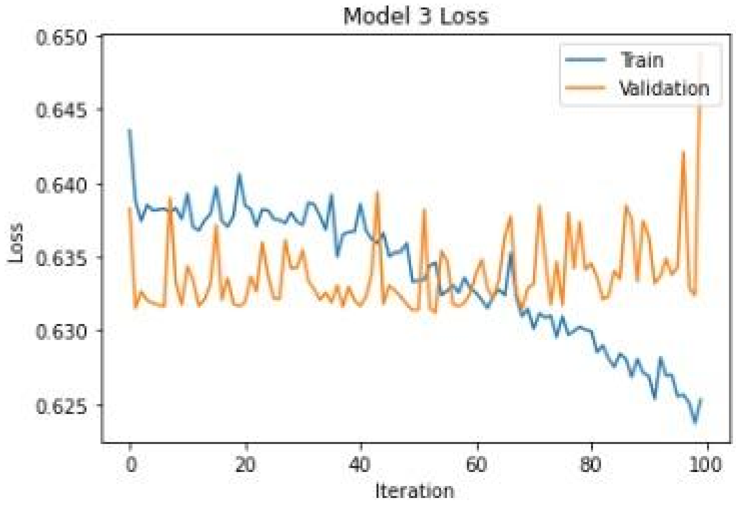
Metrics	Values
Accuracy	0.718211
Error rate	0.281789
Sensitivity/Recall/True Positive Rate	0.703822
False Negative Rate	0.296178
Specificity/True Negative Rate	0.733042
F1 score	0.718135

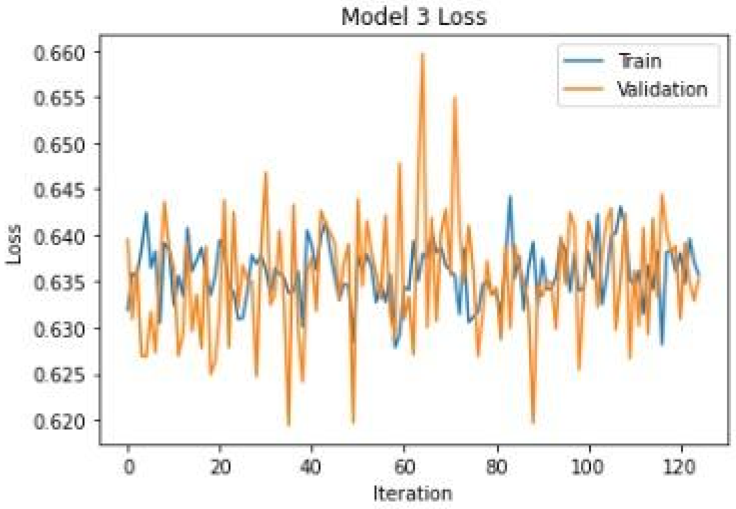




Metrics	Values
Accuracy	0.588362
Error rate	0.411638
Sensitivity/Recall/True Positive Rate	0.198514
False Negative Rate	0.801486
Specificity/True Negative Rate	0.990153
F1 score	0.330722


Model 3 Loss
Train
Validation
Loss
0.68
0.67
0.66
0.65
0.64
0.63
Iteration
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Model 3 Loss
Train
Validation
Loss
0.650
0.645
0.640
0.635
0.630
0.625
Iteration
0
20
40
60
80
100


Model 3 Loss
Train
Validation
Loss
0.660
0.655
0.650
0.645
0.640
0.635
0.630
0.625
0.620
Iteration
0
20
40
60
80
100
120

LOGISTIC REGRESSION

This is a popular classification algorithm that can be used for binary classification problems like this one. It is simple to understand and implement, and it works well when we can draw a linear line between the input variables w.r.to different class outputs.

Advantages:

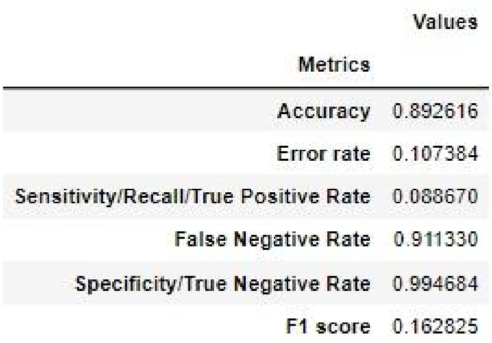
- It is simple and easy to implement.
- It can handle both binary and multiclass classification problems.
- It outputs probabilities that can be interpreted as the likelihood of a sample belonging to a particular class.
- It can be used as a baseline model to compare with other more complex models.

Disadvantages:

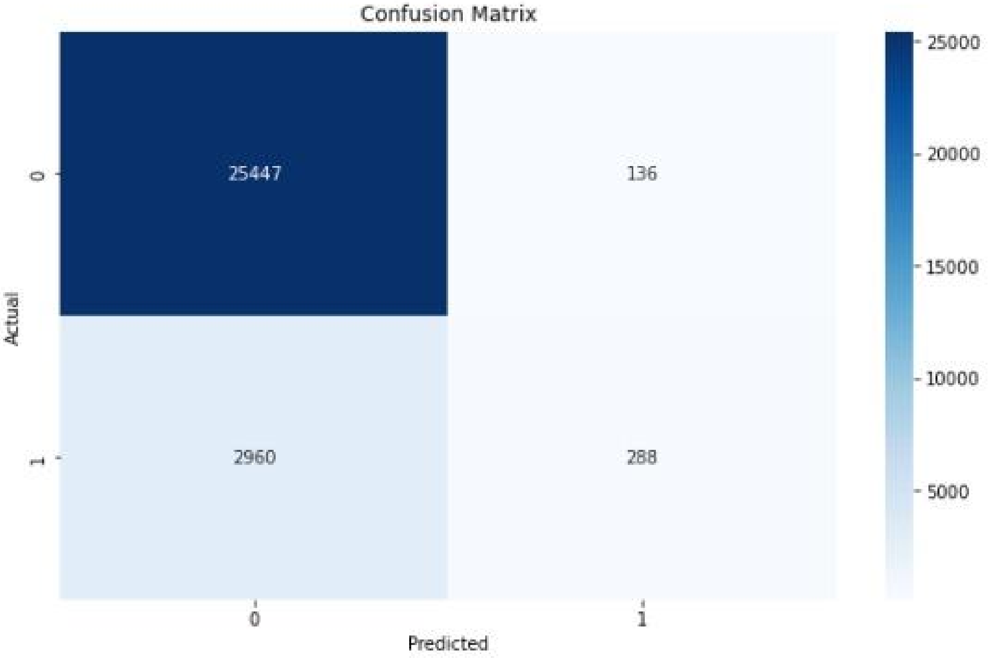
- It assumes a linear relationship between the independent variables and the log-odds of the dependent variable.
- It can be sensitive to outliers and multicollinearity.
- It does not perform well when the data is not linearly separable.

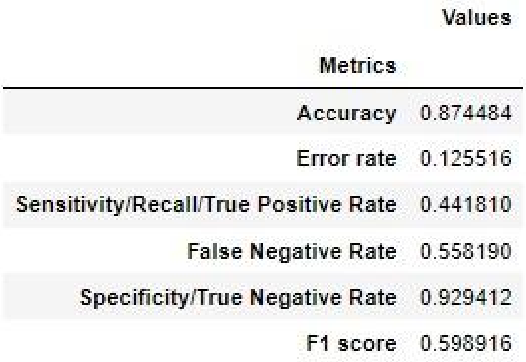
Quadratic Discriminant Analysis,

A machine learning algorithm used for classification tasks is quadratic discriminant analysis. It determines the parameters for each class using maximum likelihood estimation and makes the assumption that the input variables are normally distributed. A quadratic decision boundary is created by QDA to demarcate several classes in the feature space.

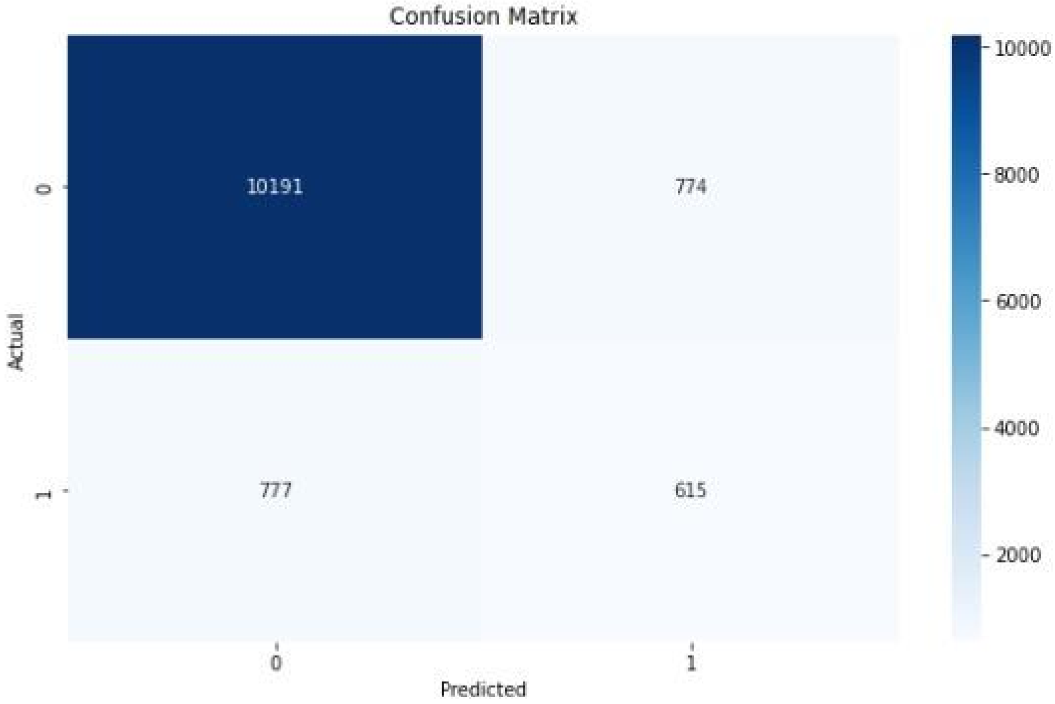


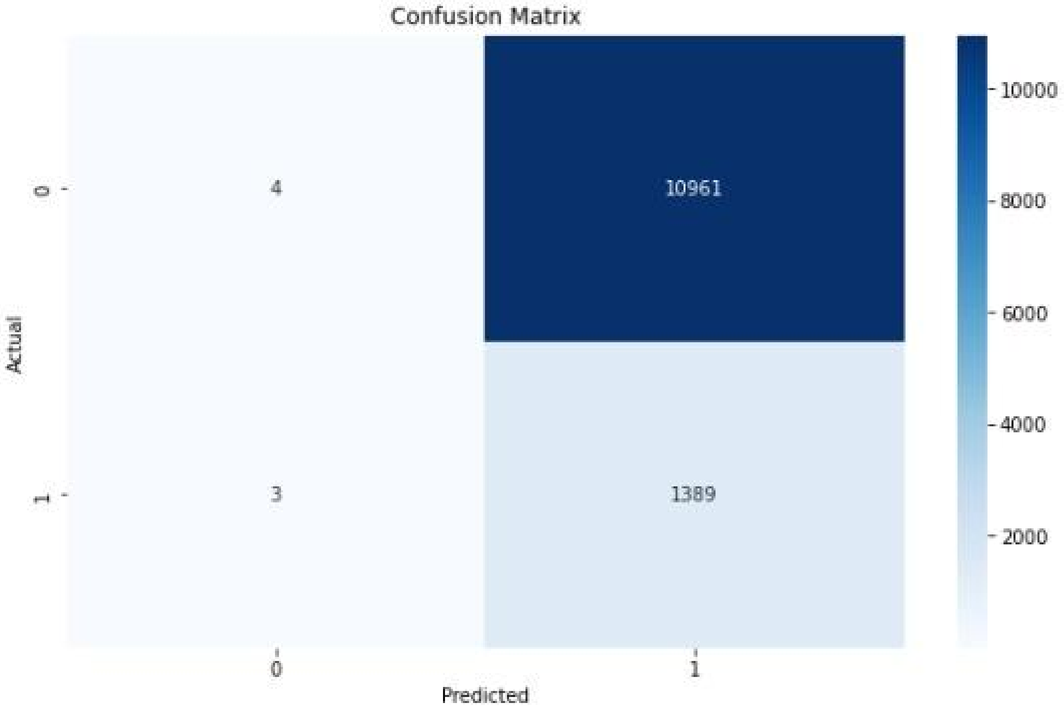
Metrics	Values
Accuracy	0.892616
Error rate	0.107384
Sensitivity/Recall/True Positive Rate	0.088670
False Negative Rate	0.911330
Specificity/True Negative Rate	0.994684
F1 score	0.162825

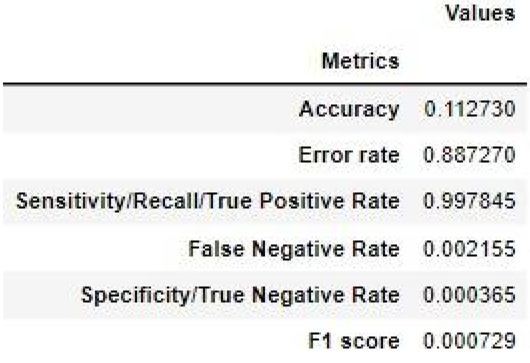




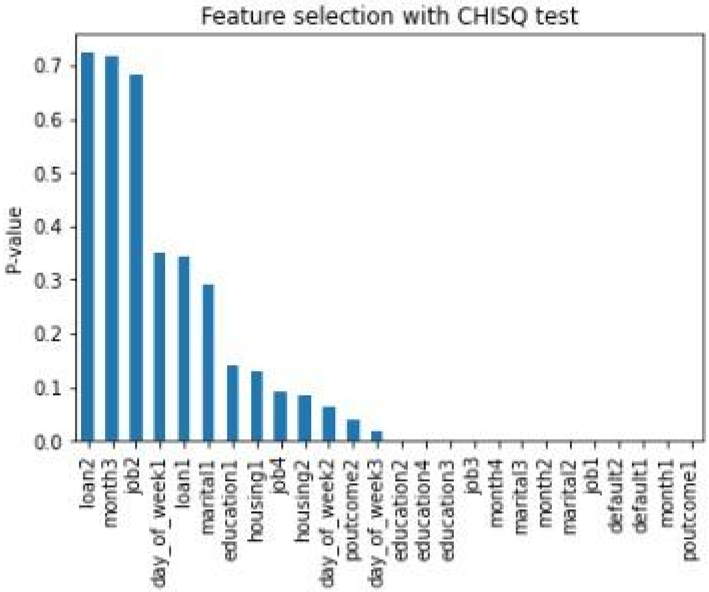
	Values
Metrics	
Accuracy	0.874484
Error rate	0.125516
Sensitivity/Recall/True Positive Rate	0.441810
False Negative Rate	0.558190
Specificity/True Negative Rate	0.929412
F1 score	0.598916

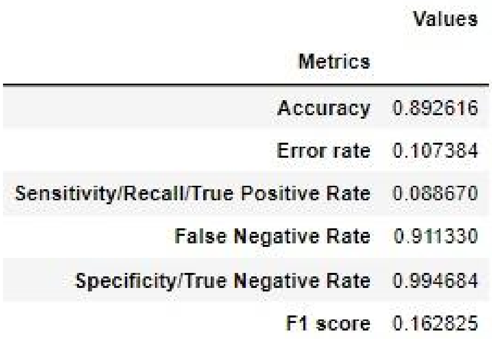




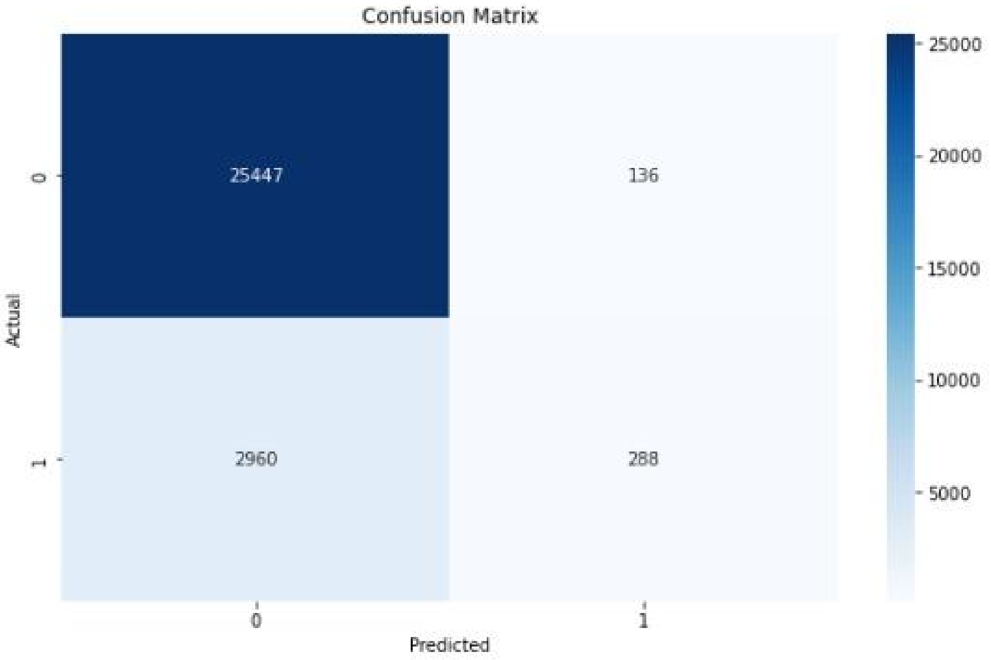


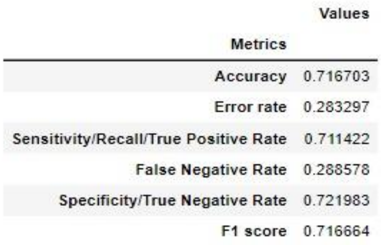
	Values
Metrics	
Accuracy	0.112730
Error rate	0.887270
Sensitivity/Recall/True Positive Rate	0.997845
False Negative Rate	0.002155
Specificity/True Negative Rate	0.000365
F1 score	0.000729



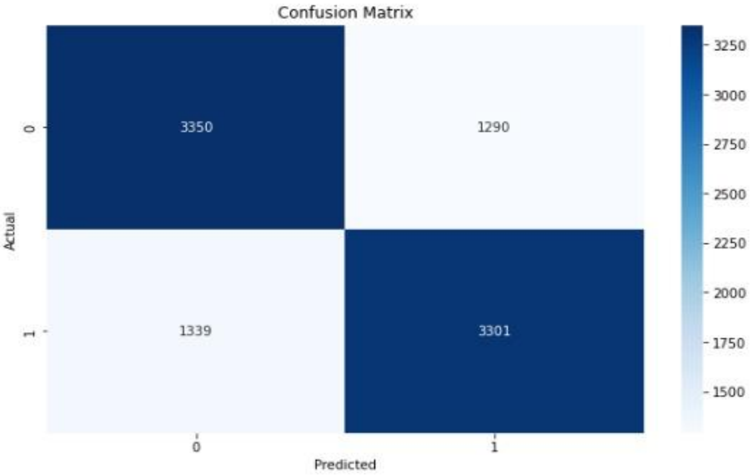


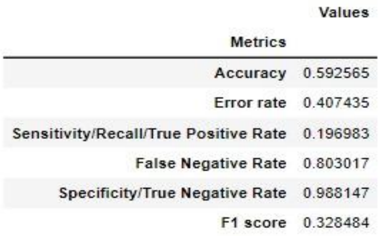
	Values
Metrics	
Accuracy	0.892616
Error rate	0.107384
Sensitivity/Recall/True Positive Rate	0.088670
False Negative Rate	0.911330
Specificity/True Negative Rate	0.994684
F1 score	0.162825



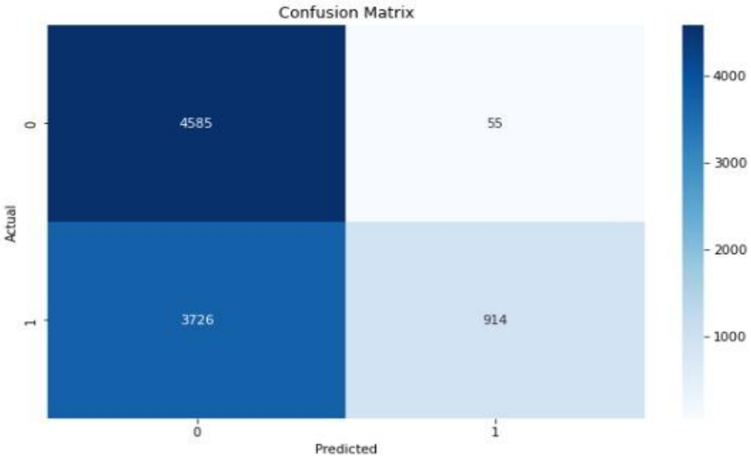


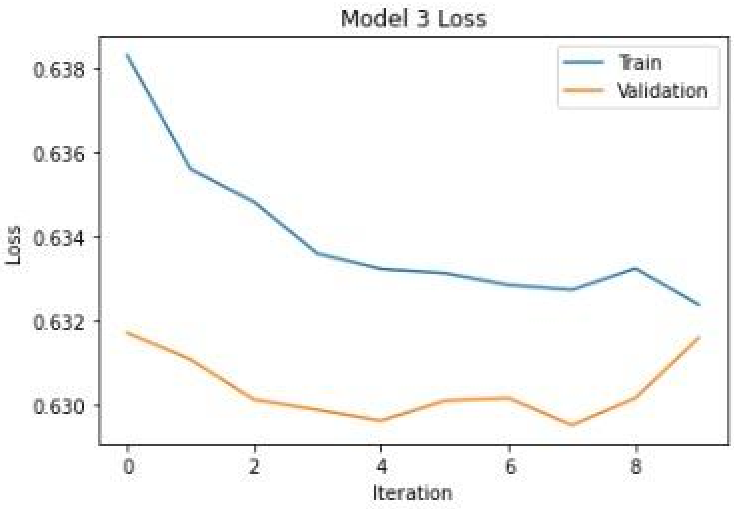
Metrics	Values
Accuracy	0.716703
Error rate	0.283297
Sensitivity/Recall/True Positive Rate	0.711422
False Negative Rate	0.288578
Specificity/True Negative Rate	0.721983
F1 score	0.716664

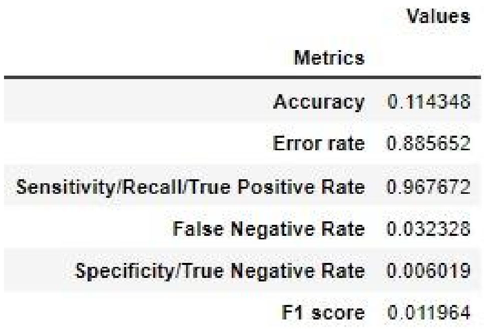




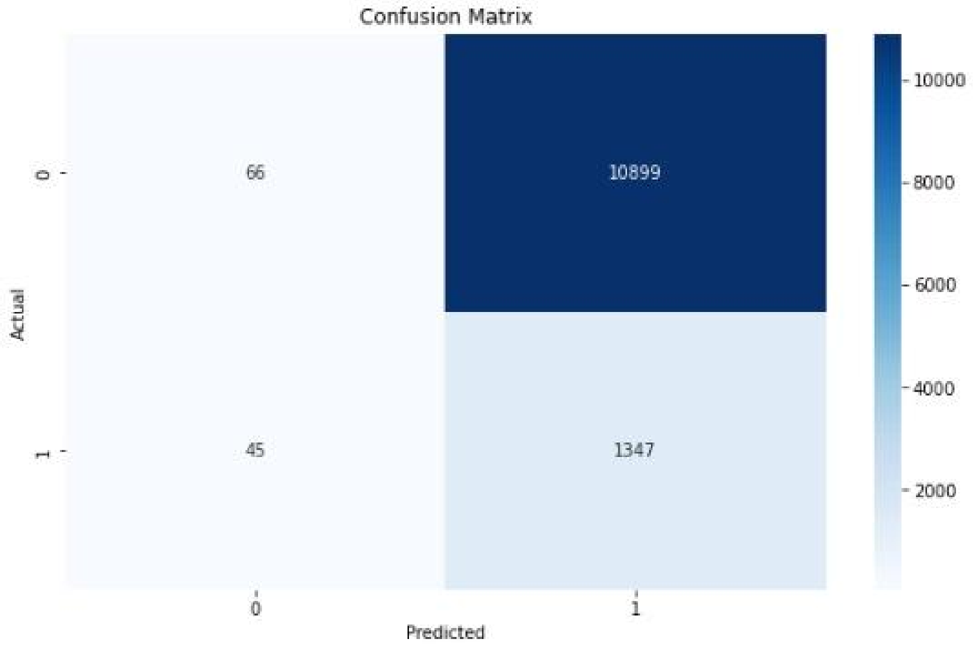
Metrics	Values
Accuracy	0.592565
Error rate	0.407435
Sensitivity/Recall/True Positive Rate	0.196983
False Negative Rate	0.803017
Specificity/True Negative Rate	0.988147
F1 score	0.328484





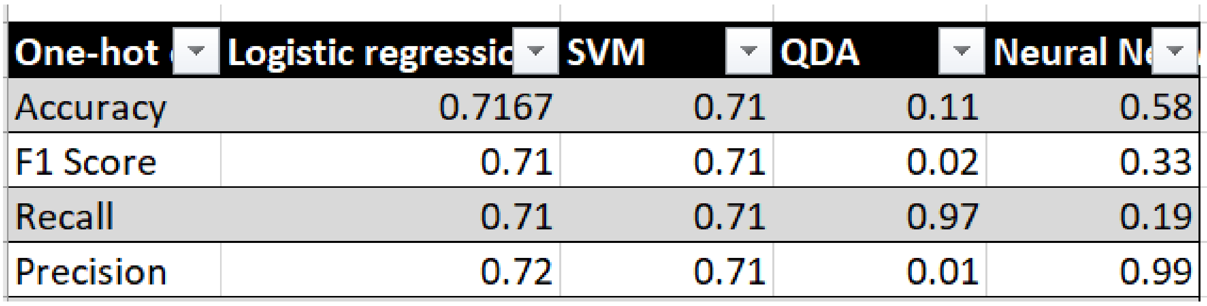


Metrics	Values
Accuracy	0.114348
Error rate	0.885652
Sensitivity/Recall/True Positive Rate	0.967672
False Negative Rate	0.032328
Specificity/True Negative Rate	0.006019
F1 score	0.011964

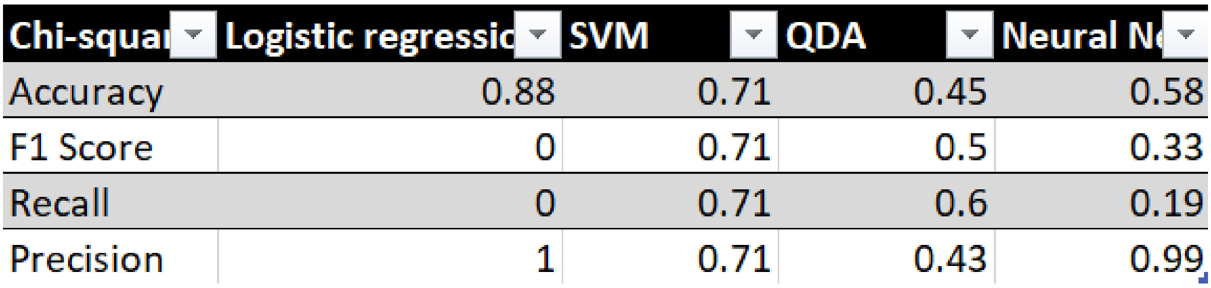


To select the best final model or models, cross-validation and performance metrics are used, such as accuracy, precision, recall, and F1 score. We will also consider the interpretability of the model, the computational complexity, and the ease of implementation. Based on the characteristics of the dataset and the candidate models, we will start with logistic regression and decision tree models and then try more complex models such as random forest, Naïve Bayes and Neural Network. After training and evaluating the models using cross-validation and performance metrics, we will select the best final model based on the overall performance and the trade-offs between interpretability and complexity. We will also consider using a combination of models to improve the performance and interpretability.

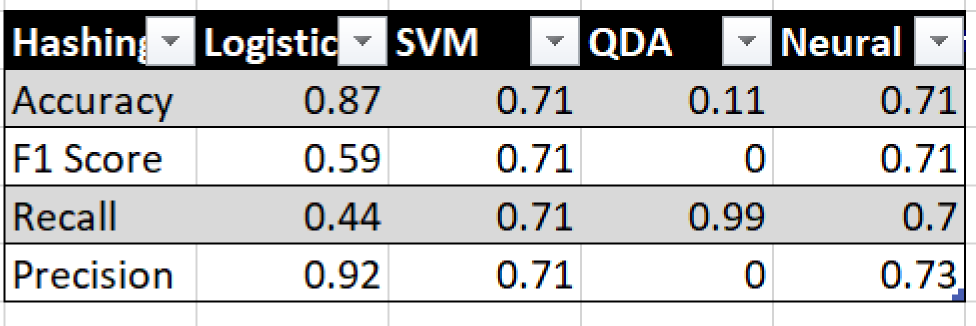
8. Model performance evaluation and Interpretation



One-hot	Logistic regressio	SVM	QDA	Neural N
Accuracy	0.7167	0.71	0.11	0.58
F1 Score	0.71	0.71	0.02	0.33
Recall	0.71	0.71	0.97	0.19
Precision	0.72	0.71	0.01	0.99



Chi-squar	Logistic regressio	SVM	QDA	Neural N
Accuracy	0.88	0.71	0.45	0.58
F1 Score	0	0.71	0.5	0.33
Recall	0	0.71	0.6	0.19
Precision	1	0.71	0.43	0.99



Hashin	Logistic	SVM	QDA	Neural
Accuracy	0.87	0.71	0.11	0.71
F1 Score	0.59	0.71	0	0.71
Recall	0.44	0.71	0.99	0.7
Precision	0.92	0.71	0	0.73

9. Conclusion

The results table shows that the F1 scores of the models on the hashing data are higher than the F1 scores of the models on the one-hot encoded data. This suggests that hashing the categorical variables is a better way to represent them for machine learning models.

There are a few reasons why this might be the case. First, hashing can reduce the dimensionality of the data, which can make it easier for the models to learn. Second, hashing can help to efficiently preserve the relationships between the categorical variables, which can also improve the performance of the models.

In terms of memory optimization and computational efficiency, hashing is also a better choice than one-hot encoding. Hashing can significantly reduce the size of the data, which can save memory and improve the speed of the models.

And overall SVM is consistent with results on all different types of data generated from our preprocessing like One hot encoding, PCA, Hashing but it also consumes lots of computational resources, but it is still worth it as the testing is cheap for SVM,
Additionally interms of discerning the non linearity in data Neural networks and logiastic does a great job.