

Introduction To Machine Learning

Project: Credit Card Fraud Detection

- Link for google colab file:-

<https://colab.research.google.com/drive/1vanBkhhb0d3GU5OkeqiVOIUETMDX7V68q?usp=sharing>

❖ Data preprocessing:-

Data Loading and Inspection:-

The code begins by loading a dataset from a CSV file named "WA_Fn-UseC_-Telco-Customer-Churn.csv" into a Pandas DataFrame called data.

- The initial step involves loading the dataset to understand its structure and characteristics.
- By displaying the first few rows of the dataset (data.head()), we get a glimpse of the data, including the column names and some sample records.

Data Cleaning and Handling Missing Values:-

- Missing values are checked for in the dataset, and the counts for each column are displayed.
- The 'customerID' column, which likely serves as an identifier, is removed, possibly as it does not contribute to the analysis.

Normalization:-

First, I divide the dataset into categorical and numerical data. Next, I normalize them using the following methods.

Here are the columns for categorical and numerical data:-

Ordinal Categorical data columns:-

['Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'Churn', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

Nominal Categorical Data:-

['gender', 'InternetService', 'Contract', 'PaymentMethod']

Numerical Data:-

['tenure', 'MonthlyCharges', 'TotalCharges']

- **Label Encoding for Ordinal Categorical Data:-**

Ordinal categorical columns are transformed using label encoding. This is particularly useful when there is an inherent order in the categories, converting them into numerical representations while preserving their ordinal relationships.

- **One-Hot Encoding for Nominal Categorical Data:-**

Nominal categorical columns are one-hot encoded, expanding the feature space to represent each category as a binary column. This transformation ensures that the model does not incorrectly interpret ordinal relationships in nominal data.

- **Normalization of Numerical Data:-**

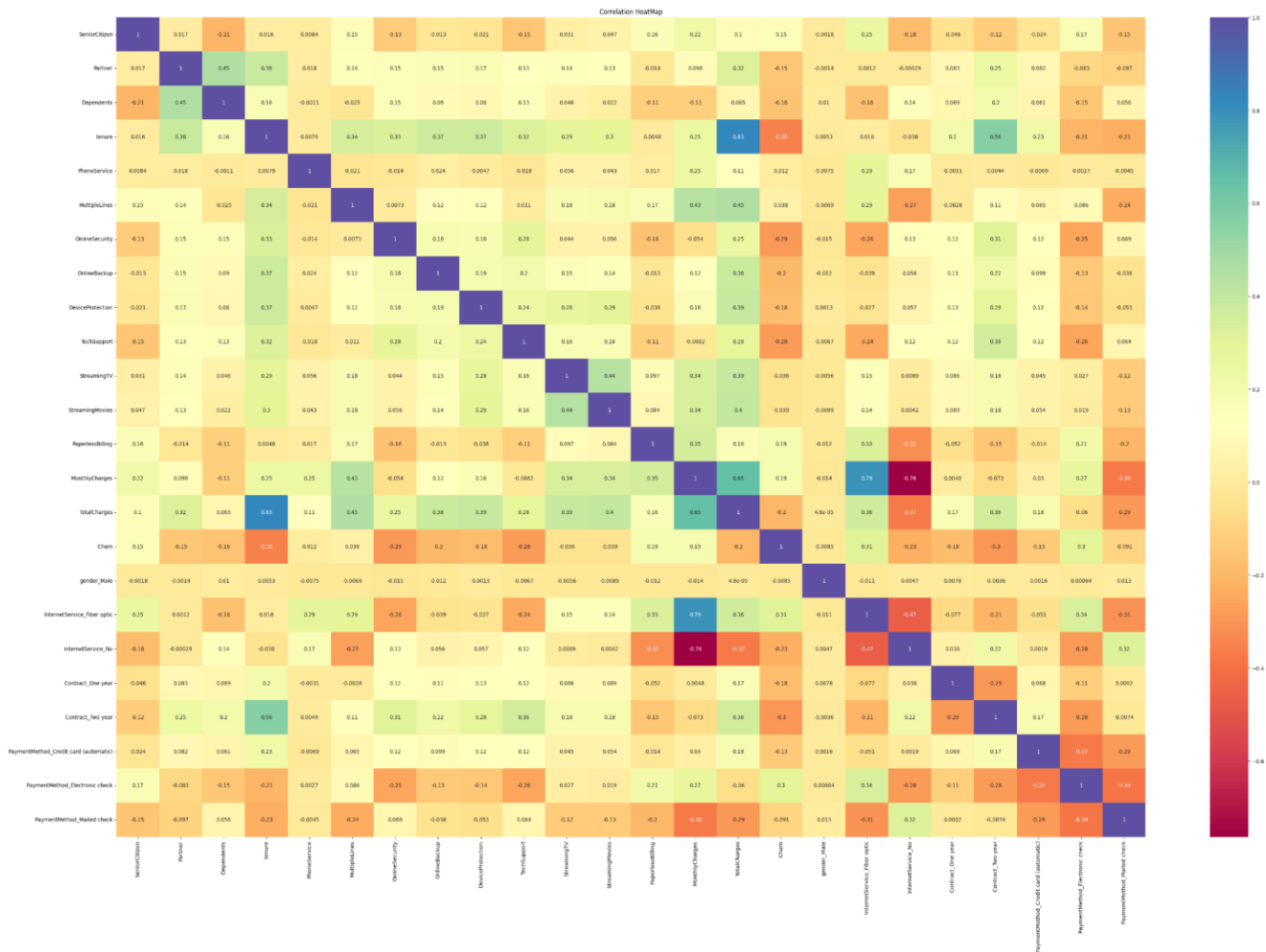
Numerical features are normalized using z-score normalization. This process standardizes the scale of the features, preventing certain features with larger magnitudes from dominating the learning process in machine learning algorithms.

❖ **Exploratory data analysis (EDA):-**

- A correlation heatmap is generated to visually inspect the relationships between different features. This aids in identifying potential patterns or dependencies in the data.
- The second-largest correlation coefficient with the 'Churn' column is identified. This coefficient provides insights into the strength and direction of the correlation between features and the target variable

Result:-

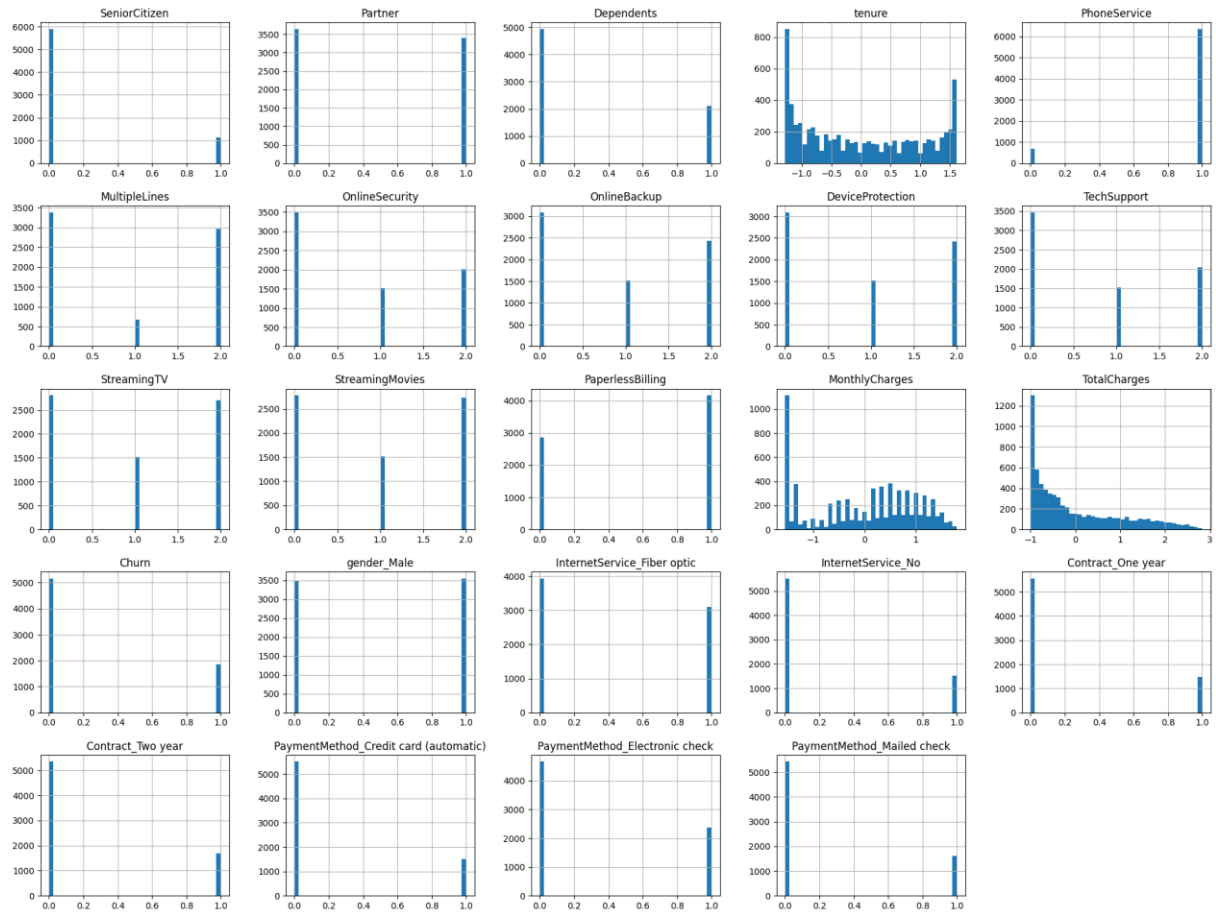
Second-largest correlation coefficient with 'Churn': 0.3540493589532519
Feature associated with the second-largest correlation coefficient: tenure



→ Histograms and Pairwise Scatterplots:-

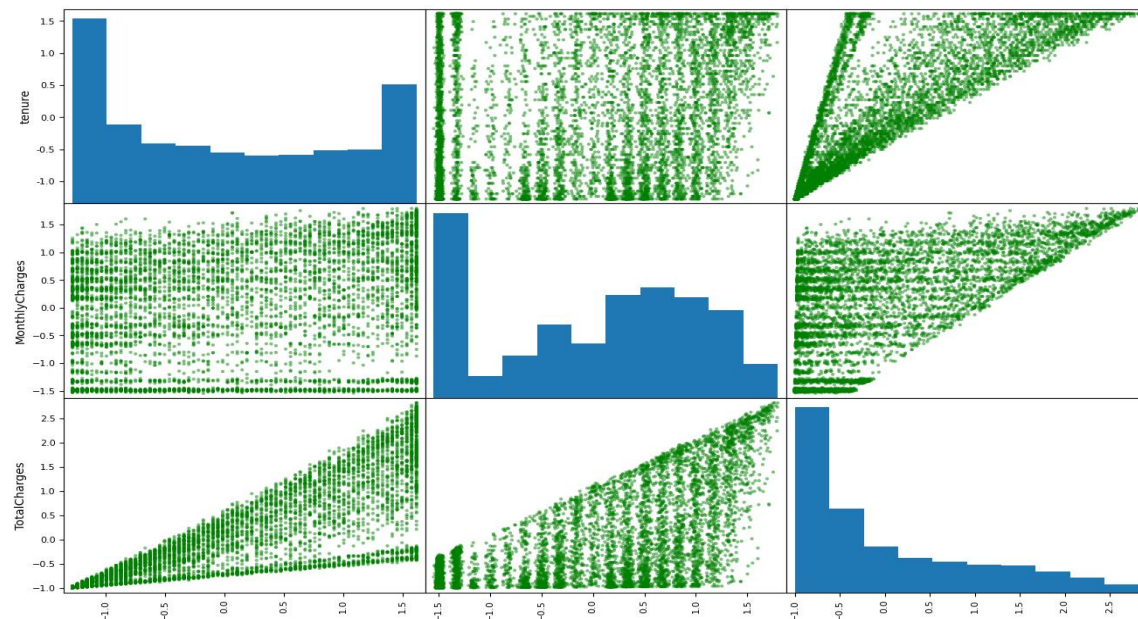
- Histograms are plotted to visualize the distribution of each numerical feature in the dataset, providing insights into the data's overall shape.
- Pairwise scatterplots offer a visual examination of the relationships between pairs of numerical features, helping to identify potential correlations or trends.

Result:-1. Histograms plot



2. Pairwise scatterplots for numerical features

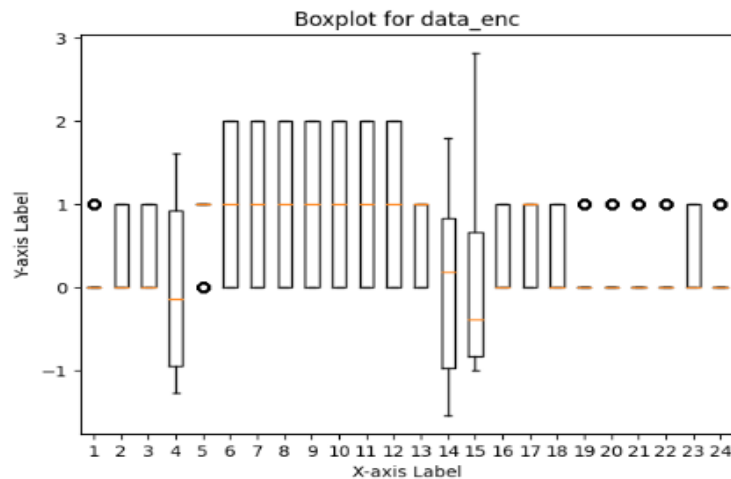
Pairwise Scatterplots of Numerical Features



→ Boxplot Visualization:-

- A boxplot is created to visualize the distribution of each feature in the dataset. Boxplots are effective in highlighting the central tendency, spread, and potential outliers in the data.

Result:- Boxplot



→ Train-Test Split:-

- The dataset is split into training and testing sets. This step is crucial for evaluating the model's performance on unseen data, helping to assess its generalization capability. The test size = 0.2

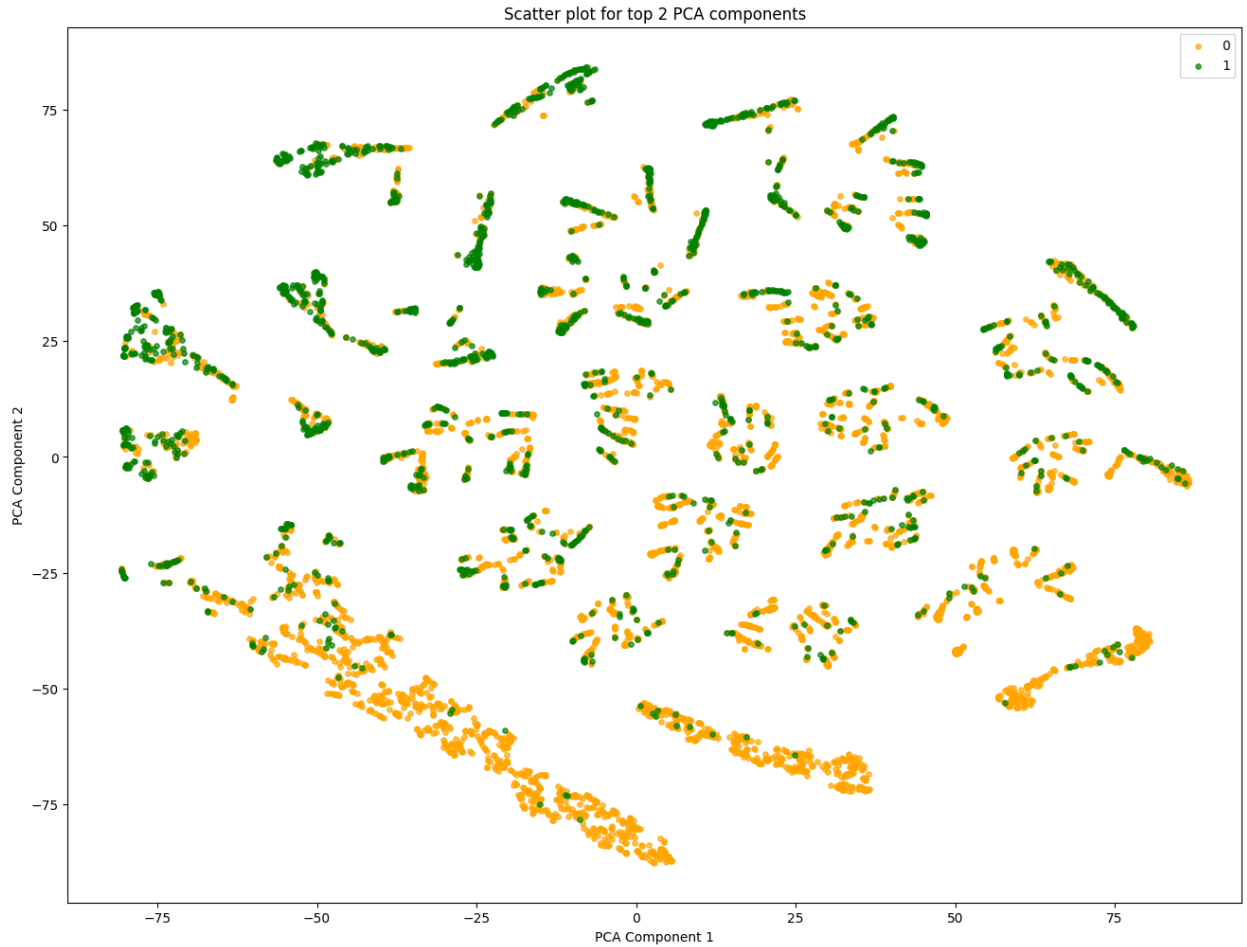
❖ Implementing PCA:-

Dimensionality Reduction with PCA and t-SNE:-

- Principal Component Analysis (PCA) is employed to reduce the dimensionality of the dataset to 8 components. This is especially useful when dealing with a large number of features.
- t-Distributed Stochastic Neighbor Embedding (t-SNE) further reduces the dimensionality to 2 components for visualization purposes. This technique is beneficial for capturing non-linear relationships between features.

Scatter Plot for Top 2 PCA Components:-

- The scatter plot visually represents the dataset in the reduced feature space, providing insights into potential clusters or patterns in the data.



★ Model Implementation:-

Implementing Logistic Regression:-

- Logistic regression is chosen as the model, which is well-suited for binary classification problems.
- The target variable, 'Churn', suggests a classification task, where the goal is to predict whether a customer will churn (1) or not (0).

Model Training and Prediction:-

- The logistic regression model is trained on the training set (X_{train} and y_{train}) using the fit method.
- Predictions are made on both the training and testing sets to evaluate the model's performance on seen and unseen data.

Confusion Matrix Analysis:-

- The confusion matrix provides a detailed breakdown of the model's predictions, including true positives, true negatives, false positives, and false negatives.
- The emphasis is placed on understanding Type-I (False Positives) and Type-II (False Negatives) errors, especially in the context of predicting customer churn.
- Plot the confusion matrix as a heatmap

Here explanation for Type-1 and Type-II Error:-

Type-I Error or False Positives: False Positives are the ones which are actually not fraud but the prediction said that they are fraud.

Type-II Error or False Negatives: False Negatives are the ones which are actually fraud but the system said that they aren't.

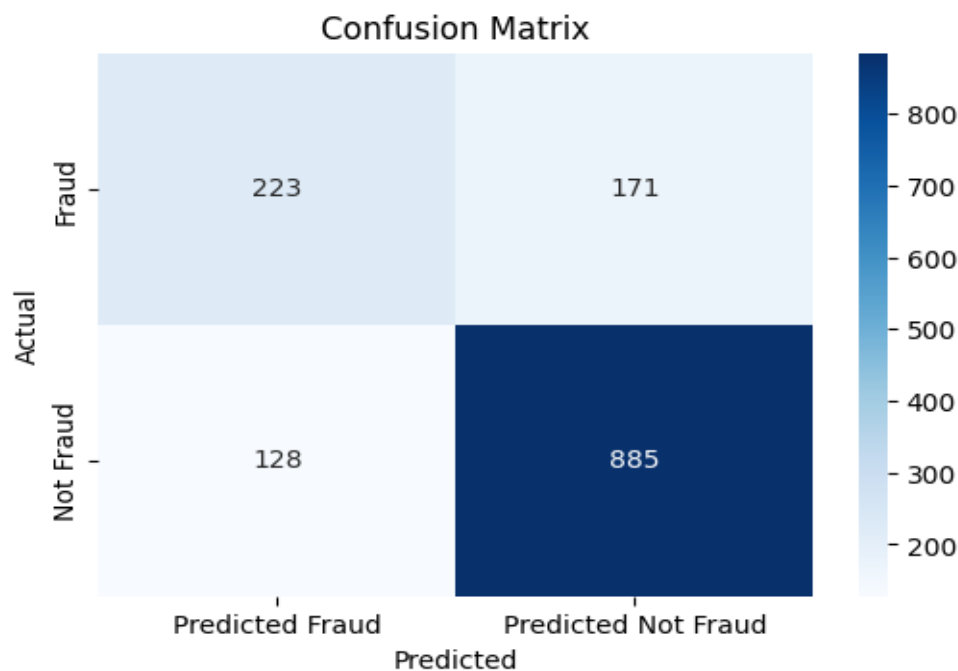
Well, we can say that Type-II Error is more significant because we don't want system to have a fraudulent credit card because that can be more dangerous.

So, for Type-II Error, We can say that recall is the important metric.

Result:-

	Predicted Fraud	Predicted Not Fraud
Fraud	223	171
Not Fraud	128	885

Confusion Matrix as a heatmap:-



Metric Selection and Importance:-

- Precision, recall, and the F1-score are chosen as evaluation metrics.
- Recall (sensitivity) is particularly highlighted due to its importance in the context of customer churn prediction. A high recall is desired to minimize false negatives (missed instances of churn).

Result:-logistic_accuracy is 0.7874911158493249
logistic_precision is 0.6353276353276354
logistic_recall_test is 0.565989847715736
logistic_recall_train is 0.5525423728813559
logistic_f1 is 0.5986577181208054

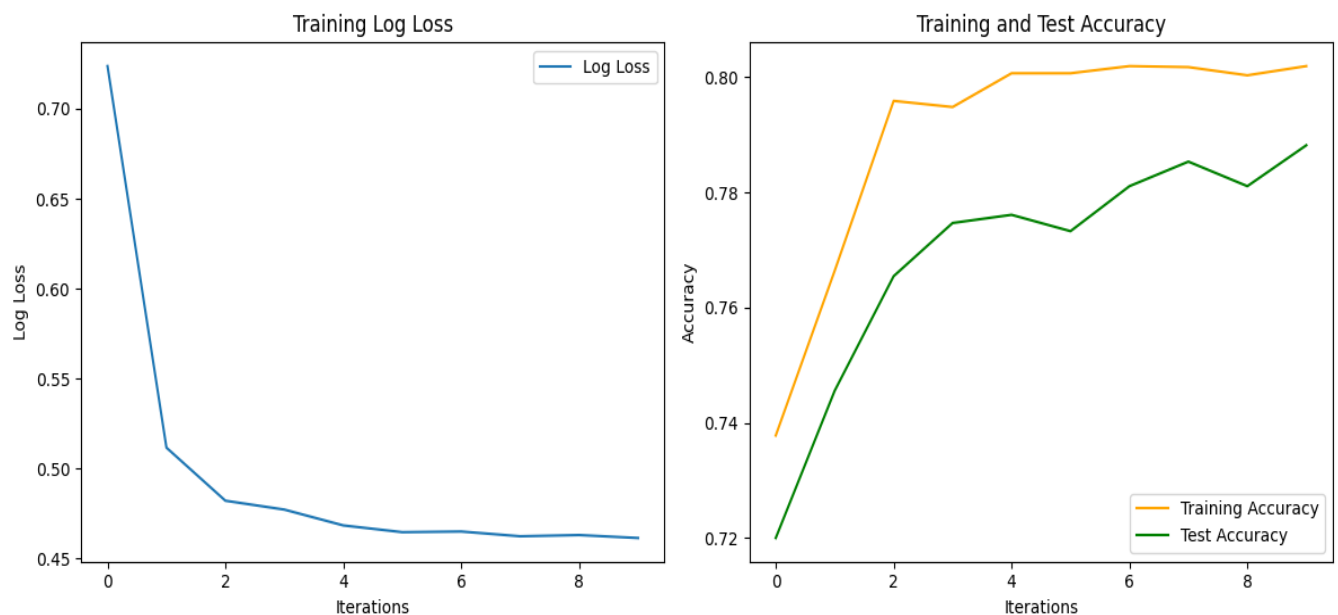
Classification Report:-

	precision	recall	f1-score	support
0	0.84	0.87	0.86	1013
1	0.64	0.57	0.60	394
accuracy			0.79	1407
macro avg	0.74	0.72	0.73	1407
weighted avg	0.78	0.79	0.78	1407

Loss and Accuracy Curves:

- The iterative training approach with partial fit is employed to observe how the model's performance evolves over multiple iterations.
- Loss values (log loss) and accuracy scores are tracked to visualize the learning process and convergence.

Result:-

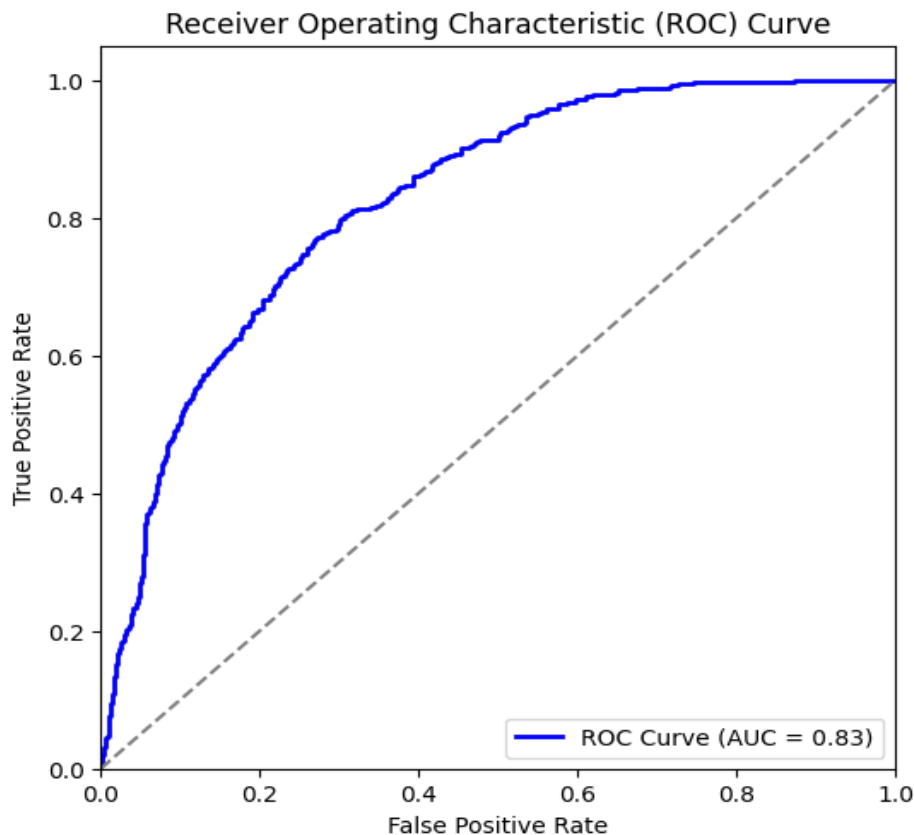


ROC Curve and AUC Score:-

- The ROC curve visually represents the trade-off between true positive rate (sensitivity) and false positive rate.
- The AUC score summarizes the overall performance of the model in distinguishing between positive and negative instances.

Result:-

```
logistic_auc score is 0.8259955602547592
```



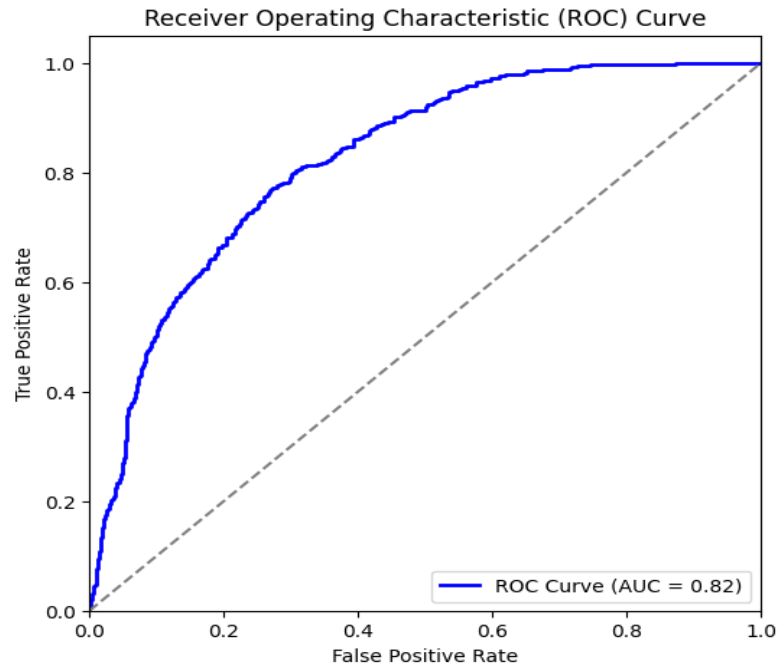
Logistic regression on 2nd degree polynomials:-

Model Complexity Experimentation:-

- The code explores the impact of increasing model complexity by introducing polynomial features of degree 2.
- The ROC curve and AUC score are recalculated, and the results are compared with the original model to assess the effect of increased complexity.

Result:-

```
roc_auc_score(y_train, train_prob) is 0.8711715744333265
roc_auc_score(y_test, test_prob) is 0.8177173896703264
logistic_auc_comp is 0.8177173896703264
recall_score for complex model is 0.5380710659898477
```



Visualization of Evaluation Metrics:-

- Evaluation metrics such as accuracy, precision, recall, and F1-score are visualized using appropriate plots and tables.
- This visual representation enhances the interpretability of the model's performance.

★ Implementation Linear SVM model:-

Introduction:-

- The code utilizes Support Vector Machines (SVMs), a powerful class of machine learning algorithms, for predictive modeling on a given dataset.
- The dataset is assumed to have features and labels where the objective is to train models that can accurately classify instances.

Linear SVM Model:-

- A linear SVM is employed initially, aiming to create a linear decision boundary to separate classes in the data.
- Grid search is utilized to fine-tune the model by identifying the optimal regularization parameter 'C'.
- The best parameters are then used to train a new linear SVM model.

Result:- best parameter : { 'C' : 0.1 }

Evaluation of Linear SVM:-

- The performance of the linear SVM is evaluated using various metrics:
 - **ROC Curve and AUC Score:** The ROC curve visualizes the trade-off between true positive rate and false positive rate, and the AUC score quantifies the overall model performance.
 - **Recall:** The recall metric is essential, especially in scenarios where false negatives (missed positive cases) are critical.
 - **Confusion Matrix:** This matrix provides a detailed breakdown of the model's predictions, highlighting true positives, true negatives, false positives, and false negatives.

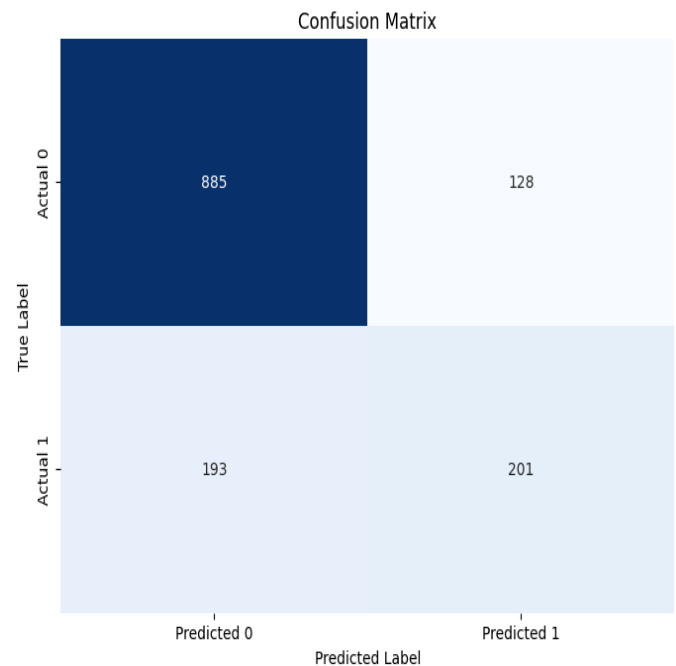
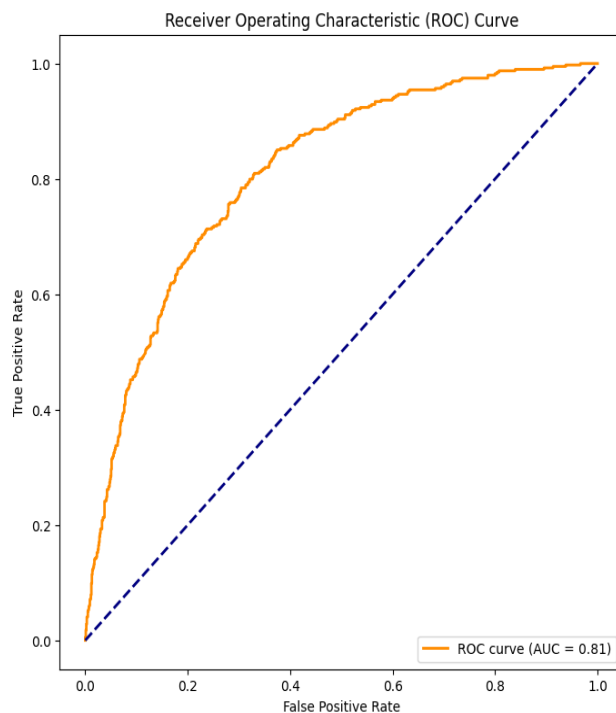
Result:- Best Hyperparameters: {'C': 0.1}

AUC: 0.8104

Recall: 0.5102

Confusion Matrix:

```
[[885 128]
 [193 201]]
```



SVM with Polynomial Kernel:-

- A more complex SVM model with a polynomial kernel is introduced to capture non-linear relationships within the data.
- Grid search is employed again to identify the optimal values for 'C' and the degree of the polynomial.

Result:- best parameter: {'C': 0.1, 'degree': 2}

Evaluation of Polynomial SVM:-

- Similar evaluation metrics are applied to assess the performance of the polynomial SVM.
- Visualization tools, including ROC curves and confusion matrices, aid in understanding how well the model classifies instances.

Result:-Best Hyperparameters: {'C': 0.1, 'degree': 2}

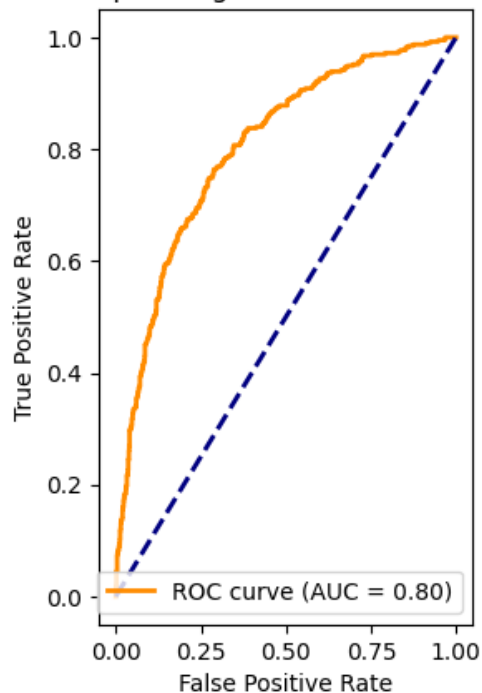
AUC: 0.8026

Recall: 0.4264

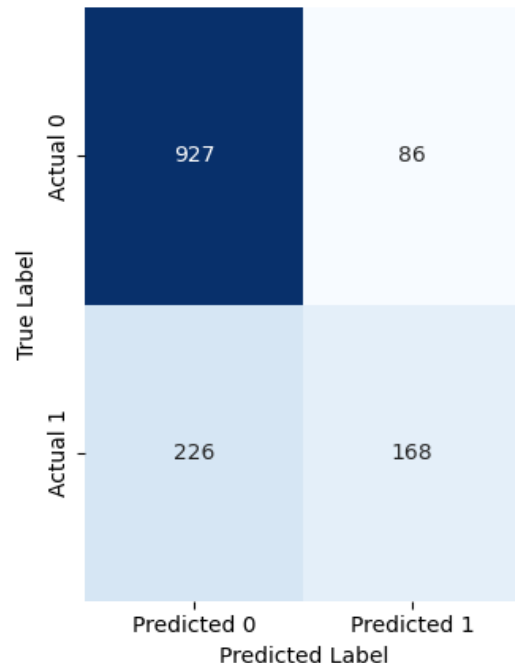
Confusion Matrix:

```
[[927  86]
 [226 168]]
```

Receiver Operating Characteristic (ROC) Curve



Confusion Matrix



SVM with RBF Kernel:-

- The third SVM model is constructed with a Radial Basis Function (RBF) kernel, enabling the model to capture intricate, non-linear decision boundaries.
- Grid search identifies optimal 'C' and 'gamma' values to fine-tune the RBF SVM.

Result:- best parameters : {'C': 10, 'gamma': 0.001}

Evaluation of RBF SVM:-

- Similar to previous models, the RBF SVM's performance is assessed using ROC curves, AUC scores, recall, and confusion matrices.

- This step provides a comprehensive understanding of how well the RBF kernel captures complex relationships in the data.

Result:- Best Hyperparameters: {'C': 10, 'gamma': 0.001}

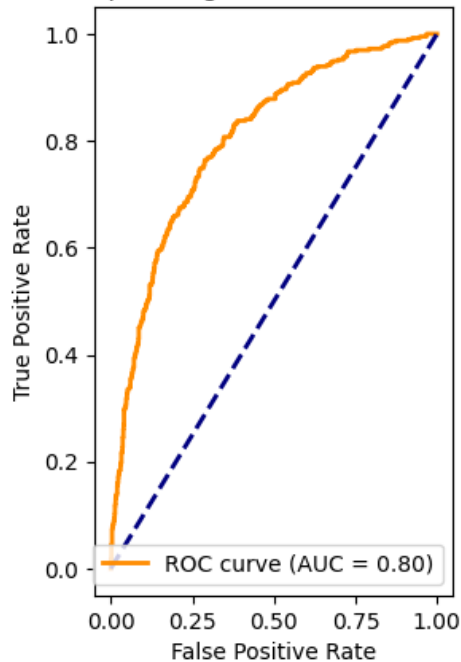
AUC: 0.8026

Recall: 0.4264

Confusion Matrix:

```
[[927  86]
 [226 168]]
```

Receiver Operating Characteristic (ROC) Curve



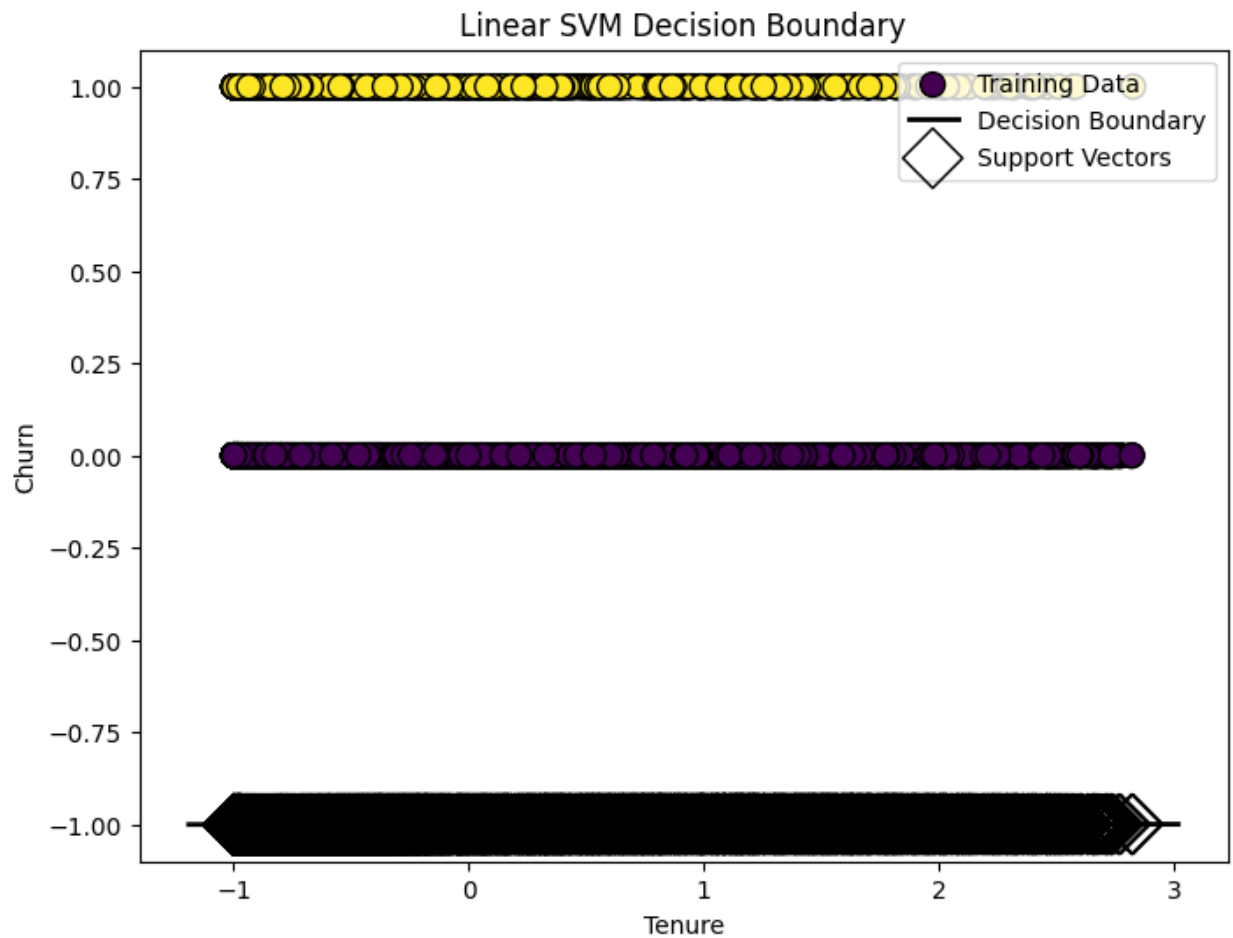
Confusion Matrix

True Label	Actual 0	927	86
	Actual 1	226	168
		Predicted 0	Predicted 1
		Predicted Label	

Decision Boundary Visualization:-

- The code includes a section to visualize the decision boundary created by the RBF SVM on a specific feature, 'TotalCharges'.
- This visualization aids in understanding how the model separates classes based on the 'Tenure' feature.

Result:- Decision Boundary



★ Linear Discriminant Analysis (LDA):-

The primary goal is to employ Linear Discriminant Analysis (LDA) as a classification technique on the given dataset.

Data Preparation:-

- The dataset is split into training and testing sets (X_{train} , y_{train} , X_{test}).
- Features (X) and labels (y) are utilized for training and evaluating the model.

LDA Model Training:-

- The LDA model is trained on the training dataset.
- LDA identifies the linear combinations of features that best separate the classes in the training data.

Prediction and Probability:-

- The trained LDA model is utilized to predict labels for the test set (y_{pred}).
- Predicted probabilities (y_{prob}) for each instance being in the positive class are also obtained.

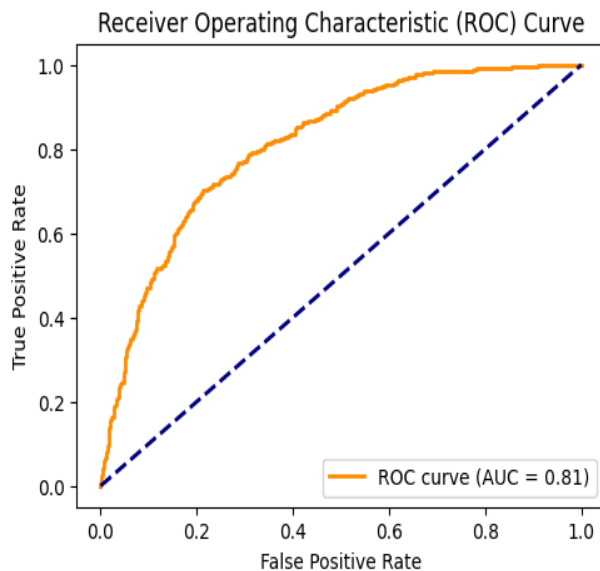
Model Evaluation Metrics:-

- **Accuracy:**
 - Accuracy is calculated to measure the overall correctness of the model's predictions.
 - It is the ratio of correctly predicted instances to the total instances in the test set.
- **AUC Score** (Receiver Operating Characteristic - ROC):
 - AUC provides a summarized performance measurement across different classification thresholds.
 - It quantifies the area under the ROC curve, which represents the trade-off between sensitivity and specificity.
- **Recall** (Sensitivity):
 - Recall measures the ability of the model to correctly identify positive instances.
 - It is particularly relevant when the cost of false negatives is high.
- **Confusion Matrix:**
 - The confusion matrix offers a detailed breakdown of the model's predictions:
 - True Positives (correctly predicted positive instances),
 - True Negatives (correctly predicted negative instances),
 - False Positives (instances incorrectly predicted as positive),
 - False Negatives (instances incorrectly predicted as negative).

Visualizations:-

- **ROC Curve:**
 - The ROC curve visually depicts the trade-off between sensitivity and specificity.
 - AUC score complements the ROC curve, providing a single metric for model performance.
- **Confusion Matrix Heatmap:**
 - A heatmap visualizes the confusion matrix, making it easier to interpret the distribution of true positives, true negatives, false positives, and false negatives.

Result:- Accuracy: 0.7711
AUC: 0.8141
Recall: 0.5355
Confusion Matrix:
[[874 139]
[183 211]]



Confusion Matrix

True Label	Predicted 0	Predicted 1
Actual 0	874	139
Actual 1	183	211

Predicted Label

Performance Summary:-

- Key metrics, including accuracy, AUC, and recall, are summarized to provide a holistic view of the model's effectiveness.

★ K-Nearest Neighbours (KNN):-

The primary goal is to implement the K-Nearest Neighbors (KNN) algorithm for classification on the given dataset.

Hyperparameter Tuning:-

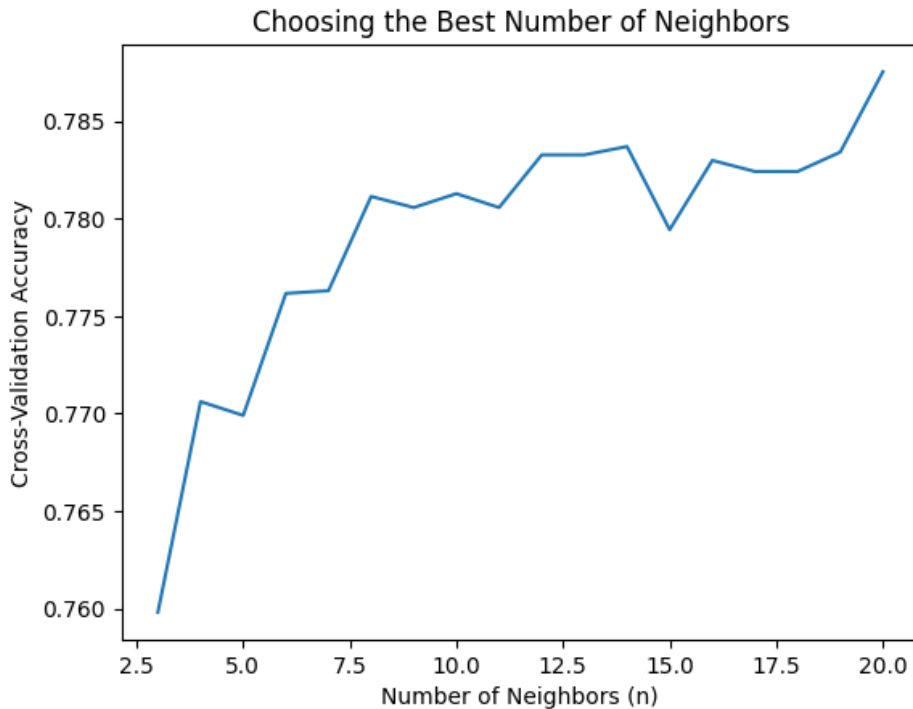
- A range of values for the number of neighbors ('n') is explored using 10-fold cross-validation.
- Cross-validation accuracy is calculated for each 'n' to determine the optimal number of neighbors.
- This process ensures the model's robustness and helps avoid overfitting.

Choosing the Optimal 'n':-

- A plot illustrates the relationship between the number of neighbors and cross-validation accuracy.

- The optimal number of neighbors is identified based on the highest cross-validation accuracy.

Result:- The optimal number of neighbors is 20



Model Training and Evaluation:-

- KNN model is trained using the optimal number of neighbors.
- Model accuracy is evaluated on the testing dataset.
- Additional metrics such as precision, recall, F1 score, and AUC score are computed to assess model performance.

Result:- Accuracy of KNN n=20 on the testing dataset is :0.770

```

Accuracy Score - KNN : 0.7704335465529495
Average Precision - KNN : 0.4400383903185271
F1 Score - KNN : 0.5365853658536585
Precision - KNN : 0.6171617161716172
Recall - KNN : 0.4746192893401015
AUC Score - KNN : 0.7960084886325485

```

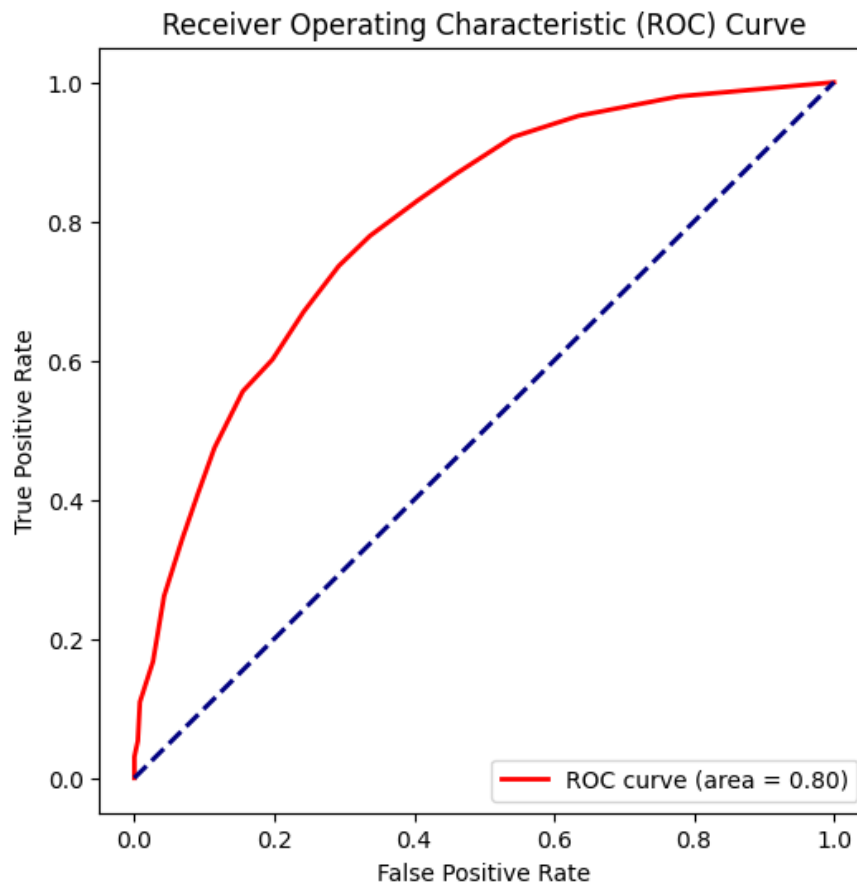
Performance Metrics:-

- Various performance metrics are computed to gain a nuanced understanding of the model's effectiveness.
- Metrics include accuracy, precision, recall, F1 score, and AUC score, offering a comprehensive view of the model's strengths and limitations.

Confusion Matrix and ROC Curve:-

- A confusion matrix is generated to visualize the model's performance in terms of true positives, true negatives, false positives, and false negatives.
- An ROC curve is plotted to illustrate the trade-off between sensitivity and specificity.

Result:- `Confusion matrix([[897, 116],
[207, 187]])`

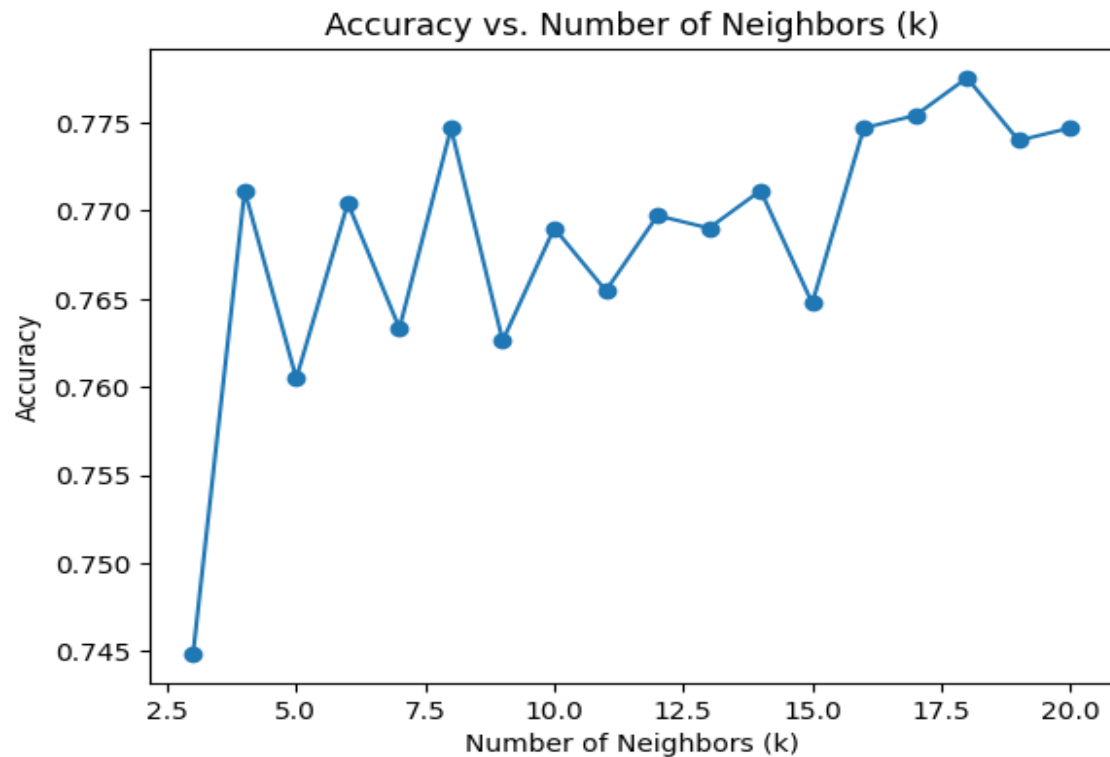


Loss and Accuracy Curves:-

:

- The loss curve illustrates how well the model is learning over epochs, with separate curves for training and validation sets.
- The accuracy curve provides insights into the model's ability to correctly classify instances during training and validation.

Result:-



User Interface:-

Logistic Regression Model:

- Logistic Regression, a well-established algorithm for binary classification, is chosen for fraud prediction. It models the probability of a binary outcome and provides interpretable results.

User Interface Setup:

- Gradio, a Python library for creating user interfaces, is employed to build an interactive interface. This interface facilitates user input and obtaining real-time predictions from the trained Logistic Regression model.

User Input Features:

- The interface includes input fields for features such as 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', etc.
- Users can input these features to get a fraud probability prediction.

Prediction Function:

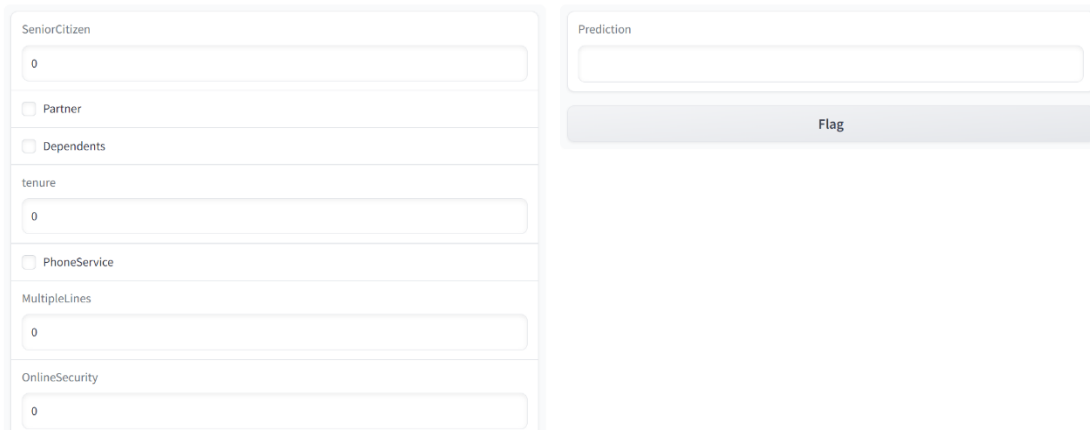
- The fraud prediction function takes user inputs and provides the corresponding fraud probability using the trained Logistic Regression model.

Launching the Interface:

- The Gradio interface is launched, allowing users to interact with the model and receive fraud probability predictions based on input features.

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
Running on public URL: <https://e750bd3670874574eb.gradio.live>

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run 'gradio deploy' from Terminal to deploy to Spaces (<https://huggingface.co/spaces>)



The Gradio interface consists of two main panels. The left panel contains input fields for various features: 'SeniorCitizen' (a slider set to 0), 'Partner' (a checkbox), 'Dependents' (a checkbox), 'tenure' (a slider set to 0), 'PhoneService' (a checkbox), 'MultipleLines' (a slider set to 0), and 'OnlineSecurity' (a slider set to 0). The right panel contains a 'Prediction' output field and a 'Flag' button.

Feature	Value
SeniorCitizen	0
Partner	<input type="checkbox"/>
Dependents	<input type="checkbox"/>
tenure	0
PhoneService	<input type="checkbox"/>
MultipleLines	0
OnlineSecurity	0

Prediction:

Flag