**Predicting Customer Churn in a Telecommunications Company**

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

By

**SHALINI GUHA**

**12107495**

ID: shaliniguha.28.01@gmail.com



**School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab (India)

**INTRODUCTION**

**Objective:**

The primary objective of this project is to develop a predictive model that can identify

customers at risk of churning, enabling the company to take proactive measures to retain

them.

Link to the executed files:-

[**Github link**](https://github.com/inashellshelley/Predicting-Customer-Churn-in-a-Telecommunications-Company/tree/main)

[**Google Colab link**](https://colab.research.google.com/drive/1TGVy8WToEo7buBRBXj5ue3cfOEVkLe-z?usp=sharing)

**Tasks:**

* Data Collection and Preprocessing

Dataset link: [https://www.kaggle.com/datasets/blastchar/telco customer-churn](https://www.kaggle.com/datasets/blastchar/telco%20customer-churn)

* Exploratory Data Analysis (EDA)
* Feature Engineering
* Building the Churn Prediction Model
* Model Evaluation
* Documentation and Reporting

**ABOUT THE DATASET**

As per my understanding, Customer churn is the rate at which any consumer moves from a particular provider to another. Churn is a crucial indicator of performance in any industry today. This particular dataset is presenting us with the Telecommunications firms. The challenge is to gather from the data on how can customer satisfaction improved and lower revenue loss can be achieved by proactively retaining customers by using predictive churn prediction. It is well-structured, with a mix of categorical and numerical data. It contains information about customers, their subscription details, and whether they have churned

About dataset from Kaggle:-

* Context:

"Predict behavior to retain customers. You can analyze all relevant customer data and develop focused customer retention programs." [IBM Sample Data Sets]

* Content

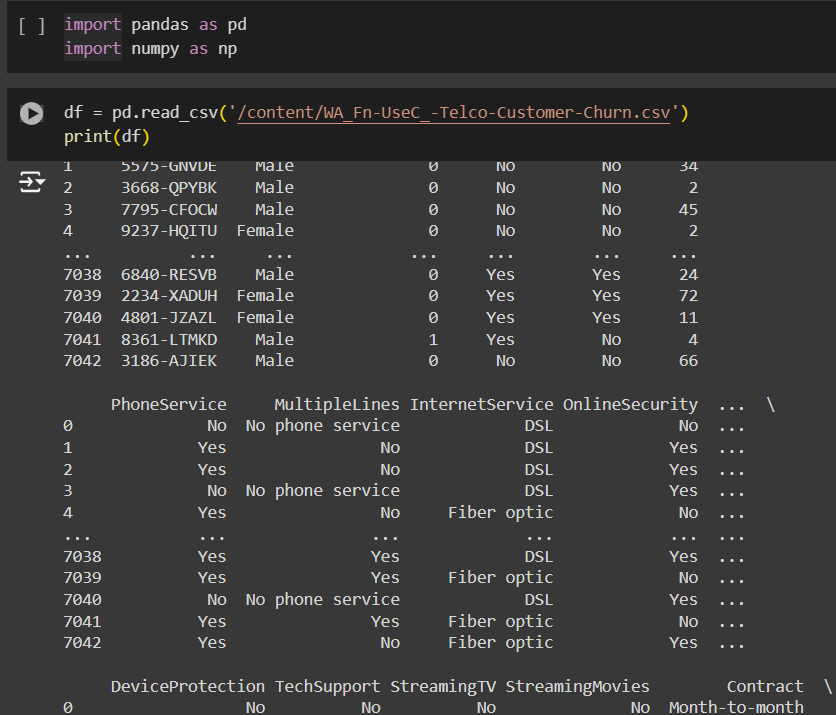
Each row represents a customer, each column contains customer’s attributes described on the column Metadata.

* The data set includes information about:

1. Customers who left within the last month – the column is called Churn
2. Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
3. Customer account information – how long they’ve been a customer, contract, payment method, paperless billing, monthly charges, and total charges
4. Demographic info about customers – gender, age range, and if they have partners and dependents

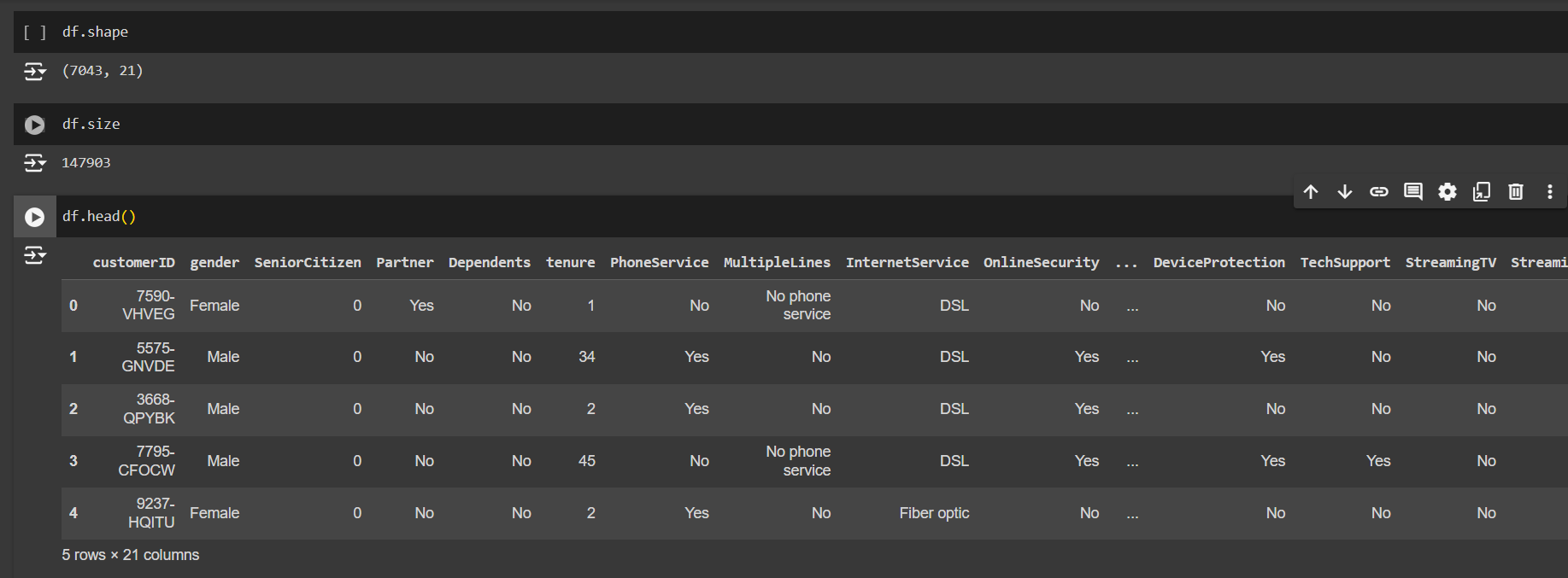
Overview:-

* The dataset contains 7043 rows and 21 columns, representing various attributes of telecom customers and whether they have churned (left the service) or not.
* The shape of the DataFrame is (7043, 21), indicating 7043 rows and 21 columns.
* The size of the DataFrame is 147903, which is the total number of elements (7043 rows \* 21 columns).
* Data Types: The dataset contains both categorical and numerical features.
* Class Distribution: The target variable Churn is binary (Yes/No), indicating whether the customer has churned or not.



Data Columns:

* The columns in the DataFrame are: customerID, gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaperlessBilling, PaymentMethod, MonthlyCharges, TotalCharges, and Churn.
* The first few rows of the DataFrame are displayed using df.head() to provide a snapshot of the data.

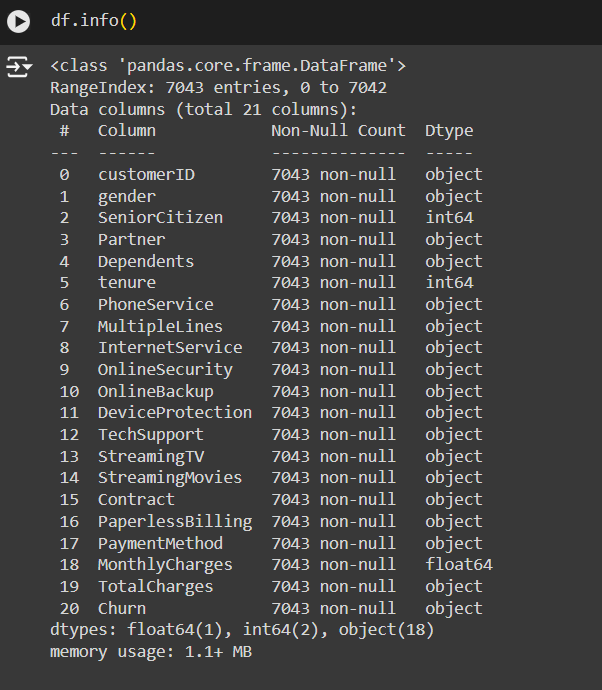


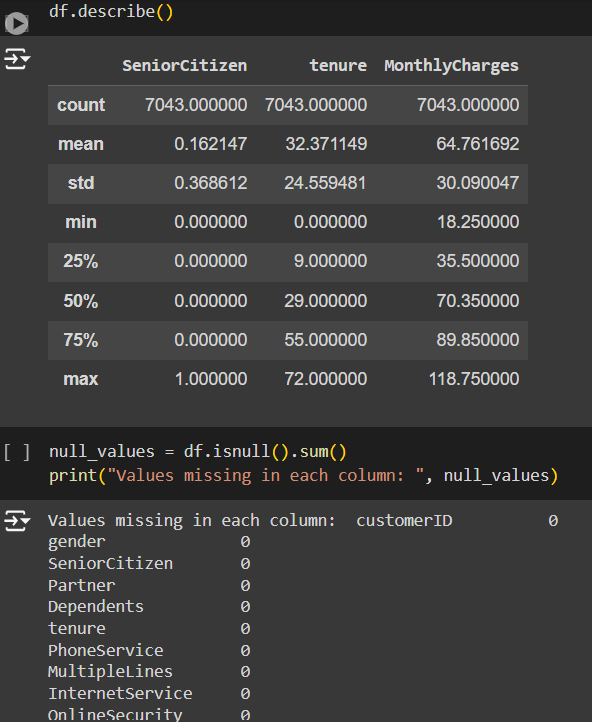
Data Types and Non-Null Counts:

* The info() method provides a summary of the DataFrame, including the data types of each column and the number of non-null entries.
* Most columns are of type object (string), except SeniorCitizen and tenure which are int64, and MonthlyCharges which is float64.

A few columns and their description:-

1. gender: Gender of the customer (Male/Female).
2. SeniorCitizen: Indicates if the customer is a senior citizen (1 for Yes, 0 for No).
3. Dependents: Whether the customer has dependents (Yes/No).
4. tenure: Number of months the customer has stayed with the company.
5. InternetService: Customer's internet service provider (DSL, Fiber optic, No).
6. PaymentMethod: The customer's payment method (Electronic check, Mailed check, Bank transfer, Credit card).
7. MonthlyCharges: The amount charged to the customer monthly.
8. TotalCharges: The total amount charged to the customer.
9. Dependents: Whether the customer has dependents (Yes/No)
10. Churn: Whether the customer has churned (Yes/No).



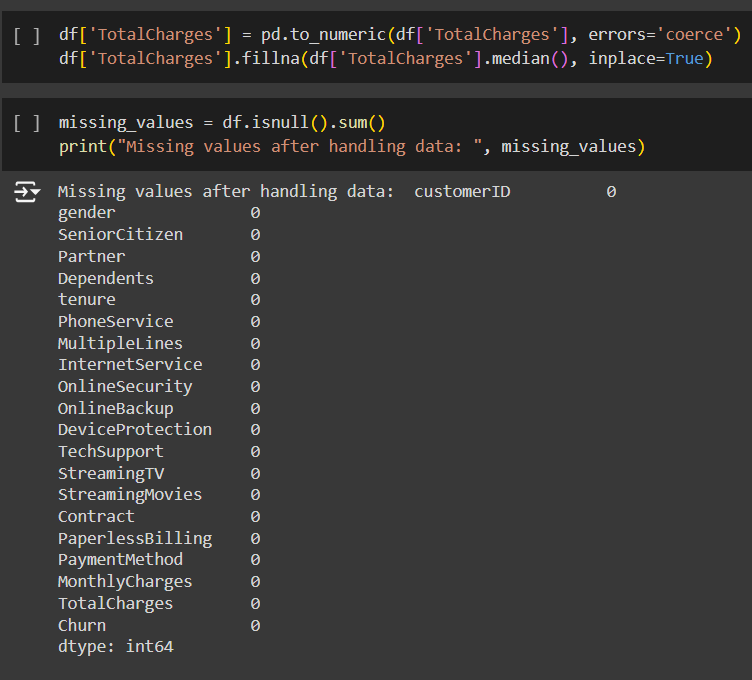


**DATA COLLECTION AND PRE-PROCESSING**

In order to prepare the dataset for training and creating a model, several pre-processing techniques are undertaken:

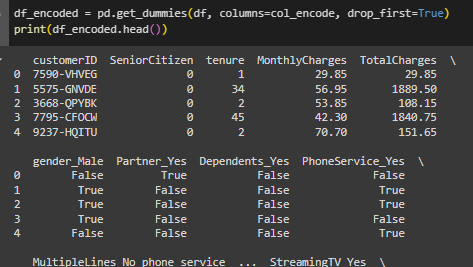
* Handling of missing values:

The TotalCharges column, contains numeric values stored as string. It is converted to numeric data type using pd.to\_numeric(). The errors='coerce' parameter converts invalid parsing to NaN values. After the conversion the missing values are then filled with the median value of the column to handle any gaps in the data using the .fillna() method.



* Encoding Categorical Variables:

One-hot encoding is applied to convert categorical variables into numerical format, which is necessary for machine learning algorithms to process. Categorical variables are one-hot encoded using the pd.get\_dummies() function, creating binary columns for each category. The drop\_first=True parameter is set to drop the first level of each categorical variable to avoid multicollinearity in the data.



* Feature Scaling:

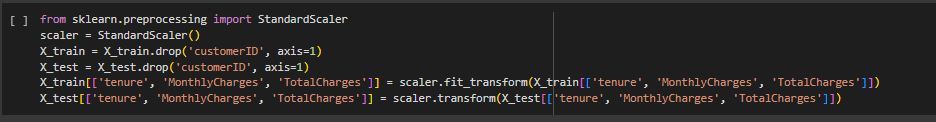
We scale numerical features (tenure, MonthlyCharges, TotalCharges) using StandardScaler to ensure they have a mean of 0 and a standard deviation of 1. This helps improve the performance of algorithms sensitive to feature scales, such as logistic regression and K-nearest neighbors.

* Splitting Data into Features and Target:

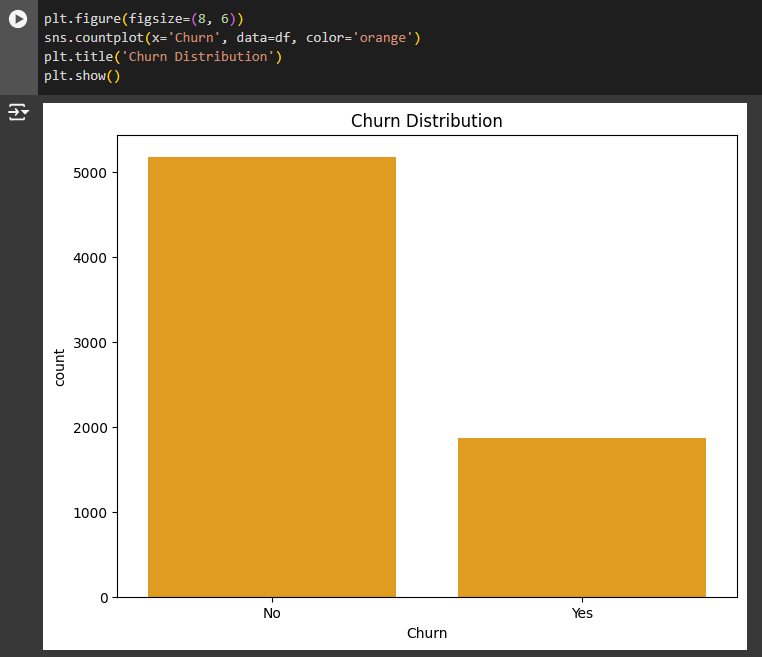
The dataset is split into feature matrix X and target vector y. X contains all columns except the target variable 'Churn\_Yes', while y contains only the target variable.



* Standardization:Numerical features are standardized to have mean 0 and standard deviation 1 using StandardScaler(). I have applied fit\_transform() to the training data to compute the mean and standard deviation and scale the features, while transform() is being used for the testing data to scale them using the parameters that we extract from the training data. We remove the 'customerID' column from both the training and test datasets. The 'customerID' column is not a numerical feature and so it should not be standardized.

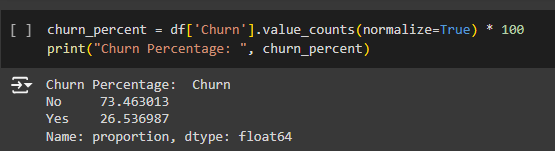


EXPLORATORY DATA ANALYSIS

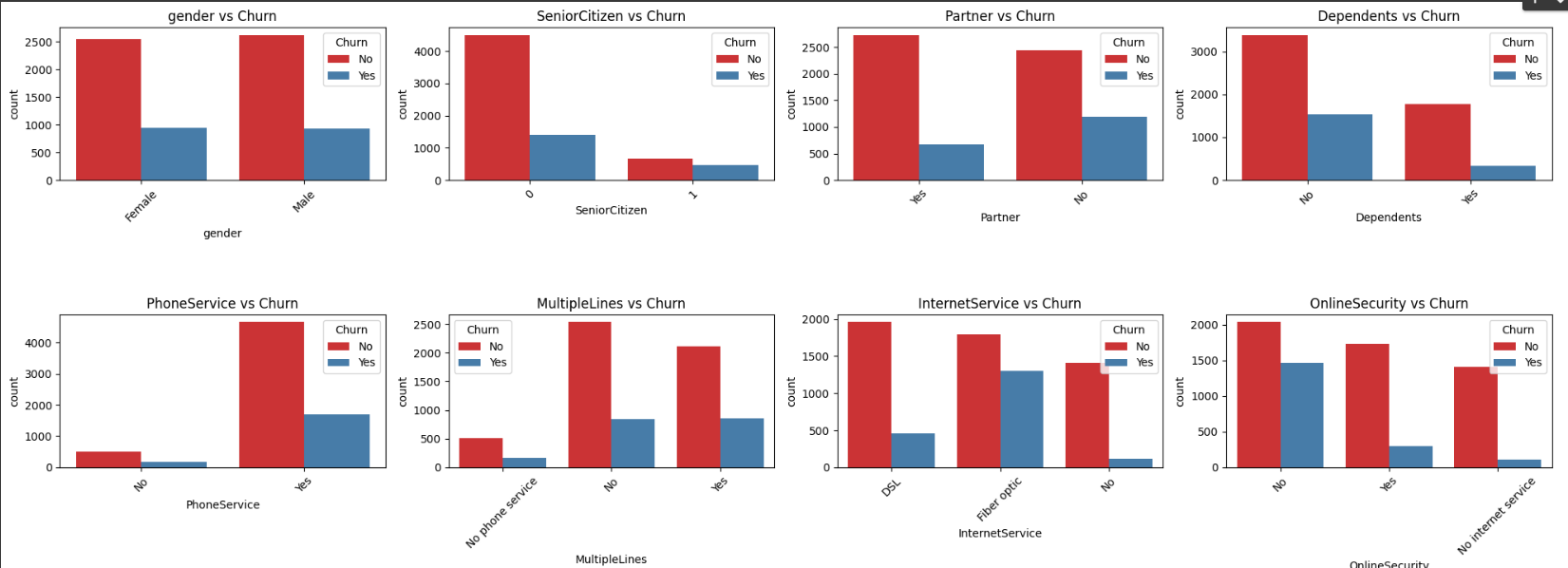


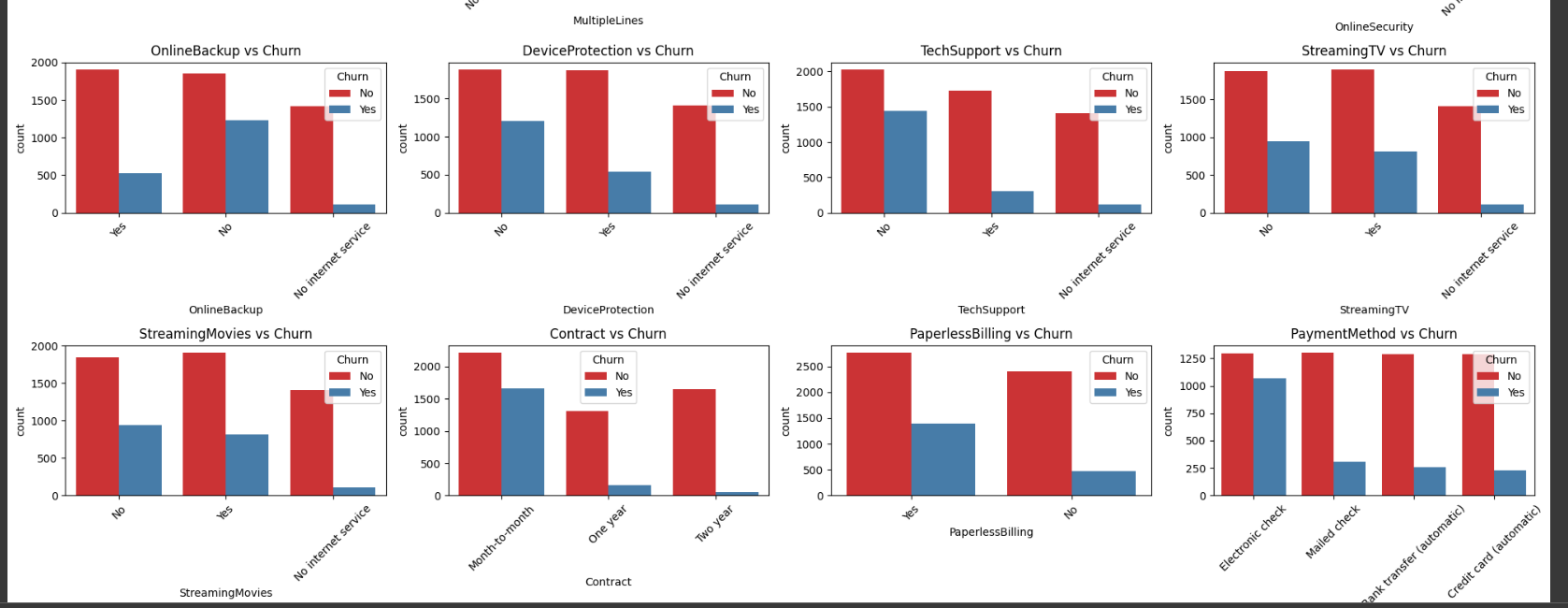
I have created a bar plot using countplot() from seaborn library, to visualize the distribution of the target variable 'Churn'.

The plot displays the count of 'Churn' (Yes/No) categories on the x-axis.

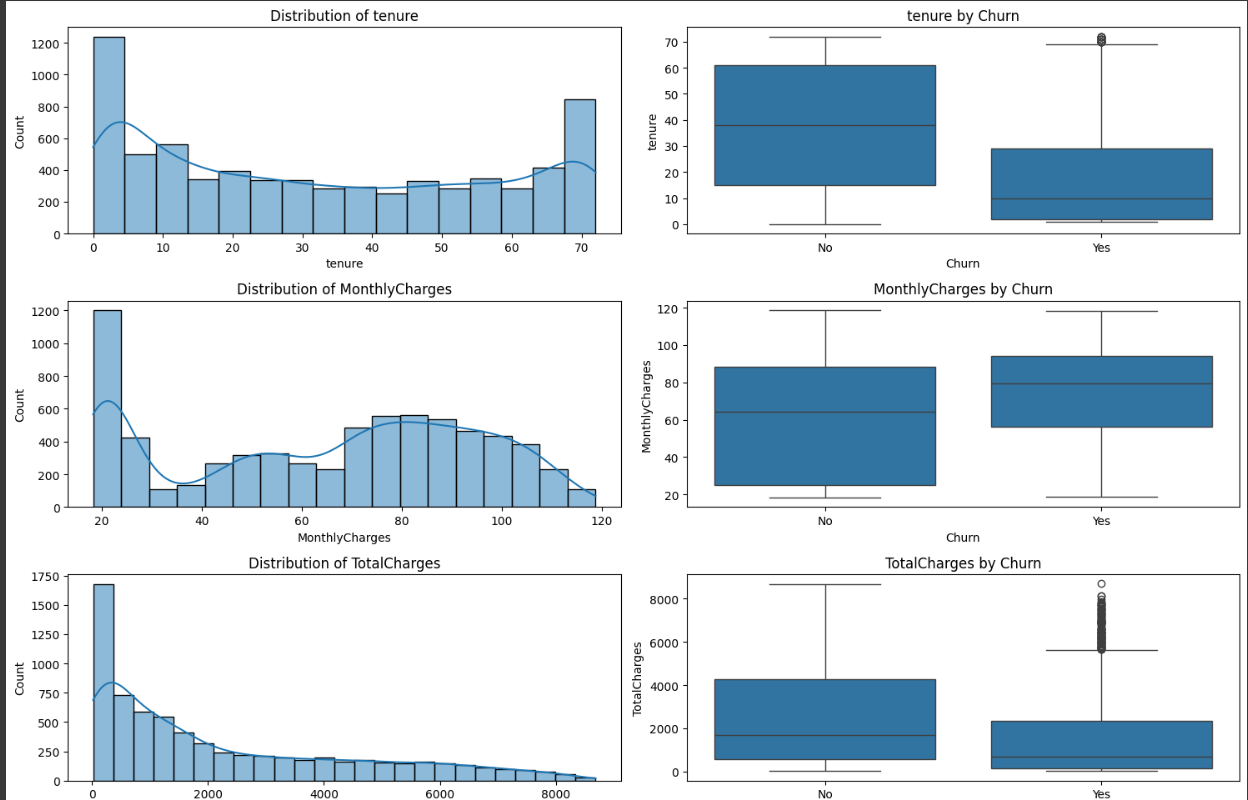


Creating multiple subplots helps us gain better understanding of the relationship between each categorical feature and the target variable 'Churn' through visualization. For each categorical feature, a count plot is made with the count of each category on the x-axis and differentiating the 'Churn' categories using color.

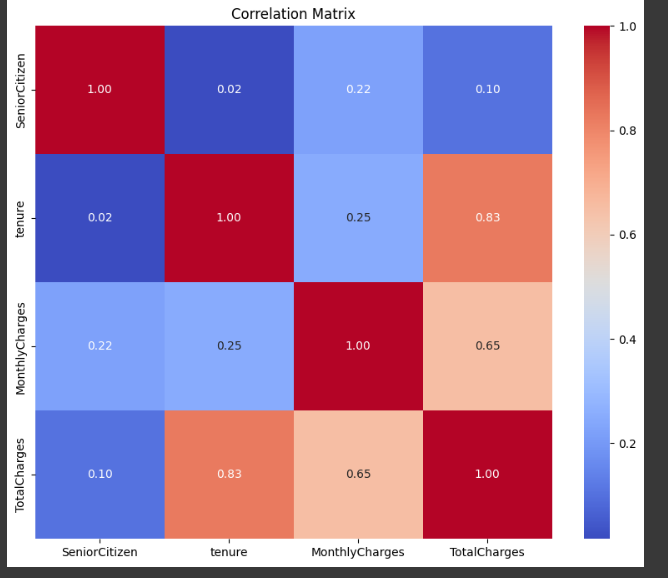




Histograms and box plots are the best methods to visualize the distributions and relationships of numerical features with the target variable 'Churn'. For each numerical feature, a histogram showing its distribution and a box plot illustrating its relationship with 'Churn' are plotted.



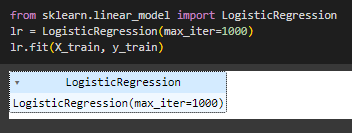
Creating a heatmap is useful in visualizing the correlation matrix of numerical features. The correlation matrix is computed using the .corr() method, and heatmap() function has been used to plot the heatmap with correlation values annotated on each cell.



MODEL TRAINING AND EVALUATION

I have trained using Classification: -

* Logistic Regression



Initializing a Logistic Regression model object with specified parameters and then setting the maximum number of iterations to 1000 for the optimization algorithm, as increasing this value can help improve convergence if the algorithm does not converge within the default number of iterations. X\_train: Features are used for training the model and y\_train: Target variable used for training the model. fit Method, fits the Logistic Regression model to the training data by learning the parameters that best fit the data to minimize the loss function.

* Random Forest

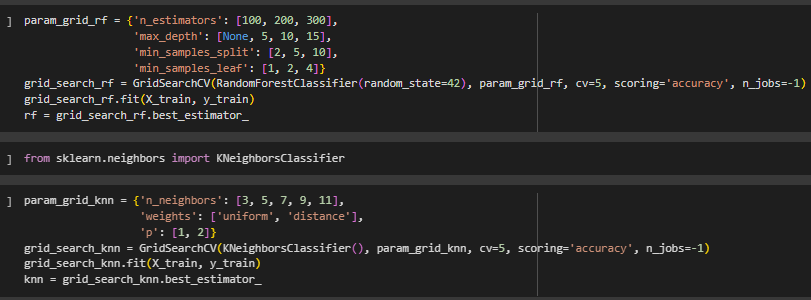
Initialize a Random Forest Classifier with a random state for reproducibility. Perform a grid search to find the best combination of hyperparameters for the Random Forest Classifier. Access the best estimator found during the grid search.

n\_estimators: Increasing the number of trees typically improves the performance of the Random Forest model by reducing overfitting. By trying different values like 100, 200, and 300, we aim to find the optimal balance between model complexity and performance

max\_depth: Controlling the maximum depth helps prevent overfitting. A deeper tree can capture more complex patterns in the data but may lead to overfitting. By including options like None, 5, 10, and 15, we explore a range of tree depths to find the optimal level of complexity.

min\_samples\_split: This parameter controls the granularity of splits in the decision tree. Setting it too low can lead to overfitting, while setting it too high can result in underfitting. By testing values like 2, 5, and 10, we explore different thresholds for splitting nodes.

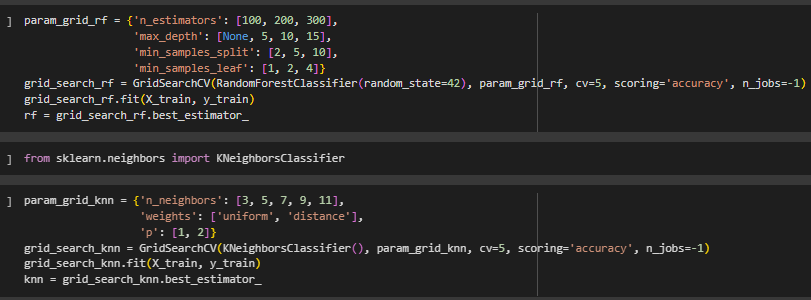
min\_samples\_leaf: Similar to min\_samples\_split, this parameter controls the minimum size of leaf nodes in the decision tree. It helps prevent overfitting by ensuring that leaf nodes have a minimum amount of data for generalization. By trying values like 1, 2, and 4, we vary the minimum leaf size to find the optimal balance between bias and variance.



* K Nearest Neighbors

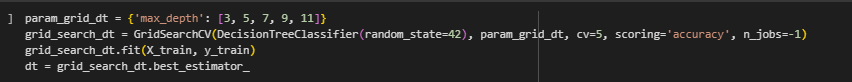
n\_neighbors:The choice of n\_neighbors affects the model's bias-variance trade-off.

Choosing odd numbers helps avoid ties when predicting binary outcomes, reducing ambiguity.By considering a range of values from 3 to 11, we explore both simpler (3 neighbors) and more complex (11 neighbors) models. A smaller n\_neighbors value leads to a more flexible model that might capture local patterns well but could be sensitive to noise, while a larger value promotes smoother decision boundaries but might miss subtle patterns.



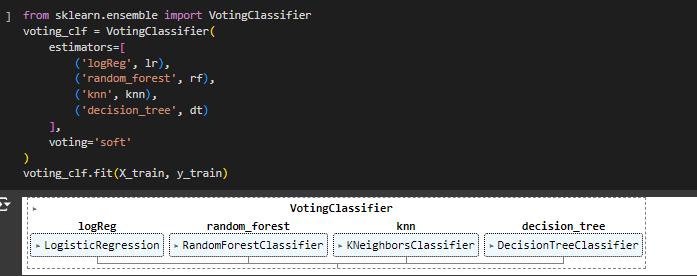
* Decision Tree

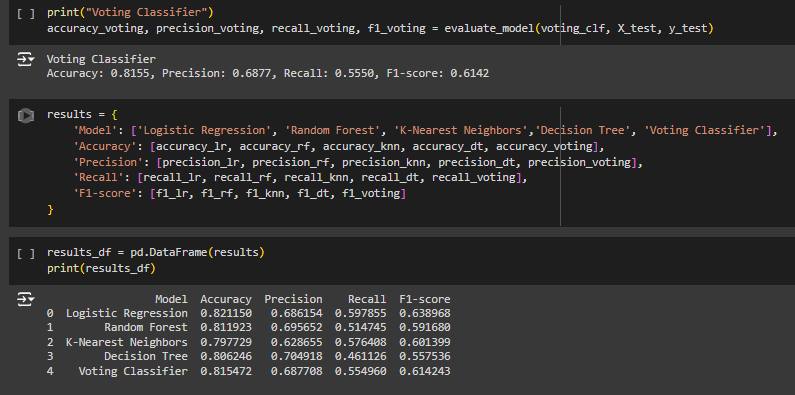
max\_depth: Controlling the max\_depth helps prevent overfitting and regulates the complexity of the decision tree. By trying different depths, ranging from shallow (3) to deep (11), we explore the trade-off between model complexity and bias. A smaller max\_depth results in a simpler tree that may generalize better but might miss capturing complex patterns in the data. The choice of these specific numbers for max\_depth allows for a comprehensive exploration of the hyperparameter space. By systematically testing different depths, we aim to identify the optimal max\_depth that yields the best performance in terms of accuracy for the Decision Tree Classifier on the given dataset. On the other hand, a larger depth allows the tree to capture more intricate relationships in the data but increases the risk of overfitting. The param\_grid\_dt dictionary specifies the grid of hyperparameters to search over, with max\_depth being the only parameter considered in this case.



* Voting Classifier

By combining the predictions of multiple classifiers, the Voting Classifier aims to achieve better predictive performance than any single classifier alone. The soft voting strategy considers the confidence levels (probabilities) of each classifier when making the final prediction, leading to potentially improved results.



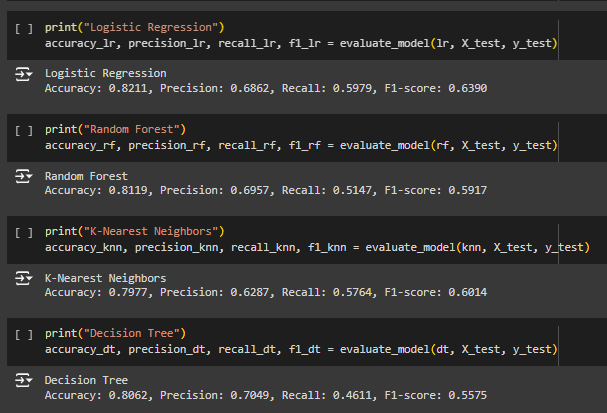


Accuracy: Accuracy measures the proportion of correctly predicted labels out of the total number of samples.

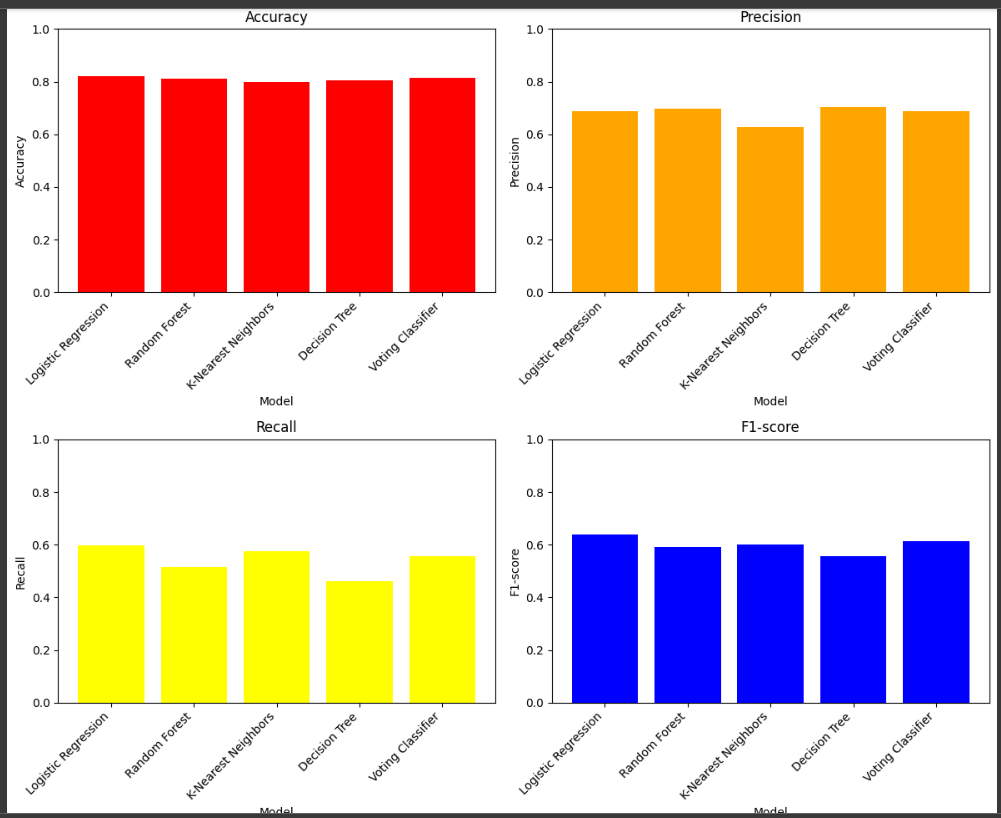
Precision: Precision measures the proportion of true positive predictions out of all positive predictions.

Recall (Sensitivity): Recall measures the proportion of true positive predictions out of all actual positive instances.

F1-score: F1-score is the harmonic mean of precision and recall, providing a balance between the two metrics.



* Accuracy: Logistic Regression achieved the highest accuracy (82.11%), closely followed by the Voting Classifier (81.55%) and Random Forest (81.19%). KNN and Decision Tree had slightly lower accuracies.
* Precision: Decision Tree had the highest precision (70.49%), followed by Random Forest (69.57%). Logistic Regression and the Voting Classifier also had relatively high precision values. KNN had the lowest precision among the models.
* Recall: Logistic Regression achieved the highest recall (59.79%), indicating its ability to correctly identify a high percentage of churn cases. Decision Tree had the lowest recall among the models.
* F1-score: Logistic Regression achieved the highest F1-score (63.90%), which is a harmonic mean of precision and recall. Decision Tree had the lowest F1-score, indicating a trade-off between precision and recall.



CONCLUSION

Logistic Regression emerges as the top-performing model in this analysis. Logistic Regression performs well across all metrics, striking a balance between accuracy, precision, recall, and F1-score. It is followed closely by Random Forest and the Voting Classifier. Decision Tree shows high precision but struggles with recall, suggesting that it might be overly biased towards non-churn predictions.KNN performs relatively well but lags behind in precision compared to other models.These models could be further optimized and fine-tuned to improve performance for practical deployment in predicting customer churn.