Telco Customer Churn Prediction

Problem Description:

The telecommunications industry allows customers to select from various providers for their communication and internet needs. Customers place great importance on the quality of the services they receive and often judge an entire company based on a single experience. Communication services have become integral to daily life, to the extent that even a brief 30-minute maintenance interruption can cause anxiety among users, underscoring our tendency to take these services for granted. Furthermore, due to the high costs associated with acquiring new customers, analyzing and mitigating churn (customer attrition) is crucial. Churn rate is a metric that quantifies the number of customers who have either canceled their subscriptions or chosen not to renew, and a higher churn rate directly impacts a company's revenue by reducing its customer base.

Hence, insights gained from churn analysis can guide companies in formulating strategies, targeting specific customer segments, enhancing service quality, and ultimately fostering trust among customers. This is why the development of predictive models and the creation of churn analysis reports are essential components in achieving growth and customer retention.

Objective:

- The goal is to categorize potential churn customers based on both numerical and categorical features.
- This is a binary classification problem, and it deals with an imbalanced dataset.

- **customerID**: Unique identifier for each customer.
- **gender**: Indicates the customer's gender (male or female).
- **SeniorCitizen**: Shows whether the customer is a senior citizen (1 for yes, 0 for no).
- **Partner**: Specifies if the customer has a partner (Yes or No).
- **Dependents**: Indicates if the customer has dependents (Yes or No).
- **tenure**: Represents the number of months the customer has been with the company.
- **PhoneService**: Denotes whether the customer has phone service (Yes or No).
- **MultipleLines**: Specifies if the customer has multiple phone lines (Yes, No, or No phone service).

- **InternetService**: Identifies the customer's internet service provider (DSL, Fiber optic, or None).
- **OnlineSecurity**: Indicates if the customer has online security (Yes, No, or No internet service).
- **OnlineBackup**: Specifies if the customer has online backup (Yes, No, or No internet service).
- **DeviceProtection**: Indicates if the customer has device protection (Yes, No, or No internet service).
- **TechSupport**: Specifies whether the customer has tech support (Yes, No, or No internet service).
- **StreamingTV**: Indicates if the customer has streaming TV (Yes, No, or No internet service).
- **StreamingMovies**: Specifies if the customer has streaming movies (Yes, No, or No internet service).
- **Contract**: Represents the customer's contract term (Month-to-month, One year, or Two years).
- **PaperlessBilling**: Denotes if the customer uses paperless billing (Yes or No).
- **PaymentMethod**: Specifies the customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), or Credit card (automatic)).
- **MonthlyCharges**: Represents the customer's monthly service charges.
- **TotalCharges**: Denotes the total amount charged to the customer.
- **Churn**: Indicates whether the customer has churned (Yes or No).

Notebook Contents :

- Dataset Information
- Exploratory Data Analysis (EDA)
- Summary of EDA
- Feature Engineering
- Modeling
- Conclusion

What you will learn :

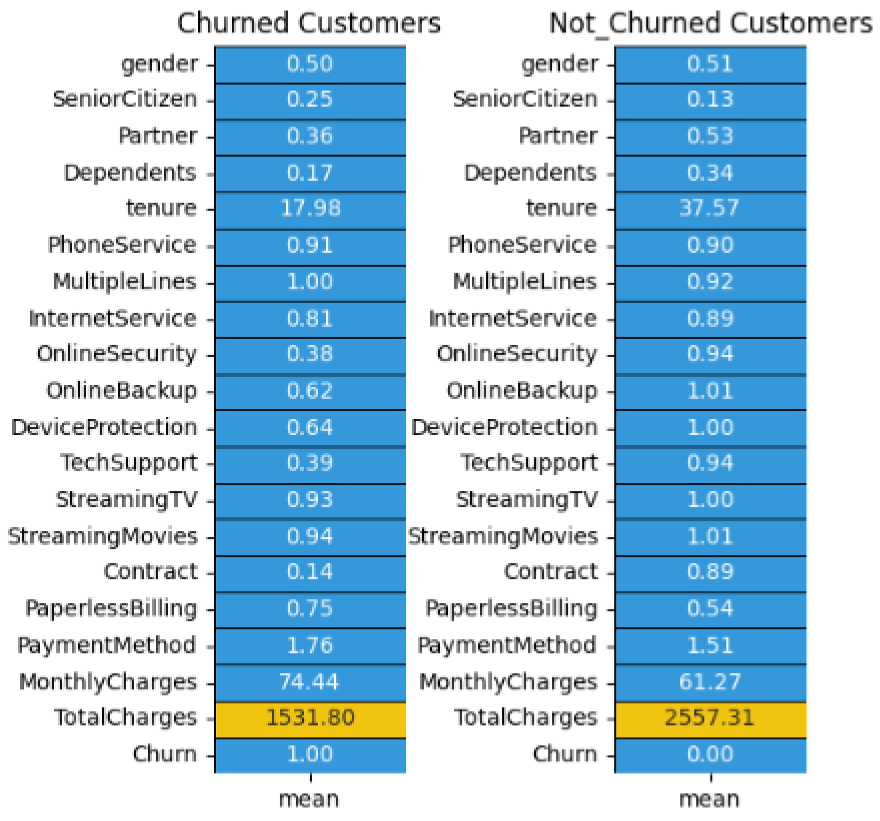
- Data Visualization
- Data Scaling
- Statistical Tests for Feature Selection
- Modeling and visualization of results for algorithms
- Balancing an unbalanced dataset
- Stacking of classifiers

Code Documentation:

- **import pandas as pd**: Imports the Pandas library and gives it an alias 'pd' for easier access. Pandas is a powerful data manipulation and analysis library.
- **import numpy as np**: Imports the NumPy library and gives it an alias 'np'. NumPy is used for numerical operations and array manipulations.
- **import matplotlib.pyplot as plt**: Imports the Matplotlib library, specifically the 'pyplot' module, and gives it an alias 'plt'. Matplotlib is used for creating data visualizations, such as charts and plots.
- **%matplotlib inline**: This is a magic command for Jupyter notebooks, which specifies that Matplotlib should generate plots within the notebook's cell outputs.
- **import seaborn as sns**: Imports the Seaborn library, commonly used for enhancing the aesthetics and statistical data visualization.
- **pd.options.display.float_format = '{:.2f}'.format**: Sets the display format for floating-point numbers in Pandas to show only two decimal places. This can make the output cleaner and more readable.
- **import warnings**: Imports the 'warnings' module, allowing you to manage warning messages generated by Python.
- **warnings.filterwarnings('ignore')**: Suppresses warning messages, preventing them from being displayed in the output. Be cautious when using this, as it might hide potentially important warnings.
- We import the Pandas library to work with data in tabular form.
- The **pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')** line reads the content of the 'WA_Fn-UseC_-Telco-Customer-Churn.csv' CSV file and stores it in a Pandas DataFrame named 'data'.
- The **data.head()** line is used to display the first 5 rows of the DataFrame, providing an initial view of the dataset's structure and content.
- **data.shape** This code snippet returns the dimensions (number of rows and columns) of the DataFrame named **data**. It's used to quickly get an overview of the size of the DataFrame, showing the number of rows and columns.
- **data.columns** This code snippet retrieves the names of the columns in the DataFrame **data**. It's helpful for listing the column names, which is useful when you need to reference specific columns in your DataFrame.

- **data.info()** This code snippet provides concise information about the DataFrame **data**, including data types, non-null values, and memory usage. It's often used to understand the structure and summary statistics of the data in your DataFrame.
- **sns.heatmap(data.isnull(), cmap='magma', cbar=False)** creates a straightforward visualization of missing (null) values within a DataFrame. It employs a heatmap to represent your data, with empty cells highlighting where data is absent. The color scheme 'magma' is used for this representation, but you can choose from other available color schemes. Additionally, the **cbar** parameter is set to **False** to remove the color bar on the side of the heatmap. This code is handy for quickly identifying areas in your dataset where data is missing, making data cleaning and analysis more efficient. Make sure to import the **seaborn** library (**sns**) and have a valid DataFrame named **data** for this code to work.
- **data.describe().T** is used to generate a statistical summary of a dataset in a more readable and transposed format. It calculates common descriptive statistics for the numerical columns in the DataFrame 'data', such as mean, standard deviation, minimum, maximum, and quartiles, and transposes the resulting table for improved readability. This transposed summary provides a quick overview of the data's central tendencies and spread, making it easier to assess the dataset's key statistics at a glance. It's a useful tool for data exploration and initial insights into the dataset's numeric characteristics.
- In this code snippet, we perform several data preprocessing tasks on a DataFrame. First, we convert the 'TotalCharges' column elements from strings to floating-point numbers using the `astype(float)` method. We then identify and handle cases where the 'TotalCharges' column contains empty spaces. We find the index positions of these empty spaces and store them in the 'l2' list. Next, we iterate through these index positions, and for each identified position, we replace the empty space with the value from the previous row. Finally, we remove the 'customerID' column from the DataFrame using the `drop` method with the `inplace=True` argument. This code snippet streamlines data preparation and conversion, making it ready for further analysis or modeling.
- The provided code snippet demonstrates how to use the LabelEncoder from the scikit-learn library for transforming categorical text data into numerical values. First, a deep copy of the DataFrame 'data' is created and stored in 'df1' to ensure the original data is not modified. Next, a list called 'text_data_features' is generated, containing column names that are not present in the statistical description of the dataset. The code then iterates through these text data features, applying the LabelEncoder to each column in 'df1.' The LabelEncoder assigns a unique numerical label to each distinct category within the column, effectively converting the text data into numerical representations. Finally, the code prints out the unique values and their corresponding text labels using the 'le.inverse_transform' function, making it easier to understand the mapping between text categories and their numeric representations. This transformation is crucial for preparing categorical data for machine learning models that require numerical inputs.

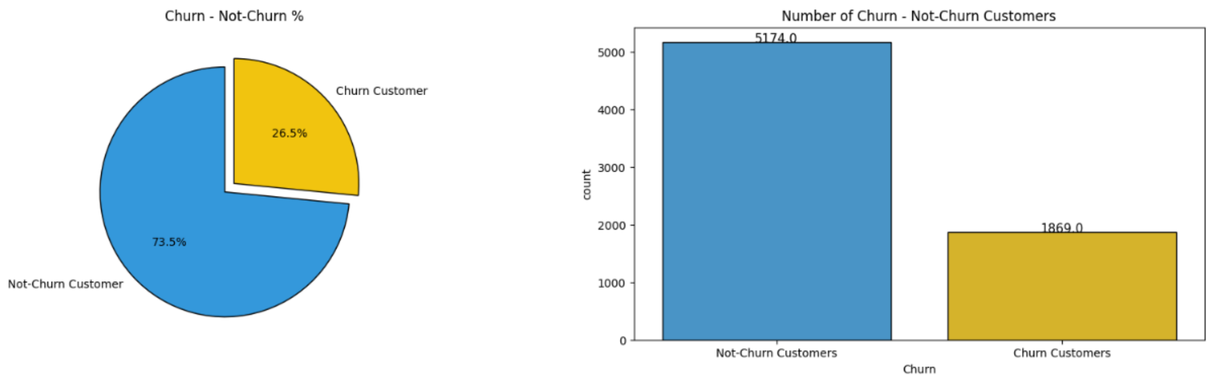
- **df1.describe()** function is used to generate a summary of statistical information for a given DataFrame `df1` in Python using the Pandas library. When you call this function, it calculates various statistical measures for each numeric column in the DataFrame, such as count (number of non-null values), mean (average), standard deviation, minimum, and maximum values. The resulting output provides a quick overview of the central tendency and spread of the data, which is valuable for data exploration and initial data analysis. It's a simple and powerful way to get a sense of the distribution and characteristics of your dataset at a glance.
- The provided code generates two side-by-side heatmaps to visualize descriptive statistics for churned and non-churned customers in a dataset. It first separates the dataset into two subsets: one containing churned customers and the other containing non-churned customers. Then, it creates a figure with two subplots using Matplotlib, arranging them in a single row. In each subplot, a seaborn heatmap is generated to display the mean values of various features, with annotations and a color map using the specified colors. The heatmaps also include grid lines, a black outline, and decimal formatting. The left subplot represents churned customers, while the right subplot represents non-churned customers. The `fig.tight_layout(pad=0)` statement ensures that the subplots are properly spaced. This code is useful for visualizing how the mean values of different features differ between churned and non-churned customer groups, providing insights into potential patterns or differences in the data.



	Churned Customers (mean)	Not_Churned Customers (mean)
gender	0.50	0.51
SeniorCitizen	0.25	0.13
Partner	0.36	0.53
Dependents	0.17	0.34
tenure	17.98	37.57
PhoneService	0.91	0.90
MultipleLines	1.00	0.92
InternetService	0.81	0.89
OnlineSecurity	0.38	0.94
OnlineBackup	0.62	1.01
DeviceProtection	0.64	1.00
TechSupport	0.39	0.94
StreamingTV	0.93	1.00
StreamingMovies	0.94	1.01
Contract	0.14	0.89
PaperlessBilling	0.75	0.54
PaymentMethod	1.76	1.51
MonthlyCharges	74.44	61.27
TotalCharges	1531.80	2557.31
Churn	1.00	0.00

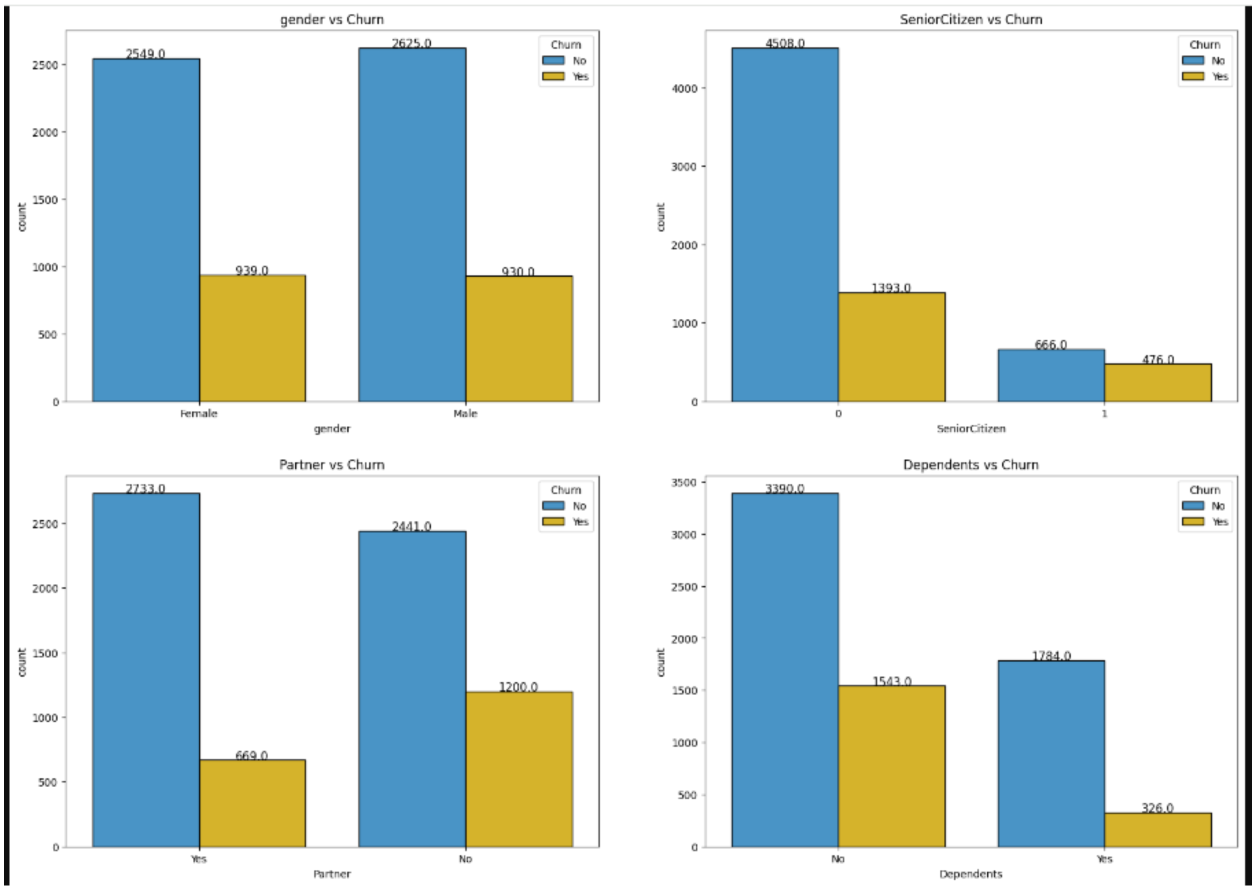
Exploratory Data Analysis

- The provided code snippet is used to categorize columns in a DataFrame into "categorical features" and "numerical features." It first creates empty lists to store these features, then iterates through the columns in the DataFrame 'df1.' For each column, it checks the number of unique values. If the number of unique values is greater than 6, the column is considered a "numerical feature" and is added to the 'numerical_features' list; otherwise, it is categorized as a "categorical feature" and added to the 'categorical_features' list. Finally, the code prints the lists of categorical and numerical features. This code is helpful for an initial exploration of your dataset, aiding in feature type identification.
- The provided code calculates the distribution of churn and non-churn customers in a dataset, visualizing the results in a side-by-side pie chart and countplot using Python's matplotlib and seaborn libraries. It first calculates the proportions of churn and non-churn customers and creates a list 'circle' with the respective percentages. Then, it generates a side-by-side figure with two subplots. The first subplot displays a pie chart representing the percentage of churn and non-churn customers with labels, percentages, and color distinctions. The second subplot presents a countplot showing the absolute number of churn and non-churn customers with corresponding labels and counts displayed above each bar. This code is useful for quickly visualizing the distribution of customer churn in a dataset, making it easier to understand and communicate key insights from the data.

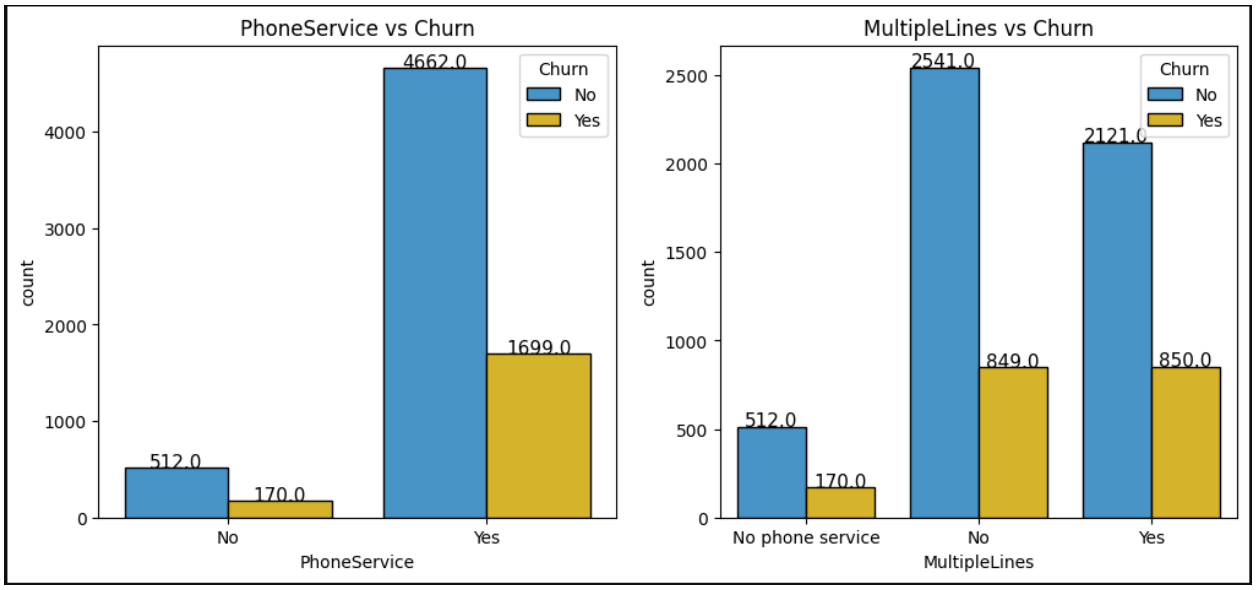


- The code snippet `categorical_features.remove('Churn')` is used to remove the feature 'Churn' from a list of categorical features, presumably for data preprocessing in a machine learning or data analysis context. This line of code modifies the `categorical_features` list by eliminating the 'Churn' feature from it. Typically, this operation is performed when 'Churn' is not considered a categorical feature in the dataset, and you want to exclude it from further processing or analysis. It is a straightforward operation to tailor the list of categorical features to the specific needs of your analysis by removing any irrelevant or misclassified features.

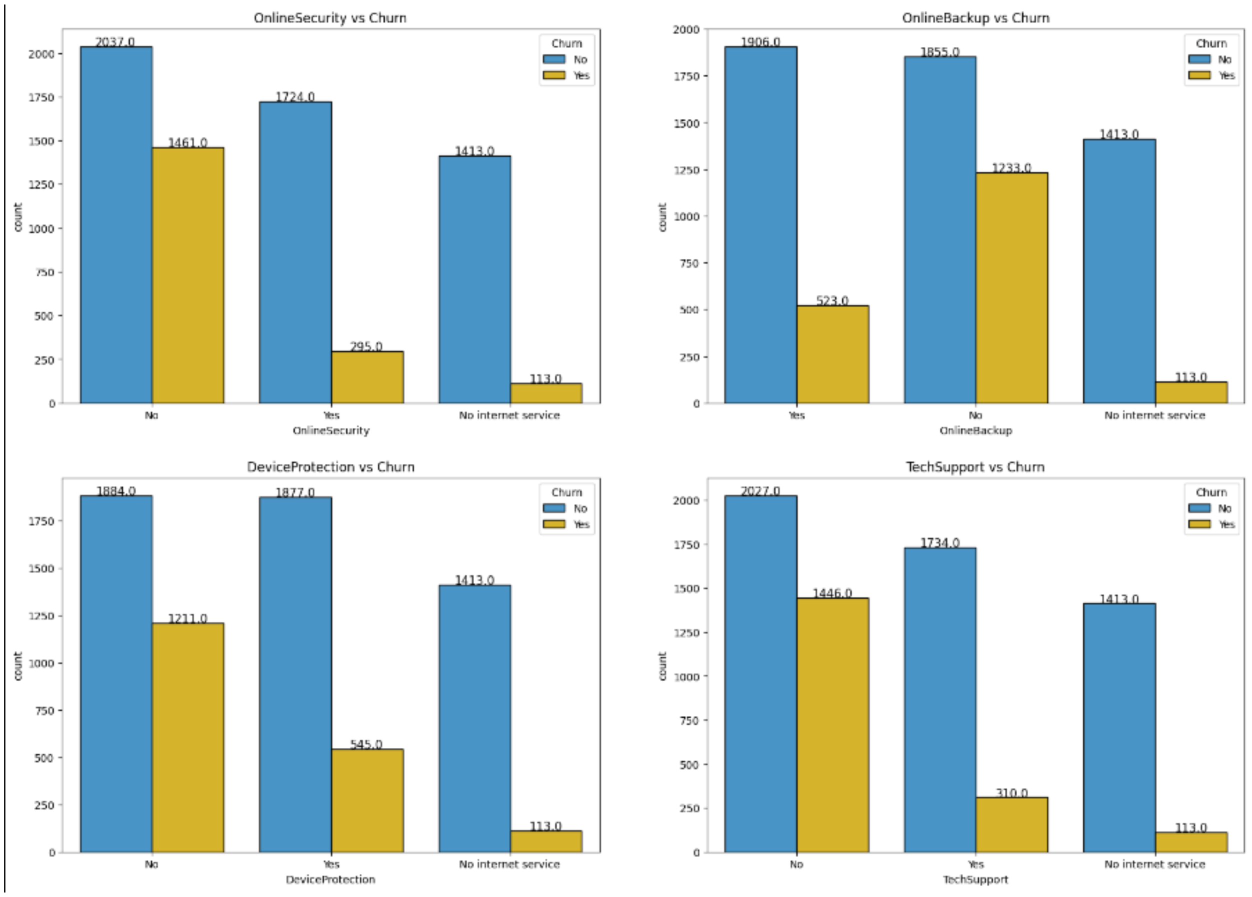
- The provided code consists of three lists: 'l1', 'l2', and 'l3', each representing different categories of information. 'l1' contains attributes related to customer information, such as gender, senior citizen status, partner, and dependents. 'l2' includes attributes indicating the services customers have signed up for, encompassing phone service, multiple lines, internet service, streaming TV, streaming movies, online security, online backup, device protection, and tech support. Finally, 'l3' comprises attributes related to payment information, covering contract type, paperless billing preferences, and payment method. These lists are likely used to categorize and organize data or feature names in a dataset, making it more structured and understandable when working with customer-related data.
- The provided code snippet uses the `matplotlib` and `seaborn` libraries to create a 2x2 grid of subplots, each displaying a countplot comparing a categorical variable (represented by `l1[i]`) with the 'Churn' variable from a dataset. The figsize of the figure is set to 20x14. Inside the loop, each subplot is created with `sns.countplot()`, and the 'Churn' variable is used for color differentiation with the specified color palette. For each subplot, text labels are added to display the count of each category on top of the bars. The subplot titles are generated based on the categorical variable and 'vs Churn' to describe the comparison. This code is useful for visualizing the distribution of categorical variables concerning the 'Churn' variable in a dataset, making it easier to understand their impact on customer churn.



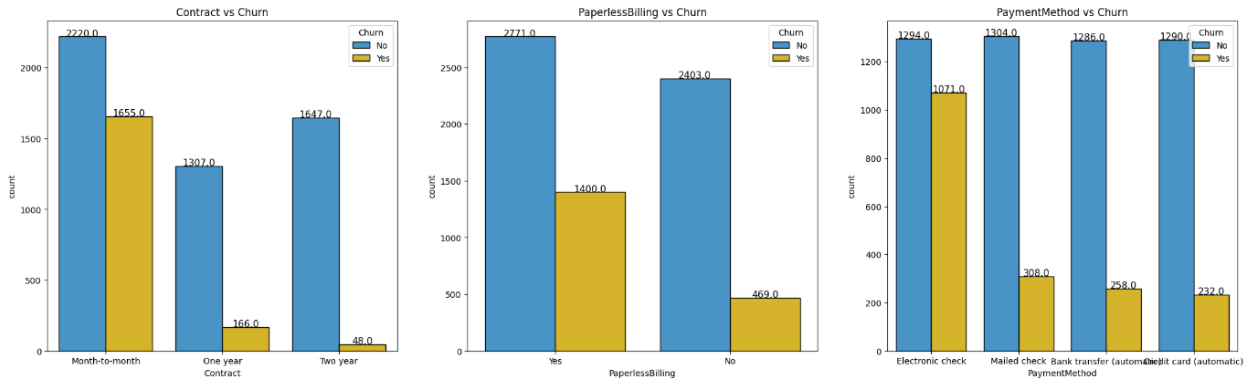
- The provided code snippets generate a series of subplots for data visualization using the Seaborn library in Python. It first creates a figure with a specific number of rows and columns, along with their respective sizes. Within each subplot, a count plot is generated to visualize the relationship between categorical variables from the 'l2' list and the 'Churn' variable in the 'data' DataFrame. Each subplot is customized with a title, and data labels are added above the bars for clarity. The code is structured into three segments: the first part visualizes the first two variables in 'l2', the second part visualizes the third variable in 'l2', and the final part visualizes the fourth and fifth variables in 'l2' in pairs. These code segments are intended for exploratory data analysis to better understand how 'Churn' relates to the selected categorical variables in your dataset.



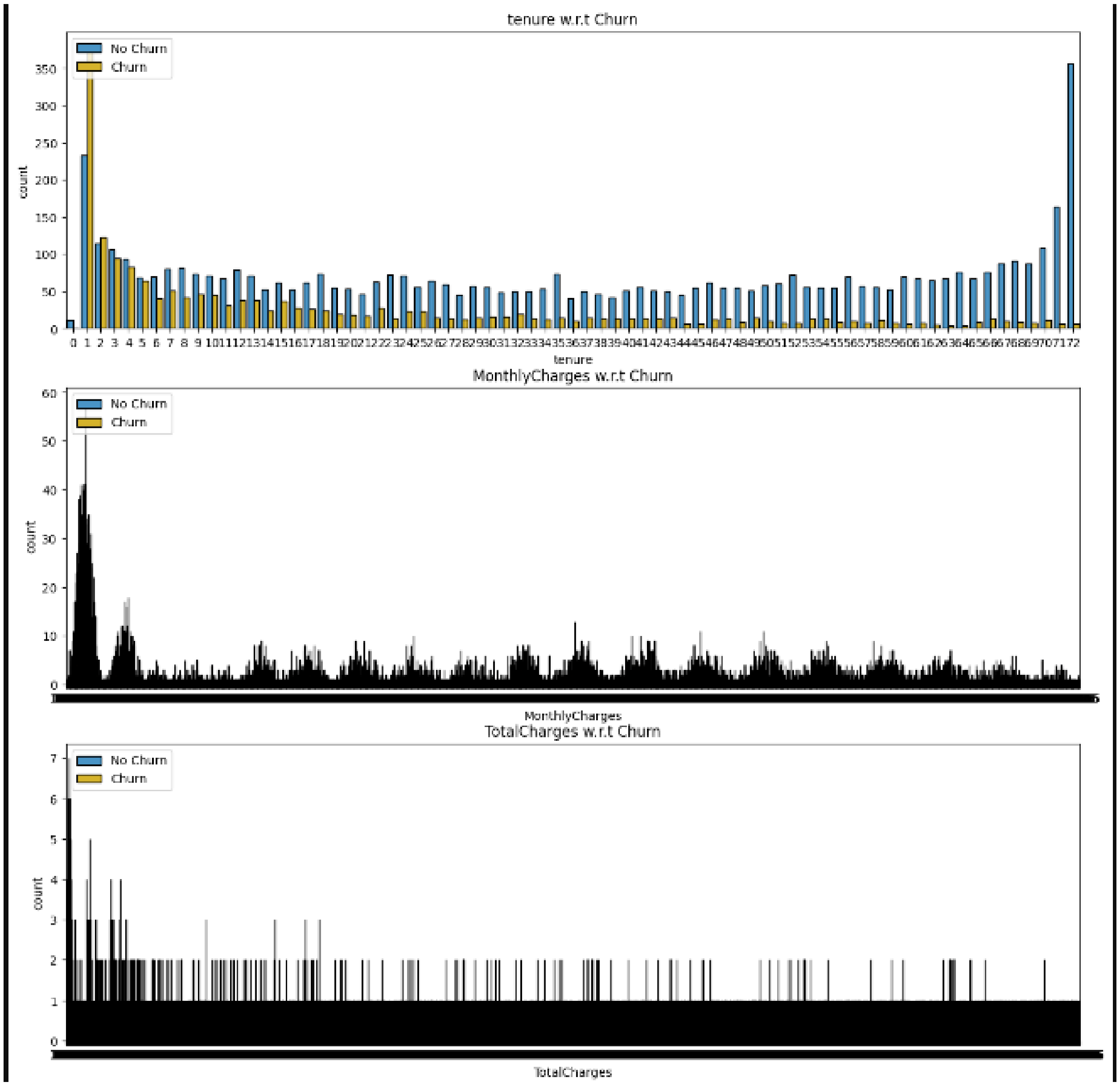
- The provided code snippet utilizes the Matplotlib and Seaborn libraries to create a 2x2 grid of subplots, each displaying count plots for specific categorical variables against the "Churn" variable in a dataset. The `fig = plt.subplots(nrows=2, ncols=2, figsize=(20, 14))` line initiates a figure with a 2x2 grid for subplots and sets the figure size. The subsequent loop iterates over a list of categorical variables (assumed to be in the list `l2[-4:]`) to create individual subplots. In each subplot, it generates a count plot using Seaborn's `sns.countplot()` function, specifying the variable to be plotted, the dataset, "Churn" as the hue, a custom color palette (`colors`), and sets edge colors for bars. For each bar in the count plot, the code annotates the bar's height with the corresponding count. Finally, it sets a title for each subplot based on the variable being plotted against "Churn." This code visually explores the relationship between these categorical variables and the "Churn" variable in the dataset, making it informative for data analysis and visualization purposes.



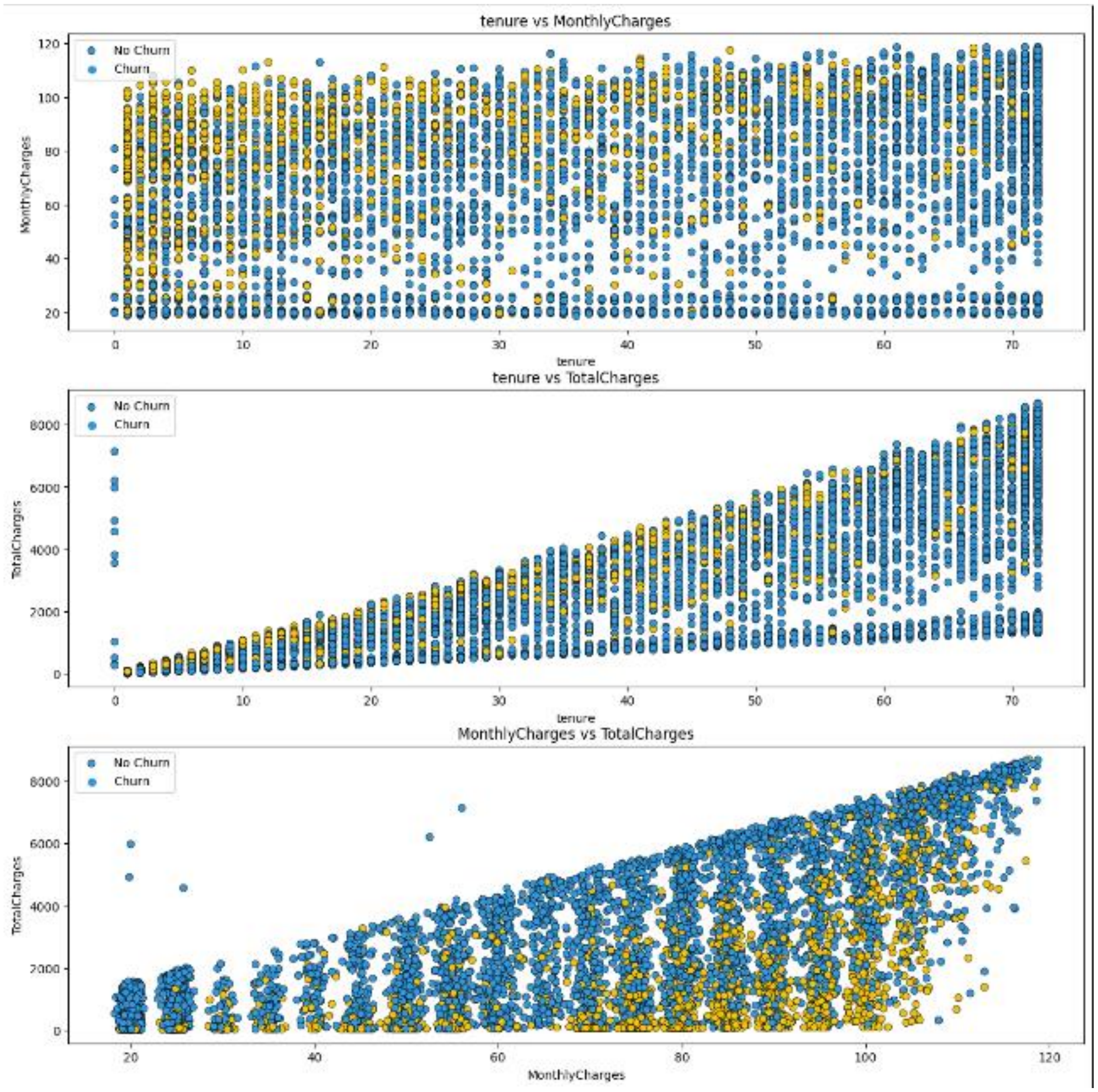
- The provided code creates a subplot with three countplots, each depicting the relationship between specific variables (l3[0], l3[1], and l3[2]) and the "Churn" variable in a dataset. It uses the Seaborn library for visualization. For each subplot, it generates a countplot, where the x-axis represents the variable of interest, and the bars are color-coded by the "Churn" variable. The count of occurrences for each category is displayed on top of the bars. The resulting subplot is divided into three sections, and each section focuses on a different variable, providing a clear visual comparison of how each variable relates to "Churn." The titles of the subplots indicate the variable being examined. This code is useful for exploring and visualizing the distribution of "Churn" across different categories of these three variables in the dataset.



- The provided code snippet generates a set of three vertically stacked subplots using Matplotlib and Seaborn for visualizing numerical features in a dataset with respect to the "Churn" variable. The loop iterates through the list of "numerical_features" and, for each feature, creates a countplot that displays the distribution of that feature, differentiated by "Churn" (No Churn and Churn), with customized colors and legend labels. The subplot title is dynamically generated to describe the feature being visualized in relation to "Churn." This code is useful for quickly visualizing how different numerical features relate to the "Churn" outcome in a clear and concise manner.



- The code snippet provided initializes a counter variable 'a' to 0 and creates a set of subplots using Matplotlib with three rows and one column, resulting in a 3x1 grid. It then iterates through pairs of numerical features in a nested loop, ensuring that 'j' is greater than 'i' and 'i' is not equal to 'j'. For each pair of features, it increments the counter 'a', selects a subplot within the grid, and generates a scatter plot using Seaborn to visualize the relationship between the selected numerical features from the DataFrame 'df1'. The hue parameter is set to 'Churn', applying different colors based on the 'Churn' column values, and a custom color palette 'colors' is used. The legend is displayed to distinguish 'No Churn' and 'Churn' data points, and a title is set for each subplot to describe the plotted features. This code efficiently creates a grid of scatter plots for pairwise feature comparisons, aiding in visualizing relationships between numerical features with respect to the 'Churn' column in the dataset.



Summary of EDA

Order / Values of features for customer churn cases :

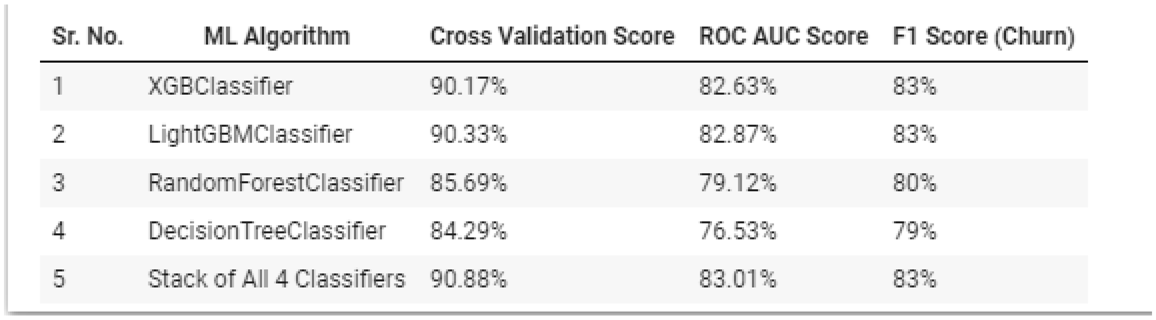
- **Categorical Features (Order) :**
  - gender : Male = Female
  - SeniorCitizen : No SeniorCitizen > SeniorCitizen
  - Partner : No Partner > Partner

  - Dependents : No Dependent > Dependent
  - PhoneService : PhoneService > No PhoneService
  - MultipleLines : MultipleLines > No MultipleLines > No PhoneService
  - InternetService : Fiber Optic > DSL > No InternetService
  - OnlineSecurity : Absent > Present > No InternetService
  - OnlineBackup : Absent > Present > No InternetService
  - DeviceProtection : Absent > Present > No InternetService
  - TechSupport : Absent > Present > No InternetService
  - StreamingTV : Absent > Present > No InternetService
  - StreamingMovies : Absent > Present > No InternetService
  - Contract : Month-to-Month > One year > Two year
  - PaperlessBilling : Present > Absent
  - PaymentMethod : Electronic check > Mailed check > Bank Transfer (automatic) > Credit Card (automatic)!

- **Numerical Features (Range) :**
  - tenure : 1 - 5 months
  - MonthlyCharges : 65 - 105
  - TotalCharges : 0 - 1000

According to the EDA, these order / range of values results in customer churn!

Algorithm Results Table



Sr. No.	ML Algorithm	Cross Validation Score	ROC AUC Score	F1 Score (Churn)
1	XGBClassifier	90.17%	82.63%	83%
2	LightGBMClassifier	90.33%	82.87%	83%
3	RandomForestClassifier	85.69%	79.12%	80%
4	DecisionTreeClassifier	84.29%	76.53%	79%
5	Stack of All 4 Classifiers	90.88%	83.01%	83%

Measures for Reducing Customer Churn & Revenue Increase

FIRST IMPRESSION IS THE LAST IMPRESSION!

- 3 types of customers should be targeted : **SeniorCitizen**, Living with a **Partner**, living all alone!
- The number of **SeniorCitizen** customers are low but their lowerlimit of **MonthlyCharges** is higher than the other customers. Thus, **SeniorCitizen** customers are ready to pay top dollar but they need to catered with that level of service. For customers with a **Partner** as well as customers living alone, they prefer services with **MonthlyCharges** below 65.

- Inorder to create a strong foundation of customers, Telco Company needs to create an easy and affordable entry point for their services. For the **tenure** of 1st 6 months, it needs to focus extensively on **OnlineSecurity**, **OnlineBackup**, **DeviceProtection** & **TechSupport** as this period is the most critical and uncertain for the customers. They must lower the churn tenure of **40 - 50** months for these services!.
- Once they build a solid pipeline of support services for customers, they need to push the usage of **MultipleLines** & **Fiber Optic** cables for the **PhoneService** & **InternetService** respectively. But the major hurdle for these 2 services is the starting point of **75+** in **MonthlyCharges**.

- Thus, they need to create combinations of options provided for **PhoneService** & **InternetService** where average of these **MonthlyCharges** will be in the range of **100 - 120** :
  - **No MultipleLines** + **OpticFiber**
  - **MultipleLines** + **DSL**
- This will increase the mean income from a user as it completely drops the option of choosing a combination of **No MultipleLines + DSL** whose mean **MonthlyCharges** is probably **60 - 70**!

- **StreamingTV** and **StreamingMovies** need to be made affordable as well as reducing it's churn **tenure**. The content of these services should be targeting all types of customers. This needs to followed up with an easy and hassle free **PaymentMethod**.

- It needs to put an end to the **Electronic check** for payment purposes due to it's high churn and focus entirely on **Bank Transfer (automatic)** & **Credit Card (automatic)**! However, they will be challenged to reduce the median churn tenure of **above 20 months** for these 2 **PaymentMethod** which is double the churn tenure of **Electronic check**.

- Lower limit of **Electronic check** is around **60** whereas that of **Bank Transfer (automatic)** & **Credit Card (automatic)** is around **20** in **MonthlyCharges**. **PaperlessBilling** is another expensive feature with a starting point of 60 whereas the other options are cheap starting at **20** in **MonthlyCharges**.

- Once the **MonthlyCharges** for any single service hits the **70** mark, customers become very conscious about their **MonthlyCharges**. Quality of service needs to be the USP of the Telco Company! These measures will push the revenue as well as improve the current value delivery process!

Conclusion

- This is a great dataset that gives an opportunity to peak into the real world business problem and can be dealt with the Data Science techniues.

- Insights gained from the EDA are very valuable for understanding the effectiveness of the existing systems that are in place. They also assist in drawing up plans & measures to counter the problems or be in an infinite loop fo improvement.

- SMOTE analysis is used for data balancing. Combinations of undersampling and oversampling can be employed as well. Undersampling was tried out for this problem but it landed the F1 Score (Churn) in the range of 60 - 70 %. There are other data balancing methods available as well.

- When it comes to model performance, feature creation by combining features was carried out however, they did not outperform the current models. Hyperparamter tuning & outlier detection could also bump up the F1 Score (Churn) & Cross Validation Score. Stack of models pipped the XGBClassifier & LGBMClassifier by a margin of 0.01 for F1 Score (Churn).

References :

- https://hockeystack.com/blog/average-customer-acquisition-cost-by-industry/
- https://www.klipfolio.com/resources/kpi-examples/call-center/subscriber-acquisition-cost
- https://www.zendesk.com/in/blog/customer-churn-rate/#georedirect
- https://www.profitwell.com/customer-churn/churn-prevention