Business Overview

The telecommunications industry is a rapidly growing sector that is constantly evolving to meet the demands of consumers. As technology advances and user behaviour changes, telecom operators face a variety of challenges that can impact their business success. In order to stay competitive and meet customer needs, it is important for telecom companies to regularly analyse their data to identify relevant problems and opportunities for improvement.

Aim:

The aim of a churn prediction notebook is to develop a machine learning model that can predict which customers are likely to churn or discontinue their use of a service or product. Churn prediction is a critical business problem for companies that operate on a subscription or recurring revenue model, such as telecommunications companies.

While the project will involve building a churn prediction model, the primary focus will be on the importance of monitoring and adapting to changes in the data that may affect the accuracy and effectiveness of the model over time. The project will also emphasize the need for a feedback loop that allows for continuous improvement and refinement of the model based on new data and changing business requirements. By highlighting these concepts, the project aims to help businesses understand the importance of staying agile and adaptable in their machine learning approaches, rather than solely focusing on the accuracy of a single model.

What is Churn Prediction?

Churn prediction is the process of identifying customers who are likely to discontinue using a service or product. In the context of the telecom sector, churn prediction is the process of identifying customers who are likely to switch to a competitor or terminate their contract with their current service provider.

Churn prediction is a critical problem in the telecom sector, as it has a significant impact on a service provider's revenue and profitability. The telecom sector is highly competitive, and service providers are constantly vying for customers' attention. It helps service providers identify customers who are at risk of leaving and take proactive measures to retain them.

Challenges

- Churn prediction is a challenging problem, as it involves analyzing large volumes of data from multiple sources. Telecom service providers generate a vast amount of data from customer interactions, network performance, and billing systems. This data is typically stored in disparate systems, making it difficult to analyze and derive insights.
- Another challenge in churn prediction is the diversity of customer behavior. Customers have different reasons for leaving a service provider, and these reasons can be difficult to predict. Some customers may leave due to poor network performance, while others may switch to a competitor offering a better deal. Predicting churn accurately requires understanding these different customer behaviour’s and identifying the most critical predictors of churn.

Business Impact of Churn Prediction

Churn prediction has a significant impact on a telecom service provider's business. A high churn rate can result in a loss of revenue and profitability. On the other hand, an effective churn prediction model can help service providers identify customers who are at risk of leaving and take proactive measures to retain them.

Here are some of the key business impacts of churn prediction:

- **Revenue Protection**: Churn prediction helps service providers protect their revenue by identifying customers who are likely to leave and taking proactive measures to retain them. This can include offering discounts, upgrading service plans, or providing additional services. By retaining customers, service providers can maintain their revenue stream and avoid the cost of acquiring new customers.
- **Customer Retention**: Churn prediction helps service providers retain their existing customers by identifying their needs and preferences. By understanding why customers leave, service providers can make improvements to their service and provide a better customer experience. This can help to build customer loyalty and increase the lifetime value of a customer.
- **Cost Reduction**: Churn prediction can help service providers reduce the cost of acquiring new customers. Acquiring new customers is more expensive than retaining existing ones, and churn prediction can help service providers focus their marketing efforts on the most valuable customers.
- **Competitive Advantage**: Churn prediction can provide a significant competitive advantage in the telecom sector. By retaining customers and improving the customer experience, service providers can differentiate themselves from their competitors. This can help to increase market share and profitability.

Approach

Data exploration

- Load the dataset and examine its structure and contents.
- Explore the distribution of the target variable (churn) and the features.

Data pre-processing

1. Handle missing values by imputing them with appropriate values.
2. Handle outliers by removing or transforming them.
3. Encode categorical variables using one-hot encoding.
4. Scale numerical variables using Standard scaler.

Model training

- Split the data into training and validation sets.
- Train logistic regression, random forest, and XGBoost models on the training set.
- Evaluate the performance of the models on the validation set using metrics such as accuracy, precision, recall, and F1 score.
- Choose the best-performing model based on the evaluation results.

Learning Outcomes

- Understanding the problem of customer churn and its impact on businesses.
- Understanding the importance of data cleaning and pre-processing in building accurate machine learning models.
- Learning how to handle missing values, outliers, and categorical variables using one-hot encoding and numerical variables using standard scaler.
- Understanding different machine learning algorithms such as logistic regression, random forest, and XGBoost, and their pros and cons in predicting churn.
- Learning how to evaluate machine learning models using metrics such as accuracy, precision, recall, and F1 score.
- Understanding the importance of monitoring data drift in machine learning models and how to use deep checks to detect it.
- Learning how to build an inference pipeline for predicting churn for new data.
- Understanding the limitations of the model and the potential impact of false positives and false negatives.
- Learning how to provide recommendations for business actions based on the model's predictions, such as targeted marketing campaigns and retention strategies.
- Understanding the iterative nature of machine learning and the need for continuous improvement and retraining of the model.

Understanding Dataset

Data exploration is a critical step in the data analysis process, where you examine the dataset to gain a preliminary understanding of the data, detect patterns, and identify potential issues that may need further investigation. Data exploration is important because it helps to provide a solid foundation for subsequent data analysis tasks, hypothesis testing and data visualization.

Data exploration is also important because it can help you to identify an appropriate approach for analysing the data.

Here are the various functions that help us explore and understand the data.

- Shape: Shape is used to identify the dimensions of the dataset. It gives the number of rows and columns present in the dataset. Knowing the dimensions of the dataset is important to understand the amount of data available for analysis and to determine the feasibility of different methods of analysis.
- Head: The head function is used to display the top five rows of the dataset. It helps us to understand the structure and organization of the dataset. This function gives an idea of what data is present in the dataset, what the column headers are, and how the data is organized.
- Tail: The tail function is used to display the bottom five rows of the dataset. It provides the same information as the head function but for the bottom rows. The tail function is particularly useful when dealing with large datasets, as it can be time-consuming to scroll through all the rows.
- Isnull: The isnull function is used to identify missing values in the dataset. It returns a Boolean value for each cell, indicating whether it is null or not. This function is useful to identify the presence of missing data, which can be problematic for regression analysis.
- Dropna: The dropna function is used to remove rows or columns with missing data. It is used to remove any observations or variables with missing data, which can lead to biased results in the regression analysis. The dropna function is used after identifying the missing data with the isnull function.

- Columns: The .columns method is a built-in function that is used to display the column names of a pandas DataFrame or Series. It returns an array-like object that contains the names of the columns in the order in which they appear in the original DataFrame or Series. It can be used to obtain a quick overview of the variables in a dataset and their names.

Code:

```
# import telecom dataset into a pandas data frame
df = pd.read_csv('TelcoCustomerChurn.csv')
#seeing what columns are there in dataset
print(df.columns)
print(df.head)
print(df.tail)
print(df.shape)
```

Output:

```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
<bound method NDFrame.head of       customerID  gender  SeniorCitizen Partner Dependents  tenure  \
0     7590-VHVEG  Female              0     Yes         No       1
1     5575-GNVDE    Male              0      No         No      34
2     3668-QPYBK    Male              0      No         No       2
3     7795-CFOCW    Male              0      No         No      45
4     9237-HQITU  Female              0      No         No       2
...          ...     ...            ...     ...        ...     ...
7038  6840-RESVB    Male              0     Yes        Yes      24
7039  2234-XADUH  Female              0     Yes        Yes      72
7040  4801-JZAZL  Female              0     Yes        Yes      11
7041  8361-LTMKD    Male              1     Yes         No       4
7042  3186-AJIEK    Male              0      No         No      66
```

```
     PhoneService     MultipleLines InternetService OnlineSecurity  ...  \
0              No  No phone service             DSL             No  ...
1             Yes                No             DSL            Yes  ...
2             Yes                No             DSL            Yes  ...
3              No  No phone service             DSL            Yes  ...
4             Yes                No     Fiber optic             No  ...
...           ...               ...             ...            ...  ...
7038          Yes               Yes             DSL            Yes  ...
7039          Yes               Yes     Fiber optic             No  ...
7040           No  No phone service             DSL            Yes  ...
7041          Yes               Yes     Fiber optic             No  ...
7042          Yes                No     Fiber optic            Yes  ...

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0                  No          No          No              No  Month-to-month
1                 Yes          No          No              No        One year
2                  No          No          No              No  Month-to-month
3                 Yes         Yes          No              No        One year
4                  No          No          No              No  Month-to-month
...               ...         ...         ...             ...             ...
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
```

```
[7043 rows x 21 columns]>
(7043, 21)
```

Checking for unique values in each column:

```
# checking unique values of each column
for col in df.columns:
    print(col, ":")
    print(df[col].unique())
    print("---------------")
```

Output:

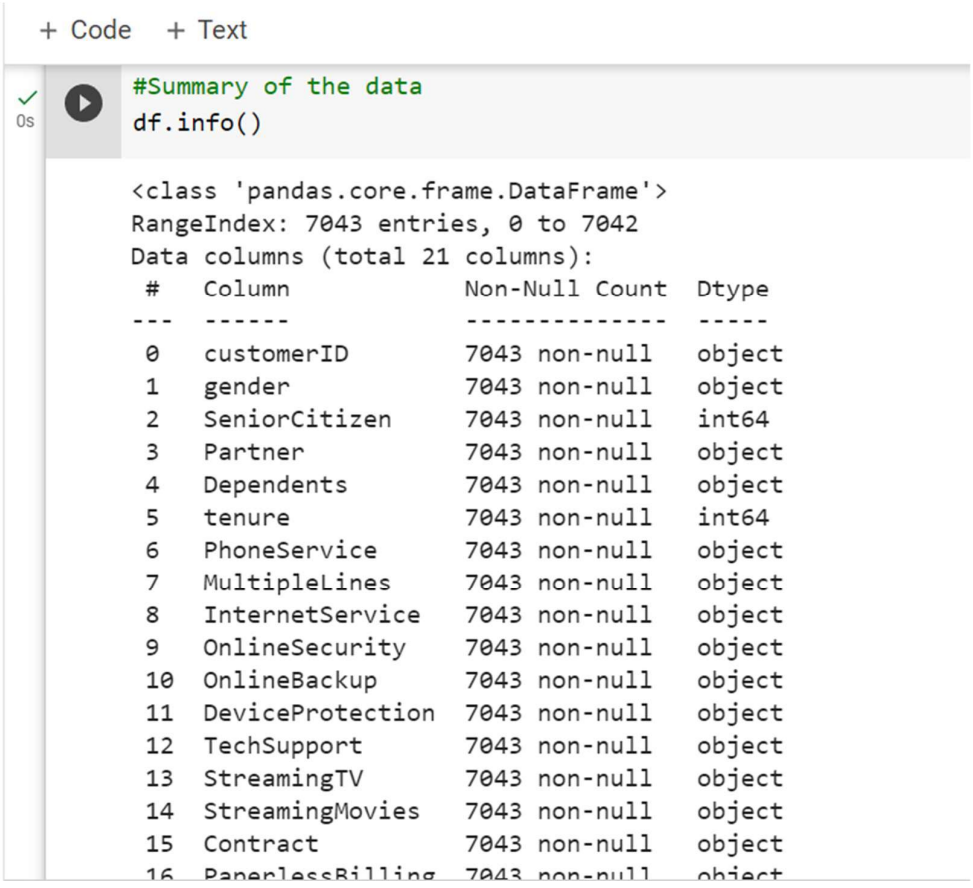
```
customerID :
['7590-VHVEG' '5575-GNVDE' '3668-QPYBK' ... '4801-JZAZL' '8361-LTMKD'
 '3186-AJIEK']
---------------
gender :
['Female' 'Male']
---------------
SeniorCitizen :
[0 1]
---------------
Partner :
['Yes' 'No']
---------------
Dependents :
['No' 'Yes']
---------------
tenure :
[ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26  0
 39]
---------------
```

```
---------------
tenure :
[ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26  0
 39]
---------------
PhoneService :
['No' 'Yes']
---------------
MultipleLines :
['No phone service' 'No' 'Yes']
---------------
InternetService :
['DSL' 'Fiber optic' 'No']
---------------
OnlineSecurity :
['No' 'Yes' 'No internet service']
---------------
OnlineBackup :
['Yes' 'No' 'No internet service']
---------------
DeviceProtection :
['No' 'Yes' 'No internet service']
---------------
```

```
---------------
Contract :
['Month-to-month' 'One year' 'Two year']
---------------
PaperlessBilling :
['Yes' 'No']
---------------
PaymentMethod :
['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
---------------
MonthlyCharges :
[29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
---------------
TotalCharges :
['29.85' '1889.5' '108.15' ... '346.45' '306.6' '6844.5']
---------------
Churn :
['No' 'Yes']
---------------
```

Data Cleaning

1. **Info –** Get summary details of the dataset.



2. **Checking Null values –** Identify the count of null values per each column.

```
#Checking null values
df.isnull().sum()
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

3. **Discrepancy in the data type –** Total charges is a numeric data type but it is in object type. Transforming the datatype for the column 'TotalCharges'

```
# we see that the total charges data type is object. Now we need to convert it into numeric data type
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
```

```
[9] #since we have converted the data type above now we will check for null valuess in TtalCharges Column
df.isnull().sum()
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

```
df[df['TotalCharges'].isnull()]
```

```
df[df['TotalCharges'].isnull()]
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Yes
753	3115-CZMZD	Male	0	No	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	DSL	Yes	...	Yes	No
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes	...	Yes	Yes
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	...	Yes	Yes
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	...	No	Yes

4. Dropna

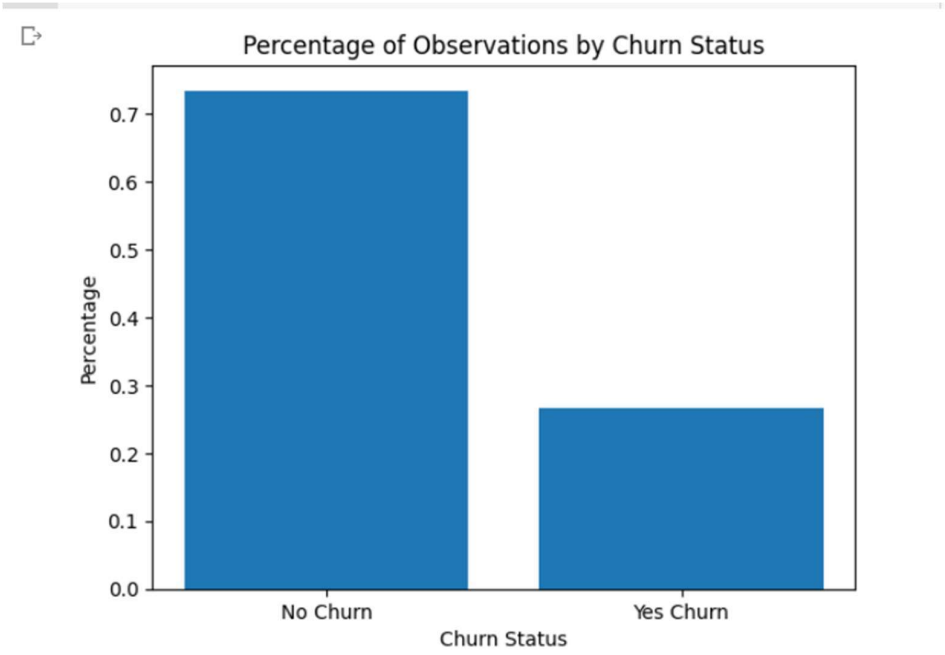
```
#Drop the rows with TotalCharges value is NUll
df.dropna(inplace=True)
```

```
[12] # Drop the customerID column from the data because it doesn't tell whether the customer will churn or not
df.drop(columns='customerID', inplace=True)
```

Exploratory Data Analysis:

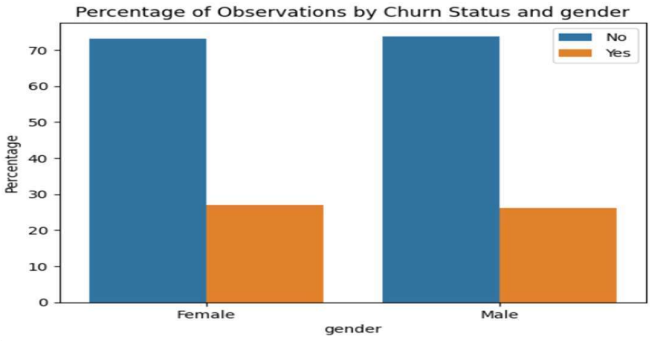
1. Percentage of Observations by Churn Status:

```
#Data Visualization
# Get the percentage of observations that correspond to each class of the response variable
no_churn_percentage = df['Churn'].value_counts()['No'] / len(df)
yes_churn_percentage = df['Churn'].value_counts()['Yes'] / len(df)

# Create a bar plot
plt.bar(['No Churn', 'Yes Churn'], [no_churn_percentage, yes_churn_percentage])

# Set the title and labels
plt.title('Percentage of Observations by Churn Status')
plt.xlabel('Churn Status')
plt.ylabel('Percentage')

# Show the plot
plt.show()
```

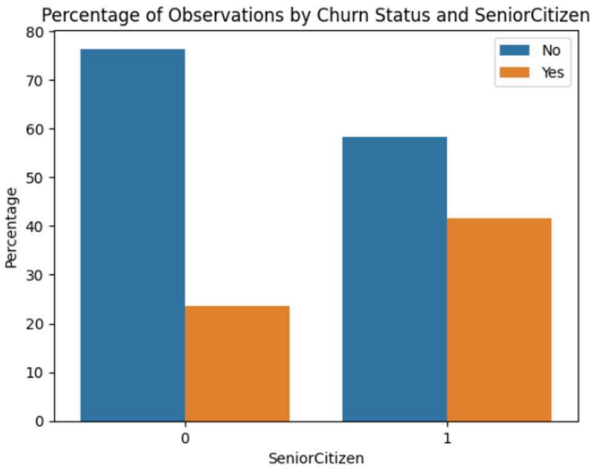


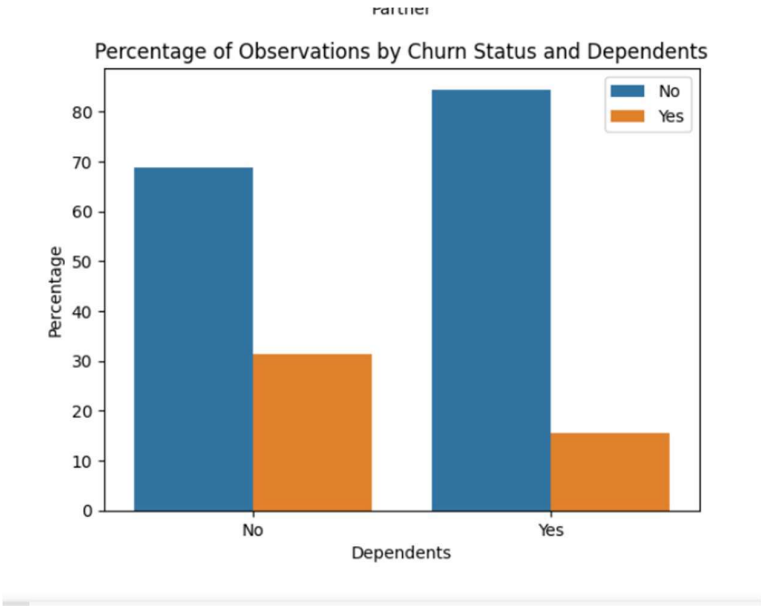
2. Percentage of Observations by Churn Status vs Gender/Senior citizen/Dependents/ Partner:

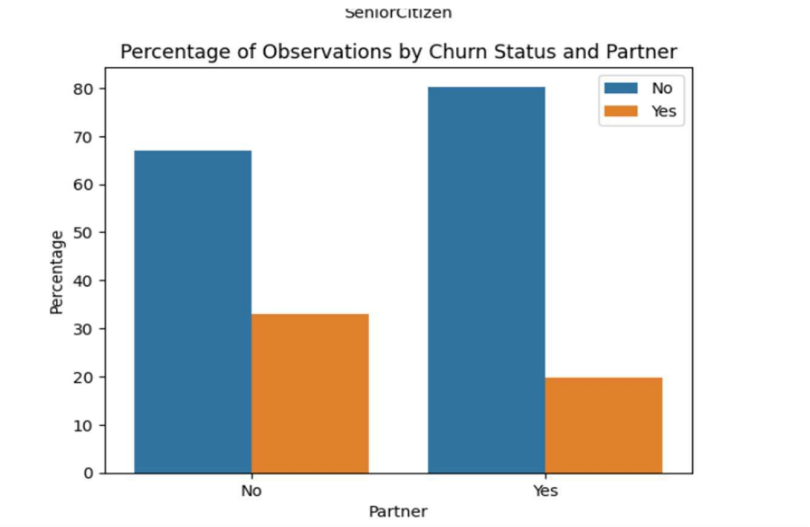
```python
# Create a function to create a group percentage bar chart for each demographic attribute
def create_group_percentage_bar_chart(attribute):
    # Get the percentage of observations that correspond to each class of the response variable for each category of the independent variable
    churn_percentages = df.groupby(attribute)['Churn'].value_counts(normalize=True).mul(100).rename('percentage').reset_index()

    # Create a bar plot
    sns.barplot(x=attribute, y='percentage', hue='Churn', data=churn_percentages)

    # Set the title and labels
    plt.title('Percentage of Observations by Churn Status and ' + attribute)
    plt.xlabel(attribute)
    plt.ylabel('Percentage')
    plt.legend()

    # Show the plot
    plt.show()

# Create group percentage bar charts for each demographic attribute
create_group_percentage_bar_chart('gender')
create_group_percentage_bar_chart('SeniorCitizen')
create_group_percentage_bar_chart('Partner')
create_group_percentage_bar_chart('Dependents')
```



✓ [17]
0s

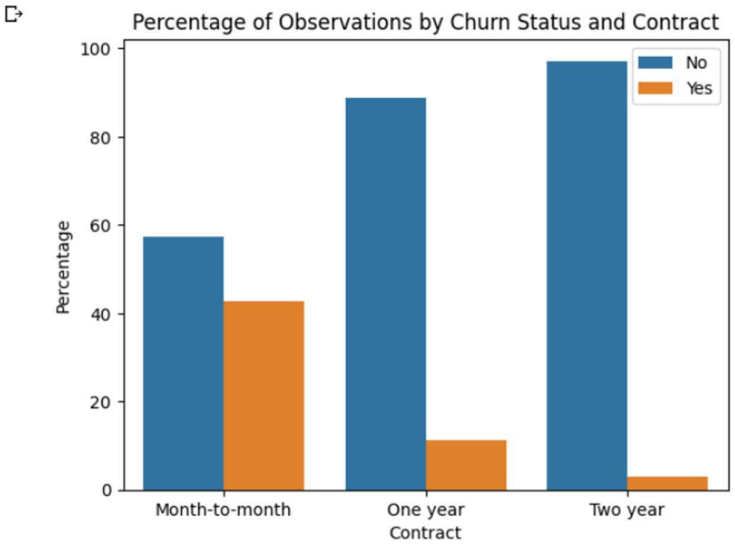


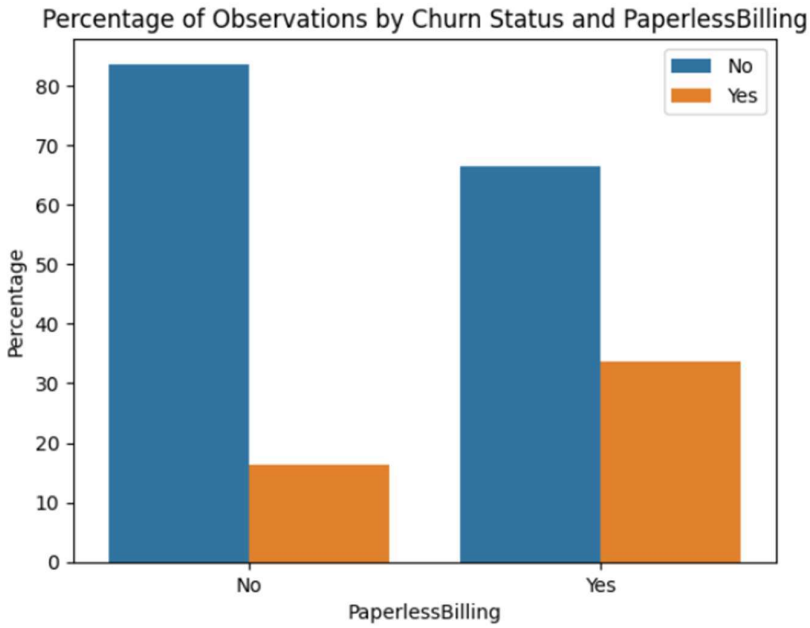


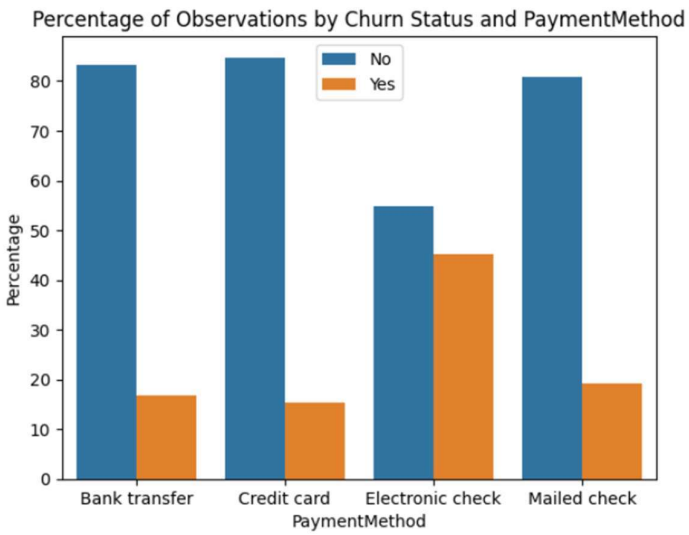


3. Percentage of Observations by Churn Status and Customer Account:

```
# Create a function to create a group percentage bar chart for each customer account information attribute
def create_group_percentage_bar_chart(attribute):
    # Get the percentage of observations that correspond to each class of the response variable for each category of the independent variable
    churn_percentages = df.groupby(attribute)['Churn'].value_counts(normalize=True).mul(100).rename('percentage').reset_index()

    # Create a bar plot
    sns.barplot(x=attribute, y='percentage', hue='Churn', data=churn_percentages)

    # Set the title and labels
    plt.title('Percentage of Observations by Churn Status and ' + attribute)
    plt.xlabel(attribute)
    plt.ylabel('Percentage')
    plt.legend()

    # Show the plot
    plt.show()

# Create group percentage bar charts for each demographic attribute
create_group_percentage_bar_chart('Contract')
create_group_percentage_bar_chart('PaperlessBilling')
create_group_percentage_bar_chart('PaymentMethod')
```

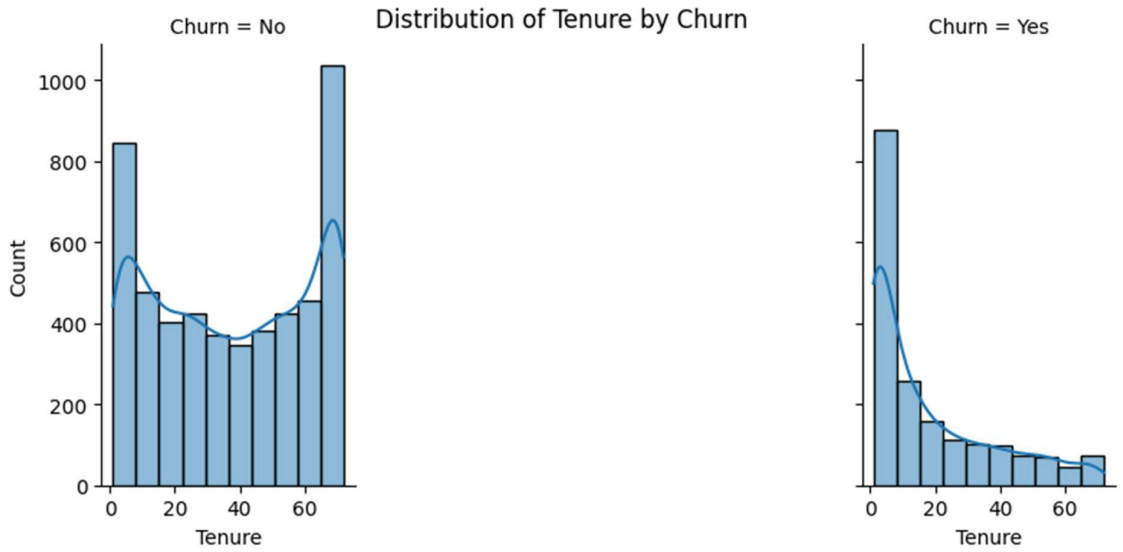

Percentage of Observations by Churn Status and Contract
No
Yes
Percentage
100
80
60
40
20
0
Month-to-month
One year
Two year
Contract

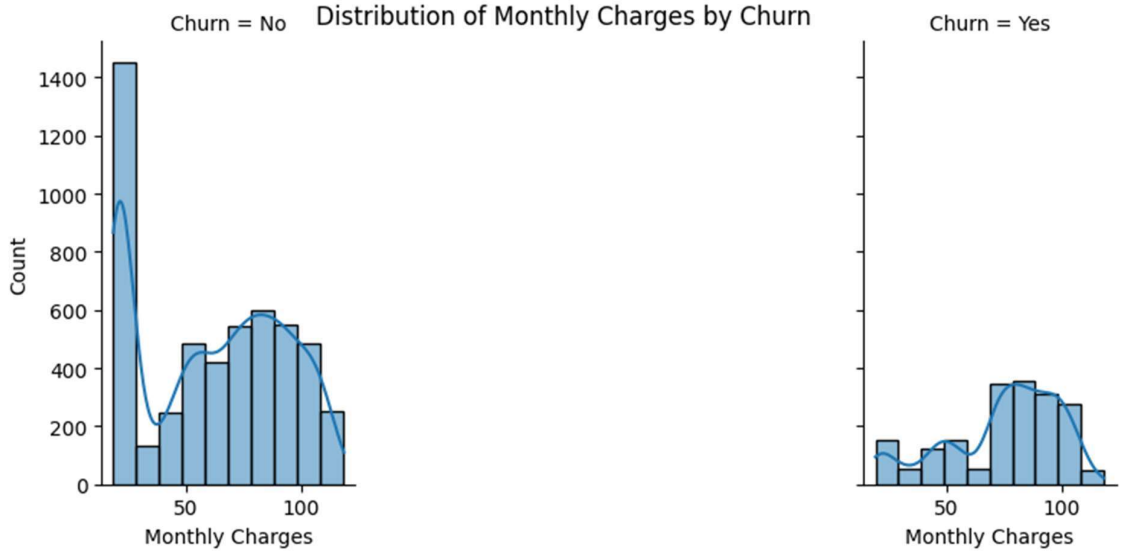

Percentage of Observations by Churn Status and PaperlessBilling
No
Yes
Percentage
80
70
60
50
40
30
20
10
0
No
Yes
PaperlessBilling

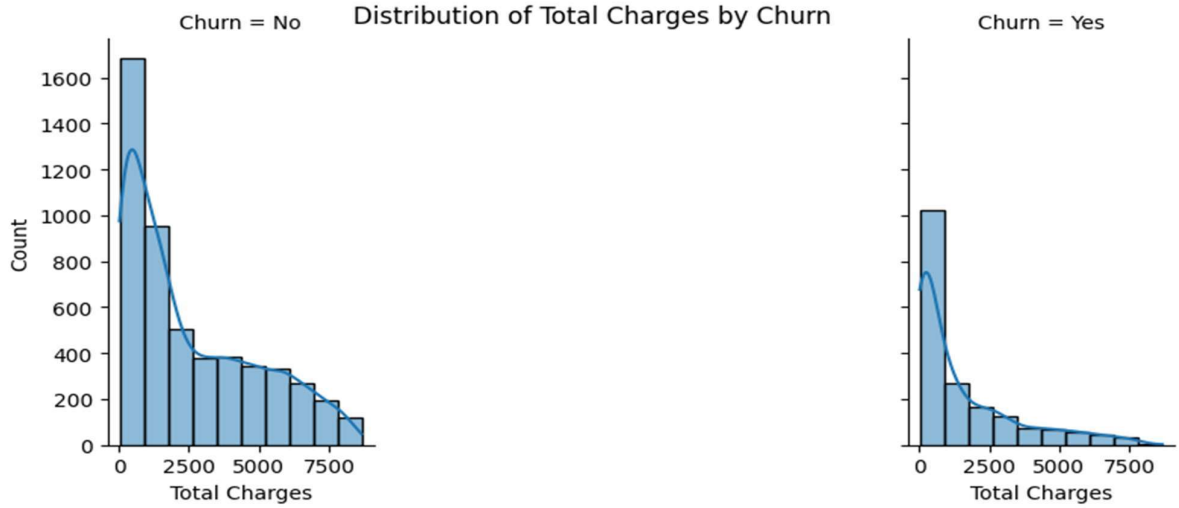


4. Histograms for the numeric attributes and churn variable:

```
# Create a DataFrame with the numeric attributes and the Churn variable
df_numeric = df[['tenure', 'MonthlyCharges', 'TotalCharges', 'Churn']]

# Create histograms for the numeric attributes by Churn
g = sns.FacetGrid(df_numeric, col='Churn', height=4)
g.map(sns.histplot, 'tenure', bins=10, kde=True)
g.set_axis_labels("Tenure", "Count")
g.fig.subplots_adjust(wspace=2, hspace=2)
g.fig.suptitle("Distribution of Tenure by Churn")
plt.show()

g = sns.FacetGrid(df_numeric, col='Churn', height=4)
g.map(sns.histplot, 'MonthlyCharges', bins=10, kde=True)
g.set_axis_labels("Monthly Charges", "Count")
g.fig.subplots_adjust(wspace=2, hspace=2)
g.fig.suptitle("Distribution of Monthly Charges by Churn")
plt.show()

g = sns.FacetGrid(df_numeric, col='Churn', height=4)
g.map(sns.histplot, 'TotalCharges', bins=10, kde=True)
g.set_axis_labels("Total Charges", "Count")
g.fig.subplots_adjust(wspace=2, hspace=2)
g.fig.suptitle("Distribution of Total Charges by Churn")
plt.show()
```

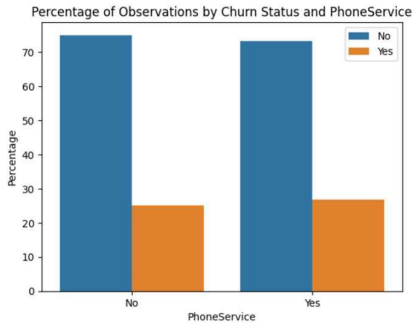

Distribution of Tenure by Churn
Churn = No
Churn = Yes
Count
Tenure
0
200
400
600
800
1000
0
20
40
60

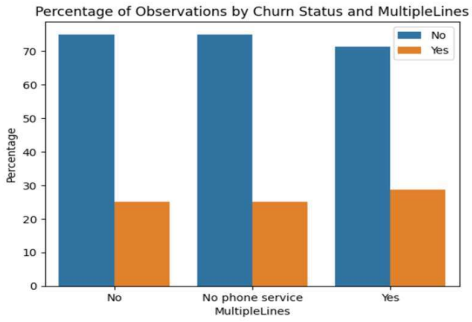

Distribution of Monthly Charges by Churn
Churn = No
Churn = Yes
Count
Monthly Charges
0
200
400
600
800
1000
1200
1400
50
100

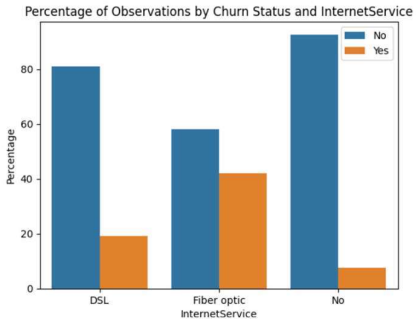

Distribution of Total Charges by Churn
Churn = No
Churn = Yes
Count
Total Charges
0
200
400
600
800
1000
1200
1400
1600
0
2500
5000
7500

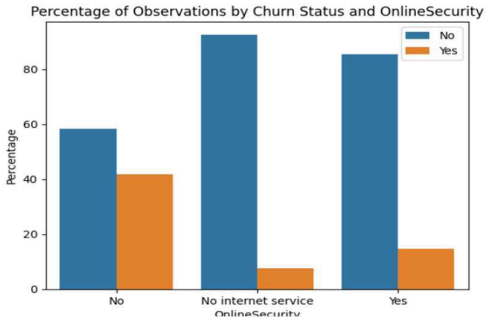
5. Group Percentage bar chart for each service information attribute:

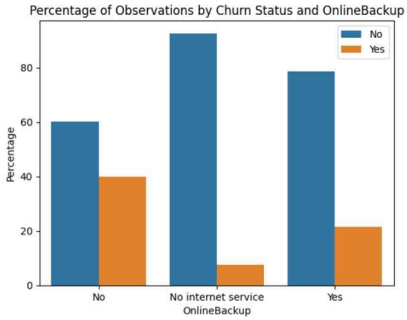
```
# Create a function to create a group percentage bar chart for each Service information attribute
def create_group_percentage_bar_chart(attribute):
    # Get the percentage of observations that correspond to each class of the response variable for each category of the independent variable
    churn_percentages = df.groupby(attribute)['Churn'].value_counts(normalize=True).mul(100).rename('percentage').reset_index()

    # Create a bar plot
    sns.barplot(x=attribute, y='percentage', hue='Churn', data=churn_percentages)

    # Set the title and labels
    plt.title('Percentage of Observations by Churn Status and ' + attribute)
    plt.xlabel(attribute)
    plt.ylabel('Percentage')
    plt.legend()

    # Show the plot
    plt.show()

# Create group percentage bar charts for each demographic attribute
create_group_percentage_bar_chart('PhoneService')
create_group_percentage_bar_chart('MultipleLines')
create_group_percentage_bar_chart('InternetService')
create_group_percentage_bar_chart('OnlineSecurity')
create_group_percentage_bar_chart('OnlineBackup')
create_group_percentage_bar_chart('DeviceProtection')
create_group_percentage_bar_chart('TechSupport')
create_group_percentage_bar_chart('StreamingTV')
create_group_percentage_bar_chart('StreamingMovies')
```

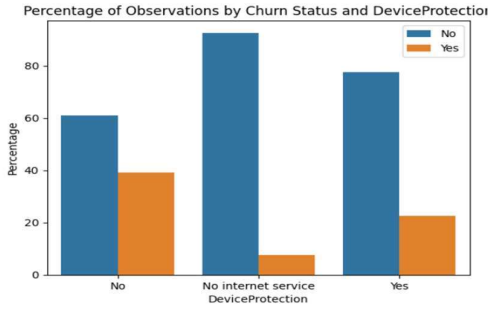


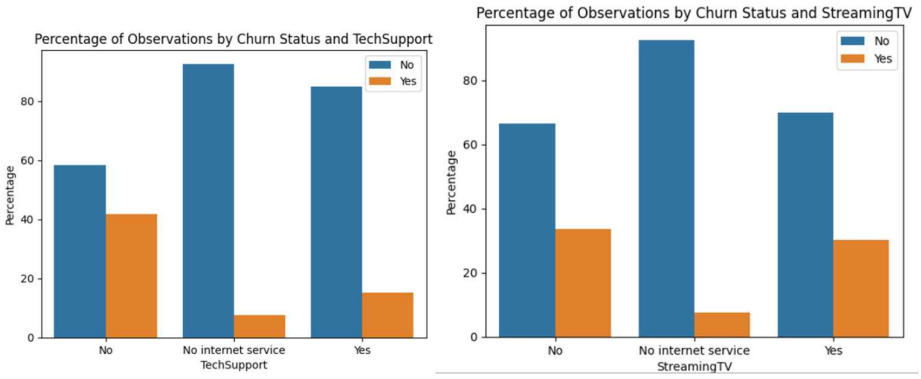


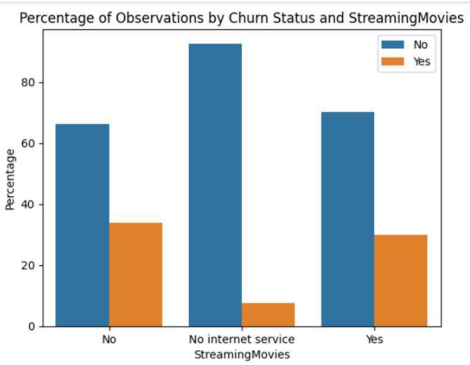












Feature Engineering:

Feature engineering is the process of transforming raw data into features that are more relevant and informative for machine learning algorithms. It is an important step in the machine learning process, as it can significantly improve the performance of the model.

Step 1 – Feature selection

```
#Feature Selection
#Checking which Features are Important
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import LabelEncoder

# Create a copy of the dataset
df_preprocessed=df.copy()

# Convert non-numeric columns to numeric columns
for col in df_preprocessed.columns:
    if df_preprocessed[col].dtype=='object':
        le=LabelEncoder()
        df_preprocessed[col]=le.fit_transform(df_preprocessed[col])

# select categorical variables excluding the response variable
categorical_variables=df_preprocessed.select_dtypes(include='int64').drop(columns=['Churn'])

# compute the mutual information score between each categorical variable and the target
feature_importance=mutual_info_classif(categorical_variables,df_preprocessed['Churn'],discrete_features=True)
feature_importance=pd.Series(feature_importance,index=categorical_variables.columns).sort_values(ascending=False)

# visualize feature importance
print(feature_importance)
```

```
Contract              0.098182
tenure                0.078950
OnlineSecurity        0.064528
TechSupport           0.062873
InternetService       0.055394
OnlineBackup          0.046659
PaymentMethod         0.044423
DeviceProtection      0.043784
StreamingMovies       0.031918
StreamingTV           0.031803
PaperlessBilling      0.019119
Dependents            0.014270
Partner               0.011383
SeniorCitizen         0.010533
MultipleLines         0.000798
PhoneService          0.000069
gender                0.000037
dtype: float64
```

Step 2 – Encoding the categorical variables

```
#Feature Engineering
#coverting categorical variables into numeric
from sklearn.preprocessing import LabelEncoder

# Create a copy of the dataset
df_transformed = df.copy()

# Encode the selected categorical features
label_encoding_columns = ['gender', 'Partner', 'Dependents', 'PaperlessBilling', 'PhoneService', 'Churn']
for col in label_encoding_columns:
    if df_transformed[col].dtype == 'object':
        le = LabelEncoder()
        df_transformed[col] = le.fit_transform(df_transformed[col])

# Show the encoded dataset
print(df_transformed.head())
```

```
   gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
0       0              0        1           0       1             0
1       1              0        0           0      34             1
2       1              0        0           0       2             1
3       1              0        0           0      45             0
4       0              0        0           0       2             1

      MultipleLines InternetService OnlineSecurity OnlineBackup  \
0  No phone service             DSL             No          Yes
1                No             DSL            Yes           No
2                No             DSL            Yes          Yes
3  No phone service             DSL            Yes           No
4                No     Fiber optic             No           No

  DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0               No          No          No              No  Month-to-month
1              Yes          No          No              No        One year
2               No          No          No              No  Month-to-month
3              Yes         Yes          No              No        One year
4               No          No          No              No  Month-to-month

   PaperlessBilling     PaymentMethod  MonthlyCharges  TotalCharges  Churn
0                 1  Electronic check           29.85         29.85      0
1                 0      Mailed check           56.95       1889.50      0
2                 1      Mailed check           53.85        108.15      1
3                 0      Bank transfer          42.30       1840.75      0
4                 1  Electronic check           70.70        151.65      1
```

Data Normalization:

Data normalization is a process of converting all the values in a dataset to a common scale. This makes it easier to compare values and to identify outliers.

There are two main types of data normalization:

- Min-max normalization: This is the most common type of data normalization. It converts all the values in a dataset to a range of 0 to 1.
- Z-score normalization: This type of data normalization converts all the values in a dataset to have a mean of 0 and a standard deviation of 1.

Data normalization is an important step in many machine learning algorithms. It helps to improve the accuracy and performance of the algorithms.

```
from sklearn.preprocessing import MinMaxScaler

# Create a copy of the dataset
df_normalized = df_transformed.copy()

# Apply Min-Max normalization to the selected columns
min_max_columns = ['tenure', 'MonthlyCharges', 'TotalCharges']
scaler = MinMaxScaler()
df_normalized[min_max_columns] = scaler.fit_transform(df_normalized[min_max_columns])

# Show the normalized dataset
print(df_normalized.head())
```

```
   gender  SeniorCitizen  Partner  Dependents    tenure  PhoneService  \
0       0              0        1           0  0.000000             0
1       1              0        0           0  0.464789             1
2       1              0        0           0  0.014085             1
3       1              0        0           0  0.619718             0
4       0              0        0           0  0.014085             1

   PaperlessBilling  MonthlyCharges  TotalCharges  Churn  ...  \
0                 1        0.115423      0.001275      0  ...
1                 0        0.385075      0.215867      0  ...
2                 1        0.354229      0.010310      1  ...
3                 0        0.239303      0.210241      0  ...
4                 1        0.521891      0.015330      1  ...

   StreamingMovies_No  StreamingMovies_No internet service  \
0                   1                                   0
1                   1                                   0
2                   1                                   0
3                   1                                   0
4                   1                                   0

   StreamingMovies_Yes  Contract_Month-to-month  Contract_One year  \
0                    0                        1                  0
1                    0                        0                  1
2                    0                        1                  0
3                    0                        0                  1
4                    0                        1                  0

   Contract_Two year  PaymentMethod_Bank transfer  PaymentMethod_Credit card  \
0                  0                            0                          0
1                  0                            0                          0
2                  0                            0                          0
3                  0                            1                          0
4                  0                            0                          0
```

Data Splitting:

```
[35] #Splitting data into train and test
     from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
                                                         random_state=40, shuffle=True)
```

Train and Evaluate Models:

The task of predicting churn of customer can be categorized as a Classification task. We tried multiple classification methods and compared which predicted the result most accurately.

1. K-nearest Neighbour Classifier
2. Logistic Regression
3. Random Forest Classifier
4. Gradient boosting classifier
5. Support vector machine

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score

# Create a list of classifiers to evaluate
classifiers = [
    KNeighborsClassifier(),
    LogisticRegression(random_state=42),
    SVC(random_state=42),
    RandomForestClassifier(random_state=42),
    GradientBoostingClassifier(random_state=42)
]

# Train and evaluate each classifier
for clf in classifiers:
    # Train the classifier
    clf.fit(X_train, y_train)
    # Predict on the testing set
    y_pred = clf.predict(X_test)
    # Evaluate the classifier's accuracy
    acc = accuracy_score(y_test, y_pred)
    # Print the classifier's name and accuracy
    print(clf.__class__.__name__, acc)
```

```
KNeighborsClassifier 0.7531285551763367
LogisticRegression 0.7923777019340159
SVC 0.7878270762229806
RandomForestClassifier 0.7667804323094426
GradientBoostingClassifier 0.7963594994311718
```

The results of the code show that the GradientBoostingClassifier has the highest accuracy, followed by the SVC, LogisticRegression, RandomForestClassifier, and KNeighborsClassifier. This suggests that the GradientBoostingClassifier is the best classifier for this dataset.

Hyperparameter Tuning:

Hyperparameter tuning is the process of finding the optimal values for the hyperparameters of a machine learning model. Hyperparameters are the parameters that control the learning process of the model, but they are not learned from the data.

There are many different hyperparameters that can be tuned, such as the number of features to use, the learning rate, and the regularization strength. The optimal values for these hyperparameters will vary depending on the dataset and the machine learning algorithm.

There are a number of different techniques that can be used for hyperparameter tuning, such as:

- Grid search: This is a brute-force approach that involves trying all possible combinations of hyperparameter values.
- Random search: This is a more efficient approach that involves randomly sampling hyperparameter values from a pre-defined distribution.
- Bayesian optimization: This is an approach that uses Bayesian inference to find the optimal hyperparameter values.

Here we are using Grid Search Hyperparameter tuning.

```
KNeighborsClassifier
Best parameters: {'n_neighbors': 7}
Test Accuracy: 0.7627986348122867
Mean validation accuracy: 0.771709848287274
---------------
LogisticRegression
Best parameters: {'C': 10}
Test Accuracy: 0.7957906712172924
Mean validation accuracy: 0.8064059282174879
---------------
SVC
Best parameters: {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
Test Accuracy: 0.7906712172923777
Mean validation accuracy: 0.8033731125839726
---------------
RandomForestClassifier
Best parameters: {'max_depth': 10, 'n_estimators': 100}
Test Accuracy: 0.7901023890784983
Mean validation accuracy: 0.8064053886345854
---------------
GradientBoostingClassifier
Best parameters: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 300}
Test Accuracy: 0.7963594994311718
Mean validation accuracy: 0.8048889808178277
---------------
```

Visualization:

We have taken the important features that are most corelated to the 'Churn' value.