Project-2: Final Project on "customer_churn" Dataset

By Vivek Prakash Upreti

thevivekai@gmail.com

Problem Statement

- You are the Data Scientist at a telecom company "Neo" whose customers are churning out to its competitors. You have to analyse the data of your company and find insights and stop your customers from churning out to other telecom companies.

Task 1: Data Manipulation

In [100]:
```
import pandas as pd
data = pd.read_csv('Customer_Churn.csv')
data
```

Out[100]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No
...	...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	No
7041	8361-LTMKD	Male	1	Yes	No	4	Yes	Yes	Fiber optic	No	...	No
7042	3186-AJIEK	Male	0	No	No	66	Yes	No	Fiber optic	Yes	...	Yes

7043 rows × 21 columns

In [101]:
```
data.shape
```

Out[101]: (7043, 21)

In [102]:
```
# Extract the 5th column & store it in 'customer_5'
customer_5 = data['Dependents']

# Extract the 15th column & store it in 'customer_15
customer_15 = data.iloc[:, 14]

# Extract all the male senior citizens whose Payment Method is Electronic check & store the result in 'senior_male_electronic
senior_male_electronic = data[
    (data['gender'] == 'Male') &
    (data['SeniorCitizen'] == 1) &
    (data['PaymentMethod'] == 'Electronic check')
]

# Extract all those customers whose tenure is greater than 70 months or their Monthly charges is more than 100$ & store the result in 'customer_
customer_total_tenure = data[
    (data['tenure'] > 70) |
    (data['MonthlyCharges'] > 100)
]

# Extract all the customers whose Contract is of two years, payment method is Mailed check the value of Churn is 'Yes' & store the result in 'tw
```

```
two_mail_yes = data[
    (data['Contract'] == 'Two year') &
    (data['PaymentMethod'] == 'Mailed check') &
    (data['Churn'] == 'Yes')
]

# Extract 333 random records from the customer_churn dataframe(since i load the customer_churn data in varible name as "data" so we use data) &
customer_333 = data.sample(n=333)

# Get the count of different levels from the 'Churn' colum
churn_counts = data['Churn'].value_counts()
```

Conclusion of Task 1: Data Manipulation

- The `customer_churn` dataset was loaded into a pandas DataFrame.
- Specific columns (`Dependents` and the 15th column) were extracted.
- Subsets of the data were created based on conditions (male senior citizens with Electronic check payment, customers with tenure > 70 or Monthly Charges > 100, and customers with a two-year contract, Mailed check payment, and Churn = 'Yes').
- A random sample of 333 records was extracted.
- The counts of different levels in the 'Churn' column were obtained.

Task 2: Data Visualization

In [103]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

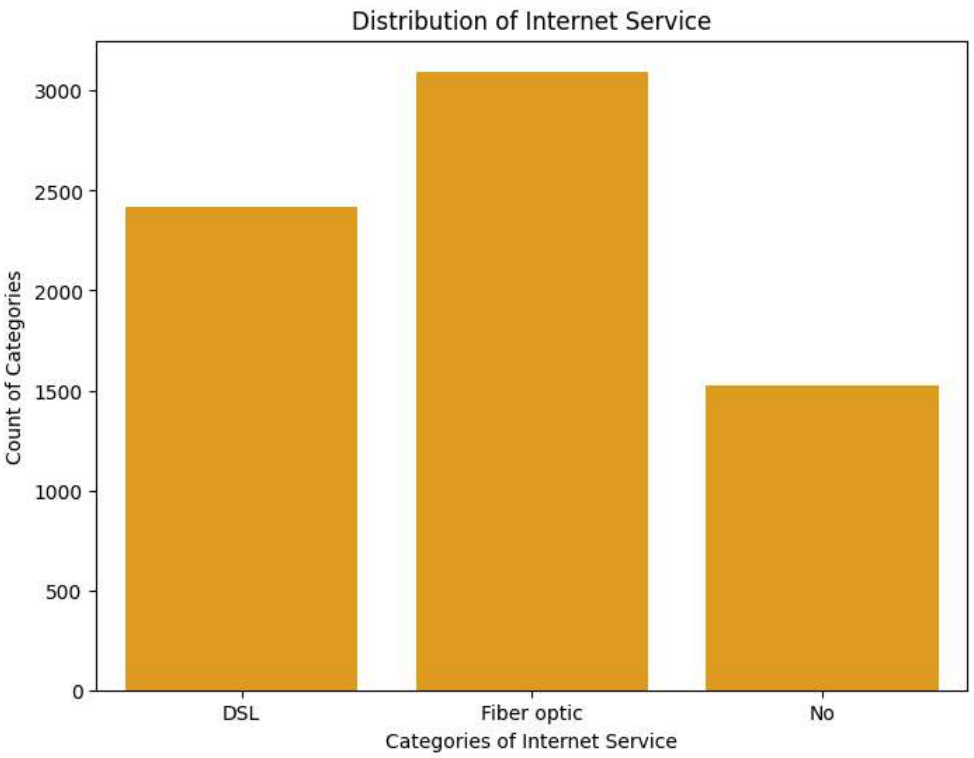
2.1 Build a bar-plot for the 'InternetService' column:

In [104]:
```
# Set the figure size for better readability
plt.figure(figsize=(8, 6))

# Create the countplot using seaborn
sns.countplot(x='InternetService', data=data, color='orange')

# Set the labels and title as requested
plt.xlabel('Categories of Internet Service')
plt.ylabel('Count of Categories')
plt.title('Distribution of Internet Service')

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



2.2 Build a histogram for the 'tenure' column

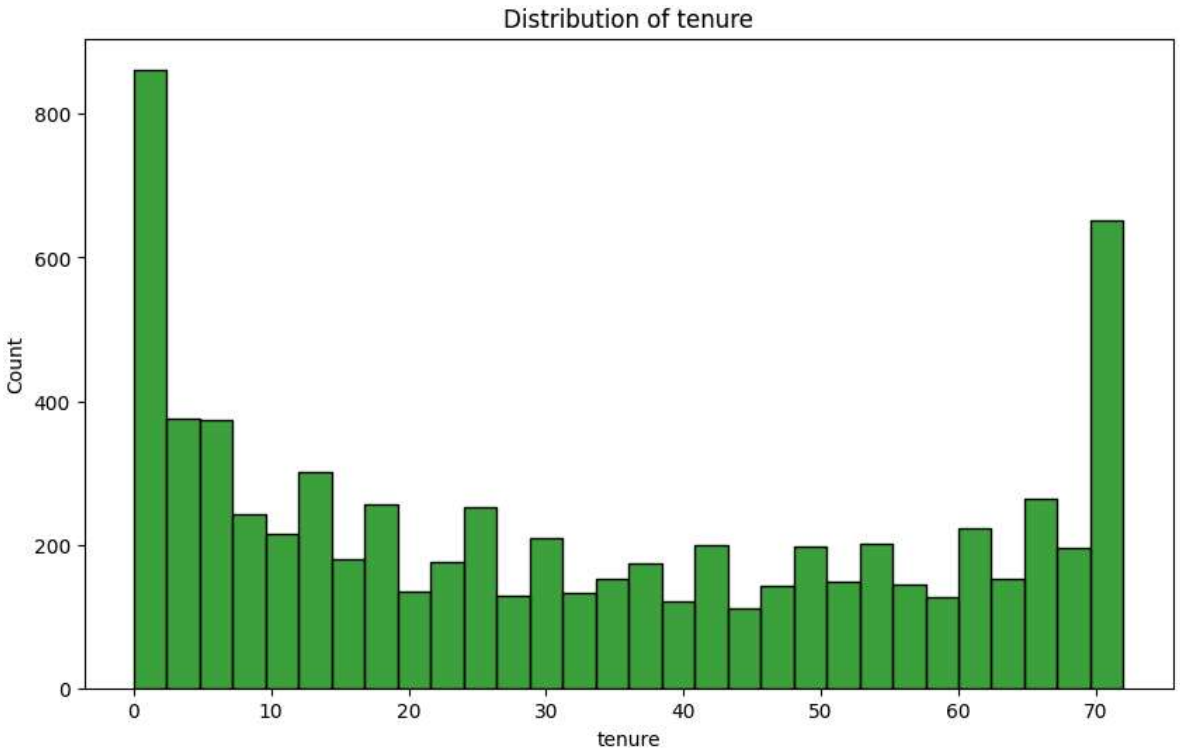
In [105]:
```
# Set the figure size
plt.figure(figsize=(10, 6))

# Create the histogram using seaborn
sns.histplot(data=data, x='tenure', bins=30, color='green')

# Assign the title
```

```
plt.title('Distribution of tenure')

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



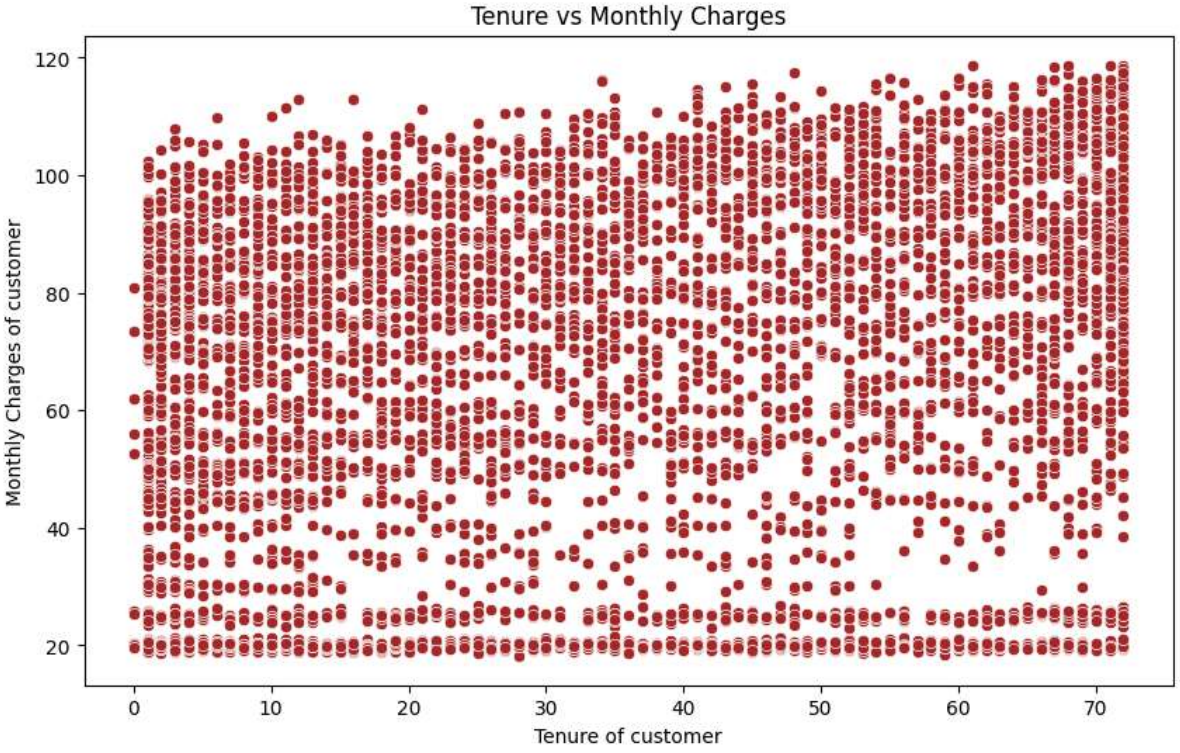
2.3 Build a scatter-plot between 'MonthlyCharges' & 'tenure'. Map 'MonthlyCharges' to the y-axis & 'tenure' to the 'x-axis':

In [106]:
```
# Set the figure size
plt.figure(figsize=(10, 6))

# Create the scatter-plot using seaborn
sns.scatterplot(x='tenure', y='MonthlyCharges', data=data, color='brown')

# Assign the labels and title
plt.xlabel('Tenure of customer')
plt.ylabel('Monthly Charges of customer')
plt.title('Tenure vs Monthly Charges')

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



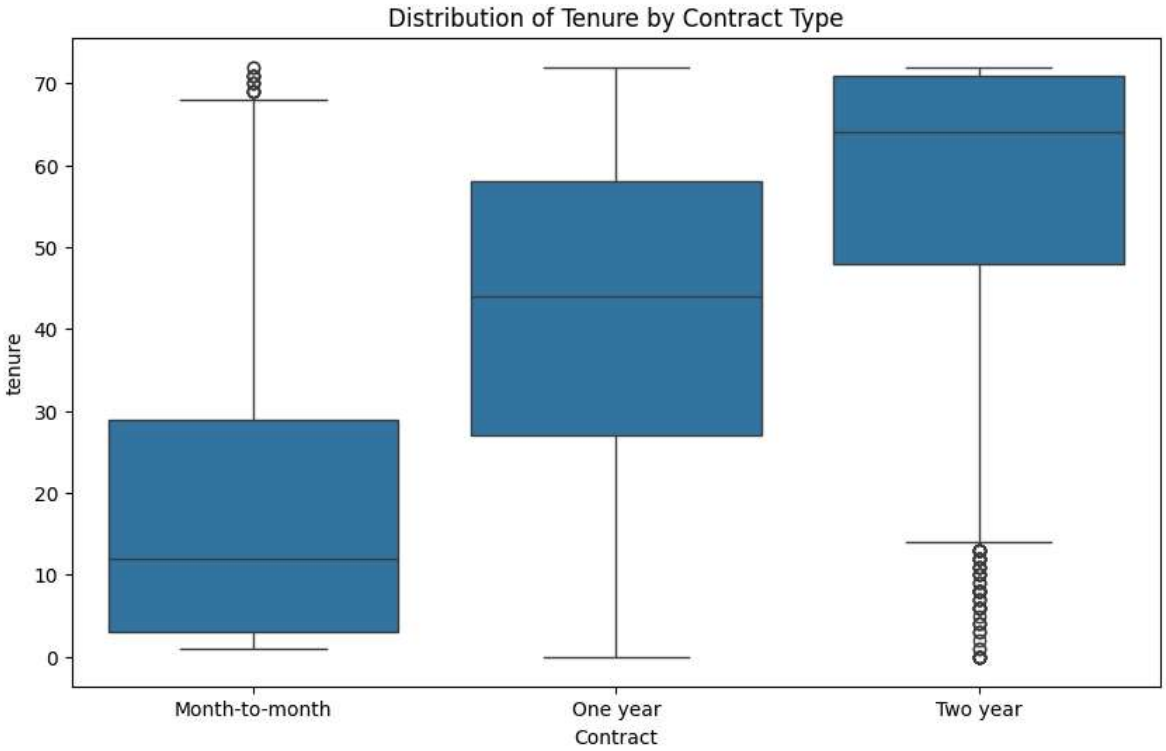
2.4 Build a box-plot between 'tenure' & 'Contract'. Map 'tenure' on the y-axis & 'Contract' on the x-axis

In [107]:
```
# Set the figure size
plt.figure(figsize=(10, 6))

# Create the box-plot using seaborn
sns.boxplot(x='Contract', y='tenure', data=data)

# Add a title for clarity
plt.title('Distribution of Tenure by Contract Type')

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



Conclusion of Task 2: Data Visualization

- A bar plot was created to visualize the distribution of 'InternetService'.
- A histogram was generated to show the distribution of 'tenure'.
- A scatter plot was created to explore the relationship between 'MonthlyCharges' and 'tenure'.
- A box plot was built to visualize the distribution of 'tenure' across different 'Contract' types.

Task 3: Linear Regression

3.1 Preparing Data

In [108]:
```
x = pd.DataFrame(data['tenure'])
y = pd.DataFrame(data['MonthlyCharges'])
```

In [109]:
```
x.size,y.size
```

Out[109]: (7043, 7043)

3.2 Split the dataset into 70:30 ratio

In [110]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=1)
```

In [111]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[111]: ((4930, 1), (2113, 1), (4930, 1), (2113, 1))

3.3 Creating the Model

In [112]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x_train,y_train)
```

Out[112]:
```
LinearRegression
LinearRegression()
```

In [113]:
```
# intercept value(c) in 'y = mx + c'
print('intercept Value(c):',lin_reg.intercept_)

# Coefficient value(m) in 'y = mx + c'
print('Coefficient Value(m):',lin_reg.coef_)
```

```
intercept Value(c): [54.80186731]
Coefficient Value(m): [[0.3100844]]
```

3.4 Predicting Using the Testing Dataset

In [114]:
```
y_pred = lin_reg.predict(x_test)
y_pred = pd.DataFrame(y_pred,columns=['Predicted'])
```

In [115]:
```
y_pred
```

Out[115]:

	Predicted
0	67.515328
1	75.267438
2	58.522880
3	56.352289
4	57.902711
...	...
2108	57.902711
2109	58.212796
2110	61.313640
2111	76.507775
2112	68.755665

2113 rows × 1 columns

In [116]:
```
y_test
```

Out[116]:

	MonthlyCharges
3381	79.85
6180	102.40
4829	45.00
3737	50.60
4249	65.90
...	...
3934	40.25
1351	20.45
2048	96.80
6218	58.40
4297	73.55

2113 rows × 1 columns

3.5 Evaluate the Model

In [117]:
```
from sklearn import metrics
import numpy as np
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))

error = np.sqrt(metrics.mean_squared_error(y_test,y_pred))
```

```
Root Mean Squared Error: 29.15550989402921
```

Conclusion of Task 3: Linear Regression

- Data for 'tenure' (x) and 'MonthlyCharges' (y) was prepared.
- The dataset was split into 70:30 training and testing sets.
- A Linear Regression model was created and trained.

- Predictions were made using the test data.
- The model was evaluated using Root Mean Squared Error (RMSE) = 29.15

Task 4: Logistic Regression

4.1 Simple Logistic Regression Model where x = 'MonthlyCharges' and y = 'Churn'

4.1.1 Preparing Data

In [118]:
```
x_log = pd.DataFrame(data['MonthlyCharges'])
y_log = pd.DataFrame(data['Churn'])
```

In [119]:
```
x_log.size,y_log.size
```

Out[119]: (7043, 7043)

4.1.2 Split the Dataset in 65:35 Ratio

In [120]:
```
x_log_train, x_log_test, y_log_train, y_log_test = train_test_split(x_log,y_log,test_size = 0.35, random_state=1)
```

In [121]:
```
x_log_train.shape,x_log_test.shape,y_log_train.shape,y_log_test.shape
```

Out[121]: ((4577, 1), (2466, 1), (4577, 1), (2466, 1))

4.1.3 Creating the Model

In [122]:
```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(x_log_train,y_log_train)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[122]: LogisticRegression

LogisticRegression()

4.1.4 Predicting Using the test Data

In [123]:
```
y_log_pred = log_reg.predict(x_log_test)
```

In [124]:
```
y_log_pred
```

Out[124]: array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)

4.1.5 Evaluate the model

In [125]:
```
# Confusion Matrix
from sklearn.metrics import confusion_matrix
sim_confusion_matrix = confusion_matrix(y_log_test,y_log_pred)
print(sim_confusion_matrix)
```

```
[[1840    0]
 [ 626    0]]
```

In [126]:
```
# Calculate the accuracy
print('Accuracy: %d', (log_reg.score(x_log_test,y_log_test)))
```

```
Accuracy: %d 0.7461476074614761
```

Conclusion of Task 4.1: Simple Logistic Regression

- Data for 'MonthlyCharges' (x) and 'Churn' (y) was prepared for simple logistic regression.
- The dataset was split into 65:35 training and testing sets.
- A Simple Logistic Regression model was created and trained.
- Predictions were made using the test data.
- The model was evaluated using a confusion matrix with accuracy = 74.6%

4.2 Multiple Logistic Regression Model Where x = 'tenure & MonthlyCharges' y = 'Churn'

4.2.1 Preparing Data

In [127]:
```
x_mul_log = pd.DataFrame(data[['tenure','MonthlyCharges']])
y_mul_log = pd.DataFrame(data['Churn'])
```

In [128]:
```
x_mul_log.shape,y_mul_log.shape
```

Out[128]:
```
((7043, 2), (7043, 1))
```

4.2.2 Split the Dataset in 80:20 Ratio

In [129]:
```
x_mul_log_train, x_mul_log_test, y_mul_log_train, y_mul_log_test = train_test_split(x_mul_log,y_mul_log,test_size = 0.2, random_state=1)
```

In [130]:
```
x_mul_log_train.shape, x_mul_log_test.shape, y_mul_log_train.shape, y_mul_log_test.shape
```

Out[130]:
```
((5634, 2), (1409, 2), (5634, 1), (1409, 1))
```

4.2.3 Creating The Model

In [131]:
```
from sklearn.linear_model import LogisticRegression
mul_log_reg = LogisticRegression()
mul_log_reg.fit(x_mul_log_train,y_mul_log_train)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[131]:
```
LogisticRegression()
```

4.2.4 Predicting with test data

In [132]:
```
y_mul_log_pred = mul_log_reg.predict(x_mul_log_test)
```

In [133]:
```
y_mul_log_pred
```

Out[133]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'Yes'], dtype=object)
```

4.2.5 Evaluate the **Model**

In [134]:
```
# Confusion Matrix
confusion_matrix_mul = confusion_matrix(y_mul_log_test,y_mul_log_pred)
print(confusion_matrix_mul)
```

```
[[965  96]
 [190 158]]
```

In [135]:
```
# Calculate the accuracy
print('Accuracy: %d', (mul_log_reg.score(x_mul_log_test,y_mul_log_test)))
```

```
Accuracy: %d 0.7970191625266146
```

Conclusion of Task 4.2: Multiple Logistic Regression

- Data for 'tenure' and 'MonthlyCharges' (x) and 'Churn' (y) was prepared for multiple logistic regression.
- The dataset was split into 80:20 training and testing sets.
- A Multiple Logistic Regression model was created and trained.
- Predictions were made using the test data.
- The model was evaluated using a confusion matrix with accuracy = 79.7%

Task 5: Decision Tree

5.1 Preparing Data

In [136]:
```
X = pd.DataFrame(data['tenure'])
Y = pd.DataFrame(data['Churn'])
```

In [137]:
```
X.shape,Y.shape
```

Out[137]:
```
((7043, 1), (7043, 1))
```

5.2 Split the data in 80:20 ratio

In [138]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=1)
```

In [139]:
```
X_train.shape,X_test.shape,Y_train.shape,Y_test.shape
```

Out[139]:
```
((5634, 1), (1409, 1), (5634, 1), (1409, 1))
```

5.3 Creating The Model

In [140]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train,Y_train)
```

Out[140]:
```
DecisionTreeClassifier()
```

5.4 Predicting with test data

In [141]:
```
Y_pred = dtc.predict(X_test)
```

In [142]:
```
Y_pred
```

Out[142]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype=object)
```

5.5 Evaluate the model

In [143]:
```
# Confusion Matrix
confusion_matrix_dtc = confusion_matrix(Y_test,Y_pred)
print(confusion_matrix_dtc)
```

```
[[983  78]
 [254  94]]
```

In [144]:
```
# Calculate the accuracy
print('Accuracy: %d', (dtc.score(X_test,Y_test)))
```

```
Accuracy: %d 0.7643718949609652
```

Conclusion of Task 5: Decision Tree

- Data for 'tenure' (X) and 'Churn' (Y) was prepared for decision tree classification.
- The dataset was split into 80:20 training and testing sets.
- A Decision Tree Classifier model was created and trained.
- Predictions were made using the test data.
- The model was evaluated using a confusion matrix with accuracy = 76.4%

Task 6: Random Forest

6.1 Preparing Data

In [145]:
```
X_ran = pd.DataFrame(data[['tenure','MonthlyCharges']])
Y_ran = pd.DataFrame(data['Churn'])
```

In [146]:
```
X_ran.shape,Y_ran.shape
```

Out[146]:
```
((7043, 2), (7043, 1))
```

6.2 Split the data in ratio 70:30 ratio

In [147]:
```
X_ran_train,X_ran_test,Y_ran_train,Y_ran_test = train_test_split(X_ran,Y_ran,test_size=0.3,random_state=1)
```

In [148]:
```
X_ran_train.shape,X_ran_test.shape,Y_ran_train.shape,Y_ran_test.shape
```

Out[148]:
```
((4930, 2), (2113, 2), (4930, 1), (2113, 1))
```

6.3 Create the Model

In [149]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_ran_train,Y_ran_train)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/base.py:1389: DataConversionWarning: A column-vector y was passed when a 1d array was expected. P
lease change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

Out[149]:
```
RandomForestClassifier()
```

6.4 Predicting with test data

In [150]:
```
Y_ran_pred = rfc.predict(X_ran_test)
```

6.5 Evaluate the Model

In [151]:
```
# Confusion Matrix
confusion_matrix_rfc = confusion_matrix(Y_ran_test,Y_ran_pred)
print(confusion_matrix_rfc)
```

```
[[1357  228]
 [ 275  253]]
```

In [152]:
```
# Calculate the accuracy
print('Accuracy: %d', (rfc.score(X_ran_test,Y_ran_test)))
```

```
Accuracy: %d 0.7619498343587316
```

Conclusion of Task 6: Random Forest

- Data for 'tenure' and 'MonthlyCharges' (X) and 'Churn' (Y) was prepared for random forest classification.
- The dataset was split into a 70:30 training and testing ratio.
- A Random Forest Classifier model was created and trained.
- Predictions were made using the test data.
- The model was evaluated using a confusion matrix with accuracy = 76.1%

Overall Conclusion and Model Comparison

Here's a summary of the tasks performed and a comparison of the model accuracies:

- **Task 1: Data Manipulation**
  - Data loaded, specific columns and rows extracted, random sample taken, and churn counts obtained.
- **Task 2: Data Visualization**
  - Visualizations created for 'InternetService', 'tenure', 'MonthlyCharges' vs 'tenure', and 'tenure' vs 'Contract'.
- **Task 3: Linear Regression**
  - Model built to predict 'MonthlyCharges' based on 'tenure'.
  - Evaluated using RMSE: **29.16** (approximately)
- **Task 4.1: Simple Logistic Regression**
  - Model built to predict 'Churn' based on 'MonthlyCharges'.
  - Evaluated using Accuracy: **74.6%**
- **Task 4.2: Multiple Logistic Regression**
  - Model built to predict 'Churn' based on 'tenure' and 'MonthlyCharges'.
  - Evaluated using Accuracy: **79.7%**
- **Task 5: Decision Tree**
  - Model built to predict 'Churn' based on 'tenure'.
  - Evaluated using Accuracy: **76.4%**
- **Task 6: Random Forest**
  - Model built to predict 'Churn' based on 'tenure' and 'MonthlyCharges'.
  - Evaluated using Accuracy: **76.2%**

Model Comparison:

Based on the accuracy scores for predicting 'Churn':

- The **Multiple Logistic Regression model (79.7%)** performed best among the classification models.
- The Simple Logistic Regression model had the lowest accuracy (74.6%).
- Decision Tree (76.4%) and Random Forest (76.2%) had similar accuracies, which were better than simple logistic regression but not as good as multiple logistic regression.

The choice of the best model would depend on further evaluation metrics (like precision, recall, F1-score) and the specific goals of the churn prediction.