SUSANTA DHURUA BSC STATISTICS 1811108010005

Churn analysis prediction

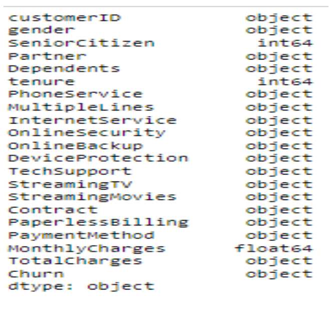
Abstract: The analysis of telecom data for customer churn has become an imperative task in the highly competitive telecommunications market. In this study, we aimed to identify the key factors that influence customer churn and assess their impact on customer retention. Through comprehensive data analysis, we examined various variables related to customer behavior, usage patterns, and demographics. The results shed light on the significant factors that contribute to customer churn, enabling telecom companies to devise effective strategies for customer retention and enhance their overall service offerings. By understanding these insights, telecom operators can better cater to customer needs, reduce churn rates, and foster long-term customer loyalty.

Introduction: Churn analysis, generally refers to the process of examining and understanding the rate at which customers discontinue their association with a particular service or product, commonly known as "Churning". In various industries, including telecommunications, finance and subscription based services, churn analysis plays a crucial role in assessing customer retention and loyalty. By analyzing churn patterns and identifying the factors that lead customer to leave, businesses can develop targeted strategies to reduce churn and retain valuable customers, ultimately improving their overall profitability and success.

Data Description: The dataset contains 7043 rows and 21 columns/attributes. The name and description of 21 columns are

'Customer ID': It contains the customer id
'Gender': Whether the customer is male or female
'Senior Citizen': whether the customer is a senior citizen or not
'Partner': Whether the customer have partner or not
'Dependents': Whether the customer is dependent or not
'Tenure': it refer to the period or duration of time that a person hold a particular plans
'Phone Service': Whether the customer uses the phone services
'Multiple Lines': Whether the customer have multiple lines who are using phone services
'Internet Service': Which internet services does the customer using
'Online Security': Whether the customer taken the online security subscription
'Online Backup': Whether the customer have opted the online backup
'Device Protection': Whether the customer have applied the device protection
'Tech Support': Whether the customer have applied for the tech support
'Streaming TV': Whether the customer is streaming TV
'Streaming Movies': Whether the customer is streaming movies
'Contract': How many months contract does have the customer with the organization through plans?
'Paperless Billing': Paperless billing yes or not
'Payment Method': Customer does their payment by Electronic mode or the cash on delivery
'Monthly Charges': Monthly charges of the plans
'Total Charges': Total charges given by the customer
'Churn': Whether the customer is churn or not

Operations:
Checking the data types: =>



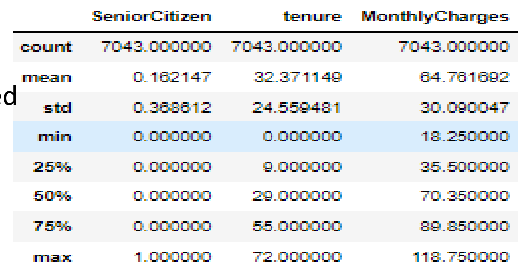
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

Upon inspecting the attribute data types,
We find that "senior citizen" is a
Categorical variable represented by 0 and 1.

Data summary: =>

Senior citizen is a categorical variable represented
By 0 and 1, thus included in the
Descriptive statistics.

Interpretation: Since tenure is given in months,
The table shows that 75% of customers
Have a tenure of less than 55 months.



	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

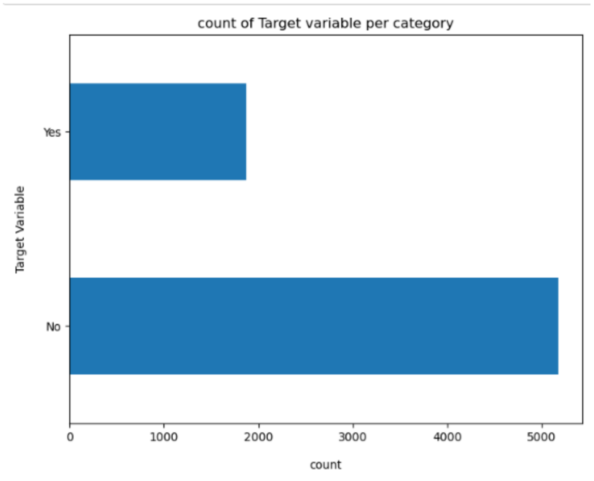
Visualize the churn value: =>

Through visualization,
We can determine the churn values:

Category No: 5174
Category Yes: 1869

The churn ratio is as follows:

'No': 73.463013%
'Yes': 26.536987%

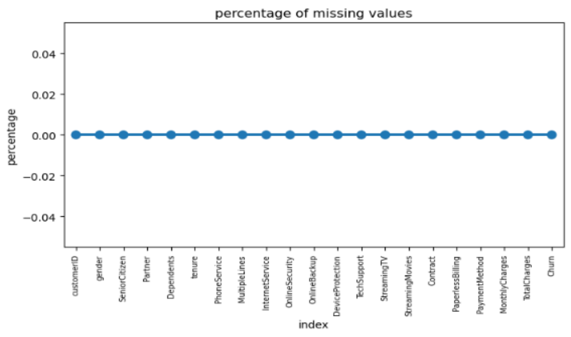


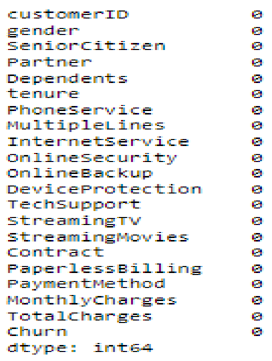
The above ratio indicates highly imbalanced data. Therefore, we analyse the data with other features while separately considering the target values to gain insights.

Check the missing values: =>

No missing values exist.
We can visualize this for clarity.

As there are no missing values,
The percentage is 0%.





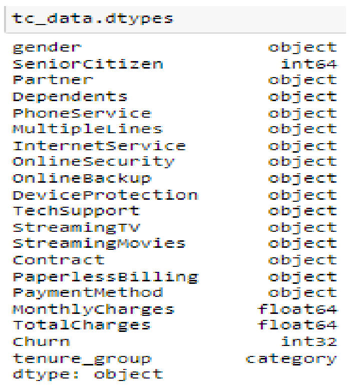
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV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

Data Cleaning:

We duplicate the original data into a new variable for analysis. In the new data, we observe that "total charges" contains numeric values, but the data type is shown as an object. Thus, we convert it to float64.

```
In [13]: # create a copy of our original data
         tc_data=tc.copy()
```

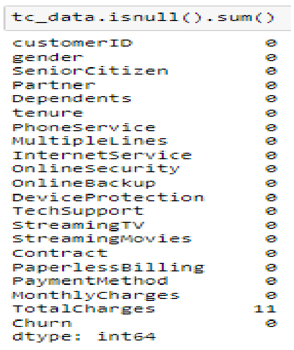
```
In [14]: # Total Charges should be numeric amount. we should convert it into numeric data types
         tc_data.TotalCharges=pd.to_numeric(tc_data.TotalCharges, errors="coerce")
```



```
tc_data.dtypes

gender                object
SeniorCitizen          int64
Partner               object
Dependents            object
PhoneService          object
MultipleLines         object
InternetService       object
OnlineSecurity        object
OnlineBackup          object
DeviceProtection      object
TechSupport           object
StreamingTV           object
StreamingMovies       object
Contract              object
PaperlessBilling      object
PaymentMethod         object
MonthlyCharges       float64
TotalCharges         float64
Churn                  int32
tenure_group        category
dtype: object
```

Now we again check the data types of the attributes, =>

We successfully converted the data type of "Total charges" to float64 using the information from the data types.



```
tc_data.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
```

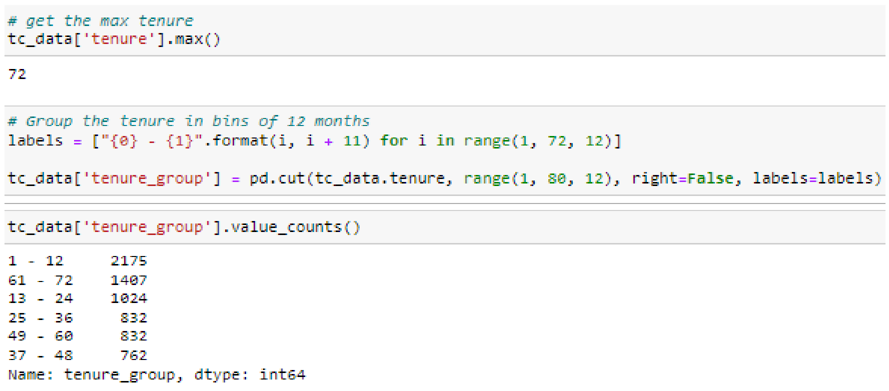
Check for missing values: =>

There are 11 missing values in the "Total charges" column.

The percentage of missing records compared to the total dataset Is very low, approximately 0.15%.
It is safe to ignore them for further processing.

After removing the missing values our data shape is 7032 rows and 21 columns.

We categorize customers into bins based on their tenure. Tenure less than 12 months falls into the 1-12 bin, between 1 and 2 years goes into the 13-24 bin, and so forth.

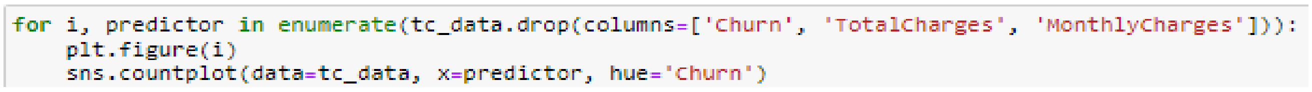


```
# get the max tenure
tc_data['tenure'].max()

72

# Group the tenure in bins of 12 months
labels = ["{0} - {1}".format(i, i + 11) for i in range(1, 72, 12)]

tc_data['tenure_group'] = pd.cut(tc_data.tenure, range(1, 80, 12), right=False, labels=labels)

tc_data['tenure_group'].value_counts()

1 - 12     2175
61 - 72    1407
13 - 24    1024
25 - 36     832
49 - 60     832
37 - 48     762
Name: tenure_group, dtype: int64
```

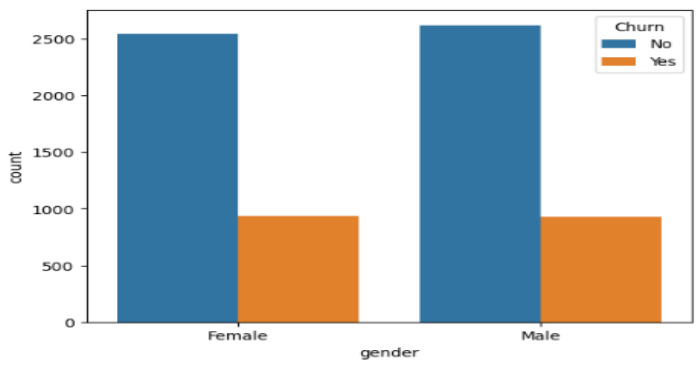
Here is the output.

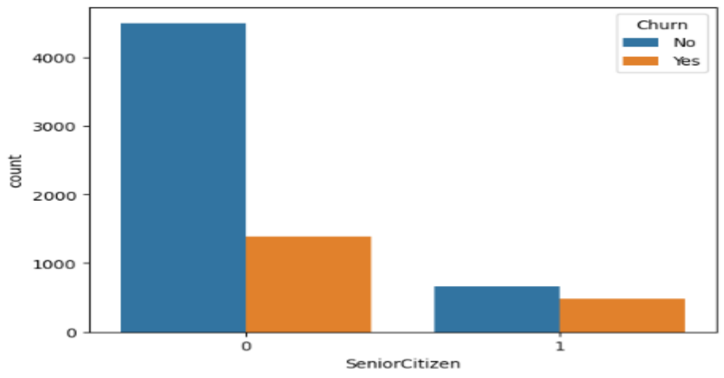
As we can have our tenure groups now
We can remove the unwanted columns "customerID" and "Tenure".

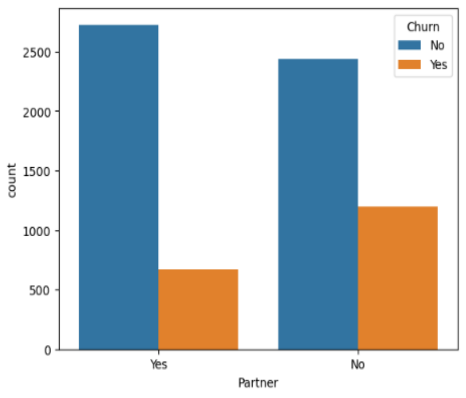
Data Exploration: Univariate analysis

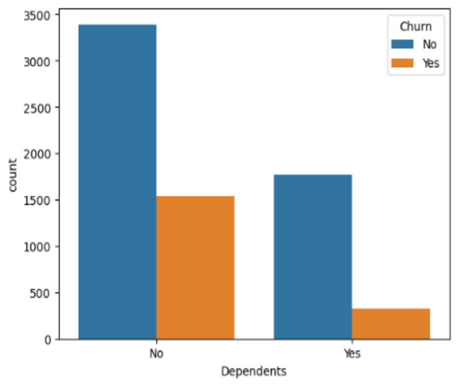


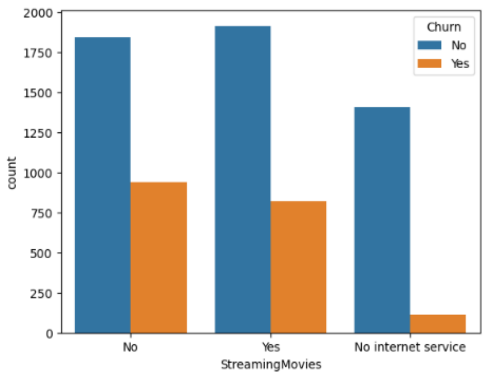
```
for i, predictor in enumerate(tc_data.drop(columns=['Churn', 'TotalCharges', 'MonthlyCharges'])):
    plt.figure(i)
    sns.countplot(data=tc_data, x=predictor, hue='Churn')
```

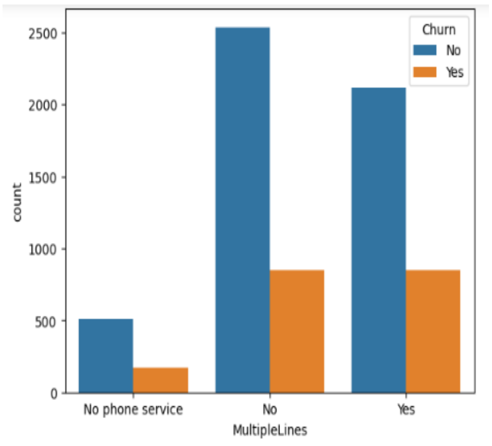


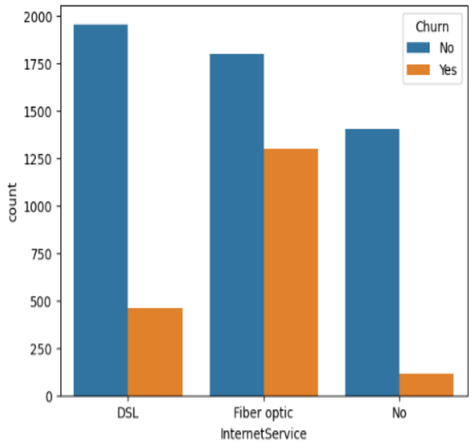


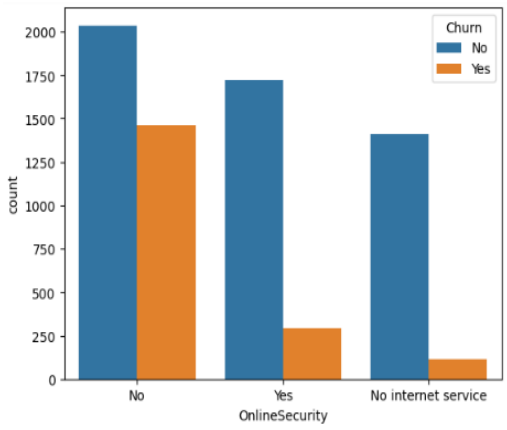

Churn
No
Yes
count
0
500
1000
1500
2000
2500
Yes
No
Partner

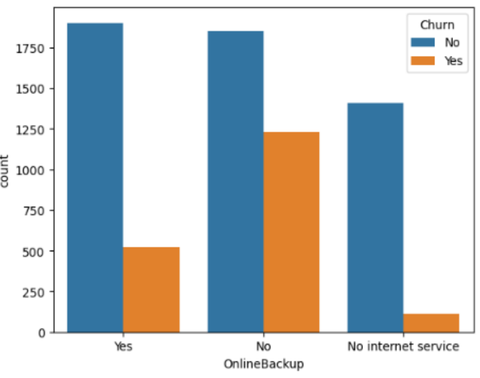

Churn
No
Yes
count
0
500
1000
1500
2000
2500
3000
3500
No
Yes
Dependents

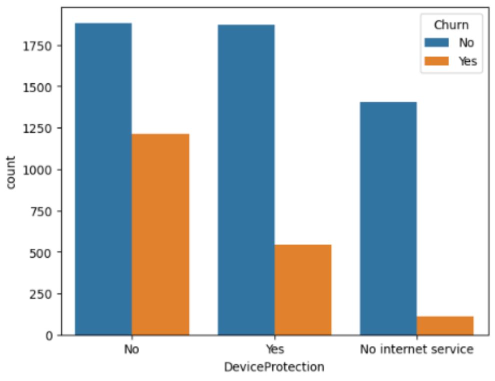

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
StreamingMovies

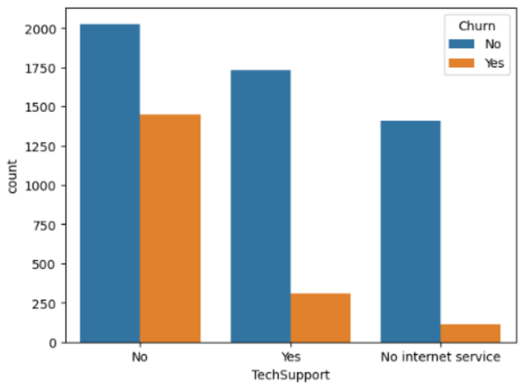

Churn
No
Yes
count
0
500
1000
1500
2000
2500
No phone service
No
Yes
MultipleLines

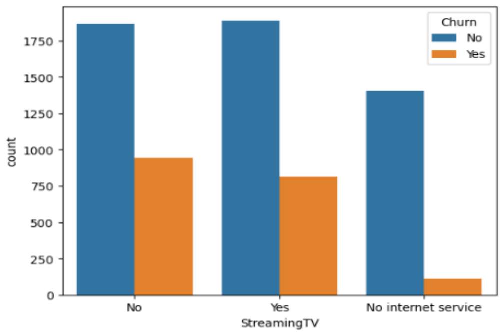

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
DSL
Fiber optic
No
InternetService

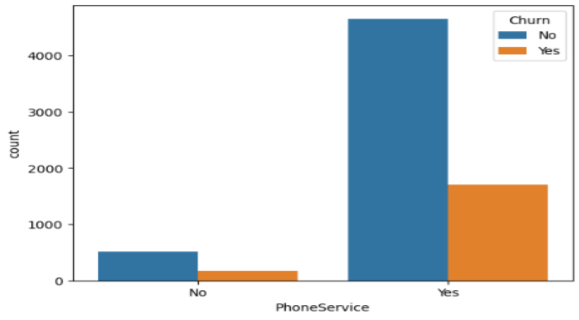

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
OnlineSecurity

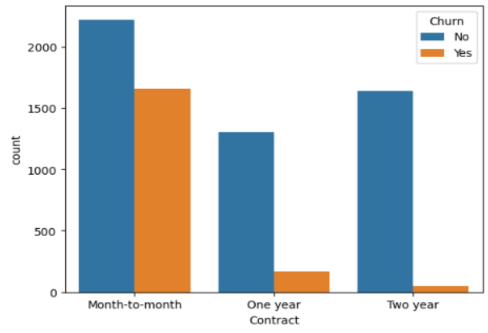

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
Yes
No
No internet service
OnlineBackup

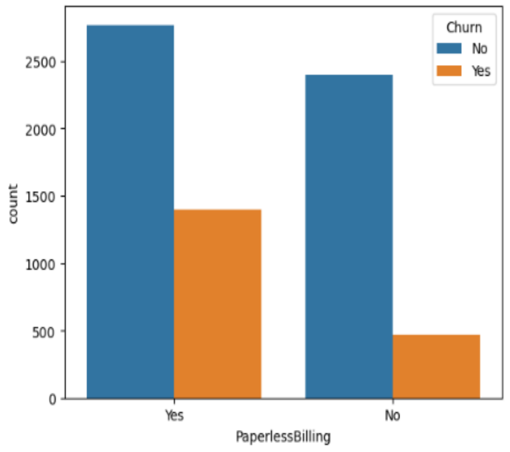

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
No
Yes
No internet service
DeviceProtection

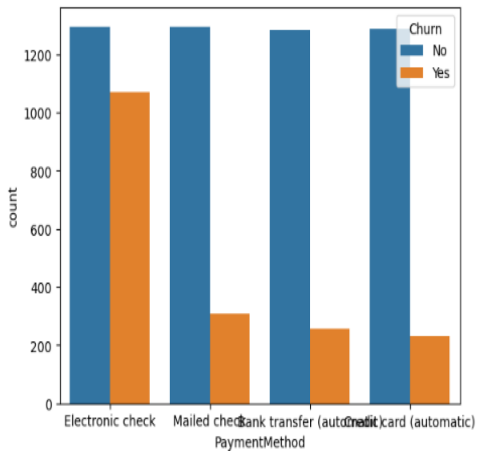

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
TechSupport

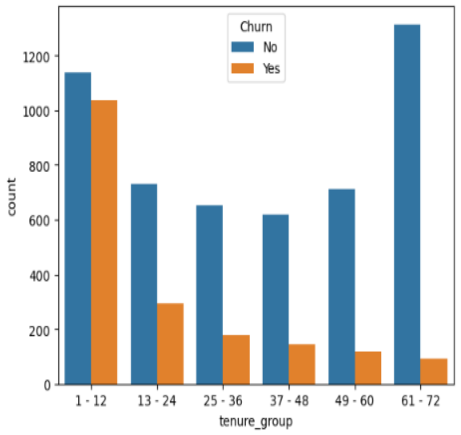

Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
No
Yes
No internet service
StreamingTV


Churn
No
Yes
count
0
1000
2000
3000
4000
No
Yes
PhoneService


Churn
No
Yes
count
0
500
1000
1500
2000
Month-to-month
One year
Two year
Contract

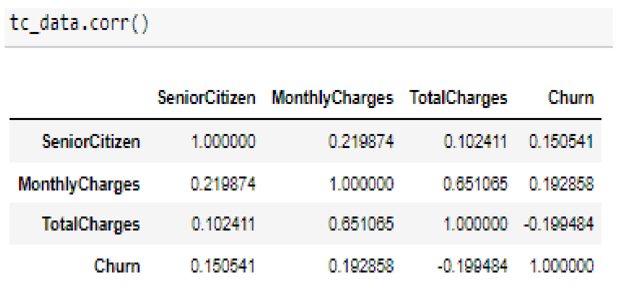

Churn
No
Yes
count
0
500
1000
1500
2000
2500
Yes
No
PaperlessBilling


Churn
No
Yes
count
0
200
400
600
800
1000
1200
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
PaymentMethod


Churn
No
Yes
count
0
200
400
600
800
1000
1200
1 - 12
13 - 24
25 - 36
37 - 48
49 - 60
61 - 72
tenure_group

Convert the target variable “Churn” as in binary numeric variable i.e. Yes=1 and no=0.
Convert all the categorical variable in to the dummy variable.

To understand the relationship between variables, we use **correlation**.



```
tc_data.corr()
```

	SeniorCitizen	MonthlyCharges	TotalCharges	Churn
SeniorCitizen	1.000000	0.219874	0.102411	0.150541
MonthlyCharges	0.219874	1.000000	0.651065	0.192858
TotalCharges	0.102411	0.651065	1.000000	-0.199484
Churn	0.150541	0.192858	-0.199484	1.000000

Interpretation:

There is a positive correlation of approximately 0.22
Between "SeniorCitizen" and "MonthlyCharges."
This suggests that there is a weak positive linear relationship
Between being a senior citizen and the monthly charges.

There is a positive correlation of about 0.10 between
"SeniorCitizen" and "TotalCharges." This indicates a
Weak positive linear relationship between being a senior citizen and the total charges.

There is a positive correlation of around 0.22 between "MonthlyCharges" and "TotalCharges." This implies a moderately strong positive linear relationship between monthly charges and total charges.

There is a negative correlation of about -0.20 between "TotalCharges" and "Churn." This indicates a weak negative linear relationship between total charges and the likelihood of churn.

There is a positive correlation of approximately 0.15 between "SeniorCitizen" and "Churn." This suggests a weak positive linear relationship between being a senior citizen and the likelihood of churn.

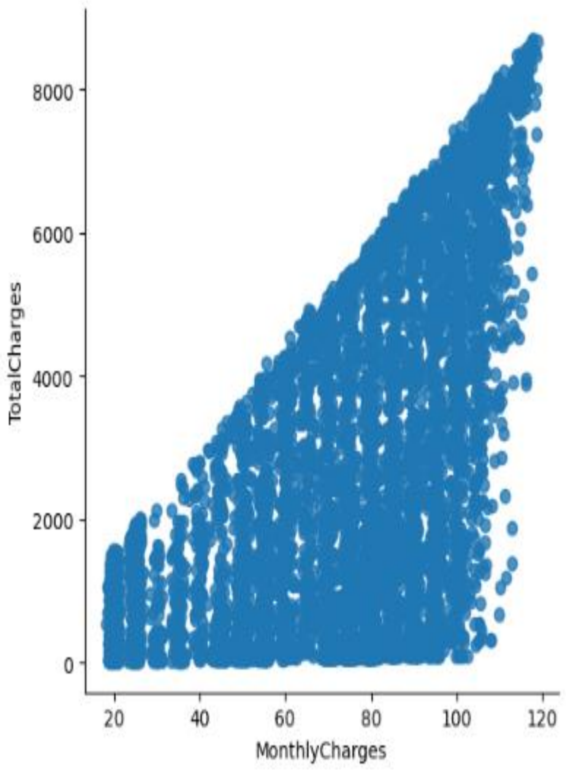
There is a positive correlation of approximately 0.19 between "MonthlyCharges" and "Churn." This indicates a weak positive linear relationship between monthly charges and the likelihood of churn.

We could visualize the monthly charges and total charges by scatter plot

```
# Relationship between monthly charges and total charges
sns.lmplot(data=tc_data_dummies,x='MonthlyCharges',y="TotalCharges", fit_reg=False)
```

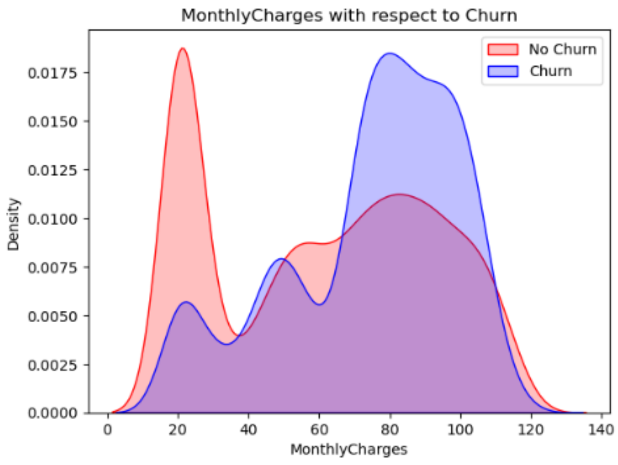
```
<seaborn.axisgrid.FacetGrid at 0x201c80449a0>
```

TotalCharges increases while increase in MonthlyCharges



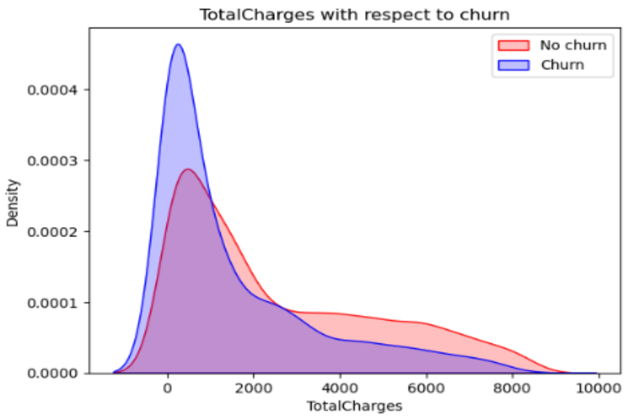
We now create a visualization of
MonthlyCharges concerning churn. : =>

The visualization reveals that churn tends to be higher
When MonthlyCharges are higher. Specifically,
There is a notable concentration of churn cases in the range of
MonthlyCharges between 65 and 120.



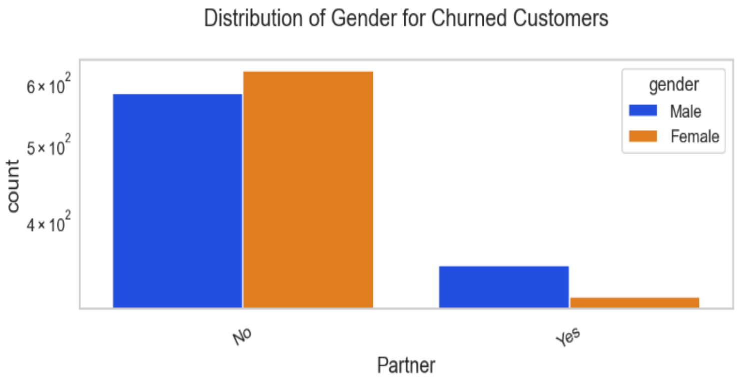
We now create a visualization of
MonthlyCharges concerning churn: =>

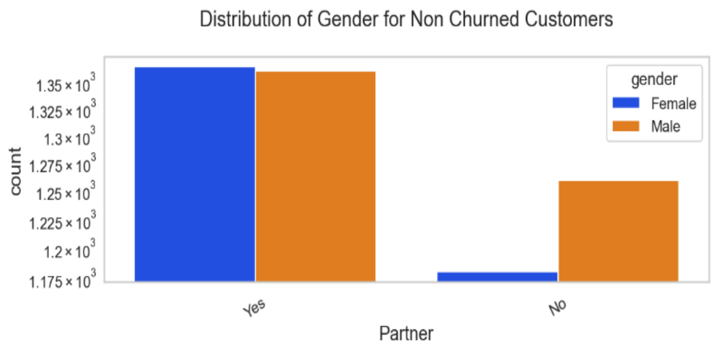
The visualization reveals that churn tends to be higher
When TotalCharges are low.

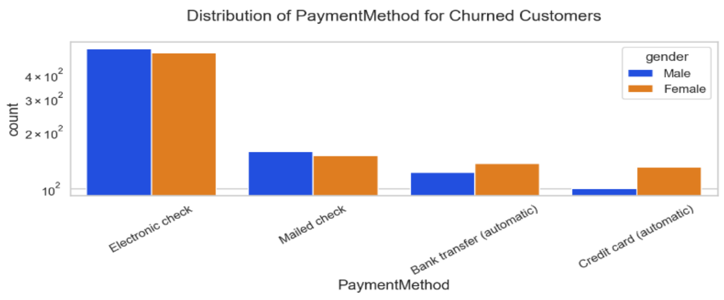


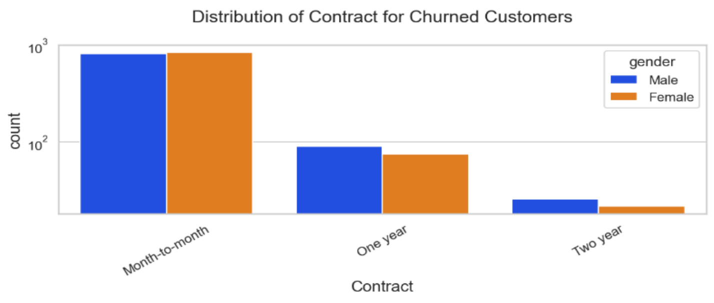
All-over Interpretation of Tenure, MonthlyCharges and TotalCharges is: The churn is higher when the customer have low tenure, high MonthlyCharges and low TotalCharges.

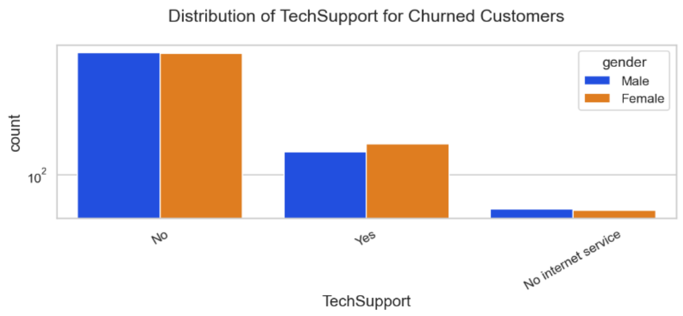
Bivariate analysis:

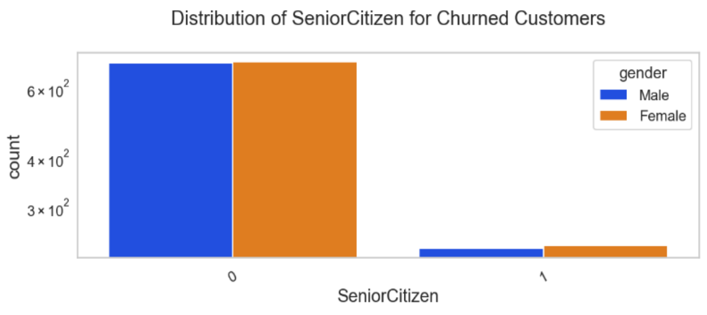












Interpretation: These are some of the quick insights from this exercise:

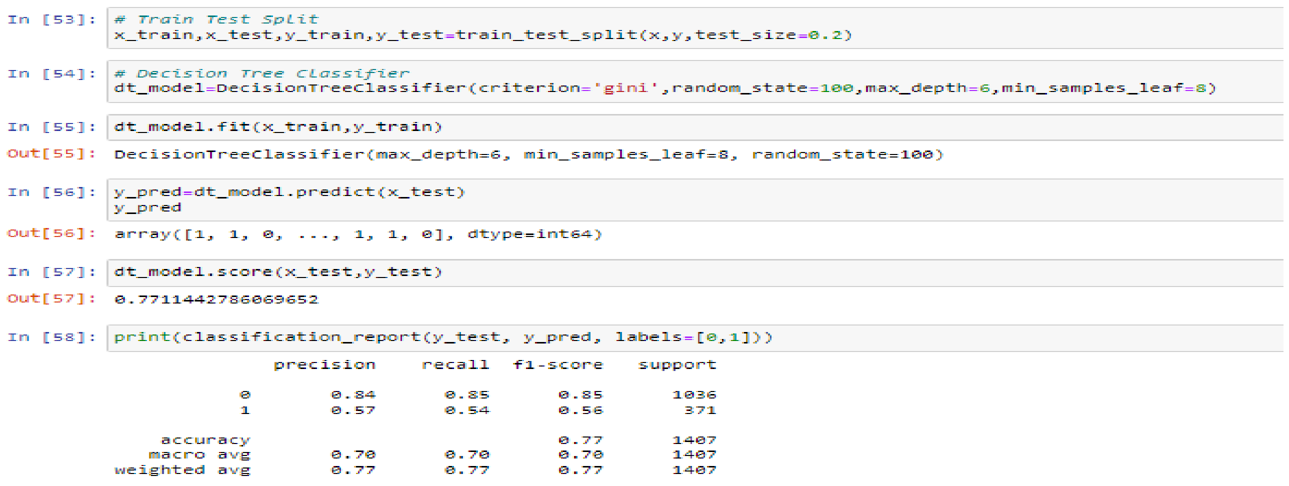
1. Electronic check medium are the highest churners.
2. Contract Type - Monthly customers are more likely to churn because of no contract terms, as they are free to go customers.
3. No Online security, No Tech Support category are high churners.
4. Non senior Citizens are high churners.

Model Building:

X contains all the independent variables whereas **y** contain only the target variables.

We are now to use the **Decision tree**, as Decision trees offer a flexible and effective approach for churn analysis, providing valuable insights into the factors influencing customer churn and enabling businesses to take proactive measures to retain customers.

Decision tree is used for several reasons that are: Interpretability, Feature Importance, Non-linearity, Handling Missing Values, and Robustness to Irrelevant Features, Scalability, Modelling Complex Rules, and Ensemble Methods.



```
In [53]: # Train Test Split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

In [54]: # Decision Tree Classifier
         dt_model=DecisionTreeClassifier(criterion='gini',random_state=100,max_depth=6,min_samples_leaf=8)

In [55]: dt_model.fit(x_train,y_train)
Out[55]: DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)

In [56]: y_pred=dt_model.predict(x_test)
         y_pred
Out[56]: array([1, 1, 0, ..., 1, 1, 0], dtype=int64)

In [57]: dt_model.score(x_test,y_test)
Out[57]: 0.7711442786069652

In [58]: print(classification_report(y_test, y_pred, labels=[0,1]))
                       precision    recall  f1-score   support

                    0       0.84      0.85      0.85      1036
                    1       0.57      0.54      0.56       371

             accuracy                           0.77      1407
            macro avg       0.70      0.70      0.70      1407
         weighted avg       0.77      0.77      0.77      1407
```

Interpretation:
The accuracy of the model is considerably low due to the imbalanced nature of the dataset. In imbalanced datasets, accuracy should not be the sole metric used to evaluate the model's performance, as it can be misleading.

To better assess the model's effectiveness, we need to focus on metrics like recall, precision, and F1 score specifically for the minority class, which in this case is the class representing churned customers (class 1).

Upon analysis, it becomes evident that the precision, recall, and F1 score for class 1 are too low, indicating that the model is not effectively identifying churned customers.

As a solution, we will employ the SMOTEENN technique, which combines up sampling (SMOTE) and the Edited Nearest Neighbours (ENN) algorithm. This approach aims to balance the dataset and improve the model's ability to correctly identify churned customers.

```
In [59]: sm=SMOTEENN()
         X_resampled, y_resampled=sm.fit_resample(x,y)
```

```
In [60]: xr_train, xr_test, yr_train, yr_test=train_test_split(X_resampled, y_resampled, test_size=0.2)
```

```
In [61]: model_dt_smote=DecisionTreeClassifier(criterion='gini', random_state=100, max_depth=6, min_samples_leaf=8)
```

```
In [62]: model_dt_smote.fit(xr_train, yr_train)
         yr_predict=model_dt_smote.predict(xr_test)
         model_score_r=model_dt_smote.score(xr_test, yr_test)
         print(model_score_r)
         print(metrics.classification_report(yr_test, yr_predict))

         0.924061433447099
                       precision    recall  f1-score   support

                    0       0.93      0.89      0.91       512
                    1       0.92      0.95      0.93       660

             accuracy                           0.92      1172
            macro avg       0.92      0.92      0.92      1172
         weighted avg       0.92      0.92      0.92      1172
```

```
In [63]: print(metrics.confusion_matrix(yr_test,yr_predict))

         [[458  54]
          [ 35 625]]
```

Interpretation:
Precision: For class 0 (not churned customers), the precision is 0.93, indicating that 93% of the instances predicted as not churned are indeed correct. For class 1 (churned customers), the precision is 0.92, meaning that 92% of the instances predicted as churned are accurate.

Recall: For class 0, the recall is 0.89, which means that the model correctly identifies 89% of the actual not churned customers. For class 1, the recall is 0.95, indicating that the model captures 95% of the actual churned customers.

F1-score: The F1-score considers both precision and recall and provides a balance between them. For class 0, the F1-score is 0.91, while for class 1, the F1-score is 0.93.

Support: The support represents the number of occurrences of each class in the dataset. There are 512 instances of class 0 (not churned customers) and 660 instances of class 1 (churned customers).

Accuracy: The overall accuracy of the model is 0.92, meaning that it correctly predicts 92% of all instances in the dataset.
In conclusion the model performs well with high precision and recall values for both classes, indicating that it is effective in identifying both churned and not churned customers. The F1-score, which considers both precision and recall, also confirms the model's balanced performance. The overall accuracy of 92% is relatively good, but it's essential to consider the precision and recall values for individual classes, especially in imbalanced datasets.

Interpretation of Confusion matrix:

True Negative (TN) = 458: There are 458 instances of not churned customers correctly predicted as not churned.
False Positive (FP) = 54: There are 54 instances of not churned customers incorrectly predicted as churned customers.
False Negative (FN) = 35: There are 35 instances of churned customers incorrectly predicted as not churned customers.
True Positive (TP) = 625: There are 625 instances of churned customers correctly predicted as churned.

In summary, the model correctly predicted 458 instances of not churned customers and 625 instances of churned customers. However, it made 54 false predictions of not churned customers as churned and 35 false predictions of churned customers as not churned. The confusion matrix provides a detailed breakdown of the model's performance, enabling a better understanding of its strengths and weaknesses in predicting churned and not churned customers.

Now we will use **Random Forest** to analyse our data as Random Forest offers a powerful and versatile approach for churn analysis, providing accurate predictions and insights into the factors influencing customer churn. Its ability to handle non-linearity, imbalanced data, and feature importance analysis makes it a popular choice in the field of churn prediction. Random Forest offers a powerful and versatile approach for churn analysis, providing accurate predictions and insights into the factors influencing customer churn. Its ability to handle non-linearity, imbalanced data, and feature importance analysis makes it a popular choice in the field of churn prediction.

Random Forest is used for several reasons that are: Accuracy, Handling Non-linearity, Feature Importance, Handling Imbalanced Data, Robustness, Outlier Handling, Easy Implementation, and Scalability.

```
In [64]: from sklearn.ensemble import RandomForestClassifier

In [65]: rf_model=RandomForestClassifier(n_estimators=100,criterion='gini', random_state=100, max_depth=6, min_samples_leaf=8)

In [66]: rf_model.fit(x_train, y_train)
Out[66]: RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)

In [67]: y_pred=rf_model.predict(x_test)

In [68]: rf_model.score(x_test,y_test)
Out[68]: 0.7974413646055437

In [69]: print(classification_report(y_test,y_pred, labels=[0,1]))
                      precision    recall  f1-score   support

                   0       0.83      0.92      0.87      1036
                   1       0.67      0.46      0.54       371

            accuracy                           0.80      1407
           macro avg       0.75      0.69      0.71      1407
        weighted avg       0.78      0.80      0.78      1407
```

Interpretation:

For class 0 (not churned customers), the precision is 0.83. This means that out of the instances predicted as not churned, 83% are correct. The recall for class 0 is 0.92, indicating that the model correctly identifies 92% of the actual not churned customers. The F1-score for class 0 is 0.87, which is a harmonic mean of precision and recall.

For class 1 (churned customers), the precision is 0.67. This implies that out of the instances predicted as churned, 67% are correct. The recall for class 1 is 0.46, meaning that the model captures 46% of the actual churned customers. The F1-score for class 1 is 0.54.

The support represents the number of occurrences of each class in the dataset. There are 1036 instances of class 0 (not churned customers) and 371 instances of class 1 (churned customers).

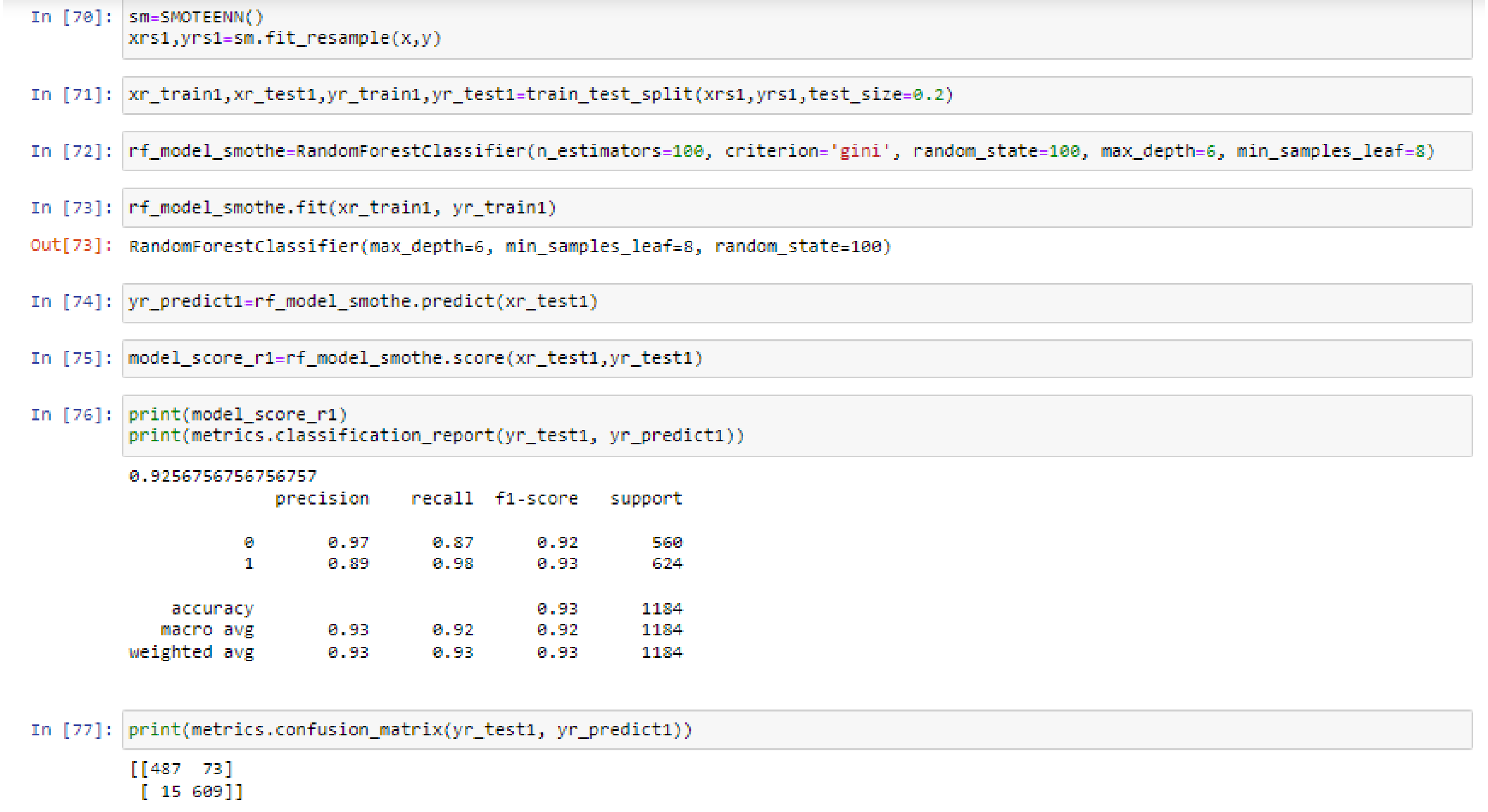
The accuracy of the model is 0.80, indicating that it correctly predicts 80% of all instances in the dataset.

The macro average (macro avg.) provides the average of precision, recall, and F1-score across both classes. In this case, the macro avg. precision is 0.75, the macro avg. recall is 0.69, and the macro avg. F1-score is 0.71.

The weighted average (weighted avg.) provides the weighted average of precision, recall, and F1-score, considering the support for each class. In this case, the weighted avg. precision is 0.78, the weighted avg. recall is 0.80, and the weighted avg. F1-score is 0.78.

The model performs better in predicting not churned customers (class 0) with higher precision, recall, and F1-score for that class. However, the model's performance is lower for predicting churned customers (class 1) with lower precision, recall, and F1-score for that class. The accuracy of 80% indicates that the model is moderately effective overall. Further analysis and fine-tuning may be necessary to improve the performance for class 1 and achieve a more balanced predictive model.

As a solution, we will employ the SMOTEENN technique, which combines up sampling (SMOTE) and the Edited Nearest Neighbours (ENN) algorithm. This approach aims to balance the dataset and improve the model's ability to correctly identify churned customers.



```
In [70]: sm=SMOTEENN()
         xrs1,yrs1=sm.fit_resample(x,y)

In [71]: xr_train1,xr_test1,yr_train1,yr_test1=train_test_split(xrs1,yrs1,test_size=0.2)

In [72]: rf_model_smothe=RandomForestClassifier(n_estimators=100, criterion='gini', random_state=100, max_depth=6, min_samples_leaf=8)

In [73]: rf_model_smothe.fit(xr_train1, yr_train1)

Out[73]: RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)

In [74]: yr_predict1=rf_model_smothe.predict(xr_test1)

In [75]: model_score_r1=rf_model_smothe.score(xr_test1,yr_test1)

In [76]: print(model_score_r1)
         print(metrics.classification_report(yr_test1, yr_predict1))

         0.9256756756756757
                       precision    recall  f1-score   support

                    0       0.97      0.87      0.92       560
                    1       0.89      0.98      0.93       624

             accuracy                           0.93      1184
            macro avg       0.93      0.92      0.92      1184
         weighted avg       0.93      0.93      0.93      1184


In [77]: print(metrics.confusion_matrix(yr_test1, yr_predict1))

         [[487  73]
          [ 15 609]]
```

Interpretation:

For class 0 (not churned customers), the precision is 0.97, indicating that 97% of the instances predicted as not churned are correct. The recall for class 0 is 0.87, meaning that the model correctly identifies 87% of the actual not churned customers. The F1-score for class 0 is 0.92, which is a harmonic mean of precision and recall.

For class 1 (churned customers), the precision is 0.89, meaning that 89% of the instances predicted as churned are accurate. The recall for class 1 is 0.98, indicating that the model captures 98% of the actual churned customers. The F1-score for class 1 is 0.93.

The support represents the number of occurrences of each class in the dataset. There are 560 instances of class 0 (not churned customers) and 624 instances of class 1 (churned customers).

The accuracy of the model is 0.93, which means it correctly predicts 93% of all instances in the dataset.

The macro average (macro avg.) provides the average of precision, recall, and F1-score across both classes. In this case, the macro avg. precision is 0.93, the macro avg. recall is 0.92, and the macro avg. F1-score is 0.92.

The weighted average (weighted avg.) provides the weighted average of precision, recall, and F1-score, considering the support for each class. In this case, the weighted avg. precision is 0.93, the weighted avg. recall is 0.93, and the weighted avg. F1-score is 0.93.

Interpretation of the confusion matrix:

True Negative (TN) = 487: There are 487 instances of not churned customers correctly predicted as not churned.
False Positive (FP) = 73: There are 73 instances of not churned customers incorrectly predicted as churned customers.
False Negative (FN) = 15: There are 15 instances of churned customers incorrectly predicted as not churned customers.
True Positive (TP) = 609: There are 609 instances of churned customers correctly predicted as churned.

The model correctly predicted 487 instances of not churned customers and 609 instances of churned customers. However, it made 73 false predictions of not churned customers as churned and 15 false predictions of churned customers as not churned. The confusion matrix provides a detailed breakdown of the model's performance, enabling a better understanding of its strengths and weaknesses in predicting churned and not churned customers.

Conclusion: We have use the decision trees and random forest to build our model, as we know that our data is imbalanced, with the imbalanced data we don’t get very good accuracy, after smoothing we get good accuracy, recall, precision and F1 score for both the class. From the model our model score is 0.92 it means 92% suggests that the model is performing well and is able to make accurate predictions on the test data.

Thank You !