Customer Churn Prediction Using Machine Learning

Phase 5 Submission Document

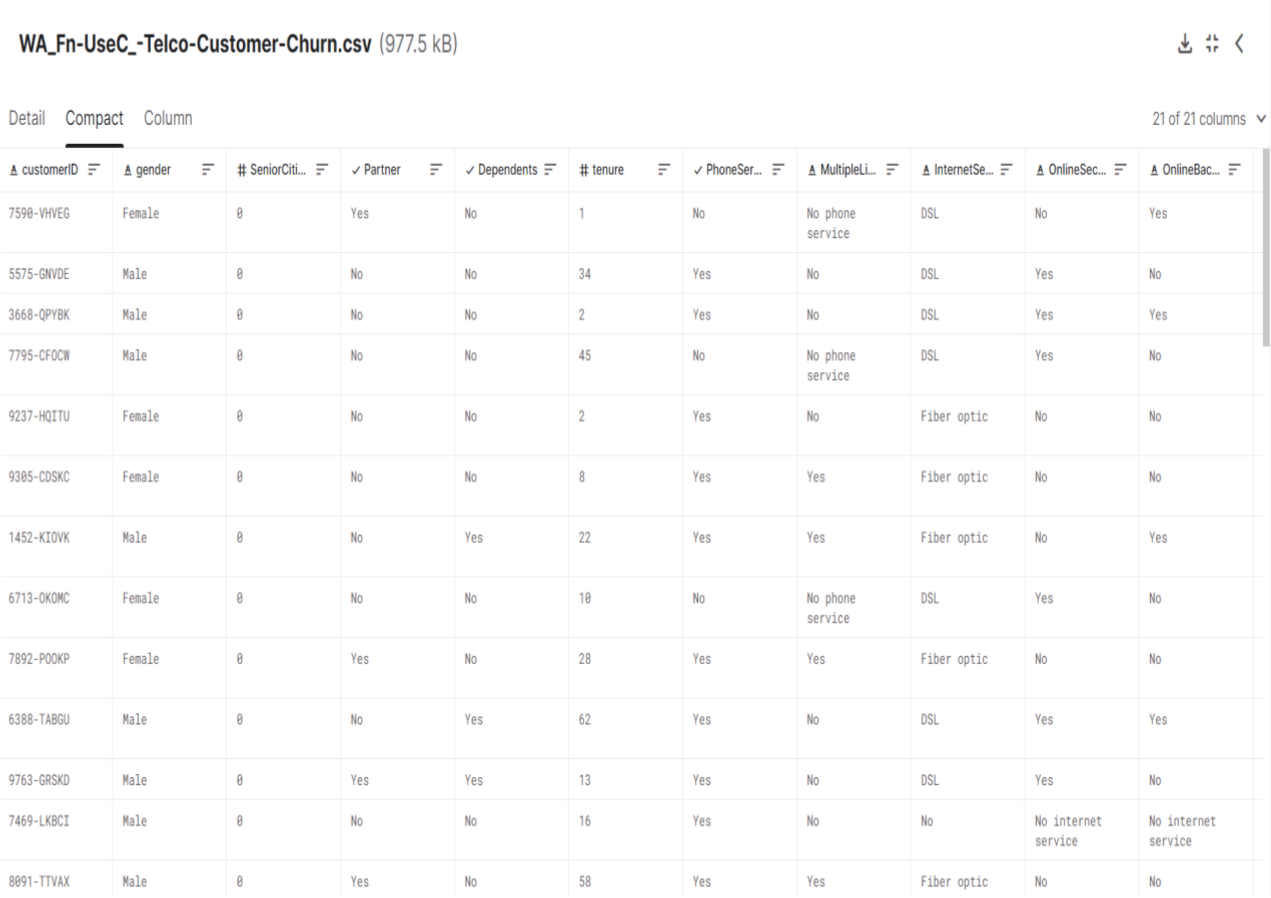
Name	S. P. Jaswanth
Reg. No	410121104048
NM ID	au410121104048
Department	CSE-III
Domain	Data Analytics with Cognos
Project Title	Customer Churn Prediction
Phase 3	Development Part III
College	4101-Adhi College of Engineering and Technology, Kanchipuram

CUSTOMER CHURN PREDICTION

Introduction to Telco Customer Churn:

In the dynamic and fiercely competitive telecommunications (telco) industry, customer churn is a persistent challenge that can significantly impact a company's bottom line and market position. Customer churn, also known as customer attrition or turnover, occurs when subscribers decide to switch their telecom service providers. This phenomenon is driven by a myriad of factors, including pricing, service quality, customer service, and evolving technology. To combat this issue, telco companies employ various strategies and initiatives aimed at retaining their customers, collectively known as customer retention programs. This introduction provides an overview of the critical concept of customer churn in the telco industry and the need for effective customer retention efforts to mitigate its negative consequences.

GIVEN DATA SET:



WA_Fn-UseC_-Telco-Customer-Churn.csv (977.5 kB)

customerID	gender	SeniorCiti...	Partner	Dependents	tenure	PhoneSer...	MultipleLi...	InternetSe...	OnlineSec...	OnlineBac...
7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes
5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No
3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes
7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	No
9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No
9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No
1452-KIOVK	Male	0	No	Yes	22	Yes	Yes	Fiber optic	No	Yes
6713-OKOMC	Female	0	No	No	10	No	No phone service	DSL	Yes	No
7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	No
6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	Yes
9763-GRSKD	Male	0	Yes	Yes	13	Yes	No	DSL	Yes	No
7469-LKBCI	Male	0	No	No	16	Yes	No	No	No internet service	No internet service
8091-TTVAX	Male	0	Yes	No	58	Yes	Yes	Fiber optic	No	No

HERE'S A LIST OF TOOLS AND SOFTWARE COMMONLY USED IN THE PROCESS:

1. Programming Language:

Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy, pandas, scikit-learn, and more.

2. Integrated Development Environment (IDE):

Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, Google Colab, or traditional IDEs like PyCharm.

3. Machine Learning Libraries:

- You'll need various machine learning libraries, including:
- scikit-learn for building and evaluating machine learning models.
- TensorFlow or PyTorch for deep learning, if needed.
- XGBoost, LightGBM, or CatBoost for gradient boosting models.

4. Data Visualization Tools:

Tools like Matplotlib, Seaborn, or Plotly are essential for data exploration and visualization.

5. Data Preprocessing Tools:

Libraries like pandas help with data cleaning, manipulation, and preprocessing.

6. Data Collection and Storage:

Depending on your data source, you might need web scrapingtools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite,PostgreSQL) for data storage.

7. Version Control:

Version control systems like Git are valuable for tracking changes in your code and collaborating with others.

8. Notebooks and Documentation:

Tools for documenting your work, such as Jupyter Notebooks or Markdown for creating README files and documentation.

9. Hyperparameter Tuning:

Tools like GridSearchCV or RandomizedSearchCV from scikit-learn can help with hyperparameter tuning.

10. Web Development Tools (for Deployment):

If you plan to create a web application for model deployment, knowledge of web development tools like Flask or Django for backend development, and HTML, CSS, and JavaScript for the front-end can be useful.

11. Cloud Services (for Scalability):

For large-scale applications, cloud platforms like AWS, Google Cloud, or Azure can provide scalable computing and storage resources.

12. External Data Sources (if applicable):

Depending on your project's scope, you might require tools to access external data sources, such as APIs or data scraping tools.

13. Data Annotation and Labeling Tools (if applicable):

For specialized projects, tools for data annotation and labeling may be necessary, such as Labelbox or Supervisely.

14. Geospatial Tools (for location-based features):

If your dataset includes geospatial data, geospatial libraries like GeoPandas can be helpful.

DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT:

Empathize:

The first step in applying design thinking to a customer chunk project is to empathize with the customers. This can be done by interviewing customers, conducting surveys, and analyzing customer data. The goal of this step is to understand the needs, wants, and pain points of the customers.

Define:

Once the business has a good understanding of its customers, it can begin to define the problem that it is trying to solve for the customer chunk. This problem should be specific, measurable, achievable, relevant, and time-bound.

Ideate:

Once the problem has been defined, the business can begin to ideate solutions. This can be done by brainstorming ideas with the team, conducting research, and looking at examples from other businesses. The goal of this step is to generate as many ideas as possible, without judgment.

Prototype:

Once the business has a number of ideas, it can begin to prototype them. This can be done by creating low-fidelity prototypes, such as sketches and wireframes, or high-fidelity prototypes, such as working models and simulations. The goal of this step is to quickly and cheaply test the ideas to see which ones have the most potential.

Test:

The final step is to test the prototypes with customers to get their feedback. This can be done by conducting user testing sessions, getting feedback from customer support representatives, or analyzing customer usage data. The goal of this step is to refine the prototypes and develop a final solution that meets the needs of the customers.

Example:

Here is an example of how design thinking could be used to improve the customer experience for a customer chunk of online shoppers who have abandoned their shopping carts:

Empathize:

The business could interview customers who have abandoned their shopping carts to understand why they did so. The business could also analyze customer data to identify trends and patterns in cart abandonment.

Define:

Based on the findings from the empathy stage, the business could define the problem as follows: "A significant number of online shoppers abandon their shopping carts before completing the checkout process."

Ideate:

The business could brainstorm solutions to the defined problem. Some possible solutions include:

- Offering free shipping on all orders.
- Providing a money-back guarantee.
- Making the checkout process easier and faster.
- Sending abandoned cart emails to remind customers about the items in their carts.

Prototype:

The business could prototype some of the solutions to see which ones are most effective. For example, the business could create a landing page with a free shipping offer to see if this reduces cart abandonment rates.

Test:

The business could test the prototypes with customers to get their feedback. For example, the business could send abandoned cart emails to a small group of customers to see if this results in more sales.

The business could then iterate on the solutions based on the feedback from customers. This process would continue until the business has developed a final solution that meets the needs of the customers.

DESIGN INTO INNOVATION

Data Collection:

Gather a comprehensive dataset that includes features such Company Database, Third-party Data Providers, Open Data Repositories, Web Scraping, Surveys and Questionnaires, APIs, Publicly Available Reports, Competitions and Challenges, Simulated Data and other relevant variables.

Ensemble Models:

Ensemble models combine the predictions from multiple machine learning algorithms to improve accuracy and reduce overfitting. You can consider using techniques like Random Forests, Gradient Boosting, or AdaBoost. Here's how you can integrate them into your project:

- **Model Selection:** Research and select the ensemble models that are most suitable for your problem. These models work well for classification problems like predicting customer churn.
- **Feature Engineering:** Before applying ensemble models, perform feature engineering to create new relevant features from

the collected data. This can improve the predictive power of your models.

- **Hyperparameter Tuning:** Optimize the hyperparameters of the ensemble models to achieve the best performance. Grid search or random search can be used for this purpose.
- **Model Evaluation:** Assess the performance of ensemble models using metrics such as accuracy, precision, recall, F1 score, and ROC AUC to ensure the best predictive accuracy.

Data-Driven Insights:

By marrying the power of advanced machine learning with our innovative features, we empower businesses to make data-driven decisions that are more precise, timely, and customer-centric. Understanding the why and how of customer churn is at the core of our innovation.

PYTHON PROGRAM:

1.Import Libraries

```
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from imblearn.combine import SMOTEENN
```

2. Load the Dataset:

```
df=pd.read_csv("C:\Users\PAZHANI SABARI RAJ\Downloads\MLProject-ChurnPrediction-main\MLProject-ChurnPrediction-main\tel_churn.csv")

df.head()
```

3. Exploratory Data Analysis(EDA):

```
#Check the various attributes of data like shape (rows and cols), Columns, datatypes

telco_base_data.shape

telco_base_data.shape

telco_base_data.columns.values
```

Output:

```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

```
# Checking the data types of all the columns

telco_base_data.dtypes
```

Output:

```
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
```

3.Feature Engineering:

Create new features or transform existing ones to extract more valuable information. For example, you can calculate the distance to the nearest public transportation, or create a feature for the overall condition of the house.

4.Model Selection:

Choose the appropriate machine learning model for the task. Common models for regression problems like house price prediction include Linear Regression, Decision Trees, Random Forest, Gradient Boosting, and Neural Networks.

5. Training:

Split the dataset into training and testing sets to evaluate the model's performance. Consider techniques like cross-validation to prevent overfitting.

6. Hyperparameter Tuning:

Optimize the model's hyperparameters to improve its predictive accuracy. Techniques like grid search or random search can help with this.

7.Evaluation Metrics:

Select appropriate evaluation metrics for regression tasks, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). Choose the metric that aligns with the specific objectives of your project.

8.Regularization:

Apply regularization techniques like L1 (Lasso) or L2 (Ridge) regularization to prevent overfitting.

9.Feature Selection:

Use techniques like feature importance scores or recursive feature elimination to identify the most relevant features for the prediction.

10. Interpretability:

Ensure that the model's predictions are interpretable and explainable. This is especially important for real estate applications where stakeholders want to understand the factors affecting predictions.

11. Deployment:

Develop a user-friendly interface or API for end-users to inputproperty details and receive price predictions.

12. Continuous Improvement:

Implement a feedback loop for continuous model improvement based on user feedback and new data.

13. Ethical Considerations:

Be mindful of potential biases in the data and model. Ensure fairness and transparency in your predictions.

14. Monitoring and Maintenance:

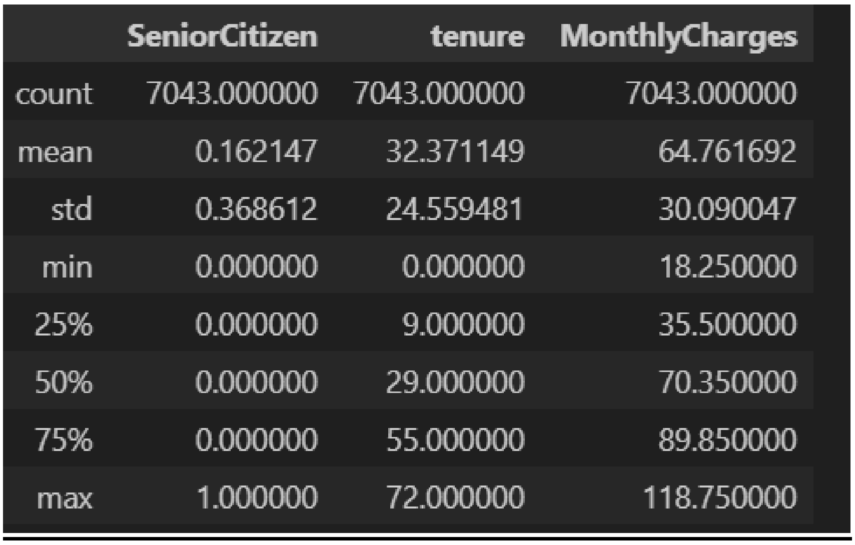
Regularly monitor the model's performance in the real world and update it as needed.

15. Innovation:

Consider innovative approaches such as using satellite imagery or IoT data for real-time property condition monitoring, or integrating natural language processing for textual property descriptions.

```
# Check the descriptive statistics of numeric variables

telco_base_data.describe()
```

Output:



	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

SeniorCitizen is actually a categorical hence the 25%-50%-75% distribution is not proper.

75% customers have tenure less than 55 months.

Average Monthly charges are USD 64.76 whereas 25% customers pay more than USD 89.85 per month.

Feature Engineering:

Depending on your dataset, you may need to create new features or transform existing ones. This can involve one-hot encoding categorical variables, handling data/time data, or scaling numerical features.

Split the Data:

Split your dataset into training and testing sets. This helps you evaluate your model's performance later.

```
#Train Test Split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Steps Involved in EDA:=

- Data Sourcing
- Data Cleaning
- Univariate Analysis with Visualisation
- Bivariate Analysis with Visualisation
- Derived Metrics

Data Sourcing:

Data Sourcing is the process of gathering data from multiple sources as external or internal data collection.

There are two major kind of data which can be classified according to the source:

- Public data
- Private data

Public Data:- The data which is easy to access without taking any permission from the agencies is called public data. The agencies made the data public for the purpose of the research. Like government and other public sector or ecommerce sites made there data public.

Private Data:- The data which is not available on public platform and to access the data we have to take the permission of organisation is called private data. Like Banking ,telecom ,retail sector are there which not made their data publicly available.

The following are some steps involve in Data Cleaning:

- Handle Missing Values
- Standardisation of the data
- Outlier Treatment
- Handle Invalid values

Missing Data - Initial Intuition

Here, we don't have any missing data.

General Thumb Rules:

- For features with less missing values- can use regression to predict the missing values or fill with the mean of the values present, depending on the feature.
- For features with very high number of missing values- it is better to drop those columns as they give very less insight on analysis.
- As there's no thumb rule on what criteria do we delete the columns with high number of missing values, but generally you can delete the columns, if you have more than 30-40% of missing values. But again there's a catch here, for example, Is_Car & Car_Type, People having no cars, will obviously have Car_Type as NaN (null), but that doesn't make this column useless, so decisions has to be taken wisely.

DATA CLEANING:

Data cleaning is the process of identifying and correcting errors, inconsistencies, and inaccuracies in a dataset to ensure its quality and reliability for analysis. This involves tasks such as handling missing values, removing duplicates, and addressing outliers to improve data integrity.

Program:

1. Create a copy of base data for manupulation & processing

```
telco_data = telco_base_data.copy()
```

2. Total Charges should be numeric amount. Let's convert it to numerical data type

```
telco_data.TotalCharges=pd.to_numeric(telco_data.TotalCharges,errors='coerce’)

telco_data.isnull().sum()
```

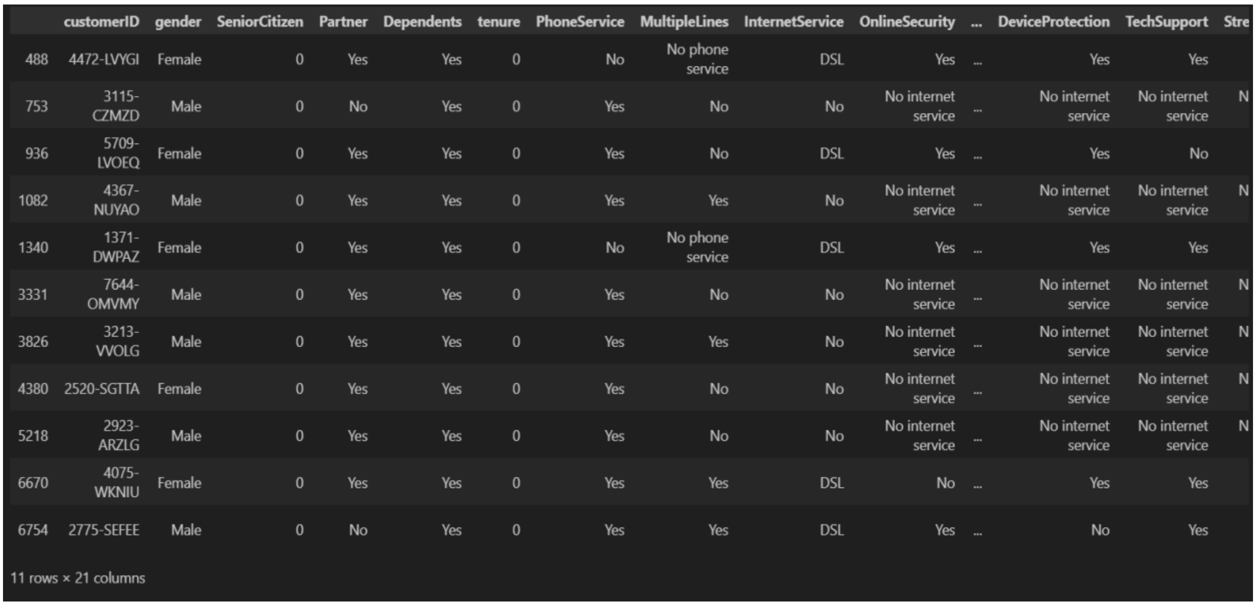
Output:

```
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

3. As we can see there are 11 missing values in TotalCharges column. Let's check these records

```
telco_data.loc[telco_data ['TotalCharges'].isnull() == True]
```



	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	Stre
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	N
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	N
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	N
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	...	No internet service	No internet service	N
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	N
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	No	No internet service	...	No internet service	No internet service	N
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	

11 rows × 21 columns

4. Missing Value Treatement

Since the % of these records compared to total dataset is very low ie 0.15%, it is safe to ignore them from further processing.

```
#Removing missing values

telco_data.dropna(how = 'any', inplace = True)

#telco_data.fillna(0)
```

5. Divide customers into bins based on tenure e.g. for tenure < 12 months: assign a tenure group if 1-12, for tenure between 1 to 2 Yrs, tenure group of 13-24; so on...

```
# Get the max tenure

print(telco_data['tenure'].max()) #72
```

<u>Output:</u>

```
72
```

```
labels = ["{0}- {1}".format(i, i + 11) for i in range(1, 72, 12)]

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

telco_data['tenure_group'].value_counts()
```

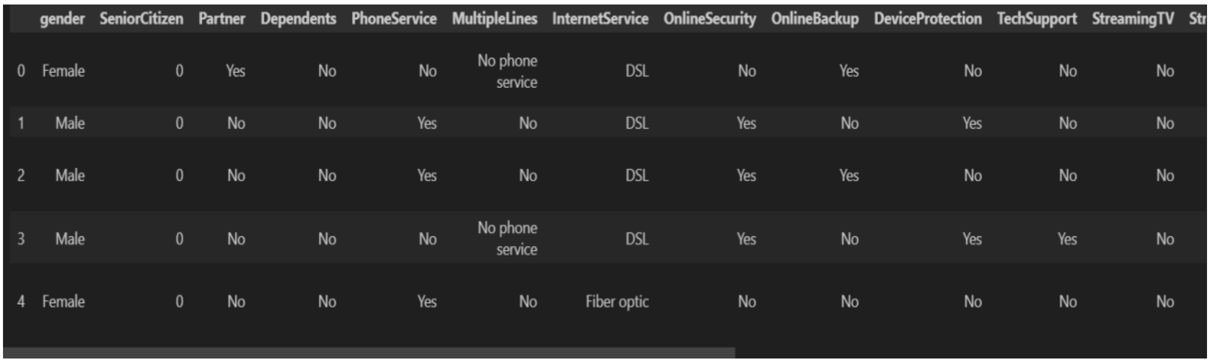
Output:

```
1- 12     2175

61- 72    1407

13- 24    1024

49- 60     832

25- 36     832

37- 48     762

Name: tenure_group, dtype: int64
```

6. Remove columns not required for processing

```
#drop column customerID and tenure

telco_data.drop(columns= ['customerID','tenure'], axis=1, inplace=True)

telco_data.head()
```

Output:



	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Str
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	No	No	No	
1	Male	0	No	No	Yes	No	DSL	Yes	No	Yes	No	No	
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	No	No	No	
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No	
4	Female	0	No	No	Yes	No	Fiber optic	No	No	No	No	No	

DATA EXPLORATION:

Data exploration refers to the process of examining and analyzing a dataset to understand its key characteristics, patterns, and relationships. It involves summarizing and visualizing data to gain insights and inform further data analysis and decision-making. Data exploration helps identify outliers, trends, and potential issues in the data, making it a crucial step in the data analysis process.

1. Plot distibution of individual predictors by churn

UNIVARIATE ANALYSIS:

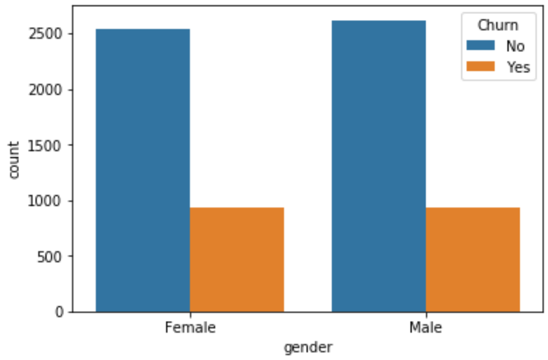
Segmented Univariate Analysis allow you to compare subset of data it help us to understand how the relevant metric varies across the different segment.

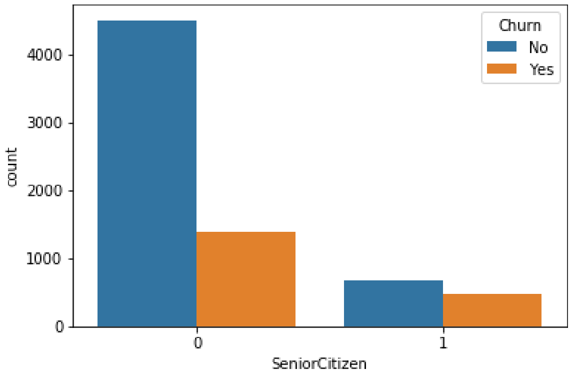
The Standard process of segmented univariate analysis is as follow:

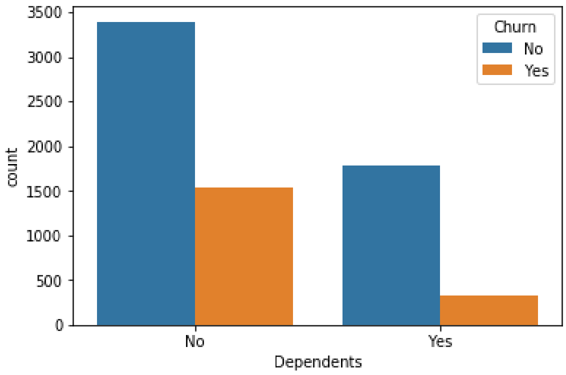
- Take a raw data
- Group by dimensions
- Summarise using a relevant metric like mean ,median.
- Compare the aggregate metric across the categories

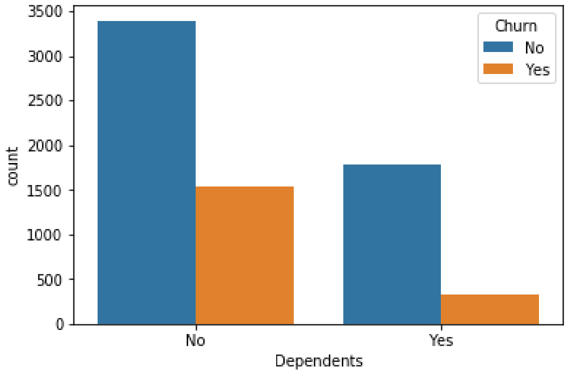
Program:

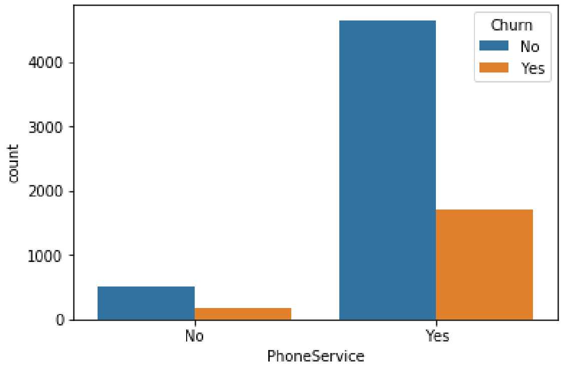
```
for i, predictor in enumerate(telco_data.drop(columns=['Churn', 'TotalCharges',
'MonthlyCharges'])):
    plt.figure(i)
    sns.countplot(data=telco_data, x=predictor, hue='Churn')
```

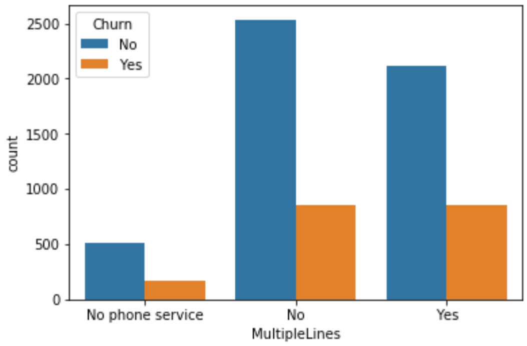


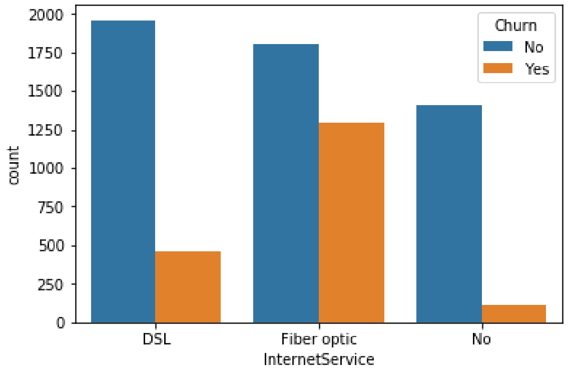


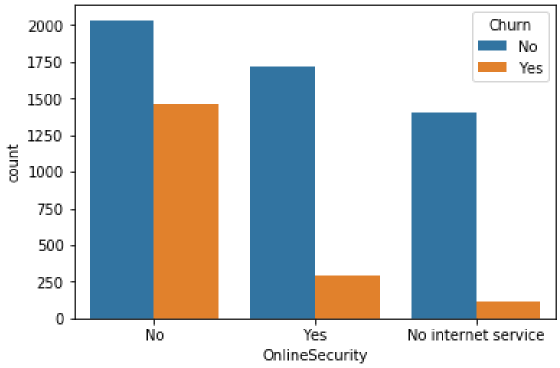


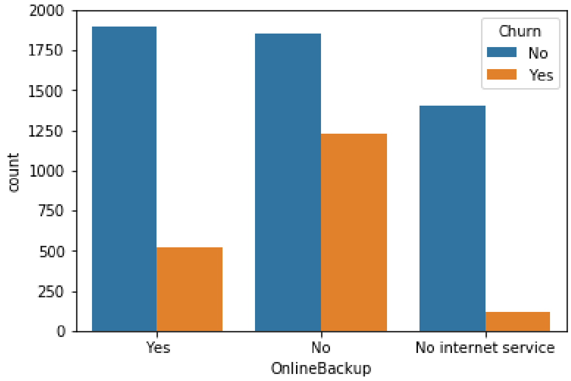


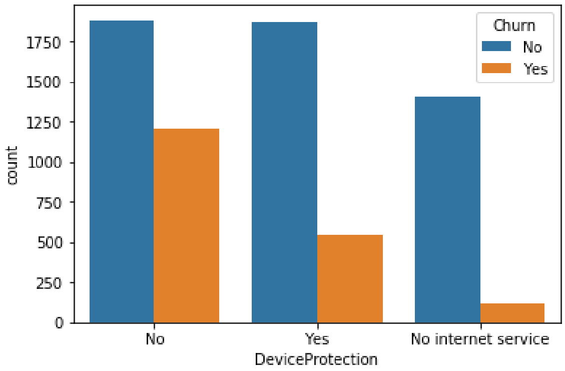

count
0
1000
2000
3000
4000
Churn
No
Yes
No
Yes
PhoneService

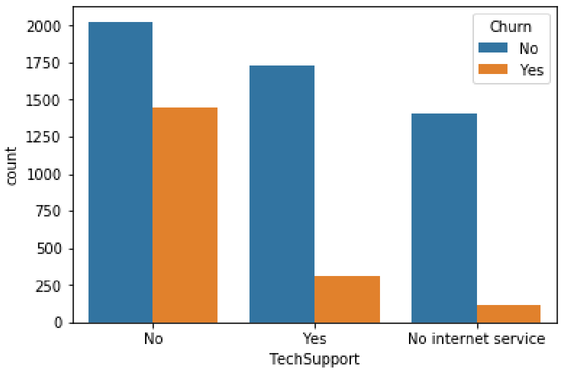

count
0
500
1000
1500
2000
2500
Churn
No
Yes
No phone service
No
Yes
MultipleLines

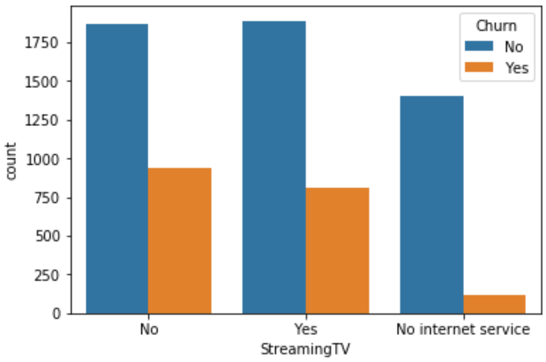

count
0
250
500
750
1000
1250
1500
1750
2000
Churn
No
Yes
DSL
Fiber optic
No
InternetService

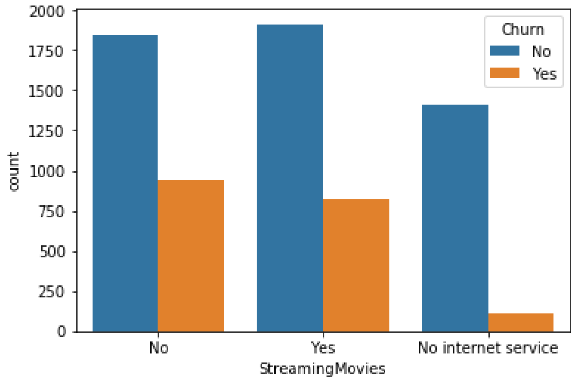

count
0
250
500
750
1000
1250
1500
1750
2000
Churn
No
Yes
No
Yes
No internet service
OnlineSecurity

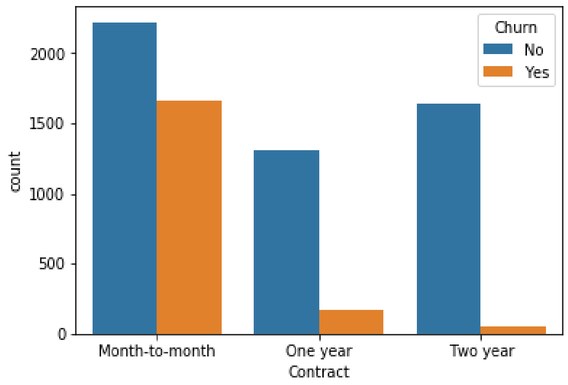

count
0
250
500
750
1000
1250
1500
1750
2000
Churn
No
Yes
Yes
No
No internet service
OnlineBackup

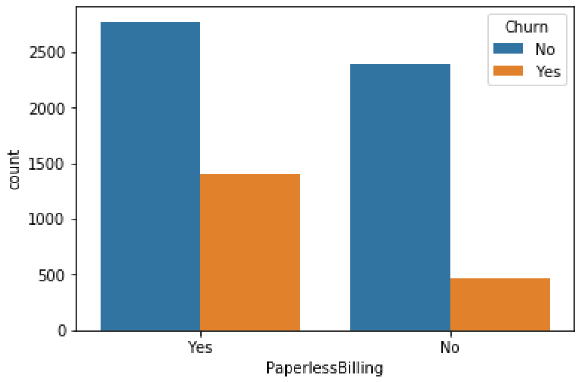

count
0
250
500
750
1000
1250
1500
1750
Churn
No
Yes
No
Yes
No internet service
DeviceProtection

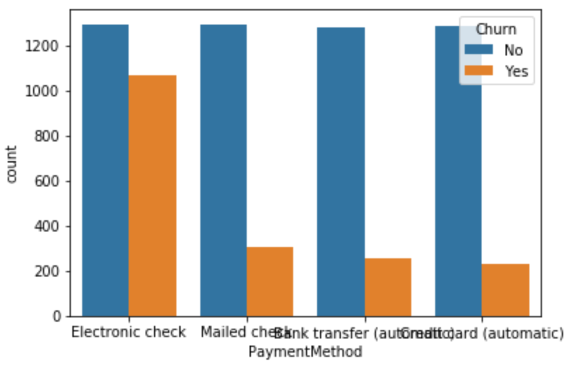

count
0
250
500
750
1000
1250
1500
1750
2000
Churn
No
Yes
No
Yes
No internet service
TechSupport

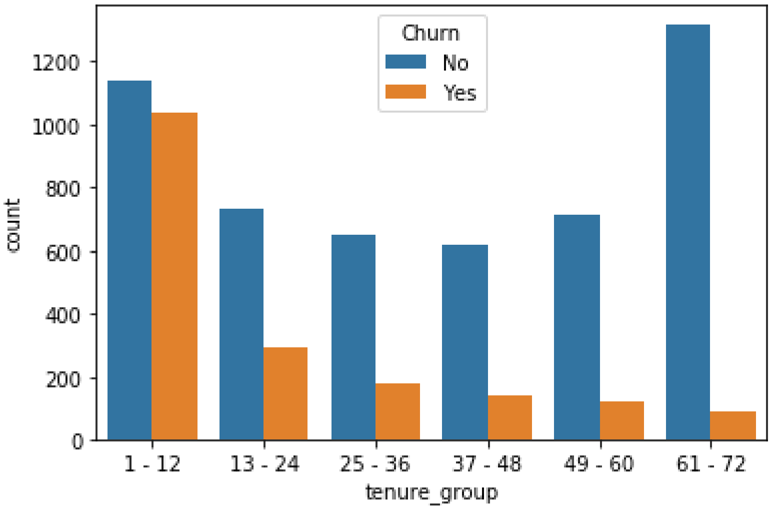

count
0
250
500
750
1000
1250
1500
1750
Churn
No
Yes
No
Yes
No internet service
StreamingTV









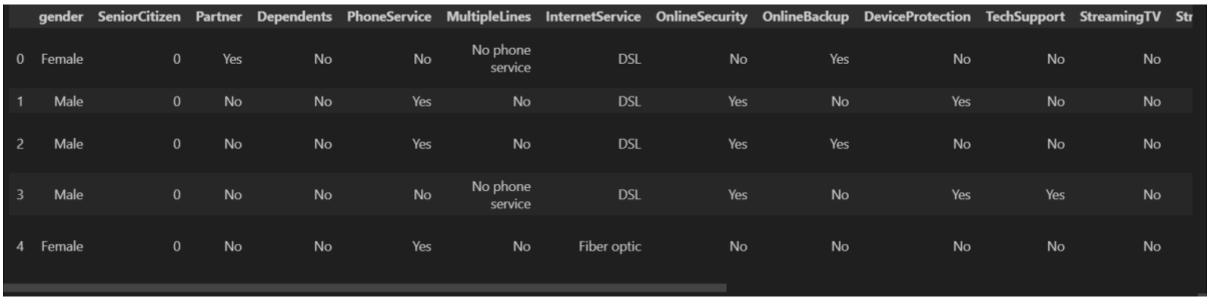


2. Convert the target variable 'Churn' in a binary numeric variable i.e. Yes=1 ; No = 0

```
telco_data['Churn'] = np.where(telco_data.Churn == 'Yes',1,0)
```

```
telco_data.head()
```

Output:

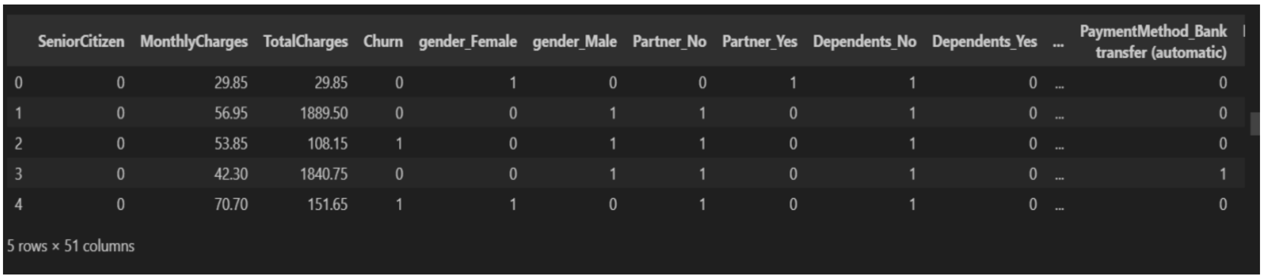


	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Str
0	Female	0	Yes	No	No	No phone service	DSL	No	Yes	No	No	No	
1	Male	0	No	No	Yes	No	DSL	Yes	No	Yes	No	No	
2	Male	0	No	No	Yes	No	DSL	Yes	Yes	No	No	No	
3	Male	0	No	No	No	No phone service	DSL	Yes	No	Yes	Yes	No	
4	Female	0	No	No	Yes	No	Fiber optic	No	No	No	No	No	

3. Convert all the categorical variables into dummy variables

```
telco_data_dummies = pd.get_dummies(telco_data)

telco_data_dummies.head()
```

Output:



	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	...	PaymentMethod_Bank transfer (automatic)
0	0	29.85	29.85	0	1	0	0	1	1	0	...	0
1	0	56.95	1889.50	0	0	1	1	0	1	0	...	0
2	0	53.85	108.15	1	0	1	1	0	1	0	...	0
3	0	42.30	1840.75	0	0	1	1	0	1	0	...	1
4	0	70.70	151.65	1	1	0	1	0	1	0	...	0

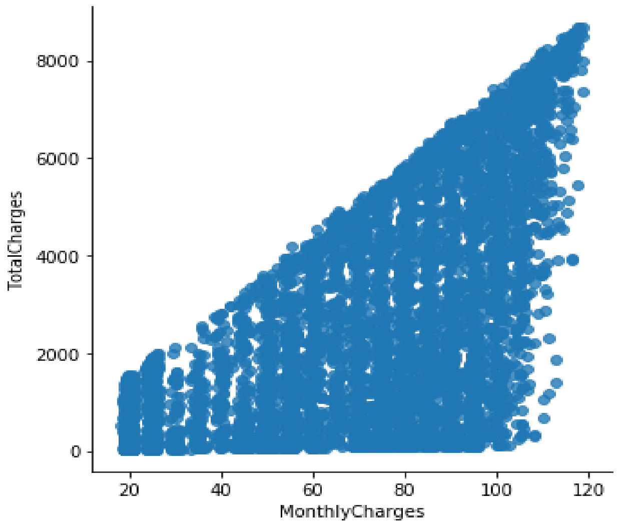
5 rows × 51 columns

4. Relationship between Monthly Charges and Total Charges

```
sns.lmplot(data=telco_data_dummies, x='MonthlyCharges', y='TotalCharges', fit_reg=False)
```

Output:

```
<seaborn.axisgrid.FacetGrid at 0x20d8a9289e8>
```



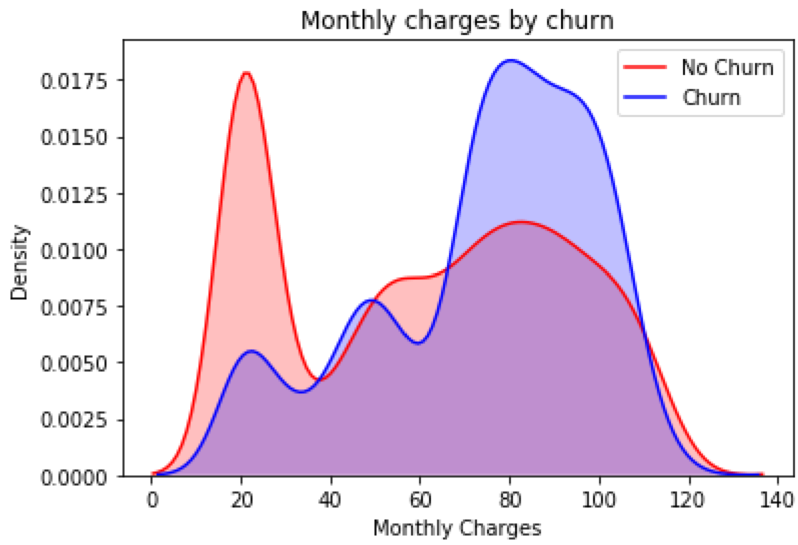
Total Charges increase as Monthly Charges increase - as expected.

5. Churn by Monthly Charges and Total Charges

```
Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 0) ],

                color="Red", shade = True)

Mth = sns.kdeplot(telco_data_dummies.MonthlyCharges[(telco_data_dummies["Churn"] == 1) ],

                ax =Mth, color="Blue", shade= True)

Mth.legend(["No Churn","Churn"],loc='upper right')

Mth.set_ylabel('Density')

Mth.set_xlabel('Monthly Charges')

Mth.set_title('Monthly charges by churn')
```

<u>Output:</u>

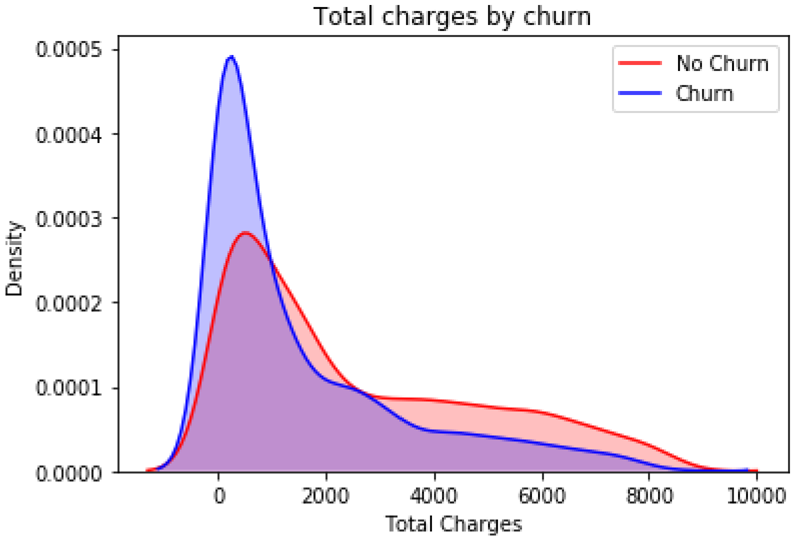
```
Text(0.5, 1.0, 'Monthly charges by churn')
```



Insight: Churn is high when Monthly Charges ar high

```
Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn"] == 0) ],

                color="Red", shade = True)

Tot = sns.kdeplot(telco_data_dummies.TotalCharges[(telco_data_dummies["Churn"] == 1) ],

                ax =Tot, color="Blue", shade= True)

Tot.legend(["No Churn","Churn"],loc='upper right')

Tot.set_ylabel('Density')

Tot.set_xlabel('Total Charges')

Tot.set_title('Total charges by churn')
```

<u>Output:</u>



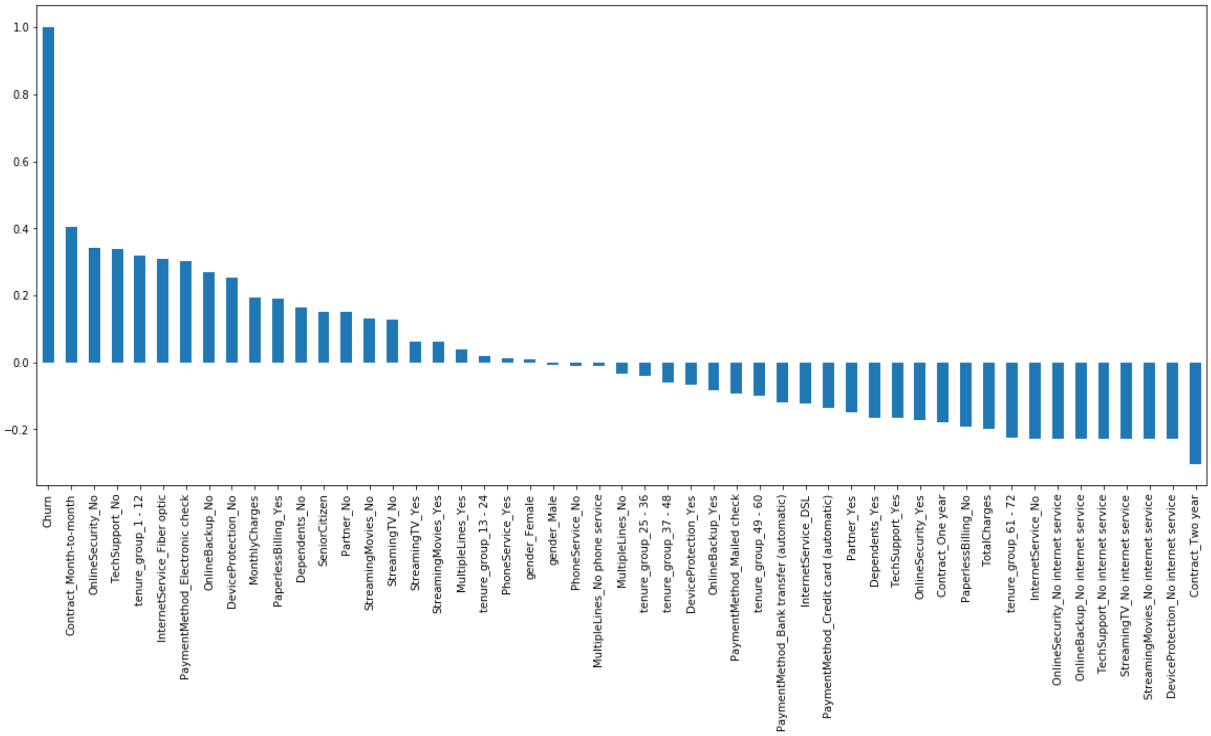
**Surprising insight ** as higher Churn at lower Total Charges

6. Build a corelation of all predictors with 'Churn'

```
plt.figure(figsize=(20,8))

telco_data_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

<u>Output:</u>

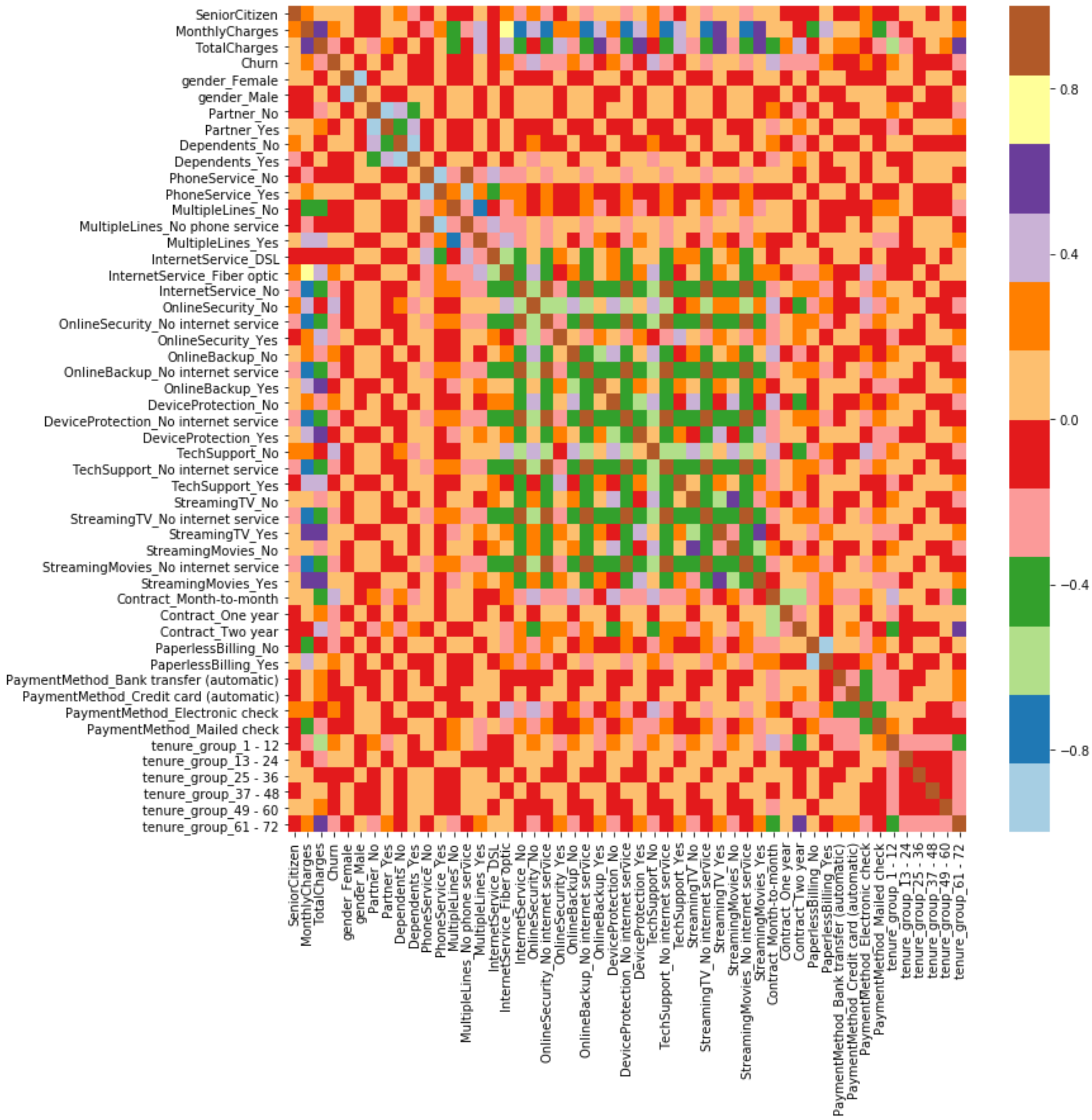
```
<matplotlib.axes._subplots.AxesSubplot at 0x20d8a979f98>
```



```
plt.figure(figsize=(12,12))

sns.heatmap(telco_data_dummies.corr(), cmap="Paired")
```

Output:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1809ebfef60>
```



BIVARIATEANALYSIS CORRELATION:

Data which has two variables ,you often want to measure the relationship that exists between these two variables.

Bi-variate Types:

Correlation: Correlation measure the strength as well as the direction of the linear relationship between the two variables. Its range is from -1 to +1.

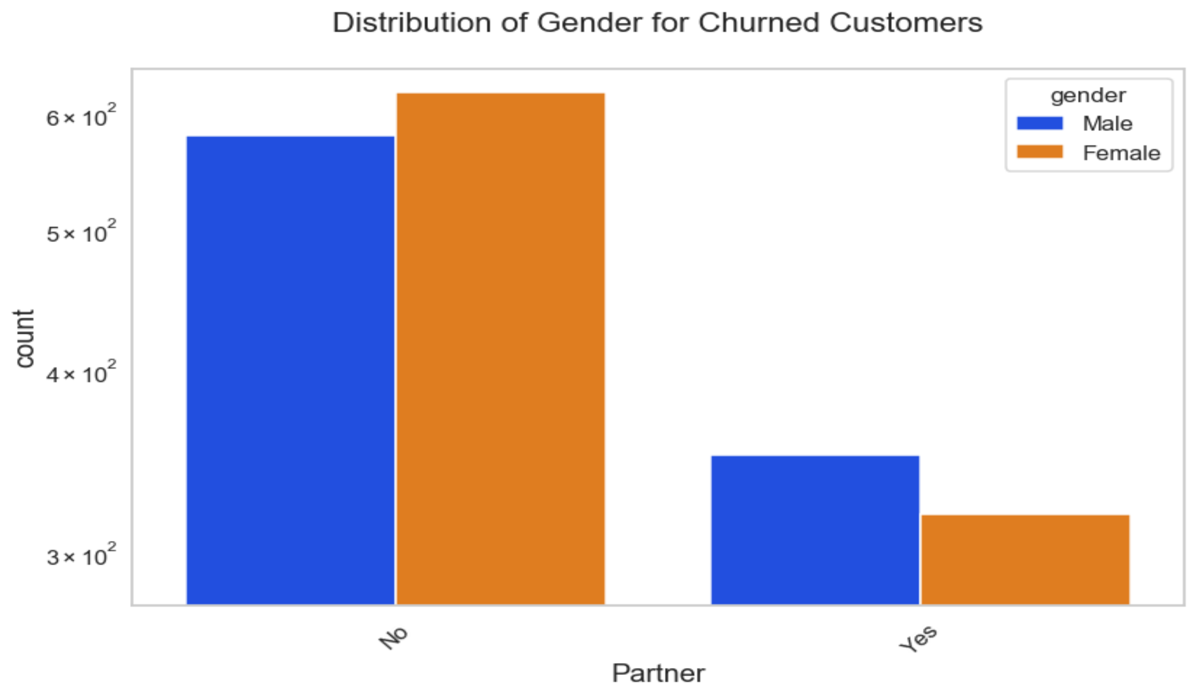
- ● If one increases as the other increases, the correlation is positive
- ● If one decreases as the other increases, the correlation is negative
- ● If one stays constant as the other varies, the correlation is zero

Covariance: Covariance measure how much two random variable vary together. Its range is from -∞ to +∞.

Program:

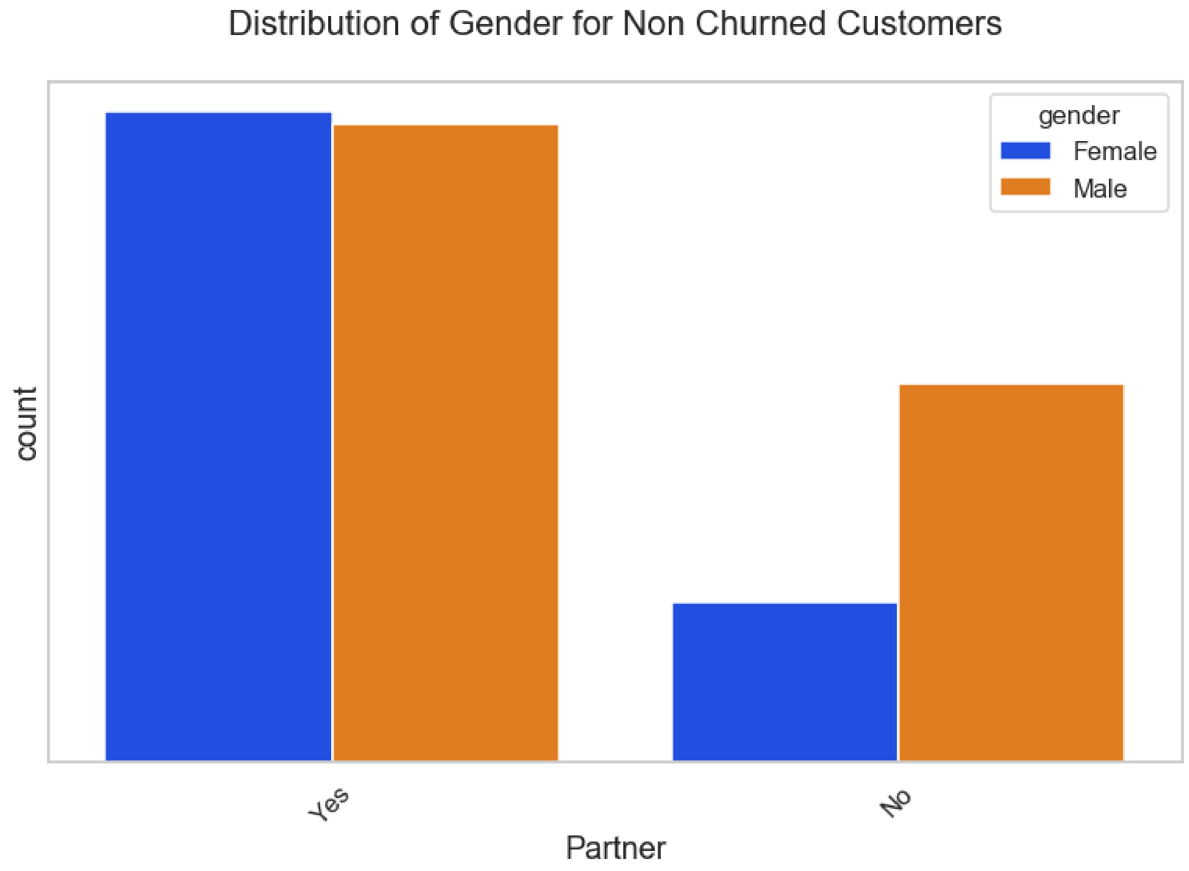
```
new_df1_target0=telco_data.loc[telco_data["Churn"]==0]

new_df1_target1=telco_data.loc[telco_data["Churn"]==1]

def uniplot(df,col,title,hue =None):

   sns.set_style('whitegrid')

   sns.set_context('talk')

   plt.rcParams["axes.labelsize"] = 20

   plt.rcParams['axes.titlesize'] = 22

   plt.rcParams['axes.titlepad'] = 30

   temp = pd.Series(data = hue)

   fig, ax = plt.subplots()

   width = len(df[col].unique()) + 7 + 4*len(temp.unique())

   fig.set_size_inches(width , 8)

   plt.xticks(rotation=45)

   plt.yscale('log')

   plt.title(title)

   ax = sns.countplot(data = df, x= col, order=df[col].value_counts().index,hue = hue,palette='bright')

   plt.show()

niplot(new_df1_target1,col='Partner',title='Distribution of Gender for Churned
Customers',hue='gender')
```

Output:



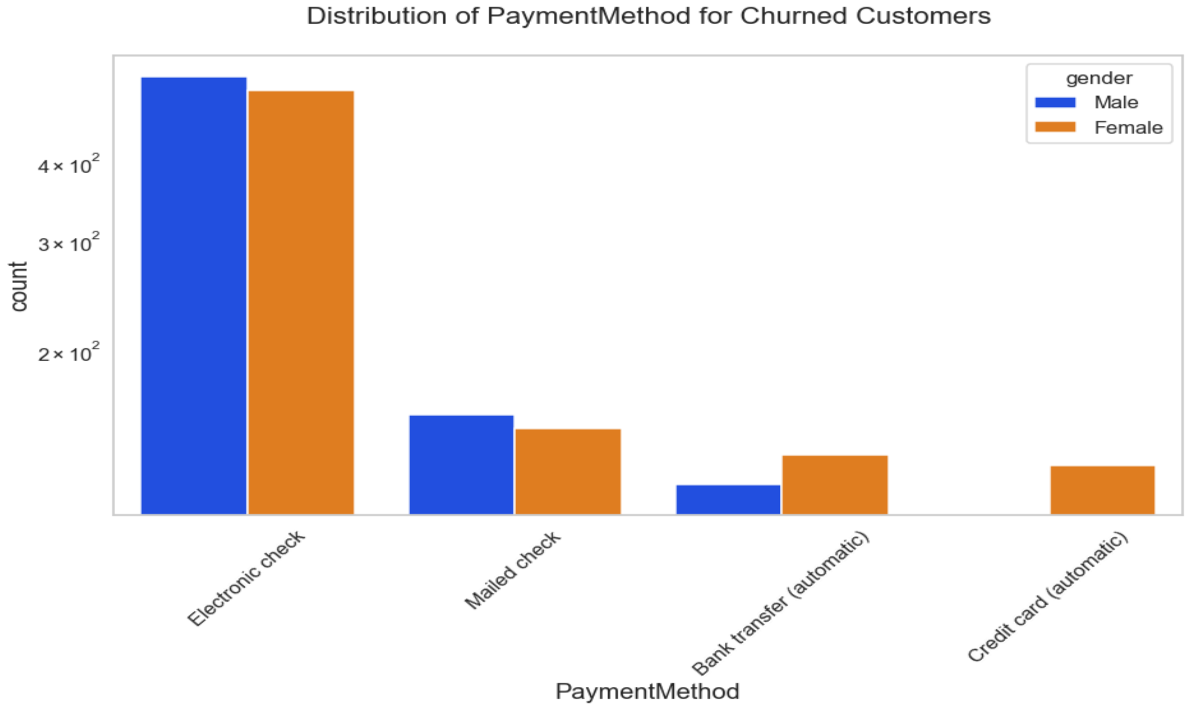
```
uniplot(new_df1_target0,col='Partner',title='Distribution of Gender for Non Churned
Customers',hue='gender')
```

Output:



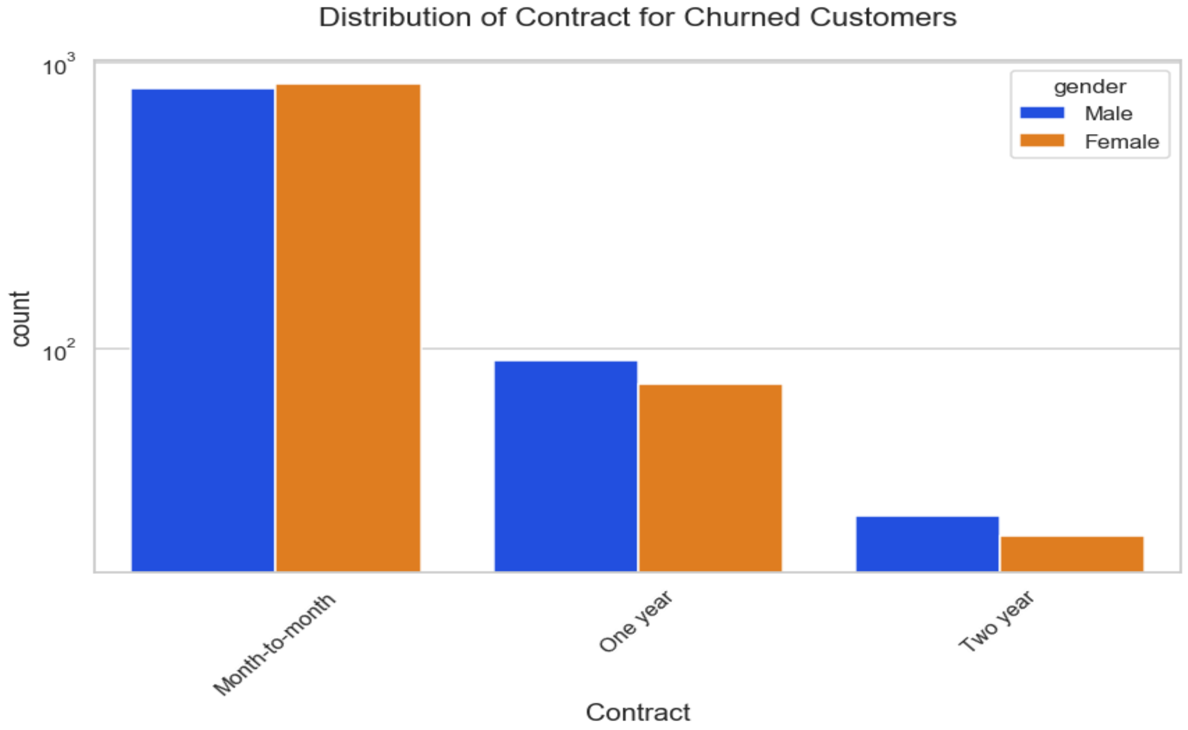
```
uniplot(new_df1_target1,col='PaymentMethod',title='Distribution of PaymentMethod for Churned
Customers',hue='gender')
```

Output:



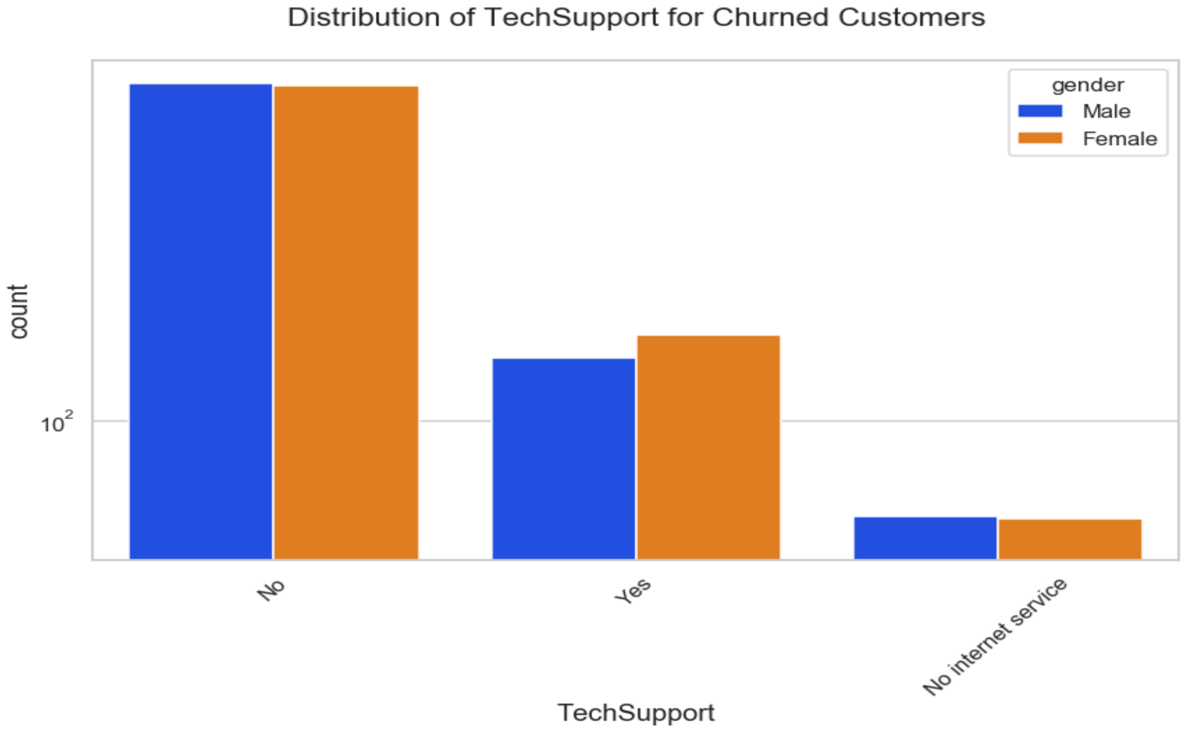
```
uniplot(new_df1_target1,col='Contract',title='Distribution of Contract for Churned
Customers',hue='gender')
```

Output:



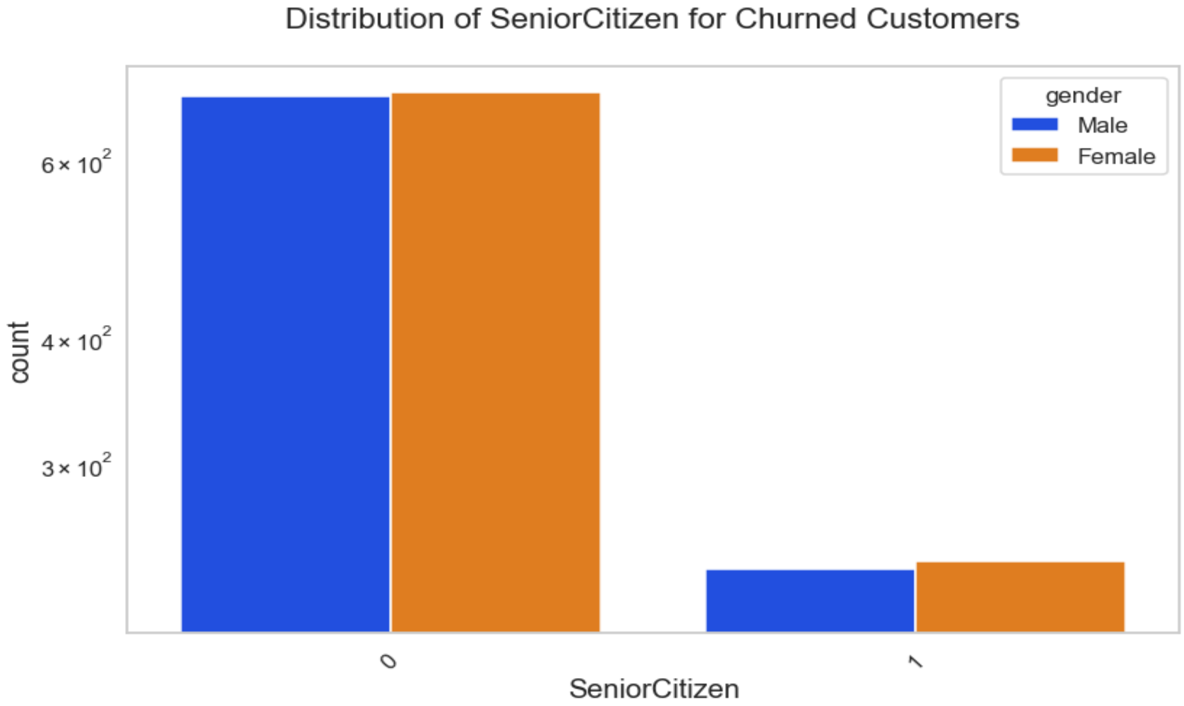
uniplot(new_df1_target1,col='TechSupport',title='Distribution of TechSupport for Churned Customers',hue='gender')

Output:



uniplot(new_df1_target1,col='SeniorCitizen',title='Distribution of SeniorCitizen for Churned Customers',hue='gender')

Output:



MODEL TRAINING:

CHOOSE MACHINE LEARNING ALGORITHM:

There are a number of different machine learning algorithms that can be used for Customer churn Prediction, such as Decision Tree, Random Forest and Performing PCA.

Machine Learning Models:

1.DECISION TREE CLASSIFIER.

A decision tree classifier in data analytics is a machine learning algorithm that uses a tree-like structure to make decisions and classify data based on a set of rules or conditions. It recursively splits the data into subsets, using specific criteria at each branch, to ultimately predict the class or category to which a data point belongs. Decision trees are widely used for tasks like classification and regression analysis, offering a straightforward and interpretable way to make data-driven decisions.

Program:

```
model_dt=DecisionTreeClassifier(criterion="gini",random_state=100,max_depth=6, min_samples_leaf=8)

model_dt.fit(x_train,y_train)
```

Output:

```
DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
y_pred=model_dt.predict(x_test)

y_pred
```

Output:

```
array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

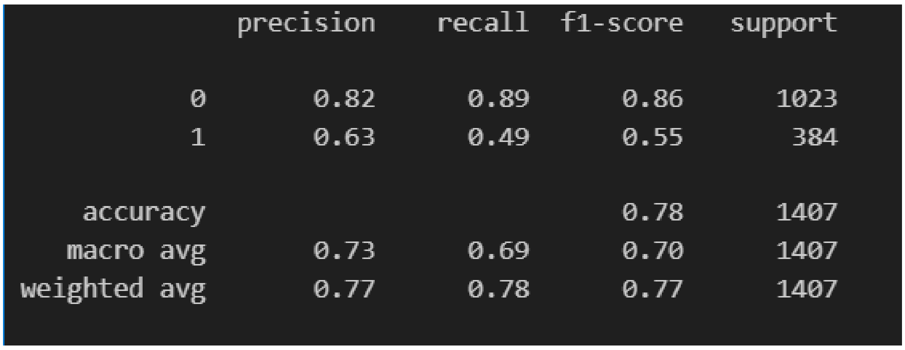
```
model_dt.score(x_test,y_test)
```

Output:

```
0.7818052594171997
```

print(classification_report(y_test, y_pred, labels=[0,1]))

Output:



```
              precision    recall  f1-score   support

           0       0.82      0.89      0.86      1023
           1       0.63      0.49      0.55       384

    accuracy                           0.78      1407
   macro avg       0.73      0.69      0.70      1407
weighted avg       0.77      0.78      0.77      1407
```

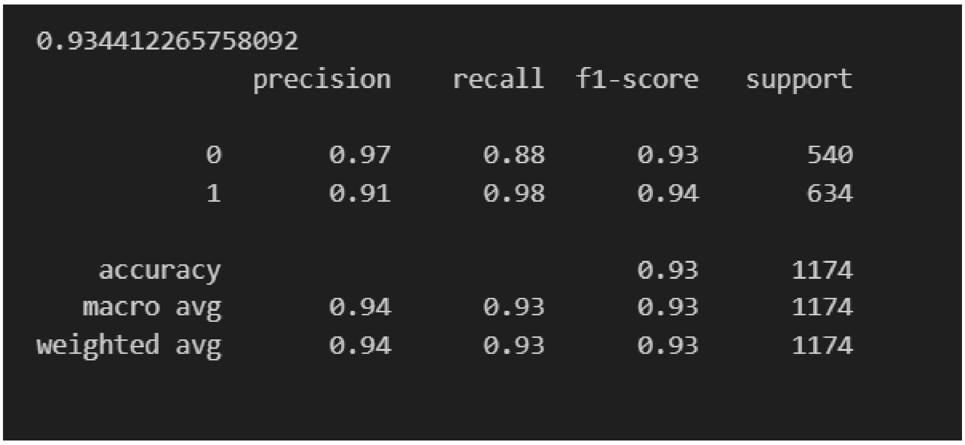
As you can see that the accuracy is quite low, and as it's an imbalanced dataset, we shouldn't consider Accuracy as our metrics to measure the model, as Accuracy is cursed in imbalanced datasets.

Hence, we need to check recall, precision & f1 score for the minority class, and it's quite evident that the precision, recall & f1 score is too low for Class 1, i.e. churned customers.

Hence, moving ahead to call SMOTEENN (UpSampling + ENN)

```
sm = SMOTEENN()

X_resampled, y_resampled = sm.fit_sample(x,y)

xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled, y_resampled,test_size=0.2)

model_dt_smote=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6,
min_samples_leaf=8)

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))
```

Output:



```
0.934412265758092
              precision    recall  f1-score   support

           0       0.97      0.88      0.93       540
           1       0.91      0.98      0.94       634

    accuracy                           0.93      1174
   macro avg       0.94      0.93      0.93      1174
weighted avg       0.94      0.93      0.93      1174
```

print(metrics.confusion_matrix(yr_test, yr_predict))

Output:

```
[[477  63]
 [ 14 620]]
```

Now we can see quite better results, i.e. Accuracy: 92 %, and a very good recall, precision & f1 score for minority class.

Let's try with some other classifier.

2. RANDOM FOREST CLASSIFIER:

A random forest classifier in data analytics is an ensemble machine learning algorithm that combines multiple decision trees to improve the accuracy and reliability of classification tasks. It works by generating a collection of decision trees and then aggregating their predictions to make a final classification. This ensemble approach helps reduce overfitting and enhances the overall performance of the model, making it a powerful tool for various classification problems.

Program:

```
from sklearn.ensemble import RandomForestClassifier

model_rf=RandomForestClassifier(n_estimators=100,criterion='gini',random_state=100,max_depth=6
, min_samples_leaf=8)

model_rf.fit(x_train,y_train)
```

Output:

```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

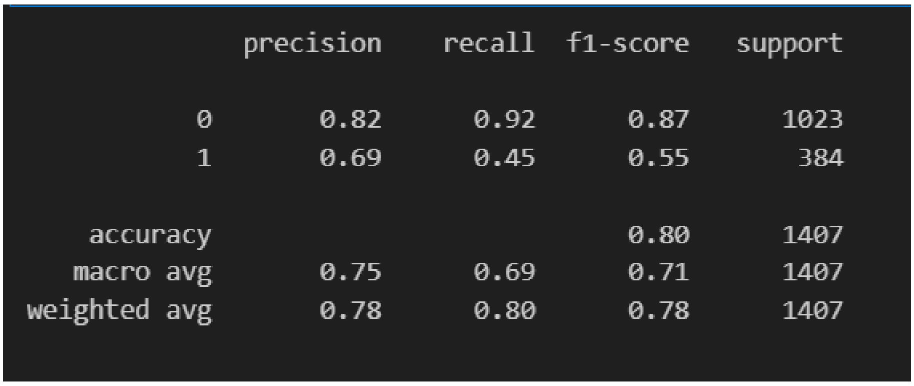
```
y_pred=model_rf.predict(x_test)

model_rf.score(x_test,y_test)
```

Output:

```
0.7953091684434968
```

```
print(classification_report(y_test, y_pred, labels=[0,1]))
```

Output:



```
              precision    recall  f1-score   support

           0       0.82      0.92      0.87      1023
           1       0.69      0.45      0.55       384

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

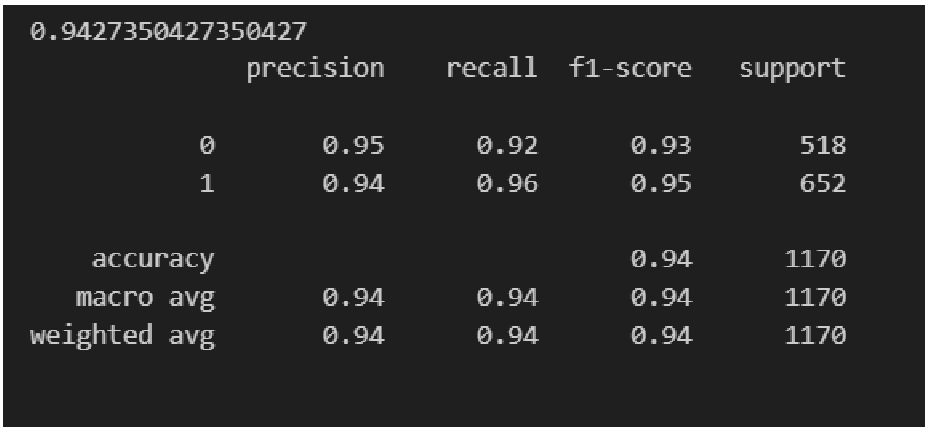
```
sm = SMOTEENN()

X_resampled1, y_resampled1 = sm.fit_sample(x,y)

xr_train1,xr_test1,yr_train1,yr_test1=train_test_split(X_resampled1, y_resampled1,test_size=0.2)

model_rf_smote=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)

model_rf_smote.fit(xr_train1,yr_train1)
```

Output:

```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

```
yr_predict1 = model_rf_smote.predict(xr_test1)

model_score_r1 = model_rf_smote.score(xr_test1, yr_test1)

print(model_score_r1)

print(metrics.classification_report(yr_test1, yr_predict1))
```

Output:



```
0.9427350427350427
              precision    recall  f1-score   support

           0       0.95      0.92      0.93       518
           1       0.94      0.96      0.95       652

    accuracy                           0.94      1170
   macro avg       0.94      0.94      0.94      1170
weighted avg       0.94      0.94      0.94      1170
```

print(metrics.confusion_matrix(yr_test1, yr_predict1))

Output:

[[478 40]

 [27 625]]

With RF Classifier, also we are able to get quite good results, infact better than Decision Tree.

3.PERFORMING PCA:

# Applying PCA

from sklearn.decomposition import PCA

pca = PCA(0.9)

xr_train_pca = pca.fit_transform(xr_train1)

xr_test_pca = pca.transform(xr_test1)

explained_variance = pca.explained_variance_ratio_

model=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)

model.fit(xr_train_pca,yr_train1)

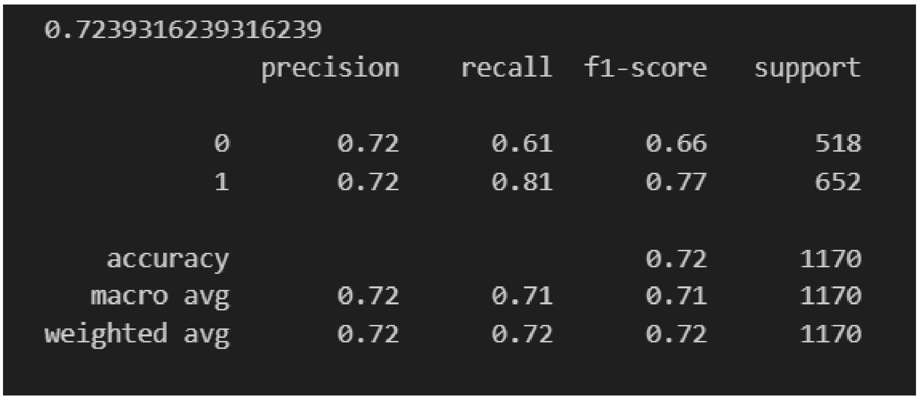
Output:

RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)

yr_predict_pca = model.predict(xr_test_pca)

```
model_score_r_pca = model.score(xr_test_pca, yr_test1)

print(model_score_r_pca)

print(metrics.classification_report(yr_test1, yr_predict_pca))
```

Output:



```
0.7239316239316239
              precision    recall  f1-score   support

           0       0.72      0.61      0.66       518
           1       0.72      0.81      0.77       652

    accuracy                           0.72      1170
   macro avg       0.72      0.71      0.71      1170
weighted avg       0.72      0.72      0.72      1170
```

MODEL TRAINING:

Data Preprocessing:

- Clean the dataset: Handle missing values and outliers.
- Encode categorical variables: Convert categorical features into numerical format using techniques like one-hot encoding or label encoding.
- Feature scaling: Normalize or standardize numerical features to ensure they are on the same scale.
- Feature selection: Identify and select relevant features that have the most impact on churn prediction.

Data Splitting:

- Divide the dataset into two parts: a training set and a testing set. A common split is 70% for training and 30% for testing.

Model Selection:

- Choose an appropriate machine learning algorithm. Common choices for customer churn prediction include logistic regression, decision trees, random forests, support vector machines, and neural networks.

Model Training:

- Train the selected model on the training dataset. The model learns to make predictions based on historical data.

- The training process involves optimizing the model's internal parameters to minimize prediction errors.

Model Evaluation:

- Assess the model's performance using relevant evaluation metrics. Common metrics include:
- Accuracy: The proportion of correct predictions.
- Precision: The ability of the model to correctly predict positive cases.
- Recall: The ability of the model to identify all positive cases.
- F1 Score: The harmonic mean of precision and recall.
- ROC-AUC: The area under the receiver operating characteristic curve.
- Hyperparameter Tuning (Optional):
- Fine-tune the model by adjusting hyperparameters (e.g., learning rate, regularization strength, tree depth) to optimize performance.

Model Validation:

- Validate the model's performance on the testing dataset to ensure it generalizes well to unseen data. This step helps identify if the model is overfitting or underfitting.
- Model Deployment (in the production environment):
- If the model meets the desired performance criteria, deploy it to a production environment where it can make real-time predictions or provide insights.

Monitoring:

- Continuously monitor the model's performance in the production environment to ensure it remains accurate and up-to-date. Retrain the model as needed with new data.

Split the data into training and test sets:

```
#Removing missing values

telco_data.dropna(how = 'any', inplace = True)

#telco_data.fillna(0)
```

Divide customers into bins based on tenure e.g. for tenure < 12 months: assign a tenure group if 1-12, for tenure between 1 to 2 Yrs, tenure group of 13-24; so on...

```
# Get the max tenure

print(telco_data['tenure'].max()) #72
```

Output:

```
72
```

```
# Group the tenure in bins of 12 months
labels = ["{0}- {1}".format(i, i + 11) for i in range(1, 72, 12)]
telco_data['tenure_group'] = pd.cut(telco_data.tenure, range(1, 80, 12), right=False, labels=labels)
telco_data['tenure_group'].value_counts()
```

Output:

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

Train the model on the training set: This involves feeding the training data to the model ad allowing it to learn the relationship between the features an the target variable.

ADVANTAGES:

1. **Improved customer satisfaction:** By focusing on specific customer segments, businesses can better understand their needs and develop solutions that meet those needs. This can lead to improved customer satisfaction and loyalty.

2. **Increased sales:** By targeting specific customer segments, businesses can develop more targeted marketing campaigns and sales strategies. This can lead to increased sales and revenue.

3. **Reduced costs:** By understanding the needs of specific customer segments, businesses can eliminate or reduce unnecessary products and services. This can lead to reduced costs and improved profitability.

4. **Improved employee engagement:** Customer chunk projects can help employees to better understand the customers they serve. This can lead to increased employee engagement and motivation.

5. **Competitive advantage:** By focusing on specific customer segments, businesses can develop a competitive advantage over their competitors.

DISADVANTAGES:

1. **Increased complexity:** Customer chunk projects can be more complex than traditional marketing and sales campaigns. This is because businesses need to have a deep understanding of the customer segments they are targeting.

2. **Increased costs:** Customer chunk projects can require upfront investments in research, marketing, and sales.

3. **Risk of failure:** Customer chunk projects can fail if businesses do not have a good understanding of the customer segments they are targeting or if they do not develop solutions that meet those needs.

BENEFITS:

1. **Improved customer experience:** Customer chunk projects can help businesses to improve the customer experience by focusing on the specific needs of each customer segment.

2. **Increased customer lifetime value:** Customer chunk projects can help businesses to increase customer lifetime value by developing solutions that meet the needs of each customer segment throughout their customer journey.

3. **Reduced customer churn:** Customer chunk projects can help businesses to reduce customer churn by identifying and addressing the needs of at-risk customers.

4. **Improved product development:** Customer chunk projects can help businesses to improve product development by providing insights into the needs of each customer segment.

5. **Increased market share:** Customer chunk projects can help businesses to increase market share by targeting underserved or underserved customer segments.

CONCLUSION :

Customer chunk projects can be a valuable tool for businesses of all sizes to improve customer satisfaction, increase sales, reduce costs, improve employee engagement, and gain a competitive advantage.

However, it is important to carefully consider the advantages, disadvantages, and benefits before embarking on a customer chunk project. Customer chunk projects can be more complex and costly than traditional marketing and sales campaigns, and they carry the risk of failure if businesses do not have a good understanding of the customer segments they are targeting or if they do not develop solutions that meet those needs.

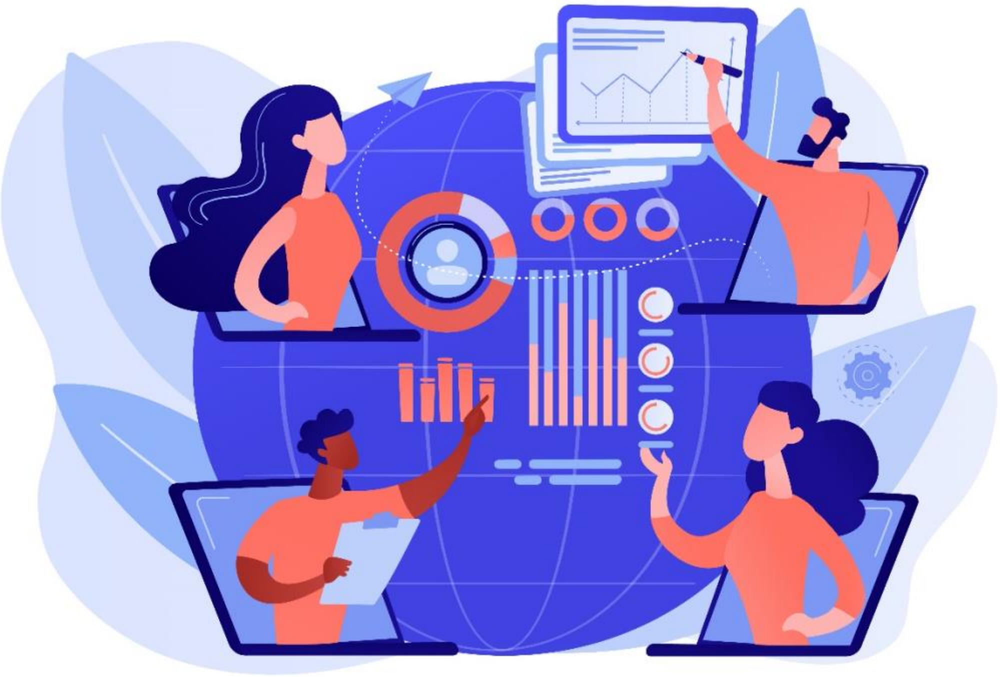
These are some of the quick insights from this exercise:

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.



*****