DIC Project Phase - 1

Nitesh padidam-50495795
Thrivikramarao Kavuri-50496382
Kowsik Kanteti-50495161

Contents

1. Problem Statement:

The objective of our project is to create a predictive model that forecasts customer turnover rates using past customer data from the telecommunications industry. We hope to address crucial questions such as: What is the likelihood of a customer churning? How can we spot high-risk customers? What strategies and interventions can be used to lower churn and increase client retention? by studying historical behavior, usage trends, demographics, and customer interactions. This predictive model will be a helpful tool in the telecom sector for proactively reducing customer attrition, optimizing marketing efforts, and eventually enhancing overall customer satisfaction and corporate profitability.

1.1 Background and Contribution:

We need to understand two important terms of our project

1. Customer churn:
   Customer churn is the rate at which customers discontinue utilizing a product or service during a given time period. It is an important indicator for businesses since it has a direct impact on sales and profitability. Churn can happen in a variety of businesses, such as telecommunications, subscription services, retail, and others.
2. Churn Rate:
   Churn rate (also known as attrition rate) is a metric which quantifies the percentage at which consumers discontinue doing business with a company over a specific time period. Churn refers to the number of subscribers who cancel or do not renew their subscriptions.

Customer churn and churn rate are significant KPIs for telecommunications firms. The churn rate calculates the percentage of consumers who stop utilizing the company's services during a given time period. Businesses may make informed decisions about resource allocation, client retention tactics, and focused marketing efforts by accurately predicting churn and understanding its underlying factors.

Identifying high-risk customer segments and implementing proactive retention efforts is essential for minimizing churn. Companies can use sophisticated analytics and machine learning approaches to estimate churn rates by examining historical data on customer interactions, service usage patterns, contract terms, and customer care interactions.

However, predicting churn in the telecom business is difficult due to the complicated nature of consumer behavior, the wide range of services available, and the volatile market landscape. It necessitates a thorough grasp of client demographics, usage behavior, and other pertinent characteristics.

To overcome this issue, we can investigate data-driven approaches with telecom-specific datasets. We can reliably anticipate attrition rates by leveraging transactional data, client demographics, service plans, and interaction history. To generate more precise churn rate estimates, these models can add parameters such as contract term, service usage patterns, customer feedback, and support interactions.

Improving the accuracy of churn rate estimates allows telecom businesses to better manage resources, devise customer-centric retention strategies, and conduct focused interventions that correspond with the industry's specific characteristics. It enables firms to identify high-risk client groupings, implement targeted retention strategies, and ultimately maximize long-term customer retention and revenue growth.

2. <u>Data Source:</u>

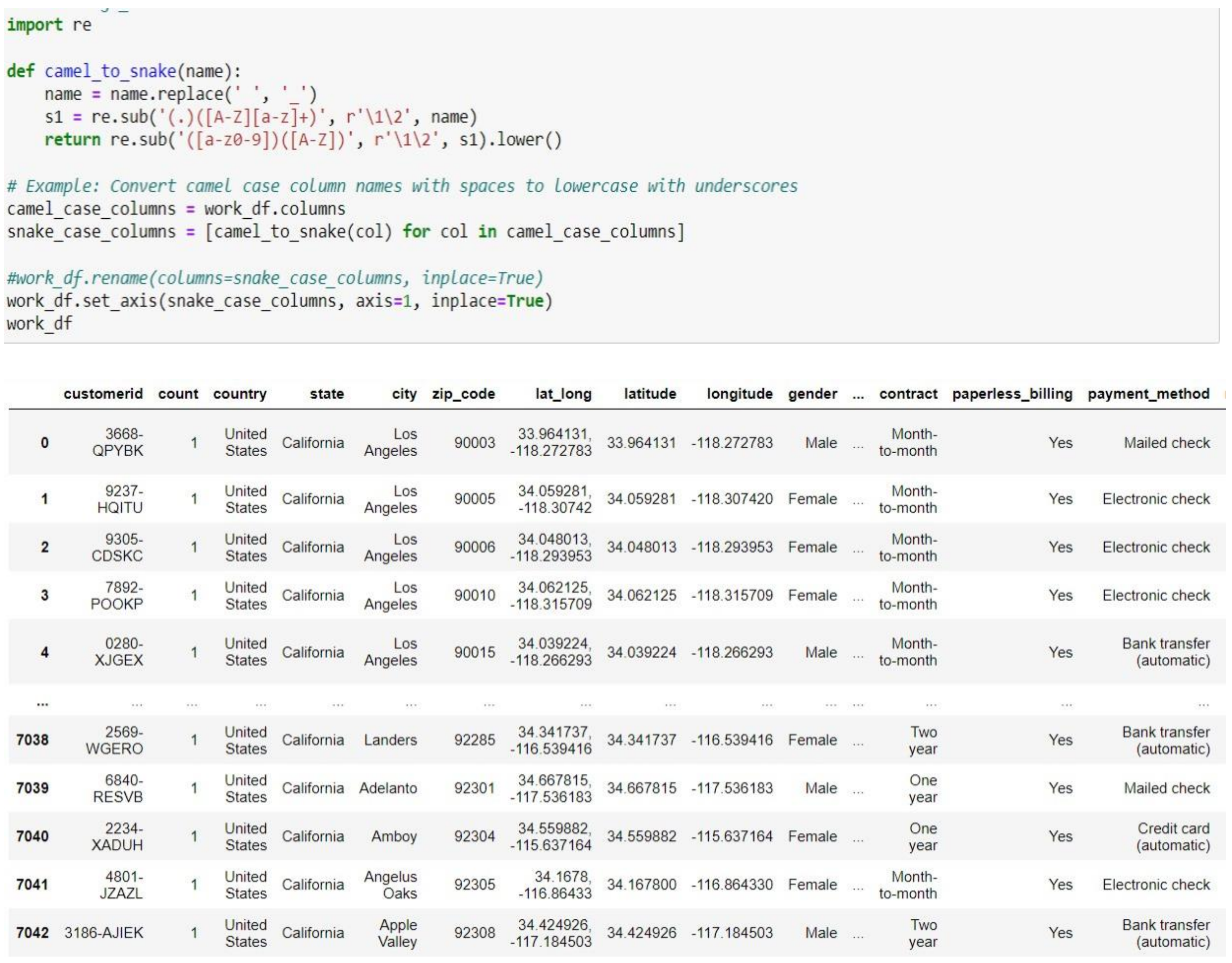
I recently obtained valuable data from the IBM platform. This information, acquired through a trusted source, offers a wealth of insights and knowledge. Leveraging the resources and capabilities of the IBM platform, I've gained access to a trove of data that will undoubtedly prove valuable for analysis, decision-making, and furthering our understanding of the subject matter at hand. This data, carefully extracted and securely downloaded, opens up exciting possibilities for exploration and research.

Here is the link of the data source

https://accelerator.ca.analytics.ibm.com/bi/?perspective=authoring&pathRef=.public_folders%2FIBM%2BAccelerator%2BCatalog%2FContent%2FDAT00148&id=i9710CF25EF75468D95FFFC7D57D45204&objRef=i9710CF25EF75468D95FFFC7D57D45204&action=run&format=HTML&cmPropStr=%7B%22id%22%3A%22i9710CF25EF75468D95FFFC7D57D45204%22%2C%22type%22%3A%22reportView%22%2C%22defaultName%22%3A%22DAT00148%22%2C%22permissions%22%3A%5B%22execute%22%2C%22read%22%2C%22traverse%22%5D%7D

3. Data Cleaning/Processing:

3.1 Case Conversion:

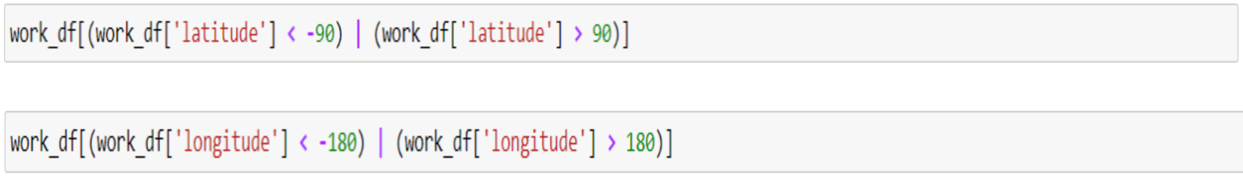


```python
import re

def camel_to_snake(name):
    name = name.replace(' ', '_')
    s1 = re.sub('(.)([A-Z][a-z]+)', r'\1_\2', name)
    return re.sub('([a-z0-9])([A-Z])', r'\1_\2', s1).lower()

# Example: Convert camel case column names with spaces to lowercase with underscores
camel_case_columns = work_df.columns
snake_case_columns = [camel_to_snake(col) for col in camel_case_columns]

#work_df.rename(columns=snake_case_columns, inplace=True)
work_df.set_axis(snake_case_columns, axis=1, inplace=True)
work_df
```

	customerid	count	country	state	city	zip_code	lat_long	latitude	longitude	gender	...	contract	paperless_billing	payment_method
0	3668-QPYBK	1	United States	California	Los Angeles	90003	33.964131, -118.272783	33.964131	-118.272783	Male	...	Month-to-month	Yes	Mailed check
1	9237-HQITU	1	United States	California	Los Angeles	90005	34.059281, -118.30742	34.059281	-118.307420	Female	...	Month-to-month	Yes	Electronic check
2	9305-CDSKC	1	United States	California	Los Angeles	90006	34.048013, -118.293953	34.048013	-118.293953	Female	...	Month-to-month	Yes	Electronic check
3	7892-POOKP	1	United States	California	Los Angeles	90010	34.062125, -118.315709	34.062125	-118.315709	Female	...	Month-to-month	Yes	Electronic check
4	0280-XJGEX	1	United States	California	Los Angeles	90015	34.039224, -118.266293	34.039224	-118.266293	Male	...	Month-to-month	Yes	Bank transfer (automatic)
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	2569-WGERO	1	United States	California	Landers	92285	34.341737, -116.539416	34.341737	-116.539416	Female	...	Two year	Yes	Bank transfer (automatic)
7039	6840-RESVB	1	United States	California	Adelanto	92301	34.667815, -117.536183	34.667815	-117.536183	Male	...	One year	Yes	Mailed check
7040	2234-XADUH	1	United States	California	Amboy	92304	34.559882, -115.637164	34.559882	-115.637164	Female	...	One year	Yes	Credit card (automatic)
7041	4801-JZAZL	1	United States	California	Angelus Oaks	92305	34.1678, -116.86433	34.167800	-116.864330	Female	...	Month-to-month	Yes	Electronic check
7042	3186-AJIEK	1	United States	California	Apple Valley	92308	34.424926, -117.184503	34.424926	-117.184503	Male	...	Two year	Yes	Bank transfer (automatic)

Figure-1:Code and outputs of case conversion

3.2 Data consistency:



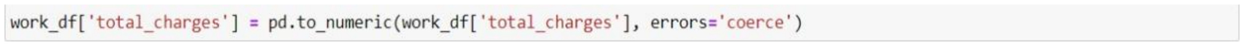
```
work_df[(work_df['latitude'] < -90) | (work_df['latitude'] > 90)]
```

```
work_df[(work_df['longitude'] < -180) | (work_df['longitude'] > 180)]
```

Figure-2: Code to check the validation of data

We are using the above snippet of code to perform data consistency(which is same as data validation in this case) on latitude and longitude features to check they are greater than 90 and less than -90 in case of latitude and greater than 180 and less than -180 in case of longitude.

3.3 Formatting the data:

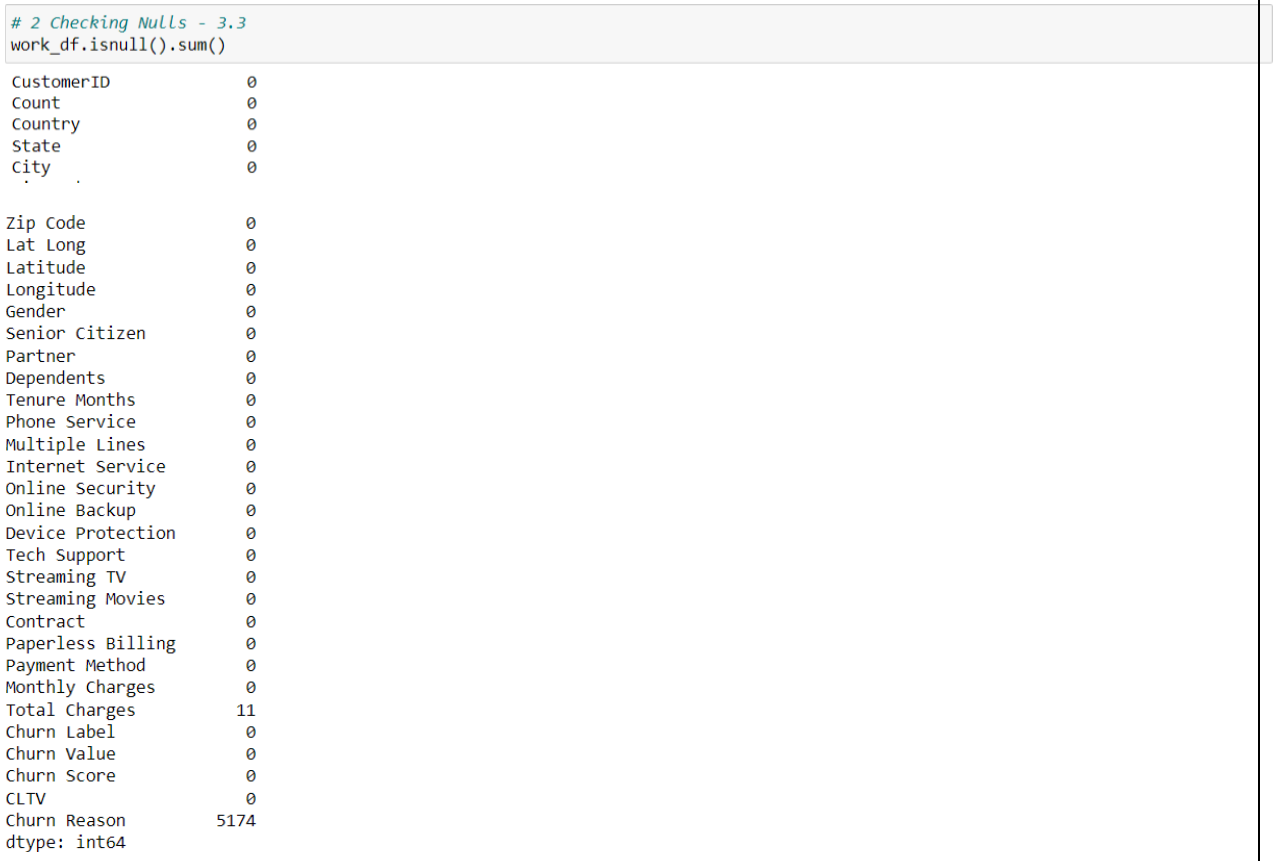


```
work_df['total_charges'] = pd.to_numeric(work_df['total_charges'], errors='coerce')
```

Figure-3: Code to format the data

This line of code is used to change the data type of “total_charges” column to numerical data type(i.e float) which actually had object type data before formatting.

3.4 Drop/Remove missing values:

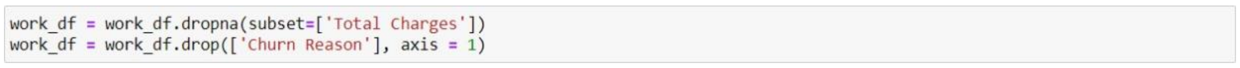


```
# 2 Checking Nulls - 3.3
work_df.isnull().sum()
```

```
CustomerID              0
Count                   0
Country                 0
State                   0
City                    0

Zip Code                0
Lat Long                0
Latitude                0
Longitude               0
Gender                  0
Senior Citizen          0
Partner                 0
Dependents              0
Tenure Months           0
Phone Service           0
Multiple Lines          0
Internet Service        0
Online Security         0
Online Backup           0
Device Protection       0
Tech Support            0
Streaming TV            0
Streaming Movies        0
Contract                0
Paperless Billing       0
Payment Method          0
Monthly Charges         0
Total Charges          11
Churn Label             0
Churn Value             0
Churn Score             0
CLTV                    0
Churn Reason         5174
dtype: int64
```

Figure-4 Code to check Nulls

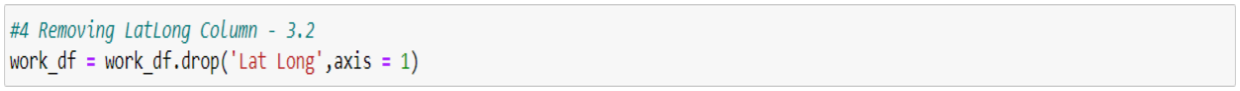
We used the above snippet of code to check if there are any missing values in any of the columns of our dataset. After executing the code we got that the “total charges” column has 11 null values and the “churn reason” column has 5174 null values. Since our data itself consists of 7038 rows of data and having a column with 5174 null values in our dataset doesn’t make any sense. So we are removing both “total charges” and “churn reason” columns from our database.



```
work_df = work_df.dropna(subset=['Total Charges'])
work_df = work_df.drop(['Churn Reason'], axis = 1)
```

Figure-4: Code to drop “Total Charges” and “Churn reason” columns

3.5 Drop unwanted columns:

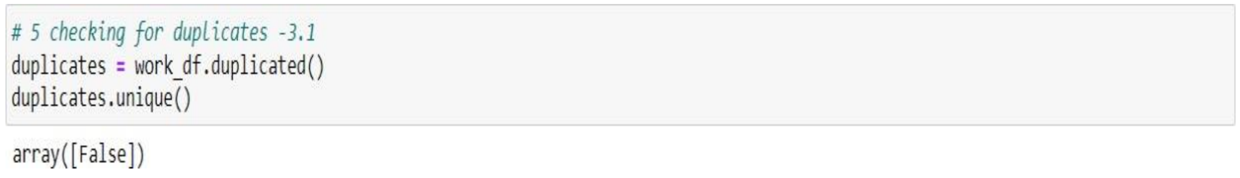


```
#4 Removing LatLong Column - 3.2
work_df = work_df.drop('Lat Long',axis = 1)
```

Figure-5: Code to drop LatLong column

We are using the above snippet of code to remove the "LatLong" column/feature from our dataset since we already have latitude and longitude columns individually.

3.6. Removing Duplicates:



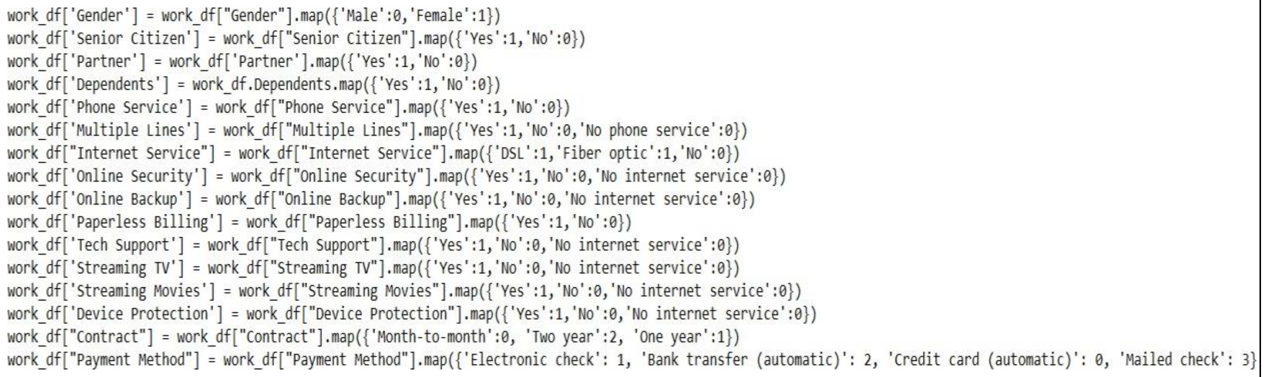
```
# 5 checking for duplicates -3.1
duplicates = work_df.duplicated()
duplicates.unique()
```

```
array([False])
```

Figure-6: Code for checking duplicates

Using the above snippet of code we are checking if there are any duplicate rows present in our dataset. General procedure involves first checking, removing them using drop_duplicates() function and then saving the clean data back to the csv file. Since our dataset does not contain any duplicate rows we are stopping our procedure after checking.

3.7 Converting Categorical to Numerical:

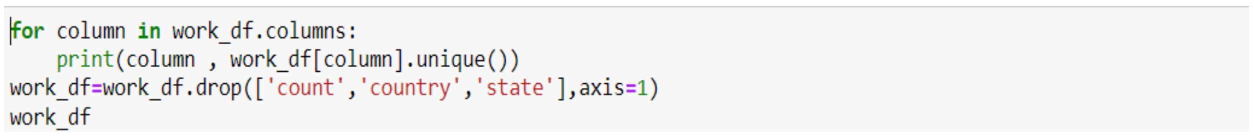


```
work_df['Gender'] = work_df["Gender"].map({'Male':0,'Female':1})
work_df['Senior Citizen'] = work_df["Senior Citizen"].map({'Yes':1,'No':0})
work_df['Partner'] = work_df['Partner'].map({'Yes':1,'No':0})
work_df['Dependents'] = work_df.Dependents.map({'Yes':1,'No':0})
work_df['Phone Service'] = work_df["Phone Service"].map({'Yes':1,'No':0})
work_df['Multiple Lines'] = work_df["Multiple Lines"].map({'Yes':1,'No':0,'No phone service':0})
work_df["Internet Service"] = work_df["Internet Service"].map({'DSL':1,'Fiber optic':1,'No':0})
work_df['Online Security'] = work_df["Online Security"].map({'Yes':1,'No':0,'No internet service':0})
work_df['Online Backup'] = work_df["Online Backup"].map({'Yes':1,'No':0,'No internet service':0})
work_df['Paperless Billing'] = work_df["Paperless Billing"].map({'Yes':1,'No':0})
work_df['Tech Support'] = work_df["Tech Support"].map({'Yes':1,'No':0,'No internet service':0})
work_df['Streaming TV'] = work_df["Streaming TV"].map({'Yes':1,'No':0,'No internet service':0})
work_df['Streaming Movies'] = work_df["Streaming Movies"].map({'Yes':1,'No':0,'No internet service':0})
work_df['Device Protection'] = work_df["Device Protection"].map({'Yes':1,'No':0,'No internet service':0})
work_df["Contract"] = work_df["Contract"].map({'Month-to-month':0, 'Two year':2, 'One year':1})
work_df["Payment Method"] = work_df["Payment Method"].map({'Electronic check': 1, 'Bank transfer (automatic)': 2, 'Credit card (automatic)': 0, 'Mailed check': 3}
```

Figure-7: Code for one-hot encoding

The above snippet of code is used for one-hot encoding where we are transforming the data of columns with categorical data such as “gender”, “senior_citizen” etc into binary columns(i.e either 1 or 0).

3.8 Feature Reduction:

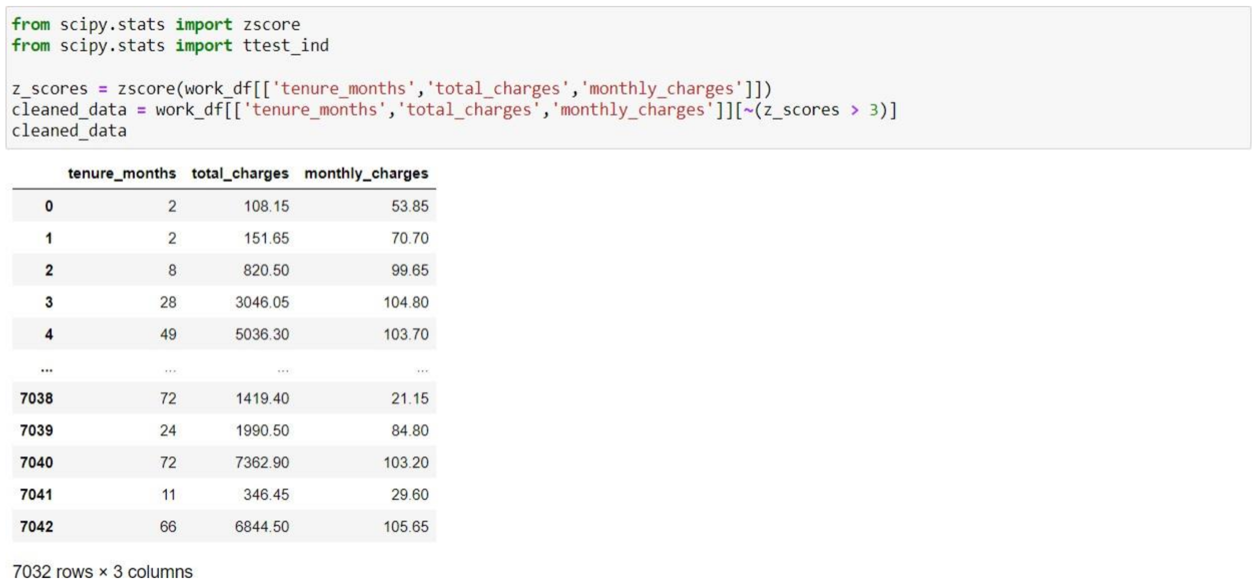


```python
for column in work_df.columns:
    print(column , work_df[column].unique())
work_df=work_df.drop(['count','country','state'],axis=1)
work_df
```

Figure-8: Code to drop features

We are using the above snippet of code to drop the “count”, “country” and “state” features from our dataset since we can retrieve the country and state details using zipcode, latitude and longitude features present in our dataset and the count column consists of same value for all the rows which makes it ineffective.

3.9 Detection of outliers:



```python
from scipy.stats import zscore
from scipy.stats import ttest_ind

z_scores = zscore(work_df[['tenure_months','total_charges','monthly_charges']])
cleaned_data = work_df[['tenure_months','total_charges','monthly_charges']][~(z_scores > 3)]
cleaned_data
```

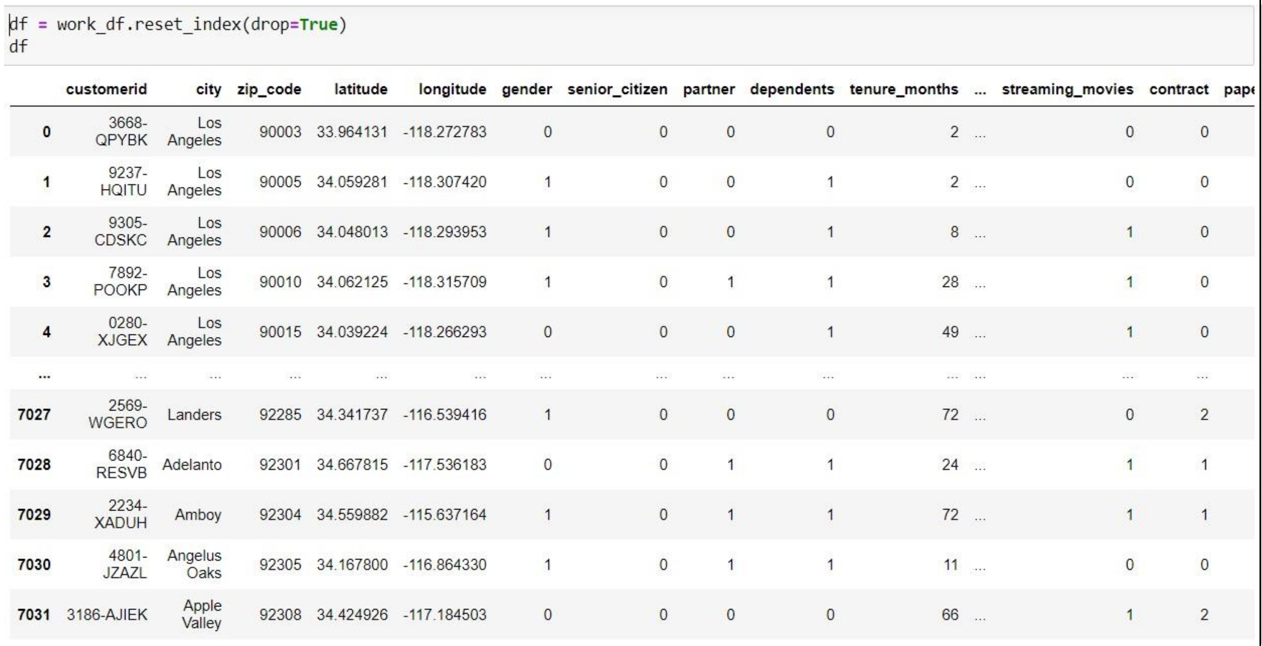
	tenure_months	total_charges	monthly_charges
0	2	108.15	53.85
1	2	151.65	70.70
2	8	820.50	99.65
3	28	3046.05	104.80
4	49	5036.30	103.70
...	...	...	...
7038	72	1419.40	21.15
7039	24	1990.50	84.80
7040	72	7362.90	103.20
7041	11	346.45	29.60
7042	66	6844.50	105.65

7032 rows × 3 columns

Figure-9: Code to check outliers

Here we are checking if there are any outliers present in our dataset. Most of the columns in our dataset have categorical data which is fine and do not have outliers in them. So we are performing this process of outlier detection only on columns that contain numerical data which are “tenure_months”, “total_charges” and “monthly_charges”. Here we are considering z value greater than 3 as the threshold value(a data point with z-value of 3 is significantly away from mean of the data) because in normal distributions 99.7 percent of data falls within three standard deviations of mean and only 0.3 percent of data is expected to have z-values greater than 3. This is the reason as to why z-value=3 is chosen as the threshold value to detect outliers in our dataset.

3.10 Resetting Index:



```
df = work_df.reset_index(drop=True)
df
```

	customerid	city	zip_code	latitude	longitude	gender	senior_citizen	partner	dependents	tenure_months	...	streaming_movies	contract	pape
0	3668-QPYBK	Los Angeles	90003	33.964131	-118.272783	0	0	0	0	2	...	0	0	
1	9237-HQITU	Los Angeles	90005	34.059281	-118.307420	1	0	0	1	2	...	0	0	
2	9305-CDSKC	Los Angeles	90006	34.048013	-118.293953	1	0	0	1	8	...	1	0	
3	7892-POOKP	Los Angeles	90010	34.062125	-118.315709	1	0	1	1	28	...	1	0	
4	0280-XJGEX	Los Angeles	90015	34.039224	-118.266293	0	0	0	1	49	...	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7027	2569-WGERO	Landers	92285	34.341737	-116.539416	1	0	0	0	72	...	0	2	
7028	6840-RESVB	Adelanto	92301	34.667815	-117.536183	0	0	1	1	24	...	1	1	
7029	2234-XADUH	Amboy	92304	34.559882	-115.637164	1	0	1	1	72	...	1	1	
7030	4801-JZAZL	Angelus Oaks	92305	34.167800	-116.864330	1	0	1	1	11	...	0	0	
7031	3186-AJIEK	Apple Valley	92308	34.424926	-117.184503	0	0	0	0	66	...	1	2	

Figure-10: Code and outputs of index reset

Since we have removed some of the rows from our dataset(because they contain null values) we need to perform index correction.

4. Exploratory Data Analysis:

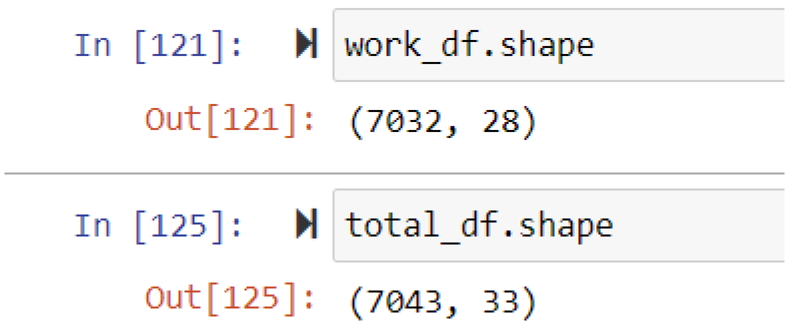
Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events and find interesting relations among the variables.

4.1 Shape of Data:

Dataset size before Data Cleaning : (7043,33)

Dataset size before Data Cleaning: (7032,28)

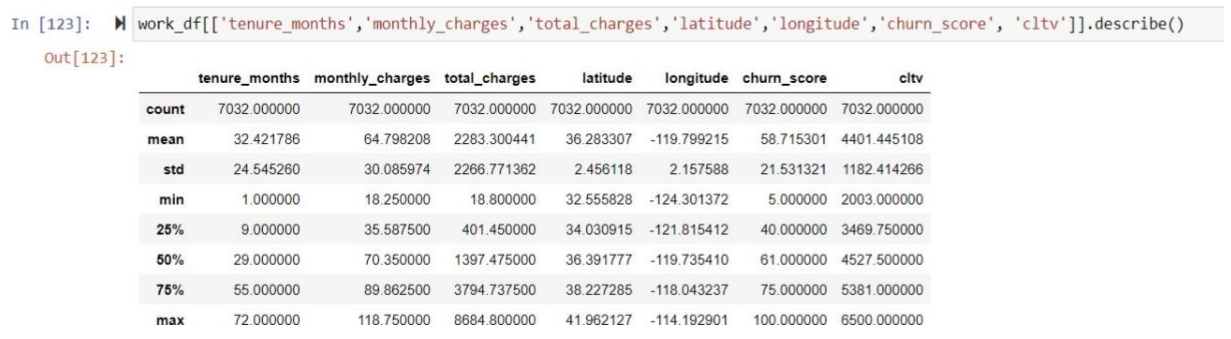


```
In [121]:   work_df.shape
Out[121]:   (7032, 28)

In [125]:   total_df.shape
Out[125]:   (7043, 33)
```

Figure -11 : shapes of datasets before and after cleaning.

4.2 Summary statistics of numerical data :

For the numerical features in the dataset, as these are continuous, statistics measures of data are as shown below.



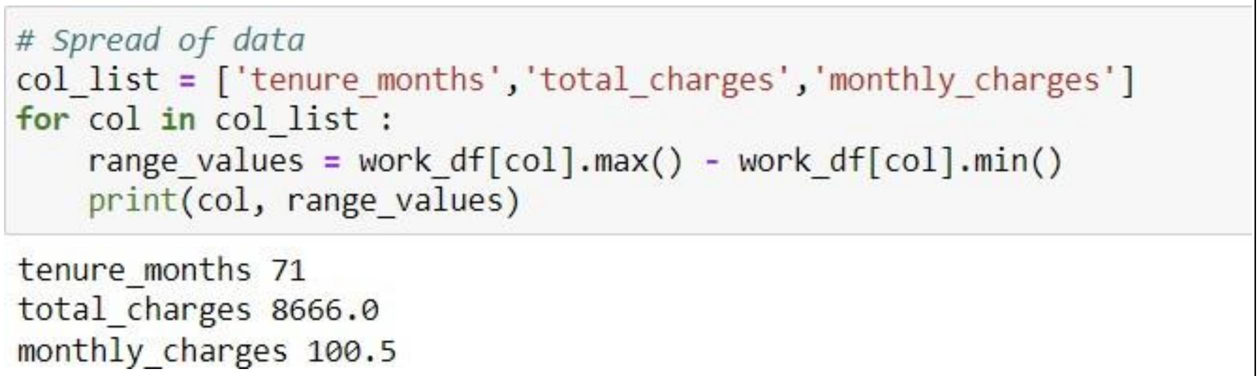
```
In [123]:  work_df[['tenure_months','monthly_charges','total_charges','latitude','longitude','churn_score', 'cltv']].describe()
Out[123]:
```

	tenure_months	monthly_charges	total_charges	latitude	longitude	churn_score	cltv
count	7032.000000	7032.000000	7032.000000	7032.000000	7032.000000	7032.000000	7032.000000
mean	32.421786	64.798208	2283.300441	36.283307	-119.799215	58.715301	4401.445108
std	24.545260	30.085974	2266.771362	2.456118	2.157588	21.531321	1182.414266
min	1.000000	18.250000	18.800000	32.555828	-124.301372	5.000000	2003.000000
25%	9.000000	35.587500	401.450000	34.030915	-121.815412	40.000000	3469.750000
50%	29.000000	70.350000	1397.475000	36.391777	-119.735410	61.000000	4527.500000
75%	55.000000	89.862500	3794.737500	38.227285	-118.043237	75.000000	5381.000000
max	72.000000	118.750000	8684.800000	41.962127	-114.192901	100.000000	6500.000000

Figure -12 : Statistic of numerical data

4.3 Spread of data:

For the numerical features in dataset, below are the ranges :



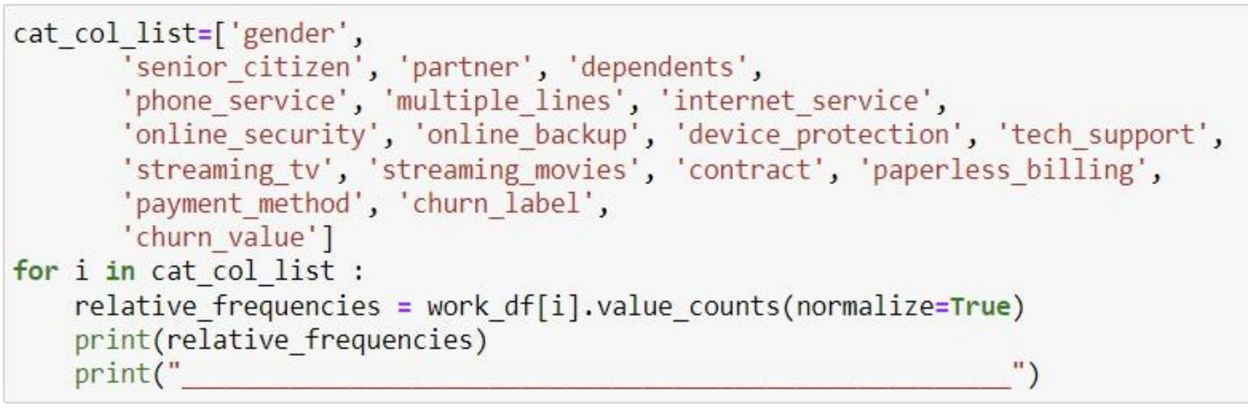
```
# Spread of data
col_list = ['tenure_months','total_charges','monthly_charges']
for col in col_list :
    range_values = work_df[col].max() - work_df[col].min()
    print(col, range_values)
```

```
tenure_months 71
total_charges 8666.0
monthly_charges 100.5
```

Figure -13 : Range of numerical data

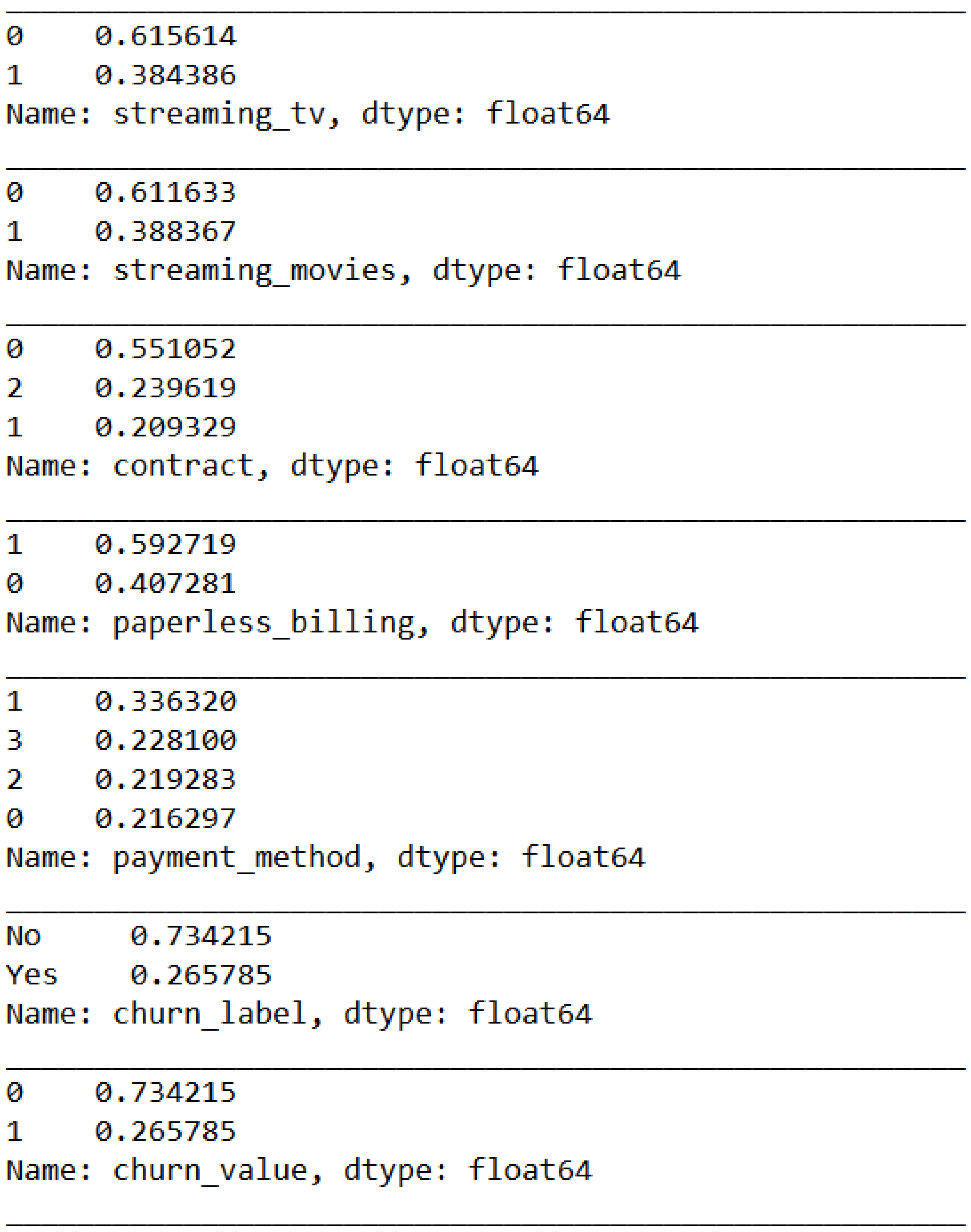
4.4 Summary statistics of categorical data

For the categorical features in dataset, distribution of data is as shown below



```python
cat_col_list=['gender',
       'senior_citizen', 'partner', 'dependents',
       'phone_service', 'multiple_lines', 'internet_service',
       'online_security', 'online_backup', 'device_protection', 'tech_support',
       'streaming_tv', 'streaming_movies', 'contract', 'paperless_billing',
       'payment_method', 'churn_label',
       'churn_value']
for i in cat_col_list :
    relative_frequencies = work_df[i].value_counts(normalize=True)
    print(relative_frequencies)
    print("_____________________________________________________")
```

```
0    0.504693
1    0.495307
Name: gender, dtype: float64
______________________________________________________
0    0.8376
1    0.1624
Name: senior_citizen, dtype: float64
______________________________________________________
0    0.517491
1    0.482509
Name: partner, dtype: float64
______________________________________________________
0    0.769625
1    0.230375
Name: dependents, dtype: float64
______________________________________________________
1    0.903299
0    0.096701
Name: phone_service, dtype: float64
______________________________________________________
0    0.578072
1    0.421928
Name: multiple_lines, dtype: float64
______________________________________________________
1    0.783845
0    0.216155
Name: internet_service, dtype: float64
______________________________________________________
0    0.713453
1    0.286547
Name: online_security, dtype: float64
______________________________________________________
0    0.655148
1    0.344852
Name: online_backup, dtype: float64
______________________________________________________
0    0.656143
1    0.343857
Name: device_protection, dtype: float64
______________________________________________________
0    0.709898
1    0.290102
Name: tech_support, dtype: float64
```

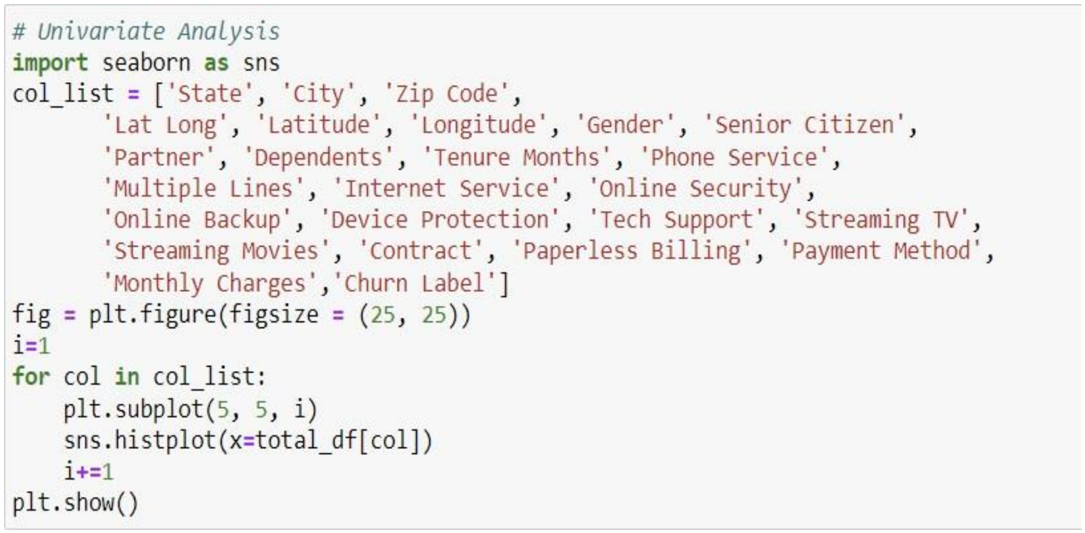


```
__________________________________________________
0    0.615614
1    0.384386
Name: streaming_tv, dtype: float64
__________________________________________________
0    0.611633
1    0.388367
Name: streaming_movies, dtype: float64
__________________________________________________
0    0.551052
2    0.239619
1    0.209329
Name: contract, dtype: float64
__________________________________________________
1    0.592719
0    0.407281
Name: paperless_billing, dtype: float64
__________________________________________________
1    0.336320
3    0.228100
2    0.219283
0    0.216297
Name: payment_method, dtype: float64
__________________________________________________
No     0.734215
Yes    0.265785
Name: churn_label, dtype: float64
__________________________________________________
0    0.734215
1    0.265785
Name: churn_value, dtype: float64
__________________________________________________
```

Figure -14 : Distributions of categorical data

4.5 Univariate Analysis:

Univariate analysis is a technique of analyzing a single variable at time to understand its distributions and characteristics. As data consists of categorical variables, frequency distribution is used.



```python
# Univariate Analysis
import seaborn as sns
col_list = ['State', 'City', 'Zip Code',
       'Lat Long', 'Latitude', 'Longitude', 'Gender', 'Senior Citizen',
       'Partner', 'Dependents', 'Tenure Months', 'Phone Service',
       'Multiple Lines', 'Internet Service', 'Online Security',
       'Online Backup', 'Device Protection', 'Tech Support', 'Streaming TV',
       'Streaming Movies', 'Contract', 'Paperless Billing', 'Payment Method',
       'Monthly Charges','Churn Label']
fig = plt.figure(figsize = (25, 25))
i=1
for col in col_list:
    plt.subplot(5, 5, i)
    sns.histplot(x=total_df[col])
    i+=1
plt.show()
```

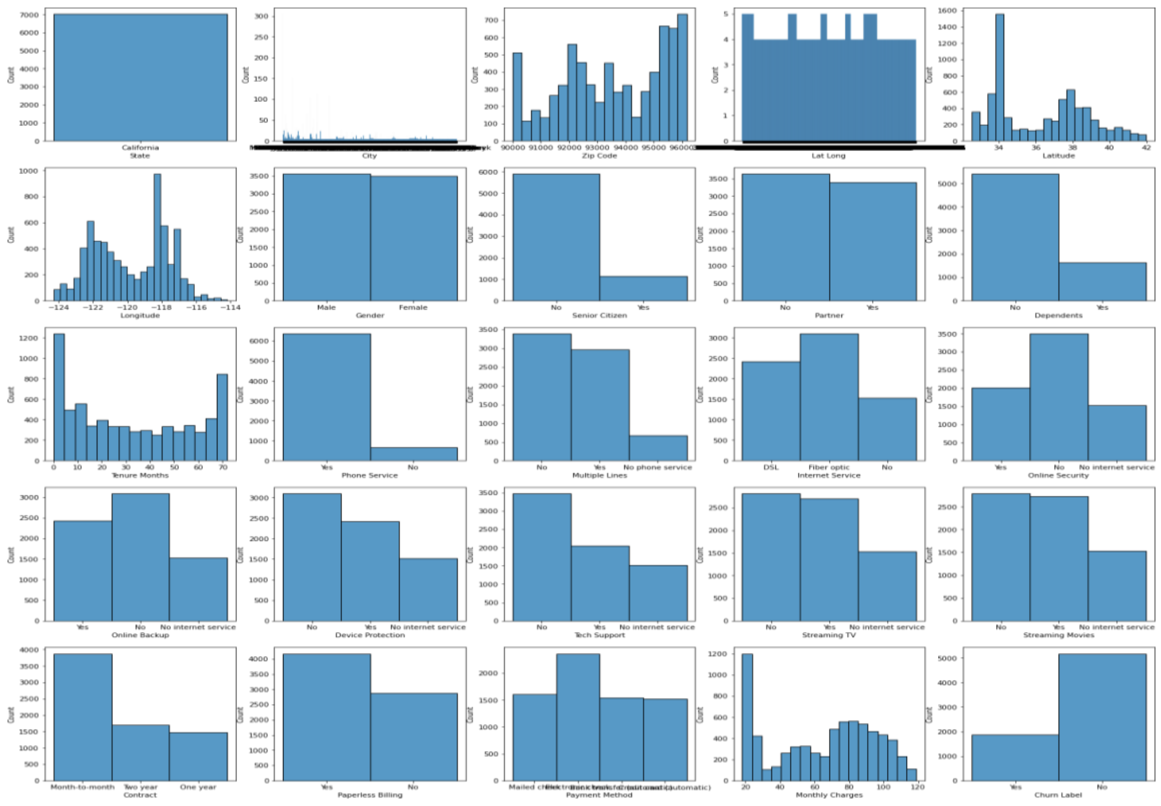


Figure -15 : Univariate Analysis

4.6 Variate Analysis with target:

It is also known as Bivariate Analysis with one of the variables as target. In this technique , frequency distributions of all the variables with target are plotted to understand its influence.

```python
# Variant analysis with target
fig = plt.figure(figsize = (25, 25))
col_list = [ 'city', 'zip_code', 'gender', 'senior_citizen', 'partner',
       'dependents', 'tenure_months', 'phone_service', 'multiple_lines',
       'internet_service', 'online_security', 'online_backup',
       'device_protection', 'tech_support', 'streaming_tv', 'streaming_movies',
       'contract', 'paperless_billing', 'payment_method', 'monthly_charges',
       'total_charges','churn_value', 'cltv','churn_label']
data_cat=work_df[col_list]
i=1
for x in col_list[:-1]:
    plt.subplot(5, 5, i)
    ax=sns.countplot(data =data_cat , x = data_cat[x], hue ='churn_label', palette = ["#95BEDD", '#FF5959'])
    ax.set(xlabel = None, ylabel = None)
    plt.title(str(x), loc='center')
    i+=1
plt.show()
```

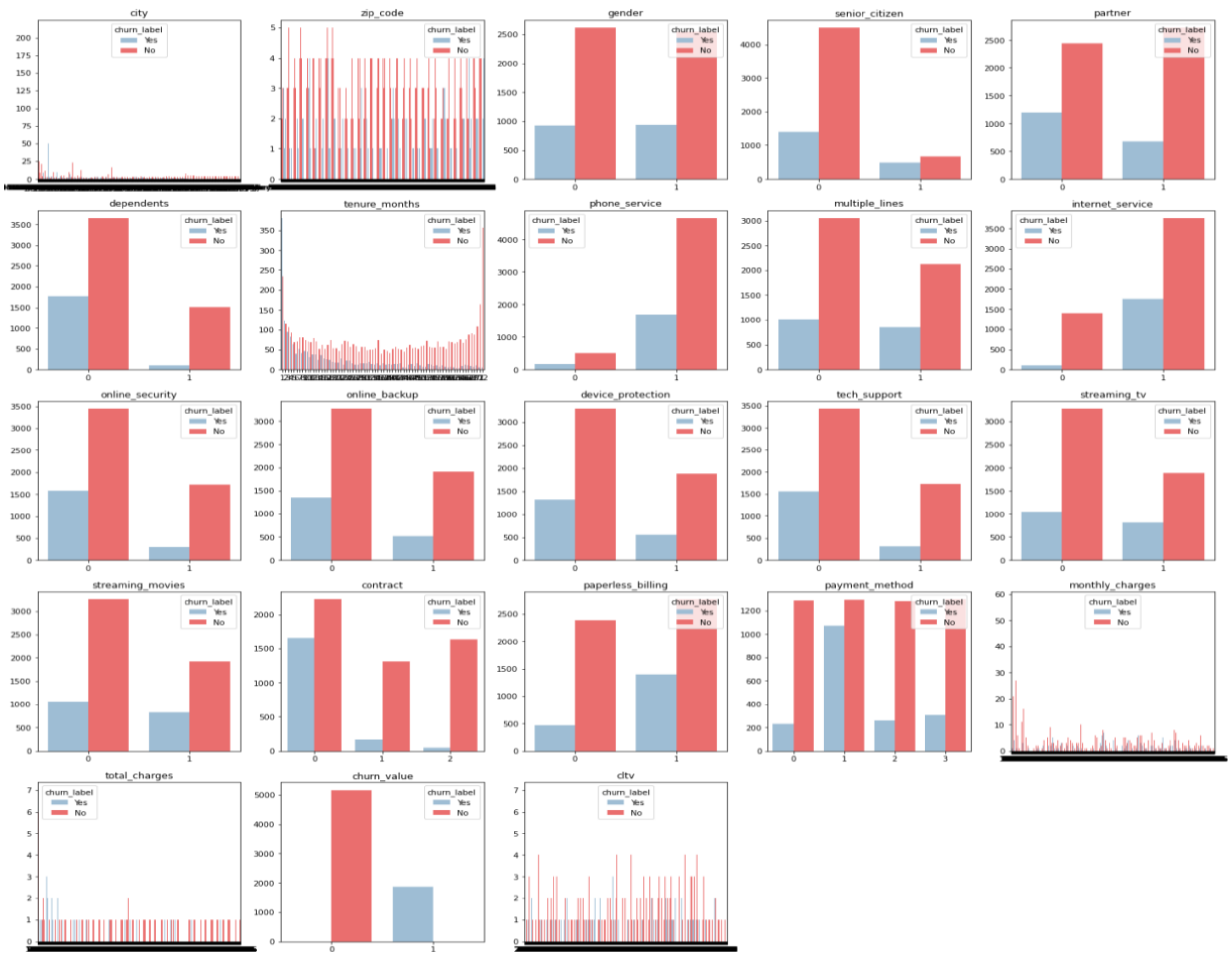


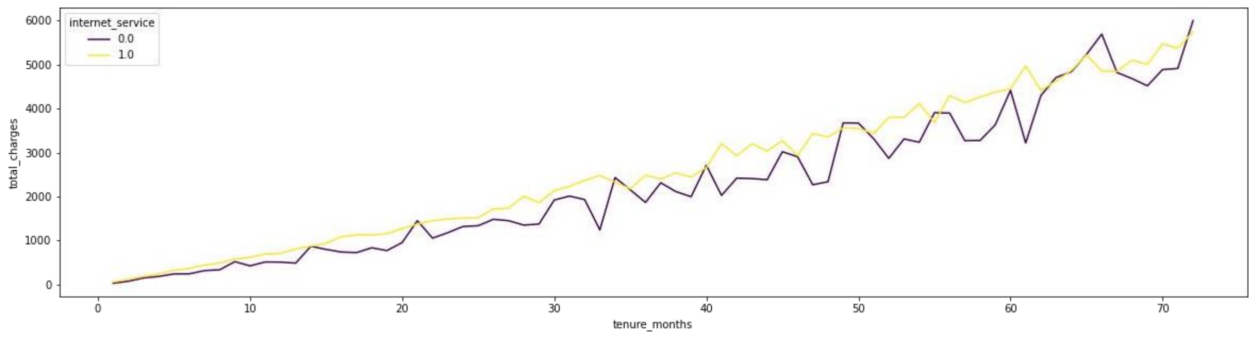
Figure -16 : Bivariate Analysis with target variable

4.7 Multivariate Analysis:

In this technique ,behavior of multiple variables in accordance with target variable are plotted to understand patterns in the data.

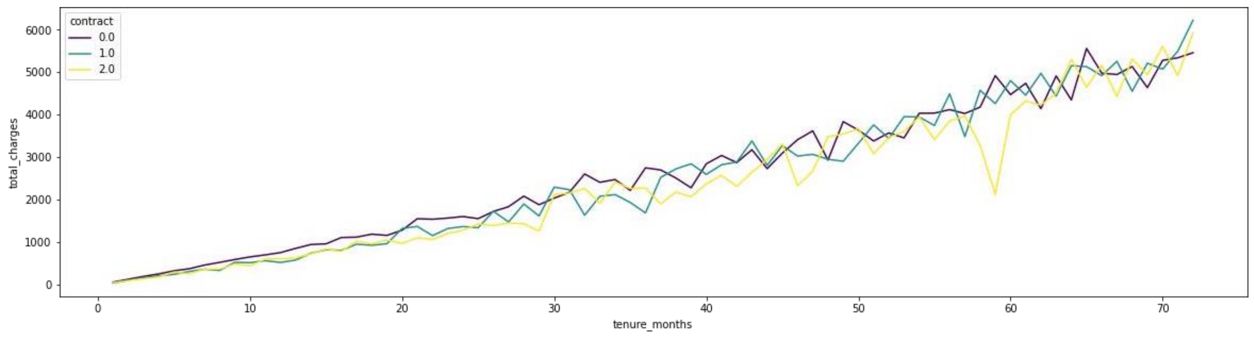
```
plt.figure(figsize=(20,5))
sns.lineplot(x=work_df['tenure_months'], y=work_df['total_charges'], hue=df['internet_service'],
             ci=None, palette='viridis')
```

```
<AxesSubplot:xlabel='tenure_months', ylabel='total_charges'>
```



```
plt.figure(figsize=(20,5))
sns.lineplot(x=work_df['tenure_months'], y=work_df['total_charges'], hue=df['contract'], ci=None, palette='viridis')
```

```
<AxesSubplot:xlabel='tenure_months', ylabel='total_charges'>
```



```
plt.figure(figsize=(20,5))
sns.lineplot(x=df['tenure_months'], y=df['monthly_charges'], hue=df['churn_value'], ci=None, palette='viridis')
```

```
<AxesSubplot:xlabel='tenure_months', ylabel='monthly_charges'>
```

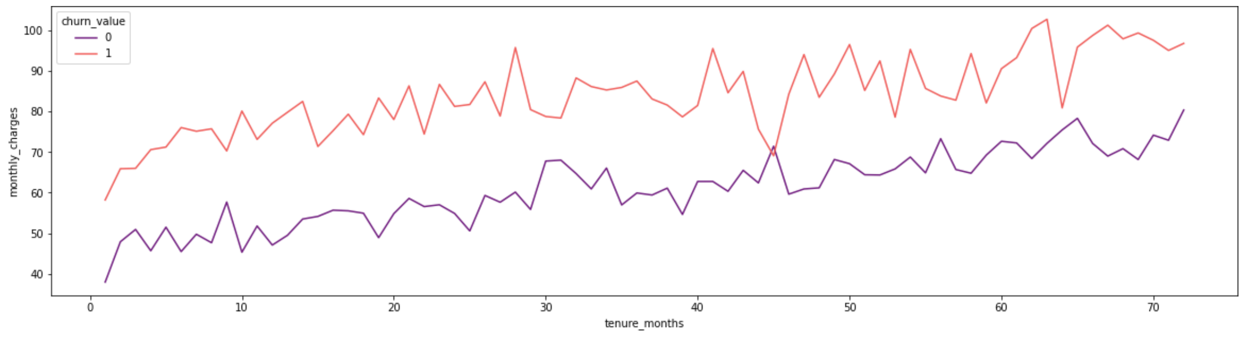
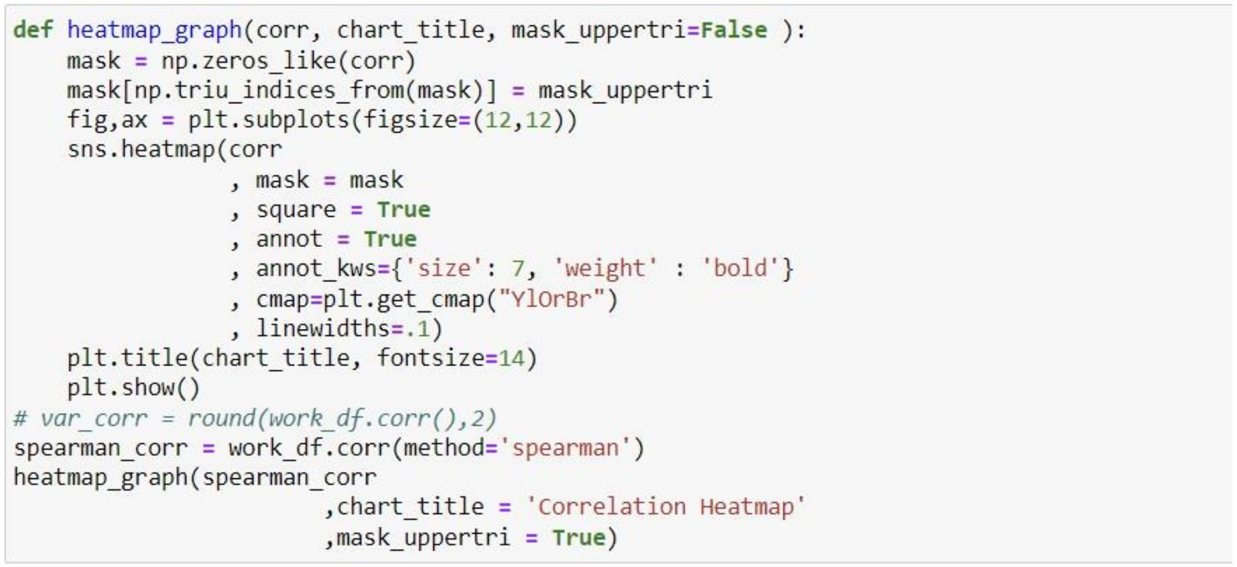


Figure -17 :Multivariate Analysis with target variable as group by

4.8 Correlation HeatMap

Correlations of all the combinations of variables are calculated and are plotted as heatmap for quick interaction to understand dependence of variables. As data involves categorical and numerical , spearman correlation is used.



```
def heatmap_graph(corr, chart_title, mask_uppertri=False ):
    mask = np.zeros_like(corr)
    mask[np.triu_indices_from(mask)] = mask_uppertri
    fig,ax = plt.subplots(figsize=(12,12))
    sns.heatmap(corr
                , mask = mask
                , square = True
                , annot = True
                , annot_kws={'size': 7, 'weight' : 'bold'}
                , cmap=plt.get_cmap("YlOrBr")
                , linewidths=.1)
    plt.title(chart_title, fontsize=14)
    plt.show()
# var_corr = round(work_df.corr(),2)
spearman_corr = work_df.corr(method='spearman')
heatmap_graph(spearman_corr
                       ,chart_title = 'Correlation Heatmap'
                       ,mask_uppertri = True)
```

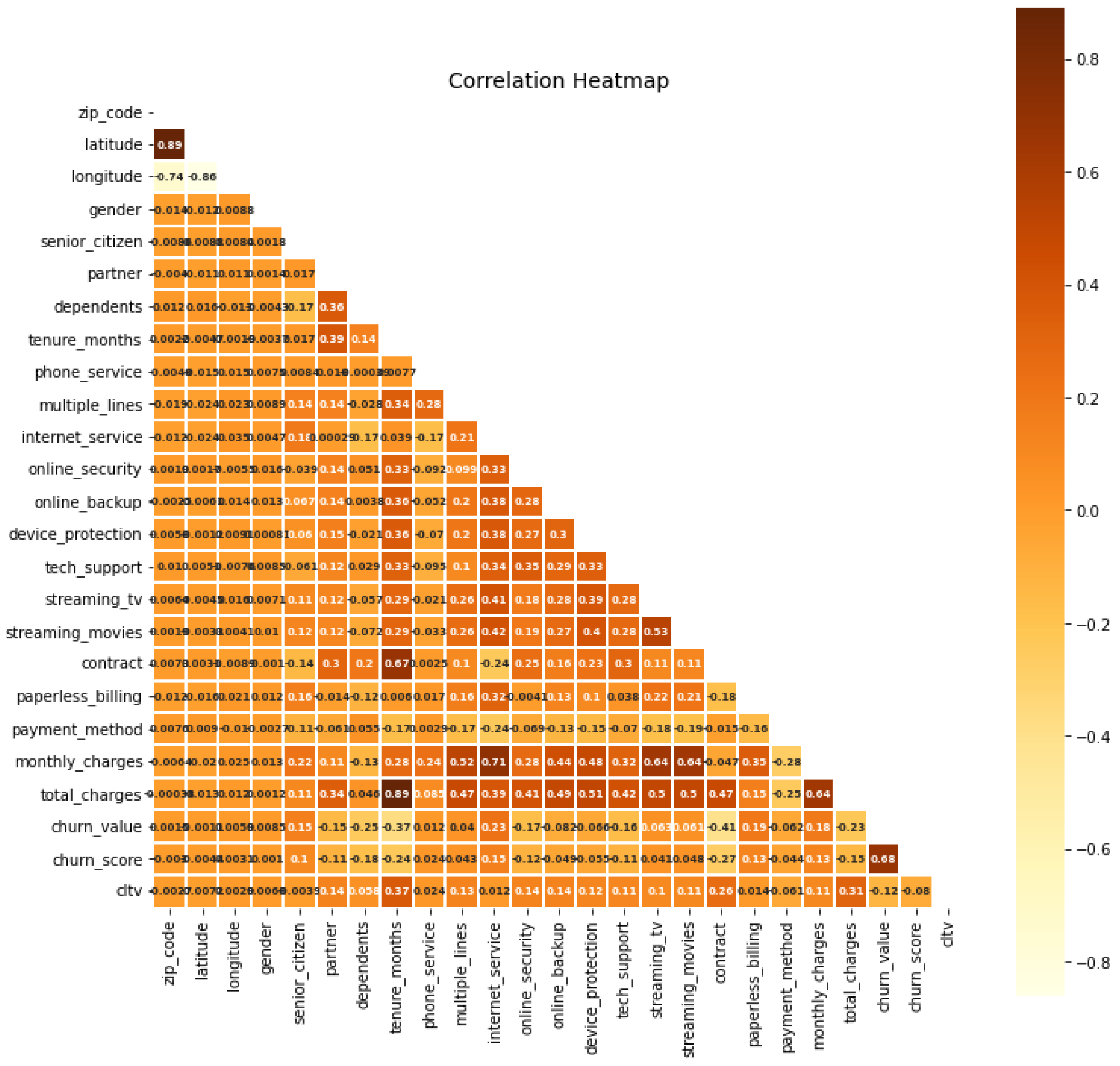
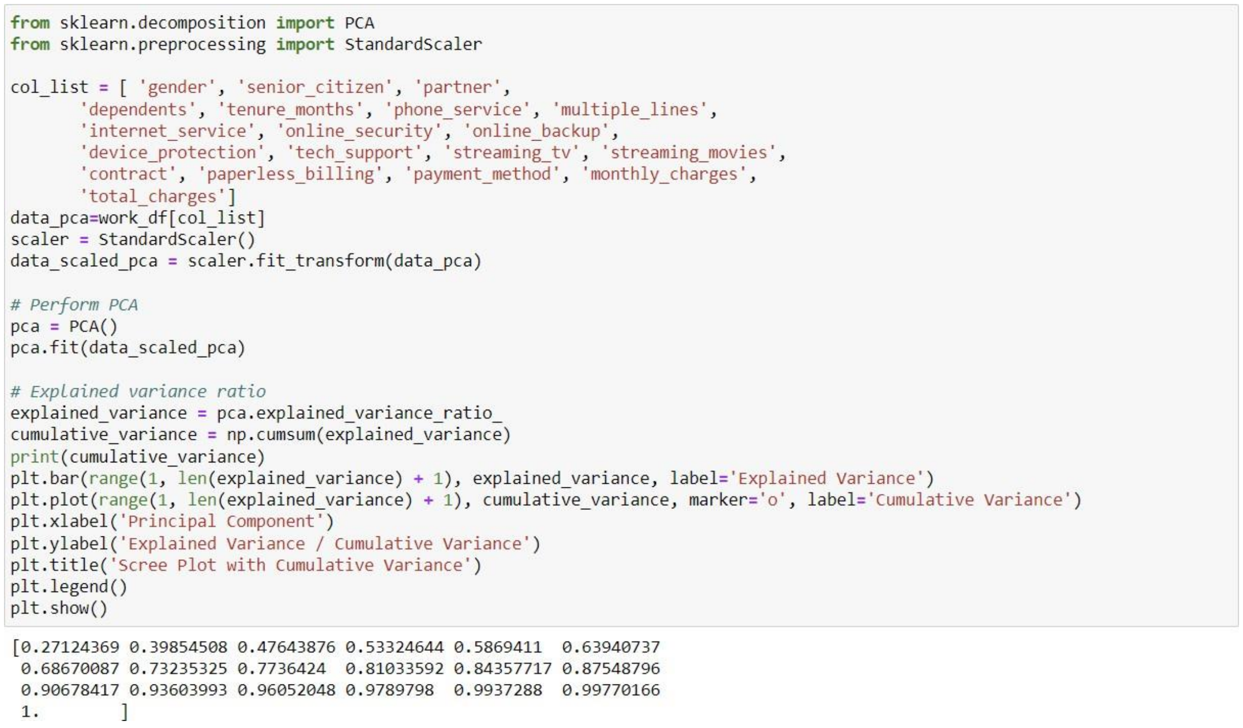


Figure -18 : Correlation heatmap

4.9 PCA scree plot:

PCA is done for the data using all the features after cleaning data to understand the variance of data behavior and to reduce features.



```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

col_list = [ 'gender', 'senior_citizen', 'partner',
       'dependents', 'tenure_months', 'phone_service', 'multiple_lines',
       'internet_service', 'online_security', 'online_backup',
       'device_protection', 'tech_support', 'streaming_tv', 'streaming_movies',
       'contract', 'paperless_billing', 'payment_method', 'monthly_charges',
       'total_charges']
data_pca=work_df[col_list]
scaler = StandardScaler()
data_scaled_pca = scaler.fit_transform(data_pca)

# Perform PCA
pca = PCA()
pca.fit(data_scaled_pca)

# Explained variance ratio
explained_variance = pca.explained_variance_ratio_
cumulative_variance = np.cumsum(explained_variance)
print(cumulative_variance)
plt.bar(range(1, len(explained_variance) + 1), explained_variance, label='Explained Variance')
plt.plot(range(1, len(explained_variance) + 1), cumulative_variance, marker='o', label='Cumulative Variance')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance / Cumulative Variance')
plt.title('Scree Plot with Cumulative Variance')
plt.legend()
plt.show()
```

```
[0.27124369 0.39854508 0.47643876 0.53324644 0.5869411  0.63940737
 0.68670087 0.73235325 0.7736424  0.81033592 0.84357717 0.87548796
 0.90678417 0.93603993 0.96052048 0.9789798  0.9937288  0.99770166
 1.        ]
```

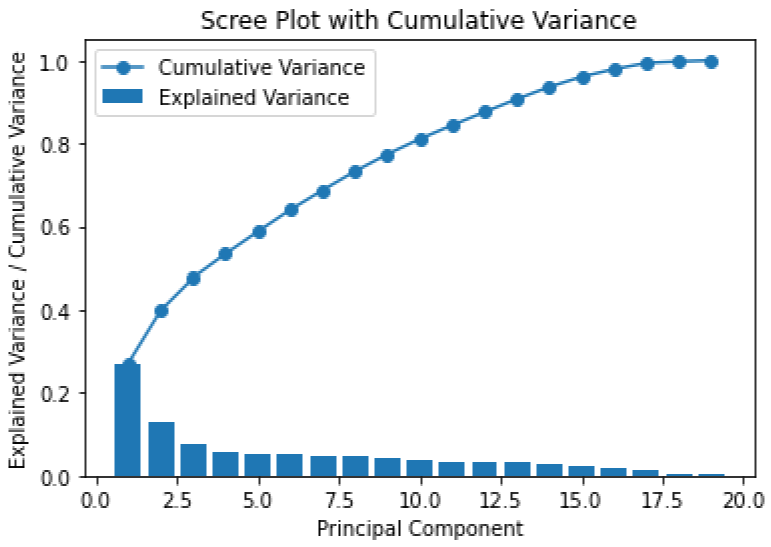


Figure -19 :PCA Scree plot

4.10 Feature Selection:

As per PCA , around 14 components should be used and as per correlation some of the unimpacted features can be discarded in the further analysis.

initial feature selection for further analysis : 'gender',

'senior_citizen', 'tenure_months',

'phone_service', 'multiple_lines', 'internet_service',

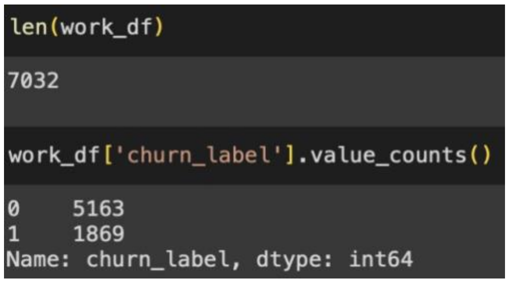
'online_security', 'online_backup', 'tech_support',

'streaming_tv', 'streaming_movies', , 'monthly_charges', 'total_charges'

These are subject to change when modeling the data and evaluating fits.

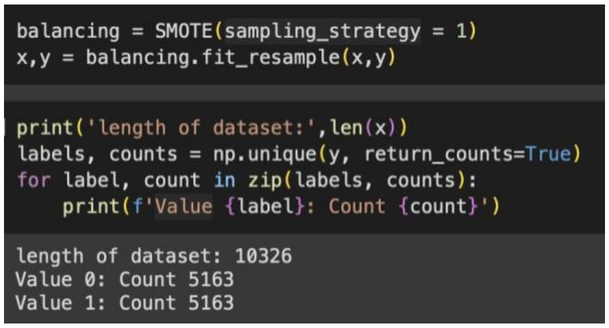
5. SMOTE ANALYSIS:

Before algorithm application, preprocessed data (dataset after applying steps from phase1) is to be verified for class imbalance as it is a major problem in classification tasks where one class significantly outnumbers the other. In a class-imbalanced dataset, the model might become biased towards the majority class, leading to poor performance on the minority class.
Length of the dataset is 7032 and the classes split of the churn label are about 84% and 16%, this indicates class imbalance in data.



```
len(work_df)

7032

work_df['churn_label'].value_counts()

0    5163
1    1869
Name: churn_label, dtype: int64
```

To address this, SMOTE technique is used over data. SMOTE helps to mitigate this issue by oversampling the minority class as it generates synthetic samples for the minority class, making it more balanced with the majority class.
SMOTE basically starts with identifying minority class Instances, then generates synthetic instances along the line segments connecting the selected instance and its neighbors. By repeating this process for multiple instances in the minority class, SMOTE increases the number of minority class samples, making it more balanced with the majority class.
Data is made class balanced using SMOTE and the length of the dataset is 10326 and is ready for algorithm application.



```
balancing = SMOTE(sampling_strategy = 1)
x,y = balancing.fit_resample(x,y)

print('length of dataset:',len(x))
labels, counts = np.unique(y, return_counts=True)
for label, count in zip(labels, counts):
    print(f'Value {label}: Count {count}')

length of dataset: 10326
Value 0: Count 5163
Value 1: Count 5163
```

6. Algorithm Application and Analysis:

Following are the 6 models that are used for the analysis. They are:

6.1 XGBClassifier
6.2 RandomForestClassifier
6.3 DecisionTreeClassifier
6.4 GaussianNB
6.5 Logistic Regression
6.6 SVM Classifier with Linear and Gaussian Kernels

The goal of this analysis is to draw conclusions and inferences that address the given problem statement. In all these analyses, most of them are used to evaluate feature importance which is to identify the features that are crucial for forecasting customer churn and aiding in customer retention.

6.1. XGBClassifier:

The XGBClassifier is part of the XGBoost (Extreme Gradient Boosting) library, which is a popular and efficient implementation of the gradient boosting algorithm. Gradient boosting is an ensemble learning method that combines the predictions of multiple decision trees to create a strong predictive model.

XGBoost uses an objective function that combines a loss function with regularization terms. The objective is optimized during the training process. It also incorporates L1 (LASSO) and L2 (Ridge) regularization terms into the objective function. This helps prevent overfitting and improves the model's generalization to new, unseen data.

Key Considerations:

1) Speed and Performance: XGBoost is computationally efficient and can handle large datasets quickly. It is optimized for parallel processing, making it suitable for multicore processors and it is designed to handle complex relationships and patterns in data, which can help in finding patterns in customer churning and features.
2) Regularization: The addition of L1 (LASSO) and L2 (Ridge) regularization terms in XGBoost's objective function functions as a form of penalty on the model's complexity. This regularization helps prevent overfitting, where the model becomes too intricate and performs well on training data but struggles with new, unseen data. L1 regularization encourages sparsity in feature selection, effectively ignoring less relevant features, while L2 regularization constrains the magnitude of the model's coefficients. By balancing these regularization techniques, XGBoost promotes a more robust and generalizable model, enhancing its performance on diverse datasets.
3) Ensemble Learning: XGBoost is an ensemble learning method that builds multiple weak decision trees and combines them to create a strong learner. Ensemble methods often result in more robust and accurate models.
4) Cross-Validation Support: The XGBoost library supports built-in cross-validation, facilitating the tuning of hyperparameters and providing a more reliable estimate of the model's performance.

Limitations:

1) Interpretability: XGBoost models can be challenging to interpret compared to simpler models like decision trees or logistic regression. Interpreting the model's predictions and understanding the feature importance is more challenging.

2) Hyperparameter Tuning: While XGBoost is powerful, it requires careful tuning of hyperparameters to achieve optimal performance. Finding the right set of hyperparameters is time-consuming.
3) Availability of Data: XGBoost may not perform well when we have a limited amount of customer churn data available for training and it is more data-hungry

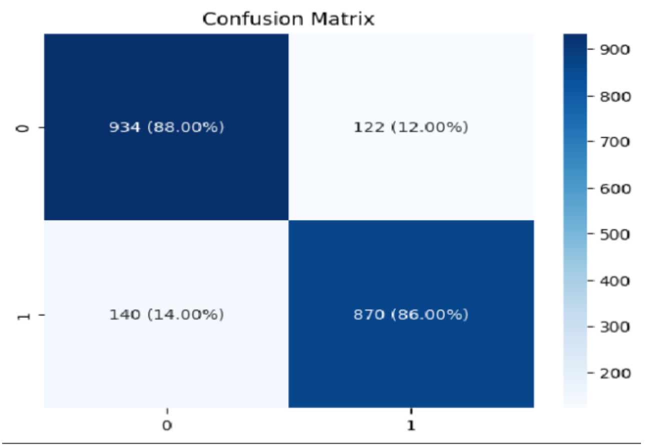
Results:

Metric	XGBClassifier
RMSE	0.356
ROC AUC	87.29%
Model accuracy	87.32%
Mean Absolute Error (MAE)	0.127

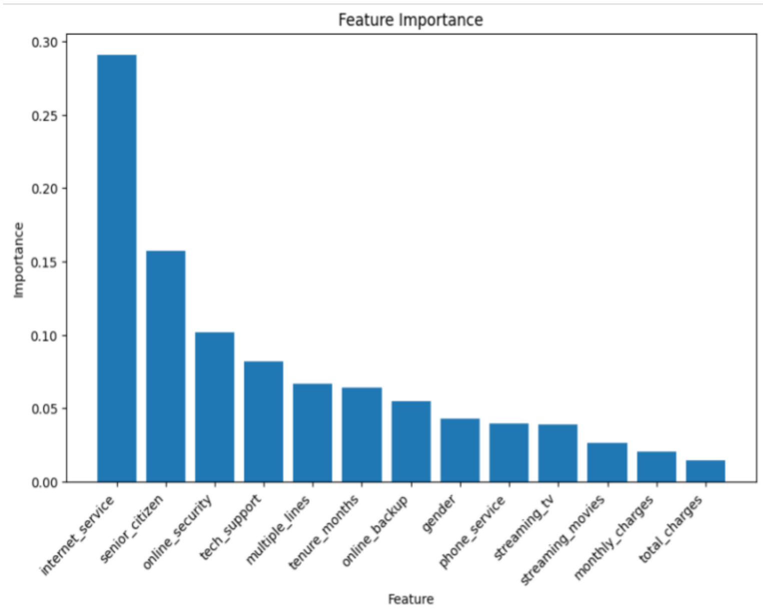
Classification Report:

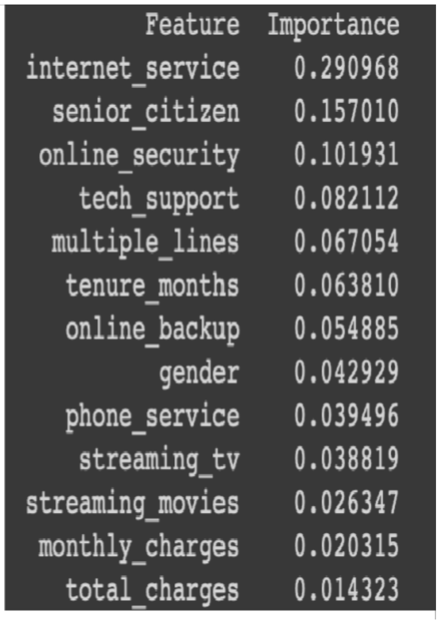
Class Label	Precision	Recall	F1-Score	Support
0	0.87	0.88	0.88	1056
1	0.88	0.86	0.87	1010

Confusion matrix:



Feature Importance:





```
           Feature  Importance
  internet_service    0.290968
    senior_citizen    0.157010
   online_security    0.101931
      tech_support    0.082112
    multiple_lines    0.067054
     tenure_months    0.063810
     online_backup    0.054885
            gender    0.042929
     phone_service    0.039496
      streaming_tv    0.038819
  streaming_movies    0.026347
   monthly_charges    0.020315
     total_charges    0.014323
```

Key Takeaways

We have learnt about the importance of features in forecasting churn which are internet_services, senior_citizen and online_security for this model. We have also understood the trade-offs between precision and recall, and obtained insights into the characteristics of consumers who are more likely to churn because of this approach. So, customers with internet services and online security and customers who are senior citizens are more likely to churn according to the results obtained from this modeling. We also learnt on how to fine-tune hyperparameters for an individual situation, as well as the significance of evaluating and displaying model findings.

6.2 RandomForestClassifier:

Random Forests are an extension of bagged decision trees, and they operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
Random forest classifiers are trained by creating a large number of decision trees on different random subsets of the training data. Each decision tree is then used to make predictions on a held-out test set. The final prediction of the random forest classifier is the average of the predictions from all the individual decision trees.

<u>Key Considerations:</u>

1) Robustness and Ensemble Learning: Random Forest is an ensemble learning method that combines the predictions of multiple decision trees. This ensemble approach often leads to more robust models, reducing the risk of overfitting and improving generalization to new, unseen data.
2) Handling Complex Relationships: Telco customer churn prediction may involve complex relationships and interactions among various features. Random Forests, by constructing multiple trees with random subsets of features, can capture these intricate patterns more effectively than individual decision trees.
3) Resistance to Overfitting: The random sampling of both training instances (through bagging) and features at each node helps prevent overfitting. This is crucial when dealing with customer churn prediction, as overly complex models might not generalize well to new customers.
4) Feature Importance Analysis: Random Forests provide insights into feature importance. Understanding which features contribute the most to predicting customer churn can guide strategic decisions for retaining customers or improving service.

<u>Limitations:</u>

1) Overfitting on Noisy Data: Telco dataset contains a significant amount of noise or irrelevant features, Random Forest may still capture and amplify these patterns, leading to overfitting. In such cases, models with higher regularization or feature selection techniques might be more appropriate.
2) Nonlinear Decision Boundaries: Since the relationship between features and customer churn is highly nonlinear and cannot be well-captured by decision trees, more sophisticated models, such as gradient boosting algorithms or neural networks, might be better suited to the task.
3) Algorithm Diversity: Random Forests are an ensemble of similar models of decision trees. In this case underlying patterns are not well captured by decision trees.
4) Non-IID Data: Since the Telco data exhibits non-independent and identically distributed (non-IID) patterns, such as temporal dependencies, Random Forests might not effectively capture these dynamics.

Results:

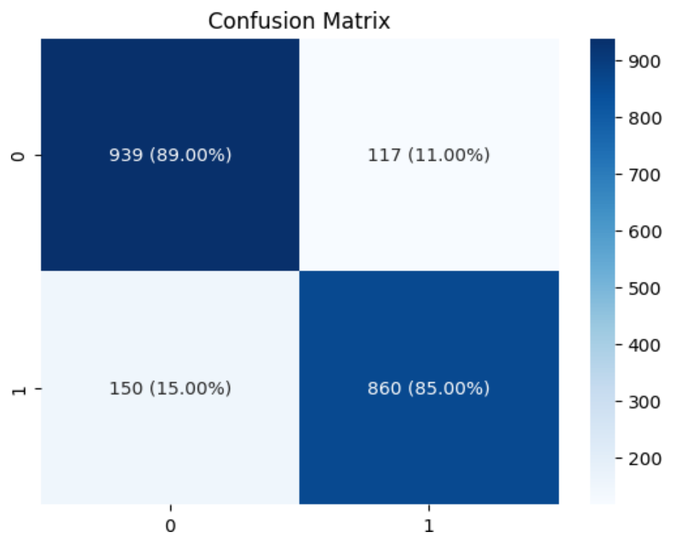
Values might vary for each run, below are the closest values based on multiple iterations of model fittings:

Metric	RandomForestClassifier
RMSE	0.36
ROC AUC	87%
Model accuracy	87%
Mean Absolute Error (MAE)	0.127

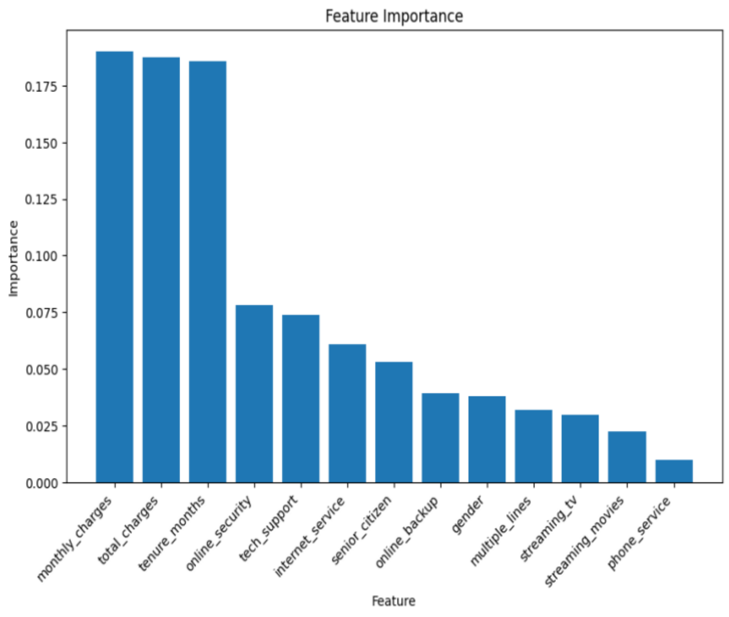
Classification Report:

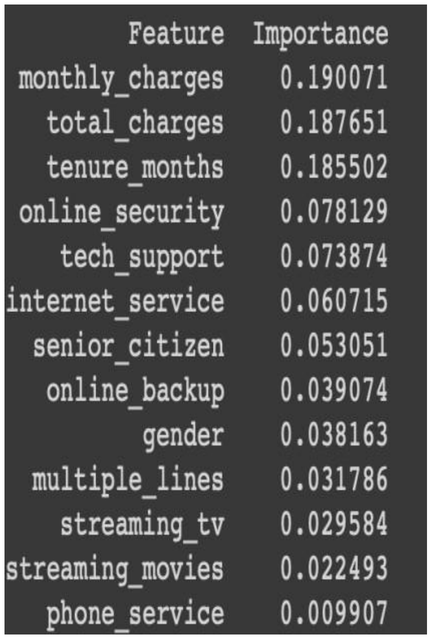
Class Label	Precision	Recall	F1-Score	Support
0	0.86	0.89	0.88	1056
1	0.88	0.85	0.87	1010

Confusion matrix:



<u>Feature Importance :</u>





```
         Feature  Importance
 monthly_charges    0.190071
   total_charges    0.187651
   tenure_months    0.185502
 online_security    0.078129
    tech_support    0.073874
internet_service    0.060715
  senior_citizen    0.053051
   online_backup    0.039074
          gender    0.038163
  multiple_lines    0.031786
     streaming_tv    0.029584
streaming_movies    0.022493
   phone_service    0.009907
```

<u>Key Takeaways</u>

We have learnt about the important features impacting churn in this dataset as a result of this procedure which are monthly_charges, total_charges and tenure_months for this model. Random Forest models frequently provide insights on feature relevance. This helps in assisting to determine which variables are most important in forecasting customer attrition.

6.3 DecisionTreeClassifier:

A DecisionTreeClassifier is a machine learning algorithm used for classification tasks. It creates a tree-like structure where each node represents a decision based on input features, leading to the classification of data into different classes. During training, the algorithm learns to make decisions by selecting the best features and thresholds to split the data into class labels. This tree structure is used for making predictions on new data points by following the decision path down the tree to reach a leaf node, which represents the predicted class.

Key Considerations:

1) Interpretability: Decision trees are inherently interpretable. The tree structure allows you to trace the path a customer takes to reach a churn prediction, making it easier to understand why a particular customer is predicted to churn. This interpretability can be valuable for businesses and stakeholders who want to know the factors influencing churn.
2) Feature Importance: Decision trees can provide information about feature importance, helping you identify which customer attributes (e.g., call duration, contract length, usage patterns) have the most significant impact on churn. This knowledge can guide targeted marketing or retention efforts.
3) Scalability: Since our telco dataset is moderately sized, decisionTreeClassifiers can provide competitive performance. They are computationally efficient and can handle thousands to tens of thousands of samples without significant issues.

Limitations:

1) Limited Predictive Power: Decision trees have their limits in terms of predictive accuracy. Data can be intricate and influenced by various interconnected factors, and a simple decision tree might not be able to capture all these nuances.
2) Nonlinear Relationships: While decision trees can capture some nonlinear relationships, they may not be as efficient as other models like Support Vector Machines or deep learning models in handling highly nonlinear patterns in the data.
3) Feature Engineering: Decision trees do not inherently perform feature engineering, meaning they don't create new features or representations of the data. In cases where feature engineering is critical to capturing churn patterns, other models might be better suited.

Results:

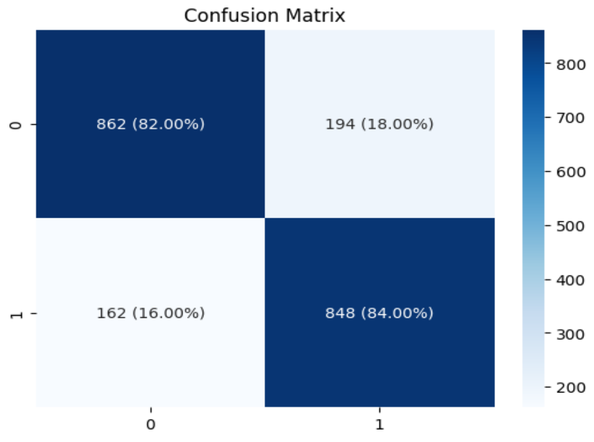
Values might vary for each run, below are the closest values based on multiple iterations of model fittings:

Metric	DecisionTreeClassifier
RMSE	0.42
ROC AUC	82%
Model accuracy	82%
Mean Absolute Error (MAE)	0.17

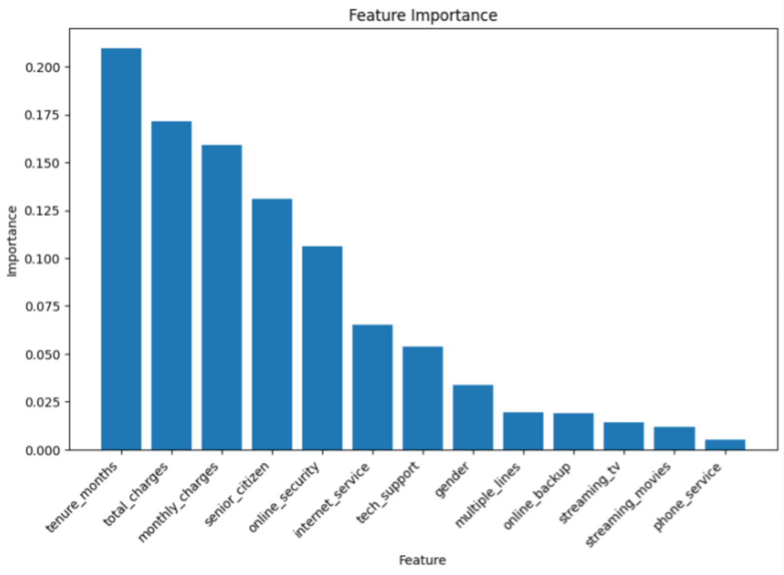
Classification Report:

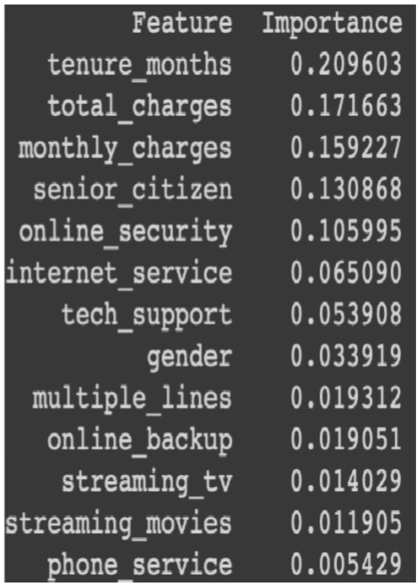
Class Label	Precision	Recall	F1-Score	Support
0	0.84	0.82	0.83	1056
1	0.81	0.84	0.83	1010

Confusion matrix:



Feature Importance:





```
         Feature  Importance
   tenure_months    0.209603
   total_charges    0.171663
 monthly_charges    0.159227
  senior_citizen    0.130868
 online_security    0.105995
internet_service    0.065090
    tech_support    0.053908
          gender    0.033919
  multiple_lines    0.019312
   online_backup    0.019051
    streaming_tv    0.014029
streaming_movies    0.011905
   phone_service    0.005429
```

Key Takeaways

Through this process, we learnt about the key features influencing churn which are tenure_months, total_charges and monthly_charges which inturn helps in gaining insights into the decision logic of the model.

6.4 Gaussian Naive Bayes Classifier:

The Gaussian Naive Bayes Classifier is a machine learning model designed for classification tasks, employing Bayes' theorem and the assumption of feature independence to calculate the probability of a data point belonging to a particular class. This classifier estimates the likelihood of observing specific feature values within each class based on the Gaussian distribution's mean and standard deviation. It then uses this information to calculate the posterior probability of each class for a given data point and selects the class with the highest probability as the prediction.

Key Considerations:

1) Continuous Features: Since our telco data includes continuous features (e.g Monthly charges, Tenure months, CLTV) that are assumed to follow a Gaussian normal distribution, the Gaussian Naive Bayes Classifier is a suitable choice. It can handle such continuous data effectively and efficiently.
2) Speed and Efficiency: Gaussian Naive Bayes is computationally efficient and typically faster to train and make predictions compared to more complex models. This can be an advantage when dealing with large datasets, and you need quick results.
3) Simple Model: If you're looking for a straightforward model that can serve as a baseline or starting point for churn prediction, Gaussian Naive Bayes is a good candidate. It doesn't require extensive hyperparameter tuning and can be a simple yet effective solution for initial analysis.

Limitations:

1) Feature Independence Assumption: The core assumption of the Naive Bayes algorithm is that features are conditionally independent given the class label. This assumption is not hold in telco data, where customer behaviors and attributes are often interrelated. For example, a customer's decision to churn may be influenced by their tenure, monthly charges, and multiple lines, which are not independent. GNB's strict independence assumption can lead to a lack of accuracy when modeling such dependencies.

2) Complex Relationships: Customer churn prediction often involves capturing complex, nonlinear relationships, and interactions between various factors, including service usage, contract length, and customer demographics. GNB, with its simplistic model structure, may struggle to capture these intricate patterns effectively.
3) Non-Normal Distributions: While GNB assumes that features follow a Gaussian (normal) distribution, many features in telco data may not have a normal distribution. For instance, monthly charges may exhibit a skewed or non-Gaussian distribution. Using GNB on non-normally distributed data can lead to suboptimal results, as it may not accurately model the underlying data distribution.

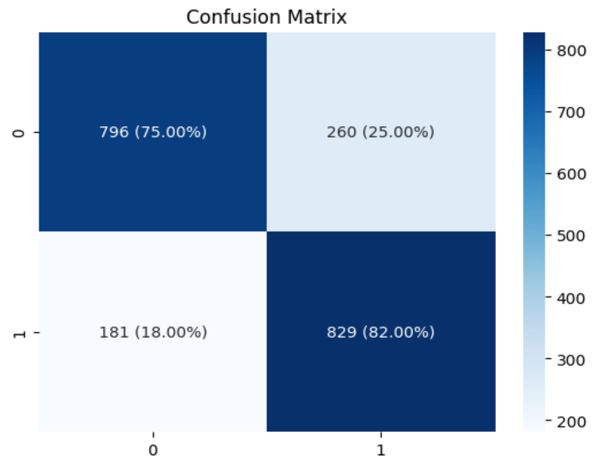
Results:

Metric	GaussianNBClassifier
RMSE	0.462
ROC AUC	78.73%
Model accuracy	78.65%
Mean Absolute Error (MAE)	0.213

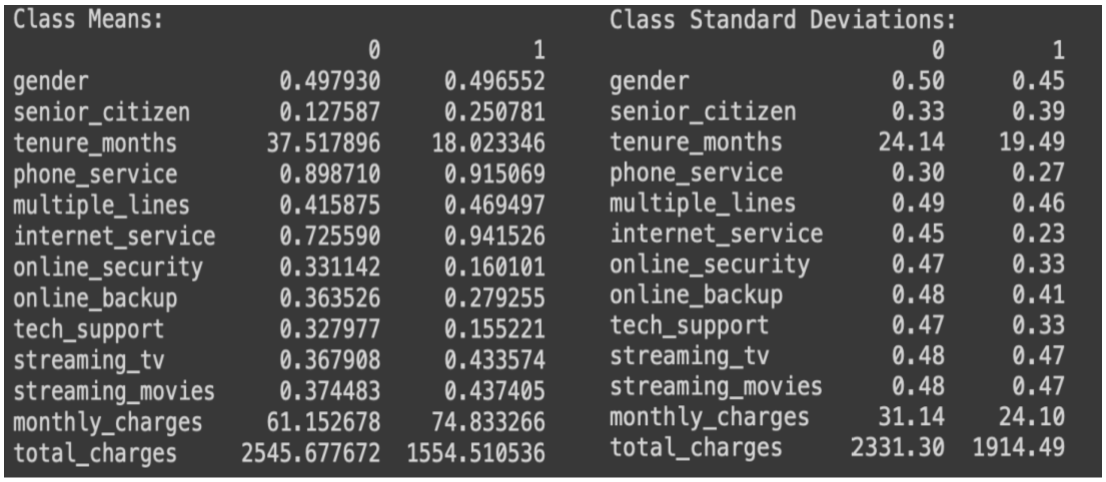
Classification Report:

Class Label	Precision	Recall	F1-Score	Support
0	0.81	0.75	0.78	1056
1	0.76	0.82	0.79	1010

Confusion matrix:



Class Means and Deviations for all features:



```
Class Means:                                  Class Standard Deviations:
                         0            1                                0        1
gender            0.497930     0.496552      gender                 0.50     0.45
senior_citizen    0.127587     0.250781      senior_citizen         0.33     0.39
tenure_months    37.517896    18.023346      tenure_months         24.14    19.49
phone_service     0.898710     0.915069      phone_service          0.30     0.27
multiple_lines    0.415875     0.469497      multiple_lines         0.49     0.46
internet_service  0.725590     0.941526      internet_service       0.45     0.23
online_security   0.331142     0.160101      online_security        0.47     0.33
online_backup     0.363526     0.279255      online_backup          0.48     0.41
tech_support      0.327977     0.155221      tech_support           0.47     0.33
streaming_tv      0.367908     0.433574      streaming_tv           0.48     0.47
streaming_movies  0.374483     0.437405      streaming_movies       0.48     0.47
monthly_charges  61.152678    74.833266      monthly_charges       31.14    24.10
total_charges  2545.677672  1554.510536      total_charges       2331.30  1914.49
```

Key Takeaways:

From the above images we can say that features such as tenure_months, phone_service and multiple_lines are useful in predicting churn(in this case these customers are less likely to churn whereas in other models we found features that are used to find to features which makes the customers more likely to churn).

6.5 <u>Logistic Regression:</u>

<u>Key Considerations:</u>

1. Simplicity and interpretability: It is easier to interpret and understand the factors that influence customer churn. This interpretability can be useful for pinpointing the main causes of churn and helping decision-makers lower churn rates.
2. Efficiency: Telco dataset can be handled by logistic regression with adequate training times and processing efficiency.
3. Robustness to outliers: Logistic regression is relatively robust to outliers and can handle noisy data without significant degradation in performance. This robustness is beneficial for customer churn data, which contain outliers or data points that deviate from the overall distribution.
4. Handling binary data: Logistic regression is specifically designed for binary classification tasks, making it well-suited for customer churn prediction, where the goal is to classify customers into two categories: churned and non-churned users.
5. Handling categorical features: Logistic regression can handle categorical features effectively by using one-hot encoding or other techniques to convert categorical values into numerical representations. This flexibility makes it suitable for analyzing various types of customer data.

<u>Limitations</u>:

1) Linearity assumption: Logistic regression assumes a linear relationship between the independent variables and the dependent variable. This assumption limits the model to not capture true underlying patterns, as nonlinear patterns may exist among features in data.
2) Feature selection: Logistic regression is sensitive to the selection of features, as irrelevant or redundant features can negatively impact model performance.

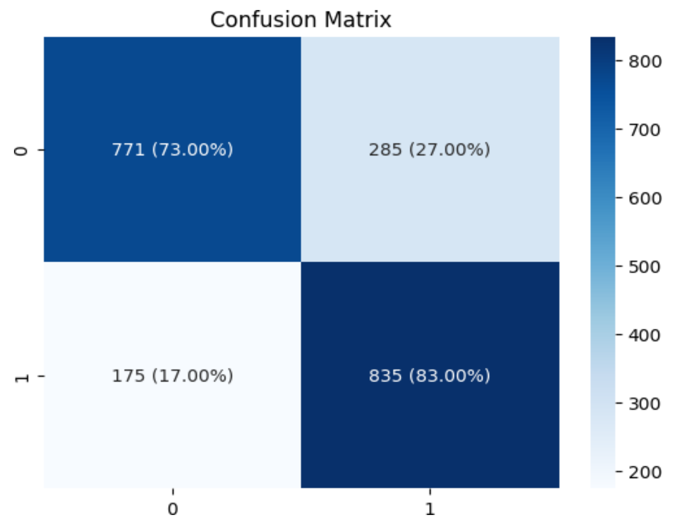
<u>Results</u>:

Metric	Logistic Regression
RMSE	0.47
ROC AUC	77.84%
Model accuracy	77.73%
Mean Absolute Error (MAE)	0.222

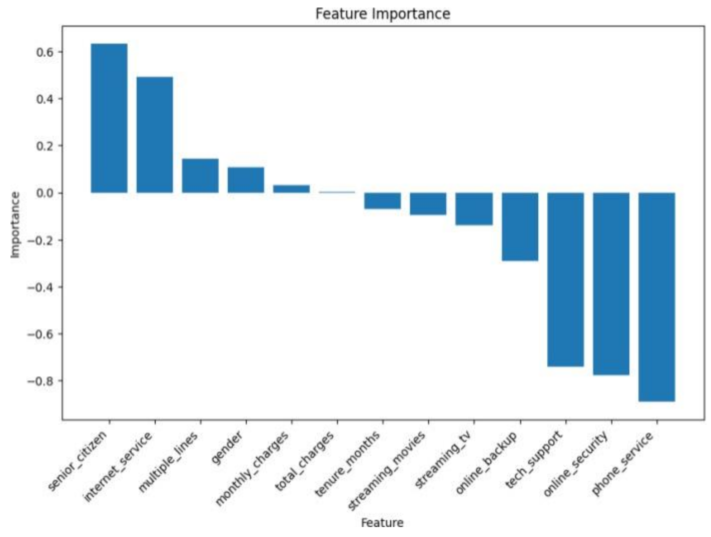
Classification Report:

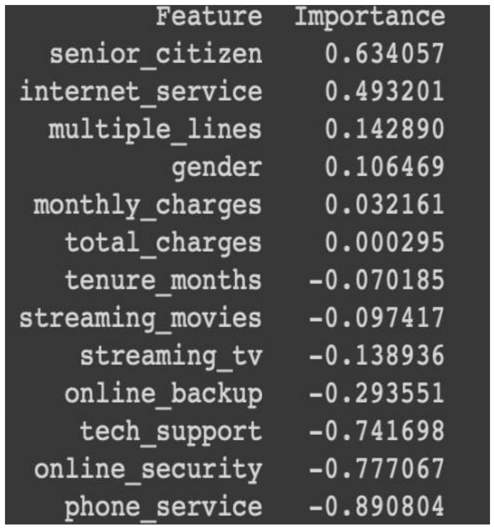
Class Label	Precision	Recall	F1-Score	Support
0	0.82	0.73	0.77	1056
1	0.75	0.83	0.78	1010

Confusion matrix:



Feature Importance:





```
         Feature  Importance
  senior_citizen    0.634057
internet_service    0.493201
  multiple_lines    0.142890
          gender    0.106469
 monthly_charges    0.032161
   total_charges    0.000295
   tenure_months   -0.070185
streaming_movies   -0.097417
    streaming_tv   -0.138936
   online_backup   -0.293551
    tech_support   -0.741698
 online_security   -0.777067
   phone_service   -0.890804
```

Key Takeaways:

The understanding of logistic regression is slightly different from models like Random Forest or XGBoost. In this modeling feature importance is typically determined by examining the coefficients of the model.
In Logistic Regression, the magnitude of the coefficients reflects the strength and direction of the association between each feature and the log-odds of the target variable (churn in this case). Larger absolute values indicate a greater impact on the prediction. A positive coefficient indicates that when the related feature increases, so do the log-odds of the target variable, contributing to the chance of churn. A negative coefficient, on the other hand, indicates that as the characteristic rises, the log-odds fall, contributing to the risk of churn. For this dataset senior_citizen, internet_service and multiple_lines are the features that have the largest positive coefficients suggesting that they are the major contributors to churn.

6.6. SVM Classifier:

Key Considerations:

1. Effective handling of non-linear data: When compared to linear models, SVMs can more accurately predict outcomes because they are better at capturing these non-linear relationships.
2. Robustness to outliers: SVMs are less sensitive to outliers compared to other models, making them more robust to noisy data.
3. High-dimensional data handling: Customer churn data often involves a large number of features, including demographics, usage patterns, and billing information. SVMs can handle high-dimensional data efficiently, making them suitable for real-world churn prediction scenarios.
4. Generalization ability: SVMs aim to find the optimal hyperplane that separates the data into classes, maximizing the margin between the hyperplane and the data points. This focus on finding the optimal hyperplane leads to better generalization ability, allowing the model to perform well on unseen data.
5. Reduced risk of overfitting: SVMs employ regularization techniques to prevent overfitting, which occurs when a model learns the training data too well but fails to generalize to new data. Regularization helps to balance the complexity of the model and its ability to generalize.

Limitations:

1) Computational complexity: In particular, SVMs can be computationally costly for large datasets including high-dimensional features. The training procedure can take a long time and requires significant computational resources.
2) Hyperparameters tuning: To maximize performance, SVMs require tuning of multiple hyperparameters. Choosing the right hyperparameters can be difficult and frequently necessitates experimenting with various values.
3) Memory usage: SVMs can use up a lot of memory, particularly when working with large data sets. This might be a drawback in settings with limited resources.

Results:

Metric	SVM-Linear	SVM - RBF
RMSE	0.48	0.43
ROC AUC	76.79%	81.63%
Model accuracy	76.82%	81.80%
Mean Absolute Error (MAE)	0.231	0.182

Classification Report:

SVM-Linear

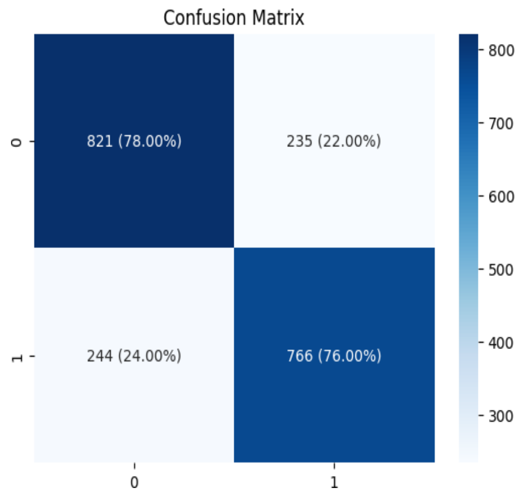
Class Label	Precision	Recall	F1-Score	Support
0	0.77	0.78	0.77	1056
1	0.77	0.76	0.76	1010

SVM-RBF

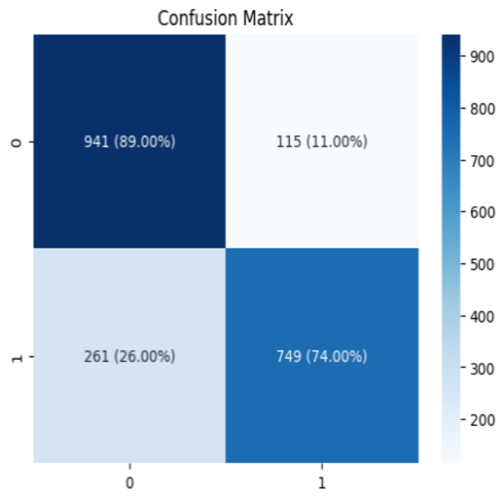
Class Label	Precision	Recall	F1-Score	Support
0	0.78	0.89	0.83	1056
1	0.87	0.74	0.80	1010

Confusion Matrix:

SVM Linear:



SVM RBF:



Key Takeaways

- For the application of SVM for customer churn prediction, choice between a linear and Gaussian kernel significantly impacts model performance, with the complexity of the data and feature relationships influencing the selection.
- Effective feature selection and engineering are essential for boosting model accuracy, requiring an iterative process that combines domain knowledge and experimentation. Addressing class imbalance is crucial to ensure the model's predictive power isn't skewed by the disproportionate number of non-churning customers.
- Hyperparameter tuning is an essential for SVM models for optimal performance, and grid search or random search can assist in finding the best hyperparameter combinations.
- Lastly, it's important to consider the balance between predictive accuracy and model interpretability, with SVM models often being less interpretable than simpler models, such as logistic regression.

7. INFERENCES:

Compiled Result:

Metric	XGB	RF	DT	GNB	LR	SVM-L	SVM - RBF
RMSE	0.356	0.36	0.42	0.462	0.47	0.48	0.43
ROC AUC	87.29%	87%	82%	78.73%	77.84%	76.79%	81.63%
Model accuracy	87.32%	87%	82%	78.65%	77.73%	76.82%	81.80%
Mean Absolute Error (MAE)	0.127	0.127	0.17	0.213	0.222	0.231	0.182

All the above 6 algorithms have good accuracies which is above 75%, RMSE values are relatively less for Random Forest and XGB Classifiers.

It is observed from the above analysis that the feature importance of the models XGBClassifier and Logistic regression are similar and Random Forest Classifier and Decision tree Classifier are similar and are not exactly same. This difference can be explained by following reasons:

1) Algorithm Basis:
- XGB and Logistic are linear models and the feature importance is directly determined by coefficient. In cases where there is a nearly linear relationship between the features and the target, or if a linear model can adequately describe the relationship, both these models can assign similar importance for feature
- Decision trees and Random Forest assign feature importance by information gain or impurity reduction. These models account for the non-linear relationships among features and target.

2) Overfitting:
- XGB and Logistic Regression have regularization parameters that can control overfitting.
- Decision trees if deep are prone to overfitting and random forest mitigates this by averaging over multiple trees.

3) Model type (Ensemble, Single):
- Logistic regression is a single method
- XGBoost is an ensemble method where base learner is shallow tree

- Random Forest is an ensemble method with multiple independent trees making the effect stronger
- Decision tree is not an ensemble method but can capture complex relationships in a single tree.

One thing we should remember is that the model's success is determined by the quality of the data, feature engineering, and the suitability of the method employed. It's also critical to think about the business environment and whether the model's predictions correspond to practical recommendations for minimizing churn.

Likelihood of Customer Churning:

XGB (XGBoost) and RF (Random Forest) show the highest ROC AUC scores, indicating that these models are the most effective at predicting customer churn likelihood. These models achieve accuracy levels above 87%, suggesting they are reliable for estimating the likelihood of churn.

Spotting High-Risk Customers:

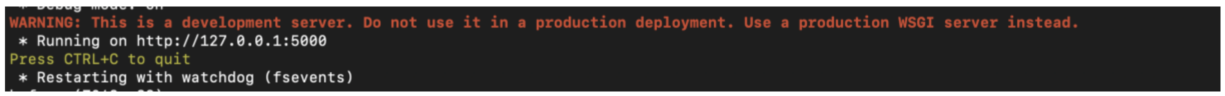
XGB and RF are the models most capable of effectively distinguishing high-risk customers. Their higher accuracy rates and ROC AUC values suggest they excel at separating churn and non-churn instances.

8. <u>Web Application:</u>

We've developed a web application based on flask that allows users to upload their data in a specific format. Once uploaded, the system processes the data to generate an output for each customer ID, assigning a binary value of either 1 or 0. This binary value indicates whether the system predicts the customer as likely to churn (1) or not to churn (0). Users can view this churn prediction outcome for their respective customer IDs within the application as an end result.

A guide to use the application:

1. The folder containing the dataset should be made the current working directory and then run the code.
2. Execute the following command to run the Flask application:
   python app.py
3. After running the script, you should see output in the terminal indicating that the Flask app is running. By default, it will be accessible at http://127.0.0.1:5000/ or http://localhost:5000/.
4. Open your web browser and go to http://127.0.0.1:5000/start or http://localhost:5000/start.



```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (fsevents)
```

5. On the web page, you'll likely find a form or a section to upload a file. Follow the instructions on the web page to upload a data file (possibly in Excel format). Input file format can be seen in link attached at end of page.
6. After uploading the data, the web application will process it, make predictions using the XGBoost model, and display the results on the webpage.
7. Explore the different routes provided by the Flask app, such as /start, /inp_data, and /input_data_format.
8. Step-8: To stop the Flask application, you can typically press Ctrl + C in the terminal where it's running.

Note: Companies often collect and organize data in various formats. The provided code is tailored to the specific format used in the Telco Churn dataset. While the code can be adapted for different data formats with minor adjustments, for a seamless user experience, we recommend users to upload data in a specific format. This ensures smooth execution and accurate predictions without delving too deeply into the intricacies of the code.

Using the generated results, the company gains insights into the percentage of customers who are predicted to churn and those expected to remain. This information helps the user understand the likelihood of customer churn within their customer base.

The following steps can be suggested as the preventive measures in order to retain the customer base who are about to churn:

- Offering personalized retention offers, discounts, or incentives to customers identified as high-risk by XGB algorithm.

- Leveraging insights from these models to understand the most influential factors driving churn, such as historical behavior and usage trends.
- Implementing proactive customer support measures to address high-risk customers' concerns and issues promptly.
- Optimizing marketing efforts based on the predictions from XGB model to reach and retain at-risk customers.
- The models with higher ROC AUC and accuracy scores (XGB and RF) can help guide the design and execution of these strategies by providing more reliable churn predictions.

9. References

1) XGBoost: A Scalable and Accurate Implementation of Gradient Boosting by T. Chen and C. Guestrin
2) Introduction to Random Forest by Will Koehrsen
3) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron
4) Data Science from Scratch- First Principles with Python by Joel Grus