Capstone Project 2

Project Report

on

Customer Churn Prediction in Telecom Sector

Submitted towards the partial fulfilment of the criteria for award of Post Graduate Program in Data Science and Analytics by Imarticus

Submitted by

Vinayak Patnaik.

Table of Contents

Abstract

In the rapidly evolving telecom industry, customer retention emerges as a pivotal challenge due to high churn rates. This project delineates the development and deployment of an innovative predictive model aimed at identifying potential customer churn. Utilizing a rich dataset comprising historical customer usage data, billing information, service tickets, and customer feedback, the study leverages advanced machine learning algorithms, including traditional techniques, ensemble methods, and artificial neural networks. The model's predictive capabilities enable telecom companies to devise targeted retention strategies effectively, thereby curtailing churn rates and bolstering customer loyalty. The comprehensive methodology spans data collection, preprocessing, exploratory data analysis, and rigorous model evaluation, culminating in the deployment of the most efficacious model. This research not only underscores the feasibility of predictive analytics in enhancing customer retention strategies but also sets a precedent for leveraging data-driven insights to foster informed decision-making in the telecom sector.

Acknowledgements

I extend my profound gratitude to the faculty and administration at Imarticus Learning, Mumbai, for their invaluable guidance and support throughout the duration of the Post Graduate Program in Data Science and Analytics. Special thanks to my project supervisor, whose expertise and insights were instrumental in navigating the complexities of this project. I am also indebted to my peers for their constructive feedback and encouragement, fostering an enriching learning environment.

I am grateful to the open-source community for making available the datasets and tools that enabled this research. This project stands as a testament to the collaborative effort of all those who shared their knowledge and resources, for which I am genuinely thankful.

Lastly, I certify that the work done by me for conceptualizing and completing this project is original and authentic.

Date: March, 2023 Vinayak Patnaik

Place: Mumbai

Certificate of Completion

I hereby certify that the project titled “Customer Churn Prediction in Telecom Sector” was undertaken and completed under my supervision from the batch of PGD-23 (March 2023)

Date: June 28, 2019

Place – Mumbai

1. Introduction

1.1 Executive Summary:

This project aims to develop a robust predictive model for identifying potential customer churn in the telecom industry. By leveraging machine learning algorithms and advanced analytical techniques, we will analyse customer data to predict which customers are most likely to churn. This predictive insight will enable telecom companies to implement targeted retention strategies, thereby reducing churn rates and increasing customer loyalty.

1.2 Problem Statement:

The telecom industry faces high customer churn rates due to competitive pricing, better service offerings by competitors, and customer dissatisfaction. This leads to increased costs related to acquiring new customers and lost revenue from existing customers.

1.3 Objective:

Develop a predictive model that can accurately identify customers at high risk of churn, allowing for proactive retention strategies.

1.4 Scope:

The project will encompass data cleaning, preprocessing, exploratory data analysis (EDA), application of traditional machine learning algorithms as well as ensemble techniques and artificial neural networks (ANN), and model deployment

1.5 Methodology:

1.5.1 Data Collection: Gather historical data of customer usage data, billing information, service tickets, customer feedback, etc to build a comprehensive dataset.

1.5.2 Data Preprocessing: Clean and preprocess the data to handle missing values, outliers, and encode categorical variables.

1.5.3 Exploratory Data Analysis (EDA): Conduct EDA to uncover patterns, trends, and correlations within the data that could influence customer churn.

1.5.4 Model Development: Apply traditional machine learning algorithms, ensemble techniques, and ANN to develop predictive models. Evaluate each model based on performance metrics to select the best approach.

1.5.5 Model Deployment: Deploy the best-performing model for real-time prediction of customer churn using Streamlit.

1.6 Data Sources:

The dataset used for this project is sourced from IBM. IBM provided customer data for Telco industry to predict behaviour of the customers. The dataset consists of 22 fields and 7043 records. The datasets contain information regarding the,

1.6.1 Demographic: gender, age range and if they have any partner or dependent.

1.6.2 Account type: How long they have been a customer, contract type, payment method, paperless billing, monthly charges.

1.6.3 Service: User has phone or internet service. Also, information related to multiline phone, streaming services and many more.

1.7 Tools and Technologies:

1.7.1 Python for data cleaning, preprocessing, model development, and deployment.

1.7.2 Libraries such as Pandas, NumPy for data manipulation, Scikit-learn for machine learning, and Keras for ANN.

1.7.3 Streamlit framework for deployment and Jupyter Notebook for interactive code execution and documentation.

1.8 Risks and Challenges:

Data privacy and security concerns when handling customer data. Ensuring the model remains accurate and relevant over time as customer behaviour and market conditions change. Integrating the predictive model into existing systems and workflows for real-time prediction and action.

- **What is Customer Churn?**

Customer churn is the number of customers that stopped using your company's product or service during a certain time frame. It can have a significant impact on a company's revenue and it's crucial for businesses to find out the reasons why customers are leaving and take steps to reduce the number of customers leaving.

- **Why is Customer Churn Important?**

Well, it's important because it costs more to acquire new customers than it does to retain existing customers. In fact, an increase in customer retention of just 5% can create at least a 25% increase in profit. This is because returning customers will likely spend 67% more on your company's products and services. As a result, your company can spend less on the operating costs of having to acquire new customers. You don't need to spend time and money on convincing an existing customer to select your company over competitors because they've already made that decision.

3. Data Preprocessing

Data preprocessing is the crucial step of understanding the data, identifying anomalies, and addressing them to prepare for further analysis and model building.

3.1. Understanding the data

3.1.1. Statistical and Descriptive analysis:

Using ‘info()’ and ‘describe()’ function from pandas library for statistical and descriptive analysis.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 22 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   LoyaltyID          7043 non-null   int64  
 1   Customer ID        7043 non-null   object 
 2   Gender             7043 non-null   object 
 3   Senior Citizen     7043 non-null   object 
 4   Partner            7043 non-null   object 
 5   Dependents         7043 non-null   object 
 6   Tenure             7043 non-null   int64  
 7   Phone Service      7043 non-null   object 
 8   Multiple Lines     7043 non-null   object 
 9   Internet Service   7043 non-null   object 
 10  Online Security    7043 non-null   object 
 11  Online Backup      7043 non-null   object 
 12  Device Protection  7043 non-null   object 
 13  Tech Support       7043 non-null   object 
 14  Streaming TV       7043 non-null   object 
 15  Streaming Movies   7043 non-null   object 
 16  Contract           7043 non-null   object 
 17  Paperless Billing  7043 non-null   object 
 18  Payment Method     7043 non-null   object 
 19  Monthly Charges    7043 non-null   float64
 20  Total Charges      7043 non-null   object 
 21  Churn              7043 non-null   object 
dtypes: float64(1), int64(2), object(19)
memory usage: 1.2+ MB
```

	LoyaltyID	Tenure	Monthly Charges
count	7043.000000	7043.000000	7043.000000
mean	550382.651001	32.371149	64.761692
std	260776.118690	24.559481	30.090047
min	100346.000000	0.000000	18.250000
25%	323604.500000	9.000000	35.500000
50%	548704.000000	29.000000	70.350000
75%	776869.000000	55.000000	89.850000
max	999912.000000	72.000000	118.750000

The dataset consists of 22 fields and 7043 records, including integer, float and lot of object data types. The 'objects' need to be converted into integers for further modelling. The 'Total Charges' is in object datatype, hence need for conversion to float.

3.1.2. Checking for cardinality/singularity and special characters.

```
# To check cardinality
for i in df.columns:
    print({i:df[i].nunique()})
```

```
[24]
{'LoyaltyID': 7021}
{'Customer ID': 7043}
{'Gender': 2}
{'Senior Citizen': 2}
{'Partner': 2}
{'Dependents': 2}
{'Tenure': 73}
{'Phone Service': 2}
{'Multiple Lines': 3}
{'Internet Service': 3}
{'Online Security': 3}
{'Online Backup': 3}
{'Device Protection': 3}
{'Tech Support': 3}
{'Streaming TV': 3}
{'Streaming Movies': 3}
{'Contract': 3}
{'Paperless Billing': 2}
{'Payment Method': 4}
{'Monthly Charges': 1585}
{'Total Charges': 6531}
{'Churn': 2}
```

```
# To check if there are any special characters in place of values
for i in df.columns:
    print({i:df[i].unique()})
```

```
{'LoyaltyID': array([318537, 152148, 326527, ..., 155157, 731782, 353947], dtype=int64)}
{'Customer ID': array(['7590-VHVEG', '5575-GNVDE', '3668-QPYBK', ..., '4801-JZAZL',
       '8361-LTMKD', '3186-AJIEK'], dtype=object)}
{'Gender': array(['Female', 'Male'], dtype=object)}
{'Senior Citizen': array(['No', 'Yes'], dtype=object)}
{'Partner': array(['Yes', 'No'], dtype=object)}
{'Dependents': array(['No', 'Yes'], dtype=object)}
{'Tenure': array([ 1, 34,  2, 45,  8, 22, 10, 28, 62, 13, 16, 58, 49, 25, 69, 52, 71,
       21, 12, 30, 47, 72, 17, 27,  5, 46, 11, 70, 63, 43, 15, 60, 18, 66,
        9,  3, 31, 50, 64, 56,  7, 42, 35, 48, 29, 65, 38, 68, 32, 55, 37,
       36, 41,  6,  4, 33, 67, 23, 57, 61, 14, 20, 53, 40, 59, 24, 44, 19,
       54, 51, 26,  0, 39], dtype=int64)}
{'Phone Service': array(['No', 'Yes'], dtype=object)}
{'Multiple Lines': array(['No phone service', 'No', 'Yes'], dtype=object)}
{'Internet Service': array(['DSL', 'Fiber optic', 'No'], dtype=object)}
{'Online Security': array(['No', 'Yes', 'No internet service'], dtype=object)}
{'Online Backup': array(['Yes', 'No', 'No internet service'], dtype=object)}
{'Device Protection': array(['No', 'Yes', 'No internet service'], dtype=object)}
{'Tech Support': array(['No', 'Yes', 'No internet service'], dtype=object)}
{'Streaming TV': array(['No', 'Yes', 'No internet service'], dtype=object)}
{'Streaming Movies': array(['No', 'Yes', 'No internet service'], dtype=object)}
{'Contract': array(['Month-to-month', 'One year', 'Two year'], dtype=object)}
{'Paperless Billing': array(['Yes', 'No'], dtype=object)}
{'Payment Method': array(['Electronic check', 'Mailed check', 'Bank transfer (automatic)',
       'Credit card (automatic)'], dtype=object)}
{'Monthly Charges': array([29.85, 56.95, 53.85, ..., 63.1 , 44.2 , 78.7 ])}
{'Total Charges': array([29.85, 1889.5, 108.15, ..., 346.45, 306.6, 6844.5], dtype=object)}
{'Churn': array(['No', 'Yes'], dtype=object)}
```

There is no case of singularity and cardinality in the features with object datatype except for ‘LoyaltyID’ and ‘CustomeID’. Also, there are no special characters present in our data.

3.1.3. Detecting missing values and duplicates.

```
df.duplicated().sum()
```

```
0
```

```
df.isnull().sum()
```

```
LoyaltyID           0
Customer ID         0
Gender              0
Senior Citizen      0
Partner             0
Dependents          0
Tenure              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               0
dtype: int64
```

The dataset is free of duplicate values but does contain few missing values

3.1.4. Data manipulating.

```
df['Total Charges'] = pd.to_numeric(df['Total Charges'], errors='coerce')
```

```
df['Total Charges'].fillna(round(df['Total Charges'].median(),2),inplace=True)
```

```
df.drop(['LoyaltyID','Customer ID'], axis=1, inplace=True)
df.shape
```

```
(7043, 20)
```

The 'Total Charges' was in object datatype, hence converted to float. And few missing values, it was imputed using 'fillna' by median aggregation. The features 'LoyaltyID' & 'Customer ID' are dropped as they do not provide any significant info in analysis and further model building.

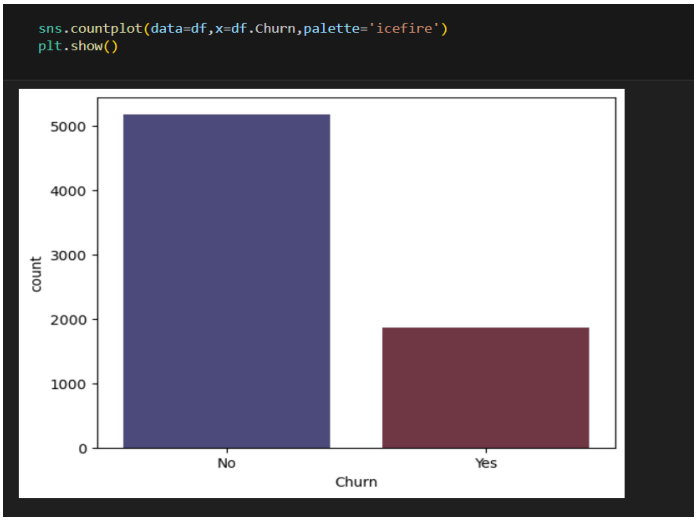
3.2. Data analysis using visualization

3.2.1. Univariate analysis.

```
churn_count = pd.DataFrame({'Count':df.Churn.value_counts()})
churn_count['PercentOfEachCOunt'] = round((churn_count['Count']/df.shape[0])*100,2)
churn_count
```

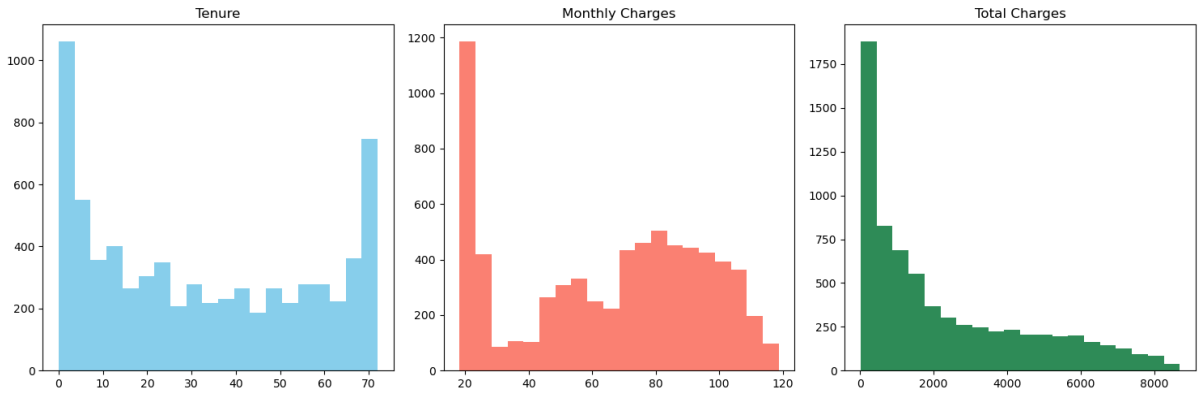
	Count	PercentOfEachCOunt
No	5174	73.46
Yes	1869	26.54

```
sns.countplot(data=df,x=df.Churn,palette='icefire')
plt.show()
```



The Count of churned customers is less accounting for 26.54%

Checking the distribution of numerical variable using Histogram and Boxplot



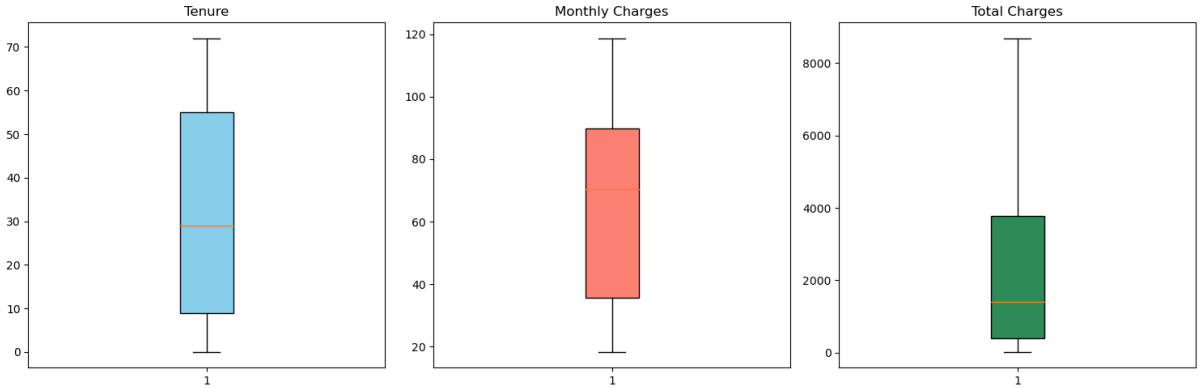

```
# check for skewness and kurtosis of each numerical variable.

for col in df_num:
    print(f"# {col}:")
    print("Skewness: %f" % df[col].skew())
    print("Kurtosis: %f" % df[col].kurt())
    print("----------------------")
```

[41]

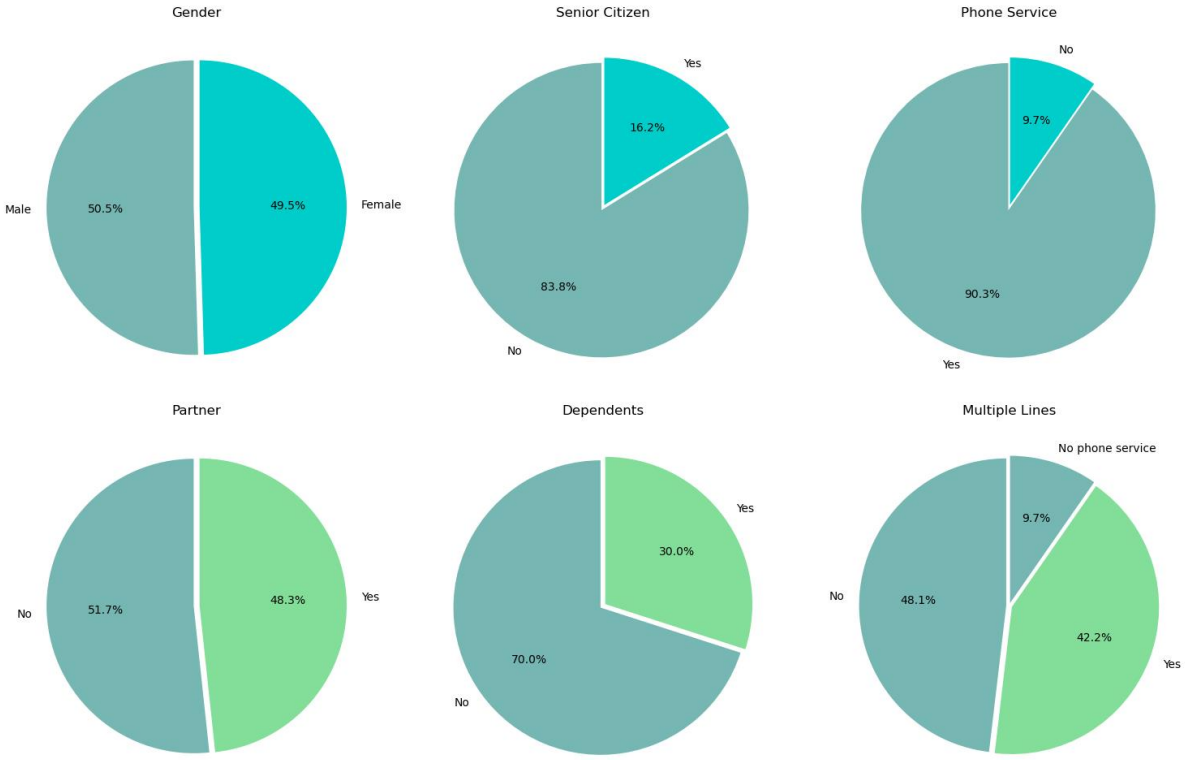
```
# Tenure:
Skewness: 0.239540
Kurtosis: -1.387372
----------------------
# Monthly Charges:
Skewness: -0.220524
Kurtosis: -1.257260
----------------------
# Total Charges:
Skewness: 0.963789
Kurtosis: -0.226400
----------------------
```

It seems like the numerical variables do not follow a normal distribution. Also, Monthly and Total charges exhibit left and right skewness resp.



No outliers are present.

Distribution of categorical variables



Inferences for above charts:

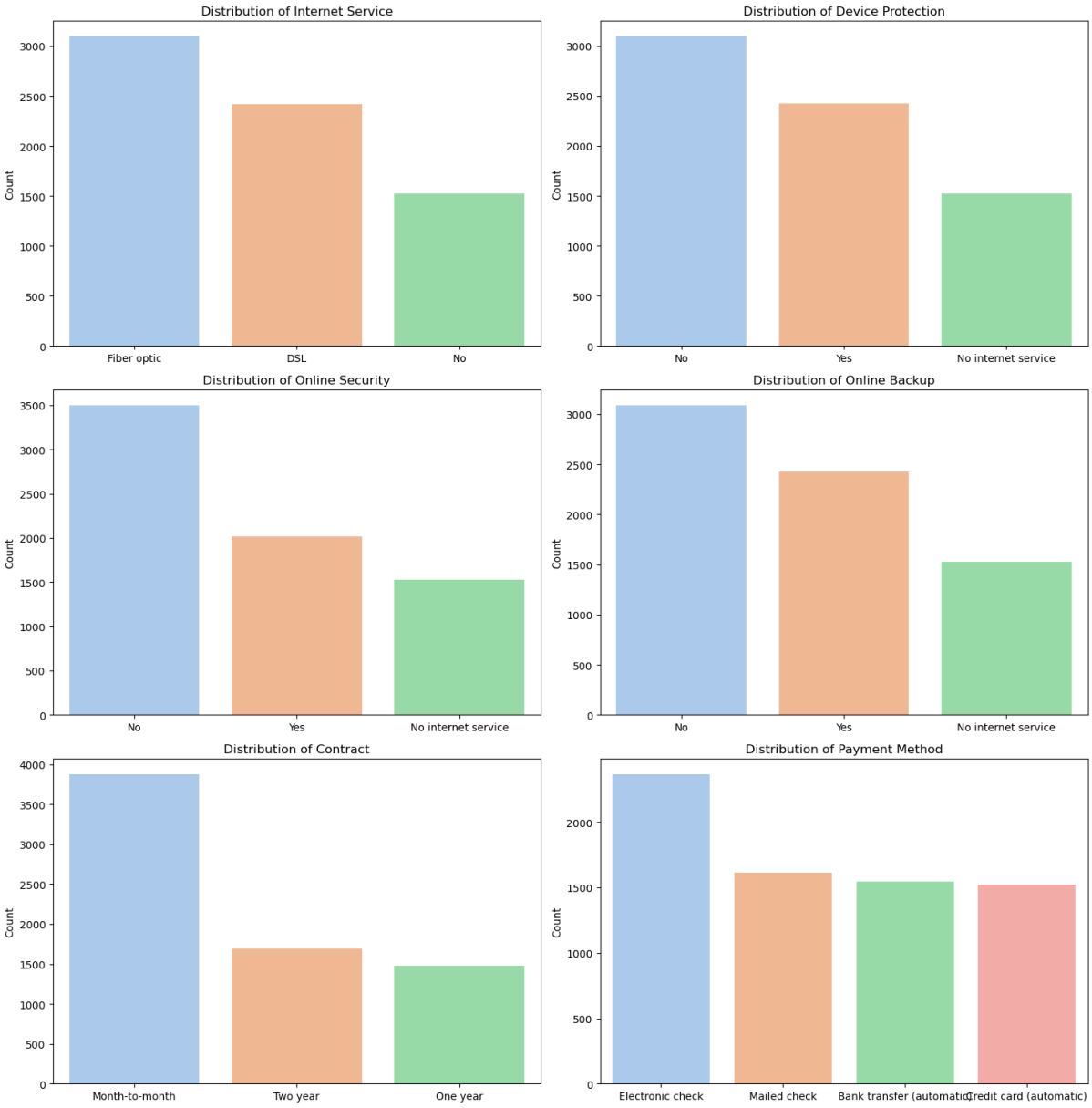
1. We can observe that the distribution between genders and the presence of a partner is almost equal, with males and those without partners slightly edging out the other categories.

2. Additionally, only 16% of the data includes senior citizens.

3. Furthermore, 90% of customers have availed themselves of phone service.

4. Approximately 70% of the customers do not have dependents.

5. Lastly, around half of the customers do not have multiple lines, and only 42% do.

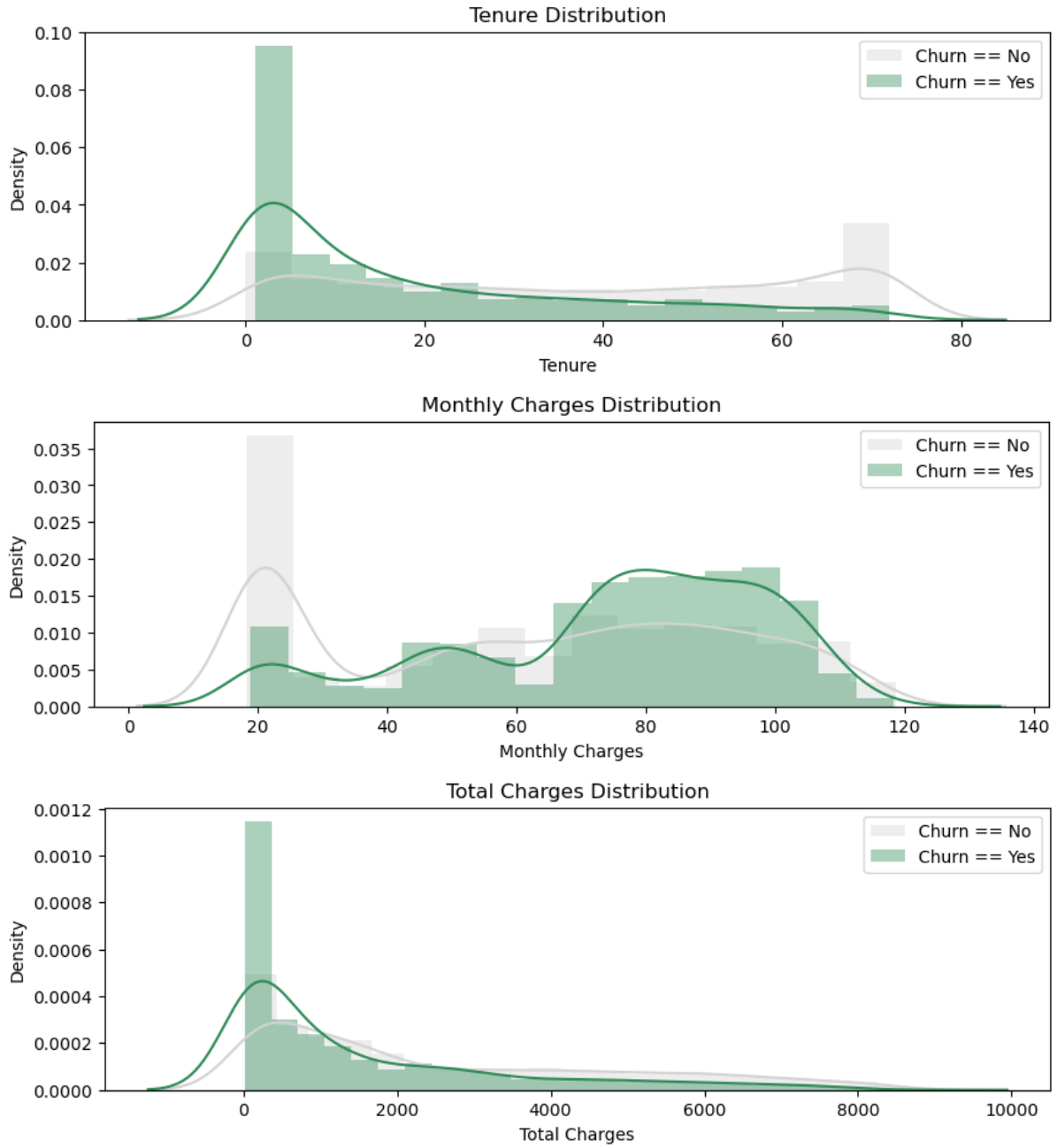


Inferences for above charts:

1. Most customers have chosen fiber optics, followed by DSL, for their service.
2. Most customers have chosen not to opt for device protection.
3. For online security and backup, most have again chosen not to opt in.
4. The month-to-month service contract is the most popular.
5. Lastly, customers primarily pay via electronic check.

3.2.2. Bivariate analysis

Distribution of numerical variables by target variable.

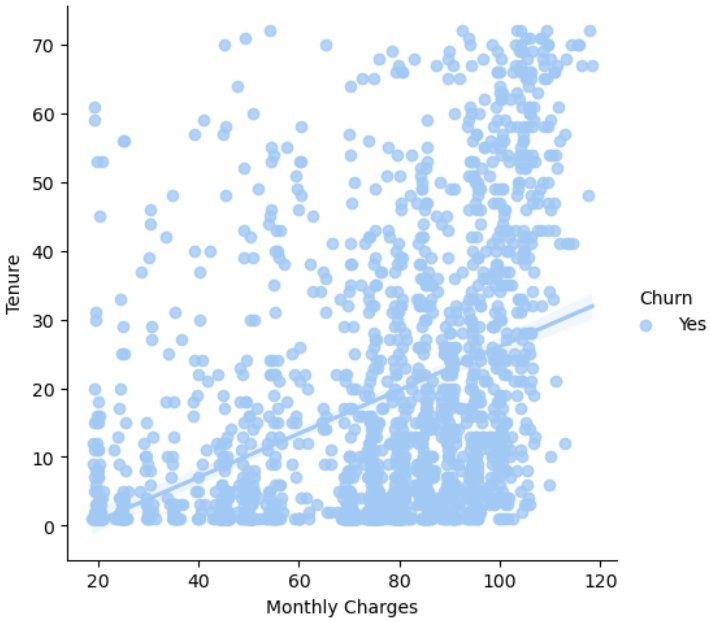


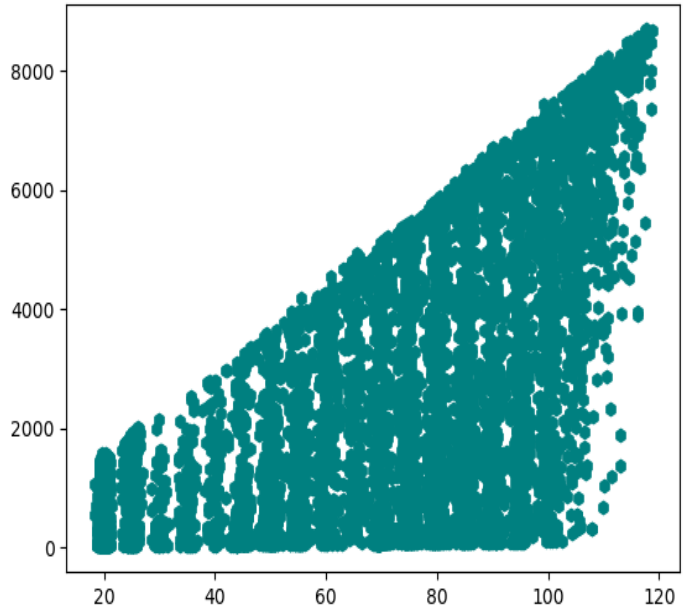
1. We can clearly infer those customers with low tenure, such as 1 to 5 months, tend to churn more frequently.

2. Customers facing monthly charges in the range of 70 to 110 are more likely to churn, whereas those with lower monthly charges tend to remain with the service.

3. For total charges, the likelihood of customers churning is higher with charges ranging from 0 to 500

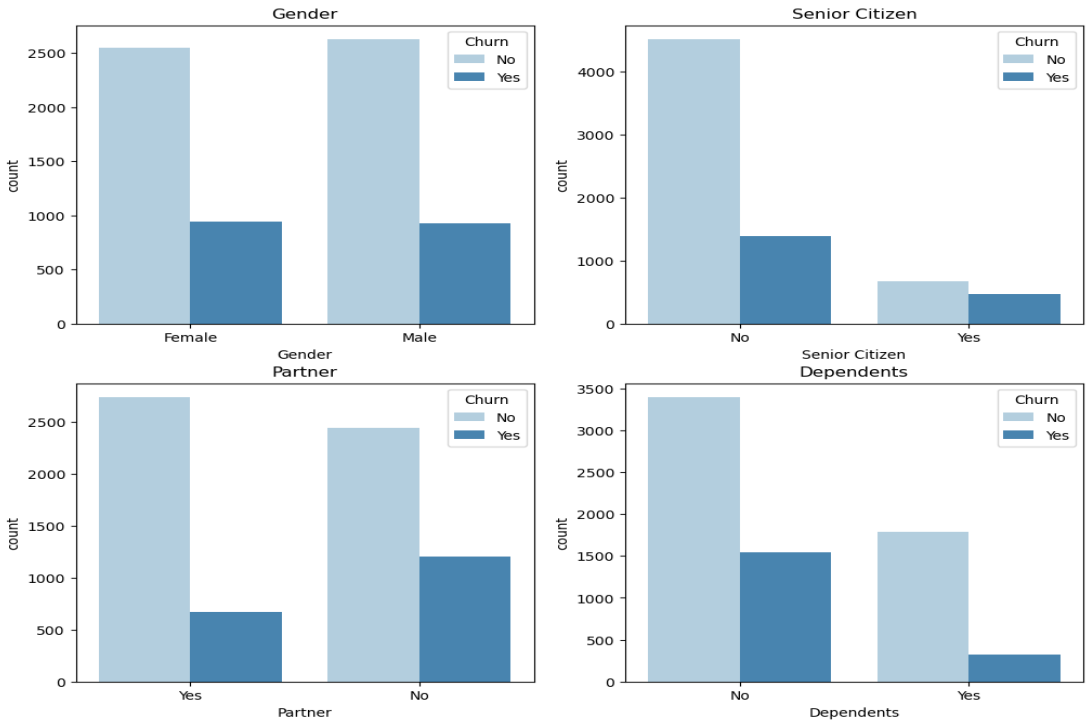
Relationship between numerical variable using scatter plot





As we can see that the most churned customers are concentrated at right bottom where the tenure is low but monthly charges are high. Higher the monthly chargers higher is the total charges, as expected.

Distribution of Categorical variable by Target variable

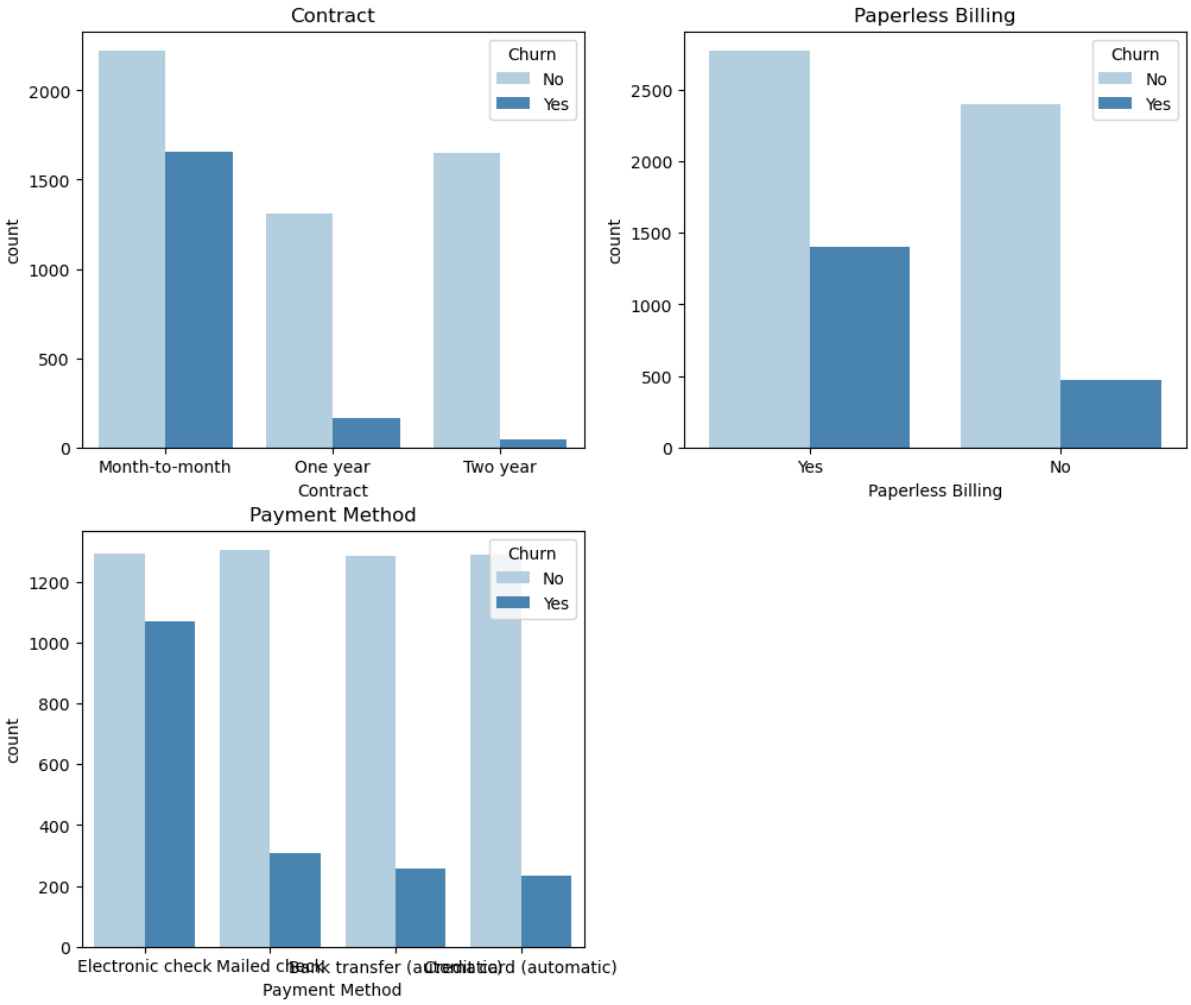


1. For Gender, we observe an equal distribution between churn and non-churn.

2. Although Senior Citizens make up a smaller portion of the customer base, they have a higher likelihood of churn compared to non-Senior Citizens.

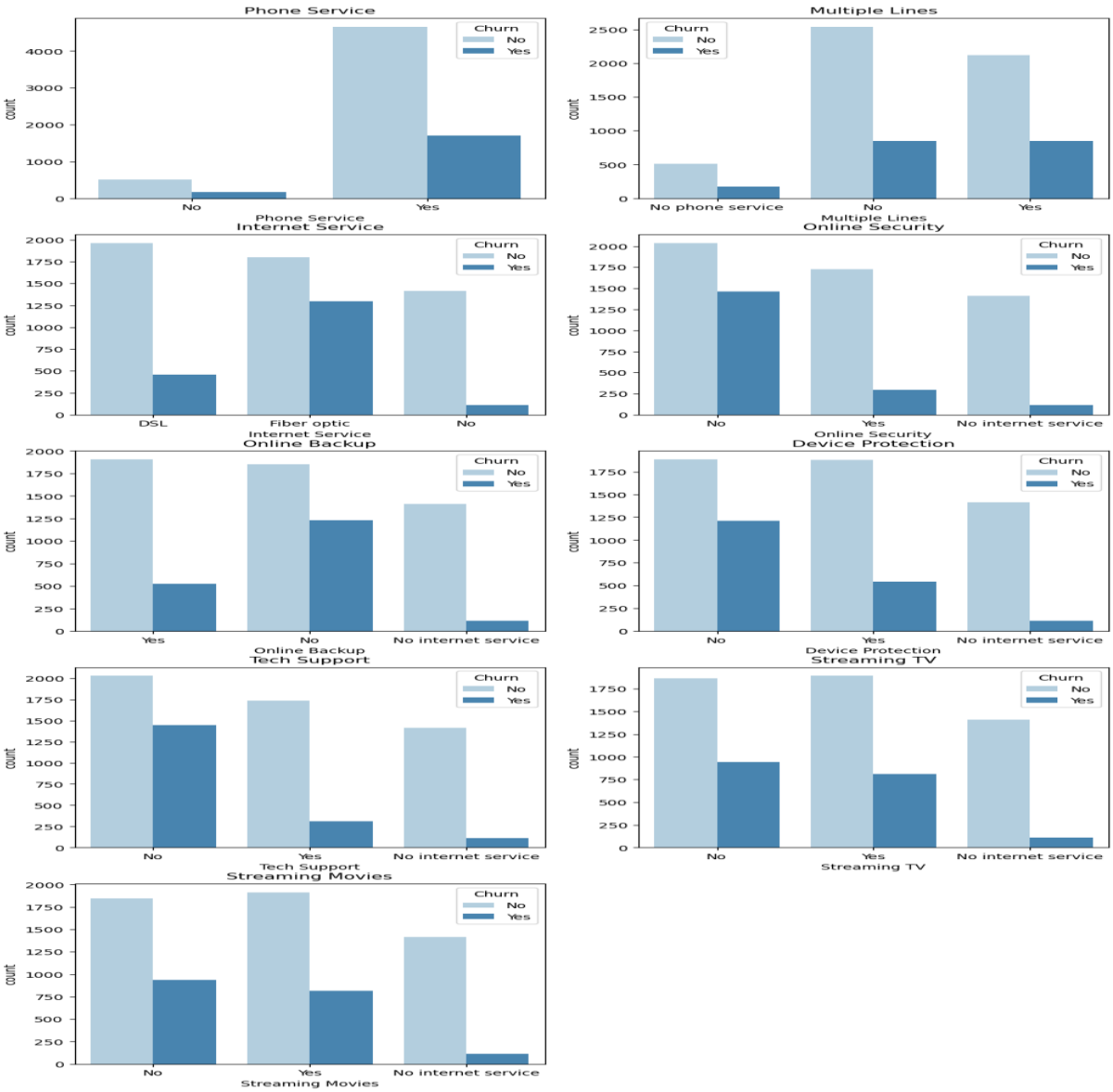
3. Furthermore, customers without partners are more likely to churn.

4. Lastly, customers with dependents are less likely to churn compared to those without dependents.



1. It appears that most customers prefer month-to-month contracts and pay via electronic check, yet these sub-categories also exhibit high churn rates, with nearly half of the customers churning.

2. Customers who opt for paperless billing experience higher churn rates.



1. Customers opting for phone services have a moderately high chance of churning.

2. Also, customers, whether opting for multiple lines or not, have an equal chance of being churned.

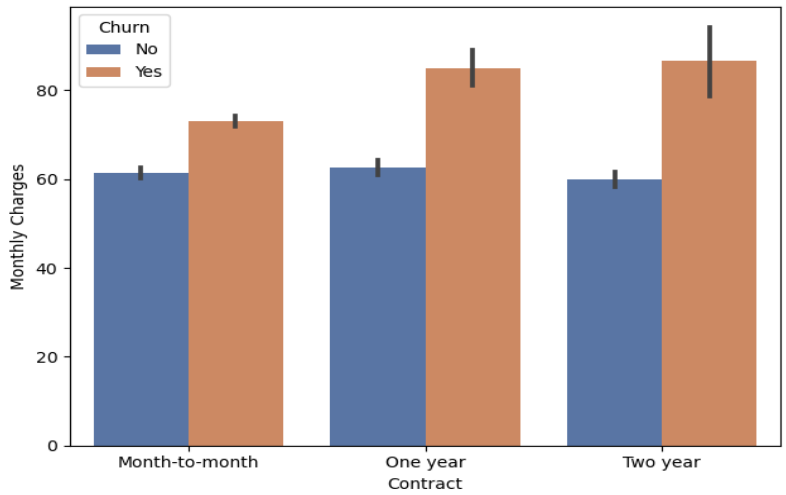
3. Customers with fibre optic internet service experience higher churn rates compared to others.

4. Furthermore, customers without online backup, online security, and device protection tend to discontinue the services more often.

5. Obviously, customers who lack tech support are more likely to churn.

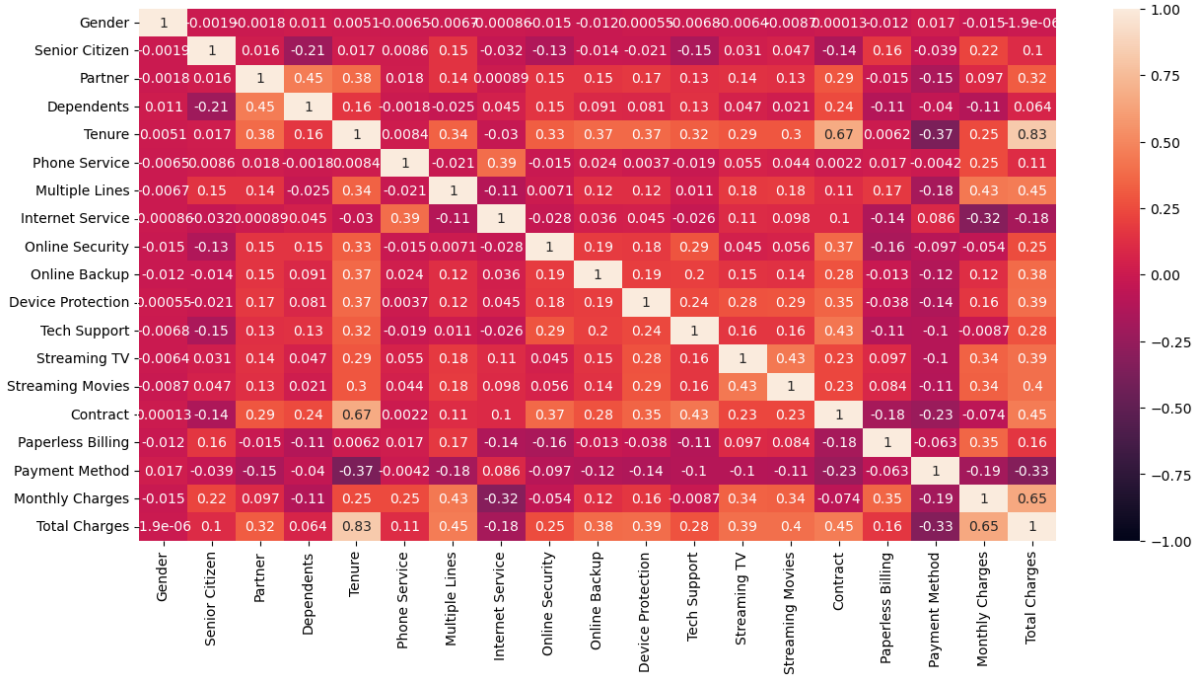
6. Lastly, customers who are not streaming TV and movies have a slightly higher chance of churning.

3.2.3. Multivariate analysis



Customers with high monthly charges tend to churn more across all contract types.

Correlation between variables using Heatmap



There are no highly correlated variables, except for a few instances. For example, we observe a moderate correlation between 'Tenure' and 'Contract'. Additionally, 'Tenure' has a high correlation with 'Total Charges.

3.3. Data Transformation:

3.3.1. Converting categorical variables to numerical

Using 'LabelEncoder()', all the categorical values will be replaced by numerical values as per ascending order of unique categorical values.

```
le=LabelEncoder()                          # using label encoder

for x in df_obj:
    df[x]=le.fit_transform(df[x])          # fit and transform values by label encoder


df.head()
```

Python

Gender	Senior Citizen	Partner	Dependents	Tenure	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
0	0	1	0	1	0	1	0	0	2	0	0	0	0	0	1	2	29.85	29.85	0
1	0	0	0	34	1	0	0	2	0	2	0	0	0	1	0	3	56.95	1889.50	0
1	0	0	0	2	1	0	0	2	2	0	0	0	0	0	1	3	53.85	108.15	1
1	0	0	0	45	0	1	0	2	0	2	2	0	0	1	0	0	42.30	1840.75	0
0	0	0	0	2	1	0	1	0	0	0	0	0	0	0	1	2	70.70	151.65	1

3.3.2. Scaling

Scaling is a technique of scaling down the values into a specific range. One of the types of scaling is Standard Scalar which scales down the values into the range of '-3 to 3' by converting the whole distribution into standard normal distribution.

Scaling

```
np.set_printoptions(suppress=True)
```

```
scaler = StandardScaler()
scaler.fit(x)                                  #training on data
x=scaler.transform(x)                          #transforming the data

print(x)
```

```
[[-1.00955867 -0.43991649  1.03453023 ...  0.39855772 -1.16032292
  -0.99424193]
 [ 0.99053183 -0.43991649 -0.96662231 ...  1.33486261 -0.25962894
  -0.17324412]
 [ 0.99053183 -0.43991649 -0.96662231 ...  1.33486261 -0.36266036
  -0.95967407]
```

4. Model Development

4.1. Pre-model building data splitting

Using 'train_test_split' from Sklearn, which splits the datasets into train and test. The data is split by 70% for training and rest for testing.

4.2. Applying different algorithms to determine which one performs the best.

Here I have created 3 parts in which the 1st one consists of all traditional algorithms like 'Decision Tree', 'KNN', 'SVM', 'Logistic Regression'. And 2nd consists of model with ensemble techniques like 'Random Forest', 'Extra Trees', 'Ada Boost', 'Gradient Boosting' and 'XG Boost'. Lastly, Artificial neural network is built upon our data.

Once all the models is build we will check which is performing better in terms of evaluation metrics such as accuracy and recall.

```
Model: DecisionTreeClassifier
Accuracy: 72.55
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.80      0.81      1568
           1       0.47      0.52      0.49       545

    accuracy                           0.73      2113
   macro avg       0.65      0.66      0.65      2113
weighted avg       0.73      0.73      0.73      2113

----------------------
Model: KNeighborsClassifier
Accuracy: 76.34
Classification report:
              precision    recall  f1-score   support

           0       0.84      0.84      0.84      1568
           1       0.54      0.54      0.54       545

    accuracy                           0.76      2113
   macro avg       0.69      0.69      0.69      2113
weighted avg       0.76      0.76      0.76      2113

----------------------
Model: SVC
Accuracy: 80.03
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1568
           1       0.66      0.48      0.55       545

    accuracy                           0.80      2113
   macro avg       0.74      0.69      0.71      2113
weighted avg       0.79      0.80      0.79      2113

----------------------
Model: LogisticRegression
Accuracy: 79.93
Classification report:
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      1568
...
   macro avg       0.74      0.71      0.72      2113
weighted avg       0.79      0.80      0.79      2113
```

```
Model: RandomForestClassifier
Accuracy: 78.47
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1568
           1       0.61      0.48      0.53       545

    accuracy                           0.78      2113
   macro avg       0.72      0.68      0.70      2113
weighted avg       0.77      0.78      0.78      2113

----------------------
Model: ExtraTreesClassifier
Accuracy: 78.14
Classification report:
              precision    recall  f1-score   support

           0       0.83      0.88      0.86      1568
           1       0.59      0.48      0.53       545

    accuracy                           0.78      2113
   macro avg       0.71      0.68      0.70      2113
weighted avg       0.77      0.78      0.77      2113

----------------------
Model: GradientBoostingClassifier
Accuracy: 80.36
Classification report:
              precision    recall  f1-score   support

           0       0.84      0.91      0.87      1568
           1       0.65      0.51      0.57       545

    accuracy                           0.80      2113
   macro avg       0.75      0.71      0.72      2113
weighted avg       0.79      0.80      0.79      2113

----------------------
Model: AdaBoostClassifier
Accuracy: 80.12
Classification report:
              precision    recall  f1-score   support

           0       0.85      0.90      0.87      1568
...
   macro avg       0.70      0.68      0.69      2113
weighted avg       0.77      0.78      0.77      2113
```

```
67/67 [==============================] - 0s 2ms/step
Accuracy: 80.08
Classification report:
              precision    recall  f1-score   support

           0       0.84      0.90      0.87      1568
           1       0.64      0.52      0.57       545

    accuracy                           0.80      2113
   macro avg       0.74      0.71      0.72      2113
weighted avg       0.79      0.80      0.79      2113
```

We can clearly see that the accuracy of most algorithms, including the neural network, is around 80%, with the best-performing models being AdaBoost and Gradient Boosting. However, it's important to remember that the dataset is imbalanced. To address this issue, we will use SMOTE.

4.3. Smote

Synthetic minority oversampling Technique. This technique allows us to up sample the minority class observation to reach the level of majority class by creating synthetic samples similar to existing samples.

```
Before OverSampling, counts of label '1':  1312
Before OverSampling, counts of label '0':  3618
After OverSampling, the shape of train_X:  (7236, 19)
After OverSampling, the shape of train_y:  (7236,)
After OverSampling, counts of label '1':  3618
After OverSampling, counts of label '0':  3618
```

4.4. Again, after running all the algorithms using Smote samples, out of all the algorithms, Gradient Boosting and AdaBoost perform the best. However, we have selected Gradient Boosting for further fine-tuning the model because it has almost the same recall for the minority class

```
Model: GradientBoostingClassifier
Accuracy: 79.46
Classification report:
              precision    recall  f1-score   support

         0.0       0.88      0.83      0.86      1556
         1.0       0.60      0.69      0.64       557

    accuracy                           0.79      2113
   macro avg       0.74      0.76      0.75      2113
weighted avg       0.81      0.79      0.80      2113
```

but with better accuracy. Hence, creating a classifier name ‘model_gb’ and storing our base model of Gradient Boosting classifier with Smote samples.

4.5. Hyperparameter Tuning using Randomized Search CV

RandomizedSearchCV is a hyperparameter tuning technique in machine learning that is used to find the best parameters for a model. It generates a grid of hyperparameter values and randomly selects combinations to train the model and score. This allows users to control the number of parameter combinations that are attempted.

```python
# Define the parameter distribtuion to search
param_dist = {
    'criterion': ['friedman_mse', 'squared_error'],
    'n_estimators': randint(100, 200),              # Uniformly sample over the given range
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': randint(2, 11),
    'min_samples_leaf': randint(1, 6),
    'max_features': ['sqrt', 'log2', None]
}

model_GB = GradientBoostingClassifier(random_state=42)

random_search = RandomizedSearchCV(estimator=model_GB, param_distributions=param_dist,
                                   n_iter=100, cv=5, verbose=3, random_state=42, n_jobs=-1,
                                   scoring='recall')

# Fit RandomizedSearchCV to the data
random_search.fit(x_train_new, y_train_new)

# Print the best parameters and the corresponding score
print("Best Hyperparameters:", random_search.best_params_)
print("Best Score:", random_search.best_score_)
```

Python

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Best Hyperparameters: {'criterion': 'friedman_mse', 'max_depth': 20, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimato
```

```python
    # buidling a model using Randomsearchcv parameters

    y_pred = random_search.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(Y_test, y_pred)

    # Generate a classification report
    class_repo = classification_report(Y_test, y_pred)

    # Print the results
    print(f'Accuracy: {round(accuracy * 100, 2)}')
    print(f'Classification report: \n{class_repo}')
```

```
Accuracy: 78.23
Classification report:
              precision    recall  f1-score   support

         0.0       0.85      0.85      0.85      1556
         1.0       0.59      0.58      0.58       557

    accuracy                           0.78      2113
   macro avg       0.72      0.72      0.72      2113
weighted avg       0.78      0.78      0.78      2113
```

Since there is no increase in performance of model, we'll try sub setting some features using PCA.

4.6. PCA

It is method of retaining minimum number of features that contribute maximum number of information (variance) in the datasets. The sole purpose of this method is to reduce the dimensionality i.e., the complexity in the dataset, and not to increase the accuracy or any other evaluation metrics.

```
# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 0.95)                    #to view the entire eigen vector,no subsetting
x_train_pca = pca.fit_transform(x_train_new)
x_test_pca = pca.transform(X_test)
explained_variance = pca.explained_variance_ratio_
print(explained_variance)
```

```
[0.22587257 0.12599686 0.07656962 0.06444913 0.05827133 0.05386417
 0.0527947  0.04386427 0.04146521 0.03932528 0.03886149 0.03463311
 0.03367991 0.03127884 0.02835668 0.02312644]
```

```
model_gb_pca = GradientBoostingClassifier(random_state=77)

model_gb_pca.fit(x_train_pca,y_train_new)
y_pred = model_gb_pca.predict(x_test_pca)

# Calculate accuracy
accuracy = accuracy_score(Y_test, y_pred)

# Generate a classification report
class_repo = classification_report(Y_test, y_pred)

# Print the results
print(f'Accuracy: {round(accuracy * 100, 2)}')
print(f'Classification report: \n{class_repo}')
```

```
Accuracy: 74.73
Classification report:
              precision    recall  f1-score   support

         0.0       0.90      0.74      0.81      1556
         1.0       0.51      0.76      0.61       557

    accuracy                           0.75      2113
   macro avg       0.71      0.75      0.71      2113
weighted avg       0.80      0.75      0.76      2113
```

After using PCA and exploring various options for tuning to optimize our model, we found that our base Gradient Boosting model ‘model_gb’ with Smote samples performs better in terms of accuracy and recall. Therefore, it will be used for further model deployment and predictions.

5. Model Deployment

5.1. Dumping our model.

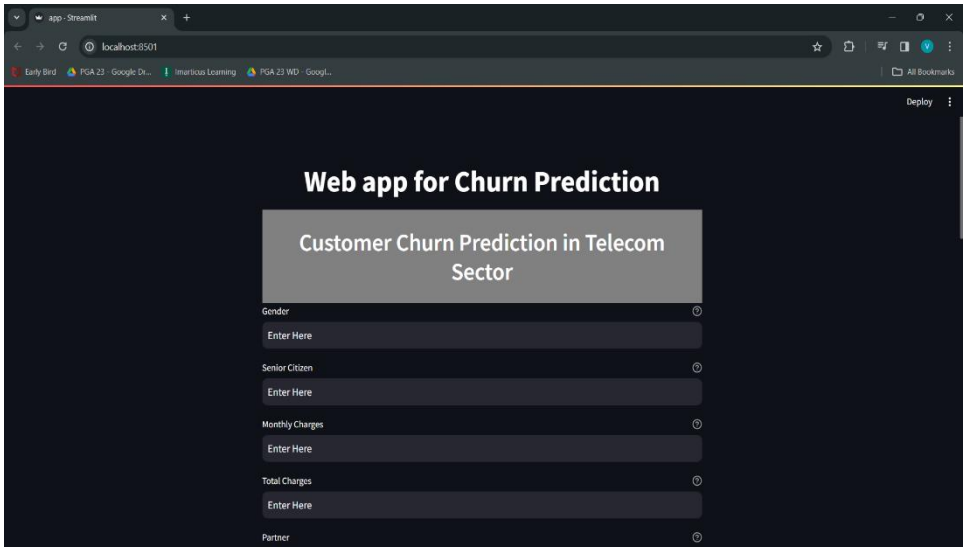
Using ‘pickle’ library for dumping our model, which can be used for deployment in the name of ‘model.sav’.

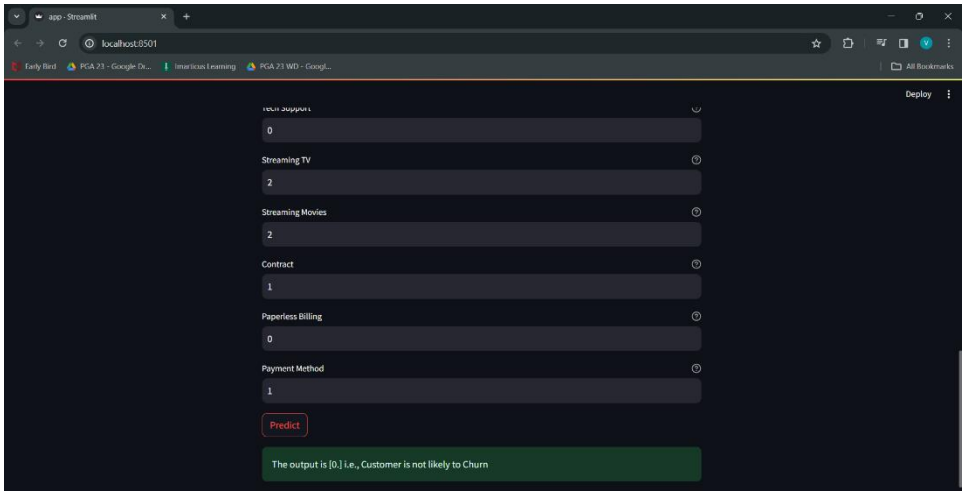
```
filname = 'model.sav'


pickle.dump(model_gb, open(filname, 'wb'))
```

5.2. Deployment using Streamlit framework.

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers.





6. Conclusion

As the aim of this project was to provide telecom companies with a powerful tool for predicting customer churn and assisting in analysing patterns of customer churn behaviour, we have performed comprehensive exploratory data analysis (EDA) on the historical customer data. Here are some key insights that follow.

- We can observe that customers with low tenure, such as 1 to 5 months, tend to churn more frequently.
- Customers facing monthly charges in the range of 70 to 110 are more likely to churn, whereas those with lower monthly charges tend to remain with the service.
- Although Senior Citizens make up a smaller portion of the customer base, they have a higher likelihood of churn compared to non-Senior Citizens.
- It appears that most customers prefer month-to-month contracts and pay via electronic check, yet these sub-categories also exhibit high churn rates, with nearly half of the customers churning.
- Customers without online backup, online security, and device protection tend to discontinue the services more often.
- Lastly, customers who lack tech support are more likely to churn.

Lastly, I have built a model that helps in determining whether a customer will churn or not, achieving an accuracy of around 80%, and recall rates of approximately 85% for the majority class and 70% for the minority class, respectively. This model has been deployed using Streamlit to create an interface that companies can use