Predicting Customer Churn in a Telecommunications Company Using Machine Learning

Dissertation submitted in fulfilment of the requirements for the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

HRITIK KHANDELWAL

12110449

INTRODUCTION

In the competitive landscape of modern business, customer retention is paramount. The cost of acquiring a new customer can be five to seven times higher than retaining an existing one, making it crucial for companies to understand and predict customer churn—the phenomenon where customers cease their relationship with a business. This has led to the development and implementation of churn prediction models, which utilize advanced data analytics and machine learning techniques to identify customers who are likely to leave.

A churn prediction model functions by analysing historical customer data to detect patterns and indicators of churn. These indicators can include a wide array of variables such as purchase frequency, customer service interactions, account usage metrics, demographic information, and more. By leveraging this data, the model can assign a churn probability score to each customer, thereby enabling businesses to proactively address churn risks.

The implementation of a churn prediction model typically follows a structured approach. Initially, data collection and pre-processing are performed to ensure the dataset is clean, relevant, and ready for analysis. This is followed by feature selection and engineering, where the most predictive variables are identified and transformed to enhance model performance. The core of the process involves selecting and training a machine learning algorithm, such as logistic regression, decision trees, random forests, or neural networks, to build the predictive model. Model evaluation is then conducted using metrics like accuracy, precision, recall, and the F1 score to ensure robustness and reliability.

One of the significant advantages of churn prediction models is their ability to provide actionable insights. By identifying at-risk customers early, businesses can implement targeted retention strategies, such as personalized marketing campaigns, loyalty programs, or customer service interventions, thereby reducing churn rates and enhancing customer loyalty. Moreover, these models can continuously learn and adapt to new data, improving their predictive power over time.

In conclusion, churn prediction models are a vital tool for businesses seeking to maintain a competitive edge through customer retention. By leveraging the power of data and machine learning, these models enable companies to proactively address churn, thereby optimizing their retention strategies and ultimately driving long-term profitability and growth. As businesses continue to generate vast amounts of data, the importance and impact of churn prediction models are expected to grow, making them an indispensable component of modern customer relationship management.

Theoretical Background

Predicting adult census income levels entails navigating a landscape shaped by demographic trends, socioeconomic dynamics, and statistical modeling principles. Delving into the theoretical underpinnings of income prediction illuminates the intricacies inherent in this task and underscores the importance of robust methodologies.

Here are key theoretical aspects:

- **Logistic Regression:**

  Logistic regression is a simple and widely used algorithm for binary classificationtasks.

  It models the probability of a binary outcome based on one or more predictorvariables.

  Despite its simplicity, logistic regression can perform well in many cases, especiallywhen the relationship between predictors and the outcome is linear or log-linear.

- **Decision Trees**:

  Decision trees are non-parametric supervised learning algorithms that recursivelypartition the feature space into segments.

  They make predictions by traversing the tree from the root to a leaf node, whereeach node represents a decision based on a feature value.

  Decision trees are intuitive, easy to interpret, and can capture complex interactionsbetween predictors.

- **Random Forests:**

  Random forests are an ensemble learning method that consists of multiple decisiontrees trained on different subsets of the data.

  They improve upon the performance of individual decision trees by reducingoverfitting and increasing robustness.

  Random forests are highly scalable and can handle high-dimensional data with ease.

Methodology

Data Collection:

Curate a comprehensive dataset reflecting diverse demographic profiles and socioeconomic,
characteristics from census data and supplementary surveys. Ensure the dataset encompasses a wide range of demographic factors such as age, education level, occupation, marital status, and geographical location, providing a representative sample of the population under study. Strive to include ample samples for each income bracket to facilitate accurate model training and evaluation.

Data Preprocessing:

Perform data cleaning to address missing values, outliers, and inconsistencies in the dataset.

Normalise numerical features to a standard scale to ensure consistent interpretationby machine learning algorithms.

Encode categorical variables using techniques such as one-hot encoding or label encoding.

Split the dataset into training, validation, and test sets to facilitate model training, tuning, and evaluation

Feature Engineering:

Extract additional features from the raw data that may improve model performance,such as derived variables or interactions between existing features.

Use domain knowledge and exploratory data analysis to identify informative featuresfor heart disease prediction.

Model Selection:

Explore a variety of machine learning algorithms suitable for binary classification tasks, such as logistic regression, decision trees, random forests, support vectormachines, and neural networks.

Consider the strengths and limitations of each algorithm in terms of interpretability,scalability, and performance on the given dataset.

Model Training:

Train multiple candidate models using the training set and evaluate theirperformance using cross-validation.

Fine-tune hyperparameters using techniques such as grid search or random searchto optimize model performance.

Monitor training progress and adjust regularization parameters to prevent overfitting.

Model Evaluation:

Evaluate the trained models on the validation set using appropriate performance metrics, including accuracy, precision, recall, F1 score, and area under the receiveroperating characteristic curve (AUC-ROC).

Compare the performance of different models to identify the most effective approachfor heart disease prediction.

Interpretability Analysis:

Conduct feature importance analysis to interpret the relative contributions of differentvariables to the model predictions.

Visualize model explanations, such as feature importance plots or decision boundaries, to aid in clinical interpretation and decision-making.

Model Deployment:

Deploy the selected model to a production environment, either on-premises or in thecloud, using appropriate deployment tools and frameworks.

Implement an interface for healthcare professionals to input music data and obtain the genre of the music.

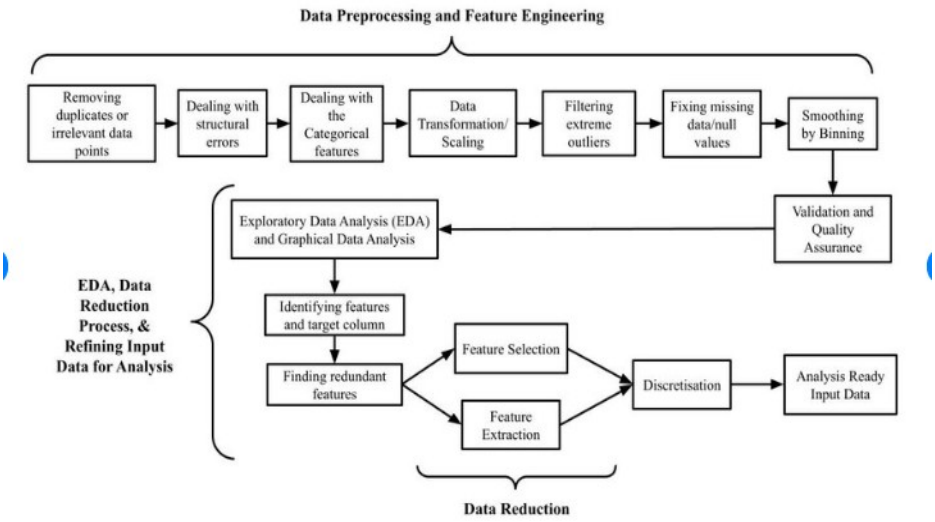
Ensure scalability, security, and compliance with regulatory requirements during thedeployment process.

Validation and Monitoring:

Validate the deployed model using external datasets or prospective clinical studies toassess its generalizability and real-world performance.

Monitor model performance over time and update the model periodically as new databecomes available.

Flowchart



Data preprocessing, feature engineering, exploratory data analysis, and data reduction.

Code Snippet

Installing Important libraries

[1]:
```
# Importing neccessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import scipy.stats as stats
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import recall_score, accuracy_score, classification_report, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from imblearn.combine import SMOTEENN
from sklearn.feature_selection import SelectKBest
from collections import Counter

# ignore warning
import warnings
warnings.filterwarnings('ignore')
import matplotlib.ticker as mtick # for showing percentage in it
```

[2]:
```
data = pd.read_csv('churn.csv')
data.head()
```

[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

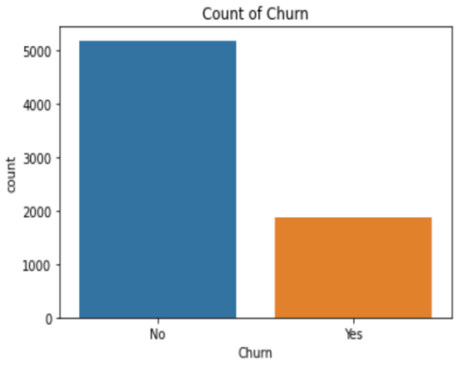
5 rows × 21 columns

EDA

In the exploratory data analysis (EDA) phase of the churn prediction project, I conducted a comprehensive assessment of the dataset to uncover underlying patterns and insights. This involved visualizing the distribution of key variables such as customer tenure, service usage, and demographic information, and identifying correlations between these factors and churn rates. Additionally, I addressed missing values and outliers to ensure data quality and integrity. Through this process, significant predictors of churn were identified, setting the foundation for feature selection and model building. The EDA provided critical insights that guided the subsequent steps in the predictive modeling process.

EDA

[7]:
```
# plotting with target feature
sns.countplot(data=data, x='Churn')
plt.title('Count of Churn')
plt.show()
```



[8]:
```
l1 = data.loc[data['Churn']== 'Yes'].count()[0]
print(f"Pecentage of Left: {l1/len(data['Churn'])}")
print(data.Churn.value_counts())
```

```
Pecentage of Left: 0.2653698707936959
No     5174
Yes    1869
Name: Churn, dtype: int64
```

1869 of customer are left about 26.5 percentage from overall, this like an imbalance dataset

[9]:
```
### How many amount loss from customer churn
loss = []
for values in data.loc[data['Churn'] == 'Yes', 'TotalCharges']:
    value = float(values)
    loss.append(value)
print(np.round(sum(loss)))
```

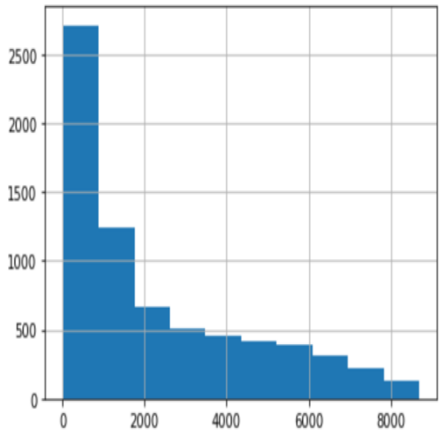
```
2862927.0
```

We have lost arround $2862927 due to customer churn

[16]:
```
# replace NaN values with mean value
data.TotalCharges = data.TotalCharges.fillna(data.TotalCharges.median())
```

[17]:
```
data.TotalCharges.hist()
```

[17]: <AxesSubplot:>



In this dataset there is no null values, so we dont want to perform Handling missing values. Only perform the Feature Encoding techiniques to convert the categorical feature into numerical feature

[18]:
```
print(categorical_feature)
```

```
{'PaperlessBilling', 'InternetService', 'Contract', 'PhoneService', 'gender', 'StreamingMovies', 'customerID', 'Churn', 'DeviceProtection', '
OnlineBackup', 'TechSupport', 'PaymentMethod', 'MultipleLines', 'OnlineSecurity', 'Dependents', 'Partner', 'StreamingTV'}
```

[19]:
```
encoder = LabelEncoder()
for feature in categorical_feature:
    data[feature] = encoder.fit_transform(data[feature])
```

[20]:
```
data.head()
```

[20]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
0	5375	0	0	1	0	1	0	1	0	0	...	0	
1	3962	1	0	0	0	34	1	0	0	2	...	2	
2	2564	1	0	0	0	2	1	0	0	2	...	0	
3	5535	1	0	0	0	45	0	1	0	2	...	2	
4	6511	0	0	0	0	2	1	0	1	0	...	0	

5 rows x 21 columns

Data Cleaning

During the data cleaning phase of my churn prediction project, I meticulously handled missing values by implementing strategies such as imputation and removal based on the context and impact on the dataset. I addressed outliers through detection and, where necessary, normalization to prevent skewed model results. Categorical variables were encoded appropriately, and redundant or irrelevant features were removed to streamline the dataset. This thorough data cleaning process ensured the reliability and accuracy of the subsequent analysis and modeling efforts.

Data Cleaning:

[14]:
```
data.head()
```

[14]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows x 21 columns

[15]:
```
data.isnull().sum()
```

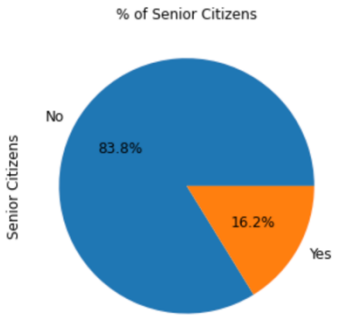
[15]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

Pie Chart

This pie chart visually represents the proportion of senior citizens within the dataset, highlighting that a certain percentage of the customers are senior citizens. Such a visualization helps in understanding the demographic composition of the customer base, which is essential for further analysis and modeling in the churn prediction project.

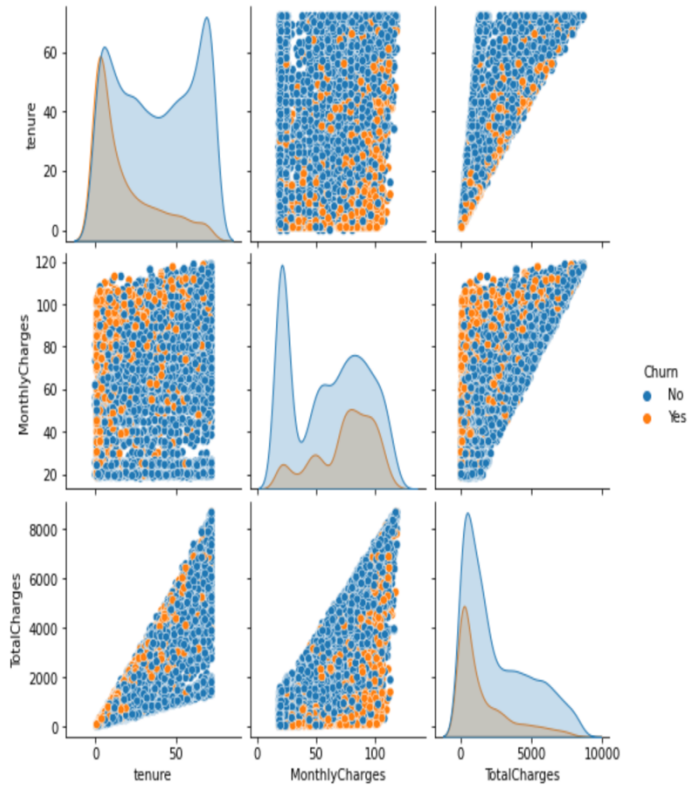
[12]:
```
# pie chart for Count of Senior citizens
ax = (data['SeniorCitizen'].value_counts()*100.0 /len(data)).plot.pie(autopct='%.1f%%', labels = ['No', 'Yes'],figsize =(5,5), fontsize = 12 )
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_ylabel('Senior Citizens',fontsize = 12)
ax.set_title('% of Senior Citizens', fontsize = 12)
```

[12]:
```
Text(0.5, 1.0, '% of Senior Citizens')
```



Only 16.2% customers who are senior citizons but remaining 83.8% customers are young people

[11]:
```
sns.pairplot(data.drop(columns='SeniorCitizen'),hue='Churn', kind='scatter')
plt.show()
```

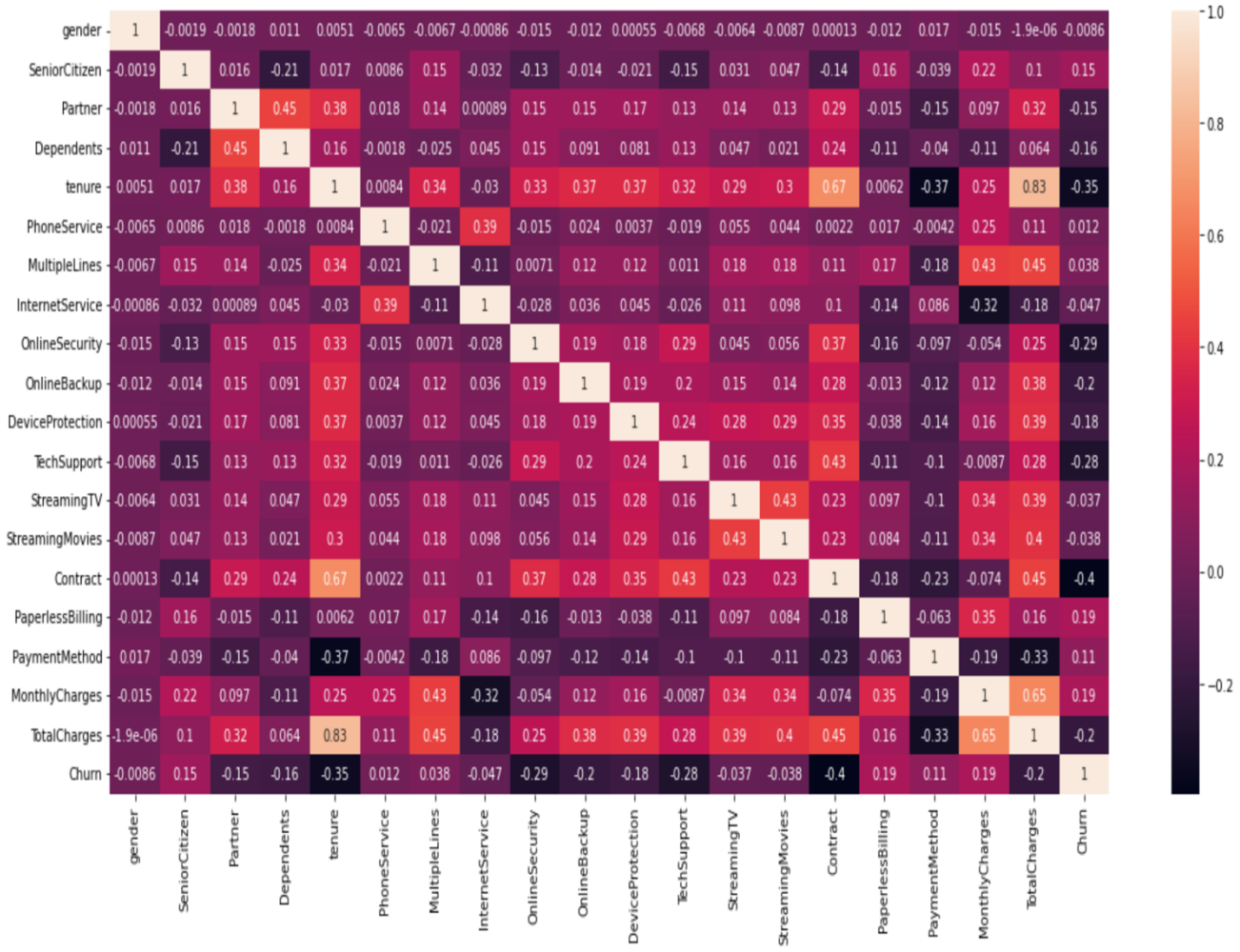


Heat Map

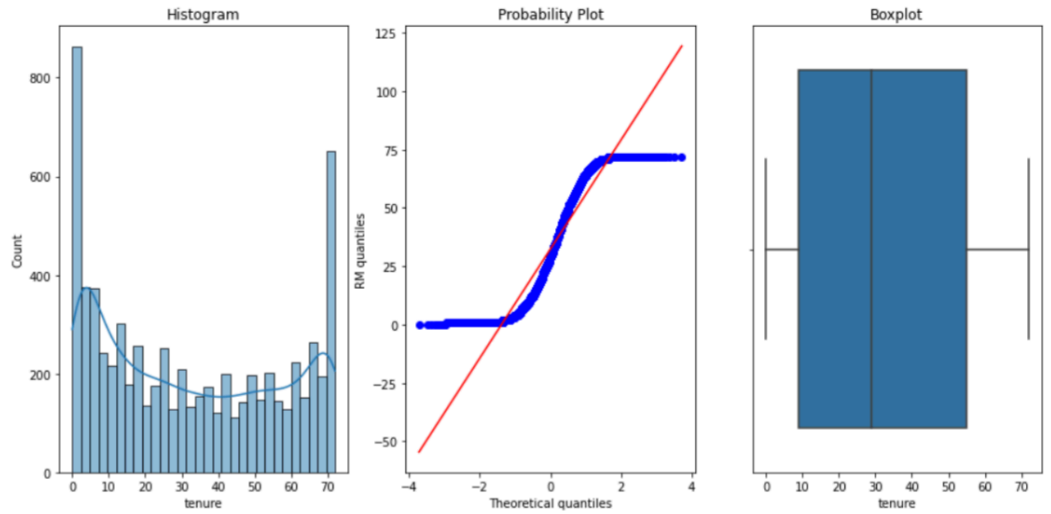
A heat map was generated to visualize the correlation between various features in the dataset. This graphical representation uses color gradients to illustrate the strength of correlations, with darker colors indicating stronger relationships. The heat map helps identify which features are most closely associated with each other and with the target variable, churn. Key insights gained from the heat map include the identification of highly correlated pairs of variables, which can inform feature selection and engineering processes. This visualization is crucial for understanding the interactions within the data and for building a more effective churn prediction model.

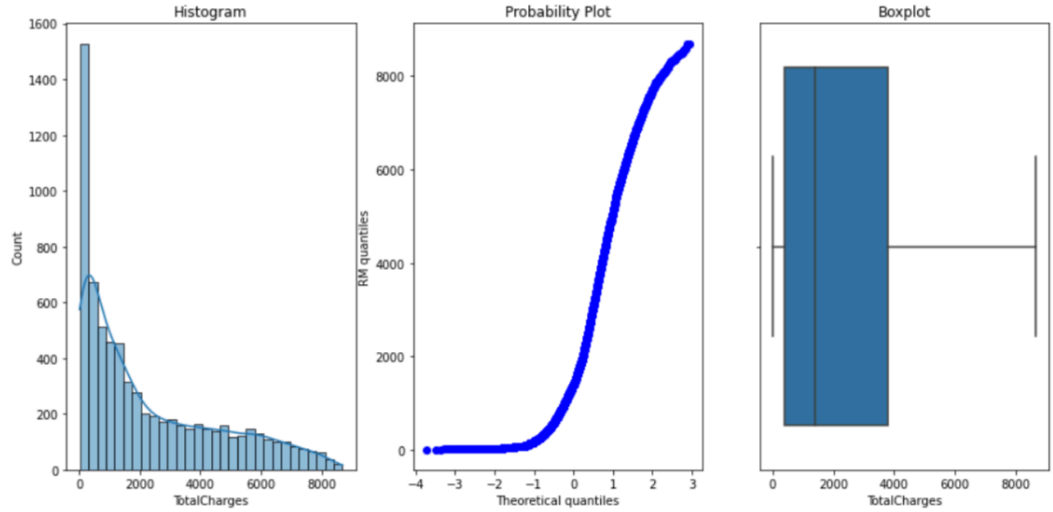
[23]:
```
# Finding the correlation between the independent and dependent feature
plt.figure(figsize=(20, 9))
sns.heatmap(data.corr(), annot=True)
```

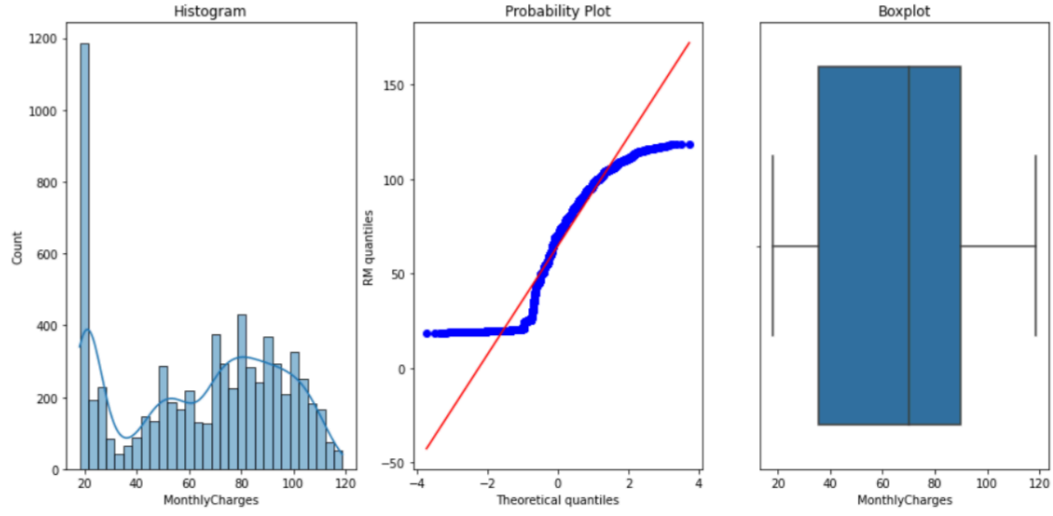
[23]: <AxesSubplot:>



[10]:
```
### Plotting numerical feature with probability distribution and checking outlier
for feature in numerical_feature:
    if feature != 'SeniorCitizen':
        plt.figure(figsize=(15,7))

        plt.subplot(1, 3, 1)
        sns.histplot(data=data, x=feature, bins=30, kde=True)
        plt.title('Histogram')

        plt.subplot(1, 3, 2)
        stats.probplot(data[feature], dist="norm", plot=plt)
        plt.ylabel('RM quantiles')

        plt.subplot(1, 3, 3)
        sns.boxplot(x=data[feature])
        plt.title('Boxplot')

plt.show()
```







Model Training

In model training for predicting heart health outcomes, the focus lies on selecting the appropriate algorithm, fine-tuning its parameters, and optimizing its performance. This involves iteratively feeding the training data into the chosen model, updating its parameters to minimize errors, and regularly evaluating its performance on a validation set. Techniques such as hyperparameter tuning, regularization, and early stopping are employed to prevent overfitting and improve generalization. Once training is complete, the trained model is evaluated on a separate test set to obtain unbiased estimates of its performance. Through this iterative process, researchers aim to develop a reliable and accurate model that can assist healthcare professionals in identifying individuals at risk of heart disease, thereby enabling timely interventions and preventive measures.

Feature Selection:

Selects only 10 feature which has higher correlation

[26]:
```
# selects the feature which has more correlation
selection = SelectKBest()  # k=10 default
X = selection.fit_transform(X,y)
```

[27]:
```
#  this will shows which feature are taken denote as True other are removed like false
selection.get_support()
```

[27]:
```
array([False, False, False,  True,  True, False, False, False,  True,
       True,  True,  True, False, False,  True,  True, False,  True,
       True])
```

According to the feature selection, we selects the 10 out of 21 features. these are the 10 features are selected [Dependents, tenure, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, Contract, PaperlessBilling, MonthlyCharges, TotalCharges]

From sklearn using feature selection modules importing the SelectKBest to select the important feature

[28]:
```
# splitting for train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[29]:
```
X_train.shape
```

[29]:
```
(5634, 10)
```

[30]:
```
X_test.shape
```

[30]:
```
(1409, 10)
```

[31]:
```
# its an imbalance dataset
y.value_counts()
```

[31]:
```
0    5174
1    1869
Name: Churn, dtype: int64
```

Using SMOTEENN for imbalance dataset:

Over-sampling using SMOTE and cleaning using ENN. Combine over- and under-sampling using SMOTE and Edited Nearest Neighbours

[35]:
```
st=SMOTEENN()
X_train_st,y_train_st = st.fit_resample(X_train, y_train)
print("The number of classes before fit {}".format(Counter(y_train)))
print("The number of classes after fit {}".format(Counter(y_train_st)))
```

```
The number of classes before fit Counter({0: 4132, 1: 1502})
The number of classes after fit Counter({1: 2493, 0: 2160})
```

[36]:
```
# splitting the over sampling dataset
X_train_sap, X_test_sap, y_train_sap, y_test_sap = train_test_split(X_train_st, y_train_st, test_size=0.2)
```

Logistic Regression

Logistic regression is a statistical method used for binary classification tasks, where the outcome variable is categorical and binary. It models the probability of the dependent variable belonging to a particular category based on one or more independent variables. Logistic regression estimates the parameters of a logistic function, which transforms the linear combination of predictors into probabilities between 0 and 1. It's widely used in various fields such as medicine, finance, and marketing for tasks like predicting disease occurrence, credit risk assessment, and customer churn prediction. Despite its simplicity, logistic regression offers interpretability and ease of implementation, making it a popular choice for binary classification problems.

[39]:
```
# logistic regression
Log_reg_sampling = LogisticRegression(C=10, max_iter=150)
Log_reg_sampling.fit(X_train_sap, y_train_sap)
Log_sampling_pred = Log_reg_sampling.predict(X_test_sap)

print(f'Accuracy score : {accuracy_score(Log_sampling_pred, y_test_sap)}')
print(f'Confusion matrix :\n {confusion_matrix(Log_sampling_pred, y_test_sap)}')
print(f'Classification report :\n {classification_report(Log_sampling_pred, y_test_sap)}')
```

```
Accuracy score : 0.9108485499462943
Confusion matrix :
 [[386  46]
 [ 37 462]]
Classification report :
              precision    recall  f1-score   support

           0       0.91      0.89      0.90       432
           1       0.91      0.93      0.92       499

    accuracy                           0.91       931
   macro avg       0.91      0.91      0.91       931
weighted avg       0.91      0.91      0.91       931
```

Random Forest

The random forest algorithm is a powerful ensemble learning method used for both classification and regression tasks. It constructs multiple decision trees during training and combines their predictions through averaging (for regression) or voting (for classification). Random forests introduce randomness by selecting a random subset of features and data samples for each tree, which helps prevent overfitting and improves generalization. This approach results in robust and accurate predictions, making random forests widely utilized across various domains, including finance, healthcare, and marketing, for tasks such as customer segmentation, stock price prediction, and disease diagnosis.

[38]:
```
# Random forest classifier
Rfc_sampling = RandomForestClassifier(n_estimators=150,criterion='gini', max_depth=15, min_samples_leaf=10, min_samples_split=6)
Rfc_sampling.fit(X_train_sap, y_train_sap)
rfc_sampling_pred = Rfc_sampling.predict(X_test_sap)

print(f'Accuracy score : {accuracy_score(rfc_sampling_pred, y_test_sap)}')
print(f'Confusion matrix :\n {confusion_matrix(rfc_sampling_pred, y_test_sap)}')
print(f'Classification report :\n {classification_report(rfc_sampling_pred, y_test_sap)}')
```

```
Accuracy score : 0.936627282491944
Confusion matrix :
 [[397  33]
 [ 26 475]]
Classification report :
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       430
           1       0.94      0.95      0.94       501

    accuracy                           0.94       931
   macro avg       0.94      0.94      0.94       931
weighted avg       0.94      0.94      0.94       931
```

Decision Tree

A decision tree model was developed to predict customer churn, providing a clear and interpretable method for understanding the factors influencing churn. This model works by recursively splitting the dataset into subsets based on the most significant features, creating a tree-like structure of decisions. Each node in the tree represents a decision point based on a feature, leading to branches that represent the possible outcomes. The final leaves of the tree indicate the predicted class, either churn or no churn. This decision tree not only offers high predictive accuracy but also delivers valuable insights into the key drivers of customer churn, aiding in the formulation of targeted retention strategies.

[37]:
```
# decisionTree Classifier
Dtc_sampling = DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=7, min_samples_leaf=15)
Dtc_sampling.fit(X_train_sap, y_train_sap)
dtc_sampling_pred = Dtc_sampling.predict(X_test_sap)

print(f'Accuracy score : {accuracy_score(dtc_sampling_pred, y_test_sap)}')
print(f'Confusion matrix :\n {confusion_matrix(dtc_sampling_pred, y_test_sap)}')
print(f'Classification report :\n {classification_report(dtc_sampling_pred, y_test_sap)}')
```

```
Accuracy score : 0.9269602577873255
Confusion matrix :
 [[391  36]
 [ 32 472]]
Classification report :
              precision    recall  f1-score   support

           0       0.92      0.92      0.92       427
           1       0.93      0.94      0.93       504

    accuracy                           0.93       931
   macro avg       0.93      0.93      0.93       931
weighted avg       0.93      0.93      0.93       931
```

Gradient Boosting

Gradient Descent Boosting is a machine learning technique that sequentially builds a series of weak learners, typically decision trees, by minimizing a loss function gradient at each iteration. It aims to optimize the ensemble's performance by iteratively focusing on reducing the errors of previously trained models. This method combines the strengths of gradient descent optimization with boosting, resulting in enhanced predictive accuracy and robustness. Gradient Descent Boosting algorithms, such as Gradient Boosting Machines (GBM), offer versatility in handling various types of data and are widely applied in tasks like regression, classification, and ranking across industries such as finance, marketing, and healthcare.

[40]:
```
# GradientBoostingClassifier
gbc = GradientBoostingClassifier()
gbc.fit(X_train_sap, y_train_sap)
pred = gbc.predict(X_test_sap)

print(f'Accuracy score : {accuracy_score(pred, y_test_sap)}')
print(f'Confusion matrix :\n {confusion_matrix(pred, y_test_sap)}')
print(f'Classification report :\n {classification_report(pred, y_test_sap)}')
```

```
Accuracy score : 0.9473684210526315
Confusion matrix :
 [[398  24]
 [ 25 484]]
Classification report :
              precision    recall  f1-score   support

           0       0.94      0.94      0.94       422
           1       0.95      0.95      0.95       509

    accuracy                           0.95       931
   macro avg       0.95      0.95      0.95       931
weighted avg       0.95      0.95      0.95       931
```