CUSTOMER CHURN PREDICTION

PROJECT REPORT

PHASE-V

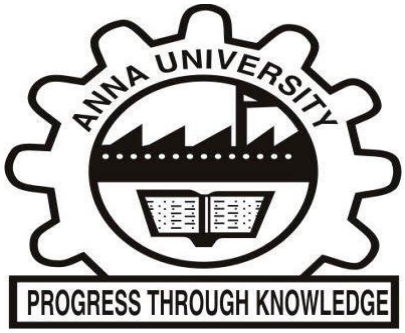
A report submitted in fulfilment of the project

of

DATA ANALYTICS with Cognos-Group I

In

NAAN MUDHALVAN



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY

CHENNAI 600025

Submission on

01-NOV-2023

Submitted by

J. NITHYASREE-2021103027

NAZMA.N -2021103301

J. DIVYA DHARSHINI-2021103011

MUTHURAMLAKSHMI K.I-2021103024

RITHIP.R-2021103566

OBJECTIVE:

The **objective** of a customer churn project is typically to reduce the rate at which customers stop doing business with a company. Churn, or customer attrition, can have a significant impact on a business's revenue and profitability. The project aims to identify the factors contributing to customer churn, develop strategies to retain customers, and implement measures to minimize churn. This often involves analyzing customer data to identify patterns and predictors of churn, creating targeted retention campaigns, and continuously monitoring and adjusting strategies based on the evolving customer landscape. Ultimately, the goal is to enhance customer loyalty and maximize customer lifetime value.

DESIGN THINKING:

1. Empathize: Understand user and website owner needs and challenges.

2. Define: Clarify the problem - predicting churn and enhancing user experience.

3. Ideate: Brainstorm data sources, analytical techniques, and potential solutions.

4. Prototype: Develop data collection methods and predictive models.

5. Test: Validate the model and gather insights.

6. Implement: Communicate findings and strategies to the website owners.

ANALYSIS OBJECTIVES:

1.Build a predictive model to identify potential churners.

2.Analyse user behaviour, demographics, and website interactions.

3.Visualize key performance indicators (KPIs) using IBM Cognos and Python.

4.Identify factors influencing customer churn. - Build a predictive model to identify potential

DEVELOPMENT PHASES:

1. Data Collection: Gather historical user data, including behaviour, demographics, and churn labels.

2. Data Preprocessing: Clean and prepare data for analysis.

3. Exploratory Data Analysis (EDA): Explore data distributions and relationships.

4. Feature Engineering: Create predictive features.

5. Model Building: Develop a churn prediction model.

6. Data Visualization: Utilize IBM Cognos and Python for visualization.

7. Insights Generation: Derive actionable insights from the analysis.

DATA VISUALIZATION:

1.Employ IBM Cognos for interactive dashboard creation with KPIs.

2.Utilize Python libraries like Matplotlib and Seaborn for supplementary visualizations.

DATA COLLECTION PROCESS:

1.Collect historical user data from the website's database.

2.Include user behaviour data, demographic information, and churn status.

3.Ensure data quality and consistency.

PYTHON CODE INTEGRATION:

1.Integrate Python code for data preprocessing, feature engineering, and model development.

2. Generate actionable insights and predictions using Python scripts.

Predicting Telco Customer Churn

Creating a machine learning model to predict customer churn. In this notebook we will build the prediction model using the SparkML library.

In [2]:
```
import pandas as pd
import numpy as np
import json
import os

# Import the Project Library to read/write project assets
from project_lib import Project
project = Project.access()

import warnings
warnings.filterwarnings("ignore")
```

1.0 Load and Clean data

We'll load our data as a pandas data frame.

In [3]:
```
# Place cursor below and insert the Pandas DataFrame for the Telco churn data
df_data_1 = pd.read_csv('Telco-Customer-Churn.csv')
df_data_1.head()
```

Out[3]:

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

5 rows × 21 columns

In [4]:
```
# for virtualized data
# df = data_df_1

# for local upload
df = df_data_1
```

1.1 Drop CustomerID feature (column)

In [5]:
```
df = df.drop('customerID', axis=1)
df.head(5)
```

Out[5]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtectio
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	N
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Ye
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	N
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Ye
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	N

1.2 Examine the data types of the features

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
gender              7043 non-null object
SeniorCitizen       7043 non-null int64
Partner             7043 non-null object
Dependents          7043 non-null object
tenure              7043 non-null int64
PhoneService        7043 non-null object
MultipleLines       7043 non-null object
InternetService     7043 non-null object
OnlineSecurity      7043 non-null object
OnlineBackup        7043 non-null object
DeviceProtection    7043 non-null object
TechSupport         7043 non-null object
StreamingTV         7043 non-null object
StreamingMovies     7043 non-null object
Contract            7043 non-null object
PaperlessBilling    7043 non-null object
PaymentMethod       7043 non-null object
MonthlyCharges      7043 non-null float64
TotalCharges        7043 non-null object
Churn               7043 non-null object
dtypes: float64(1), int64(2), object(17)
memory usage: 1.1+ MB
```

1.6 Visualize data

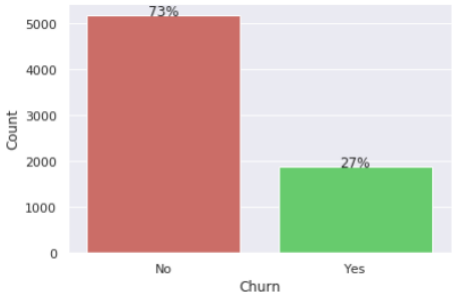
Data visualization can be used to find patterns, detect outliers, understand distribution and more. We can use graphs such as:

- Histograms, boxplots, etc: To find distribution / spread of our continuous variables.
- Bar charts: To show frequency in categorical values.

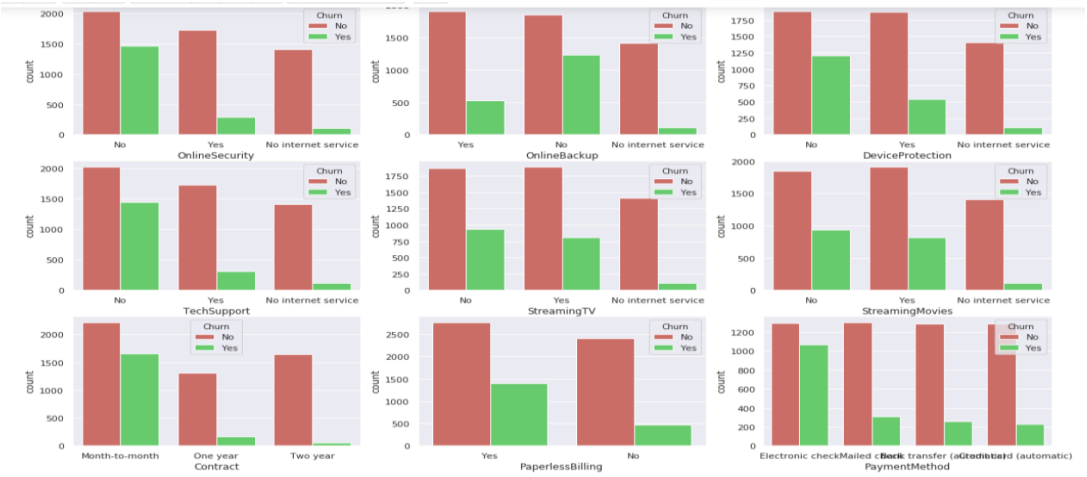
In [14]:
```
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

%matplotlib inline
sns.set(style="darkgrid")
sns.set_palette("hls", 3)
```

In [15]:
```
print(df.groupby(['Churn']).size())
churn_plot = sns.countplot(data=df, x='Churn', order=df.Churn.value_counts().index)
plt.ylabel('Count')
for p in churn_plot.patches:
    height = p.get_height()
    churn_plot.text(p.get_x()+p.get_width()/2., height + 1,'{0:.0%}'.format(height/float(len(df))),ha="center")
plt.show()
```

```
Churn
No     5174
Yes    1869
dtype: int64
```

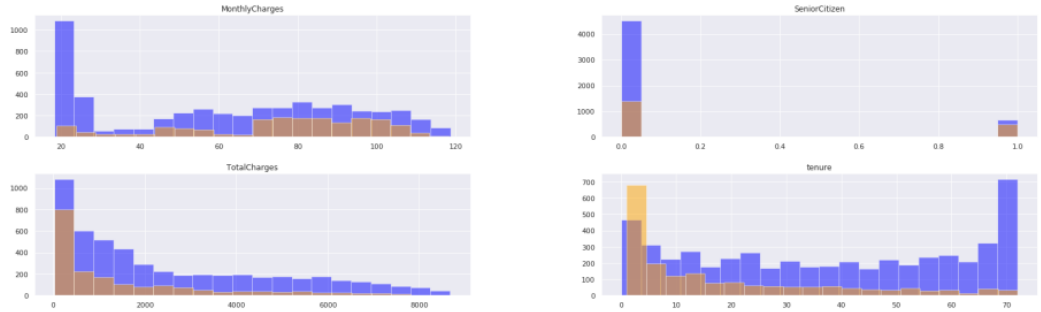


In [16]:
```
# categorical feature count plots
f, ((ax1, ax2, ax3), (ax4, ax5, ax6), (ax7, ax8, ax9), (ax10, ax11, ax12), (ax13, ax14, ax15)) = plt.subplots(5, 3, figsize=(20,
ax = [ax1, ax2, ax3, ax4, ax5, ax6, ax7, ax8, ax9, ax10, ax11, ax12, ax13, ax14, ax15 ]

for i in range(len(categorical_features)):
    sns.countplot(x = categorical_features[i], hue="Churn", data=df, ax=ax[i])
```

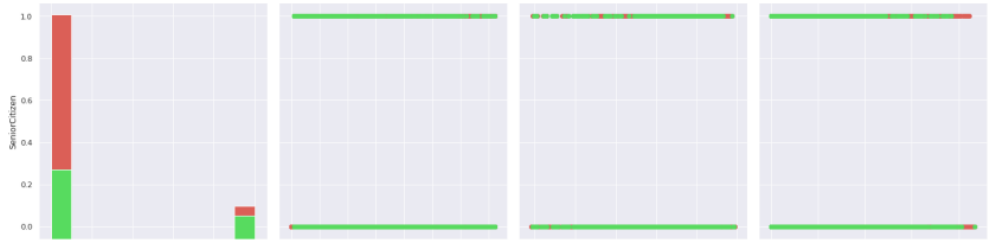


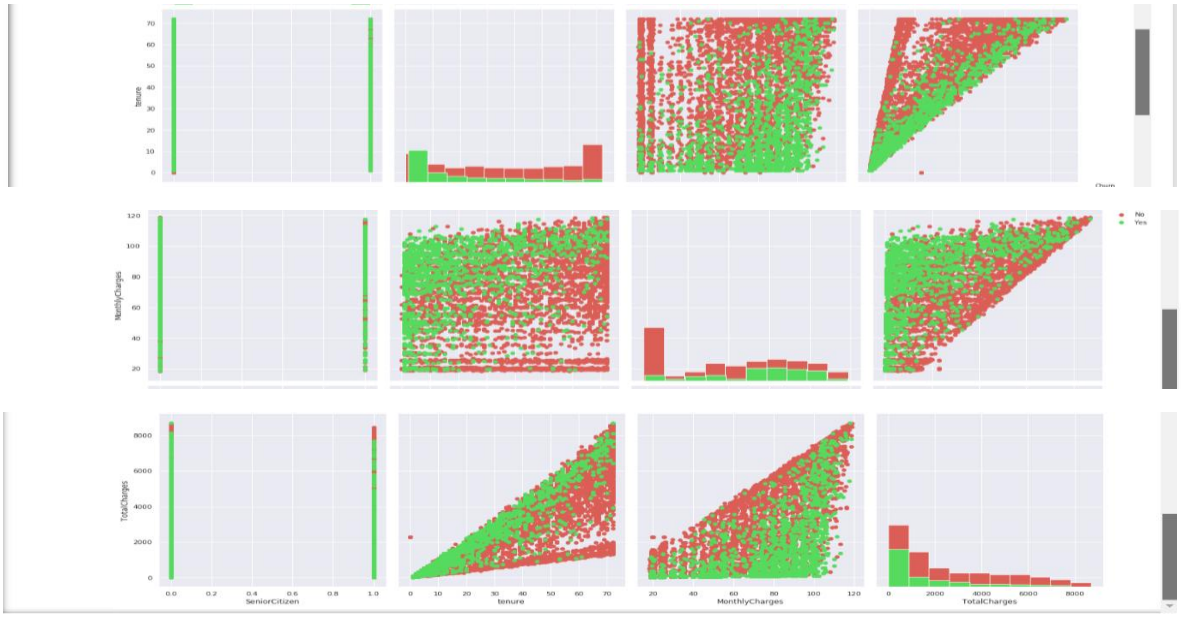
In [17]:
```
# Continuous feature histograms.
fig, ax = plt.subplots(2, 2, figsize=(28, 8))
df[df.Churn == 'No'][continuous_features].hist(bins=20, color="blue", alpha=0.5, ax=ax)
df[df.Churn == 'Yes'][continuous_features].hist(bins=20, color="orange", alpha=0.5, ax=ax)

# Or use displots
#sns.set_palette("hls", 3)
#f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(25, 25))
#ax = [ax1, ax2, ax3, ax4]
#for i in range(len(continuous_features)):
#    sns.distplot(df[continuous_features[i]], bins=20, hist=True, ax=ax[i])
```

Out[17]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f76c0cfda20>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f76c0cedeb8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f76c0c96898>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f76c0c51278>],
      dtype=object)
```

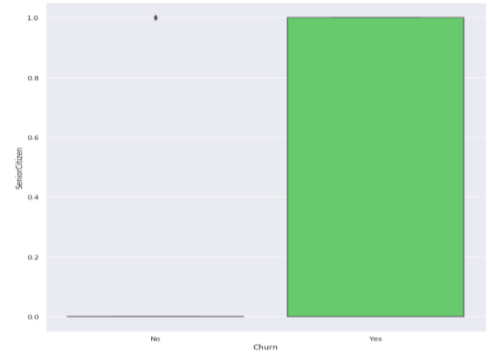


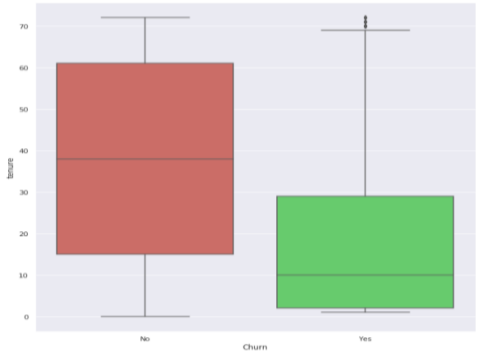
In [18]:
```
# Create Grid for pairwise relationships
gr = sns.PairGrid(df, height=5, hue="Churn")
gr = gr.map_diag(plt.hist)
gr = gr.map_offdiag(plt.scatter)
gr = gr.add_legend()
```

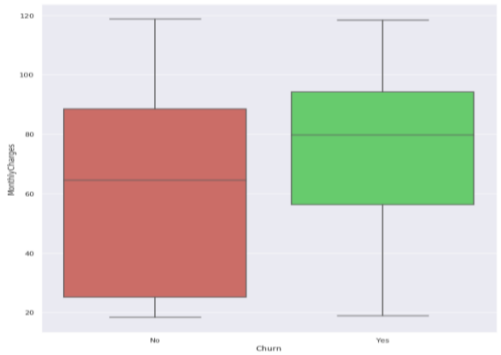


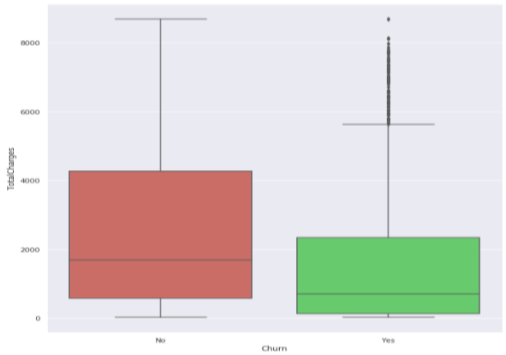


In [19]:
```
# Plot boxplots of numerical columns. More variation in the boxplot implies higher significance.
f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(25, 25))
ax = [ax1, ax2, ax3, ax4]

for i in range(len(continuous_features)):
    sns.boxplot(x = 'Churn', y = continuous_features[i], data=df, ax=ax[i])
```









2.0 Create a model

In [20]:
```
from pyspark.sql import SparkSession
import pandas as pd
import json

spark = SparkSession.builder.getOrCreate()
df_data = spark.createDataFrame(df)
df_data.head()
```

Out[20]:
```
Row(gender='Female', SeniorCitizen=0, Partner='Yes', Dependents='No', tenure=1, PhoneService='No', MultipleLines='No
phone service', InternetService='DSL', OnlineSecurity='No', OnlineBackup='Yes', DeviceProtection='No', TechSupport='N
o', StreamingTV='No', StreamingMovies='No', Contract='Month-to-month', PaperlessBilling='Yes', PaymentMethod='Electro
nic check', MonthlyCharges=29.85, TotalCharges=29.85, Churn='No')
```

2.5 Create a pipeline, and fit a model using RandomForestClassifier

```
Assemble all the stages into a pipeline. We don't expect a clean linear regression, so we'll use
RandomForestClassifier to find the best decision tree for the data.
```

In [26]:
```
classifier = RandomForestClassifier(featuresCol="features")

pipeline = Pipeline(stages=[si_gender, si_Partner, si_Dependents, si_PhoneService, si_MultipleLines, si_InternetServic
                            si_TechSupport, si_StreamingTV, si_StreamingMovies, si_Contract, si_PaperlessBilling, si_P
                            classifier, label_converter])

model = pipeline.fit(train_data)
```

In [27]:
```
predictions = model.transform(test_data)
evaluatorDT = BinaryClassificationEvaluator(rawPredictionCol="prediction")
area_under_curve = evaluatorDT.evaluate(predictions)

evaluatorDT = BinaryClassificationEvaluator(rawPredictionCol="prediction",  metricName='areaUnderROC')
area_under_curve = evaluatorDT.evaluate(predictions)
evaluatorDT = BinaryClassificationEvaluator(rawPredictionCol="prediction",  metricName='areaUnderPR')
area_under_PR = evaluatorDT.evaluate(predictions)
print("areaUnderROC = %g" % area_under_curve)
```

```
areaUnderROC = 0.709654
```

INSIGHTS FOR WEBSITE OWNERS:

The insights from the analysis can assist website owners in the following ways:

1. Identify users at risk of churning and take proactive retention measures.

2. Understand which website features or content are associated with higher retention.

3.Optimize marketing and user engagement strategies based on demographic data.

4.Monitor KPIs to assess the effectiveness of user experience improvements. - Integrate Python code for data preprocessing, feature engineering, and model development.

5.Generate actionable insights and predictions using Python scripts.