Introduction to Data Science (S2-22_DSECLZG532)-ASSIGNMENT

Group No : 03

Group Member Names:

1. AISHWARYA A
2. MINI M ALAPPATT

1. Business Understanding

Students are expected to identify an analytical problem of your choice. You have to detail the Business Understanding part of your problem under this heading which basically addresses the following questions.

1. What is the business problem that you are trying to solve?
2. What data do you need to answer the above problem?
3. What are the different sources of data?
4. What kind of analytics task are you performing?

Score: 1 Mark in total (0.25 mark each)

--------------Type the answers below this line--------------

1) Goal of churn analysis is to reduce churn and increase profits. As more customers stay longer, revenue should increase, and profits should follow.Customer churn refers to the rate at which customers stop doing business with a company. It is a critical metric for many businesses, and analyzing customer churn data can help address several business problems, including:

Retention, Segmentation, Product/Service Improvement, Cost Reduction, Predictive Analytics, Customer Lifetime Value (CLV), Competitive Analysis, Feedback Loop, Subscription-Based Models In essence, the business problem that churn data aims to solve is how to reduce customer attrition, increase customer loyalty, and ultimately improve the overall health and profitability of a business. Businesses use various analytics techniques, machine learning models, and data-driven strategies to tackle these challenges based on their specific industry and customer base.

2) To effectively address the business problem of customer churn and implement data-driven solutions, businesses typically need a range of data from various sources. Here is a list of essential data types and sources required to analyze and mitigate customer churn:

Customer Data: Customer demographics (age, gender, location). Contact information (email, phone number). Account creation date. Customer segment or classification. Historical purchasing behavior.

Transactional Data: Purchase history (what, when, how much). Usage patterns (how frequently and intensively customers use the product or service). Billing and payment information (for subscription-based models).

Customer Interaction Data: Customer support interactions (customer service calls, chat logs, email correspondence). Feedback and survey responses. NPS (Net Promoter Score) or CSAT (Customer Satisfaction Score) data.

Product or Service Data: Product or service features and attributes. Product/service performance metrics (e.g., response time, uptime, reliability). Changes or updates to the product/service.

Competitor Data: Information on competitors in the industry. Competitor pricing, offerings, and customer reviews. Market share data.

Churn Data: Churn events (when customers left or canceled their subscriptions). Reasons for churn (if available through customer feedback or exit surveys). Churn timing (e.g., seasonal patterns or after specific events).

Communication Data: Marketing and promotional campaign data (email marketing, advertising channels). Communication history (outreach attempts, response rates).

User Behavior Data (for digital products or services): Website/app usage data (clickstreams, navigation paths). Feature adoption and engagement metrics. In-app or on-site behavior data (e.g., abandoned shopping carts).

External Data: Economic indicators (e.g., unemployment rates, GDP) that may influence customer behavior. Social media sentiment analysis related to the brand or industry.

Subscription and Contract Data (for subscription-based businesses): Contract terms and renewal dates. Pricing tiers and changes. Subscription cancelation reasons.

Historical Data: Historical customer churn data to build predictive models. Past marketing and retention strategies and their outcomes. Having access to a comprehensive and well-structured dataset that combines these types of data can provide valuable insights into customer churn patterns and help businesses develop effective strategies to mitigate churn and improve customer retention.

3) Data can be sourced from various places, and the choice of data sources depends on the specific needs of a project or business. Here are different sources of data:

Internal Data:

Customer Data: Information about your customers, including demographics, contact details, and historical transaction data.

Customer Support Data: Data from customer service interactions, including chat logs, call recordings, and email correspondence.

Website and App Analytics: Data on user interactions with your website or application, including page views, clickstreams, and user behavior.

External Data:

Subscription Data: Data obtained through subscriptions to services or databases that provide specific industry or market information.

4) We can uses all the 4 analytics methods Descriptive, Predictive, Diagonastic and predictive. But majorily we focus on:

Descriptive Analytics: Describing historical data to understand what has happened in the past. Summarizing data using statistics, charts, and graphs. Providing insights into trends, patterns

Predictive analytics: Customer Churn prediction means knowing which customers are likely to leave or unsubscribe from your service. For many companies, this is an important prediction. This is because acquiring new customers often costs more than retaining existing ones. Once you've identified customers at risk of churn, you need to know exactly what marketing efforts you should make with each customer to maximize their likelihood of staying.

2. Data Acquisition

For the problem identified , find an appropriate data set (Your data set must be unique with minimum **20 features and 10k rows**) from any public data source.

2.1 Download the data directly

In [1]:

```
##---------Type the code below this line------------------##

import csv
import requests
import pandas as pd

CSV_URL = 'https://www.kaggle.com/datasets/blastchar/telco-customer-churn'


with requests.Session() as s:
    download = s.get(CSV_URL)
    decoded_content = download.content.decode('utf-8')
    cr = csv.reader(decoded_content.splitlines(), delimiter=',')
    my_list = list(cr)
    for row in my_list:
        row
```

2.2 Code for converting the above downloaded data into a dataframe

In [2]:
```
##---------Type the code below this line------------------##

import pandas as pd
churn_data=pd.read_csv('customer_churn1.csv')
churn_data
```

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0.0	Yes	No	1.0	No	No phone service
1	5575-GNVDE	Male	0.0	No	No	34.0	Yes	No
2	3668-QPYBK	Male	0.0	No	No	2.0	Yes	No
3	7795-CFOCW	Male	0.0	No	No	45.0	No	No phone service
4	9237-HQITU	Female	0.0	No	No	2.0	Yes	No
...	...	...	...	...	...	...	...	...
7043	6840-RESVB	Male	0.0	Yes	Yes	NaN	Yes	Yes
7044	2234-XADUH	Female	NaN	Yes	Yes	72.0	Yes	Yes
7045	4801-JZAZL	Female	0.0	Yes	Yes	11.0	No	No phone service
7046	8361-LTMKD	Male	1.0	Yes	No	4.0	Yes	Yes
7047	3186-AJIEK	Male	0.0	No	No	66.0	Yes	No

7048 rows × 21 columns

2.3 Confirm the data has been correctly by displaying the first 5 and last 5 records.

In [3]:
```
##---------Type the code below this line------------------##
```

```
churn_data.head()                      #Display first 5 records
```

Out[3]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Inte
0	7590-VHVEG	Female	0.0	Yes	No	1.0	No	No phone service	
1	5575-GNVDE	Male	0.0	No	No	34.0	Yes	No	
2	3668-QPYBK	Male	0.0	No	No	2.0	Yes	No	
3	7795-CFOCW	Male	0.0	No	No	45.0	No	No phone service	
4	9237-HQITU	Female	0.0	No	No	2.0	Yes	No	

5 rows × 21 columns

In [4]:

```
##---------Type the code below this line------------------##

churn_data.tail()                      #Display last 5 records
```

Out[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
7043	6840-RESVB	Male	0.0	Yes	Yes	NaN	Yes	Yes
7044	2234-XADUH	Female	NaN	Yes	Yes	72.0	Yes	Yes
7045	4801-JZAZL	Female	0.0	Yes	Yes	11.0	No	No phone service
7046	8361-LTMKD	Male	1.0	Yes	No	4.0	Yes	Yes
7047	3186-AJIEK	Male	0.0	No	No	66.0	Yes	No

5 rows × 21 columns

2.4 Display the column headings, statistical information, description and statistical summary of the data.

In [5]:
```python
##---------Type the code below this line------------------##

churn_data.columns.values            #Column Headings
```

Out[5]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

In [6]:
```python
##---------Type the code below this line------------------##

churn_data.describe()                #Data Description
```

Out[6]:

	SeniorCitizen	tenure	MonthlyCharges
count	7047.000000	7047.000000	7047.000000
mean	0.162197	32.374486	66.185760
std	0.368657	24.563960	122.976256
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	10074.400000

In [7]:
```python
##---------Type the code below this line------------------##

churn_data.info()              #Column details
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7048 entries, 0 to 7047
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7048 non-null   object 
 1   gender            7048 non-null   object 
 2   SeniorCitizen     7047 non-null   float64
 3   Partner           7048 non-null   object 
 4   Dependents        7048 non-null   object 
 5   tenure            7047 non-null   float64
 6   PhoneService      7048 non-null   object 
 7   MultipleLines     7048 non-null   object 
 8   InternetService   7048 non-null   object 
 9   OnlineSecurity    7048 non-null   object 
 10  OnlineBackup      7047 non-null   object 
 11  DeviceProtection  7047 non-null   object 
 12  TechSupport       7047 non-null   object 
 13  StreamingTV       7048 non-null   object 
 14  StreamingMovies   7047 non-null   object 
 15  Contract          7047 non-null   object 
 16  PaperlessBilling  7048 non-null   object 
 17  PaymentMethod     7048 non-null   object 
 18  MonthlyCharges    7047 non-null   float64
 19  TotalCharges      7048 non-null   object 
 20  Churn             7047 non-null   object 
dtypes: float64(3), object(18)
memory usage: 1.1+ MB
```

In [8]:
```
##---------Type the code below this line------------------##

churn_data.shape                    #Size of dataset
```

Out[8]:
```
(7048, 21)
```

2.5 Write your observations from the above.

1. Size of the dataset
2. What type of data attributes are there?
3. Is there any null data that has to be cleaned?

Score: 2 Marks in total (0.25 marks for 2.1, 0.25 marks for 2.2, 0.5 marks for 2.3, 0.25 marks for 2.4, 0.75 marks for 2.5)

--------------Type the answers below this line--------------

1. Size of data = 7048*21
2. [(Qualitative Categorical Nominal:'customerID') (Binary Symmetric:'gender') (Binary Asymmetric: 'SeniorCitizen' 'Partner' 'Dependents' 'PaperlessBilling' 'Churn') (Quantitative Numerical Ratio 'tenure') (Binary Asymmetric:'PhoneService') (Qualitative Categorical Nominal:'MultipleLines' 'InternetService') (Qualitative Categorical Ordinal:'OnlineSecurity' 'OnlineBackup' 'DeviceProtection' 'TechSupport' 'StreamingTV' 'StreamingMovies'

'PaymentMethod') (Qualitative Categorical Nominal:'Contract') (Quantitative Numerical Ratio:'MonthlyCharges' 'TotalCharges')]

3. Yes; inconsistency, missing, duplicates

3. Data Preparation

If input data is numerical or categorical, do 3.1, 3.2 and 3.4 If input data is text, do 3.3 and 3.4

In [9]:
```
churn_data.tail(6)   #Data to be cleansed
```

Out[9]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
7042	3186-AJIEK	Male	0.0	No	No	66.0	Yes	No
7043	6840-RESVB	Male	0.0	Yes	Yes	NaN	Yes	Yes
7044	2234-XADUH	Female	NaN	Yes	Yes	72.0	Yes	Yes
7045	4801-JZAZL	Female	0.0	Yes	Yes	11.0	No	No phone service
7046	8361-LTMKD	Male	1.0	Yes	No	4.0	Yes	Yes
7047	3186-AJIEK	Male	0.0	No	No	66.0	Yes	No

6 rows × 21 columns

3.1 Check for

- duplicate data
- missing data
- data inconsistencies

In [10]:
```
##---------Type the code below this line------------------##

churn_data.columns
duplicateRowsDF = churn_data.duplicated()
print("Duplicate data :")
print(duplicateRowsDF)                  #Index Position of duplicate
print(churn_data[duplicateRowsDF])      #Duplicate row
print("Missing data in each column :")
print(churn_data.isnull().sum())         #Missing values in each column
churn_data.fillna(-999).tail()           #Replace Null value with -999
```

```
Duplicate data :
0       False
1       False
2       False
3       False
4       False
        ...
7043    False
7044    False
7045    False
7046    False
7047     True
Length: 7048, dtype: bool
      customerID gender  SeniorCitizen Partner Dependents  tenure  \
7047  3186-AJIEK   Male            0.0      No         No    66.0

     PhoneService MultipleLines InternetService OnlineSecurity  ...  \
7047          Yes            No     Fiber optic            Yes  ...

     DeviceProtection TechSupport StreamingTV StreamingMovies  Contract  \
7047              Yes         Yes         Yes             Yes  Two year

     PaperlessBilling              PaymentMethod MonthlyCharges  TotalCharges  \
7047              Yes  Bank transfer (automatic)         105.65        6844.5

     Churn
7047    No

[1 rows x 21 columns]
Missing data in each column :
customerID          0
gender              0
SeniorCitizen       1
Partner             0
Dependents          0
tenure              1
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        1
DeviceProtection    1
TechSupport         1
StreamingTV         0
StreamingMovies     1
Contract            1
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      1
TotalCharges        0
Churn               1
dtype: int64
```

Out[10]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
7043	6840-RESVB	Male	0.0	Yes	Yes	-999.0	Yes	Yes
7044	2234-XADUH	Female	-999.0	Yes	Yes	72.0	Yes	Yes
7045	4801-JZAZL	Female	0.0	Yes	Yes	11.0	No	No phone service
7046	8361-LTMKD	Male	1.0	Yes	No	4.0	Yes	Yes
7047	3186-AJIEK	Male	0.0	No	No	66.0	Yes	No

3.2 Apply techiniques

- to remove duplicate data
- to impute or remove missing data
- to remove data inconsistencies

In [11]:

```
##---------Type the code below this line------------------##

clean_data=churn_data
clean_data=clean_data.drop_duplicates()
clean_data.tail()                #Removed duplicate
```

Out[11]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
7042	3186-AJIEK	Male	0.0	No	No	66.0	Yes	No
7043	6840-RESVB	Male	0.0	Yes	Yes	NaN	Yes	Yes
7044	2234-XADUH	Female	NaN	Yes	Yes	72.0	Yes	Yes
7045	4801-JZAZL	Female	0.0	Yes	Yes	11.0	No	No phone service
7046	8361-LTMKD	Male	1.0	Yes	No	4.0	Yes	Yes

5 rows × 21 columns

In [12]:

```
##---------Type the code below this line------------------##
```

```python
print("Missing data(filled with next values) :")
clean_data=clean_data.fillna(method ='bfill')                    #Filled Data
clean_data.tail()
```

Missing data(filled with next values) :

Out[12]:

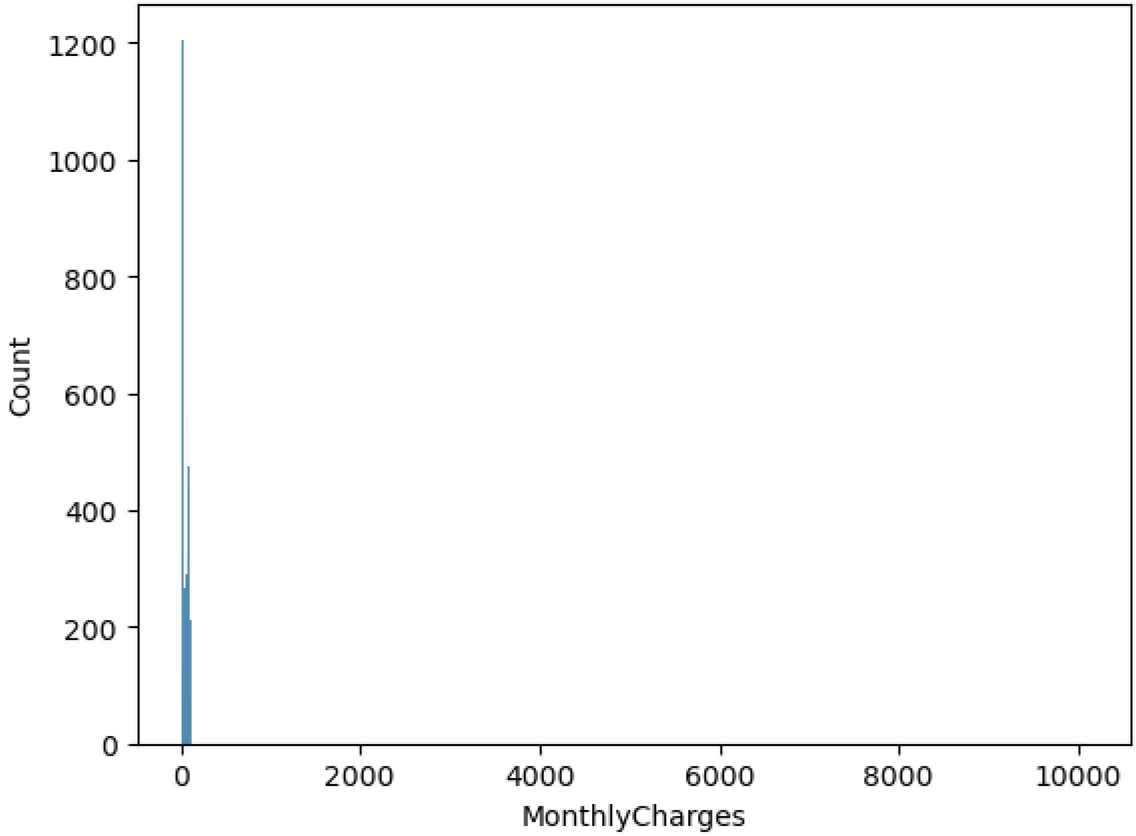
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
7042	3186-AJIEK	Male	0.0	No	No	66.0	Yes	No
7043	6840-RESVB	Male	0.0	Yes	Yes	72.0	Yes	Yes
7044	2234-XADUH	Female	0.0	Yes	Yes	72.0	Yes	Yes
7045	4801-JZAZL	Female	0.0	Yes	Yes	11.0	No	No phone service
7046	8361-LTMKD	Male	1.0	Yes	No	4.0	Yes	Yes

5 rows × 21 columns

In [13]:

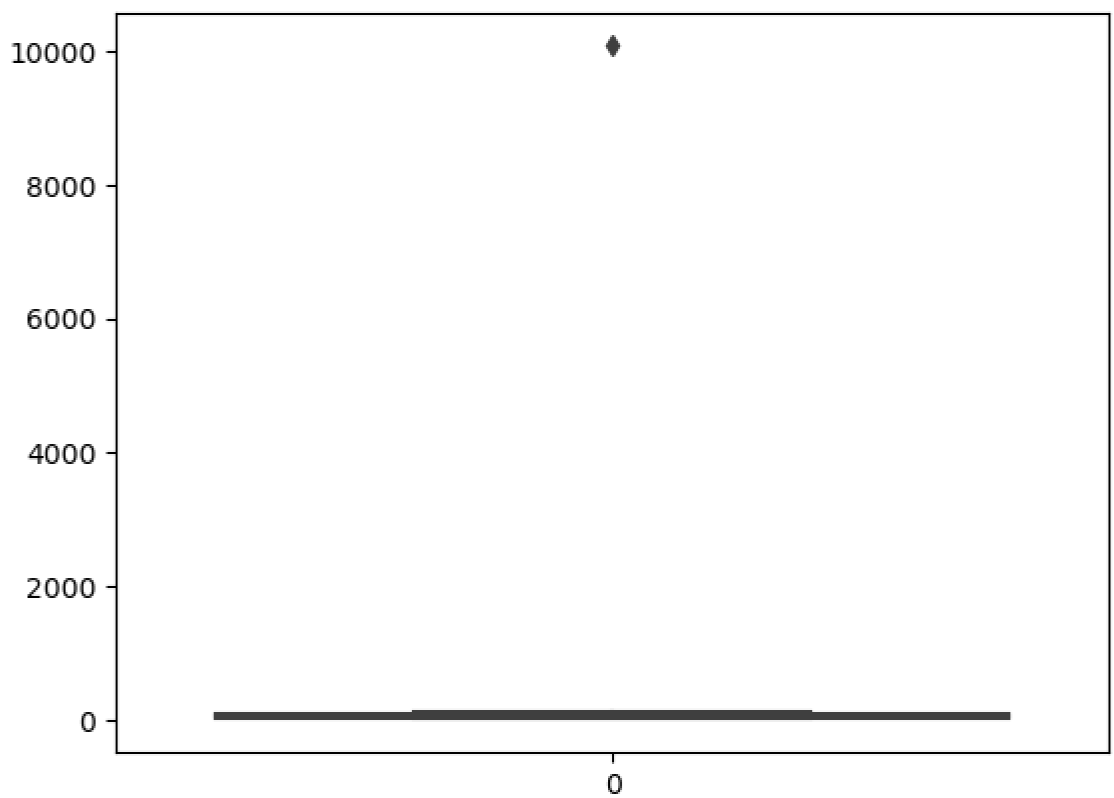
```python
import matplotlib.pyplot as plt
import seaborn as sns
sns.histplot(clean_data['MonthlyCharges'])   #plot to identify outliers
```

Out[13]: <Axes: xlabel='MonthlyCharges', ylabel='Count'>



In [14]:
```
sns.boxplot(clean_data['MonthlyCharges'])
```

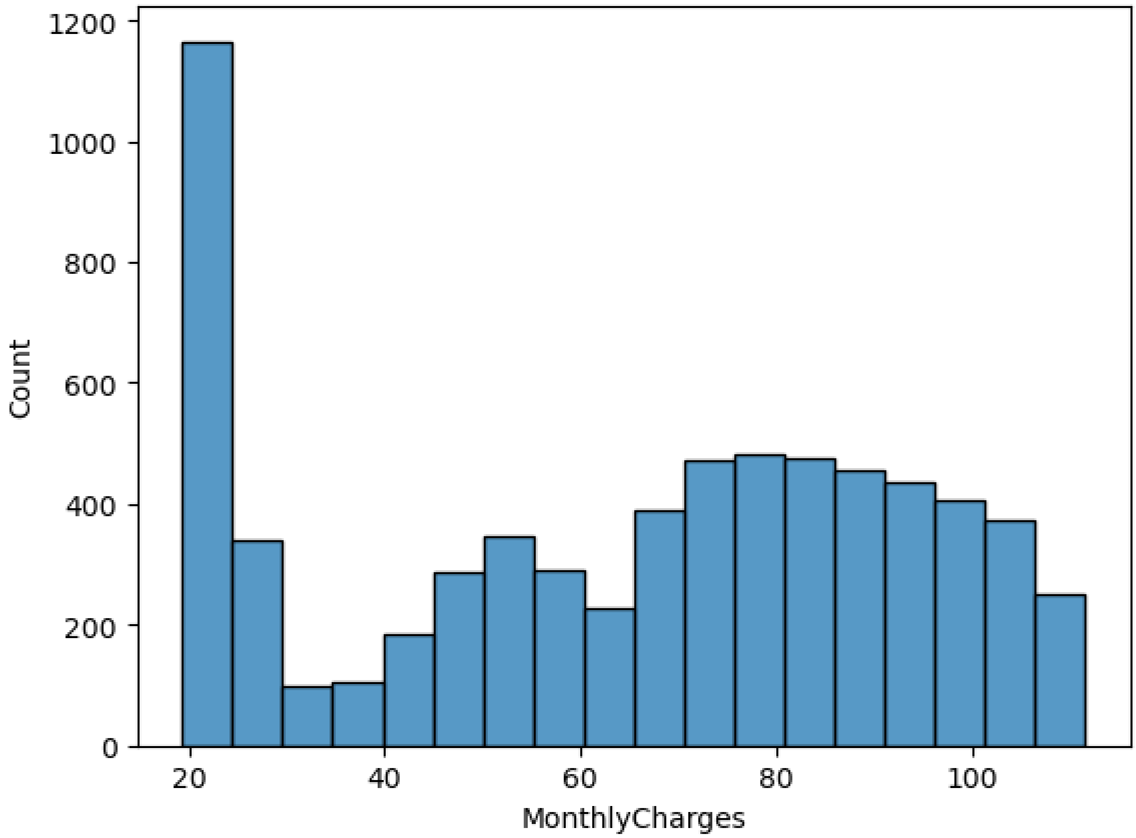
Out[14]: <Axes: >



In [46]:
```
import warnings
warnings.filterwarnings('ignore')
```

```
upper_limit = clean_data['MonthlyCharges'].quantile(0.99)        #upper limit for outli
lower_limit = clean_data['MonthlyCharges'].quantile(0.01)        #lower limit for outli
clean_data = clean_data[(churn_data['MonthlyCharges'] <= upper_limit) & (churn_data['M
sns.histplot(clean_data['MonthlyCharges'])
```

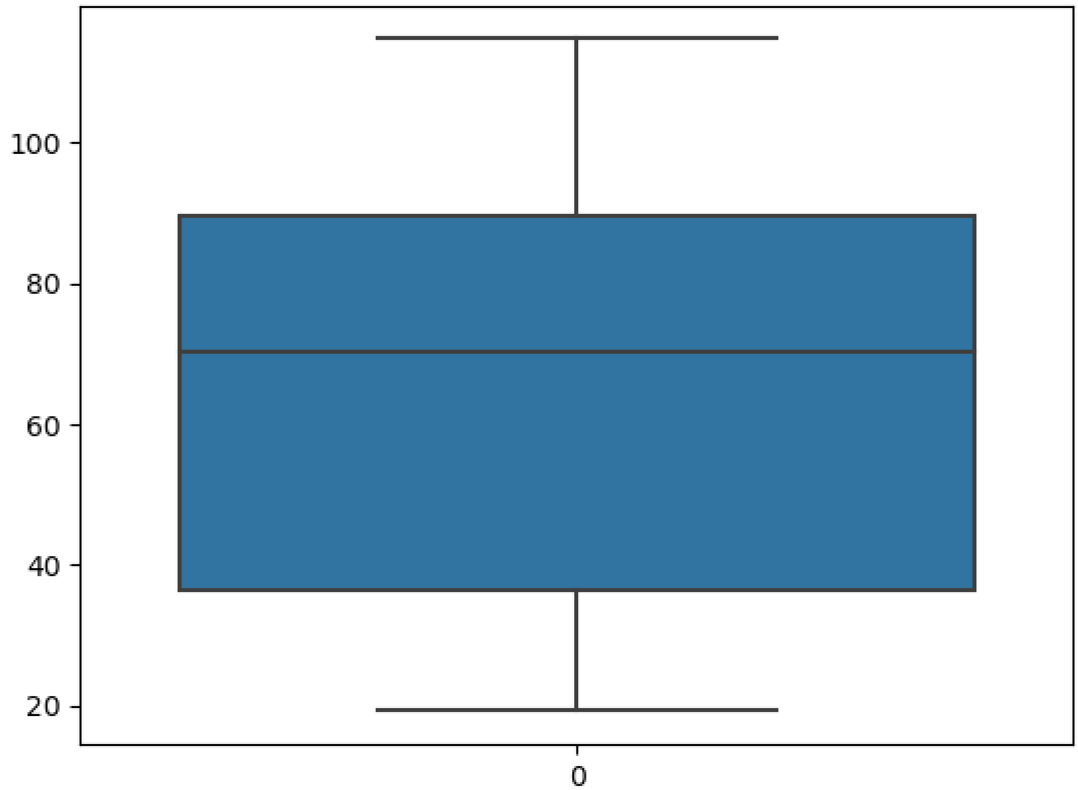
Out[46]: <Axes: xlabel='MonthlyCharges', ylabel='Count'>



In [16]:

```
sns.boxplot(clean_data['MonthlyCharges'])        #Boxplot after removing outliners
```

Out[16]: <Axes: >



3.3 Encode categorical data

In [17]:
```
##---------Type the code below this line------------------##
from sklearn.preprocessing import OneHotEncoder
import warnings
warnings.filterwarnings('ignore')

one_hot_encoded_data = pd.get_dummies(clean_data, columns = ['gender', 'Partner','Depe
one_hot_encoded_data

# Converting type of columns to category
clean_data['gender'] = clean_data['gender'].astype('category')
clean_data['Partner'] = clean_data['Partner'].astype('category')
clean_data['Dependents'] = clean_data['Dependents'].astype('category')
clean_data['PhoneService'] = clean_data['PhoneService'].astype('category')
clean_data['MultipleLines'] = clean_data['MultipleLines'].astype('category')
clean_data['InternetService'] = clean_data['InternetService'].astype('category')
clean_data['OnlineSecurity'] = clean_data['OnlineSecurity'].astype('category')
clean_data['OnlineBackup'] = clean_data['OnlineBackup'].astype('category')
clean_data['DeviceProtection'] = clean_data['DeviceProtection'].astype('category')
clean_data['TechSupport'] = clean_data['TechSupport'].astype('category')
clean_data['StreamingTV'] = clean_data['StreamingTV'].astype('category')
clean_data['StreamingMovies'] = clean_data['StreamingMovies'].astype('category')
clean_data['Contract'] = clean_data['Contract'].astype('category')
clean_data['PaperlessBilling'] = clean_data['PaperlessBilling'].astype('category')
clean_data['PaymentMethod'] = clean_data['PaymentMethod'].astype('category')


# Assigning numerical values and storing it in another columns
clean_data['gender'] = clean_data['gender'].cat.codes
clean_data['Partner'] = clean_data['Partner'].cat.codes
clean_data['Dependents'] = clean_data['Dependents'].cat.codes
```

```
clean_data['PhoneService'] = clean_data['PhoneService'].cat.codes
clean_data['MultipleLines'] = clean_data['MultipleLines'].cat.codes
clean_data['InternetService'] = clean_data['InternetService'].cat.codes
clean_data['OnlineSecurity'] = clean_data['OnlineSecurity'].cat.codes
clean_data['OnlineBackup'] = clean_data['OnlineBackup'].cat.codes
clean_data['DeviceProtection'] = clean_data['DeviceProtection'].cat.codes
clean_data['TechSupport'] = clean_data['TechSupport'].cat.codes
clean_data['StreamingTV'] = clean_data['StreamingTV'].cat.codes
clean_data['StreamingMovies'] = clean_data['StreamingMovies'].cat.codes
clean_data['Contract'] = clean_data['Contract'].cat.codes
clean_data['PaperlessBilling'] = clean_data['PaperlessBilling'].cat.codes
clean_data['PaymentMethod'] = clean_data['PaymentMethod'].cat.codes


# Create an instance of One-hot-encoder
enc = OneHotEncoder()

# Passing encoded columns
enc_data = pd.DataFrame(enc.fit_transform(
    clean_data[['gender', 'Partner','Dependents','PhoneService','MultipleLines','Inter

# One Hot Encoded Data
clean_data
```

Out[17]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	0	0.0	1	0	1.0	0	1
1	5575-GNVDE	1	0.0	0	0	34.0	1	0
2	3668-QPYBK	1	0.0	0	0	2.0	1	0
3	7795-CFOCW	1	0.0	0	0	45.0	0	1
4	9237-HQITU	0	0.0	0	0	2.0	1	0
...	...	...	...	...	...	...	...	...
7040	4801-JZAZL	0	0.0	1	1	11.0	0	1
7041	8361-LTMKD	1	1.0	1	0	4.0	1	2
7042	3186-AJIEK	1	0.0	0	0	66.0	1	0
7043	6840-RESVB	1	0.0	1	1	72.0	1	2
7045	4801-JZAZL	0	0.0	1	1	11.0	0	1

6910 rows × 21 columns

3.4 Text data

1. Remove special characters
2. Change the case (up-casing and down-casing).
3. Tokenization — process of discretizing words within a document.
4. Filter Stop Words.

In [18]:
```
churn_data = churn_data.replace(r'[^0-9a-zA-Z ]', '', regex=True).replace("'", '')
churn_data
```

Out[18]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590VHVEG	Female	0.0	Yes	No	1.0	No	No phone service
1	5575GNVDE	Male	0.0	No	No	34.0	Yes	No
2	3668QPYBK	Male	0.0	No	No	2.0	Yes	No
3	7795CFOCW	Male	0.0	No	No	45.0	No	No phone service
4	9237HQITU	Female	0.0	No	No	2.0	Yes	No
...	...	...	...	...	...	...	...	...
7043	6840RESVB	Male	0.0	Yes	Yes	NaN	Yes	Yes
7044	2234XADUH	Female	NaN	Yes	Yes	72.0	Yes	Yes
7045	4801JZAZL	Female	0.0	Yes	Yes	11.0	No	No phone service
7046	8361LTMKD	Male	1.0	Yes	No	4.0	Yes	Yes
7047	3186AJIEK	Male	0.0	No	No	66.0	Yes	No

7048 rows × 21 columns

In [19]:
```
##---------Type the code below this line------------------##

# converting and overwriting values in column
churn_data["Churn"]= churn_data["Churn"].str.upper()             #convert Chu
churn_data["PaymentMethod"]= churn_data["PaymentMethod"].str.lower()      #convert Pay
churn_data.head()
```

Out[19]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Int
0	7590VHVEG	Female	0.0	Yes	No	1.0	No	No phone service	
1	5575GNVDE	Male	0.0	No	No	34.0	Yes	No	
2	3668QPYBK	Male	0.0	No	No	2.0	Yes	No	
3	7795CFOCW	Male	0.0	No	No	45.0	No	No phone service	
4	9237HQITU	Female	0.0	No	No	2.0	Yes	No	

5 rows × 21 columns

In [20]:

```
##---------Type the code below this line------------------##

import nltk
from nltk import sent_tokenize, word_tokenize
#nltk.download('punkt');
def tokenize(column):
    tokens = nltk.word_tokenize(column)
    return [w for w in tokens if w.isalpha()]
churn_data['tokenized'] = churn_data.apply(lambda x: tokenize(x['PaymentMethod']), axi
churn_data[['tokenized']].head()
```

Out[20]:

	tokenized
0	[electronic, check]
1	[mailed, check]
2	[mailed, check]
3	[bank, transfer, automatic]
4	[electronic, check]

In [21]:

```
import nltk
#nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words ='automatic' #set stopwords as automatic
churn_data['without_stopwords'] = churn_data['tokenized'].apply(lambda x: ' '.join([wo
churn_data['without_stopwords'].head()
```

Out[21]:

```
0    electronic check
1        mailed check
2        mailed check
3       bank transfer
4    electronic check
Name: without_stopwords, dtype: object
```

3.4 Report

Mention and justify the method adopted

- to remove duplicate data, if present
- to impute or remove missing data, if present
- to remove data inconsistencies, if present

OR for textdata

- How many tokens after step 3?
- how may tokens after stop words filtering?

If the any of the above are not present, then also add in the report below.

Score: 2 Marks (based on the dataset you have, the data prepreation you had to do and report typed, marks will be distributed between 3.1, 3.2, 3.3 and 3.4)

---------Type the answer below this line---------

Drop duplicate method was used to remove the duplicate data, our data set contained one such row Missing data was replaced by -999 to indicate they were missed and to be corrected when accurate data is found Data inconsistency happens due to manual error, this can be corrected when the accurate data is found Outliers can be treated in different ways, such as trimming, capping, discretization, or by treating them as missing value Percentile Method This technique works by setting a particular threshold value.While removing the outliers using capping, this particular method is known as Winsorization. Here, we always maintain symmetry on both sides, meaning if we remove 1% from the right, the left will also drop by 1%.

---------Type the answer below this line---------

Tokenization was applied on column contains string(PaymentMethod) 3 tokens [bank, transfer, automatic] was reduced to 2 tokens [bank, transfer]

3.5 Identify the target variables.

- Separate the data from the target such that the dataset is in the form of (X,y) or (Features, Label)

- Discretize / Encode the target variable or perform one-hot encoding on the target or any other as and if required.

- Report the observations

Score: 1 Mark

In [22]:
```
##---------Type the code below this line------------------##
```

```
y=clean_data['Churn'].value_counts().keys().tolist()                  #Churn is the tar
y
```

Out[22]: ['No', 'Yes']

In [23]:
```
##---------Type the code below this line------------------##

x=clean_data['Churn'].value_counts().tolist()
x
```

Out[23]: [5051, 1859]

In [24]:
```
clean_data['Churn'] = clean_data['Churn'].astype('category')

clean_data['Churn'] = clean_data['Churn'].cat.codes
enc = OneHotEncoder()                                                          #Perform
enc_data = pd.DataFrame(enc.fit_transform(clean_data[['Churn']]).toarray())
clean_data
```

Out[24]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	0	0.0	1	0	1.0	0	1
1	5575-GNVDE	1	0.0	0	0	34.0	1	0
2	3668-QPYBK	1	0.0	0	0	2.0	1	0
3	7795-CFOCW	1	0.0	0	0	45.0	0	1
4	9237-HQITU	0	0.0	0	0	2.0	1	0
...	...	...	...	...	...	...	...	...
7040	4801-JZAZL	0	0.0	1	1	11.0	0	1
7041	8361-LTMKD	1	1.0	1	0	4.0	1	2
7042	3186-AJIEK	1	0.0	0	0	66.0	1	0
7043	6840-RESVB	1	0.0	1	1	72.0	1	2
7045	4801-JZAZL	0	0.0	1	1	11.0	0	1

6910 rows × 21 columns

Observations: Almost 2k customers are tending towards churning

4. Data Exploration using various plots

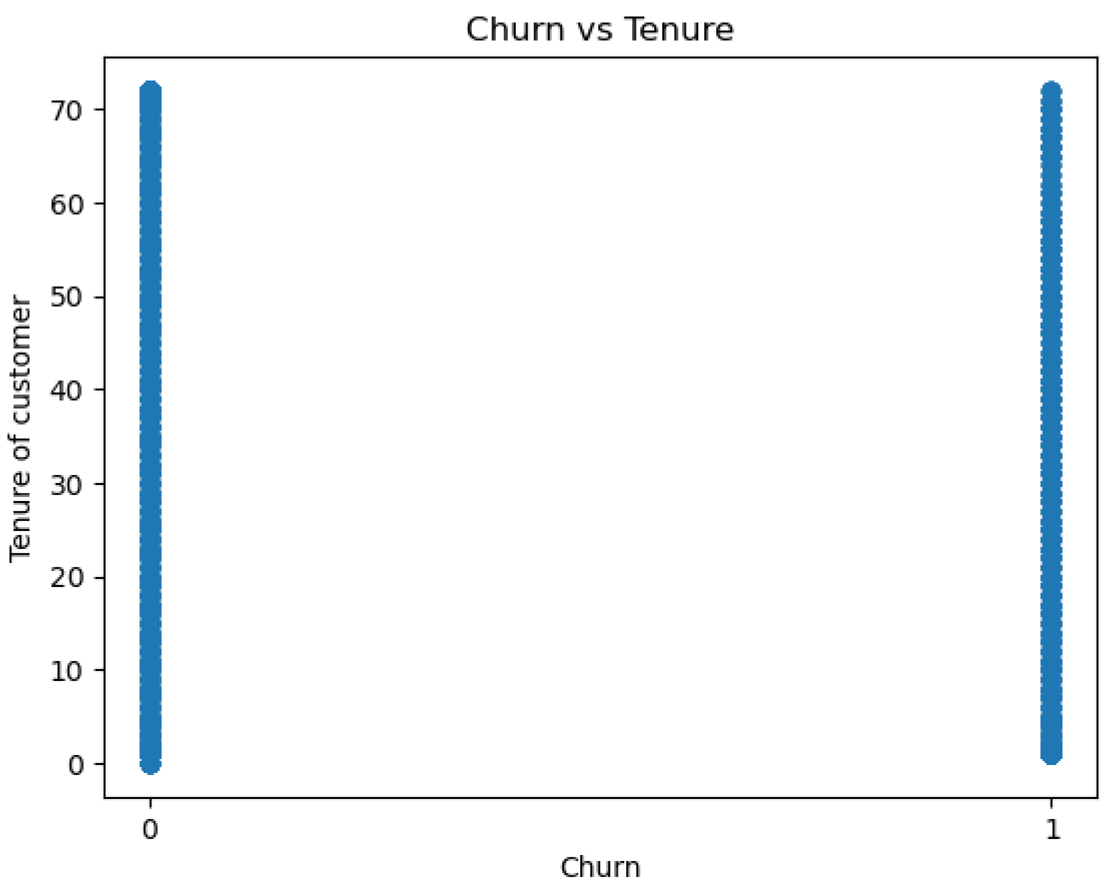
4.1 Scatter plot of each quantitative attribute with the target.

Score: 1 Mark

In [25]:

```
##---------Type the code below this line------------------##
#'Churn’ to the x-axis & ‘tenure’ to the ‘y-axis
import matplotlib.pyplot as plt


x =clean_data['Churn'].astype(str)
y =clean_data['tenure']
plt.scatter(x,y);
plt.title("Churn vs Tenure")
plt.ylabel("Tenure of customer")
plt.xlabel("Churn")
# To show the plot
plt.show()
```

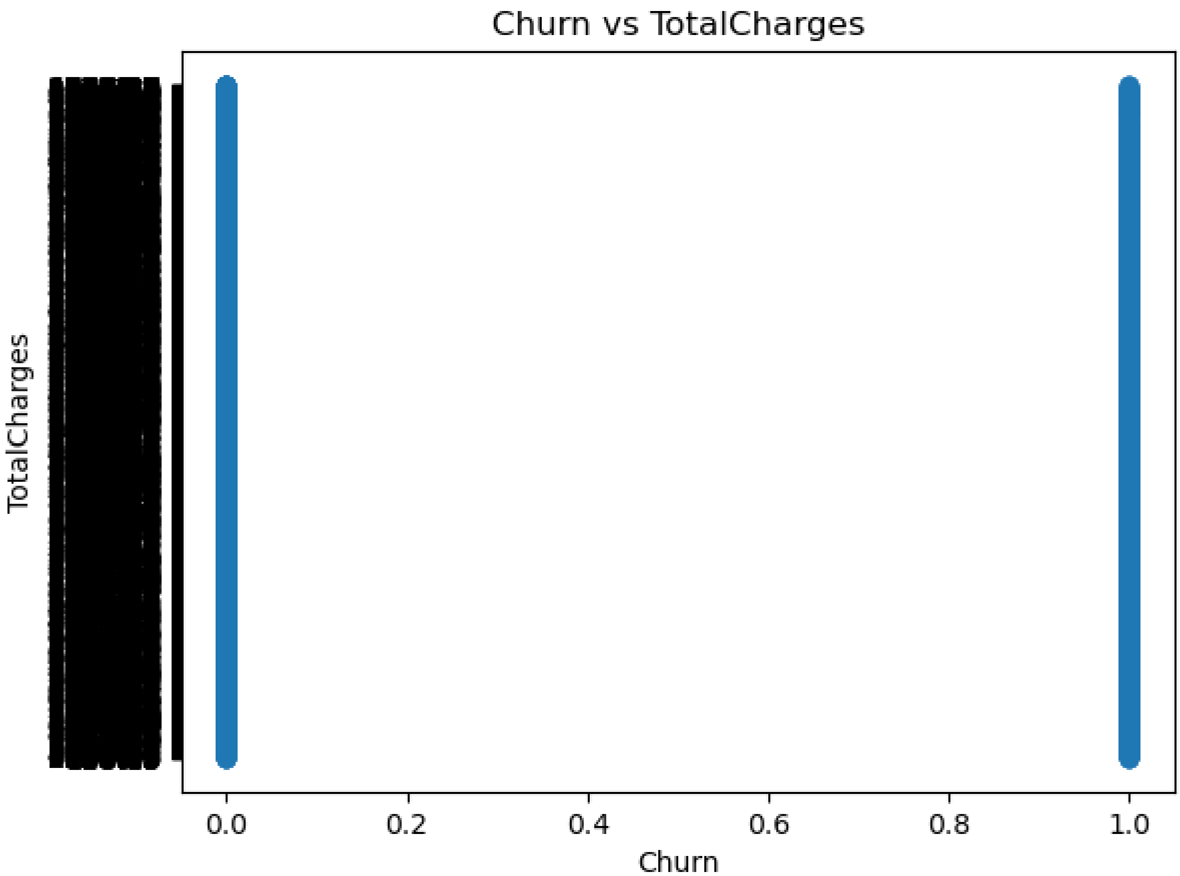


In [26]:

```
plt.scatter(clean_data['Churn'],clean_data['TotalCharges']);
plt.title("Churn vs TotalCharges")
plt.ylabel("TotalCharges")
plt.xlabel("Churn")
```

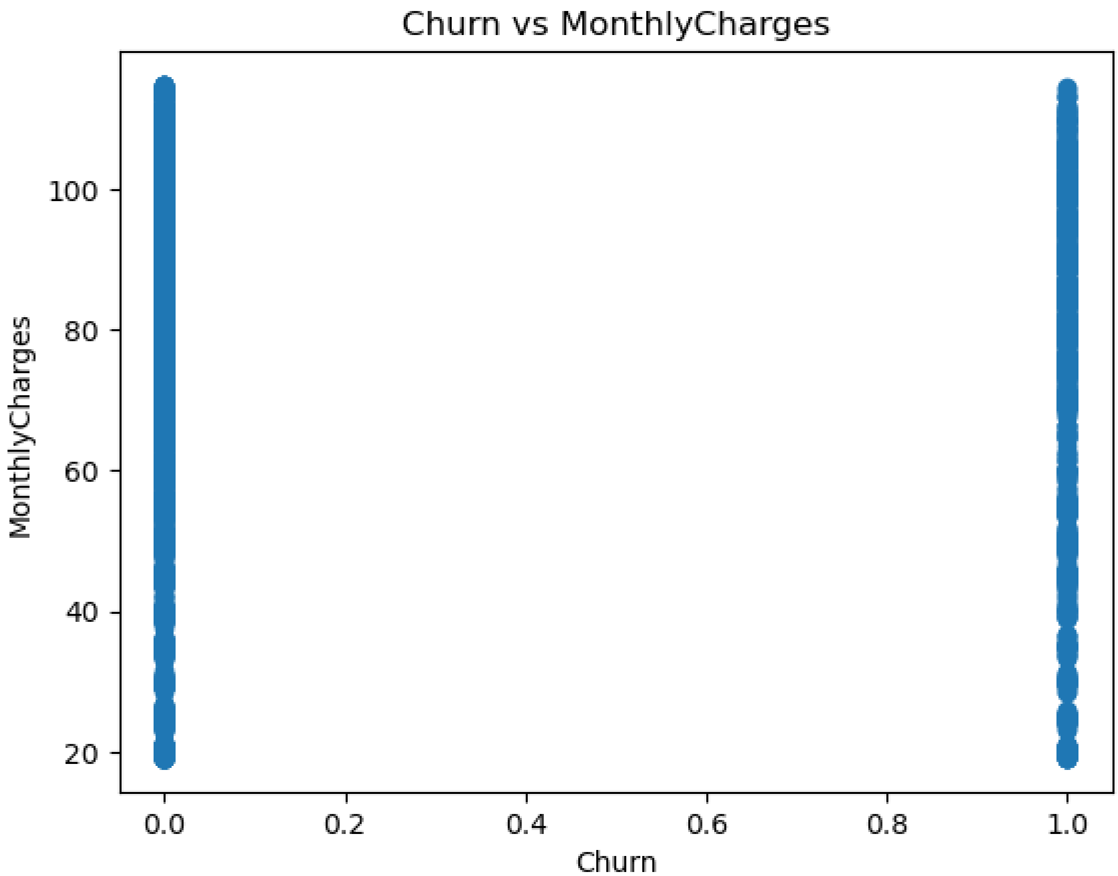
Out[26]:

```
Text(0.5, 0, 'Churn')
```



In [27]:
```
plt.scatter(clean_data['Churn'],clean_data['MonthlyCharges']);
plt.title("Churn vs MonthlyCharges")
plt.ylabel("MonthlyCharges")
plt.xlabel("Churn")
```

Out[27]:
```
Text(0.5, 0, 'Churn')
```



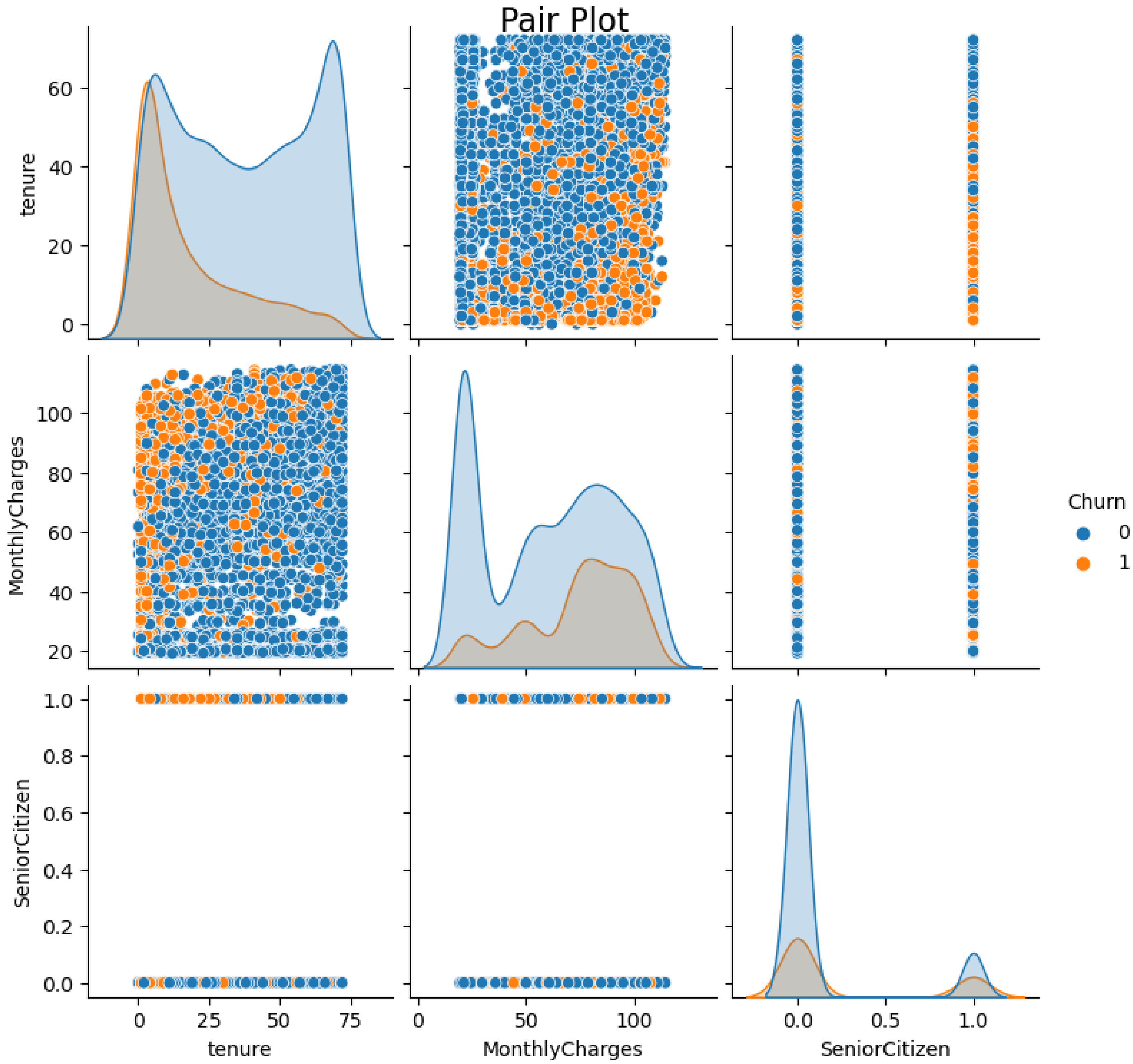
4.2 EDA using visuals

- Use (minimum) 2 plots (pair plot, heat map, correlation plot, regression plot...) to identify the optimal set of attributes that can be used for classification.
- Name them, explain why you think they can be helpful in the task and perform the plot as well. Unless proper justification for the choice of plots given, no credit will be awarded.

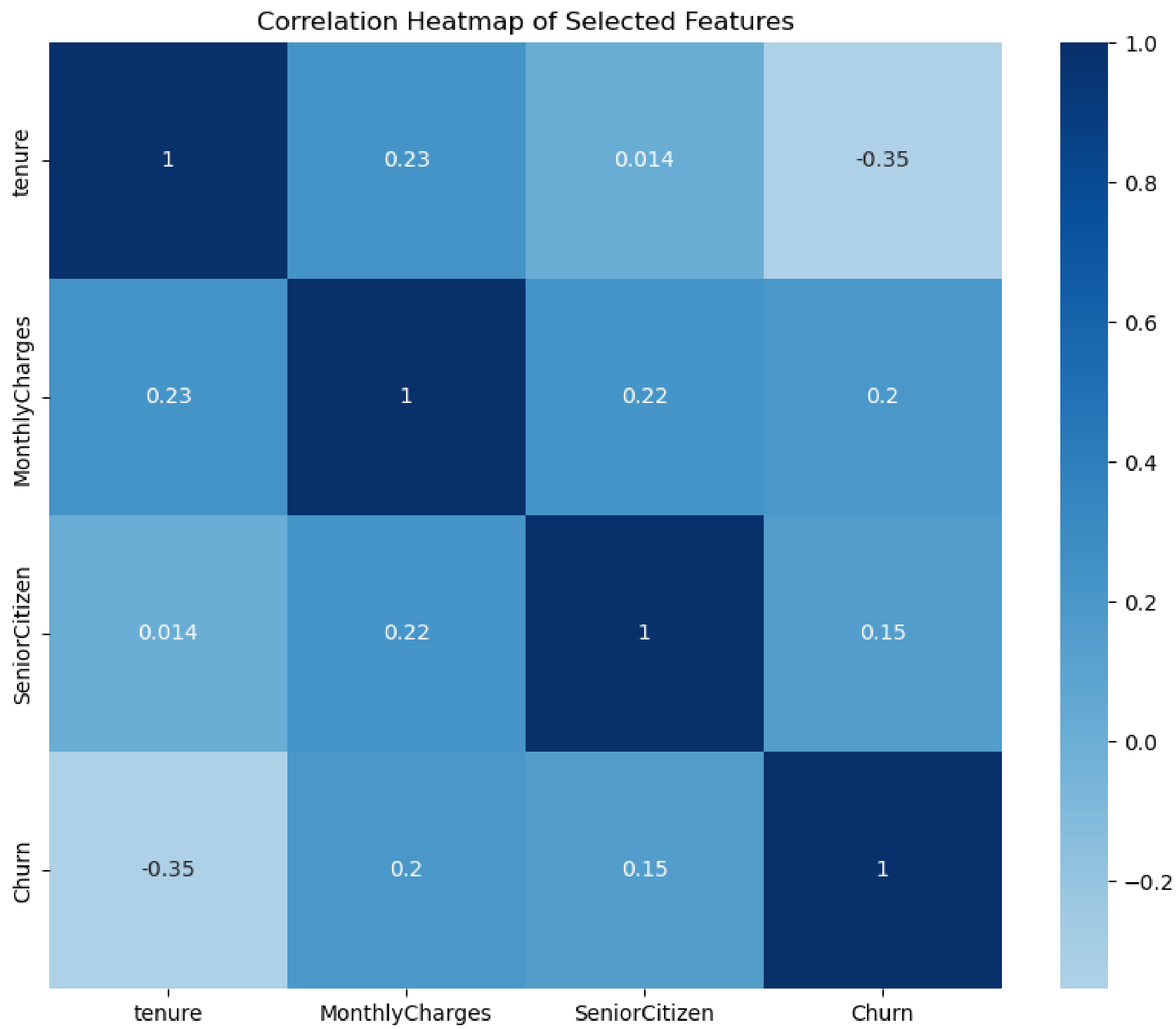
Score: 2 Marks

In [28]:
```
##---------Type the code below this line------------------##
import seaborn as sns
import matplotlib.pyplot as plt

selected_features = ['tenure', 'MonthlyCharges', 'TotalCharges', 'SeniorCitizen', 'Chu
sns.pairplot(clean_data[selected_features], hue='Churn', diag_kind='kde')
plt.suptitle('Pair Plot', y=1, fontsize = 16)
plt.show()
```



In [29]:
```
import warnings
warnings.filterwarnings('ignore')
correlation_matrix = clean_data[selected_features].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap=plt.cm.Blues, center=0)
plt.title('Correlation Heatmap of Selected Features')
plt.show()
```



SelectedFeatures'tenure', 'MonthlyCharges', 'TotalCharges', and 'SeniorCitizen' for visualization. The 'Churn' column is used to color-code the data points in the pair plot, and the correlation heatmap provides insights into the relationships between these features.

Here's what each plot can help you understand:

Pair Plot:

The diagonal shows kernel density estimates for each feature, which can provide insights into the distribution of values for churned and non-churned customers. Scatter plots show the relationships between pairs of features. Look for patterns and differences between churned and non-churned customers.

Correlation Heatmap:

The heatmap displays the correlation coefficients between the selected features. Positive values indicate positive correlation, while negative values indicate negative correlation. Features with strong correlations might be redundant, so you can consider removing some of them to avoid multicollinearity. These visualizations can help you identify relationships, trends, and potential patterns in the data. Based on what you observe, you can make informed decisions about the attributes to include in your classification model. Keep in mind that the specific features you select will depend on your dataset and the insights you're seeking.

5. Data Wrangling

5.1 Univariate Filters

Numerical and Categorical Data

- Identify top 5 significant features by evaluating each feature independently with respect to the target variable by exploring

1. Mutual Information (Information Gain)
2. Gini index
3. Gain Ratio
4. Chi-Squared test
5. Fisher Score (From the above 5 you are required to use only any **two**)

For Text data

1. Stemming / Lemmatization.
2. Forming n-grams and storing them in the document vector.
3. TF-IDF (From the above 2 you are required to use only any **two**)

Score: 3 Marks

In [30]:
```
##---------Type the code below this line------------------##
import pandas as pd
from sklearn.feature_selection import mutual_info_classif, chi2
from sklearn.preprocessing import LabelEncoder

# Load the data into a DataFrame
df =pd.read_csv('customer_churn.csv')

# Encode categorical variables using Label Encoding
label_encoder = LabelEncoder()
for col in df.select_dtypes(include=['object']):
    df[col] = label_encoder.fit_transform(df[col])

# Separate features and target variable
X = df.drop('Churn', axis=1).drop('customerID', axis=1).drop('MonthlyCharges', axis=1)
y = df['Churn']
```

In [31]:
```
# Calculate Mutual Information
mutual_info = mutual_info_classif(X, y, discrete_features='auto', random_state=42)
mi_scores = pd.Series(mutual_info, index=X.columns)
mi_scores = mi_scores.sort_values(ascending=False)
# Print top 5 significant features for each method
print("Top 5 significant features based on Mutual Information:")
print(mi_scores.head(5))
```

```
Top 5 significant features based on Mutual Information:
Contract           0.099649
TechSupport        0.076487
OnlineSecurity     0.067008
InternetService    0.056226
tenure             0.050420
dtype: float64
```

In [32]:
```
# Calculate Chi-Squared scores
chi_scores, _ = chi2(X, y)
chi_scores = pd.Series(chi_scores, index=X.columns)
chi_scores = chi_scores.sort_values(ascending=False)

# Print top 5 significant features for each method
print("\nTop 5 significant features based on Chi-Squared:")
print(chi_scores.head(5))
```

```
Top 5 significant features based on Chi-Squared:
tenure            16278.923685
Contract           1115.780167
OnlineSecurity      551.611529
TechSupport         523.303866
OnlineBackup        230.086520
dtype: float64
```

In [33]:
```
import nltk
#nltk.download('wordnet')
#nltk.download('omw-1.4')
import warnings
warnings.filterwarnings('ignore')
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
word_stemmer = PorterStemmer()               #Stemming
print("Stemming on churning:")
print(word_stemmer.stem('churning'))

lemmatizer = WordNetLemmatizer()             #Lemmantizer

print("Lemmantizing on churning:")
print(lemmatizer.lemmatize('churning'))
```

```
Stemming on churning:
churn
Lemmantizing on churning:
churning
```

In [34]:
```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline
corpus = ['this is the first churn',
           'this churn is the second churn',
          'and this is the third one',
         'is this the first churn']
vocabulary = ['this', 'churn', 'first', 'is', 'second', 'the','and', 'one']
pipe = Pipeline([('count', CountVectorizer(vocabulary=vocabulary)),
                ('tfid', TfidfTransformer())]).fit(corpus)
pipe['count'].transform(corpus).toarray()
pipe['tfid'].idf_
```

```
Out[34]: array([1.        , 1.22314355, 1.51082562, 1.        , 1.91629073,
       1.        , 1.91629073, 1.91629073])
```

5.2 Report observations

Write your observations from the results of each method. Clearly justify your choice of the method.

Score 1 mark

---------Type the code below this line------------------

Information Gain is a concept commonly used in the context of decision trees and machine learning algorithms, particularly for tasks like classification and feature selection. To apply Information Gain effectively, we typically need labeled data (data with known outcomes or classes) to evaluate the informativeness of features or variables. Here's how Information Gain can be relevant:
Decision Trees: Information Gain is commonly used in the construction of decision trees, a popular machine learning technique for classification and regression tasks. Decision trees help make decisions by recursively splitting data based on the feature that provides the most Information Gain at each step. This process helps identify the most significant features for classification or prediction. Churn Analysis: In churn analysis, we can use Information Gain to identify which factors or variables are the strongest predictors of customer churn. By focusing on these high-information-gain features, we can prioritize efforts to address those factors most likely to influence customer attrition. As seen in the results above 'Contract' is the major influencer for deciding the 'Churn'

Pearson’s chi-square (X2) tests, often referred to simply as chi-square tests, are among the most common nonparametric tests. Chi-square (χ^2) tests are statistical tests that assess the association between two categorical variables. They are commonly used in various analytics tasks, particularly when dealing with categorical data or when we need to determine if there's a statistically significant relationship between two variables. Chi-square tests come in different forms, such as the Pearson chi-square test, the chi-square test for independence, and the chi-square goodness-of-fit test. The choice of test depends on the specific research question or analytics task. It's important to note that chi-square tests are suitable for categorical data analysis but may not be appropriate for continuous or ordinal data. Additionally, the interpretation of chi-square results should consider the sample size and practical significance in addition to statistical significance. As seen in the results above 'tenure' is the major influencer for deciding the 'Churn'

PorterStemmer class chops off the ‘ing’ from the word. On the other hand, WordNetLemmatizer class finds a valid word. In simple words, stemming technique only looks at the form of the word whereas lemmatization technique looks at the meaning of the word. It means after applying lemmatization, we will always get a valid word.

tf-idf means term-frequency times inverse document-frequency. This is a common term weighting scheme in information retrieval, that has also found good use in document classification. The goal of using tf-idf instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.

6. Implement Machine Learning Techniques

Use any 2 ML algorithms

1. Classification -- Decision Tree classifier
2. Clustering -- kmeans
3. Association Analysis
4. Anomaly detection
5. Textual data -- Naive Bayes classifier (not taught in this course)

A clear justification have to be given for why a certain algorithm was chosen to address your problem.

Score: 4 Marks (2 marks each for each algorithm)

In [35]:
```
x = df.loc[:,['tenure']]
y = df.loc[:,['Churn']]
from sklearn.model_selection import train_test_split
x_train, x_test , y_train , y_test= train_test_split(x,y,test_size=0.20)    #Training-
```

6.1 ML technique 1 + Justification

Classification analysis is a data analytics technique that can be used to predict customer churn. Data analytics professionals typically use machine learning algorithms such as decision trees, and K- nearest neghbour to predict customer churn using classification analysis. These algorithms analyze data such as customer demographics, purchase history, and interactions with the company to identify patterns that can predict customer churn.

In [36]:
```
##---------Type the code below this line------------------##
from sklearn.tree import DecisionTreeClassifier
dec_tree = DecisionTreeClassifier()
dec_tree.fit(x_train, y_train)
tree_pred = dec_tree.predict(x_test)
#for comparing, tells correct prediction or  not.diagonals are correct prediction othe
from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_test,tree_pred))
print(accuracy_score(y_test,tree_pred))
```

```
[[933  73]
 [290 113]]
0.7423704755145494
```

In [37]:

```
##---------Type the code below this line------------------##
# K-Nearest Neighbor
from sklearn.neighbors import KNeighborsClassifier
knnmodel = KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
knnmodel.fit(x_train,y_train)
knn_pred=knnmodel.predict(x_test)
from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_test,knn_pred))
print(accuracy_score(y_test,knn_pred))
```

```
[[848 158]
 [240 163]]
0.7175301632363378
```

6.2 ML technique 2 + Justification

Association analysis, specifically Apriori algorithm, is a technique used to identify patterns of co-occurrence in datasets. It's commonly used for market basket analysis to find relationships between items that are frequently purchased together. In our case, we can adapt association analysis to find relationships between features and the target variable.

In this code, we're using the Apriori algorithm to find frequent item sets and association rules. We're treating each value of the selected categorical features as an item, and we're looking for associations between these items and the target variable 'Churn'.

Association rules are generated based on metrics like lift, confidence, and support. You can choose to sort the rules based on the metric that's most relevant to your analysis.association analysis may not directly rank feature significance like some other techniques, but it can help you identify interesting relationships between features and the target variable

High Confidence: The confidence value for both rules is quite high, indicating that when "Dependents" are present, there is a strong likelihood that "Partners" will also be present in the same transaction (around 82.89% confidence).

High Lift: The lift value is greater than 1 for both rules (approximately 1.716053), which suggests a positive association. This means that the presence of "Dependents" is associated with a higher likelihood of the presence of "Partners," and vice versa, compared to what would be expected if the two were independent.

Strong Conviction: The conviction values for both rules are greater than 1 (around 3.021609), indicating a strong association. Conviction measures how much more likely the consequent ("Partners") is when the antecedent ("Dependents") is present.

High Zhang's Metric: Zhang's metric, which evaluates the significance of association rules, is also relatively high for both rules (approximately 0.595746 for Rule 1 and 0.807145 for Rule 2). This suggests that the association between "Dependents" and "Partners" is significant.

In summary, the conclusion is that customers who have "Dependents" are significantly and strongly associated with also having "Partners," and vice versa. This association can have implications for business decisions, such as marketing strategies or product recommendations, as understanding these relationships can help tailor services or offers to specific customer segments more effectively.

In [45]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

# Load the data into a DataFrame
df = pd.read_csv('customer_churn.csv')

# Select categorical features for association analysis (customize as needed)
selected_features = ['gender', 'Partner', 'Dependents', 'SeniorCitizen', 'Churn']

# Convert categorical columns to binary values
for col in selected_features:
    df[col] = df[col].apply(lambda x: 1 if x == "Yes" else 0)

# Extract selected features for association analysis
X = df[selected_features]

# Find frequent item sets using Apriori
frequent_itemsets = apriori(X.drop('Churn', axis=1), min_support=0.1, use_colnames=Tru

# Generate association rules
association_rules_df = association_rules(frequent_itemsets, metric="lift", min_thresho

# Sort association rules by lift in descending order
association_rules_df = association_rules_df.sort_values(by="lift", ascending=False)

# Print top 5 significant features based on association analysis
print("Top 5 significant features based on Association Analysis:")
print(association_rules_df.head(5))
```

```
Top 5 significant features based on Association Analysis:
    antecedents   consequents  antecedent support  consequent support  \
0  (Dependents)     (Partner)            0.299588            0.483033
1     (Partner)  (Dependents)            0.483033            0.299588

    support  confidence      lift  leverage  conviction  zhangs_metric
0  0.248332    0.828910  1.716053  0.103621    3.021609       0.595746
1  0.248332    0.514109  1.716053  0.103621    1.441501       0.807145
```

In [39]:
```
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
import numpy as np
kmmodel = KMeans(n_clusters=2, random_state=42)

kmmodel.fit(x_train)
# Calculate the distances of each point in the test set to the nearest centroid
distances = cdist(x_test, kmmodel.cluster_centers_)

# Calculate the minimum distance for each point in the test set
min_distances = np.min(distances, axis=1)
```

```
# Set a threshold for the minimum distance to classify transactions as normal
threshold = np.percentile(min_distances, 99.9)

# Classify transactions as normal or anomalous
km_pred = (min_distances < threshold).astype(int)
print(confusion_matrix(y_test,km_pred))
print(accuracy_score(y_test,km_pred))
```

```
[[  9 997]
 [  4 399]]
0.2895670688431512
```

Goal of clustering is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. We can see that Kmean clustering cannot be applied on this dataset the accuracy is very low.

7. Conclusion

Compare the performance of the ML techniques used.

Derive values for preformance study metrics like accuracy, precision, recall, F1 Score, AUC-ROC etc to compare the ML algos and plot them. A proper comparision based on different metrics should be done and not just accuracy alone, only then the comparision becomes authentic. You may use Confusion matrix, classification report, Word cloud etc as per the requirement of your application/problem.

Score 1 Mark

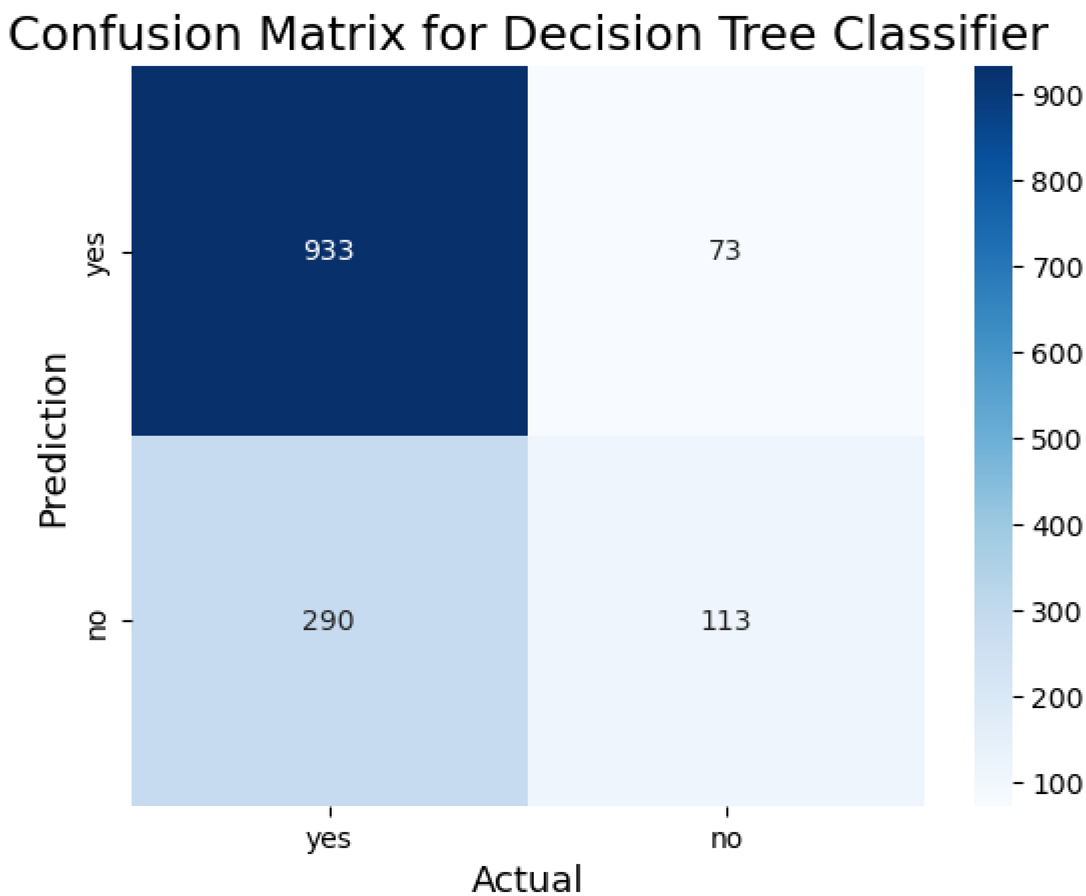
Based on accuracy score we can see the Decision tree classification is the best choice algorithm for the given dataset.

In [40]:

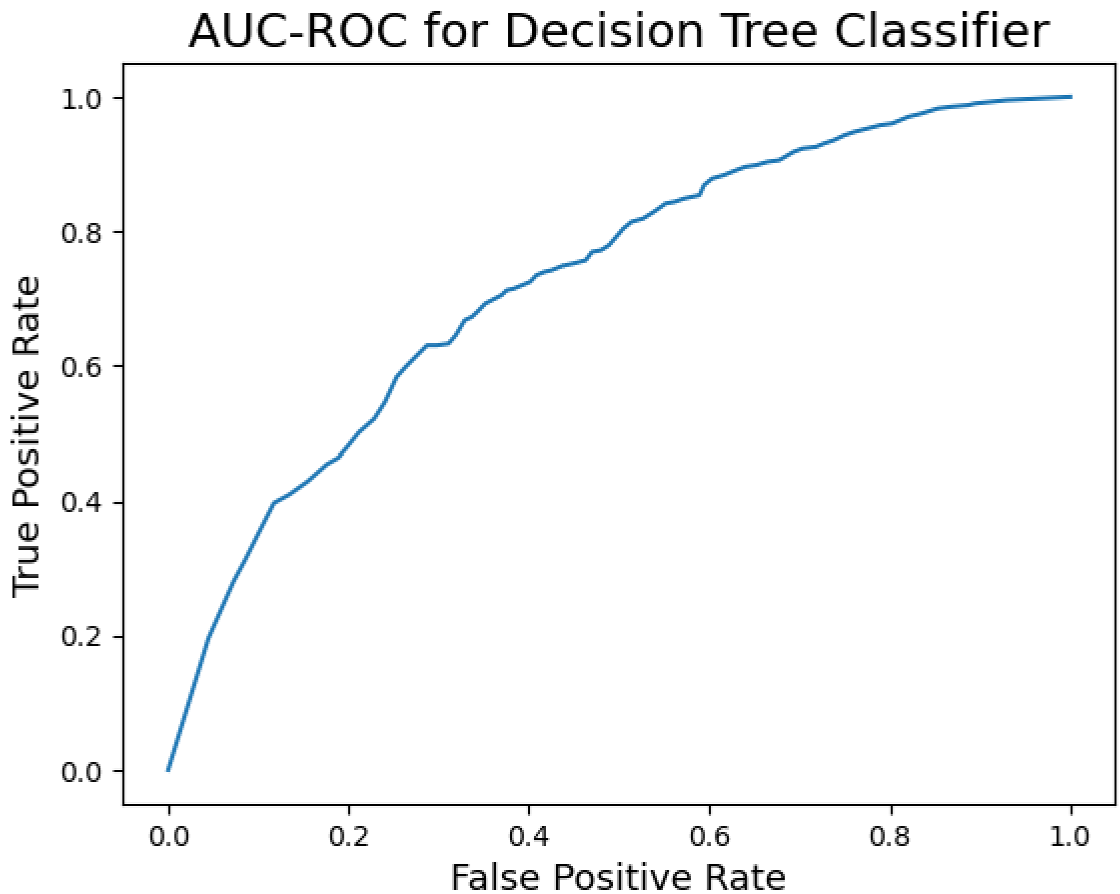
```
# Define a function to evaluate the performance of the model
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import confusion_matrix, accuracy_score
import numpy as np
from sklearn import metrics
from sklearn .metrics import roc_auc_score
def evaluate_metrics(y_test, y_pred ,model,yp_pred):
    ConfusionMatrix_tree=confusion_matrix(y_test,y_pred)
    sns.heatmap(ConfusionMatrix_tree,
            annot=True,
            fmt='g',
            xticklabels=['yes', 'no'],
            yticklabels=['yes', 'no'],cmap=plt.cm.Blues)
    plt.ylabel('Prediction',fontsize=13)
    plt.xlabel('Actual',fontsize=13)
    plt.title('Confusion Matrix for '+model,fontsize=17)
    plt.show()
    print('Performance Metrics for '+model)
    print('Accuracy:', accuracy_score(y_test, y_pred))
    print('Precision:', precision_score(y_test, y_pred))
    print('Recall:', recall_score(y_test, y_pred))
```

```
    print('F1 Score:', f1_score(y_test, y_pred))
    auc = np.round(roc_auc_score(y_test,y_pred), 3)
    print("Auc is {}".format(auc))
    fpr, tpr, _ = metrics.roc_curve(y_test, yp_pred)
    plt.plot(fpr,tpr)
    plt.ylabel('True Positive Rate',fontsize=13)
    plt.xlabel('False Positive Rate',fontsize=13)
    plt.title('AUC-ROC for '+model,fontsize=17)
    plt.show()
```

In [41]:
```
##---------Type the code below this line------------------##
#Decision tree Related metrics'
treey_pred = dec_tree.predict_proba(x_test)[:, 1]
evaluate_metrics(y_test,tree_pred,'Decision Tree Classifier',treey_pred)
```

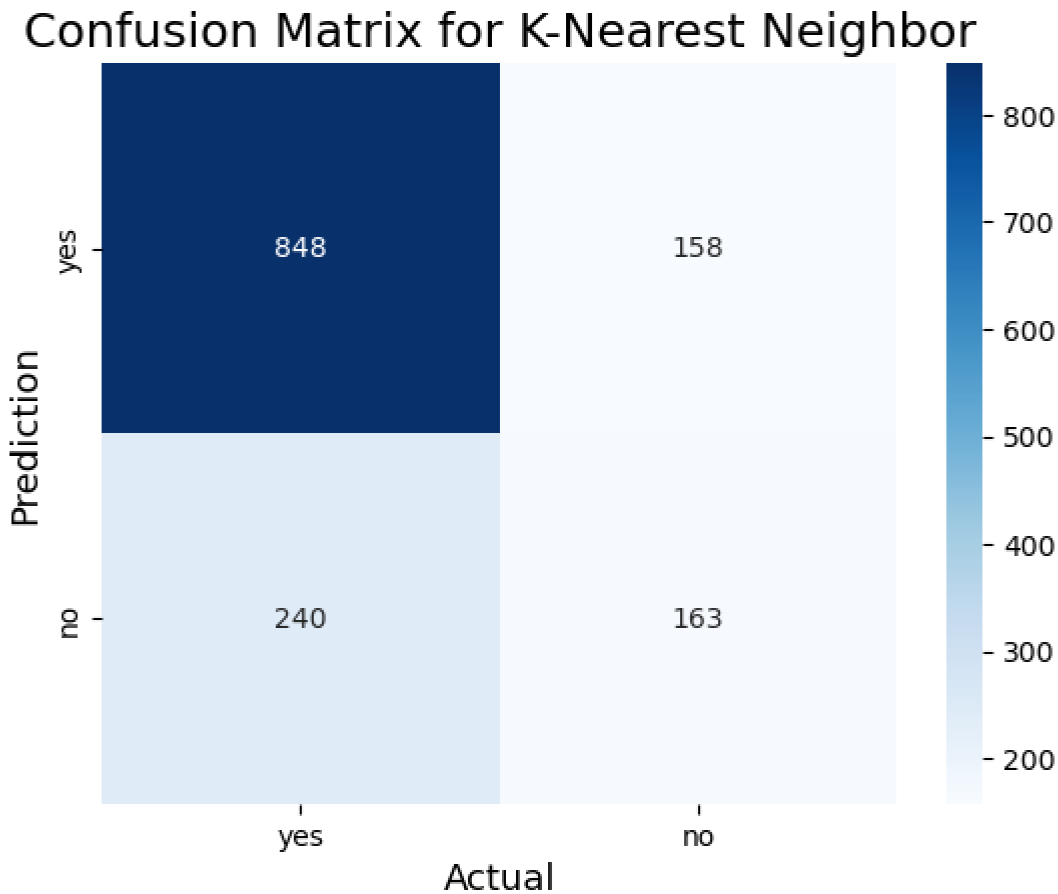



```
Performance Metrics for Decision Tree Classifier
Accuracy: 0.7423704755145494
Precision: 0.6075268817204301
Recall: 0.2803970223325062
F1 Score: 0.3837011884550085
Auc is 0.604
```

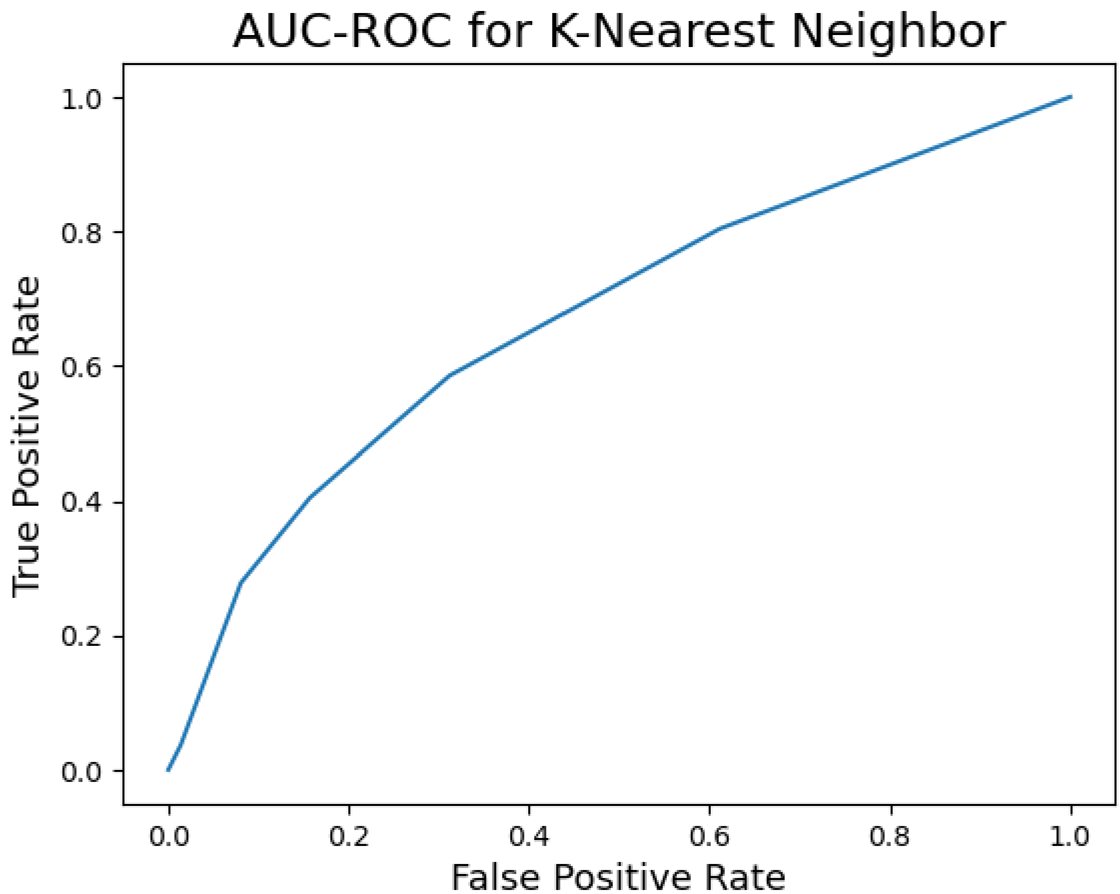


In [42]:

```
# Knn Related metrics
knny_pred=knnmodel.predict_proba(x_test)[:, 1]
evaluate_metrics(y_test,knn_pred,'K-Nearest Neighbor',knny_pred)
```

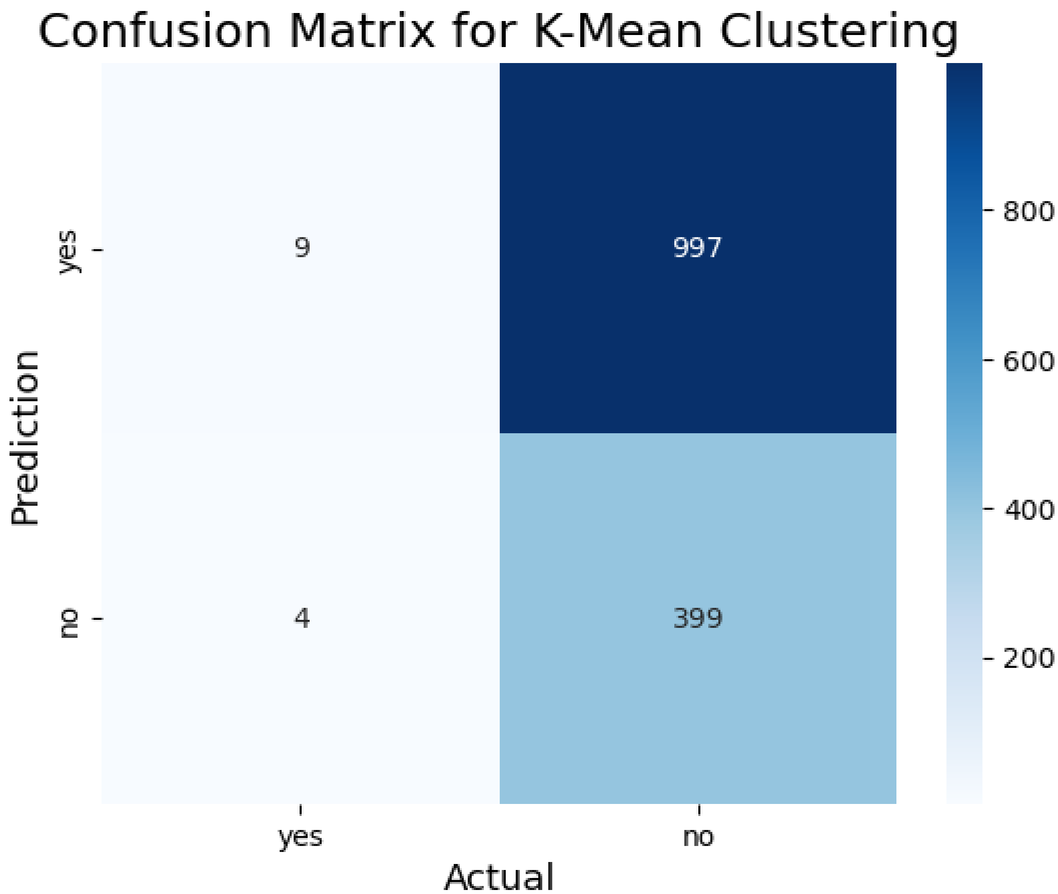



```
Performance Metrics for K-Nearest Neighbor
Accuracy: 0.7175301632363378
Precision: 0.5077881619937694
Recall: 0.4044665012406948
F1 Score: 0.45027624309392267
Auc is 0.624
```

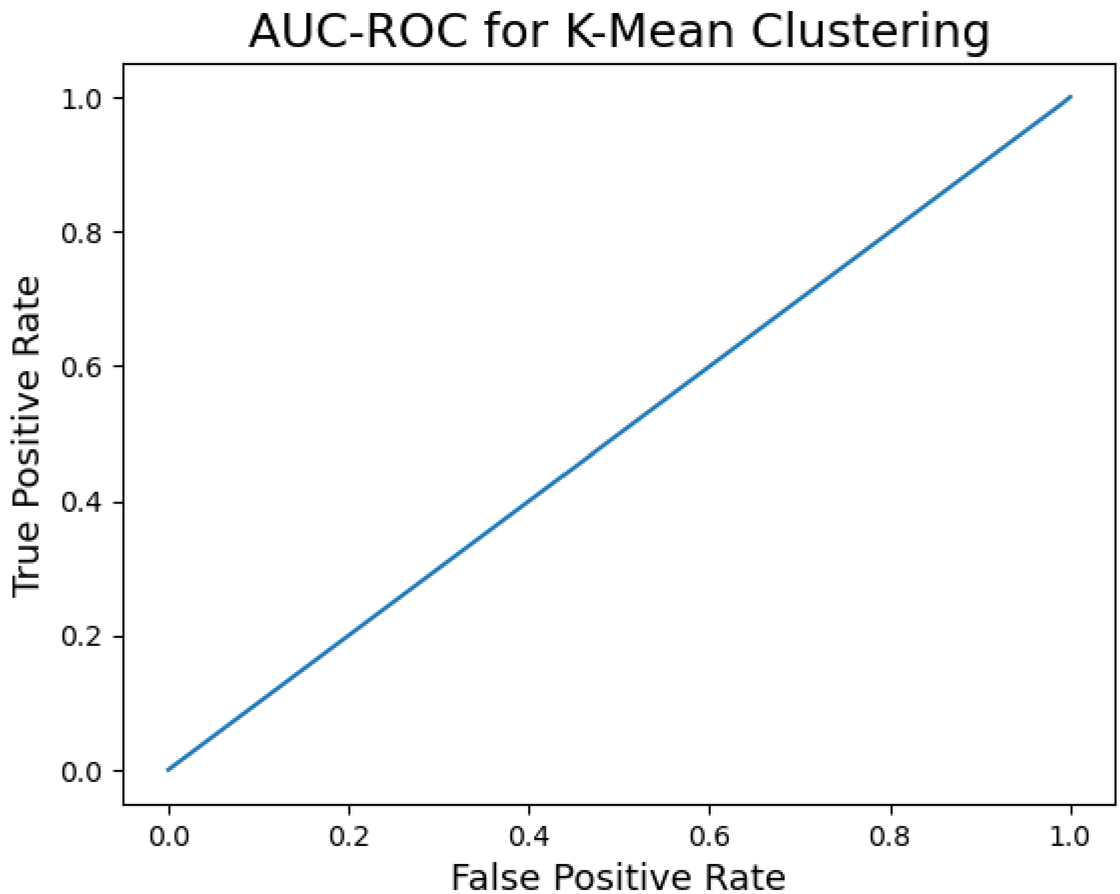


In [43]:

```
# K-mean Related metrics
#kmy_pred=kmmodel.predict_proba(x_test)[:, 1]
evaluate_metrics(y_test,km_pred,'K-Mean Clustering',km_pred)
```




```
Performance Metrics for K-Mean Clustering
Accuracy: 0.2895670688431512
Precision: 0.28581661891117477
Recall: 0.9900744416873449
F1 Score: 0.443579766536965
Auc is 0.5
```



In [44]:

```
model_comparison = pd.DataFrame ({
     'Model_Name': ['Decision Tree Classification','K-Nearest Neighbor','K-Mean Cluste
     'Accuracy Score': [accuracy_score(y_test, tree_pred)*100,accuracy_score(y_test, k
                        accuracy_score(y_test, km_pred)*100],
     'Recall Score': [recall_score(y_test, tree_pred)*100,recall_score(y_test, knn_pre
                      recall_score(y_test,km_pred)*100],
     'Precision Score': [precision_score(y_test, tree_pred)*100,precision_score(y_test
                         precision_score(y_test, km_pred)*100],
     'F1 Score': [f1_score(y_test, tree_pred)*100,f1_score(y_test, knn_pred)*100,
                 f1_score(y_test, km_pred)*100],
     'AUC': [np.round(roc_auc_score(y_test,tree_pred), 3),np.round(roc_auc_score(y_tes
             np.round(roc_auc_score(y_test,km_pred), 3)]

})
model_comparison_df  = model_comparison.sort_values(by='Accuracy Score',ascending=False
model_comparison_df  = model_comparison_df.set_index('Model_Name')
model_comparison_df.reset_index()
```

Out[44]:

	Model_Name	Accuracy Score	Recall Score	Precision Score	F1 Score	AUC
0	Decision Tree Classification	74.237048	28.039702	60.752688	38.370119	0.604
1	K-Nearest Neighbor	71.753016	40.446650	50.778816	45.027624	0.624
2	K-Mean Clustering	28.956707	99.007444	28.581662	44.357977	0.500

8. Solution

What is the solution that is proposed to solve the business problem discussed in Section 1. Also share your learnings while working through solving the problem in terms of challenges, observations, decisions made etc.

Score 2 Marks

---------Type the answer below this line---------

Businesses rely on the insights gained from data analysis to guide a myriad of activities, ranging from budgeting to strategy execution. Customer churn is a prominent issue facing companies. Therefore, preventing customer churn and retaining and retaining customers has become an essential issue for business operations and development. Churn prediction is the process of using data and analytical models to identify which customers are most likely to stop doing business with or using a company's product or service in the near future.Customers have different behaviors and preferences, and reasons for cancelling their subscriptions. Therefore, it is important to actively communicate with each of them to keep them on your customer list. You need to know which marketing activities are most effective for individual customers and when they are most effective.Challenges faced while preprocessing the dataset was to fill in the inconsistencies, when data is manually entered into the system essential information like invalid cutomerID would lose track of a customer and is difficult to fetch the corresponding information. Machine learning algorithms analyze data such as customer demographics, purchase history, and interactions with the company to identify patterns that can predict customer churn. With churn prediction, a company can take proactive measures to retain customers who are at risk of leaving. Churn prediction helps them to focus more on the customers that are at a high risk of leaving. A company with a high churn rate loses many subscribers, resulting in lower growth rates and a greater impact on sales and profits. Companies with low churn rates can retain customers.

NOTE

All Late Submissions will incur a penalty of -2 marks. Do ensure on time submission to avoid penalty.

Good Luck!!!