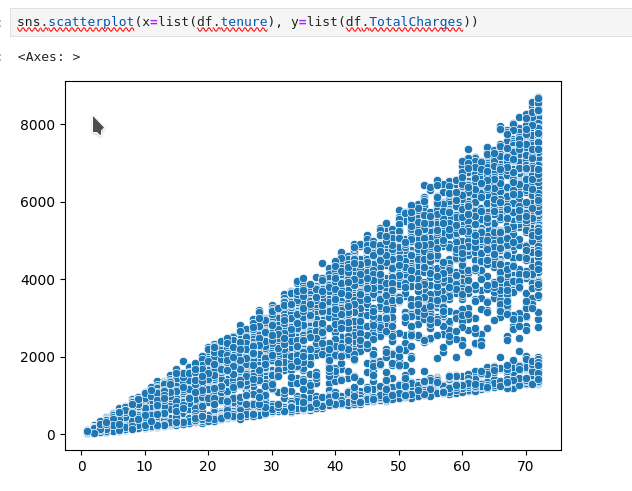
Phase 4 : Development part 2

Customer Churn Prediction

1.Visualizations using IBM Cognos and developing a predictive model

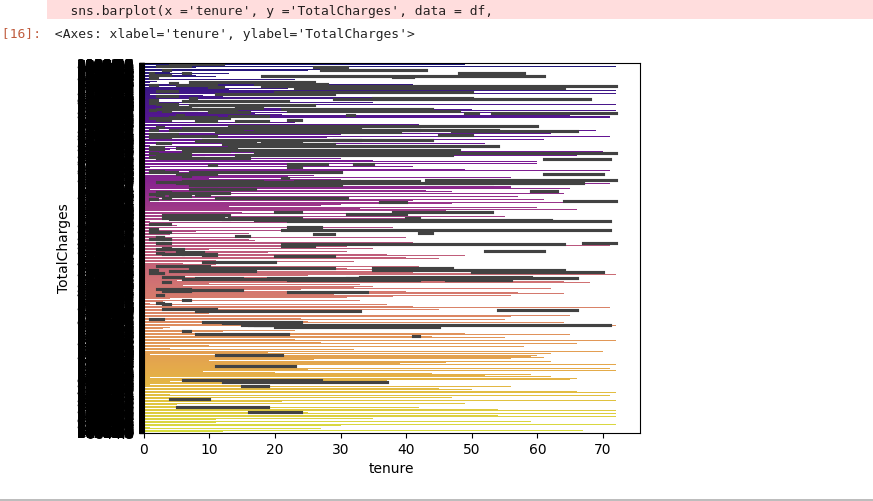
Sactterplot :

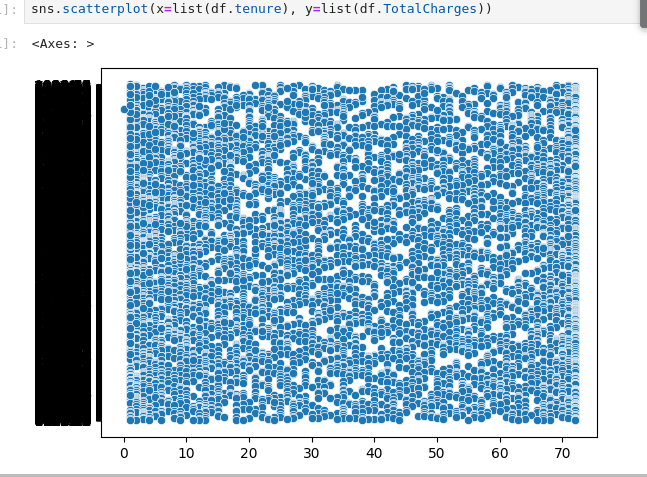


Swarmplot:

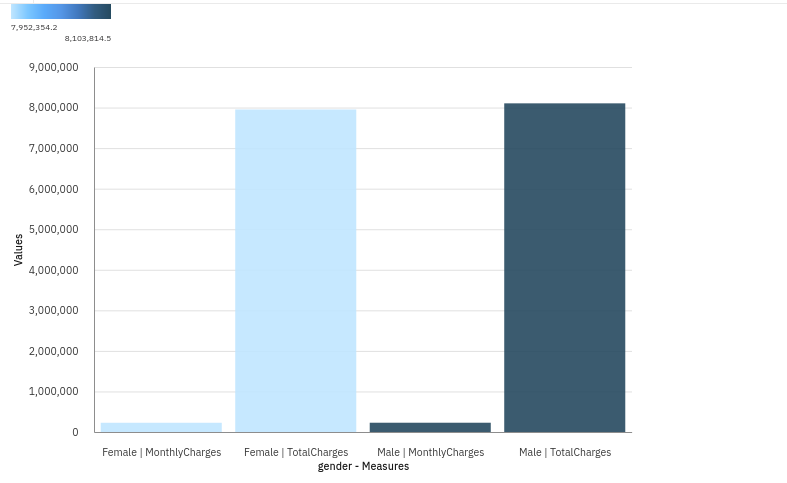


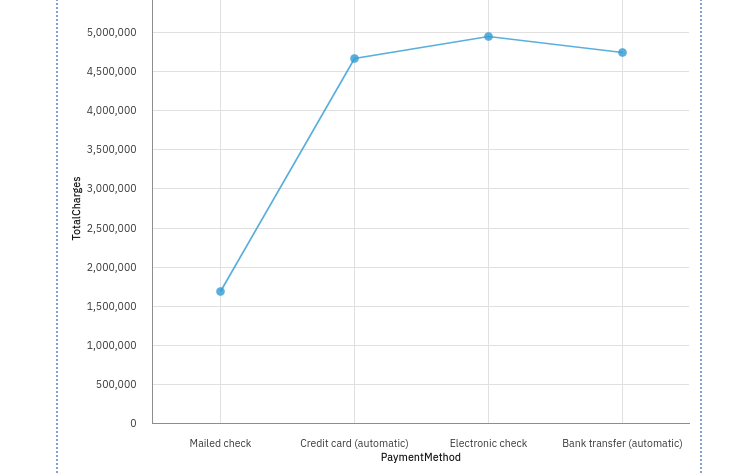
Barplot**:**

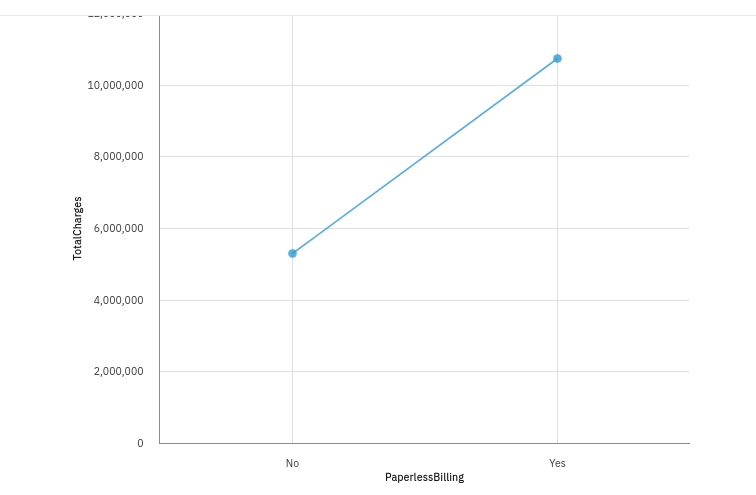




2.Interactive dashboards and reports in IBM Cognos to visualize churn patterns, retention rates, and key factors influencing churn:







3.machine learning algorithms to build a predictive model that identifies potential churners based on historical data and relevant features

import pandas as pd

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

cols = ['gender','SeniorCitizen',"Partner","Dependents"]

numerical = cols

plt.figure(figsize=(20,4))

for i, col in enumerate(numerical):

ax = plt.subplot(1, len(numerical), i+1)

sns.countplot(x=str(col), data=df)

ax.set\_title(f"{col}")

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import train\_test\_split

from sklearn import preprocessing

from sklearn.datasets import make\_classification

from sklearn.preprocessing import binarize, LabelEncoder, MinMaxScaler

# models

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier

from sklearn import metrics

from sklearn.metrics import accuracy\_score, mean\_squared\_error, precision\_recall\_curve

from sklearn.model\_selection import cross\_val\_score

#Neural Network

from sklearn.neural\_network import MLPClassifier

#Bagging

from sklearn.ensemble import BaggingClassifier, AdaBoostClassifier

from sklearn.neighbors import KNeighborsClassifier

#Naive bayes

from sklearn.naive\_bayes import GaussianNB

#Stacking

from mlxtend.classifier import StackingClassifier

finaldf = finaldf.dropna()

finaldf = finaldf.drop(['customerID'],axis=1)

X = finaldf.drop(['Churn'],axis=1)

y = finaldf['Churn']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

y\_train.value\_counts()

preds = rf.predict(X\_test)

print(accuracy\_score(preds,y\_test))

Output:

Accuracy\_score= **approximately 0.78 on the test dataset.**

