Phase 2: Innovation

Advance Machine learning Technique Ensemble Model, Feature Engineering.

Ensemble Models:

Ensemble methods combine predictions from multiple machine learning models to produce a more accurate and robust prediction. Some popular ensemble techniques include:

**a. Random Forests:** This ensemble method builds multiple decision trees and combines their predictions. It is effective for both classification and regression tasks.

**b. Gradient Boosting Machines (GBM):** Algorithms like XGBoost, LightGBM, and CatBoost use boosting techniques to build an ensemble of decision trees sequentially. They are known for their high predictive accuracy.

**c. Stacking:** Stacking combines the predictions of multiple base models using another model, called a meta-learner or blender. It can be an effective way to leverage the strengths of different algorithms.

**d. Voting:** In voting ensembles, multiple models make predictions, and the final prediction is based on a majority vote (for classification) or an average (for regression).

Feature Engineering:

Feature engineering involves creating new features or transforming existing ones to provide more information to the machine learning model. Here are some techniques for effective feature engineering:

**a. Feature Scaling:** Ensure that numerical features are scaled appropriately (e.g., using Min-Max scaling or Standardization) to make them comparable.

**b. One-Hot Encoding:** Convert categorical variables into binary vectors to make them suitable for machine learning algorithms that require numerical inputs.

**c. Feature Interactions:** Create new features by combining or interacting existing ones. For example, if you have height and weight, you can create a BMI (Body Mass Index) feature.

**d. Feature Selection:** Identify and keep only the most relevant features to reduce dimensionality and noise in the dataset.

Python Program:

import pandas as pd

df = pd.read\_csv('/media/johnzavax/Data/WA\_Fn-UseC\_-Telco-Customer-Churn.csv')

df.head()

**df.info()**

**df["Churn"].value\_counts()**

**import matplotlib.pyplot as plt import seaborn as sns import numpy as np 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}")**

**sns.boxplot(x='Churn', y='MonthlyCharges', data=df)**

**cols = ['InternetService',"TechSupport","OnlineBackup","Contract"]**

**plt.figure(figsize=(14,4))**

**for i, col in enumerate(cols):**

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

**sns.countplot(x ="Churn", hue = str(col), data = df)**

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

**df['TotalCharges'] = df['TotalCharges'].apply(lambda x: pd.to\_numeric(x, errors='coerce')).dropna()**

**cat\_features = df.drop(['customerID','TotalCharges','MonthlyCharges','SeniorCitizen','tenure'],axis=1)**

**cat\_features.head()**

**from sklearn import preprocessing**

**le = preprocessing.LabelEncoder()**

**df\_cat = cat\_features.apply(le.fit\_transform)**

**df\_cat.head()**

**num\_features = df[['customerID','TotalCharges','MonthlyCharges','SeniorCitizen','tenure']]**

**finaldf = pd.merge(num\_features, df\_cat, left\_index=True, right\_index=True)**

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

**finaldf = finaldf.dropna()**

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

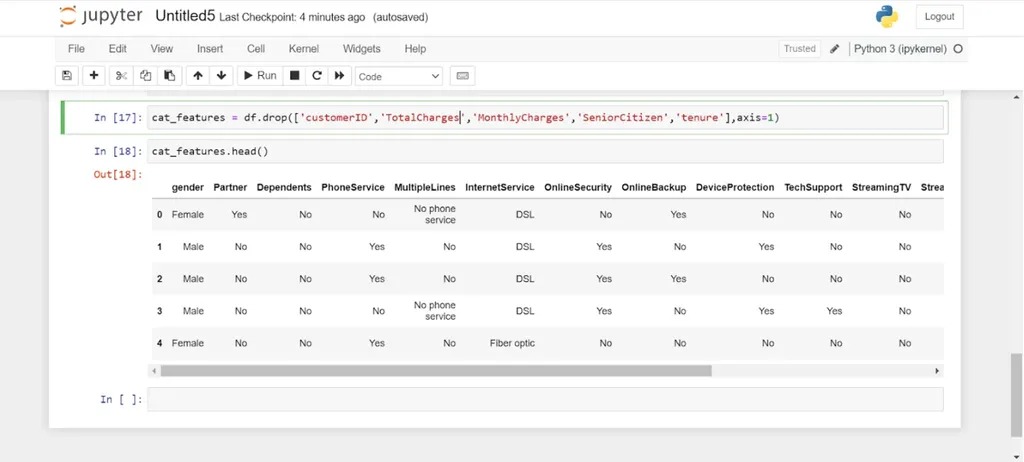
**X = finaldf.drop(['Churn'],axis=1)**

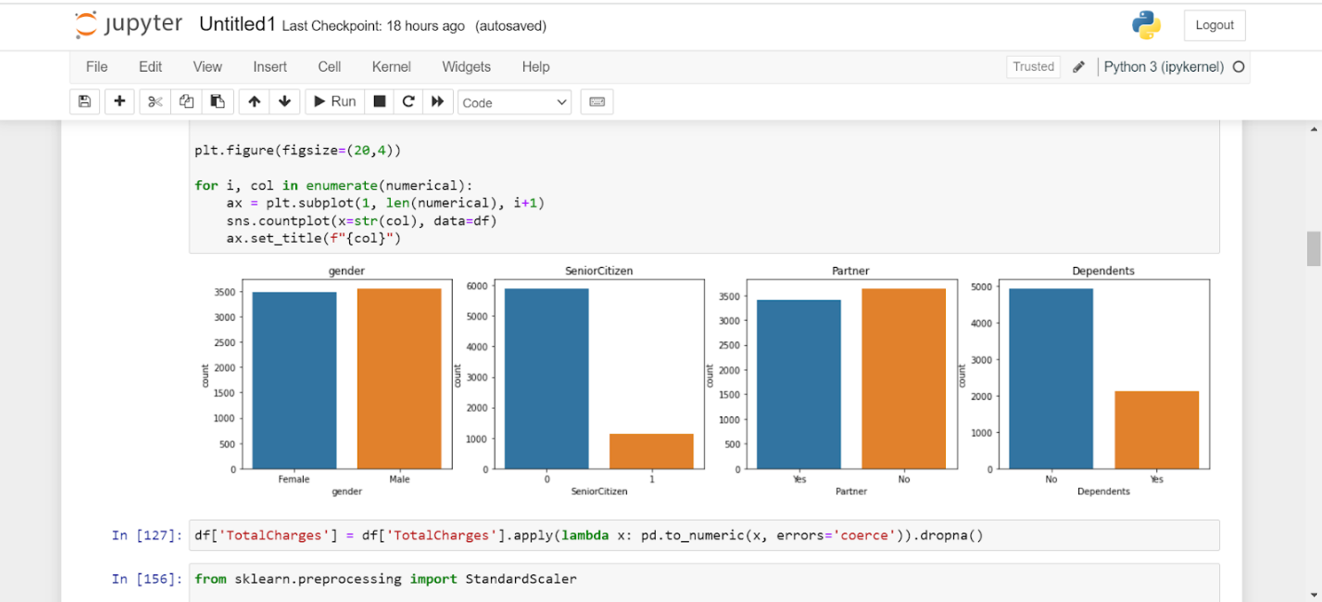
**y = finaldf['Churn']**

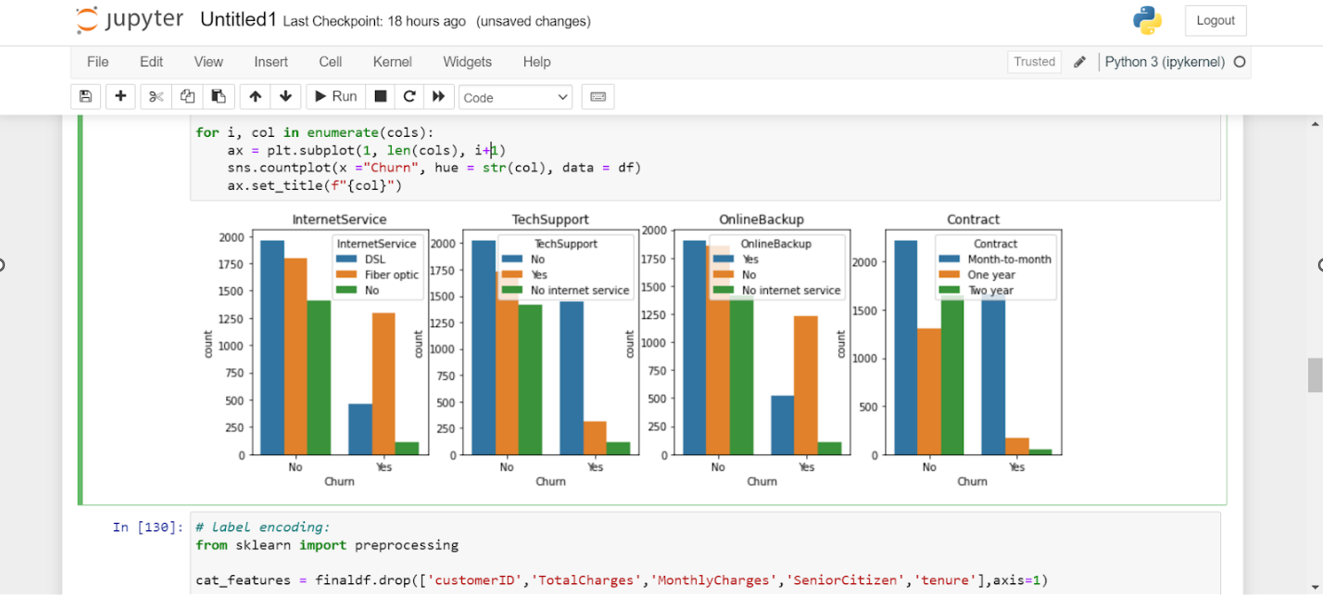
**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)**

**from sklearn.metrics import accuracy\_score preds = rf.predict(X\_test) print(accuracy\_score(preds,y\_test))**

**Out Put:**







**accuracy of approximately 0.78 on the test dataset.**