**Predictive Analytics**

**NAME - MANAS ARORA**

**B.TECH CSE AIML BATCH 5**

**SAP ID – 500109627**

**ROLL.NO - R2142221204**

**Lab 6: Python Library Pandas**

**Customer Churn Prediction (regression)**

* **Dataset: Telecom customer data (e.g., from Kaggle): Suggested link -**[**https://www.kaggle.com/datasets/abhinav89/telecom-customer**](https://www.kaggle.com/datasets/abhinav89/telecom-customer)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, accuracy\_score, precision\_score, recall\_score, f1\_score

# Optional for warnings

import warnings

warnings.filterwarnings("ignore")

from google.colab import drive

drive.mount('/content/drive')

url = '/content/sample\_data/Telecom\_customer churn.csv'

data = pd.read\_csv(url)

print(data.head())

print(data.info())

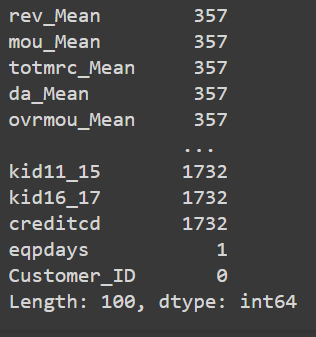
print(data.describe())

* **Data preprocessing (handling missing values, outliers, feature scaling): Suggested link -**[**https://medium.com/womenintechnology/data-preprocessing-steps-for-machine-learning-in-phyton-part-1-18009c6f1153**](https://medium.com/womenintechnology/data-preprocessing-steps-for-machine-learning-in-phyton-part-1-18009c6f1153)

**Data Preprocessing**

**Handle Missing Values**

print(data.isnull().sum())



numeric\_columns = data.select\_dtypes(include=[np.number]).columns

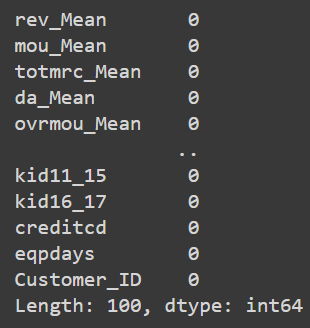
non\_numeric\_columns = data.select\_dtypes(exclude=[np.number]).columns

data[numeric\_columns] = data[numeric\_columns].fillna(data[numeric\_columns].mean())

for col in non\_numeric\_columns:

    data[col].fillna(data[col].mode()[0], inplace=True)

print(data.isnull().sum())



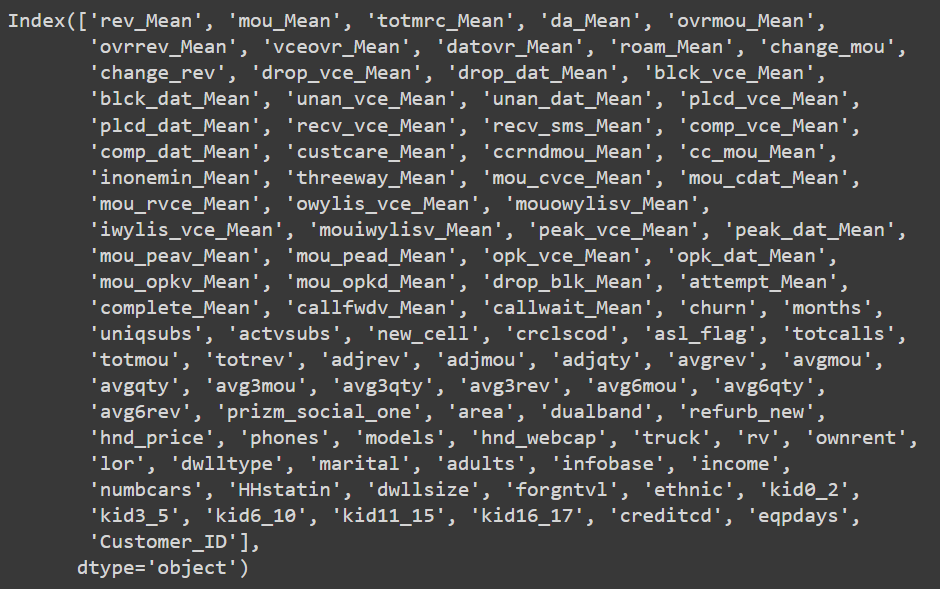
**Handle Outliers**

from scipy.stats import zscore

data = data[(np.abs(zscore(data.select\_dtypes(include=[np.number]))) < 3).all(axis=1)]

**Feature Scaling**

print(data.columns)



numeric\_features = ['rev\_Mean', 'mou\_Mean', 'totmrc\_Mean', 'da\_Mean', 'ovrmou\_Mean', 'ovrrev\_Mean', 'vceovr\_Mean',

                    'datovr\_Mean', 'roam\_Mean', 'drop\_vce\_Mean', 'drop\_dat\_Mean', 'blck\_vce\_Mean', 'blck\_dat\_Mean',

                    'plcd\_vce\_Mean', 'recv\_vce\_Mean', 'totrev', 'totmou']

scaler = StandardScaler()

data[numeric\_features] = scaler.fit\_transform(data[numeric\_features])

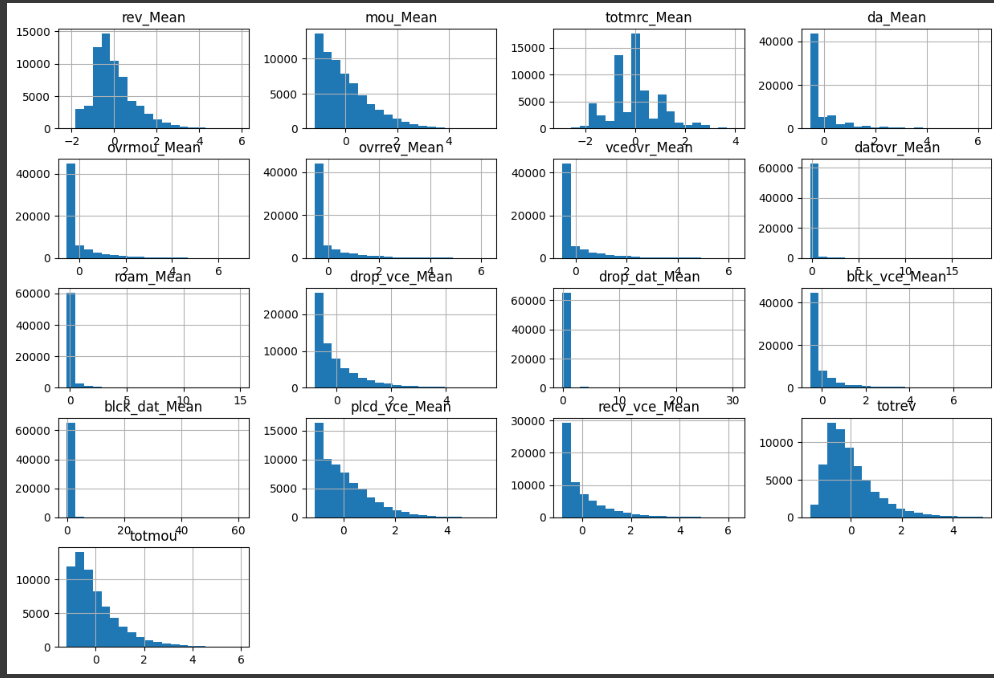
* **Exploratory data analysis (EDA) to identify potential predictors: Samples:**[**https://www.kaggle.com/code/julnazz/exploratory-data-analysis-eda-and-feature-select**](https://www.kaggle.com/code/julnazz/exploratory-data-analysis-eda-and-feature-select)

**Exploratory Data Analysis (EDA)**

**Univariate Analysis**

data[numeric\_features].hist(bins=20, figsize=(15, 10))

plt.show()

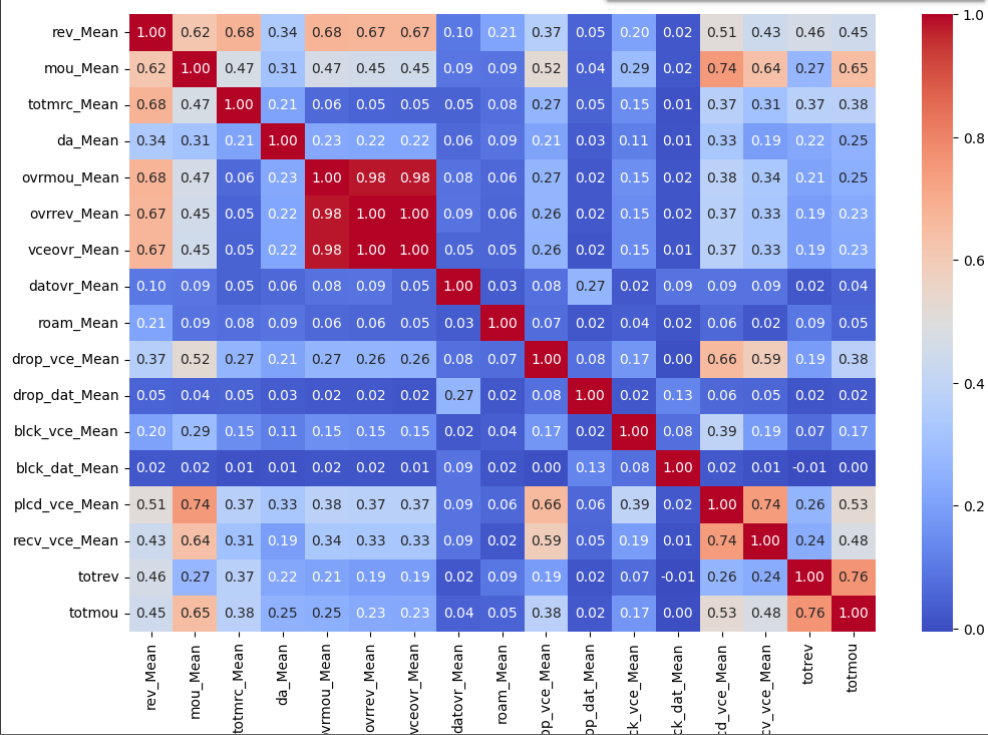


**Correlation Heatmap**

plt.figure(figsize=(12, 8))

sns.heatmap(data[numeric\_features].corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.show()



**Feature Selection**

X = data[['rev\_Mean', 'mou\_Mean', 'totmrc\_Mean', 'totrev', 'totmou', 'uniqsubs', 'actvsubs', 'months']]

y = data['churn']

**Train-Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

**Building a simple linear regression model**

**Build Linear Regression Model**

model = LinearRegression()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

* **Evaluating model performance using accuracy, precision, recall, F1-score; Suggested link -**[**https://medium.com/@maxgrossman10/accuracy-recall-precision-f1-score-with-python-4f2ee97e0d6**](https://medium.com/@maxgrossman10/accuracy-recall-precision-f1-score-with-python-4f2ee97e0d6)

y\_pred\_class = [1 if pred > 0.5 else 0 for pred in y\_pred]

accuracy = accuracy\_score(y\_test, y\_pred\_class)

precision = precision\_score(y\_test, y\_pred\_class)

recall = recall\_score(y\_test, y\_pred\_class)

f1 = f1\_score(y\_test, y\_pred\_class)

print(f'Accuracy: {accuracy}')

print(f'Precision: {precision}')

print(f'Recall: {recall}')

print(f'F1-Score: {f1}')

**Visualizing Predictions**

plt.scatter(y\_test, y\_pred, alpha=0.5)

plt.xlabel("Actual Churn")

plt.ylabel("Predicted Churn")

plt.title("Actual vs Predicted Churn")

plt.show()

