### Import libraries and packages

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

### Load the Dataset

df=pd.read\_csv(“Path of the file”)

### Basic exploration

df.head()  
  
df.tail()

df.info()

df.dtypes

df.describe()

df.describe(include=”all”)

df.isnull().sum()

df.columns()

Charts/graphs/Plots

Categorical-> single column :

sns.countplot(x="Product",data=df)

Two categorical:

sns.countplot(x="Product",hue="Gender",data=df)

One categorical one numerical :

sns.boxplot(x="Product",y="Age",data=df)

sns.violinplot(x="Product",y="Age",data=df)

sns.swarmplot(x="Product",y="Age",data=df)

Two categorical ,one numerical->

sns.boxplot(x="Product",y="Age",hue="Gender",data=df)

Correlation: df.corr(numeric\_only=True)

sns.pairplot(df)

Dropping insignificant columns:

df=df.drop(["Col 1",”Col 2”],axis=1)

Missing Values:

for col in df.select\_dtypes(include=['number']).columns:

    median\_value = df[col].median()

    df[col].fillna(median\_value, inplace=True)

for col in df.select\_dtypes(include=['object']).columns:

    mode\_value = df[col].mode()[0]

    df[col].fillna(mode\_value, inplace=True)

Dealing with categorical columns:

from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

for col in df.select\_dtypes(include=['object']).columns:

    df[col] = label\_encoder.fit\_transform(df[col])

Modelling:

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

X=df.drop(["Dep"],axis=1)

y=df["Dep"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

model = LogisticRegression()

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

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

model.score(X\_test,y\_test)

Confusion Matrix:

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

# Compute confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'], yticklabels=['No', 'Yes'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix Heatmap')

plt.show()