Logistic Regression

1. **Import Libraries**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler,LabelEncoder

from sklearn.linear\_model import LogisticRegression

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

1. **Loading the data and displaying**

df = pd.read\_csv("https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv")

print(df.head())

1. **To drop particular column and save it**

df.drop(['Name','Ticket','Cabin'],axis=1,inplace=True)

#if we don’t use inplace=True then it shows in output but will not save

1. **Data preprocessing**

# check for missing values before handling

print("Missing values before handling:\n",df.isnull().sum())

#handle missing values

if 'Age' in df.columns:

  df['Age'].fillna(df['Age'].median(),inplace=True)

if 'Embarked' in df.columns:

  df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)

#check for missing values after handling

print("\nMissing values after handling:\n",df.isnull().sum())

1. **Code breakdown**

df['Sex'] = LabelEncoder().fit\_transform(df['Sex'])   #Male=1, Female=0

df['Embarked'] = LabelEncoder().fit\_transform(df['Embarked'])

1. **Fill the missing values**

# Drop non-numeric columns to avoid errors in correlation matrix

numeric\_df = df.select\_dtypes(include=['float64','int64'])

#optional: fill missing values to avoid NaNs in corellation

numeric\_df.fillna(numeric\_df.median(),inplace=True)

1. **To generate HeatMap**

#plot the heatmap

plt.figure(figsize=(10,6))

sns.heatmap(numeric\_df.corr(),annot=True,cmap='coolwarm',fmt=".2f")

plt.title("Feature Corelation Heatmap")

plt.show()

1. **Data preprocessing**

x = df.drop("Survived",axis=1)

y = df["Survived"]

1. **Feature Selection**

# select relevant numeric features for scaling (you can customize this)

features = ['Age','Fare','Pclass','SibSp','Parch']

X = df[features]

#handle missing values (if any)

X = X.fillna(X.median())

# std scaling

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# convert back to DataFrame (optional)

X\_scaled\_df = pd.DataFrame(X\_scaled,columns=features)

# view results

print(X\_scaled\_df.head())

1. **Training and Testing**

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

#model=LogisticRegression()

#model.fit(x\_train,y\_train)

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

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