**Data Science**

**Assignment 3**



**Name**: Mohammad Ahmed Shahbaz

**Roll No**: FA21-BSE-050

**Date:** December 20,2023

**COMSATS University Islamabad, Lahore Campus**

# December 20 2023

# CSC461 – Assignment no. 03

# Mohammad Ahmed Shahbaz

# FA21-BSE-050

# "gender-prediction"

#Question 1

import pandas as pd

file\_path = 'gender-prediction.csv'

file\_content = pd.read\_csv (file\_path)

print (file\_content)

#1

instances = file\_content.shape [0]

print ("Number of instances :", instances)

#2

input\_attributes = file\_content.columns[:-1].tolist ()

print ("Number of input attributes :", len (input\_attributes))

#3

output\_attributes = file\_content [file\_content.columns [-1]].unique ()

print ("Number of output attributes :", len (output\_attributes))

#4

categorical\_attributes = file\_content.select\_dtypes (include = ['object']).columns.tolist ()

print ("Number of output attributes :", len (categorical\_attributes))

#5

outputAttribute\_name = 'gender'

class\_ratio = file\_content [outputAttribute\_name].value\_counts (normalize = True)

print ("Class Ratio (Male vs Female):")

print(class\_ratio)

# December 20 2023

# CSC461 – Assignment no. 03

# Mohammad Ahmed Shahbaz

# FA21-BSE-050

# "gender-prediction"

#Question 2

# 1

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

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

from sklearn.metrics import accuracy\_score

X = df.drop("gender", axis=1)

y = df["gender"]

categorical\_columns = X.select\_dtypes(include=['object']).columns

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), X.select\_dtypes(include=['float64', 'int64']).columns),

('cat', OneHotEncoder(), categorical\_columns)

]

)

lr\_model = Pipeline([

('preprocessor', preprocessor),

('classifier', LogisticRegression())

])

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=1/3, random\_state=42)

lr\_model.fit(X\_train, y\_train)

y\_pred\_lr = lr\_model.predict(X\_test)

incorrect\_instances\_lr = (y\_test != y\_pred\_lr).sum()

print(f"1. Incorrect instances (Logistic Regression): {incorrect\_instances\_lr}")

# 2

categorical\_columns = X.select\_dtypes(include=['object']).columns

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), X.select\_dtypes(include=['float64', 'int64']).columns),

('cat', OneHotEncoder(), categorical\_columns)

]

)

# Create a pipeline with the preprocessor and the Logistic Regression model

lr\_model = Pipeline([

('preprocessor', preprocessor),

('classifier', LogisticRegression())

])

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

lr\_model.fit(X\_train, y\_train)

y\_pred\_lr = lr\_model.predict(X\_test)

incorrect\_instances\_lr = (y\_test != y\_pred\_lr).sum()

print(f"2. Re-runing the Incorrect instances (Logistic Regression): {incorrect\_instances\_lr}")

# 3

# Assuming 'gender' as target variable

X = df.drop('gender', axis=1)

y = df['gender']

lr\_model.fit(X, y)

coefficients = abs(lr\_model.named\_steps['classifier'].coef\_[0])

top2\_indices = coefficients.argsort()[-2:][::-1]

top2\_features = X.columns[top2\_indices]

print(f"The 2 powerful attributes: {top2\_features}")

#4

top2\_attributes = ['beard', 'shoe\_size']

X\_excluded = X.drop(columns=top2\_attributes)

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), X\_excluded.select\_dtypes(include=['float64', 'int64']).columns),

('cat', OneHotEncoder(), X\_excluded.select\_dtypes(include=['object']).columns)

]

)

lr\_model\_excluded = Pipeline([

('preprocessor', preprocessor),

('classifier', LogisticRegression())

])

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

lr\_model\_excluded.fit(X\_train\_excluded, y\_train)

y\_pred\_lr\_excluded = lr\_model\_excluded.predict(X\_test\_excluded)

incorrect\_instances\_lr\_excluded = (y\_test != y\_pred\_lr\_excluded).sum()

print(f"Incorrect instances (Logistic Regression) after excluding the 2 powerful attributes: {incorrect\_instances\_lr\_excluded}")

# December 20 2023

# CSC461 – Assignment no. 03

# Mohammad Ahmed Shahbaz

# FA21-BSE-050

# "gender-prediction"

#Question 3

!pip install scikit-plot

from sklearn import preprocessing

from sklearn.preprocessing import LabelEncoder

import pandas as pd

import numpy as np

from sklearn.naive\_bayes import GaussianNB, BernoulliNB, MultinomialNB

from sklearn.svm import SVC, LinearSVC

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import accuracy\_score, f1\_score, classification\_report, confusion\_matrix

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, cross\_val\_predict, LeavePOut

from sklearn import metrics, model\_selection

import scikitplot as skplt

file\_content = pd.read\_csv ('gender-prediction.csv')

x = file\_content.drop ("gender", axis = 1)

y = file\_content ['gender']

label\_encoder\_x = LabelEncoder ()

X\_encoded = x.apply (label\_encoder\_x.fit\_transform)

label\_encoder\_y = LabelEncoder ()

Y\_encoded = label\_encoder\_y.fit\_transform (y)

X\_train, x\_test, Y\_train, y\_test = train\_test\_split (X\_encoded, Y\_encoded, test\_size = 0.33, random\_state = 1)

model\_RF = RandomForestClassifier()

model\_RF.fit (X\_train, Y\_train)

monte\_carlo\_accuracy = cross\_val\_score (model\_RF, X\_encoded, Y\_encoded, cv = 6, scoring = 'accuracy')

print("Monte Carlo Accuracy:", monte\_carlo\_accuracy.mean() \* 100)

leave\_p\_out = LeavePOut (p = 3)

leave\_p\_out\_accuracy = cross\_val\_score (model\_RF, X\_encoded, Y\_encoded, cv = leave\_p\_out, scoring = 'accuracy')

print("Leave p out Accuracy:", leave\_p\_out\_accuracy.mean() \* 100)