**ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ**

**ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ**

**ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ**

**ΤΟΜΕΑΣ ΒΙΟΜΗΧΑΝΙΚΗ ΔΙΟΙΚΗΣΗΣ**

**Βαρνάς Γεώργιος**

**ΑΕΜ: 6132**

**2η εργασία στο μάθημα «Ανάλυση Δεδομένων» #MENG228 2023-2024**

**Εισαγωγή**

Η παρούσα εργασία αποτελεί συνέχεια και προέκταση της προηγούμενης(1ης) εργασίας στο μάθημα. Χρησιμοποιείται το ίδιο dataset, το οποίο περιλαμβάνει 20 συνολικά μεταβλητές(20 στήλες), η μία εκ των οποίων είναι η εξαρτημένη μεταβλητή(“Life expectancy”) και αφορά το προσδόκιμο όριο ζωής για τον πληθυσμό διαφόρων χωρών. Οι υπόλοιπες 19 μεταβλητές, ως στήλες του dataframe μας, αποτελούν διαφορετικούς παράγοντες εξάρτησης του life expectancy, εξάρτηση την οποία διερευνούμε και επιχειρούμε να προσδιορίσουμε στα πλαίσια αυτής της εργασίας.

**Μέρος Α**

Στο πρώτο μέρος εφαρμόζεται η μέθοδος της διασταυρωμένης επικύρωσης(cross validation) για τα 2 μοντέλα πολλαπλής γραμμικής παλινδρόμησης, όπως αυτά είχαν προκύψει από την 1η εργασία και εκτιμάται το test MSE για το κάθε μοντέλο. Με το cross validation, επιτυγχάνουμε μία πιο αξιόπιστη εκτίμηση της επίδοσης του μοντέλου μας σε σύγκριση με τον απλούστερο διαχωρισμό των δεδομένων σε σύνολα εκπαίδευσης/ελέγχου ενώ επίσης σιγουρευόμαστε πως δεν θα υπάρξει overfitting, καθώς αυτό θα φανεί στη διαδικασία του cross-validation εάν ένα μοντέλο δεν έχει καλό performance στα δεδομένα επικύρωσης σε αντίθεση με τα δεδομένα εκπαίδευσης για κάποιο iteration.

Χρησιμοποιούνται 2 τεχνικές για cross validation, οι οποίες είναι οι εξής :

* K-Fold cross validation, όπου το σετ δεδομένων μας χωρίζεται σε k υποσύνολα/folds(στην περίπτωσή μας k=5), από τα οποία τα k-1 χρησιμοποιούνται ως δεδομένα εκπαίδευσης(training set) και το 1 fold χρησιμοποιείται ως δεδομένα επικύρωσης(validation set). Η διαδικασία επαναλαμβάνεται k φορές και σε κάθε επανάληψη αλλάζω το validation set, ώστε όλα τα k folds να λειτουργήσουν 1 φορά ως validation set.
* Leave-One-Out cross validation, όπου χρησιμοποιείται η ίδια μεθοδολογία με το k-fold cross validation, με τη διαφορά ότι εκτελούμε επαναλήψεις ίσες με n, όπου n ο αριθμός των data points του set μας. Οπότε σε κάθε επανάληψη το training set περιλαμβάνει n-1 δεδομένα και το validation set 1 δεδομένο, εξού και η ονομασία της μεθόδου.

Για την εκτέλεση της διασταυρωμένης επικύρωσης, καταρχάς ανατρέχουμε στην 1η εργασία και ορίζουμε τους predictors για καθένα από τα 2 μοντέλα, όπου και τα 2 είναι μοντέλα πολλαπλής γραμμικής παλινδρόμησης, με τη διαφορά ότι το 2ο έχει προκύψει από τη μέθοδο forward-selection, οπότε δεν έχουν τις ίδιες ανεξάρτητες μεταβλητές(predictors). Συγκεκριμένα, βάση της ανάλυσης μας στην προηγούμενη εργασία, το 1ο μας μοντέλο περιέχει ως ανεξάρτητες μεταβλητές τις: ‘Year’, ‘Adult Mortality’, ‘Alcohol’, ‘BMI’, ‘under five deaths’, ‘Diphtheria’, ‘HIV/AIDS’, ‘Income composition of resources’, ‘Schooling’, ενώ το 2ο μοντέλο περιέχει αντίστοιχα τις: ‘Year’, ‘Adult Mortality’, ‘Alcohol’, ‘BMI’, ‘percentage expenditure’, ‘Hepatitis B’, ‘Measles’, ‘under five deaths’, ‘Polio’, ‘Total expenditure’, ‘Diphtheria’, ‘HIV/AIDS’, ‘GDP’, ‘Population’, ‘thinness 5-9 years’, ‘Income composition of resources’, ‘Schooling’.

Δημιουργούμε από ένα KFold cross validator για κάθε μοντέλο, ορίζουμε αριθμό επαναλήψεων ίσο με 5 και υπολογίζουμε το test MSE, ως τον μέσο όρο των test MSE κάθε μίας από τις 5 επαναλήψεις της διασταυρωμένης επικύρωσης. Λαμβάνουμε τα εξής αποτελέσματα : 

Το 2ο μοντέλο παλινδρόμησης που προέκυψε από τη μέθοδο forward selection έχει χαμηλότερο test MSE, οπότε για το μοντέλο αυτό θα εκτελέσουμε και την μέθοδο Leave-One-Out cross validation με αντίστοιχο τρόπο και τα αποτελέσματα προκύπτουν ως εξής:



Το test MSE που προέκυψε από την Leave-One-Out cross validation είναι λίγο μικρότερο από αυτό της k-fold cross validation για το ίδιο μοντέλο. Η εκτίμηση σφάλματος της Leave-One-Out τεχνικής θεωρείται γενικά εγκυρότερη από την k-fold, διότι εκτελούνται αρκετά περισσότερες(n >> k) επαναλήψεις επικύρωσης, καθώς κάνουμε χρήση όλων των data points. Ωστόσο αξίζει να αναφερθεί ότι η L-O-O cross validation μέθοδος απαιτεί αρκετά σημαντική υπολογιστική ισχύ, καθώς εκτελούνται πολλές επαναλήψεις(iterations) και θα πρέπει να χρησιμοποιείται με σύνεση, ειδικά σε υπερμεγέθη datasets.

**Μέρος B**

Σε αυτό το μέρος της εργασίας εξετάζονται τεχνικές ταξινόμησης(classification) και άρα η εξαρτημένη μας μεταβλητή “life expectancy” μετατρέπεται σε δυαδική. Συγκεκριμένα, υπολογίζουμε τη μέση τιμή του life expectancy, δημιουργούμε τη νέα μεταβλητή “life\_expectancy\_dummies” με χρήση της συνάρτησης .get\_dummies από τη βιβλιοθήκη Pandas και κριτήριο να ξεπερνά ή όχι τη μέση τιμή που είχε η αριθμητική έκδοση της μεταβλητής. Έπειτα χρησιμοποιούμε τη συνάρτηση .astype ώστε να μετατρέψουμε από Boolean σε integer(0 ή 1) τις τιμές της νέας δυαδικής μεταβλητής μας. Τέλος με χρήση του pd.concat ενώνουμε τα δύο dataframes και δημιουργούμε το df2, από το οποίο διαγράφουμε τώρα την numerical μεταβλητή “life expectancy”, καθώς και τις μεταβλητές “Year”, “Country”, “Status” που δεν μετέχουν στην ανάλυση μας.

Η πρώτη μέθοδος ταξινόμησης που εφαρμόζεται είναι η ***λογιστική παλινδρόμηση***. Η λογιστική παλινδρόμηση είναι μια στατιστική μέθοδος όπου χρησιμοποιείται για προβλήματα ταξινόμησης και στοχεύει στην εκτίμηση της πιθανότητας(0 ως 1) και τη μετέπειτα ταξινόμηση του αποτελέσματος στην αντίστοιχη κατηγορία, θέτοντας ένα νούμερο ανάμεσα στο 0 και το 1 ως σύνορο μεταξύ των 2 τάξεων(threshold) (π.χ. threshold=0.5). Χρησιμοποιείται η σιγμοειδής συνάρτηση, ώστε να εξασφαλισθεί ότι το αποτέλεσμα θα είναι μεταξύ 0 και 1, εφόσον αποτυπώνει πιθανότητα. Το παραγόμενο μοντέλο παρέχει(αντίστοιχα με τη γραμμική παλινδρόμηση) συντελεστές(coefficients) για τις ανεξάρτητες μεταβλητές(predictors) αλλά και σταθερό όρο(intercept). Στο πρόβλημά μας κάνουμε χρήση της διωνυμικής λογιστικής παλινδρόμησης, αφού έχουμε 2 classes(υψηλό life expectancy/True(1) και χαμηλό life expectancy/False(0).

Ορίζουμε τις μεταβλητές-predictors και δημιουργούμε το μοντέλο, κάνουμε το fit και την πρόβλεψη και λαμβάνουμε το σύνολο των συντελεστών για κάθε ανεξάρτητη μεταβλητή, τον πίνακα σύγχυσης(confusion matrix) όπου αποτυπώνει το σύνολο των True Negatives, True Positives, False Positives(σφάλμα 1ου είδους), False Negatives(σφάλμα 2ου είδους), καθώς επίσης την ακρίβεια στα training data, την ακρίβεια στα test data και το classification report.

Ο σταθερός όρος του μοντέλου και ο συντελεστής συσχέτισης για κάθε predictor:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Ο πίνακας σύγχυσης:

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, τυπογραφία

Περιγραφή που δημιουργήθηκε αυτόματα

True Negatives=95, False Negatives=53, True Positives=114, False Positives=18

Η ακρίβεια για training/test data και το classification report:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Στη συνέχεια για το παραγόμενο μοντέλο λογιστικής παλινδρόμησης, θα εκτελέσουμε προβλέψεις με αλλαγή του threshold από 0.5 σε 0.4 και 0.6 αντίστοιχα και θα ελέγξουμε την ακρίβεια που επιτυγχάνεται στις προβλέψεις.

Για threshold = 0.4 :

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Για threshold = 0.6:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Καταρχάς, για να επεξηγηθούν οι επιπτώσεις που επέρχονται με τις αυξομειώσεις του threshold θα εξηγήσουμε τις ενδείξεις του classification report.

Συγκεκριμένα:

* Precision = δηλαδή το precision δείχνει τι ποσοστό των περιπτώσεων που προβλέφθηκαν ως Positives είναι όντως Positives. Υψηλό precision σημαίνει χαμηλό ποσοστό σφαλμάτων 1ου είδους
* Recall = δηλαδή το recall δείχνει τι ποσοστό των πραγματικών Positives προβλέφθηκαν όντως ως Positives. Υψηλό recall σημαίνει χαμηλό ποσοστό σφαλμάτων 2ου είδους
* F1-score = 2\*(precision + recall)/precision + recall δηλαδή το F1-score είναι απλώς ο αρμονικός μέσος όρος από precision και recall

Μειώνοντας το threshold από 0.5 σε 0.4, όπως γίνεται αντιληπτό δεχόμαστε περισσότερες περιπτώσεις(ευκολότερη αποδοχή) ως True(υψηλό life expectancy). Όπως βλέπουμε, αυξάνεται το precision και μειώνεται το recall. Αντίστοιχα, αυξάνοντας το threshold από 0.5 σε 0.6, μειώνεται το precision αλλά αυξάνεται το recall.

Η ιδανική επιλογή ενός threshold απαιτεί την διερεύνηση επιπλέον στοιχείων κόστους και εξαρτάται από τη φύση και τις απαιτήσεις του προβλήματος μας. Για παράδειγμα εάν θεωρηθεί ότι το κόστος για τα σφάλματα 1ου είδους είναι υψηλό και έχουμε ως προτεραιότητα να τα αποφύγουμε, θα προτιμήσουμε ένα χαμηλότερο threshold, ενώ αντίστοιχα για αποφυγή σφαλμάτων 2ου είδους θα προτιμήσουμε ένα υψηλότερο threshold.

Επόμενη μέθοδος ταξινόμησης που χρησιμοποιείται είναι η ***linear discriminant analysis***. Η συγκεκριμένη τεχνική αποσκοπεί στην εύρεση του γραμμικού συνδυασμού των predictors ενός προβλήματος που οδηγεί στον βέλτιστο διαχωρισμό των τάξεων που μελετώνται. Η λειτουργία του αλγορίθμου της linear discriminant analysis(LDA) μειώνει τον αριθμό των διαστάσεων του διαγράμματος που απεικονίζει τα data ορισμένων ανεξάρτητων μεταβλητών, έτσι ώστε να είναι απλούστερη η ταξινόμηση τους σε κλάσεις. Η εφαρμογή της μεθόδου υποθέτει πως οι predictors που μετέχουν ακολουθούν κανονική κατανομή και πως οι τάξεις έχουν παρόμοιους πίνακες συμμεταβλητότητας(covariance matrixes).

Παρακάτω παρουσιάζονται οι συντελεστές(coefficients) που προκύπτουν από το μοντέλο LDA, η ακρίβεια στα training και test δεδομένα μας, ο πίνακας σύγχυσης(confusion matrix) και το classification report για το παραγόμενο μοντέλο:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Τελευταία μέθοδος ταξινόμησης που εξετάζεται είναι η ***K-Nearest-Neighbors(KNN)***. Η βασική λογική αυτής της μη-παραμετρικής μεθόδου είναι η ταξινόμηση ενός νέου data point με βάση τις τάξεις στις οποίες ανήκουν τα K κοντινότερα σε αυτό γειτονικά data points κατά πλειοψηφία. Μια σημαντική παράμετρος για τον αλγόριθμο είναι ο τρόπος υπολογισμού της απόστασης μεταξύ του νεοεισερχόμενου data point που επιχειρούμε τα ταξινομήσουμε και των όλων των υπολοίπων data points του training dataset μας, ώστε να αποσαφηνιστεί ποια είναι τα κοντινότερα του(K σε πλήθος). Ο συχνότερος τρόπος υπολογισμού της απόστασης είναι μέσω του τύπου της ευκλείδειας απόστασης(Euclidean distance), ωστόσο μπορούν να χρησιμοποιηθούν και άλλα metrics της απόστασης(π.χ. Manhattan distance, Minkowski distance).

Ο KNN αλγόριθμος είναι μη-παραμετρικός, δηλαδή δεν πραγματοποιεί υποθέσεις για την κατανομή των ανεξάρτητων μεταβλητών που μετέχουν στους υπολογισμούς. Η εκλογή του αριθμού K των γειτονικών σημείων που θα μετέχουν στην πρόβλεψη της ταξινόμησης κάθε νέου σημείου μπορεί να επηρεάσει σημαντικά την επίδοση της μεθόδου. Χαμηλές τιμές του K οδηγούν σε αρκετά περίπλοκα όρια απόφασης για το cutoff των τάξεων και συνεπακόλουθα υψηλή διακύμανση και χαμηλή μεροληψία(high variance/low bias), ενώ αντιθέτως υψηλές τιμές του Κ θα μας δώσουν χαμηλή διακύμανση και υψηλή μεροληψία(low variance/high bias).

Εφαρμόζουμε την KNN μέθοδο στα δεδομένα μας για τιμές K = 1,3,5,7,9 και λαμβάνουμε τα εξής αποτελέσματα:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Για K=7 έχω τα χαμηλότερα σφάλματα και την καλύτερη ακρίβεια.

Βάση των classification reports που προέκυψαν για όλες τις μεθόδους ταξινόμησης και των training και test errors, η καλύτερη μέθοδος φαίνεται ξεκάθαρα πως είναι αυτή της ***linear discriminant analysis(LDA)***.

Ο κώδικας σε python(VSCode):

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

import statsmodels.api as sm

from sklearn.preprocessing import PolynomialFeatures

from sklearn.model\_selection import KFold

from sklearn.model\_selection import LeaveOneOut

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix

# Part A

# Read our data, create the dataframe

df1 = pd.read\_excel(r"c:\Users\Γιωργος\Desktop\MECHANICAL ENGINEERING\4ο ΕΤΟΣ\ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ\1η εργασια\Data Analysis\_2024 1st Case\_Data.xlsx")

print(df1.head())

print(df1.describe())

print(df1.info())

# Check for NaN values

print(df1.isnull().values.any()) #prepei na vgazei False, an den exw NaN values se olo to dataframe

# Remove years 2007,2013 from the dataframe

i = df1[(df1['Year']==2013) | (df1['Year']==2007)].index

print(i)

df1.drop(index=i,inplace=True)

print(df1.info()) #gia epalitheysh, se sygrish me to prohgoumeno info

# Get the distribution plot of our target-variable

print(sns.histplot(data=df1['Life expectancy '],stat='density',kde=True))

plt.show()

predictors = ['Year', 'Adult Mortality','Alcohol', 'percentage expenditure',

              'Hepatitis B', 'Measles ', ' BMI ','under-five deaths ', 'Polio', 'Total expenditure',

              'Diphtheria ',' HIV/AIDS', 'GDP', 'Population', ' thinness 5-9 years',

              'Income composition of resources', 'Schooling']

# Define the y,X for the first model based on the 1st project

y\_m1 = df1['Life expectancy ']

X\_m1 = df1[['Year', 'Adult Mortality','Alcohol', ' BMI ','under-five deaths ',

              'Diphtheria ',' HIV/AIDS','Income composition of resources', 'Schooling']]

# Define the y,X for the second(forward selection) model based on the 1st project

y\_m2 = df1['Life expectancy ']

X\_m2 = df1[['Year', 'Adult Mortality','Alcohol', 'percentage expenditure',

              'Hepatitis B', 'Measles ', ' BMI ','under-five deaths ', 'Polio', 'Total expenditure',

              'Diphtheria ',' HIV/AIDS', 'GDP', 'Population', ' thinness 5-9 years',

              'Income composition of resources', 'Schooling']]

m1 = LinearRegression()

m2 = LinearRegression()

test\_mse\_m1 = []

test\_mse\_m2 = []

# k-fold cross-validation on the first model

kf1 = KFold(n\_splits=5, shuffle=True, random\_state=33)

for train\_index, test\_index in kf1.split(X\_m1,y\_m1):

    X\_train, X\_test = X\_m1.values[train\_index], X\_m1.values[test\_index]

    y\_train, y\_test = y\_m1.values[train\_index], y\_m1.values[test\_index]

    m1.fit(X\_train,y\_train)

    y\_pred = m1.predict(X\_test)

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

    test\_mse\_m1.append(mse)

# k-fold cross-validation on the second model

kf2 = KFold(n\_splits=5, shuffle=True, random\_state=33)

for train\_index, test\_index in kf2.split(X\_m2,y\_m2):

    X\_train, X\_test = X\_m2.values[train\_index], X\_m2.values[test\_index]

    y\_train, y\_test = y\_m2.values[train\_index], y\_m2.values[test\_index]

    m2.fit(X\_train,y\_train)

    y\_pred = m2.predict(X\_test)

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

    test\_mse\_m2.append(mse)

# Calculate each model's test MSE

avg\_mse\_m1 = sum(test\_mse\_m1) / len(test\_mse\_m1)

avg\_mse\_m2 = sum(test\_mse\_m2) / len(test\_mse\_m2)

print("test MSE of the 1st model is:",avg\_mse\_m1)

print("test MSE of the 2nd(forward selection) model is:",avg\_mse\_m2)

# Leave-One-Out cross-validation for the 2nd model

test\_mse\_m2\_loo = []

loo = LeaveOneOut()

for train\_index, test\_index in loo.split(X\_m2):

    X\_train, X\_test = X\_m2.values[train\_index], X\_m2.values[test\_index]

    y\_train, y\_test = y\_m2.values[train\_index], y\_m2.values[test\_index]

    m2.fit(X\_train, y\_train)

    y\_pred = m2.predict(X\_test)

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

    test\_mse\_m2\_loo.append(mse)

avg\_mse\_m2\_loo = np.mean(test\_mse\_m2\_loo)

print("test MSE of the 2nd(forward selection) model with the loo method is:",avg\_mse\_m2\_loo)

# Part B

print(df1.head())

print(df1.tail())

print(df1.info())

# Convert Life expectancy into a dummy variable

avg\_life\_expectancy = df1['Life expectancy '].mean()

print(avg\_life\_expectancy)

life\_expectancy\_dummies = pd.get\_dummies(df1['Life expectancy '] > avg\_life\_expectancy ,

                                        prefix='Life expectancy ', drop\_first=True)

life\_expectancy\_dummies = life\_expectancy\_dummies.astype(int)

print(life\_expectancy\_dummies)

print(life\_expectancy\_dummies.value\_counts()) #epalitheysh

df2 = pd.concat([df1, life\_expectancy\_dummies], axis=1)

df2.drop(['Life expectancy ','Year','Country', 'Status'], axis=1, inplace=True)

print(df2.head())

print(df2.info())

print(df2.columns)

# to df2 einai pleon etoimo gia ton ML algorithmo

# Logistic Regression

y\_m3 = df2['Life expectancy \_True']

X\_m3 = df2[['Adult Mortality', 'Alcohol', 'percentage expenditure', 'Hepatitis B',

       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',

       'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population', ' thinness 5-9 years',

       'Income composition of resources', 'Schooling',]]

X3\_train, X3\_test, y3\_train, y3\_test = train\_test\_split(X\_m3, y\_m3, test\_size=0.2, random\_state=33)

logreg\_model = LogisticRegression()

logreg\_model.fit(X3\_train,y3\_train)

print("Intercept of the logistic regression model:", logreg\_model.intercept\_[0])

print("Coefficients of the logistic regression model:")

for i, coef in enumerate(logreg\_model.coef\_[0]):

    print(f"Coefficient of {X\_m3.columns[i]}: {coef}")

y3\_pred = logreg\_model.predict(X3\_test)

print("Confusion Matrix for Logistic Regression:")

print(confusion\_matrix(y3\_test, y3\_pred))

train\_acc\_logreg = accuracy\_score(y3\_train, logreg\_model.predict(X3\_train))

print("Training accuracy of the logistic regression model:", train\_acc\_logreg)

acc05 = accuracy\_score(y3\_test,y3\_pred)

print("accuracy on test data with 0.5 threshold is:",acc05)

print("classification report for threshold=0.5:")

print(classification\_report(y3\_test,y3\_pred))

y3\_pred\_prob = logreg\_model.predict\_proba(X3\_test)[:, 1]

# Apply threshold=0.4

y3\_pred\_04 = (y3\_pred\_prob > 0.4).astype(int)

acc04 = accuracy\_score(y3\_test,y3\_pred\_04)

print("accuracy on test data with 0.4 threshold is:",acc04)

print("classification report for threshold=0.4:")

print(classification\_report(y3\_test,y3\_pred\_04))

# Apply threshold=0.6

y3\_pred\_06 = (y3\_pred\_prob > 0.6).astype(int)

acc06 = accuracy\_score(y3\_test,y3\_pred\_06)

print("accuracy on test data with 0.6 threshold is:",acc06)

print("classification report for threshold=0.6:")

print(classification\_report(y3\_test,y3\_pred\_06))

# LDA

y\_m4 = df2['Life expectancy \_True']

X\_m4 = df2[['Adult Mortality', 'Alcohol', 'percentage expenditure', 'Hepatitis B',

       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',

       'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population', ' thinness 5-9 years',

       'Income composition of resources', 'Schooling',]]

X4\_train, X4\_test, y4\_train, y4\_test = train\_test\_split(X\_m4, y\_m4, test\_size=0.2, random\_state=33)

lda\_model = LinearDiscriminantAnalysis()

lda\_model.fit(X4\_train, y4\_train)

print("Intercept of the LDA model:", lda\_model.intercept\_[0])

print("Coefficients of the LDA model:")

for i, coef in enumerate(lda\_model.coef\_[0]):

    print(f"Coefficient of {X\_m4.columns[i]}: {coef}")

train\_acc\_lda = accuracy\_score(y4\_train, lda\_model.predict(X4\_train))

print("Training accuracy of the LDA model:", train\_acc\_lda)

y4\_pred = lda\_model.predict(X4\_test)

acc\_lda = accuracy\_score(y4\_test, y4\_pred)

print("\nConfusion Matrix for LDA:")

print(confusion\_matrix(y4\_test, y4\_pred))

print("accuracy on test data with LDA model is:",acc\_lda)

print("classification report for LDA model:")

print(classification\_report(y4\_test,y4\_pred))

# KNN

y\_m5 = df2['Life expectancy \_True']

X\_m5 = df2[['Adult Mortality', 'Alcohol', 'percentage expenditure', 'Hepatitis B',

       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',

       'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population', ' thinness 5-9 years',

       'Income composition of resources', 'Schooling',]]

X5\_train, X5\_test, y5\_train, y5\_test = train\_test\_split(X\_m5, y\_m5, test\_size=0.2, random\_state=33)

k\_values = [1,3,5,7,9]

for k in k\_values:

    knn\_model = KNeighborsClassifier(n\_neighbors=k)

    knn\_model.fit(X5\_train,y5\_train)

    y5\_pred = knn\_model.predict(X5\_test)

    acc\_knn = accuracy\_score(y5\_test,y5\_pred)

    print(f"accuracy on test data with K={k} is: {acc\_knn}")

    print(f"classification report for KNN with K={k}:")

    print(classification\_report(y5\_test,y5\_pred))

    print(f"\nConfusion Matrix for KNN with k={k}:")

    print(confusion\_matrix(y5\_test, y5\_pred))