**ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ**

**ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ**

**ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ**

**ΤΟΜΕΑΣ ΒΙΟΜΗΧΑΝΙΚΗ ΔΙΟΙΚΗΣΗΣ**

**Βαρνάς Γεώργιος**

**ΑΕΜ: 6132**

**1η εργασία στο μάθημα «Ανάλυση Δεδομένων» #MENG228 2023-2024**

**Εισαγωγή**

Στην παρούσα εργασία μελετάται ένα σετ δεδομένων που περιέχει 20 συνολικά μεταβλητές. Η ανάλυσή μας έχει επίκεντρο τη μεταβλητή “Life expectancy”(εξαρτημένη μεταβλητή), δηλαδή το προσδόκιμο όριο ζωής του πληθυσμού διαφόρων χωρών, σύμφωνα με δεδομένα του Παγκόσμιου Οργανισμού Υγείας. Τα δεδομένα μας αφορούν τα έτη 2000-2015. Οι υπόλοιπες 19 μεταβλητές αφορούν στοιχεία τα οποία μελετάμε κατά πόσο επηρεάζουν τη μεταβλητή “Life expectancy”(ανεξάρτητες μεταβλητές), εξαιρώντας τις μεταβλητές Country και Status από τα δημιουργούμενα μοντέλα, καθώς οι 2 αυτές μεταβλητές δεν είναι αριθμητικές.

**1ο ερώτημα – Απλή γραμμική παλινδρόμηση**

Αφού εισάγουμε το dataset μας στην Python, δημιουργούμε το dataframe, εμφανίζουμε το head του(5 πρώτες σειρές) και ορισμένες βασικές πληροφορίες με τη μέθοδο .describe.

Οφείλουμε επίσης να ελέγξουμε για τυχόν ελλείψεις δεδομένων με τη μέθοδο .isnull.

Αφού βεβαιωθούμε ότι δεν υπάρχουν NaN στοιχεία, προχωράμε στην αφαίρεση των ετών 2007 και 2013 από το dataframe, σύμφωνα με τις οδηγίες της εκφώνησης. Δημιουργούμε ένα διάγραμμα κατανομής για την ανεξάρτητη μεταβλητή-στόχο μας, ώστε να έχουμε μια καλύτερη εικόνα.

Εικόνα που περιέχει διάγραμμα, γράφημα, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Προχωράμε στη δημιουργία των 17 σύνολο απλών μοντέλων παλινδρόμησης. Για κάθε ανεξάρτητη μεταβλητή(predictor), υπολογίζουμε τον συντελεστή της παλινδρόμησης(coefficient), καθώς και το p-value και το R-squared. Δημιουργούμε ένα νέο dataframe με αυτά τα metrics για καλύτερη οπτικοποίησή, το οποίο προκύπτει ως εξής:



Άρα έχουμε δημιουργήσει τα 17 ζητούμενα μοντέλα, καθένα εκ των οποίων λαμβάνει τη μορφή :

Όπου ***α*** είναι ο σταθερός όρος της παλινδρόμησης(Intercept) και ***β*** είναι ο συντελεστής παλινδρόμησης(coefficient).

Έπειτα, για να θεωρήσουμε στατιστικά σημαντική την εξάρτηση της μεταβλητής στόχους μας “Life expectancy” από κάθε μία από τις ανεξάρτητες μεταβλητές(predictors), θεωρούμε 95% επίπεδο σημαντικότητας, και άρα θέλουμε να έχουμε p-value < 0.05.

Έτσι προκύπτει το dataframe ως υποσύνολο του dataframe που δημιουργήσαμε και περιλαμβάνει τους predictors που προκύπτουν ως στατιστικά σημαντικοί :

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Οι 2 μεταβλητές που οι δημιούργησαν αντίστοιχα μοντέλα παλινδρόμησης με p-value > 0.05 ήταν οι μεταβλητές “Year” και “Population”.

Χαμηλά p-values, κάτω του επιπέδου σημαντικότητας σηματοδοτούν την απόρριψη της μηδενικής υπόθεσης, και άρα το συμπέρασμα ότι η ανεξάρτητη μεταβλητή επιδρά στην εξαρτημένη μεταβλητή(μεταβλητή-στόχο).

Τέλος, στα metrics συμπεριλήφθηκε και η τιμή του R-squared, που καταδεικνύει το ποσοστό της μεταβλητότητας της μεταβλητής-στόχου μας που ερμηνεύεται από την εκάστοτε ανεξάρτητη μεταβλητή-predictor. Παρατηρούμε αρκετά χαμηλές τιμές R-squared για τα περισσότερα μοντέλα απλής γραμμικής παλινδρόμησης, που σημαίνει ότι παρότι οι εξαρτήσεις της μεταβλητής-στόχου μας από όλες πλην 2 των predictors είναι στατιστικά σημαντικές, προφανώς υπάρχουν άλλοι παράγοντες που δεν λαμβάνει ένα μοντέλο απλής γραμμικής παλινδρόμησης οι οποίοι επηρεάζουν την εξαρτημένη μας μεταβλητή.

Παρουσιάζονται τα διαγράμματα διασποράς που προέκυψαν:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, χάρτης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, διάγραμμα, χάρτης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, χάρτης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, χάρτης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, χάρτης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, διάγραμμα, στιγμιότυπο οθόνης, χάρτης

Περιγραφή που δημιουργήθηκε αυτόματα

**2ο ερώτημα – Πολλαπλή γραμμική παλινδρόμηση**

Ομοίως εκτελούμε την πολλαπλή γραμμική παλινδρόμηση και λαμβάνουμε τα εξής αποτελέσματα:

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Όπως βλέπουμε από τα παραγόμενα αποτελέσματα, οι μεταβλητές για τις οποίες απορρίπτεται η μηδενική υπόθεση(p-value > 0.05) είναι οι: “percentage expenditure”, “Hepatitis B”, “Measles”, “Polio”, “Total expenditure”, “GDP”, “Population”, “thinness 5-9 years”. Παρατηρείται επίσης ότι το R-squared είναι αρκετά υψηλότερο σε σύγκριση με όλα τα R-squared των απλών μοντέλων παλινδρόμησης, κάτι που ήταν αναμενόμενο καθώς το μοντέλο είναι πιο σύνθετο και άρα ερμηνεύει καλύτερα τη μεταβλητότητα της μεταβλητής-στόχου μας.

**3ο ερώτημα – Forward selection**

Για να δημιουργήσουμε το μοντέλο πολλαπλής γραμμικής παλινδρόμησης με τη μέθοδο forward selection που ανήκει στην κατηγορία της stepwise regression, ακολουθούμε τα εξής βήματα :

* Δημιουργούμε ένα κενό μοντέλο και θέτουμε τη λίστα των υποψήφιων predictors(ίδιες μεταβλητές με τα παραπάνω ερωτήματα)
* Προσθέτουμε έναν-έναν τους predictors ξεχωριστά σε κάθε επανάληψη του αλγορίθμου
* Επιλέγουμε ένα metric ως κριτήριο αξιολόγησης(εδώ το R-squared), δηλαδή κάθε ανεξάρτητη μεταβλητή που ελέγχεται η συμπερίληψη της στο μοντέλο και αυξήσει το R-squared, συμπεριλαμβάνεται στο μοντέλο, μέχρι να εξαντληθούν οι υποψήφιες μεταβλητές

Το τελικό μοντέλο που προέκυψε έχει ως εξής :

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, κατάλογος, γραμματοσειρά

Περιγραφή που δημιουργήθηκε αυτόματα

Παρατηρούμε ότι στο μοντέλο συμπεριλήφθηκαν όλες οι ανεξάρτητες μεταβλητές(predictors). Αρκετές από αυτές πάντως έχουν πάρα πολύ χαμηλό coefficient(συντελεστή προσδιορισμού)

**4ο ερώτημα**

Με βάση τα scatter plots που έχουμε δημιουργήσει, σίγουρα υπάρχουν ενδείξεις μη-γραμμικής σχέσης μεταξύ των 4 αυτών μεταβλητών και της μεταβλητής-στόχου μας. Το διάγραμμα διασποράς που προσεγγίζει περισσότερο μια γραμμική σχέση είναι της μεταβλητής “Schooling”.

Εκτελούμε την πολυωνυμική παλινδρόμηση και λαμβάνουμε τα εξής:

Εικόνα που περιέχει κείμενο, διάγραμμα, χάρτης, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, χάρτης, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμμή, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, χάρτης, γραμμή

Περιγραφή που δημιουργήθηκε αυτόματα

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

# 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()

print(df1.columns)

df1cr = df1.corr(numeric\_only=True)

slr\_results = []

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']

for predictor in predictors:

    X = df1[predictor]

    X = sm.add\_constant(X)

    y = df1['Life expectancy ']

    model = sm.OLS(y,X).fit()

    coefficient = model.params[predictor]

    intercept = model.params['const']

    p\_value = model.pvalues[predictor]

    r\_squared = model.rsquared

    slr\_results.append({'Predictor': predictor, 'Intercept': intercept,

                        'Coefficient': coefficient, 'P-value': p\_value, 'R-squared': r\_squared})

slr\_results\_df = pd.DataFrame(slr\_results)

print(slr\_results\_df)

#slr\_results\_df.to\_excel('slr.results.xlsx')

significant\_predictors = slr\_results\_df[slr\_results\_df['P-value'] < 0.05]

print("\nSignificant predictors:")

print(significant\_predictors)

for predictor in predictors:

    plt.figure(figsize=(8, 6))

    sns.scatterplot(x=predictor, y='Life expectancy ', data=df1, )

    plt.title(f'{predictor} vs Life expectancy ')

    plt.xlabel(predictor)

    plt.ylabel('Life expectancy ')

    plt.show()

#Multiple linear regression

X\_all = df1[predictors]

X\_all = sm.add\_constant(X\_all)

y\_all = df1['Life expectancy ']

model\_all = sm.OLS(y\_all, X\_all).fit()

print(model\_all.summary())

#Stepwise regression-forward selection

def forward\_selection(df, target, predictors):

    included\_predictors = []

    best\_r\_squared = -1

    while len(predictors) > 0:

        remaining\_predictors = list(set(predictors) - set(included\_predictors))

        best\_predictor = None

        best\_result = None

        for predictor in remaining\_predictors:

            X = df[included\_predictors + [predictor]]

            X = sm.add\_constant(X)

            y = df[target]

            model = sm.OLS(y, X).fit()

            # Update best predictor and results if R-squared improves

            if model.rsquared > best\_r\_squared:

                best\_predictor = predictor

                best\_result = model

        # Check if adding the best predictor improves the model

        if best\_predictor is not None:

            included\_predictors.append(best\_predictor)

            best\_r\_squared = best\_result.rsquared

            print("Added predictor:", best\_predictor)

            print("Coefficients:", best\_result.params)

            print("R-squared:", best\_r\_squared)

            print()

        else:

            break  # Stop forward selection if no predictor improves the model

    return included\_predictors

included\_predictors = forward\_selection(df1, 'Life expectancy ', predictors)

# Polynomial regression

X\_year = df1['Year'].values.reshape(-1, 1)

X\_alcohol = df1['Alcohol'].values.reshape(-1, 1)

X\_bmi = df1[' BMI '].values.reshape(-1, 1)

X\_schooling = df1['Schooling'].values.reshape(-1, 1)

y = df1['Life expectancy '].values

def fit\_polynomial\_regression(X, y, degree):

    polynomial\_features = PolynomialFeatures(degree=degree)

    X\_poly = polynomial\_features.fit\_transform(X)

    model = LinearRegression()

    model.fit(X\_poly, y)

    return model

# Fit polynomial regression models

degrees = [2, 3]  # Try different degrees for polynomial regression

for X, predictor in zip([X\_year, X\_alcohol, X\_bmi, X\_schooling], ['Year', 'Alcohol', ' BMI ', 'Schooling']):

    plt.figure(figsize=(10, 5))

    plt.scatter(X, y, label='Actual data')

    polynomial\_models= []

    for degree in degrees:

        model = fit\_polynomial\_regression(X, y, degree)

        y\_pred = model.predict(PolynomialFeatures(degree=degree).fit\_transform(X))

        plt.plot(X, y\_pred, label=f'Degree {degree} polynomial')

    plt.title(f'{predictor} vs Life expectancy')

    plt.xlabel(predictor)

    plt.ylabel('Life expectancy')

    plt.legend()

    plt.show()

for degree, model in zip(degrees, polynomial\_models):

        print(f"{predictor} - Degree {degree} Polynomial Model Coefficients:")

        print("Intercept:", model.intercept\_)

        print("Coefficients:", model.coef\_)

        print()