המחלקה למדעי המידע

פרויקט סיכום – Machine Learning

שם קורס: Machine Learning

מספר קורס: 35887-01

שנה אקדמית: תש"פ

סמסטר (בו ניתן הקורס): ב'

תאריך הגשה: 7.9.2020

מוגש למרצה: ד"ר אבשלום אלימלך

שם הסטודנטית: מאשה שמידוב

ת.ז. 320908361

טלפון: 054-2185432

כתובת מייל: mashmidov@gmail.com

שאלה 1:

בשלב הראשון לאחר שעשיתי יבוא לנתונים, הפעלתי את ה dataset.head() כדי לראו את השורות הראשונות ומה הנתונים שיש בהן.

לאחר מכן הפעלתי את dataset.info() וזה היה נראה שאין חוסרים בעמודות, מה שפועל לא נכון כי עברתי על הקובץ א=ואני יודעת שיש בו סימני שאלה בכל מני עמודות.

ביצעתי החלפה של סימן "?" בNaN על ידי replace:

dataset2=dataset.replace('?',"NaN")

ושמרתי את זה בdataset2 ועליו בדקתי האם ניתן לראות איפה יש חסורים וכמה. לצערי באופן הזה קיבלתי שיש אותם נתונים בכל שדה.(מבחינת מספר ואין חוסרים).

שלב נוסף ניסיתי להחליף את "?" במקום ריק בתקווה שזה כן יעזור לי לראות איפה יש חסורים במקום לעשות את זה בקובץ אקסל:

dataset4=dataset.replace('?',” “)

אבל גם זה החזיר לי תמיד את אותה תשובה הבאה למרות שאני יודעת בודאות שבעמודה B יש לי חסורים:

dataset4.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 205 entries, 0 to 204

Data columns (total 25 columns):

symboling 205 non-null int64

normalized-losses 205 non-null object

fuel 205 non-null object

aspiration 205 non-null object

num-of-doors 205 non-null object

לכן החלטתי לעשות את ה replace בתוך האקסל, הכנתי העתק של האקסל המקורי בשם: autos_copy.csv

ולאחר מכן הרצתי עליו info() ושם כבר ניתן היה לראות שיש חוסרים בשדות.

dataset.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 205 entries, 0 to 204

Data columns (total 25 columns):

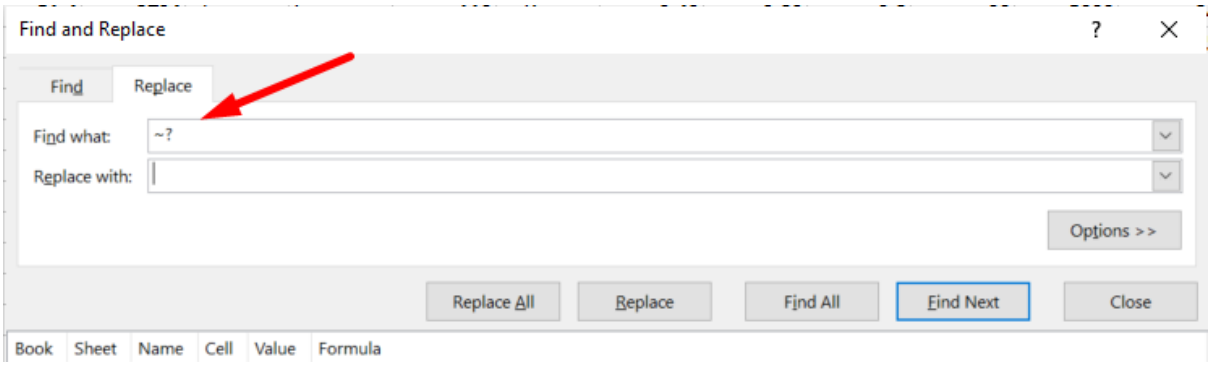
symboling 205 non-null int64

normalized-losses 164 non-null float64

fuel 205 non-null object

aspiration 205 non-null object

כדי להחליף את כל הסימני שאלה בקובץ אקסל בתיבת חיפוש רשמתי ?~ והחלפתי את זה במקום ריק:



אחרי הפעולה כבר ניתן היה לראות באיזה עמודות יש חוסרים:

```
dataset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 25 columns):
symboling          205 non-null int64
normalized-losses   164 non-null float64
fuel               205 non-null object
aspiration         205 non-null object
num-of-doors       203 non-null object
body-type          205 non-null object
drive-wheels       205 non-null object
engine-location    205 non-null object
wheel              205 non-null float64
length             205 non-null float64
width              205 non-null float64
height             205 non-null float64
curb-weight        205 non-null int64
engine-type        205 non-null object
cylinders          205 non-null object
engine-size        205 non-null int64
```

fuel-system 205 non-null object

bore 201 non-null float64

stroke 201 non-null float64

compression 205 non-null float64

horsepower 203 non-null float64

peak 203 non-null float64

city-mpg 205 non-null int64

highway-mpg 205 non-null int64

price 201 non-null float64

dtypes: float64(11), int64(5), object(9)

לאחר פעולה זו ניתן גם לראות איזה סוג נתון חסר.

שלב 2:

מכיוון שחסר לי גם מספרים וגם טקסט חשבתי להשלים בבת אחת את כולם על ידי שימוש ב fillna ולקחת את הנתון החסר משורה שנמצאת מעל הנתון החסר:

```
dataset2=dataset.fillna(method='bfill')
```

שלב 3:

לאחר השלמת חוסרים גזרתי מהDS הזה את משתנה התלוי שלי:

```
y = dataset2.iloc [:, -1 ].values #dependent variable

x = dataset2.iloc [:, :-1 ].values #Independent variable
```

לאחר מכן הוספתי לכל עמודה שיש בה קטגוריה מילולית- קטגוריה מספריתעל ידי LabelEncoder . הפכתי את העמודות בהם יש קטגוריה מילולית לקטגוריה מספרית.

כל שורה עשיתי בנפרד (שורות 20-29) עשיתי את המספור גם לעמודה H למרות שהערך בה היה זהה לכל השורות (התלבטתי אם למחוק אותה אבל בסוף השארתי).

שלב 5: dummy encoding

לאחר מכן ניסיתי לבטל את התלות בין השדות כפי שלמדנו בעזרת oneHotEncoder אבל לא הצלחתי.

```
onehotencoder = OneHotEncoder(categorical_features = [0])

 x =onehotencoder.fit_transform(x).toarray()
```

הוספתי לכל עמודה שעשיתי בה מספור עמודות נוספות שלא תהיה תלות בין המשתנים (במקום onehotencoder בעזרת get_dummies).

```
a=pd.DataFrame(x) #Independent variable as dataframe

x=pd.get_dummies(a, columns=[2, 3, 4, 5, 6, 7, 13, 14,16])
```

ראיתי שאין צורך למחוק עמודה מיותרת עבור על יצירה של עמודות דמה כי הget_dummies מוחק לבד את העמודה המקורית כדי לבטל קשר בין הקטגוריות.

שאלה 2:

כדי להבין איזה פיצ'רים מתוך כל הרשימה נקח לשלב הבא השתמשתי ב Backward elimination, בהתחלה זה לא עבד לי. ראיתי שיש ב stackoverflow פתרון לשגיאה שלי והפתרון זהה עבד גם לי.

```
x=x.astype(np.float64)
```

לאחר מכן יכלתי להריץ את השורות הבאות:

```
x_opt=x[:,
[0,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,
34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53]]

ols=sm.OLS(endog=y,exog=x_opt).fit()

ols.summary()
```

א. כדי להבין איזה משתנים רלוונטיים לחיזוי עשיתי backward elimination.
ב. השיטה עובדת באופן הבא:

כדי להשתמש בשיטה אנחנו צריכים לדעת significant level והמספר הזה ניתן לנו בעבודה – 5 אחוז.

עבור כל משתנה בלתי תלוי אנחנו מחשבים p value (כמה הX הזה תורם לרגרסיה), סוג של ציון עבור כל פרמטר. לאחר שעשינו חישוב של P לכל הפרמטרים מחפשים את הX הכי גבוה וזה אומר שההסתברות שהוא תורם לחיזוי היא נמוכה מאוד ולכן יש להוציא אותו מהרשימה של העמודות בתוך x_opt ולהמשיך לחשב לכל השאר.

הפרמטרים לפי סדר הוצאה וה P שלהם:

משתנה	P שלו	משתנה שהוצאתי ב x_opt
X2	0.956	2
X15	0.829	16
X44	0.841	46
X48	0.902	51
X46	0.864	49
X29	0.808	31
X1	0.775	1
X19	0.757	22
X41	0.706	47
X10	0.739	12
X1	0.617	3
X39	0.524	48
X10	0.379	14
X3	0.377	6
X21	0.446	29
X12	0.407	19
X13	0.324	21
X12	0.450	20
X16	0.321	27

X5	0.351	9
X28	0.642	43
X24	0.532	39
X15	0.508	28
X8	0.331	15
X26	0.269	45
X21	0.269	38
X19	0.975	36
X17	0.874	34
X15	0.549	32
X6	0.161	11
X21	0.147	50
X11	0.103	25
X7	0.083	17
X8	0.074	23

שאלה 3:

האלגוריתם ההראשון שבחרתי להפעיל הוא רגרסיה לינארית מרובת משתנים מכיוון שזה יחסית פשוט יותר ורצוי להתחיל מזה כדי לראות מה הכיוון של הנתונים. סיבה נוספת היא כי הפיצ'רים היו נראים מתאימים לאלגוריתם הזה.

נוסף חישבתי לאלגוריתם הזה mean, std ב crossvalidation כדי לבצע השוואה עם המודל הבא שנריץ.

תוצאה:

-54292337.14938577

87285839.59745863

תוצאות חיזוי ברגרסיה לינארית מורבת משתנים לא נראית מוצלחת:

ב y_pred המספרים יותר נמוכים ממה שיש ב train .

לאחר הפעלתו האלגוריתם רגרסיה לינארית פשוטה ניסיתי היה רגרסיה פולינומית וזה נראה מתאים כי אני יודעת שיש קבוצות שונות של מכוניות למשל: מיני, משפחתי וכו' ואולי כאן זה יהיה יותר מתאים.

ואז נראה מה היחס בינהם מי חוזה יותר טוב.

<u>אלגוריתם רגרסיה פולינומית:</u>

כאן רציתי להריץ רק על פיצ'רים שהיה להם ערך p נמוך בסעיף הקודם:

	mean	std
Degree=2	-5.727325297227207e+19	87285839.59745863
Degree=3	-2.5032705999593595e+18	5.349249726703366e+18
Degree=4	-1.8623776363038925e+19	4.861304365660241e+19

נראה שככל שהדרגה של החזקה עולה המצב משתפר, עצרתי בדרגה 4 כי זה היו מספרים טובים וכי מ 4 כבר לקח זמן למחשב לחשב את הדברים ולכן לא המשכתי מעבר.

לפי cross validation נראה שהגרסיה הפולינומית טובה יותר מרגרסיה מרובת משתנים:

מרובת משתנים		פולינומית	
mean	**std**	**mean**	**std**
-54292337.14938577	87285839.59745863	-1.8623776363038925e+19	4.861304365660241e+19

הלאגוריתם הנבחר הוא גרסיה פולינומית, משתנים במשוואה:

B_0

ו- b_i זה המקדמים של ה x עבור כל חזקה שלו עד 4 כולל:

```
In [134]: b_0
Out[134]: -66166215.23252612

In [135]: b_i
Out[135]:
array([ 9.71201505e+00, -4.40221928e+00, -2.98175096e+00, ...,
        0.00000000e+00,  0.00000000e+00, -5.89991345e-15])
```

שאלה 4:

הערך המוסף שיכול להיות ליצרני המכוניות מהחיזוי הוא הערכה לעלות המכונית שהיא הולכת לייצר על סמך הפיצ'רים שהיא רוצה שיהיו בה. זה בעצם נותן לה כיוון ויש בזה המון כי כבר לפני הייצור הם יכולים לדעת מה יהיה מחיר שלה (על סמך התחזית) ובהתאם לזה להחליט החלטות.

הקוד נמצא כאן מטה וגם בגיט.

```
#Q1

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd
```

```
dataset=pd.read_csv("C:\\Users\\97254\\Documents\\ML\\autos_copy.csv")
dataset.head(6) #check the data

#step 2 > fiil the empty fields
dataset2=dataset.fillna(method='bfill') #fill in missing data for all columns

#step3 > split the data in 2 arrays:
x = dataset2.iloc [:,0:-1 ].values #Independent variable
y = dataset2.iloc [:, -1 ].values #dependent variable

#step 4> encoding categorical data to numbers
from sklearn.preprocessing import Imputer
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()

x[:, 2]=labelencoder_X.fit_transform(x[:, 2]) #fuel column
x[:, 3]=labelencoder_X.fit_transform(x[:, 3]) #aspiration column
x[:, 4]=labelencoder_X.fit_transform(x[:, 4]) #num_of_doors
x[:, 5]=labelencoder_X.fit_transform(x[:, 5]) #body_type
x[:, 6]=labelencoder_X.fit_transform(x[:, 6]) #drive_wheels
x[:, 7]=labelencoder_X.fit_transform(x[:, 7]) #drive_wheels

x[:, 13]=labelencoder_X.fit_transform(x[:, 13]) #engine_type
x[:, 14]=labelencoder_X.fit_transform(x[:, 14]) #cylinders
x[:, 16]=labelencoder_X.fit_transform(x[:, 16]) #fuel_system

#step 5 > dummy encoding
a=pd.DataFrame(x) #Independent variable as dataframe
x=pd.get_dummies(a, columns=[2, 3, 4, 5, 6, 7, 13, 14,16])

#Q2
```

```
#backward elimination - preparation
import statsmodels.regression.linear_model as sm
import statsmodels.tools.tools as tl
x=np.append(arr=np.ones((205,1)).astype(int),values=x,axis=1) #add constant

x=x.astype(np.float64)
x_opt=x[:, [0,4,5,7,8,10,13,18,24,26,30,33,35,37,40,41,42,44,52]]
ols=sm.OLS(endog=y,exog=x_opt).fit()
ols.summary()

#Q3
#splitting data: training & test set א+ב
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_opt, y, test_size = 0.2, random_state = 0)

#linear regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)
y_pred=regressor.predict(x_test)
b_0=regressor.intercept_
b_i=regressor.coef_

#cross validation to regression

from sklearn.model_selection import cross_val_score
accuracies=cross_val_score(regressor,x,y,cv=10,scoring='neg_mean_squared_error')
print(accuracies.mean())
print(accuracies.std())

#polynomial regression till degree=4
```

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg=PolynomialFeatures(degree=4)
x_poly=poly_reg.fit_transform(x)

lin_reg_2=LinearRegression()
lin_reg_2.fit(x_poly,y)

accuracies=cross_val_score(lin_reg_2,x_poly,y,cv=10,scoring='neg_mean_squared_error')
print(accuracies.mean())
print(accuracies.std())
b_0=lin_reg_2.intercept_
b_i=lin_reg_2.coef_
```