Homework -3 Multiple Linear Regression

Name : Shadeeb Hossain

ID: sh7492

1. (a) The target variable could be ***"Quantity of Products sold"*** .

(b) The suggested Linear Model could be equivalent to :

$$y = \beta_1 + \beta_2 x_2 - \beta_3 x_2 - \beta_4 x_2$$

Where, β_1 is the Quantity of Products sold in 1 month.
β_2 is the frequency of the occurrence of the word "good" .
β_3 is the frequency of the occurrence of the word "bad" .
β_4 is the frequency of the occurrence of the word "does not work" .
x_2 is the number of total reviews .

(c) **Normalization** can be used as shown below:

```
user_input=int(input("Please enter the total score (out of 5 or 10):"))
print("The total score is out of :",(user_input))
```

```
Please enter the total score (out of 5 or 10):5
The total score is out of : 5
```

```
if user_input==5:
    actual_score=int(input("Enter how much the product scored out of 5:"))
    score=actual_score/5
    print("The normalized score is:",score)
else:
     actual_score=int(input("Enter how much the product scored out of 10:"))
     score=(actual_score/10)*2
     print("The normalized score is:",score)
```

```
Enter how much the product scored out of 5:4
The normalized score is: 0.8
```

(d) For missing data. dropna() command can be used as shown below

```
# (d)
df1=df[['review']]
df2=df1.dropna()
```

(e) Fraction of reviews with the word *"good"* is a better choice between the two. This is because it accounts for normalization. Some products might have a higher number of "*good"* reviews, but their *"bad"* reviews might also be significantly higher. This error can be reduced by opting for fraction of reviews with the word *"good"*.

2. (a) Linear model for y in terms of x_1 and x_2 is:

$$y_i = \beta_o + \beta_1 x_{1i} + \beta_2 x_{2i}$$

(b)

```
b=np.array([[0,0,1],
            [0,1,4],
            [1,0,3],
            [1,1,7]])
# Save array into a CSV file
np.savetxt("data1.csv",b)
names=['x1','x2','y']
df=pd.read_csv('data1.csv',header=None,delim_whitespace=True,names=names,na_values='?')
df.head()
df2=df['y']
df1=df[['x1','x2']]
print(df1)
ym=np.mean(df2)
y1=df2-ym
xm=np.mean(df1,axis=0)
x1=df1-xm[None,:]
# Computing the correlation
syy=np.mean(y1**2)
sxx=np.mean(x1**2,axis=0)
sxy=np.mean(x1*y1[:,None],axis=0)
# Computing the coefficients
beta1=sxy/sxx
beta0=ym-beta1*xm
Rsq=sxy**2/sxx/syy
print("The value of beta0 is", beta0)
print("The value of beta1 is",beta1)
print("The value of Rsq is",Rsq)
```

```
    x1   x2
0  0.0  0.0
1  0.0  1.0
2  1.0  0.0
3  1.0  1.0
The value of beta0 is x1    2.5
x2    2.0
dtype: float64
The value of beta1 is x1    2.5
x2    3.5
dtype: float64
The value of Rsq is x1    0.333333
x2    0.653333
```

3.

3(a) $\hat{y} = (a_1 x_1 + a_2 x_2)\, e^{-x_1 - x_2}$

We know, $\boxed{y = Ae^{kx} \Rightarrow \log y = kx + \log A}$

Hence,

$$\boxed{\log \hat{y} = -(x_1 + x_2) + \log(a_1 x_1 + a_2 x_2)}$$

Hence. $\log y_i = -(x_{i1} + x_{i2}) + \log(a_1 x_{1i} + a_2 x_{2i}$

$$\boxed{\log \begin{bmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{in} \end{bmatrix} = \log \begin{bmatrix} a_{i11} x_{i1} + a_{2i} x_{i2} \\ a_{i21} x_{i21} + a_{2i} x_{i21} \\ \vdots \\ a_{in} x_{in} + a_{ni} x_{in} \end{bmatrix} - \begin{bmatrix} x_{i11} + x_{22} \\ x_{i11} + x_{21i} \\ \vdots \\ \ \end{bmatrix}}$$

3(b) $\hat{y} = a_1 + a_2 x$ if $x < 1$

$= a_3 + a_4 x$ if $x \geq 1$

Hence, y can be written as

$$\hat{Y} = [1 \quad x] \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \quad \text{for } x < 1$$

$$\hat{y} = [1 \quad x] \begin{bmatrix} a_3 \\ a_4 \end{bmatrix} \quad \text{for } x >= 1$$

So,

$$\boxed{\hat{Y} = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \text{ for } x < 1. \quad \text{or} \begin{bmatrix} a_3 \\ a_4 \end{bmatrix} \text{ for } x >= 1.}$$

$$\boxed{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{i1} \\ 1 & x_{i2} \\ 1 & x_{i3} \end{bmatrix} \begin{bmatrix} a_{1i} \\ a_{2i} \end{bmatrix} \text{ for } x < 1 \quad \begin{bmatrix} a_{3i} \\ a_{4i} \end{bmatrix} \text{ for } x >= 1}$$

3(c) $\hat{y} = (1 + a_1 x_1) e^{-x_2 + a_2}.$

$$\log \hat{y} = -x_2 + a_2 + \log(1 + a_1 x_i)$$

$$\log \hat{y}_i = -x_2 + a_2 + \log(1 + a_1 x_1)$$

$$\boxed{\log \begin{bmatrix} \hat{y}_{i1} \\ y_{i2} \\ \vdots \\ y_{in} \end{bmatrix} = \cancel{\begin{bmatrix} -1 + a_2 \end{bmatrix}} \begin{bmatrix} -x_2 + 1 \end{bmatrix} \begin{bmatrix} 1 \\ a_2 \end{bmatrix} + \log \begin{bmatrix} 1 + x_1 \end{bmatrix} \begin{bmatrix} a_1 \end{bmatrix}}$$

Please turn to next page

Page left intentionally blank

4(a)

$$y_k = \sum_{j=1}^{M} a_j y_{k-j} + \sum_{j=0}^{N} b_j x_{k-j} + \epsilon_k .$$

where a_j & b_j are coefficients

$$\boxed{y_k = \beta_0 + \beta_1 y_{k-j} + \beta_2 x_{k-j} +} \; \text{- - -}$$

There are three unknown parameters.

4(b)

$$y = A\beta + \epsilon$$

$$y_k = a_1 y_{k-1} + b_0 x_k + \epsilon_k + a_2 y_{k-2} + b_1 x_{k-1}$$
$$+ a_3 y_{k-3} + b_3 x_{k-3} + \text{- - -} \; a_M y_{k-M} + b_N x_{k-N}$$

In terms of $y = A\beta + \epsilon$

where $k = 0 \text{ - - - - } T-1$

$$\cancel{y_0 = a_1 y_{-1} + b_0 x_0 + \epsilon_0 + a_2 y}$$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{T-1} \end{bmatrix} = \begin{bmatrix} 1 & y_{k-1} & x_k \\ 1 & y_{k-2} & x_{k-1} \\ 1 & y_{k-3} & x_{k-2} \\ \vdots & & \vdots \\ \vdots & y_{T-1-M} & x_{T-1-N} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_{T-1} \end{bmatrix}$$

Please continue to the next page for Number 5

Blank Page

5.(a) (i) $x_k = \sum_{l=1}^{L} a_l \cos(\Omega_l k) + b_l \sin(\Omega_l k)$

where L = no. of tones present

Ω_l = tonal frequencies

a_l & b_l are coefficients

x_k where $k = 0 \ldots N-1$.

$$x_k = a_0 \cos(\Omega_0 \cdot 0) + b_0 \sin(\Omega_0 \cdot 0) + a_1 \cos(\Omega_1 \cdot 1) + b_1 \sin(\Omega_1 \cdot 1) + a_2 \cos(\Omega_2 \cdot 2) + b_2 \sin(\Omega_2 \cdot 2) + \ldots$$

$$\ldots a_N \cos(\Omega_N \cdot N) + b_N \sin(\Omega_N \cdot N).$$

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \cos + b_1 \sin \\ a_2 \cos 2 + b_2 \sin(2) \\ \vdots \\ a_N \cos N + b_N \sin(N) \end{bmatrix} \begin{bmatrix} \Omega_0 \\ \Omega_1 \\ \Omega_2 \\ \vdots \\ \Omega_N \end{bmatrix}$$

$$\boxed{x = A \cdot \beta}$$

(ii) The lst sq method can be used to solve $x = A\beta$.
It returns the least square solution to a linear matrix equation.

```
N = 100
ones = np.ones((N, 1))
A = np.hstack((ones, ohm))   # where ohm is equivalent to x
A.shape.
```

out = np. linalg.lstsq (A , x , rcond = None) # where x is equivalent to y.

beta = out [0]

beta.

The value of β (beta) will give the coefficients a_ℓ & b_ℓ for this model (Ans).

(b) NO, if Ω_ℓ is not known, it cannot be a linear regression problem. This is because for linear regression problem the X train needs to be known to apply.

regr = linear_model. LinearRegression()

6

```
#(a)
 import numpy as np
n=X.shape [0]
yhat=np.zeros(n)
# Without 'For' loop and using Vectorization and Python Broadcasting
yhat=np.sum((beta[0]*X[:,0])+(beta[1]*X[:,1])+(beta[2]*X[:,1]*X[:,2]),axis=1) # axis=1 since it is along the rows
```

```
#(b)
n=len(X)
m=len(alpha)
yhat=np.zeros(n)
yhat+=np.sum(alpha*np.exp(-beta*X),axis=1)
```

```
#(c)
n,d=X.shape
m,d=y.shape
dist=np.zeros((n,m))
dist=np.sum((X[:,None]-y[None,:,:])**2,axis=2)
```