Vivian Joseph

(20BCE0777)

Machine Learning

Lab Da-2

Regression Implementation

Q1) Create your own dataset for given sample column name (minimum 30 samples).

1	Year	Model	Price	buy
2	2010	1.1	120	no
3	2011	1.2	150	no
4	2011	1.8	160	no
5	2012	2	145	no
6	2012	2.1	150	no
7	2012	2.4	160	no
8	2013	2.6	190	no
9	2013	2.8	210	no
10	2014	2.9	220	no
11	2014	3	200	no
12	2015	3.1	210	no
13	2015	3.2	215	yes
14	2016	3.3	220	yes
15	2016	3.4	260	no
16	2017	3.5	270	no
17	2017	3.7	280	no
18	2018	4	250	yes
19	2018	4.2	264	yes
20	2018	4.4	270	yes
21	2019	4.6	282	no
22	2019	4.7	295	no
23	2020	4.8	303	no
24	2020	4.82	312	no
25	2020	4.9	323	yes
26	2021	4.95	337	yes
27	2021	5	300	yes
28	2022	5.1	315	yes
29	2022	5.2	340	yes
30	2022	5.4	360	no
31	2023	5.8	399	yes

(make csv file)

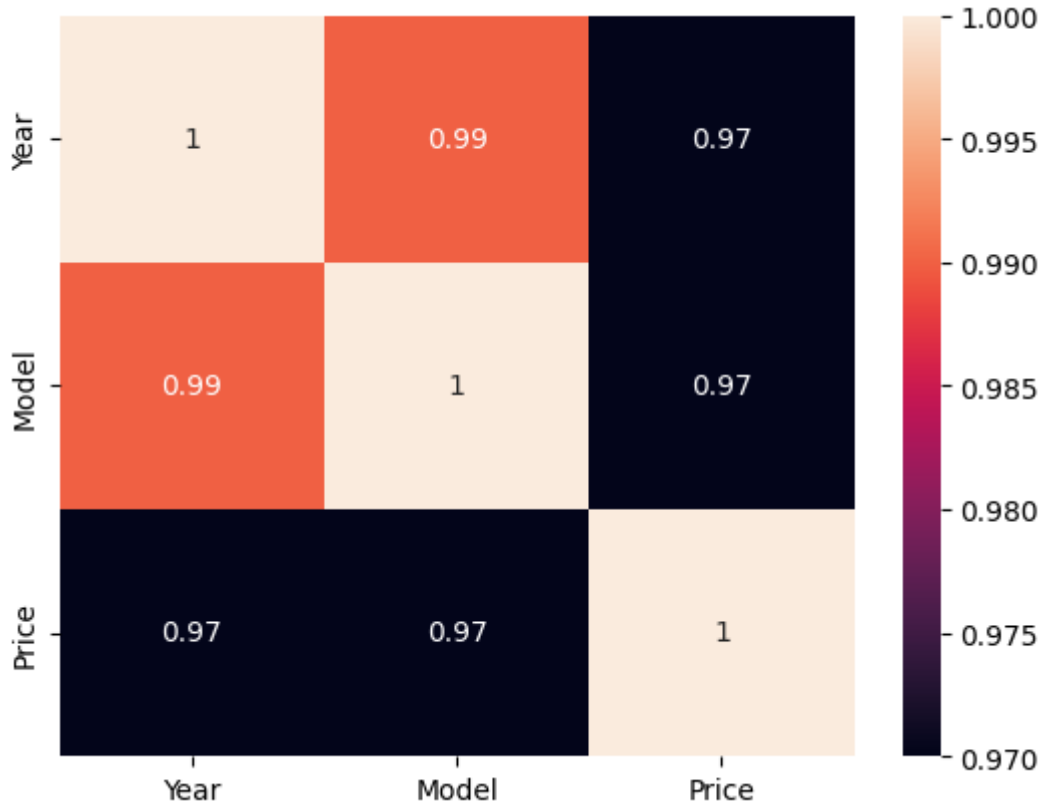
Q2) Apply linear regression (Hot Code) and visualize the same

Take the Appropriate variables as target and attributes

```
X=df[["Year","Model"]]
Y=df["Price"]
```

Check correlation matrix

```
correlation_matrix = df.corr().round(2)
sn.heatmap(data=correlation_matrix, annot=True)
```



Write a function to fit the line

Price = b0 + b1(year) + b2(Model)

```
# Taking Price = b0 + b1(Year) + b2(Model)
def linReg(X1,X2,Y):
    N=len(Y)

    X1_square=X1**2
    sig_x1_2= X1_square.sum() - (X1.sum()**2/N)
    X2_square=X2**2
    sig_x2_2= X2_square.sum() - (X2.sum()**2/N)
    X1_X2=X1*X2
    sig_x1x2= X1_X2.sum() - (X1.sum()*X2.sum()/N)
    X1_Y=X1*Y
    sig_x1y= X1_Y.sum() - (X1.sum()*Y.sum()/N)
    X2_Y=  X2*Y
```

```
    sig_x2y= X2_Y.sum() - (X2.sum()*Y.sum()/N)

    b1_num=((sig_x2_2)*(sig_x1y)-(sig_x1x2)*(sig_x2y))
    b2_num=((sig_x1_2)*(sig_x2y)-(sig_x1x2)*(sig_x1y))
    denom= ((sig_x1_2)*(sig_x2_2)-(sig_x1x2)**2)
    b1=b1_num/denom
    b2=b2_num/denom
    b0=Y.mean()-b1*X1.mean()-b2*X2.mean()
    return b0, b1, b2
```

Call the function

```
b0,b1,b2=(linReg(X["Year"],X["Model"],Y))
```

We get values for b0,b1,b2

```
print(b0,b1,b2)
```

b_0 b_1 b_2

```
-22907.63348685766 11.44668731129901 20.032514184125056
```

Put values in the line to get any two points on the line (in order to plot later)

```
print(b0+b1*2010+b2*1)
print(b0+b1*2023+b2*6)
```

```
120.24052303747627
369.2100290049883
```

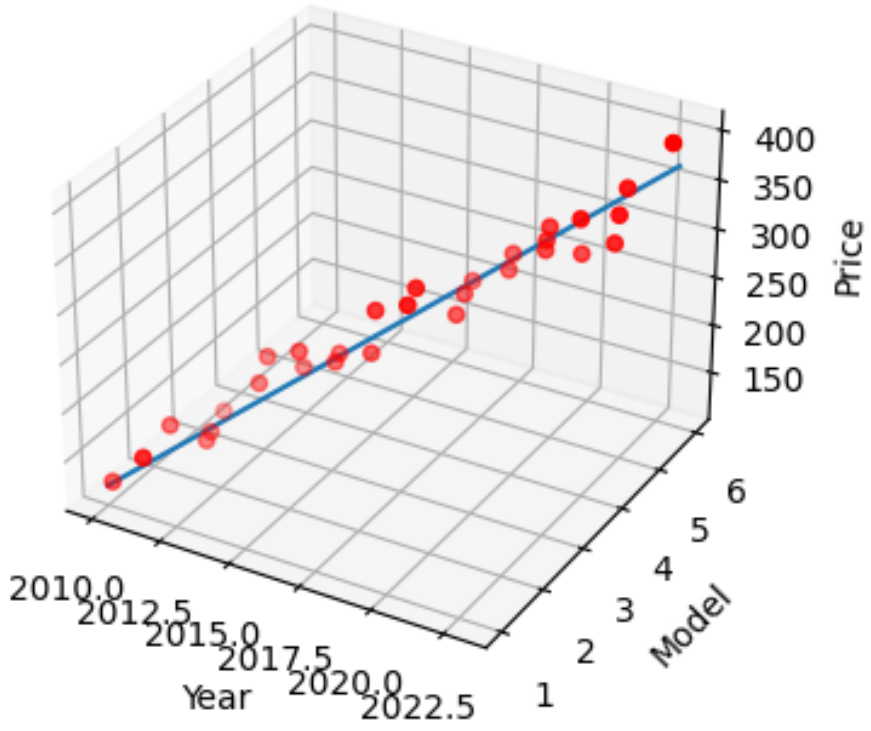
Plot it on a graph to see if the line fits the given data

```
fig = plt.figure(figsize=(4,4))
ax = fig.add_subplot(111, projection='3d')

x=np.array([2010,2023])
y=np.array([1,6])
z=np.array([120.24,369.21])

ax.scatter(X["Year"],X["Model"],Y, c='r')
ax.plot3D(x,y,z)

plt.show()
```



(line fits data pretty well)

Q3) Apply Polynomial regression and visualize the same

Import relevant functions

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

make a function f(x1,x2)= a0+(a1*x1^2)+(a2*x2^2)+(a3*x)+(a4*y)+(a5*xy) and fit the new features to the given data

```
poly=PolynomialFeatures(degree=2)
X_poly=poly.fit_transform(X)
reg=LinearRegression().fit(X_poly,df['Price'])
```

Plot the formed polynomial along with the given data points

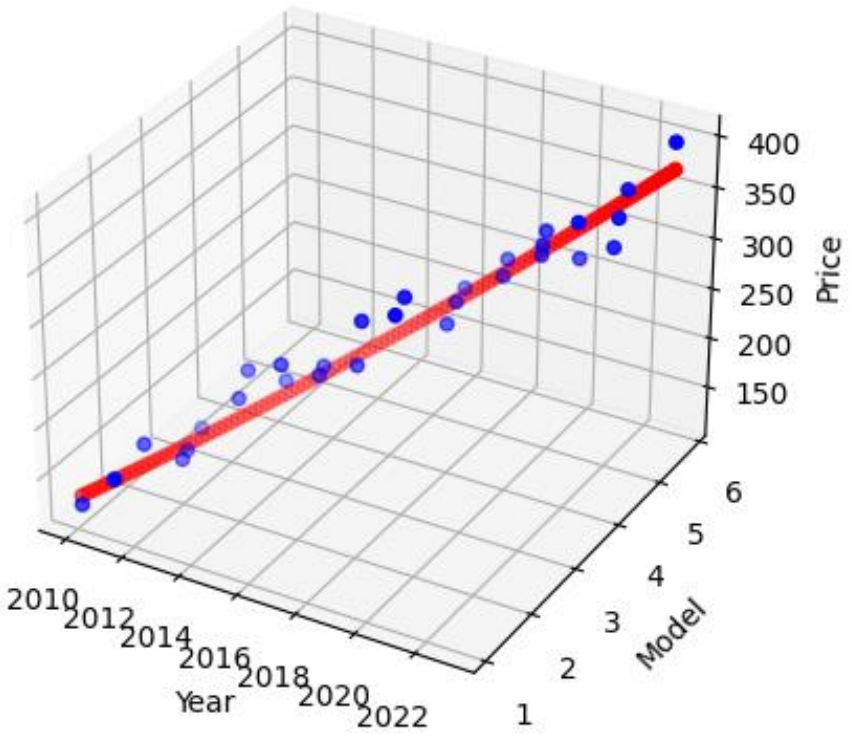
```
fig=plt.figure()
ax=fig.add_subplot(111,projection='3d')
ax.scatter(df['Year'],df['Model'],df['Price'],color='b',marker='o')

x1_pred=np.linspace(df['Year'].min(),df['Year'].max(),100).reshape(-1,1)
x2_pred=np.linspace(df['Model'].min(),df['Model'].max(),100).reshape(-1,1)
x_pred=np.hstack((x1_pred,x2_pred))
y_pred=reg.predict(poly.transform(x_pred))

ax.scatter(x1_pred,x2_pred,pd.DataFrame(y_pred),color='r',marker='o')
```

```
ax.set_xlabel("Year")
ax.set_ylabel("Model")
ax.set_zlabel("Price")

plt.show()
```



(Polynomial comes close to a straight line but is not exactly linear – as linear line fits the given data very well)

The actual coefficients are as follows:

```
print(reg.coef_)
```

```
[ 0.00000000e+00 -1.74139235e+03  9.25983752e+03  4.37927373e-01
 -4.61779954e+00  1.14979849e+01]
```

Best fit= degree 1 (linear)

Q4) Logistic regression add one attribute(target) user is interested to buy/not buy and visualize the same

Prepare the data and fit it to the logistic regression function

```
X=df[['Year','Price']]
Y=df['buy']
clf= LogisticRegression()
clf.fit(X,Y)
coef = clf.coef_[0]
intercept = clf.intercept_[0]
print(coef,intercept)
```

```
[-0.00248297  0.01712361] -1.7906123016963876e-06
```

Visualize the data with the logistic regression line

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x1_min, x1_max = X['Year'].min(), X['Year'].max()
x2_min, x2_max = X['Price'].min(), X['Price'].max()
xx1, xx2 = np.meshgrid(np.linspace(x1_min, x1_max), np.linspace(x2_min, x2_max))
grid = np.c_[xx1.ravel(), xx2.ravel()]
probs = clf.predict_proba(grid)[:, 1].reshape(xx1.shape)
ax.contour(xx1, xx2, probs, levels=[.5], cmap="Greys", vmin=0, vmax=.6)

ax.scatter(df['Year'], df['Price'], df['buy'].map({'yes':1,'no':0}), c='r', marker='o')
ax.set_xlabel('Year')
ax.set_ylabel('Price')
ax.set_zlabel('buy or not')
ax.set_title('Logistic Regression Result Visualization')
plt.show()
```

Logistic Regression Result Visualization