**Vivian Joseph**

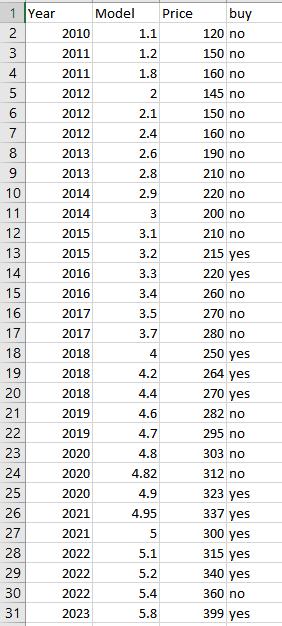
**(20BCE0777)**

**Machine Learning**

**Lab Da-2**

**Regression Implementation**

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

** (make csv file)**



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

Take the Appropriate variables as target and attributes

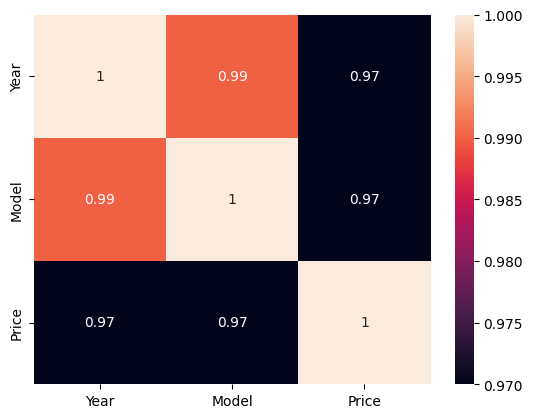
X=df[["Year","Model"]]

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)





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)





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])

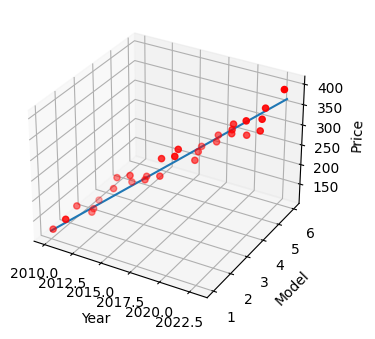
y=np.array([1,6])

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))

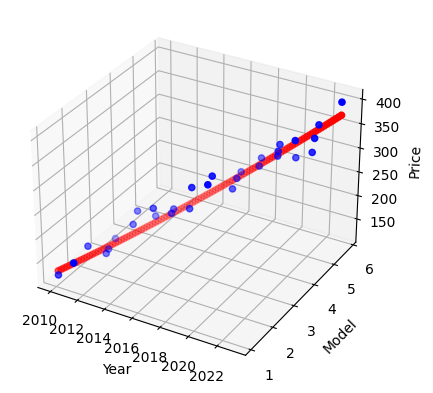
ax.scatter(x1\_pred,x2\_pred,pd.DataFrame(y\_pred),color='r',marker='o')

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\_)



**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)



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()

