CSE 4020 Machine Learning
Lab Assessment - 1

Sujay Kumar M 20BDS0294
Computer Science Engineering with Specialization with DataScience
sujaykumarreddy.m2020@vitstudent.ac.in
https://github.com/sujaykumarmag/CSE4020

February 1, 2023

1 Making a DataSet

I made a Laptop with prices Dataset with 3 Columns and 60 rows

Out[3]:

	Model Year	Model Name	Price (in INR)
0	2021	Apple	154000
1	2020	Apple	150000
2	2020	Apple	49000
3	2019	Apple	48000
4	2022	Apple	47000
5	2018	Apple	45000
6	2017	Apple	47000
7	2016	Apple	59999
8	2015	Apple	39999
9	2014	Apple	25000
10	2013	Apple	40000
11	2012	Apple	37000
12	2011	Apple	100000
13	2010	Apple	67000
14	2009	Apple	49123
15	2008	Apple	36000
16	2007	Apple	34000
17	2006	Apple	54000
18	2005	Apple	50000
19	2004	Apple	30000
20	2021	Lenovo	45000
21	2020	Lenovo	47000
22	2020	Lenovo	56000
23	2019	Lenovo	39999
24	2022	Lenovo	45999
27	2016	Lenovo	43000
28	2015	Lenovo	51000
29	2014	Lenovo	34000
30	2013	Lenovo	35000
31	2012	Lenovo	48000
32	2011	Lenovo	45000
33	2010	Lenovo	31000
34	2009	Lenovo	29999
35	2008	Lenovo	34500
36	2007	Lenovo	46711
37	2006	Lenovo	38766
38	2005	Lenovo	43200
39	2004	Lenovo	35000
40	2021	Dell	50000
41	2020	Dell	46790
42	2020	Dell	39450
43	2019	Dell	40235
44	2022	Dell	42785
45	2018	Dell	39865
46	2017	Dell	45000
47	2016	Dell	43785
48	2015	Dell	46599
49	2014	Dell	54000
50	2013	Dell	52000
51	2012	Dell	61000
52	2011	Dell	37839
53	2010	Dell	51000
54	2009	Dell	39999
55	2008	Dell	42000
56	2007	Dell	58974
57	2006	Dell	59999
58	2005	Dell	41990
59	2004	Dell	34897

2 Data Pre-Processing

1. I used Indexing Order to identify each model name.
2. For more than 3 attributes its best to use one-hot encoding but multiple regression for different variables is hard to find.

2.1 Algorithm for Imputing the new column from old column

1. Create an array and append the values according to each row.
2. Create a new Column and insert it as a dataframe.
3. Drop the Old Column

```
y = []
for i in range(len(data)):
  if(data["Model Name"][i]=="Apple"):
    y.append(1)
  elif (data["Model Name"][i] == "Lenovo"):
    y.append(2)
  else:
    y.append(3)

datax = pd.DataFrame(y, columns =['Model Name'])
x = datax.size
c = 0
data["Model Name_opt"] = datax
while(c<x):
  data["Model Name_opt"].iloc[c]=datax["Model Name"].iloc[c]
  c=c+1

data = data.drop(["Model Name"],axis=1)
```

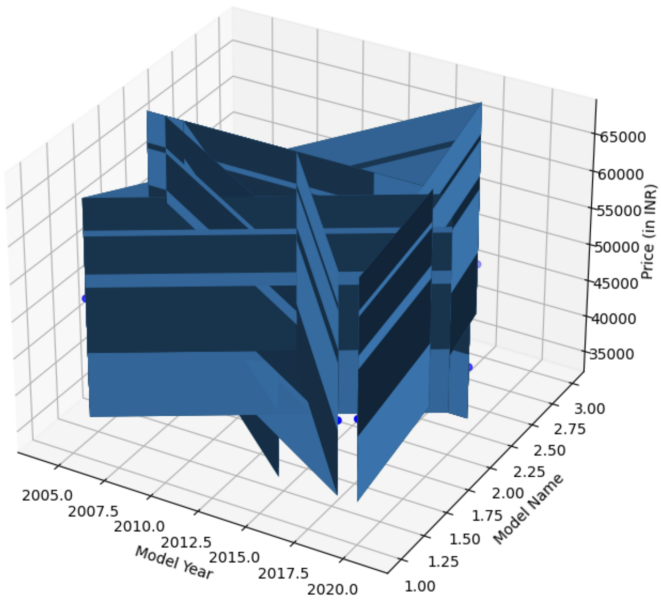
Out[65]:

	Model Year	Price (in INR)	Model Name_opt
0	2021	154000	1
1	2020	150000	1
2	2020	49000	1
3	2019	48000	1
4	2022	47000	1
5	2018	45000	1
6	2017	47000	1
7	2016	59999	1
8	2015	39999	1
9	2014	25000	1

3 Multi-Linear Regression (Hot Code)

```
1
2  class Regression:
3    # To store the No of dependent variables
4    def __init__(self,no_of_vars):
5      self.size = no_of_vars
6
7    # Assumption : Only the dependent and target variables are passed separately
8    def fit(self,x1,x2,y):
9      a = self.double_summation(x2,x2)
10     b = self.double_summation(x1,y)
11     c = self.double_summation(x1,x2)
12     d = self.double_summation(x2,y)
13     e = self.double_summation(x1,x1)
14     f = c*c
15     b1 = ((a*b)-(c*d))/((a*c)-(f))
16     b2 = ((e*d)-(c*b))/((a*c)-(f))
17     b0 = (y.sum()/self.size)-(b1*(x1.sum()/self.size))-(b2*(x2.sum()/self.size))
18     arr = np.array([b0,b1,b2]);
19     self.eq = arr
20     return arr
21
22   # For Printing the Equation
23   def __str__(self):
24     x = str(self.eq[0][0]) +" + "+str(self.eq[1][0])+"(x1)"+" + "+str(self.eq[2][0])+"(x2
       )"
25     return x
26
27   # This is an Helper Method
28   def double_summation(self,x1,x2):
29     sumz = 0
30     for i in range(self.size):
31       sumz = sumz + (x1[i]*x2[i])
32     return sumz
33
34   # Assumption : Give me as Numbers not as a np-array
35   def predict(self,x1,x2):
36     return (self.eq[0]) + ((self.eq[1])*x1) + ((self.eq[2])*x2)
```

```
[-2442558.02113646]
[[ 1246.12182825 -6227.9807602  -5484.68512565]]
```




```
48900.90611248054 + -0.976444119320599(x1) + 0.004997180677577198(x2)
```

Results Interferred :

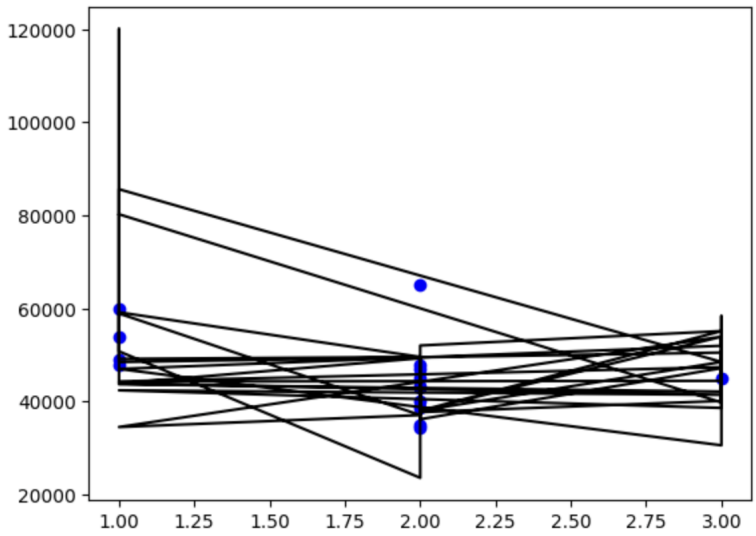
1. As all the datapoints are covered its known as "Overfitting"
2. Conclusion : Not a good Algorithm for this Dataset.

4 Polynomial Regression

```
polynomial_features1 = PolynomialFeatures(degree=8)
x_poly1 = polynomial_features1.fit_transform(X_train)
model1 = LinearRegression()
model1.fit(x_poly1, y_train)
y_poly_pred1 = model1.predict(x_poly1)
rmse1 = np.sqrt(mean_squared_error(y_train,y_poly_pred1))
r21 = r2_score(y_train,y_poly_pred1)
print(rmse1)
print(r21)
plt.scatter(X_test.iloc[:,1], y_test, color ='b')
plt.plot(X_train.iloc[:,1], y_poly_pred1, color ='k')
```

```
18764.150122169616
0.4454562730127314
```

```
Out[87]: [<matplotlib.lines.Line2D at 0x16b14b4f0>]
```



Results Interferred :

1. As all the datapoints are covered its known as "Overfitting"
2. The Optimized Degree is 8, after 8 the R$\hat{2}$ value becomes almost same.
3. Conclusion : Not a good Algorithm for this Dataset.

5 Logistic Regression

5.1 Applying the Class Label

1. Create a new array
2. Create a new Column in the dataset and insert the column

```
arr = ["Yes","No","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","
  Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","
  Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","
  Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","Yes","No","Yes","
  Yes"]
```

```
datax1 = pd.DataFrame(arr, columns =["Buying_Option"])
x=datax1.size
c=0
data["Buy_Option"]=datax1
while(c<x):
data["Buy_Option"].iloc[c]=datax1["Buying_Option"].iloc[c]
c=c+1
```

	Model Year	Price (in INR)	Model Name_opt	Buy_Option
0	2021	154000	1	Yes
1	2020	150000	1	No
2	2020	49000	1	Yes
3	2019	48000	1	No
4	2022	47000	1	Yes
5	2018	45000	1	Yes
6	2017	47000	1	No

5.2 Fitting the Model

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
  random_state = 0)

sc_x = StandardScaler()
xtrain = sc_x.fit_transform(X_train)
xtest = sc_x.transform(X_test)

classifier = LogisticRegression(random_state = 0)
classifier.fit(xtrain, y_train)

y_pred = classifier.predict(xtest)

cm = confusion_matrix(y_test, y_pred)

print ("Confusion Matrix : \n", cm)
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```