Vivian Joseph

(20BCE0777)

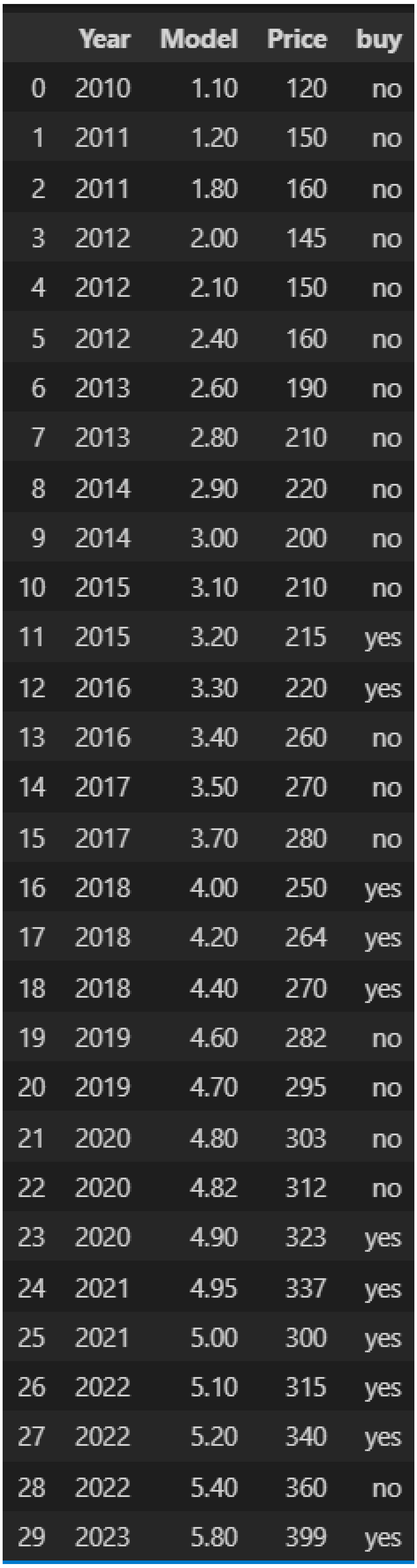
# Machine Learning

Lab Da-3

## Single Layer Perceptron

## and KNN Classifier

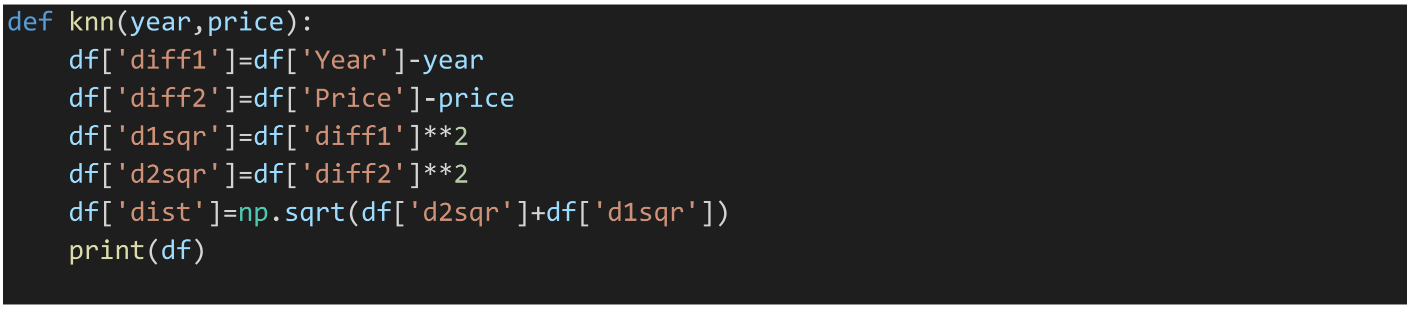
**Dataset:**



| | Year | Model | Price | buy |
|---|---|---|---|---|
| 0 | 2010 | 1.10 | 120 | no |
| 1 | 2011 | 1.20 | 150 | no |
| 2 | 2011 | 1.80 | 160 | no |
| 3 | 2012 | 2.00 | 145 | no |
| 4 | 2012 | 2.10 | 150 | no |
| 5 | 2012 | 2.40 | 160 | no |
| 6 | 2013 | 2.60 | 190 | no |
| 7 | 2013 | 2.80 | 210 | no |
| 8 | 2014 | 2.90 | 220 | no |
| 9 | 2014 | 3.00 | 200 | no |
| 10 | 2015 | 3.10 | 210 | no |
| 11 | 2015 | 3.20 | 215 | yes |
| 12 | 2016 | 3.30 | 220 | yes |
| 13 | 2016 | 3.40 | 260 | no |
| 14 | 2017 | 3.50 | 270 | no |
| 15 | 2017 | 3.70 | 280 | no |
| 16 | 2018 | 4.00 | 250 | yes |
| 17 | 2018 | 4.20 | 264 | yes |
| 18 | 2018 | 4.40 | 270 | yes |
| 19 | 2019 | 4.60 | 282 | no |
| 20 | 2019 | 4.70 | 295 | no |
| 21 | 2020 | 4.80 | 303 | no |
| 22 | 2020 | 4.82 | 312 | no |
| 23 | 2020 | 4.90 | 323 | yes |
| 24 | 2021 | 4.95 | 337 | yes |
| 25 | 2021 | 5.00 | 300 | yes |
| 26 | 2022 | 5.10 | 315 | yes |
| 27 | 2022 | 5.20 | 340 | yes |
| 28 | 2022 | 5.40 | 360 | no |
| 29 | 2023 | 5.80 | 399 | yes |

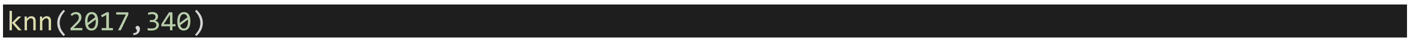
## Q1) Apply KNN classification on the dataset and visualize the same

## Making a function to check the distances of an input point from all points in the dataset

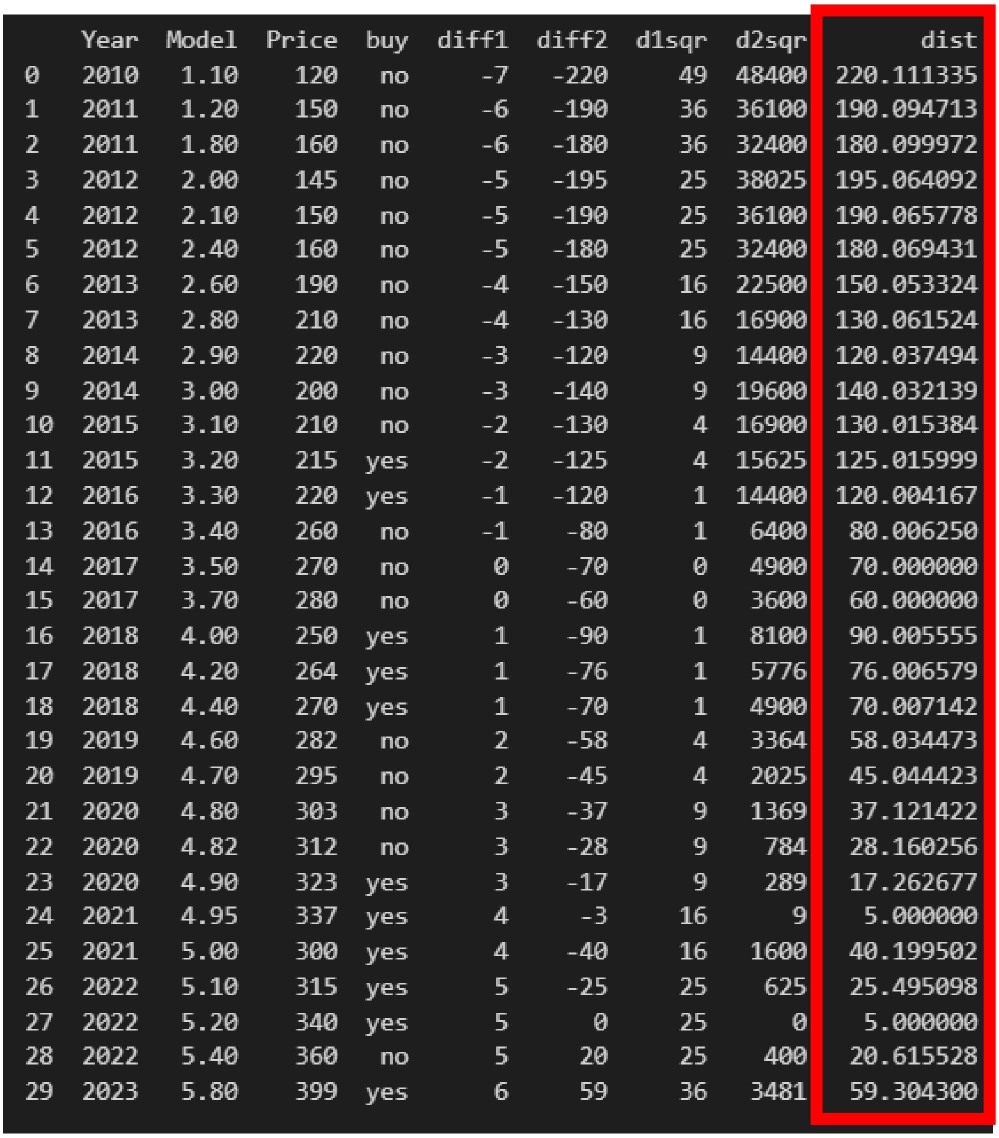


```
def knn(year,price):
    df['diff1']=df['Year']-year
    df['diff2']=df['Price']-price
    df['d1sqr']=df['diff1']**2
    df['d2sqr']=df['diff2']**2
    df['dist']=np.sqrt(df['d2sqr']+df['d1sqr'])
    print(df)
```

## eg:



```
knn(2017,340)
```



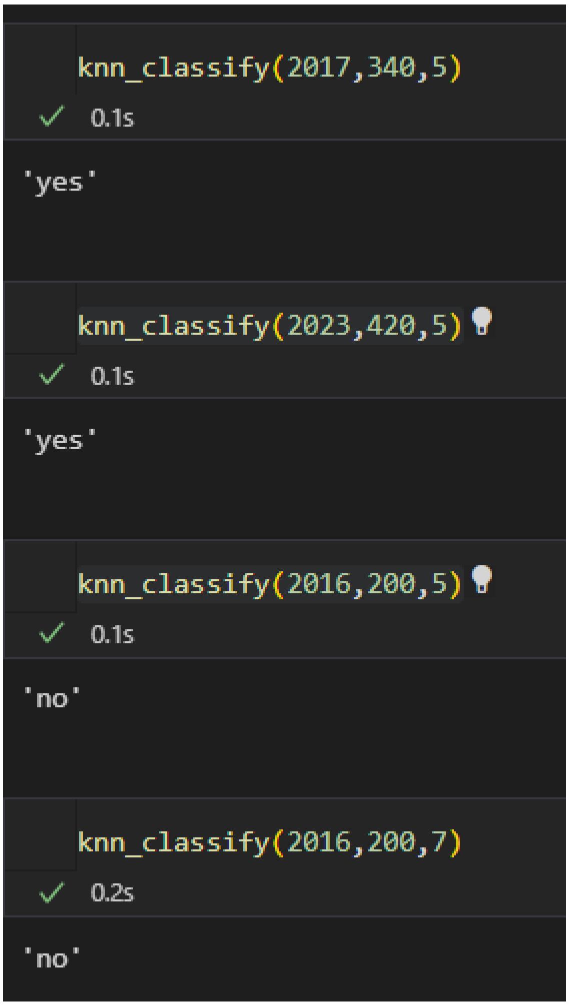
```
    Year  Model  Price  buy  diff1  diff2  d1sqr  d2sqr        dist
0   2010   1.10    120   no     -7   -220     49  48400  220.111335
1   2011   1.20    150   no     -6   -190     36  36100  190.094713
2   2011   1.80    160   no     -6   -180     36  32400  180.099972
3   2012   2.00    145   no     -5   -195     25  38025  195.064092
4   2012   2.10    150   no     -5   -190     25  36100  190.065778
5   2012   2.40    160   no     -5   -180     25  32400  180.069431
6   2013   2.60    190   no     -4   -150     16  22500  150.053324
7   2013   2.80    210   no     -4   -130     16  16900  130.061524
8   2014   2.90    220   no     -3   -120      9  14400  120.037494
9   2014   3.00    200   no     -3   -140      9  19600  140.032139
10  2015   3.10    210   no     -2   -130      4  16900  130.015384
11  2015   3.20    215  yes     -2   -125      4  15625  125.015999
12  2016   3.30    220  yes     -1   -120      1  14400  120.004167
13  2016   3.40    260   no     -1    -80      1   6400   80.006250
14  2017   3.50    270   no      0    -70      0   4900   70.000000
15  2017   3.70    280   no      0    -60      0   3600   60.000000
16  2018   4.00    250  yes      1    -90      1   8100   90.005555
17  2018   4.20    264  yes      1    -76      1   5776   76.006579
18  2018   4.40    270  yes      1    -70      1   4900   70.007142
19  2019   4.60    282   no      2    -58      4   3364   58.034473
20  2019   4.70    295   no      2    -45      4   2025   45.044423
21  2020   4.80    303   no      3    -37      9   1369   37.121422
22  2020   4.82    312   no      3    -28      9    784   28.160256
23  2020   4.90    323  yes      3    -17      9    289   17.262677
24  2021   4.95    337  yes      4     -3     16      9    5.000000
25  2021   5.00    300  yes      4    -40     16   1600   40.199502
26  2022   5.10    315  yes      5    -25     25    625   25.495098
27  2022   5.20    340  yes      5      0     25      0    5.000000
28  2022   5.40    360   no      5     20     25    400   20.615528
29  2023   5.80    399  yes      6     59     36   3481   59.304300
```

## Make function that take input values and k value and gives the classification of that point according to k nearest neighbours

```
def knn_classify(year,price,k):
    df['diff1']=df['Year']-year
    df['diff2']=df['Price']-price
    df['d1sqr']=df['diff1']**2
    df['d2sqr']=df['diff2']**2
    df['distance']=np.sqrt(df['d2sqr']+df['d1sqr'])
    top_k = df.nsmallest(k, 'distance')
    prediction = top_k['buy'].value_counts().head(1).index[0]
    return prediction
```

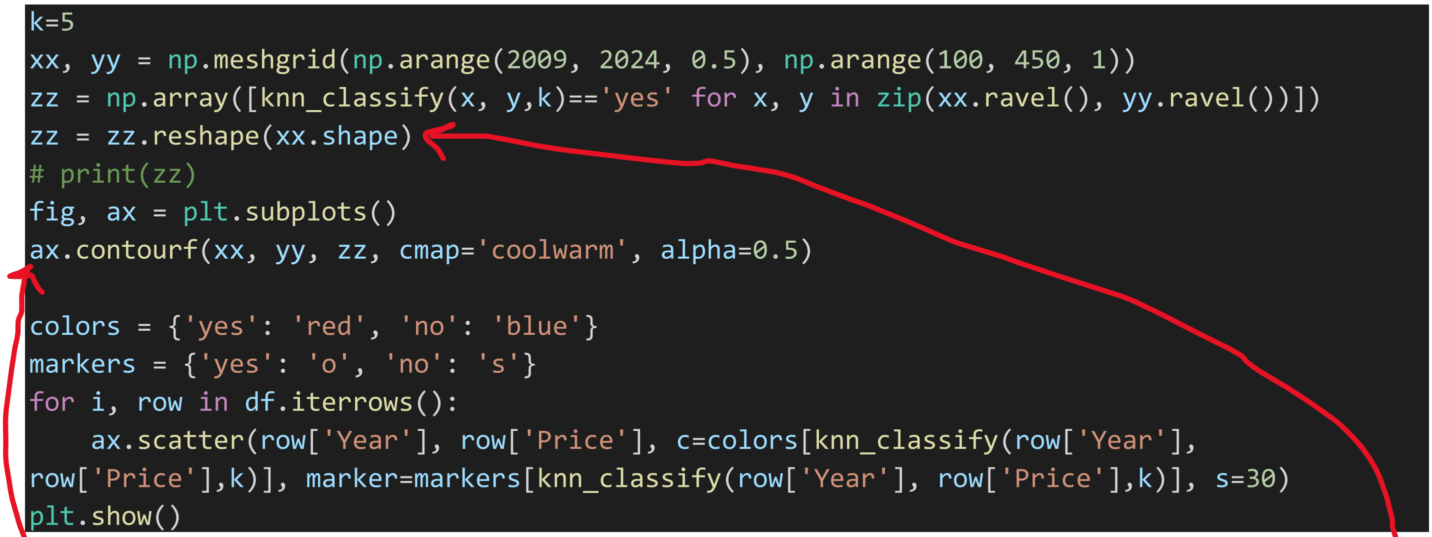
## returns yes/no

```
knn_classify(2017,340,5)
```



```
knn_classify(2017,340,5)
✓ 0.1s

'yes'

knn_classify(2023,420,5)
✓ 0.1s

'yes'

knn_classify(2016,200,5)
✓ 0.1s

'no'

knn_classify(2016,200,7)
✓ 0.2s

'no'
```

## Plotting the dataset and plotting regions according to what classification they would give

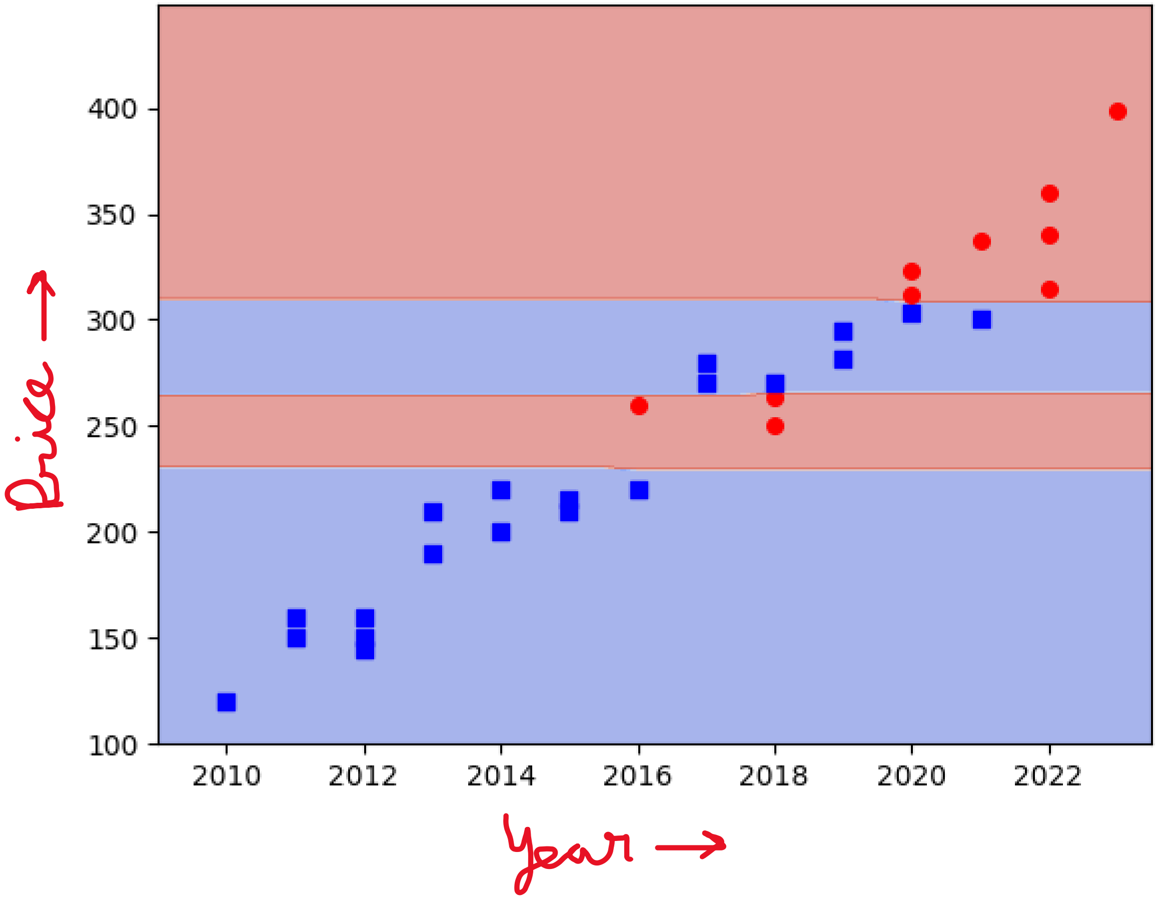


```
k=5
xx, yy = np.meshgrid(np.arange(2009, 2024, 0.5), np.arange(100, 450, 1))
zz = np.array([knn_classify(x, y,k)=='yes' for x, y in zip(xx.ravel(), yy.ravel())])
zz = zz.reshape(xx.shape)
# print(zz)
fig, ax = plt.subplots()
ax.contourf(xx, yy, zz, cmap='coolwarm', alpha=0.5)

colors = {'yes': 'red', 'no': 'blue'}
markers = {'yes': 'o', 'no': 's'}
for i, row in df.iterrows():
    ax.scatter(row['Year'], row['Price'], c=colors[knn_classify(row['Year'],
row['Price'],k)], marker=markers[knn_classify(row['Year'], row['Price'],k)], s=30)
plt.show()
```

plot the contour

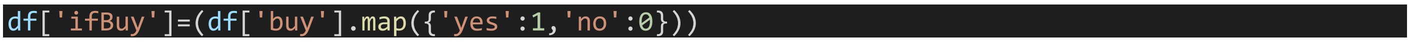
Make contours according to classification and color them accordingly

**Red="Yes" Blue="No"**

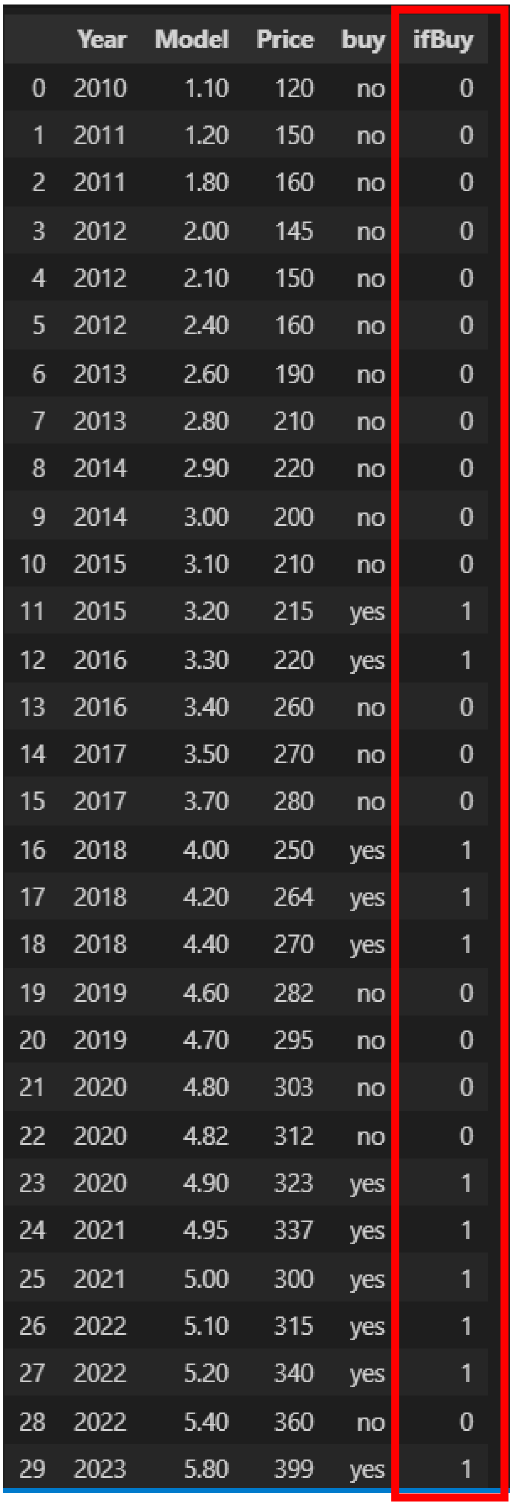


## Q2) Pass the dataset through a single layer perceptron to find the weights of each feature and the bias

## Converting class labels “yes” and “no” to ordinal 0 and 1

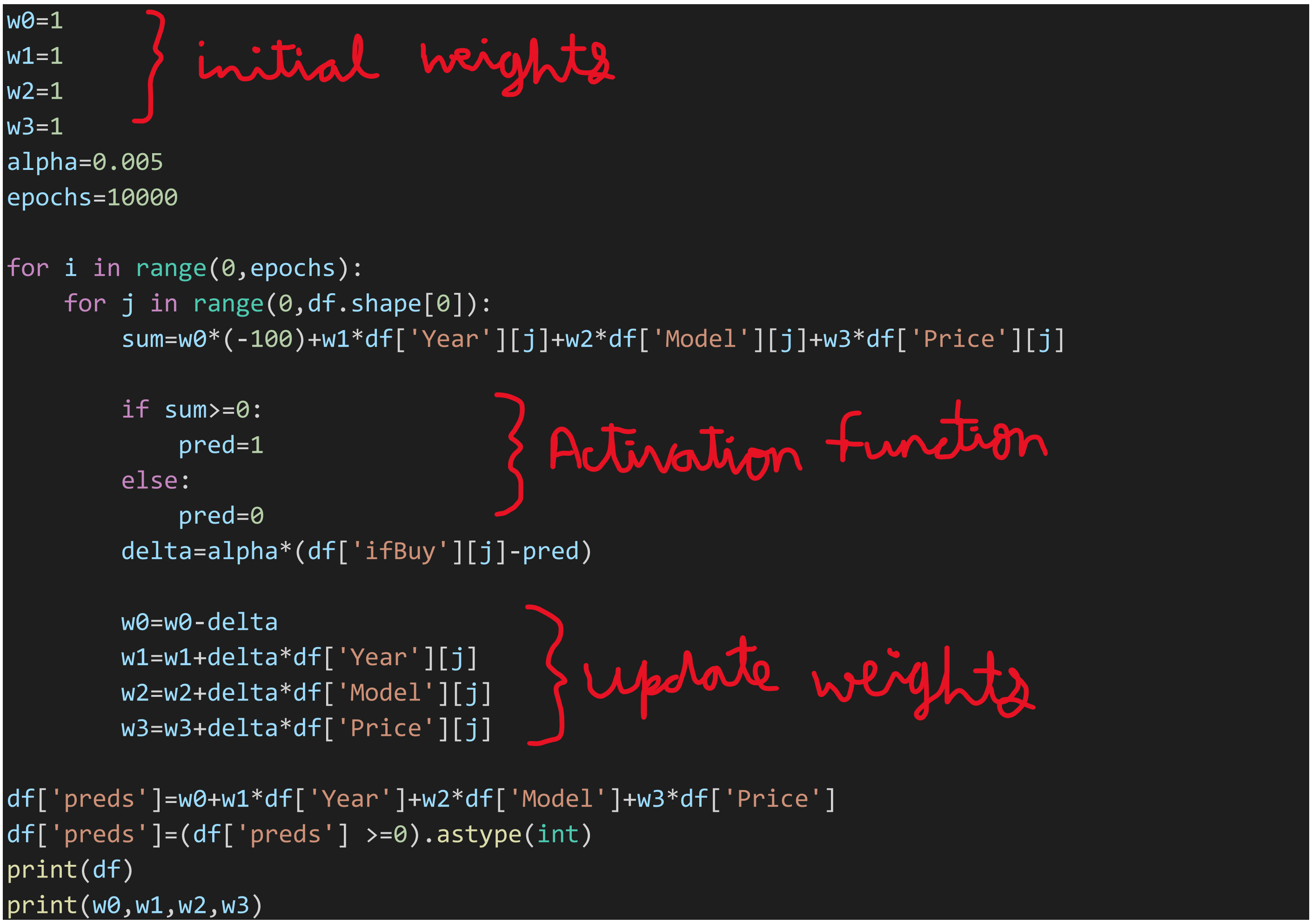


```
df['ifBuy']=(df['buy'].map({'yes':1,'no':0}))
```

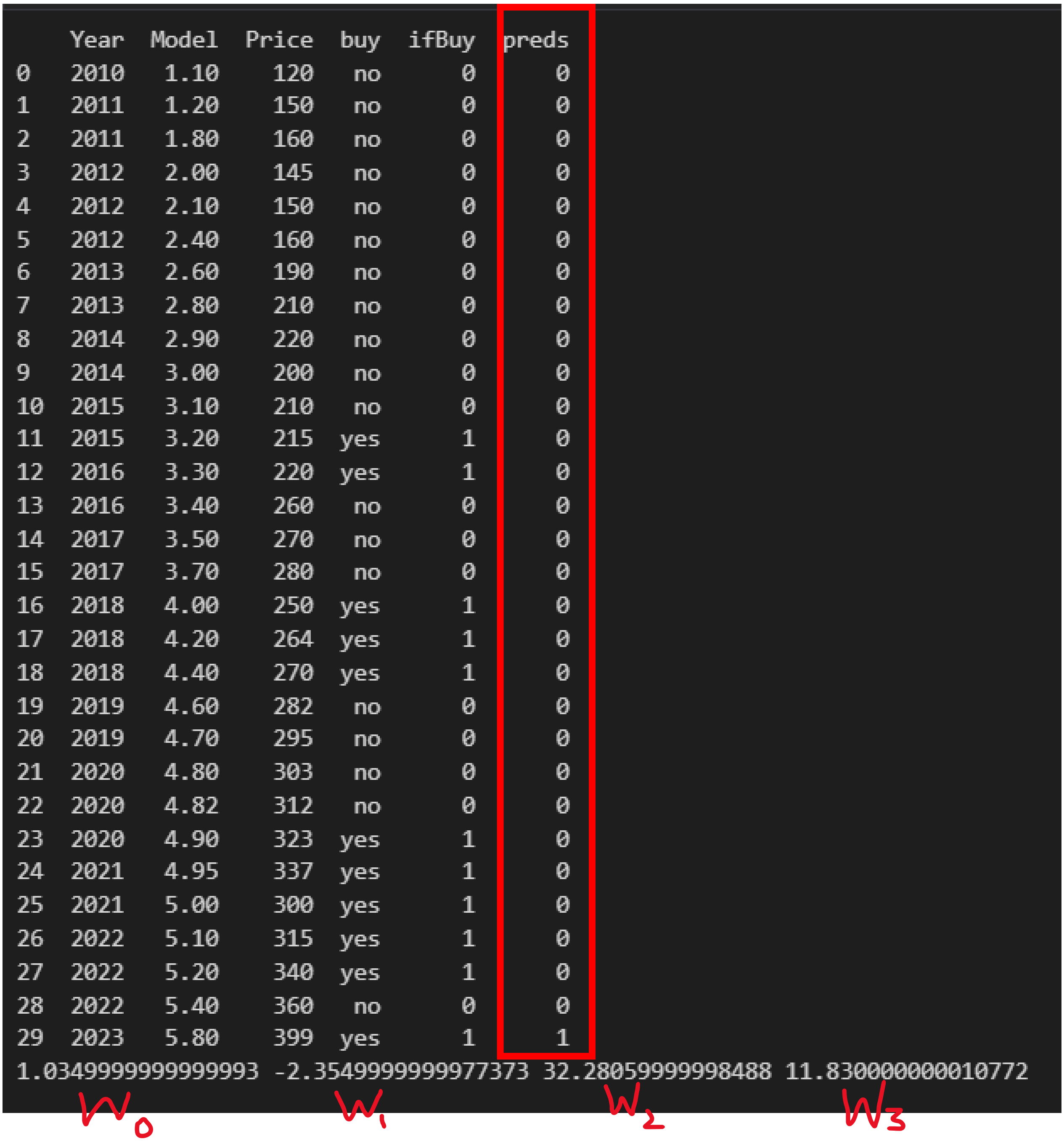


| | Year | Model | Price | buy | ifBuy |
|---|---|---|---|---|---|
| 0 | 2010 | 1.10 | 120 | no | 0 |
| 1 | 2011 | 1.20 | 150 | no | 0 |
| 2 | 2011 | 1.80 | 160 | no | 0 |
| 3 | 2012 | 2.00 | 145 | no | 0 |
| 4 | 2012 | 2.10 | 150 | no | 0 |
| 5 | 2012 | 2.40 | 160 | no | 0 |
| 6 | 2013 | 2.60 | 190 | no | 0 |
| 7 | 2013 | 2.80 | 210 | no | 0 |
| 8 | 2014 | 2.90 | 220 | no | 0 |
| 9 | 2014 | 3.00 | 200 | no | 0 |
| 10 | 2015 | 3.10 | 210 | no | 0 |
| 11 | 2015 | 3.20 | 215 | yes | 1 |
| 12 | 2016 | 3.30 | 220 | yes | 1 |
| 13 | 2016 | 3.40 | 260 | no | 0 |
| 14 | 2017 | 3.50 | 270 | no | 0 |
| 15 | 2017 | 3.70 | 280 | no | 0 |
| 16 | 2018 | 4.00 | 250 | yes | 1 |
| 17 | 2018 | 4.20 | 264 | yes | 1 |
| 18 | 2018 | 4.40 | 270 | yes | 1 |
| 19 | 2019 | 4.60 | 282 | no | 0 |
| 20 | 2019 | 4.70 | 295 | no | 0 |
| 21 | 2020 | 4.80 | 303 | no | 0 |
| 22 | 2020 | 4.82 | 312 | no | 0 |
| 23 | 2020 | 4.90 | 323 | yes | 1 |
| 24 | 2021 | 4.95 | 337 | yes | 1 |
| 25 | 2021 | 5.00 | 300 | yes | 1 |
| 26 | 2022 | 5.10 | 315 | yes | 1 |
| 27 | 2022 | 5.20 | 340 | yes | 1 |
| 28 | 2022 | 5.40 | 360 | no | 0 |
| 29 | 2023 | 5.80 | 399 | yes | 1 |

## Running a single layer perceptron with all weights starting at 1, alpha as 0.005 an for 10000 epochs



```
w0=1
w1=1
w2=1
w3=1
alpha=0.005
epochs=10000

for i in range(0,epochs):
    for j in range(0,df.shape[0]):
        sum=w0*(-100)+w1*df['Year'][j]+w2*df['Model'][j]+w3*df['Price'][j]

        if sum>=0:
            pred=1
        else:
            pred=0
        delta=alpha*(df['ifBuy'][j]-pred)

        w0=w0-delta
        w1=w1+delta*df['Year'][j]
        w2=w2+delta*df['Model'][j]
        w3=w3+delta*df['Price'][j]

df['preds']=w0+w1*df['Year']+w2*df['Model']+w3*df['Price']
df['preds']=(df['preds'] >=0).astype(int)
print(df)
print(w0,w1,w2,w3)
```



```
    Year  Model  Price  buy  ifBuy  preds
0   2010   1.10    120   no      0      0
1   2011   1.20    150   no      0      0
2   2011   1.80    160   no      0      0
3   2012   2.00    145   no      0      0
4   2012   2.10    150   no      0      0
5   2012   2.40    160   no      0      0
6   2013   2.60    190   no      0      0
7   2013   2.80    210   no      0      0
8   2014   2.90    220   no      0      0
9   2014   3.00    200   no      0      0
10  2015   3.10    210   no      0      0
11  2015   3.20    215  yes      1      0
12  2016   3.30    220  yes      1      0
13  2016   3.40    260   no      0      0
14  2017   3.50    270   no      0      0
15  2017   3.70    280   no      0      0
16  2018   4.00    250  yes      1      0
17  2018   4.20    264  yes      1      0
18  2018   4.40    270  yes      1      0
19  2019   4.60    282   no      0      0
20  2019   4.70    295   no      0      0
21  2020   4.80    303   no      0      0
22  2020   4.82    312   no      0      0
23  2020   4.90    323  yes      1      0
24  2021   4.95    337  yes      1      0
25  2021   5.00    300  yes      1      0
26  2022   5.10    315  yes      1      0
27  2022   5.20    340  yes      1      0
28  2022   5.40    360   no      0      0
29  2023   5.80    399  yes      1      1
1.0349999999999993 -2.3549999999977373 32.28059999998488 11.830000000010772
```

**Clearly, a single layer perceptron provides a very bad fit for the given dataset even after 10000 epochs, as it has 3 features and 30 samples.**

**More layers would be required to get any substantial fit in a neural network.**