Arin Santry
Machine Learning

Predicting the Price of a Computer

Using machine learning algorithms, I will be predicting the price of various consumer computers.

I collected data from several different shopping websites: Best Buy, Amazon, Walmart, and Staples. I looked at new (not refurbished) desktop and laptop computers on the front page of their respective websites.

Taking a cursory look at the data, I found some patterns, all of which were rather predictable. First, laptops are generally more expensive than desktops with the same specifications. Next, of the operating systems, MacOS is consistently more expensive, ChromeOS is consistently cheaper, and Windows 10 and 11 were incredibly variable, with prices between $200 and $10,000. Finally, in a more strange pattern, the prices of computers from Walmart tend to end in ".00," while the other stores' prices tend to end in ".99." Generally, the business and gaming computers tended to have more strange prices, like $2,093.41 or $9,224.49

Since I'm looking for price, I first needed to make sure the prices I was looking at were consistent. To do this, I ignored sale or promotional prices, since they tend to change, and only tracked sticker prices. Next, I decided which parameters to use. I wanted to use parameters that would be applicable to both laptops and desktops, so while variables like screen size and resolution factor into the cost of laptops, they are not applicable to desktops, so I did not add them as parameters. Ultimately, I decided on these parameters: desktop/laptop, manufacturer, model, CPU, CPU speed, # CPU cores, storage, RAM, operating system, and selling store.

<u>Data breakdown:</u>
Total computers: 239
Type of computer:
# of desktops: 96
# of laptops: 143

Manufacturer:
Acer: 18
Apple: 9
Asus: 30
CyberPower: 2
Dell: 20
Gateway: 9
HP: 89

iBuyPower: 2
Intel: 1
Lenovo: 43
Microsoft: 5
MSI: 7
Samsung: 5

Store:
Amazon: 42
Best Buy: 75
Staples: 69
Walmart: 53

Price:
>$1000: 53
$500-$1000: 125
<$500: 61

Since I am looking for prices, I will need to use a regression model. There are plenty of models to choose from, like linear or polynomial regression, neural networks, decision trees and forests, and Gaussian regression.

Since I have a large dataset with 10 parameters, I want to use a model that can accurately handle so many dimensions, so linear regression is not a good option here. Polynomial regression also wouldn't work well because of the large number of parameters. I ended up choosing the feed-forward neural networks and random forest algorithms as my two best candidates. These will work well with the large number of variables, and are rather customizable, so I will be able to adjust multiple parts of the model to fine-tune results.

I will be using mean square error, root mean square error, and R2 scores to determine accuracy of the models. I am looking for the smallest possible mean square errors and an R2 score of at least 0.4.

Feed-forward neural network:
Initial parameters: test_size=0.20, activation="identity", max_iter=3000, num_hidden_layer=3, hidden_layer_sizes=64

```
MSE: 488.7372520967613
RMSE: 238864.10158709323
R2: 0.3378906925687296
```

Fine tuning (changes from initial parameters listed)
num_hidden_layers=5

```
MSE: 547.6242142221013
RMSE: 299892.2800023739
R2: 0.5552609097699883
```

Errors got larger, but R2 score improved.

num_hidden_layers=5, hidden_layer_sizes=128

```
MSE: 1125.008256743207
RMSE: 1265643.5777403896
R2: 0.2943329310625804
```

All metrics got worse.

num_hidden_layers=5, hidden_layer_sizes=32

```
MSE: 471.2990696311762
RMSE: 222122.8130352123
R2: 0.43954746462636873
```

Similar performance to 64 neurons, with better error, but worse R2 score.

num_hidden_layers=5, hidden_layer_sizes=32, activation=”relu”

```
MSE: 438.53331047905755
RMSE: 192311.46439972147
R2: 0.5690787670413344
```

Better than identity, but was given the warning “maximum iterations reached and the optimization hasn’t converged left,” so I will increase the number of iterations for the next run.

num_hidden_layers=5, hidden_layer_sizes=64, activation=”relu”

```
MSE: 521.7418401869704
RMSE: 272214.54780168616
R2: 0.6229648539821449
```

Again, similar to 32 neurons. It took much longer than the last run, but it did not throw the warning.

num_hidden_layers=5, hidden_layer_sizes=32. activation=”relu”, max_iter=10000

```
MSE: 435.667086773042
RMSE: 189805.8104973093
R2: 0.6010823151378597
```

No warning, and this run took a while (around 15 seconds). More accurate than both 64 and 32 neurons with 3000 iterations. This is the best configuration so far.

num_hidden_layers=5, hidden_layer_sizes=32. activation=”logistic”, max_iter=10000

```
MSE: 751.1285061582051
RMSE: 564194.0327634568
R2: -0.7729453668919377
```

This one is the worst by far. This also gave the convergence warning. Strangely, it predicted the same price for every computer:

```
Example predictions:
      Actual   Predicted
213  1649.99  317.408992
129   749.99  317.408992
0     199.00  317.408992
72    559.99  317.408992
196  1199.99  317.408992
232  2799.99  317.408992
190  1079.99  317.408992
163   859.99  317.408992
12    249.99  317.408992
37    399.99  317.408992
8     229.99  317.408992
85    599.99  317.408992
41    449.00  317.408992
```

num_hidden_layers=5, hidden_layer_sizes=32. activation=”tanh”, max_iter=10000

```
MSE: 1679.3123182839774
RMSE: 2820089.8623403064
R2: -0.33412099987781185
```

This configuration had the same problems as the logistic activation, including the convergence warning and the identical predictions.

```
Example predictions:
      Actual   Predicted
27    309.00  325.907235
167   879.00  325.907235
117   699.99  325.907236
26    299.99  325.907234
231  2799.00  325.907236
25    299.99  325.907235
137   789.99  325.907236
218  1827.49  325.907236
236  3511.20  325.907236
159   849.00  325.907236
215  1710.00  325.907236
2     199.00  325.907235
```

Given these test runs, the best configuration is: num_hidden_layers=5, hidden_layer_sizes=32. activation=”relu”, max_iter=10000

Below are several runs of the program, including the metrics and predictions.

```
MSE: 351.92516833140076
RMSE: 123851.32410508476
R2: 0.7065698720847575

Example predictions:
      Actual    Predicted
195  1199.00  1851.185731
138   793.99   681.995993
119   699.99   446.152252
14    249.99   238.869032
11    249.99   129.741575
211  1599.99  1304.894950
204  1399.99  1065.529668
41    449.00   711.583033
71    549.99   522.041174
121   729.00   685.197806
152   829.99   828.557604
89    599.99   582.380055
90    599.99   963.166452
30    362.98   163.659113
196  1199.99   939.438523
96    649.99  1232.084430
150   829.99   848.308041
227  2149.00  2398.608275
218  1827.49  1754.851398
101   659.99  1586.246065
46    459.00   764.520824
50    469.99   523.689819
166   870.99  1010.764336
233  2902.49  2197.260550
98    649.99   290.178420
4     225.00   132.381251
```

```
MSE: 224.85229862306505
RMSE: 50558.55619607602
R2: 0.8777512300272671

Example predictions:
      Actual    Predicted
204  1399.99  1127.367968
67    539.99   471.993713
185   999.99  1093.959553
63    519.99   596.367898
110   692.49   715.603924
82    589.99   578.170207
3     219.00   180.234477
83    589.99   436.725406
59    499.99   657.208090
214  1649.99  1828.114247
154   829.99   589.686571
84    599.50   696.396387
116   699.99  1269.001231
232  2799.99  3278.347530
193  1190.99  1326.472310
22    299.00   133.562986
28    309.99   298.799083
58    499.99   850.388268
74    569.99   511.883671
95    639.00   599.284547
211  1599.99  2037.382898
206  1409.99  1320.745359
48    469.00   588.457229
49    469.00   627.958299
57    499.99   482.275237
85    599.99   609.878117
```

```
MSE: 396.97631370097076
RMSE: 157590.19363961153
R2: 0.45878498622716624

Example predictions:
      Actual    Predicted
42    449.99   573.010888
75    569.99   672.469548
139   799.99   690.833635
83    589.99   780.192181
11    249.99   394.148563
233  2902.49  3807.453465
102   659.99  1079.715708
7     229.99   263.864733
89    599.99   850.337461
33    379.99   669.229666
101   659.99  1073.767598
136   789.00   625.821115
17    289.99   320.531345
126   749.99   808.274239
29    339.99   323.730146
100   649.99   349.391802
87    599.99   658.595137
50    469.99   575.979769
59    499.99   882.871151
184   999.99   668.865418
16    289.99   335.698988
203  1399.00  1403.563733
124   739.99   729.493240
19    292.68   247.491782
208  1449.99  1460.285057
```

Citations/resources:

https://towardsdatascience.com/7-of-the-most-commonly-used-regression-algorithms-and-how-to-choose-the-right-one-fc3c8890f9e3
https://towardsdatascience.com/deep-neural-multilayer-perceptron-mlp-with-scikit-learn-2698e77155e
https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.html
amazon.com
bestbuy.com
staples.com
walmart.com