Computer Monitor Pricing

Xiaorui Yuan

Design

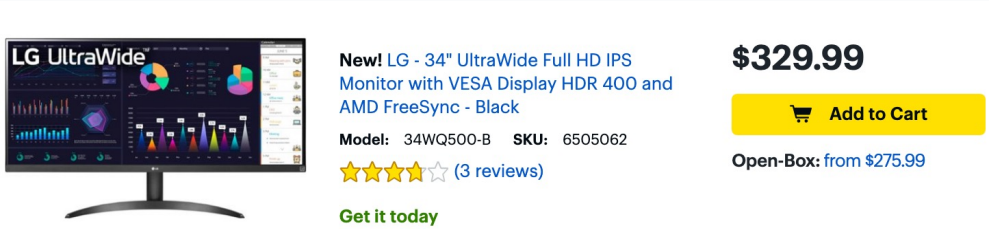
Client: Computer monitor customers

Objective: Explore monitor pricing: how prices are decided?

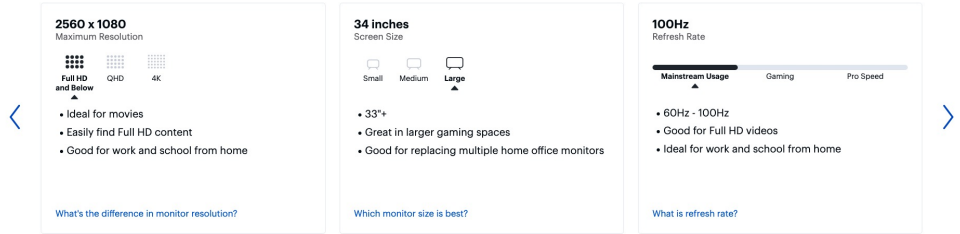
Goal: Be able to predict monitor prices given certain set of monitor specifications

Data

- Monitor data scraped from BEST BUY
- Target:Monitor pricing



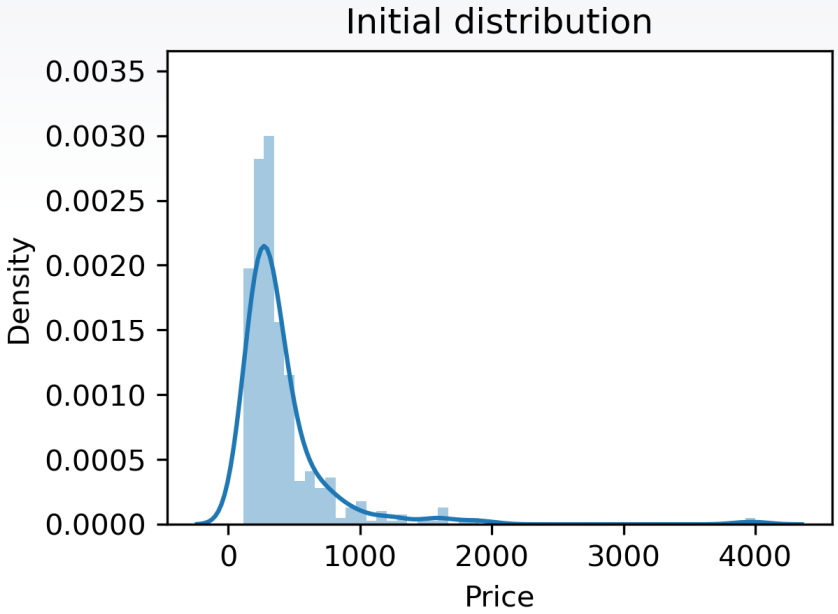
- Features: monitor specifications

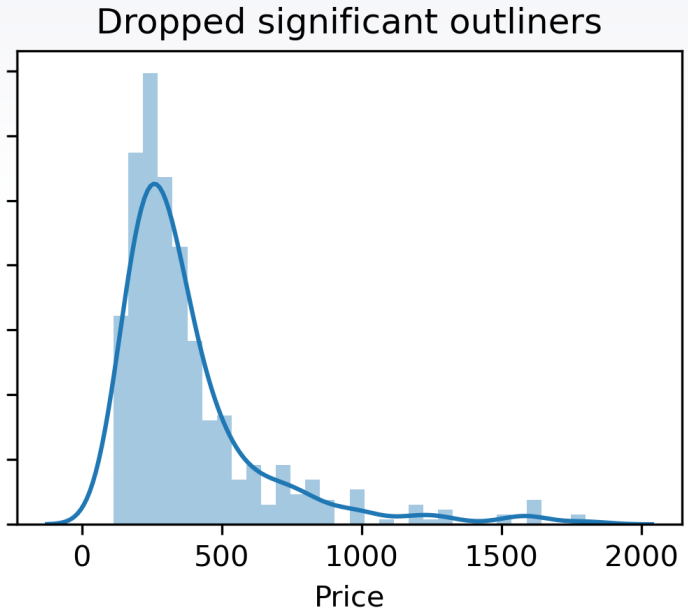


Final Cleaned Data

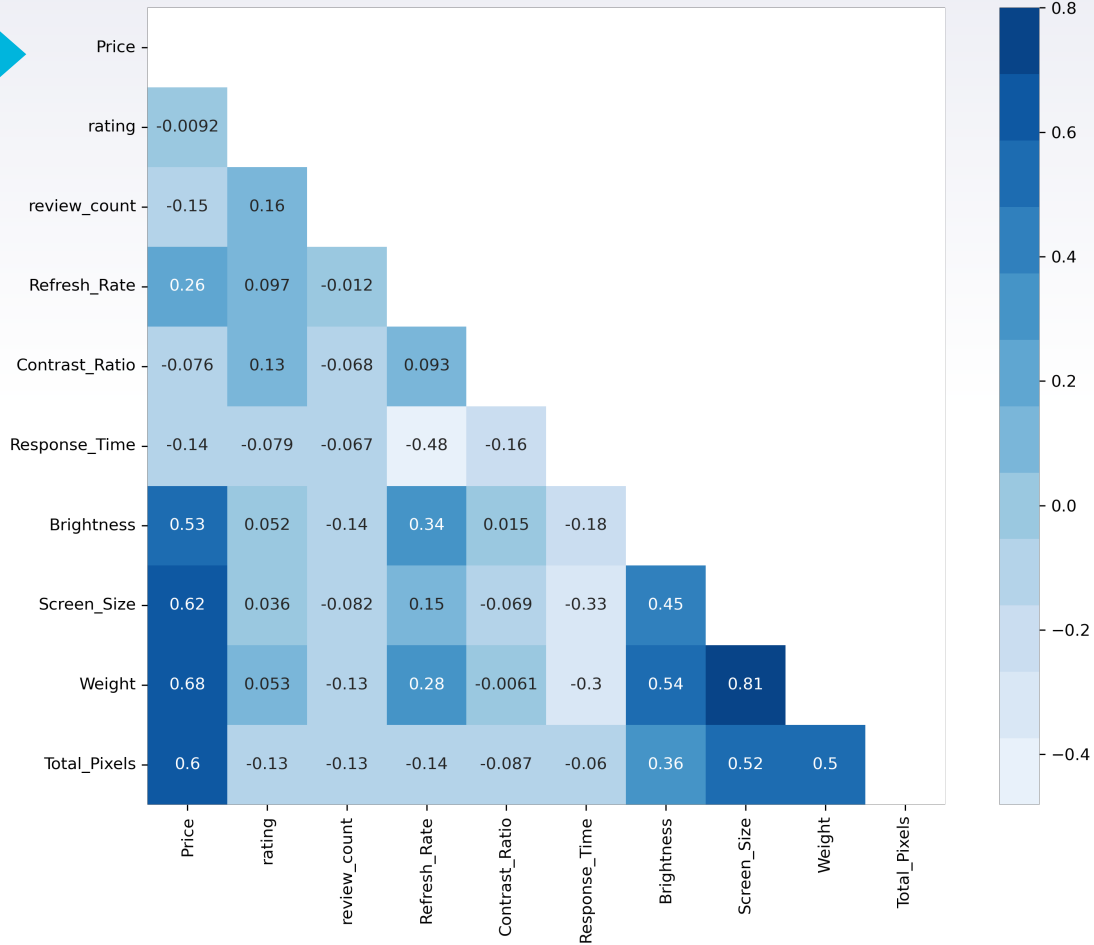
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 502 entries, 0 to 501
Data columns (total 25 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   brand                        502 non-null    object
 1   Price                        502 non-null    float64
 2   rating                       502 non-null    float64
 3   review_count                 502 non-null    float64
 4   Refresh_Rate                 502 non-null    float64
 5   Contrast_Ratio               502 non-null    float64
 6   Response_Time                502 non-null    float64
 7   Synchronization_Technology   502 non-null    object
 8   Panel_Type                   502 non-null    object
 9   Curved_Screen                502 non-null    object
 10  Brightness                   502 non-null    float64
 11  Screen_Size                  502 non-null    float64
 12  HDR                          502 non-null    object
 13  Smart_Display                502 non-null    object
 14  Touch_Screen                 502 non-null    object
 15  Quantum_Dot_Technology       502 non-null    object
 16  Headphone_Jack               502 non-null    object
 17  Voice_Assistant              502 non-null    object
 18  Wall_Mountable               502 non-null    object
 19  Anti_Glare                   502 non-null    object
 20  Speaker                      502 non-null    object
 21  Webcam                       502 non-null    object
 22  Weight                       502 non-null    float64
 23  Resolution                   502 non-null    object
 24  Total_Pixels                 502 non-null    int64
dtypes: float64(9), int64(1), object(15)
memory usage: 98.2+ KB
```

Data Deep dive (1): Distribution of Target





Data Deep dive (2): Obvious Relationships



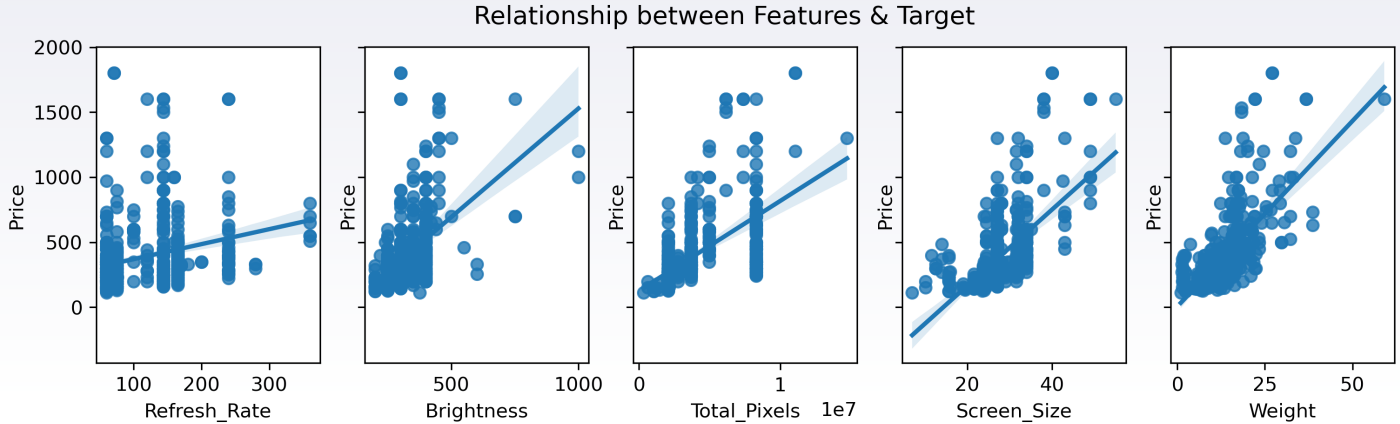
Notable relationship with target:

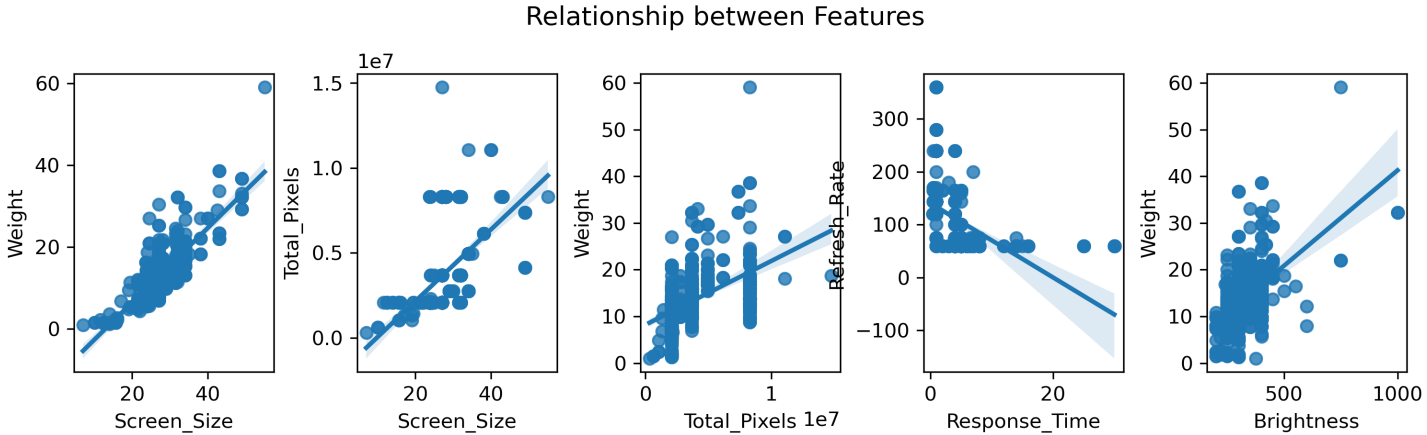
- Refresh Rate: 0.26
- Brightness:0.53
- Total Pixels: 0.6
- Screen Size:0.62
- Weight: 0.68

Warning of multicollinearity:

- Screen Size & Weight: 0.81
- Screen Size & Total Pixels: 0.52
- Total Pixels & Weight: 0.5
- Response Time & Refresh Rate: -0.48
- Brightness & Weight: 0.45
- Screen Size & Brightness: 0.45
- Total Pixels & Brightness: 0.36

Data Deep dive (3)





Baseline OLS

```
train_val_test(x,y)
```

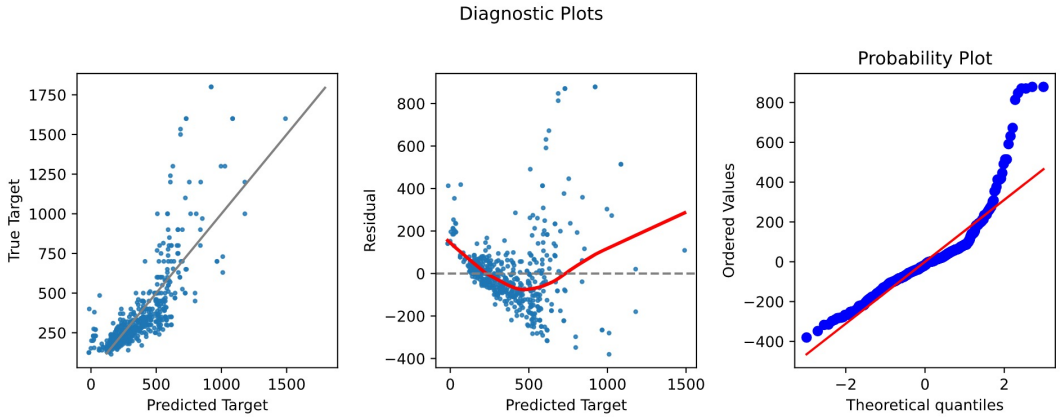
```
train r score: 0.594
validation r score: 0.629
test r score: 0.658
```

```
KFold3_cross_validation(x,y)
```

```
Simple regression scores:  [0.54588404 0.59138039 0.5834882 ]
Simple mean cv r^2:  0.574 +- 0.02
```

```
KFold5_cross_validation(x,y)
```

```
Simple regression scores:  [0.52381485 0.607549   0.59280324 0.01615653 0.63758722]
Simple mean cv r^2:  0.476 +- 0.233
```



Dep. Variable:	Price	**R-squared:**	0.616
Model:	OLS	**Adj. R-squared:**	0.609
Method:	Least Squares	**F-statistic:**	86.49
Date:	Wed, 15 Jun 2022	**Prob (F-statistic):**	4.55e-95
Time:	17:16:36	**Log-Likelihood:**	-3249.7
No. Observations:	496	**AIC:**	6519.
Df Residuals:	486	**BIC:**	6562.
Df Model:	9		
Covariance Type:	nonrobust		

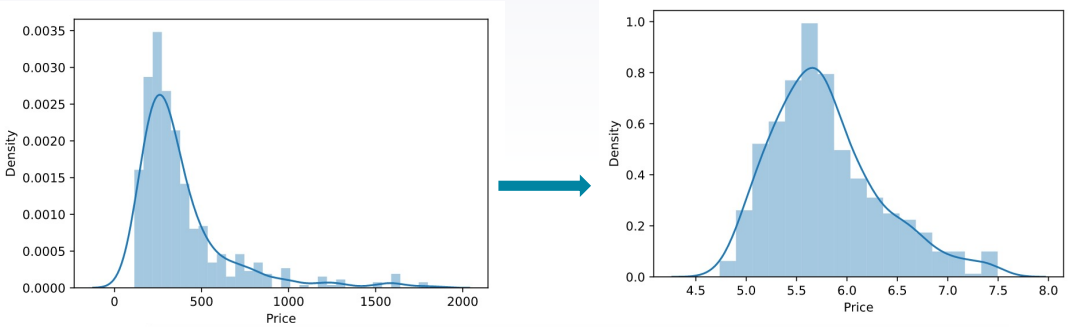
	coef	std err	t	P>|t|	[0.025	0.975]
const	-414.2909	76.457	-5.419	0.000	-564.518	-264.064
rating	3.4965	11.775	0.297	0.767	-19.640	26.633
review_count	-0.0163	0.016	-1.041	0.299	-0.047	0.014
Refresh_Rate	1.0760	0.166	6.479	0.000	0.750	1.402
Contrast_Ratio	-2.833e-07	2.38e-07	-1.189	0.235	-7.51e-07	1.85e-07
Response_Time	9.3568	2.292	4.083	0.000	4.854	13.860
Brightness	0.3492	0.113	3.091	0.002	0.127	0.571
Screen_Size	7.5140	2.419	3.106	0.002	2.761	12.267
Weight	11.6788	2.214	5.276	0.000	7.329	16.028
Total_Pixels	4.313e-05	4.22e-06	10.215	0.000	3.48e-05	5.14e-05

Omnibus:	247.326	**Durbin-Watson:**	1.903
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1674.416
Skew:	2.088	**Prob(JB):**	0.00
Kurtosis:	10.974	**Cond. No.**	3.70e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.7e+08. This might indicate that there are strong multicollinearity or other numerical problems.

Feature Engineering(1)

- Drop insignificant features
- Log transformation on target
- Add polynomial terms & interaction terms




```
train_val_test(x,y)
```

```
train r score: 0.738
validation r score: 0.74
test r score: 0.664
```

R score improved but overfit

```
KFold3_cross_validation(x,y)
```

```
Regression scores:  [0.69955686 0.71784872 0.63479348]
Mean cv r^2:  0.684 +- 0.036
```

```
KFold5_cross_validation(x,y)
```

```
Regression scores:  [0.679269   0.72165913 0.73456801 0.56923704 0.66944311]
Mean cv r^2:  0.675 +- 0.058
```

Dep. Variable:	Log_Price	**R-squared:**	0.728
Model:	OLS	**Adj. R-squared:**	0.724
Method:	Least Squares	**F-statistic:**	163.3
Date:	Wed, 15 Jun 2022	**Prob (F-statistic):**	1.34e-132
Time:	00:32:22	**Log-Likelihood:**	-75.277
No. Observations:	496	**AIC:**	168.6
Df Residuals:	487	**BIC:**	206.4
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.7329	0.183	25.831	0.000	4.373	5.093
Refresh_Rate	0.0022	0.000	8.461	0.000	0.002	0.003
Brightness	0.0035	0.001	5.795	0.000	0.002	0.005
Screen_Size	-0.0626	0.013	-4.887	0.000	-0.088	-0.037
Weight	0.0178	0.004	4.641	0.000	0.010	0.025
Total_Pixels	1.611e-07	4.3e-08	3.747	0.000	7.66e-08	2.46e-07
Screen_Size^2	0.0015	0.000	5.182	0.000	0.001	0.002
Brightness^2	-3.15e-06	6.2e-07	-5.078	0.000	-4.37e-06	-1.93e-06
Screen_Size*Pixel	-2.141e-09	1.46e-09	-1.471	0.142	-5e-09	7.18e-10

Omnibus:	16.772	**Durbin-Watson:**	1.958
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	17.659
Skew:	0.426	**Prob(JB):**	0.000146
Kurtosis:	3.359	**Cond. No.**	2.00e+09

Feature Engineering(2)

Add Dummies

```
train_val_test(x,y)
```

```
train r score: 0.851
validation r score: 0.789
test r score: 0.723
```

```
KFold3_cross_validation(x,y)
```

```
Regression scores:  [0.74531605 0.72603497 0.73027329]
Mean cv r^2:  0.734 +- 0.008
```

```
KFold5_cross_validation(x,y)
```

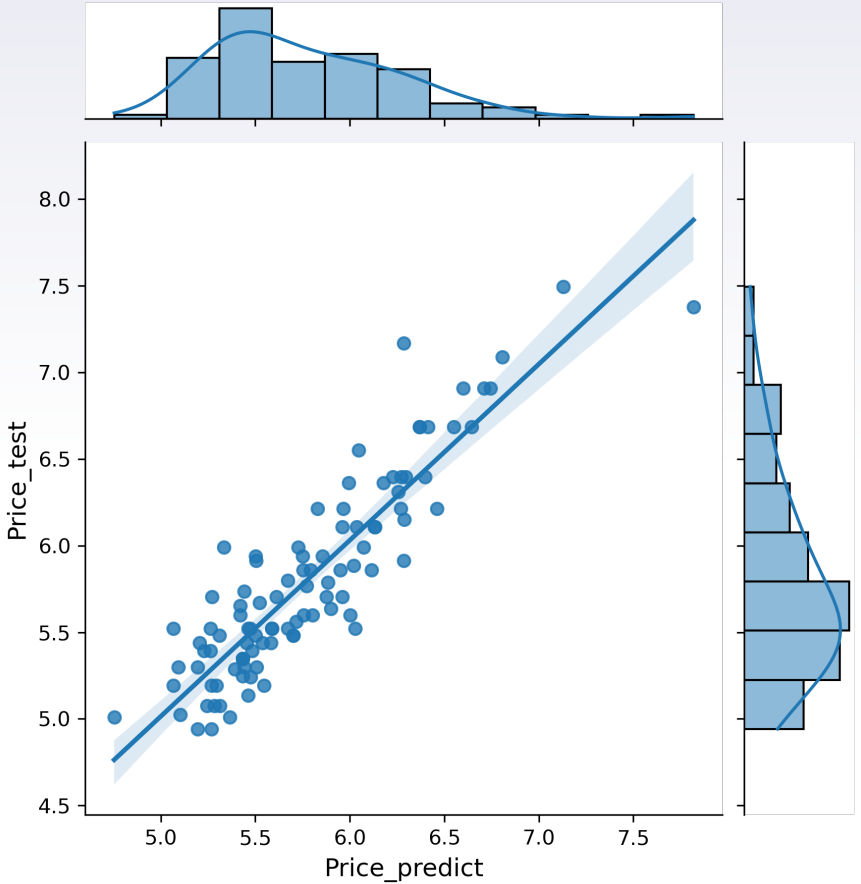
```
Regression scores:  [0.78341605 0.66847107 0.82317504 0.66866058 0.75132861]
Mean cv r^2:  0.739 +- 0.062
```

R score improved but overfit !

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.6081	0.216	21.321	0.000	4.183	5.033
Refresh_Rate	0.0026	0.000	8.756	0.000	0.002	0.003
Brightness	0.0021	0.001	3.403	0.001	0.001	0.003
Screen_Size	-0.0388	0.014	-2.727	0.007	-0.067	-0.011
Weight	0.0136	0.004	3.754	0.000	0.006	0.021
Total_Pixels	1.485e-07	3.81e-08	3.900	0.000	7.37e-08	2.23e-07
Screen_Size^2	0.0011	0.000	3.985	0.000	0.001	0.002
Brightness^2	-1.89e-06	6.02e-07	-3.141	0.002	-3.07e-06	-7.08e-07
Screen_Size*Pixel	-1.814e-09	1.29e-09	-1.409	0.160	-4.34e-09	7.16e-10
brand_AOC	-0.0142	0.076	-0.186	0.852	-0.164	0.136
brand_AOPEN	-0.5466	0.241	-2.267	0.024	-1.020	-0.073
brand_ASUS	-0.0337	0.056	-0.599	0.549	-0.144	0.077
brand_Acer	-0.0013	0.062	-0.021	0.983	-0.123	0.121
brand_Alienware	0.1172	0.116	1.007	0.314	-0.112	0.346
brand_Asus	-0.0768	0.148	-0.520	0.604	-0.368	0.214
brand_BenQ	0.0338	0.069	0.493	0.622	-0.101	0.168
brand_CORSAIR	0.2168	0.181	1.196	0.232	-0.139	0.573
brand_Dell	0.1042	0.065	1.613	0.107	-0.023	0.231
brand_DoubleSight	-0.1649	0.213	-0.773	0.440	-0.585	0.255
brand_Element	-0.1474	0.142	-1.039	0.299	-0.426	0.131
brand_GIGABYTE	-0.0423	0.074	-0.572	0.568	-0.188	0.103
brand_Geek	0.1410	0.240	0.586	0.558	-0.331	0.613
brand_HP	0.1949	0.070	2.788	0.006	0.057	0.332
brand_LG	0.1402	0.051	2.741	0.006	0.040	0.241
brand_Lenovo	0.1035	0.064	1.615	0.107	-0.022	0.229
brand_MSI	0.0414	0.091	0.455	0.650	-0.137	0.220
brand_NHT	0.5028	0.254	1.978	0.049	0.003	1.002
brand_Philips	-0.0052	0.095	-0.055	0.956	-0.192	0.182
brand_Razer	0.3868	0.163	2.371	0.018	0.066	0.707
brand_SideTrak	0.4274	0.140	3.061	0.002	0.153	0.702
brand_VIOTEK	0.0436	0.241	0.181	0.856	-0.429	0.516
brand_ViewSonic	0.0087	0.068	0.127	0.899	-0.126	0.143
brand_Viewsonic	0.1078	0.261	0.412	0.680	-0.406	0.622
brand_Viotek	-0.1676	0.117	-1.435	0.152	-0.397	0.062
Synchronization_Technology_FreeSync	-0.0619	0.031	-2.014	0.045	-0.122	-0.001
Synchronization_Technology_FreeSync&G-SYNC	0.0329	0.054	0.603	0.547	-0.074	0.140
Synchronization_Technology_G-SYNC	0.1071	0.060	1.779	0.076	-0.011	0.225
Synchronization_Technology_G-SYNC&VSync	0.4593	0.179	2.560	0.011	0.107	0.812
Synchronization_Technology_VSync	0.0150	0.093	0.161	0.872	-0.168	0.198
Panel_Type_IPS	0.1582	0.045	3.506	0.001	0.070	0.247
Panel_Type_MVA	0.0556	0.087	0.639	0.523	-0.115	0.226
Panel_Type_PLS	-0.2104	0.242	-0.868	0.386	-0.687	0.266
Panel_Type_TFT	0.2052	0.113	1.819	0.070	-0.017	0.427
Panel_Type_TN	-0.1001	0.065	-1.543	0.124	-0.227	0.027
Panel_Type_VA	-0.1086	0.051	-2.122	0.034	-0.209	-0.008
Curved_Screen_Yes	0.2777	0.040	6.868	0.000	0.198	0.357
HDR_Yes	0.0621	0.032	1.925	0.055	-0.001	0.125
Smart_Display_Yes	-0.1060	0.115	-0.924	0.356	-0.331	0.119
Touch_Screen_Yes	0.3544	0.131	2.699	0.007	0.096	0.612
Quantum_Dot_Technology_Yes	0.1825	0.080	2.277	0.023	0.025	0.340
Headphone_Jack_Yes	-0.1139	0.029	-3.882	0.000	-0.172	-0.056
Voice_Assistant_Yes	0.0918	0.098	0.941	0.347	-0.100	0.283
Wall_Mountable_Yes	-0.0546	0.035	-1.579	0.115	-0.123	0.013
Anti_Glare_Yes	-0.0731	0.036	-2.028	0.043	-0.144	-0.002
Speaker_Yes	0.0949	0.029	3.266	0.001	0.038	0.152
Webcam_Yes	0.1127	0.065	1.730	0.084	-0.015	0.241

Regularization

- Elastic Net Regression Wins !
- R score: train 0.816; test 0.815
- R-square improves & no longer overfit !



Important Features

Refresh Rate: 0.150
Brightness: 0.167
Weight: 0.121
Total Pixels: 0.202
Screen_Size^2: 0.219

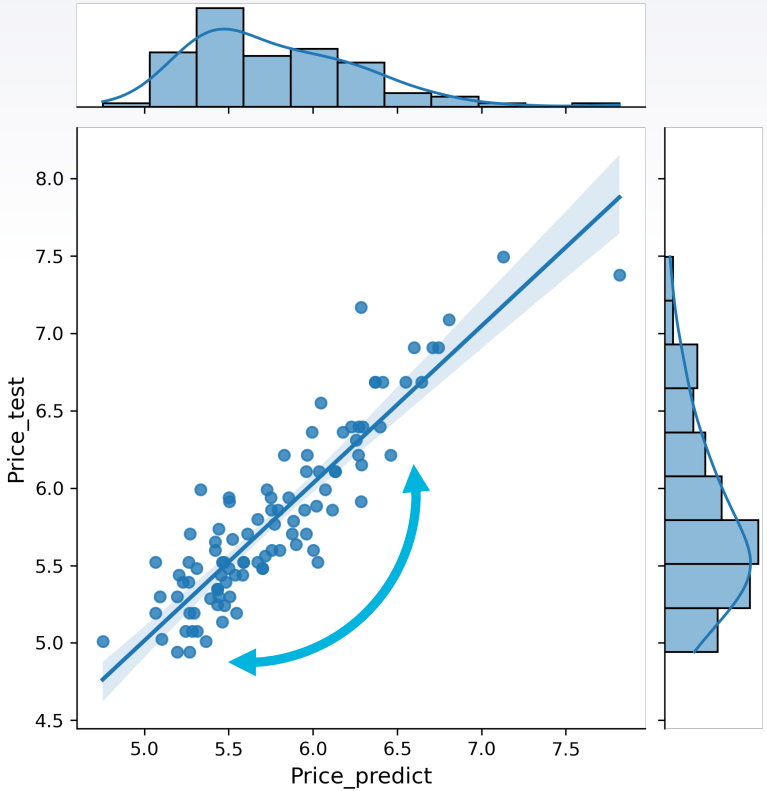
Note in my final Elastic Net Regression, neither target and features are original data.

Target is log price AND features reflect standard deviation from the mean of the feature data.

Ex: Refresh Rate: 0.150
Exp(0.150)- 0.0058-1.16%
Interpretation: 1 std deviation above the mean of Refresh Rate is associated with a 16% increase in price

Recommendation for Clients

- Consider buying the monitor with price underneath the prediction linear line



Future Work

- Deeper dive into multicollinearity problem
- Test data on other website's monitor data such as microcenter.com