Caleb Kathman, Felipe Chiavegatto, Mishka Philizaire, Jamon Oxendine-Blackmon
COMP562
Dr. Silva
5 May 2022

The Prediction and Analysis of Laptop Prices

For our final project, we decided to focus on creating regressions that would assist individuals that do not understand laptop prices in making the best decision possible. We found a dataset on Kaggle by Santosh Kumar that provided details regarding various aspects of laptops being sold in March 2022[1]. Before beginning to work with the data, we used pandas to simplify the data to the features that most influenced the overall results of the latest price of the laptops. By examining the correlation of the numerical features with the feature 'latest_price', we were able to choose a few that most impacted the result. While the 'discount' feature did have a relatively strong correlation to 'latest_price', it was not included because it could be used directly with the 'old_price' to calculate the latest price. It would not be reasonable for us to assume that a user of the model had access to the rate of discount compared to the old price. Instead, we are assuming that they know the old price of the laptop. For categorical features, we used the Kaggle webpage of the data to observe which categories had more variance. Since some of the categories had over 90% with the same category, they were not included. We then proceeded to perform a dummy encoding of the categorical features that were selected and combined that with the numerical features.

Our first model that we looked at was a Linear Regression Model. In the linear regression, we used SciKit-Learn's linear regression to create a model to calculate the latest price of a portion of the dataset. The linear regression was able to get a coefficient of determination of 0.93 which is a pretty strong score. When training on ~700 of the laptops and testing on 195, the average error between actual and predicted laptop price was 9.16%. When training the model, we split the dataset by the first 700 and last 195 for training/testing since there was no evidence that the data scrapped was not randomly sorted. If Linear Regression was found to be the best model of all of them that we test, we could then proceed to form the model using the entire dataset instead of a portion of it. This would further increase the accuracy of the model in creating a prediction regarding the price of laptops. The model could then be used in areas where the data was scrapped to assist in deciding whether the price of a given laptop is too high relative to what is expected for the quality.

[1] https://www.kaggle.com/datasets/kuchhbhi/latest-laptop-price-list

2. Ridge Regression

The second model chosen was the Ridge Regression model, a modification on the Ordinary Least Squares (Linear Regression) model where a penalty is included to regularize coefficients and optimize the model for prediction.[2]

$$\min_{w} ||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 + \alpha||\mathbf{w}||_2^2$$

The ridge penalty aims to prevent the overfitting of data in a model. While an overfitted, linear regression model can prove to be optimized for current train and test data, complications may arise when exposed to new data. Ridge regression's penalty imposed onto prediction coefficients generalizes the model and improves applicability to new, unseen data sets while maintaining model accuracy.

As market rates often fluctuate erratically, especially in laptop pricing, the ridge regression approach aims to minimize over-analyzation of insignificant trends in the data. The penalty applied to the approach decreases the weights of these insignificant trends (see *Figure 1*). To do so, optimal alpha values were calculated to maximize the model's performance. By solving for an alpha value that minimized prediction error through brute-force, a1 was calculated at 0.0473 with an average error of 8.87% (see *Figure 2*). Alpha value a2 was then calculated with sklearn's RidgeCV library, a library that facilitates Ridge Regression with built-in cross-validation.[3] The library calculated an optimal a2 at 0.1 with an average error of 8.93%. Sklearn provides an alpha default value of 1, providing a3 equal to 1 with an average error of 9.64%.

RidgeCV's cross-validation approach to calculating alpha minimizes overfitting and encourages the model corresponding to a2 to be selected as the best predictor for future customers with an average error of 8.93%. Although alpha value a1's model predicted with an average error of only 8.87%, the decrease in error can safely be attributed to the overfitting of data. Thus, a1 provides better results for the current data set while foregoing applicability for future datasets. Compared to section 1's model using linear regression, the ridge regression module provides a 0.23% accuracy increase while being resistant to overfitting and more applicable to unseen data, deeming it the favorable option for consumers.

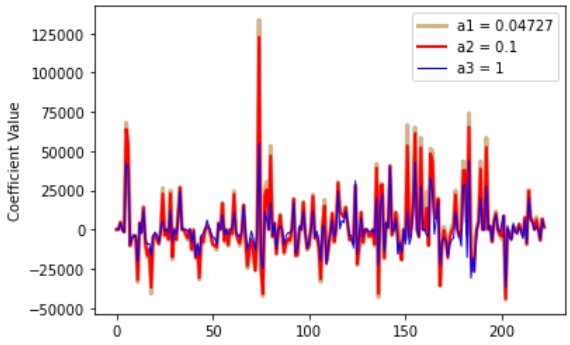

Figure 1: Fitted coefficients as a function of parameters corresponding to varying alpha values

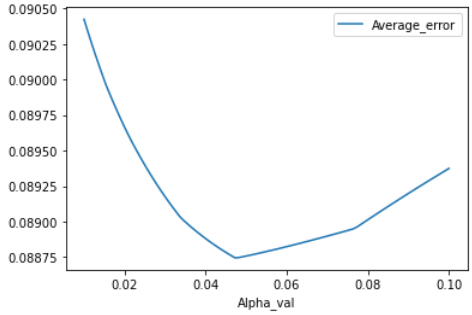

Figure 2: Average prediction as a function of penalty coefficient

[2] https://scikit-learn.org/stable/modules/linear_model.html#ridge-regression-and-classification

[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html#sklearn.linear_model.RidgeCV

3. Lasso Regression

The third model that we chose to use is the Lasso Regression model, which is a linear model that estimates sparse coefficients. Thus, it differs from the previous two models because it prefers solutions with more zero coefficients, meaning the solution is dependent on fewer features. The model is a modification of the linear least squares model with an added regularization term for the coefficients. The objective function for Lasso is:

$$\min_{w} \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha||w||_1$$

In the objective function above, α is a constant that controls the degree of sparsity of the coefficients and ω is the coefficient vector.

We used the sklearn Lasso class to fit the coefficients for our regression on laptop price. The sklearn Lasso implementation uses coordinate descent to find the best solution. Using sklearn Lasso, we noticed that for alpha values of approximately 11.1 and below, the coordinate descent would fail to converge to a solution in 1000 iterations. Thus, we decided to find the optimal alpha value by comparing the average error of Lasso models of alpha values 12 to 50. The optimal alpha value we found was 35.8930 and it had an associated average error of 0.09674, which is greater than the error from the most optimal Ridge Regression model. The plot of the average error on the test data for the Lasso models of different alpha values is shown below in Figure 3.

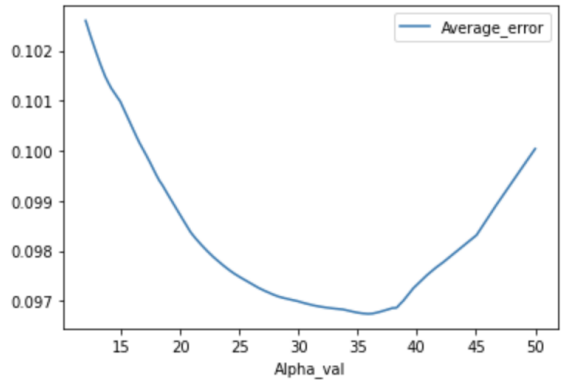

Figure 3: Average error for Lasso models of different alpha values

When taken into consideration that the default alpha value is 1, we noticed that our optimal alpha value that we found is very high. This indicates that many of our feature coefficients are zero. Below (Figure 4) is a graph of the coefficients and their value for our optimal alpha.

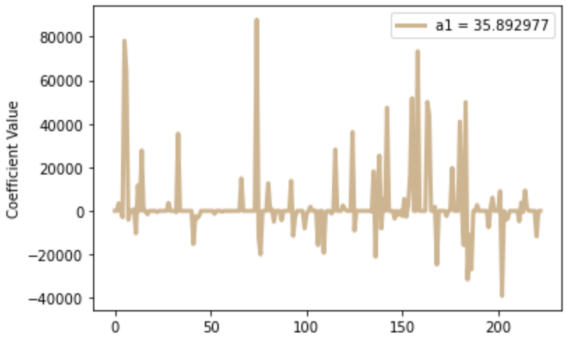

Figure 4: Coefficient values for alpha value of 35.8930

4. The last model was Bayesian Ridge Regression. It makes estimates based on probability distribution. However, this method was a poor choice as Bayesian Ridge Regression is better suited for small data-sets and data received in real-time.