Data Science and Artificial Intelligence II
Final Project Report:
Mobile Devices Pricing Predictions

For:
Professor Hanqin Cai
Class: Data Science and AI II
STA 4365

By:
AJ Romaniello
Baylor Dalsemer
Giovanni Rosa
Sebastian Gonzalez Zurita

Dataset: https://www.kaggle.com/datasets/abdulmalik1518/mobiles-dataset-2025/data

Introduction

Our data consists of detailed hardware specifications and launch prices of different mobile devices from different companies. Our goal with this dataset is to predict and identify trends in mobile device prices using the hardware and software specifics. This information is very valuable to people who wish to have a better understanding of the value they are getting from their phones, and help consumers make more informed purchases to save money.

The dataset comes from Kaggle and includes 930 observations and 15 features. However, the pricing data includes different countries' currency as well, but we will be solely focusing on the prices in USD to avoid collinearity issues. Only 11 features will be used, 8 of which are numerical and 3 are categorical. These features include hardware specifications such as RAM (in GB), processor details, battery capacity (in mAh), front and back camera specifications (in MP), and screen size (in inches). Other features included in the dataset include the weight of the phone (in grams), the model name, company name, and the launch year (2014 to 2025). There are 19 companies represented in this dataset, including Apple, Samsung, OnePlus, Vivo, iQOO, Oppo, Realme, Xiaomi, Lenovo, Motorola, Huawei, Nokia, Sony, Google, Tecno, Infinix, Honor, POCO, and Poco.

Approaches

The initial dataset was very messy and needed tons of preprocessing in order for it to be usable. All columns except company name, processor, and launched year consisted of units along with the data to give context to the numerical value. To make the columns useful we extracted the numerical values and converted them to a common unit in each column. After this processing we were able to extract numerical values for all columns except processor types. We were unable to find enough commonality in different processors to make it numerical with discrete data or find numerical values within the column. Due to this we had to remove the processor from the dataset. We also couldn't keep the model name as there were over 900 different models, but we were able to extract storage size from this column. We created a new column called Storage which holds the storage size for each device in gigabytes. Lastly we converted the company names into categorical discrete data for analysis. There are 19 companies listed in the dataset and we made a dummy variable for each one, assigning a number for each company. For example, Apple = 0, Samsung = 1, OnePlus = 2, etc.

Now that the dataset has been preprocessed, we can do some statistical analysis. We first wanted to see if there were any strong correlations between prices and the device details. The raw data had practically no correlations showing between prices and the predictors. The only predictor decently correlated with price was storage size at 0.56. The rest were between 0 and 0.11. Very weak correlations. After looking at the scatter plots we noticed some very extraneous outliers in the data. We were hesitant to remove outliers so we decided to look for a potential transformation. There was definite skewness observed in the data from histograms and boxplots and we implemented a logarithmic transformation. This transformation reduced the skew and gave us better correlational results. Price now had 2 correlational values above 0.5 with RAM and storage size, compay at 0.35, and the rest between 0.1 and 0.2. These better results were

promising. We then thought of removing the outliers from the data to help this even more. After removing all outliers, we saw that our correlations didn't get any better and our dataset went from 930 observations to 450 observations. We thought removing half of our observations for no correlational improvement wasn't worth it. We decided to keep the logarithmic transformation for the price values and expect normalizing our predictors for our machine learning models will help more. This theory will be tested in our models. After the preprocessing and data cleaning we were ready to move on to our final step before creating the models, feature selection .

The first step in deciding which features to keep was running a principle component analysis (PCA) algorithm. This transformed our high dimensional data into a lower dimension, we decided to bring the dimensions down to two for this step. After running the PCA and looking at the new data, we decided to analyze how much each predictor variable contributed to the two dimensions that were created during PCA. We did this by taking the average of how much each variable contributed, giving us a decimal point (number) for how much each variable affects PCA. What these numbers represent is actually how much each variable contributes to the overall variance of the dataset. Our top contributors were weight (g), screen size (in), RAM (GB), and storage (GB), meaning these contributed most to the variance. Every variable seemed to be at a good level of contribution, except for our Company variable. However, this is to be expected since there is a limited amount of outputs for the company variable (categorical variable), thus limiting how much it can contribute to the variance. With that in mind, we decided to not use this to get rid of any of our predictor variables, mainly because variance contribution is not necessarily reflective of the overall prediction power for each feature.

Our next step in feature selection was to run a Random Forest Regressor for feature importance. This algorithm will give us a better understanding of the prediction power for each

predictor variable, which allows us to make decisions on which to keep. After running the Random Forest Algorithm, we saw that company, storage, weight, and RAM had the highest importance scores. Seeing weight, RAM, and storage as top predictors for both PCA and feature importance is good considering these variables also contribute a considerable amount to the overall variance. This should mean we have very strong predictors for our final model. The bottom two predictors in terms of importance were Launched Year and Back Camera (MP), these had an importance of less than 0.03. With that, we decided that these two variables were going to just take away from a model's accuracy since they are so low on importance, so in the end we decided to remove them from our final model. With those two variables removed, we have our final predictor variables/features being Battery (mAh), Weight (g), Company, Storage (GB), RAM (GB), Screen Size (in), and Front Camera (MP). Now that we have our features selected for our final model, we were ready to use our data for analysis.

Findings

For linear regression, we decided to try two different models to see the relationship pricing had with the remaining variables. The first model was a base linear model, with no transformations or adjustments. Most variables ended up having a positive relationship, with only “Company” having a negative relationship. However, this base model had pretty high MAE and MSE values and a lower R-squared value of 0.32 or 32%. Obviously this model needed much improvement if we wanted to consider using it for any kind of prediction, so for the next model we normalized the X values and used a log transformation on y to see if the results would

improve. This did improve the model by a bit, giving us an R-squared of 0.49 or 49% which is average. The MSE and MAE values were also significantly lower thanks to the log transformation dealing with any outliers and skewness. While the linear regression model was decent, we knew that there were better methods to improve our predictions and obtain a better model.

To contrast with the linear model, a non-linear and non-parametric algorithm was used for the final model. We utilized the K-Nearest Neighbors (KNN) regression model from scikit-learn which utilizes a 'k' number of closest distances from the data points such that their average value is used for prediction. With the understanding that similar input values of phone components and hardware will produce similar output prices for those built phones, the predicted price of the device will be calculated with the distance between this new observation and every single instance in our training set. First we used a minimal model using Euclidean distance, defined as the square root of the squared differences of our six-dimensional feature vector, to find the 'k' training instances with the smallest distances to the new observation. We then determined a good 'k' value based on the R-squared performance of the model initialized with several 'k' values. A small 'k' may lead to predictions that closely follow the training data and predictions might be overly sensitive to individual data points, but a large 'k' smooths out the estimation.

We fit our first minimal model with 5 neighbors resulting in a R-squared of 0.69 but extremely high MSE of 43664.96 and MAE of 145.18. These metrics are pointing towards poor model performance, but not terrible. Our graphs comparing the predicted and tested values

showcases the influence of outliers in the distance calculations, and our residual plot clearly lacks proper linearity and constant variance. As our predictors vary in scale, those with larger magnitudes dominate the distance measurements, which we aim to remedy with standardizing our predictors by removing the mean and scaling to unit variance. Additionally, we utilize the log transform of our price values to stabilize the variance for better performance.

We immediately see the results of our transformations with our second graph comparing different initializations of 'k' values for which the R-squared values are higher. Finally, to ensure the best model is created we use exhaustive grid search with 5-fold cross validation to obtain the best possible combination of hyperparameters for our KNN regression model. These include the already discussed 'k' neighbors, the distance metric being either Euclidean or Manhattan, and the weights being 'uniform' or 'distance'. We obtain: Manhattan or L1 distance to calculate the 'k' neighbors distance, meaning the sum of the absolute differences between feature values. This reduces the sensitivity to outliers as the differences are not squared like with Euclidean and the curse of dimensionality is lowered since it scales more linearly with the spread of each feature such that hardware specs contribute proportionally to the distance calculation. Additionally we utilize the 'distance' weight as opposed to the previous 'uniform' one, meaning that the model assigns greater importance to closer neighbors instead of giving them equal weight. And finally, a best 'k' value of 5 neighbors. Our final model is initialized with the best hyperparameters and our transformed data resulting in a R-squared of 0.85 and very good MAE of 87.66 and good MSE of 20,183.56. The predicted vs testing plot showcases the desired linear pattern and our residuals plot showcases the fixed constant variance and linearity. These results are superior to

the previous minimal KNN initialization and the linear model. We note that the prices are predicted in log-space for which they must be transformed back for normal price scale.

The strengths of this algorithm rely on the non-linear relationships in the data with which it leverages the local patterns of neighbor distances as phone hardware patterns may cluster in price ranges, as opposed to assuming a global linear relationship for the data set. Since the relationship between phone hardware and price may not be strictly linear, our KNN model offers better performance.

Conclusions

We used multiple machine learning models along with statistical processes to determine the best predictors of mobile device prices in America and make meaningful price predictions based on the given data. We found that storage capacity, weight, and RAM of the device are some of the strongest predictors of price due to the PCA and Random Forest models. Finding the strongest predictors of price can help consumers and businesses. This can help consumers because it enables them to know which device hardware and details to look for before purchasing. They can see that the more storage or RAM a device has, the more expensive it tends to be. A smart consumer may stray away from devices with larger storage capacities to save money if they don’t need the extra storage. This knowledge can benefit businesses too because knowing what specifications drive up prices allows them to invent more effective and cheaper ways to increase storage and potentially drive up revenue.

Along with identifying strong predictors, having a strong prediction model is equally as important. Companies can use the prediction model to see where prices and margins will be

going in the future to help make business decisions. They can also use it to see how much they can price a new device model that may have new specifications outside of the training data set.

Having strong predictors is the stepping stone for strong prediction. We were able to find strong predictors with logarithmic transformations and normalization. We know we have strong models because our testing metrics came out indicating strong performance. Having an R^2 value of almost 0.9 indicates strong prediction and MAE of $87 indicates very small error in our model. Overall our consumers and suppliers can use these models confidently to help make their device decisions.