Predictive Analytics HW 3

GitHub Link

Problem Statement

Sub-Field: Shopping
Problem Statement: Can we predict how expensive a laptop would be given different features of the laptop?

Data Collection

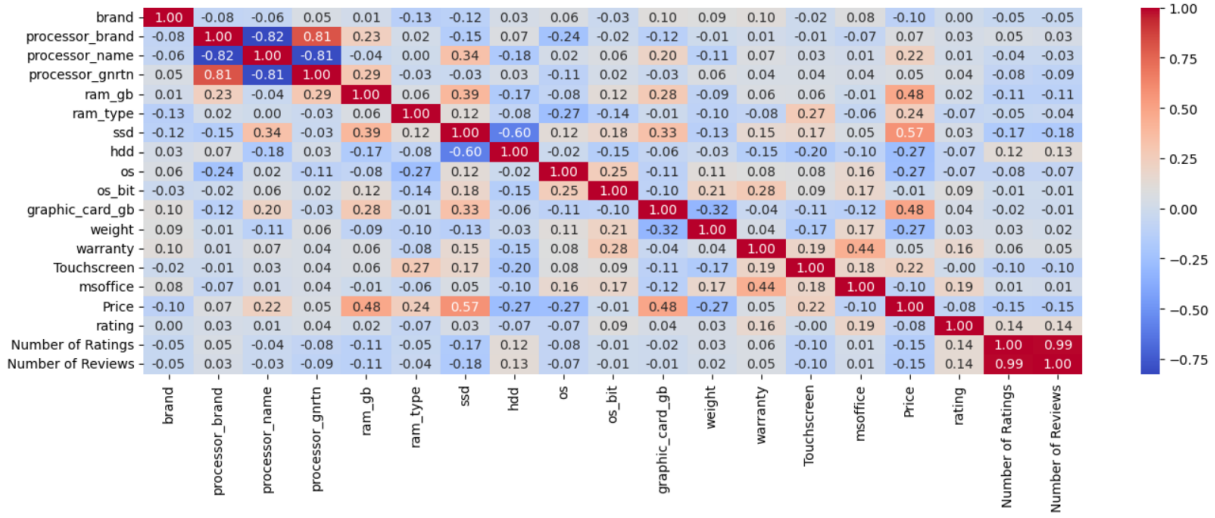
The Dataset was obtained from this Kaggle link.

Preprocessing

The following steps were taken to preprocess the data:

- Duplicate data points were dropped
- The categorical columns were encoded using Scikit Learn's Label Encoder
- Binary Variables were encoded using 1s and 0s
- Numerical columns that had values like 'GB' or '2 ratings', etc, were cleaned to ensure they had only numeric values
- As this is a classification task, the price was turned into a categorical variable using the following criteria:
    - <= 800: Affordable (0)
    - 800 < x <= 1500: Expensive (1)
    - 1500 < x <= 2200: Very Expensive (2)
    - > 2200: Outrageous (3)

For Feature Selection, columns that had very high correlation to each other were dropped: the processor name, processor generation and processor brand all had very high correlations, so we just kept the processor brand. The number of reviews and number of ratings also had very high correlation, so we only kept the number of ratings.



For Normalization/Standardization, since most of the variables are categorical, standardization is not really needed for most columns; we only standardize the Number of Ratings column.

Modeling

We split the data into training (80%) and test (20%) first. The following results were obtained from all the models:

Model	Accuracy
KNN	72.05%
SVM	76.40%
Decision tree	65.22%
Random Forest	73.91%
Gradient Boosting	80.12%

Hyperparameter Tuning

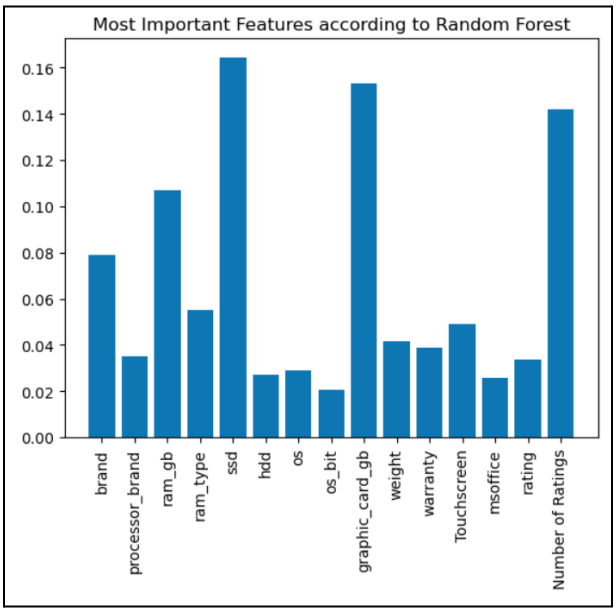
The hyperparameter tuning was done on the random forest and gradient boosting models, here are the results:

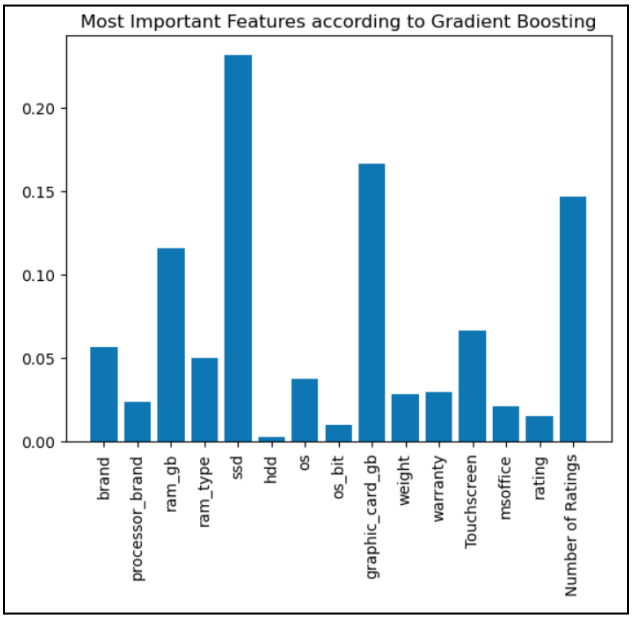
- Random Forest:
    - Best Random Forest Accuracy: 0.7702

    - Best Random Forest Parameters: {'class_weight': None, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 150}
- Gradient Boosting:
    - Best Gradient Boosting Accuracy: 0.8075
    - Best Gradient Boosting Parameters: {'learning_rate': 0.1, 'min_samples_split': 5, 'n_estimators': 100}

Conclusions

The tuned gradient boosting model performed the best with an accuracy of 80.75%. A graph of the most predictive features according to the random forest and gradient boosting models are shown below:





The most predictive features according to both models are ssd, graphics card, brand and ram. This seems pretty accurate when thinking about laptop prices in real life. Note that we don't count the number of ratings as according to the documentation: "impurity-based feature importances can be misleading for high cardinality features (many unique values)", so we can ignore the high importance of the Number of Ratings.