Laptop Price Prediction using Machine Learning Algorithms

EE 559 Course Project

Data Set : Laptop Prices

Sanskruti Raut, sanskrut@usc.edu Ruifan Wang, ruifanwa@usc.edu

April 26, 2024

1 Abstract

In today's digital era, laptops have become a crucial tool for performing variety of tasks from work to entertainment. In such scenario, determining the price of laptops with the proliferation of numerous laptop models in the market has become a challenging task. This project addresses this challenge by leveraging various Machine Learning algorithms, namely, Linear Regression, Random Forest, Neural Network, and Support Vector Regression to predict the price of laptops. Here, we have utilized the Laptop Prices dataset encompassing key features such as CPU, Company, Memory, GPU, RAM, etc. among others and employed state-of-the-art ML algorithms to accurately predict laptop prices based on these features. To evaluate the performance of the regression models, key metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2) are computed and compared. These metrics provide insights into the models' predictive accuracy and their ability to capture the underlying relationships between the features and laptop prices. The findings reveal that Random Forest and Support Vector Regression exhibits superior performance, as evidenced by its lower RMSE and MAE values and higher R^2 score compared to other models.

2 Introduction

2.1 Problem Assessment and Goals

These days laptops have become an integral part of professional and personal use, and the ability to predict their prices reliably is of utmost importance for consumers, retailers, and manufacturers alike. Hence, the accurate determination of laptop prices is a crucial task within the technology market, influenced by various factors such as hardware specifications, brand reputation, and market trends. Accurately determining laptop prices holds significant implications for various stakeholders in the technology market. There are diverse array of laptop models available in the market with multitude of features influencing their prices. Furthermore, the pricing dynamics are influenced by various factors such as technological advancements, brand name, and consumer demand, adding complexity to the prediction task.

We have utilized Laptop prices dataset from Kaggle. The dataset provided contains detailed information about various laptops from different manufacturers. Each entry in the dataset includes key specifications such as screen size, resolution, processor type, RAM, storage capacity, graphics card, operating system, weight, and price. This comprehensive dataset offers insights into the diverse range of laptops available in the market, enabling thorough analysis and comparison for various research and decision-making purposes.

Our aim is to address these challenges which requires complicated analytical techniques capable of capturing the intricate relationships between the myriad of features and the resulting price variations. Machine learning algorithms offer a promising approach to tackle this problem by leveraging historical pricing data and extracting meaningful patterns to make accurate predictions.

3 Approach and Implementation

3.1 Dataset Usage

Prior to training our models, we conducted extensive feature preprocessing and engineering to enhance the quality and informativeness of the data. This included standardization, and encoding categorical variables.The given original train dataset and test dataset were both loaded using the read_csv() function of the pandas library.

3.2 Preprocessing

The preprocessing steps involved in this project are described below:

- Data Loading: The original train dataset and test dataset were both loaded using the `read_csv()` function of the pandas library.
- Handling Missing Values: The missing values in the dataset were identified and handled appropriately.
- Data Cleaning: Certain columns such as 'Ram' and 'Weight' were cleaned by removing units and converting them to the appropriate data types.
- Feature Extraction: The 'ScreenResolution' column was split into 'ResolutionWidth' and 'ResolutionHeight' to extract screen resolution width and height.
- Feature Engineering: New features such as 'TotalStorageGB' and 'PrimaryStorageType' were derived from the 'Memory' column to represent total storage capacity and the type of primary storage.
- CPU and GPU Parsing: The 'Cpu' and 'Gpu' columns were parsed to extract CPU brand, type, speed, GPU brand, and model information.
- Log Transformation: The target variable 'Price' was transformed using the natural logarithm to achieve a more normal distribution.

3.3 Feature Engineering

The feature engineering process involved the creation of new features and refinement as described below:

- Simplified CPU Features: Simplified CPU type dummies multiplied by the CPU speed were created and added as new features to the dataset.
- GPU Features: GPU brand and model were extracted from the 'Gpu' column and added as new features.
- Final Feature Selection: The final set of features used for modeling included 'Company', 'TypeName', 'Ram', 'OpSys', 'Weight', 'ResolutionWidth', 'ResolutionHeight', 'TotalStorageGB', 'CpuBrand', 'CpuType', 'CpuSpeedGHz', 'GpuBrand', and 'GpuModel'.

3.4 Training, Classification or Regression, and Model Selection

Among all the machine learning models, we have chosen Linear Regression, Support Vector Regression, K-Nearest Neighbors, Random Forest, and Neural Networks Model. Because we are trying to predict laptop prices based on their features, it's better to use regression models instead of classification models. Then, based on the characteristics of our training data, where we have both numerical and categorical data.

- **Linear Regression** In the process of training the models for predicting laptop prices, the Linear Regression model was selected first because of its efficiency and straightforwardness in handling regression tasks. This model assumes a linear relationship exists between the input variables (such as CPU, RAM, and TotalMemoryGB) and the single output variable, the laptop's price.

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n + \epsilon \tag{1}$$

where y is the dependent variable, β_0 is the intercept, $\beta_1, \ldots, \beta_n$ are coefficients, $x_1, \ldots, x_n$ are the explanatory variables, and ϵ is the error term, assumed to be normally distributed. Parameters were chosen based on the least squares criterion, minimizing the sum of squared residuals.

During the training phase, the model was fitted using a training dataset, ensuring the model learns to predict prices from the given features. The performance of the Linear Regression model was evaluated using metrics such as Root Mean Squared Error (RMSE) and R-squared (R^2), which provided insights into its predictive accuracy and the variance explained by the model. Despite its simplicity, Linear Regression served as a critical baseline for comparing more complex models, highlighting its importance in the initial stages of the model selection process.

- **K Nearest Neighbors(KNN)** In addition to Linear Regression, we incorporated the K-Nearest Neighbors (KNN) algorithm into our analytical arsenal to predict laptop prices. KNN is a non-parametric, instance-based learning method that operates on the principle of feature similarity. This method assumes that new laptop cases can be predicted based on their proximity to existing cases in the feature space. For our regression task, the KNN model was configured to determine the average value of the 'k' closest points, where 'k' is a hyperparameter representing the number of neighboring data points to consider. For the laptop dataset, with more than 900 observations in the training set, we choose k=5 as the parameter. One of the key advantages of KNN is its simplicity and the intuitive nature of its mechanism, which is particularly useful in scenarios where the data distribution is poorly understood. The performance of KNN was assessed using the same metrics, RMSE, MAE, and R^2, to ensure consistency and comparability in our evaluation. From the result, we can see that KNN performs better than Linear Regression.

- **Support Vector Regression(SVR)** After comparing the performances of different kernels, we find that an SVR (Support Vector Regression) model with a radial basis function (RBF) kernel works best for the regression task. The SVR model is particularly effective for handling high-dimensional data. The RBF kernel is mathematically defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \tag{2}$$

where γ is a hyperparameter that defines the extent of the influence of a single training example. High values of γ imply that the model considers only 'close' points in calculating the decision function, whereas low values mean 'far' points are included as well. The optimal selection of γ and the regularization parameter C was determined through grid search optimization in conjunction with cross-validation, thereby ensuring the model's robustness and generalization capabilities.

- **Neural Network** For the fourth model, we decided to include a Neural Network model in our laptop price prediction because we wanted to tap into its ability to understand the complexities of our data. Using Keras, we built a model with layers that can learn from the relationships between laptop features and their prices. We made sure to fine-tune it by adjusting parameters like the number of neurons and regularization strength to get the best performance. The architecture comprised several fully connected layers, also known as Dense layers, each equipped with rectified linear unit (ReLU) activation functions. ReLU helps introduce non-linearity into the model, enabling it to learn intricate relationships between input features and target variables. Additionally, we incorporated L2 regularization to prevent

overfitting by penalizing large weight values. To optimize the Neural Network's performance, we engaged in hyperparameter tuning using GridSearchCV. When we tested it, the model showed promise, and the errors in its predictions were notably small, meaning it's capturing a good amount of the price variability. This suggests that our Neural Network is doing a good job of understanding what factors influence laptop prices.

- **Random Forest** We also chose to implement Random Forest because it can help us determine feature importance. Although Random Forest was taught and discussed in class, we did not go through the technical depth of Random Forest in EE559. Random Forest is a common machine learning technique that can be used for both classification and regression tasks. It involves constructing numerous decision trees during training time and then using the output class of the majority of the trees (in the case of classification) or the mean prediction (in the case of regression) as the final output. It also helps us to overcome the problem of over-fitting that may happen to other models we chose.

- **How the parameters of the model were chosen.**To enhance the model's accuracy, we created a Random Forest model incorporating all of the original features. Upon analyzing the importance data frame, we discovered a high correlation between the 'inches' and 'weight' features. The model performed better when we only included the 'weight' feature. Our analysis of the top 20 important features revealed that RAM has a significant impact on laptop prices. Other factors such as 'Company,' 'ResolutionWidth,' 'ResolutionHeight,' 'TotalStorageGB,' 'CpuBrand,' 'CpuType,' 'CpuSpeedGHz,' 'GpuBrand,' and 'GpuModel' also contribute to the model's prediction.

- **Degrees of freedom and the number of constraints** Degrees of freedom in a statistical model measure the amount of independent information available to estimate parameters or evaluate the model's fit. In laptop datasets, the degrees of freedom are calculated as the total number of observations minus the number of parameters. For instance, the SVR model has 902 observations, six numerical parameters, and seven categorical parameters. The degree of freedom for the SVR model is 889, which is more than ten times the number of constraints. The high degree of freedom of the dataset leads to more reliable statistical inference.

4 Results and Analysis: Comparison and Interpretation

Performance comparison The following table summarizes the performance metrics (RMSE, MAE, and R^2) for the Baseline model, Random Forest, SVR, and Linear Regression models. Each model was evaluated based on 902 observations involving 6 numerical parameters and 7 categorical parameters.

Model	RMSE	MAE	R^2
Baseline Model	0.42571	0.33473	0.54015
Linear Regression	0.29380	0.22947	0.78098
KNN	0.30863	0.217808	0.75831
SVR	0.25495	0.18659	0.83507
Neural Networks	0.29822	0.21993	0.77432
Random Forest	0.24788	0.18386	0.84409

Table 1: Comparison of machine learning models based on RMSE, MAE, and R^2.

Analysis and interpretation of model results. For the baseline model, we used Linear regression with only processed numerical data, including 'Ram', 'Weight', 'ResolutionWidth', 'ResolutionHeight', 'TotalStorageGB', and 'CpuSpeedGHz.' The model achieved an R^2 value of 0.54015, which means that our model explained half of the test data. Although this is not a significant result in general, we can see that our feature engineering that modifies categorical parameters, such

Model	Observations	Numerical Parameters	Categorical Parameters
Baseline Model	932	6	0
Linear Regression	932	6	5
KNN	902	6	4
SVR	902	6	7
Neural Networks	902	6	7
Random Forest	902	6	7

Table 2: Number of observations and number of parameters of each model

as resolution, memory, and CPU, to numerical parameters has a meaningful impact on predicting laptop prices. So we can continue to work on more complex models that utilize both numerical and categorical parameters.

As depicted in Table 1, the Random Forest regressor emerged as the top performer among the models tested, with the lowest RMSE of 0.24788 and MAE of 0.18386, coupled with the highest R^2 value of 0.84409. These results underscore Random Forest's ability to handle the complex nonlinear relationships within the data, thanks to its ensemble method of constructing multiple decision trees and averaging their predictions to reduce overfitting.

The Support Vector Regression (SVR) also displayed commendable predictive capabilities, with RMSE and MAE scores closely trailing those of the Random Forest model, and an R^2 score of 0.83507. This suggests that SVR's kernel trick is adept at managing the high-dimensional feature space typical of laptop specification data.

Linear Regression showed higher RMSE and MAE values of 0.29380 and 0.22947, respectively, and a lower R^2 value of 0.78098. While these figures indicate a decent fit, they reflect Linear Regression's limitations in capturing more complex patterns within the data.

These models were analyzed on a robust dataset comprising 902 observations, with each observation described by 6 numerical and 7 categorical parameters, revealing significant insights into the predictive dynamics of laptop pricing. The performance measures reported herein were based on the models' ability to generalize to unseen data, which is critical for practical deployment scenarios.

In conclusion, the Random Forest model's superior performance in our experiments highlights its suitability for the task of laptop price prediction. It effectively balances the trade-off between bias and variance, making it the most reliable choice among the tested algorithms.

5 Libraries used

In the preprocessing stage, we utilized several libraries including Pandas, NumPy, Seaborn, and Matplotlib for data manipulation, visualization, and basic operations. The main preprocessing steps, such as data loading, cleaning, and feature extraction, were performed using Pandas functions. Additionally, the scikit-learn library was employed for more advanced preprocessing tasks, such as data scaling using StandardScaler and one-hot encoding of categorical variables using OneHotEncoder. These preprocessing steps were implemented using scikit-learn's Pipeline and ColumnTransformer classes to ensure efficient and reproducible data preprocessing pipelines.

For modeling, scikit-learn's RandomForestRegressor for building a random forest model. Furthermore, the implementation for Support Vector Regression (SVR) was done using scikit-learn's SVR class. Evaluation metrics such as mean squared error (MSE), mean absolute error (MAE), and (R^2) score were calculated using scikit-learn's mean_squared_error and r2 score functions.

In addition, we explored deep learning techniques using TensorFlow and Keras. TensorFlow was utilized for building and training neural network models, with Keras providing a high-level interface for constructing neural networks. This included defining the model architecture, compiling the model, and training it using the training data.

No libraries or functions outside of the allowed list for EE 559 methods were used in this project. All implementations adhere to the guidelines and requirements of EE 559.

6 Contributions of each team member

Sanskruti and Ruifan teamed up to tackle the data preprocessing phase, brainstorming together to refine our methods and ensure top-notch results. We had regular meetings every week where we brainstormed ideas, troubleshooted issues, and implemented improvements to our preprocessing pipeline.

When it came to the machine learning part, we split the workload evenly. Sanskruti took on Linear Regression, KNN, and Neural Networks, while Ruifan dove into SVR and Random Forest.

As for writing the report, we divided and conquered. Sanskruti focused on sections like the Abstract, Introduction, Approach and Implementation, References, and Libraries Used. Meanwhile, Ruifan handled sections on Training, Classification or Regression, Model Selection, Results and Analysis, and Summary and Conclusion.

This balanced approach not only ensured that each task was completed thoroughly but also allowed us to leverage each other's strengths. Plus, it taught us valuable skills like teamwork, effective communication, and time management along the way. Working together as a team made the entire process smoother and more rewarding.

7 Summary and conclusions

In this study, we endeavored to address the complex challenge of predicting laptop prices in a dynamic and diverse marketplace. Utilizing a dataset replete with specifications such as CPU performance, GPU brand, memory size, and company name, we applied various machine-learning algorithms. Our methodology involves data preprocessing, feature engineering, and model evaluation to ensure the integrity and validity of our findings.

The core of our analysis pitted five predictive models against one another: baseline Linear Regression model with only numerical parameters, advanced Linear Regression model, K Nearest Neighbor (KNN), Support Vector Regression (SVR), Random Forest, and Neural Networks. The Random Forest model distinguished itself, demonstrating the highest R^2 value of 0.84409 and the lowest RMSE and MAE. While linear regression provided a baseline reference, the SVR model affirms its efficacy in high-dimensional spaces and performs better.

Key insights from this project include the critical role of feature selection and the intricate balance required between model complexity and generalizability. Realizing that sophisticated models like Random Forest can significantly outperform simpler models in complex tasks was particularly instructive. For future work, it would be intriguing to explore the integration of time-series data to capture price trends and fluctuations over time. Additionally, deploying deep learning techniques could unveil even more nuanced patterns in the data. Further refinement of feature engineering through the use of unsupervised learning could also enhance model performance.

Through this project, we have deepened our understanding of machine learning's practical applications. We have honed our skills in data manipulation, model tuning, and interpreting model outputs, which are invaluable hands-on experiences in machine learning.

References

[1] Pandas Documentation, available at `https://pandas.pydata.org/docs/`.

[2] NumPy Documentation, available at `https://numpy.org/doc/`.

[3] Seaborn Documentation, available at `https://seaborn.pydata.org/`.

[4] Matplotlib Documentation, available at `https://matplotlib.org/stable/contents.html`.

[5] Scikit-learn Documentation, available at `https://scikit-learn.org/stable/user_guide.html`.

[6] TensorFlow Documentation, available at `https://www.tensorflow.org/guide`.

[7] Kaggle Laptop Prices Documentation, available at https://www.kaggle.com/datasets/ara001/laptop-prices-based-on-its-specifications/data.