Indian Institute of Technology, Dharwad

CS209 : Artificial Intelligence
And
CS214 : Artificial Intelligence Laboratory
Project Report : **Turkish e-Commerce Market Analysis and Computer Price Prediction**

Course Instructor:
Dr. Dileep A.D.

Mentor Name:
Mr. Vikas Kumar

Submitted by:

1.	Samartha	CS23BT019
2.	Yashas Subramanya	MC23BT007
3.	Nilesh Kumar	CS23BT006
4.	Nandru Jagan	CS23BT031

Abstract

This project addresses the problem of analyzing desktop computer configurations and pricing trends within the Turkish e-commerce market. Using a dataset containing detailed hardware specifications such as processor type, RAM, storage, graphics card, and display features, the objective was to uncover how various components influence price and to develop models capable of predicting pricing based on these features. To achieve this, extensive exploratory data analysis (EDA) was conducted to understand the frequency and popularity of different hardware setups. Multiple regression techniques were implemented, including Linear Regression, Polynomial Regression, Support Vector Regression (SVR), Neural Networks, Decision Trees, and Random Forests. Among these, the Random Forest model demonstrated the highest performance with an R^2 score of 0.88, making it the most effective predictor. The results reveal that key specifications significantly drive pricing trends, and the developed model can accurately estimate prices for any given combination of components. These insights can assist both consumers and retailers in making informed, data-driven decisions in the desktop hardware market.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, the rapid expansion of the e-commerce sector has significantly impacted the way consumers purchase electronic devices, especially desktop computers. With a wide array of hardware configurations available, ranging from processors and memory to graphics cards and displays, understanding the relationship between these components and the final price becomes crucial. For both consumers aiming to make cost-effective choices and retailers seeking competitive pricing strategies, data-driven insights into pricing trends can offer a strong advantage.

This project focuses on analyzing and predicting desktop computer prices based on hardware specifications using machine learning techniques. The dataset used for this analysis originates from Turkish e-commerce platforms and comprises detailed information on 28 attributes, including processor details, RAM capacity, storage type, GPU model, screen size, and resolution.

The primary objective is twofold:

1. To explore the influence of various hardware components on the overall price of a desktop computer.

2. To develop accurate machine learning models capable of predicting prices for unseen configurations.

To achieve these goals, we perform Exploratory Data Analysis (EDA) to uncover trends in component popularity and price distributions. We then implement and compare multiple regression techniques, including Linear Regression, Polynomial Regression, Support Vector Regression (SVR), Decision Trees, and Random Forests, evaluating their performance based on the adjusted R^2 score.

By the end of this project, our aim is to provide a predictive system that estimates pricing with high accuracy and delivers insights that support informed decision making for various stakeholders in the desktop hardware market.

1.1 Dataset Description

The dataset consists of 28 attributes capturing key technical specifications and pricing information. The primary attributes include:

1.1.1 General Information

- **Brand (Marka)** – Manufacturer of the computer (e.g., XASER, HP, Dell).
- **Price (Fiyat)** – Selling price of the desktop (in Turkish Lira - TL).
- **Country of Manufacture (Menşei)** – Country where the product was manufactured.

1.1.2 Processor and Memory Specifications

- **Processor Type (İşlemci Tipi)** – Brand and model of the CPU (e.g., Intel Core i5, AMD Ryzen 7).
- **Base Clock Speed (Temel İşlemci Hızı)** – Processor's base speed in GHz.

- **Processor Frequency (İşlemci Frekansı)** – Maximum speed of the processor (above 3.00 GHz, etc.).
- **RAM (Sistem Belleği)** – Installed system memory (e.g., 8 GB, 16 GB).
- **Expandable Maximum Memory (Arttırılabilir Azami Bellek)** – Maximum RAM capacity.
- **SSD Capacity (SSD Kapasitesi)** – Storage capacity of the SSD (e.g., 256 GB, 512 GB).

1.1.3 Graphics and Display Features

- **Graphics Card (Ekran Kartı)** – Model of the GPU.
- **Graphics Card Memory (Ekran Kartı Kapasitesi)** – VRAM capacity (e.g., 4 GB, 8 GB).
- **Graphics Memory Type (Ekran Kartı Bellek Tipi)** – Type of GPU memory (e.g., DDR3, GDDR5).
- **Graphics Card Type (Ekran Kartı Tipi)** – Dedicated or integrated GPU.
- **Monitor Size (Ekran Boyutu)** – Screen size in inches (e.g., 24", 27").
- **Screen Refresh Rate (Ekran Yenileme Hızı)** – Display refresh rate (e.g., 75 Hz, 165 Hz).
- **Panel Type (Panel Tipi)** – Monitor panel technology (VA, IPS, TN).

1.1.4 Connectivity and Physical Attributes

- **Connections (Bağlantılar)** – Available ports such as HDMI, DisplayPort.
- **Weight (Cihaz Ağırlığı)** – Total weight of the desktop computer.

2 Methodology

To predict desktop computer prices effectively, we adopted a structured machine learning pipeline involving data preprocessing, feature selection, model training, and evaluation. The entire methodology was implemented using Python with essential libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn.

2.1 Data Conversion

The dataset was collected from Turkish e-commerce platforms and consisted of 28 columns describing various specifications of desktop computers, such as processor type, graphics card, storage, RAM, and warranty type. Since the original data were in Turkish, a significant part of the preprocessing phase involved translating values to make the dataset user-friendly, universal, and suitable for machine learning.

2.1.1 Column Translation

A dictionary mapping was used to convert Turkish column names to English. For example, *"Marka"* was renamed to *"Brand"*, and *"Fiyat"* to *"Price"*. This rewording helped simplify analysis and ensured consistency throughout the execution process.

2.1.2 Value Conversion and Cleaning

Key categorical values were also translated into English using a series of `replace()` and `map()` operations. For instance:

- *"Dahili Ekran Kartı"* was translated to *"Integrated Graphics Card"*.
- Storage values like *"1 TB"* and *"2 TB"* were converted to their numerical equivalents (e.g., 1024, 2048).
- Screen sizes expressed as ranges or in non-standard formats (e.g., *"5 - 5,5 inç"*) were averaged and mapped to float values.
- Turkish usage categories like *"Ev - Okul"* and *"Oyun"* were mapped to their English counterparts (*"Home - School"* and *"Gaming"*).
- Warranty types and brand-specific terms (e.g., *"HP Türkiye Garantili"*) were normalized to English equivalents (*"HP Turkey Guaranteed"*).

2.1.3 Numerical Conversion

Several features such as *SSD Capacity*, *RAM*, *Expandable Max Memory*, and *Graphics Card Memory* were originally stored as strings with units (e.g., "8 GB"). These were cleaned using lambda functions to extract numerical values. Processor generation (e.g., "10. Nesil") was also reduced to a single integer value, and all core counts and base processor speeds were cast to numerical types to support quantitative analysis.

2.1.4 Data Normalization and Imputation

To address inconsistencies, certain uncommon or ambiguous values like *"4 GB ve altı"*, which means *4 GB or less*, were replaced with randomized discrete values from a predefined distribution (e.g., 1, 2, or 4), simulating a realistic hardware assumption. Null or missing values were later handled during the preprocessing phase.

2.1.5 Initial Data Cleaning

Before Exploratory Data Analysis (EDA), columns with more than 70% missing values were dropped from the dataset. These columns were considered statistically insignificant for meaningful analysis, as working with such limited data could introduce bias or result in unreliable interpretations.Duplicate values were also removed during this phase. This step helped streamline the dataset, retain analytical clarity, and focus on features with robust representation.

This preprocessing ensured that all data fields were clean, consistent, and ready for exploratory analysis and machine learning workflows.

2.2 Exploratory Data Analysis (EDA)

2.2.1 Univariate Analysis

We use the most revealing graphs to understand individual characteristic distributions and trends:

1. Brand

Acer(blue region) and HP(orange region) constitute the majority of the dataset, representing approximately 41% and 36% of the total entries, respectively. Dell follows with a modest share of around 5%, while the remaining brands each account for less than 5%. This distribution highlights the dominance of Acer and HP in the Turkish desktop computer market, as reflected in the collected data.

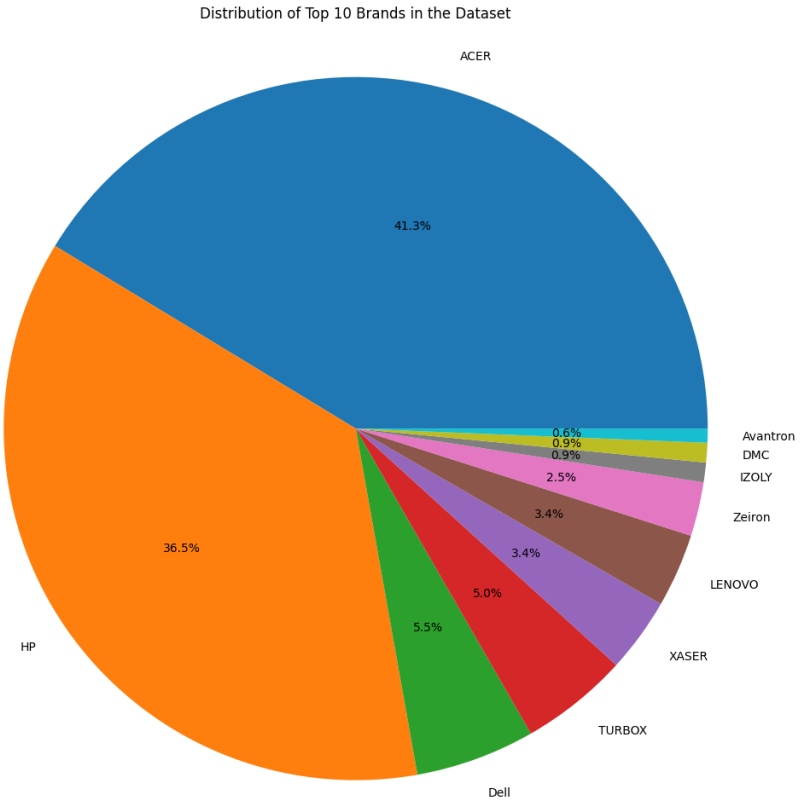

Figure 2.1: Distribution of Top 10 Brands

2. Price

The dataset shows a price range from 2649.0 TL to 3,80,353.35 TL. However, the majority of desktops are priced between 20,000 TL to 30,000 TL, indicating that mid-range models dominate the market, with a few high-end configurations pushing the upper limit of the price distribution.

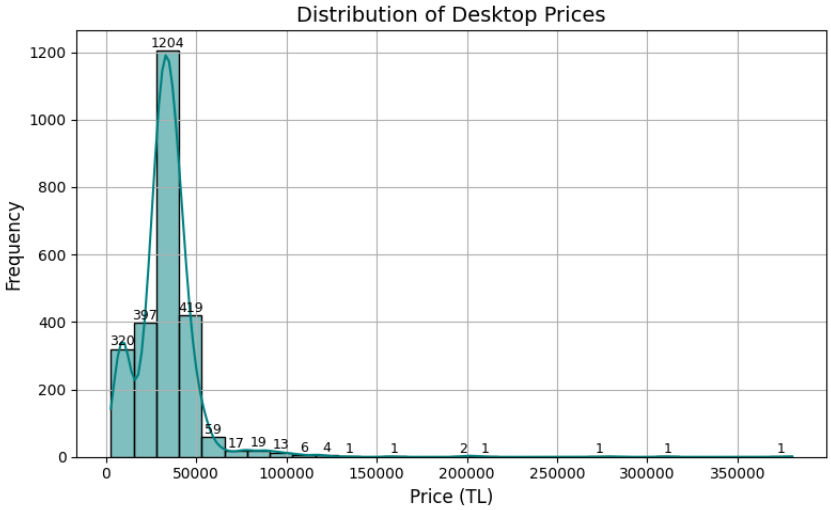

Figure 2.2: Price Distribution

3. Processor

1. **Top Processor Types:**
   Intel Core i5 dominates by a huge margin, followed by i3 and i7. AMD processors have significantly lower representation.

2. **Core Count Distribution:**
   Most processors have **4, 8, or 12 cores**, with 12-core being the most common.

3. **Processor Generation Distribution:**
   Newer generations, especially **12th and 13th gen**, are widely used, showing a clear trend toward recent Intel chips.

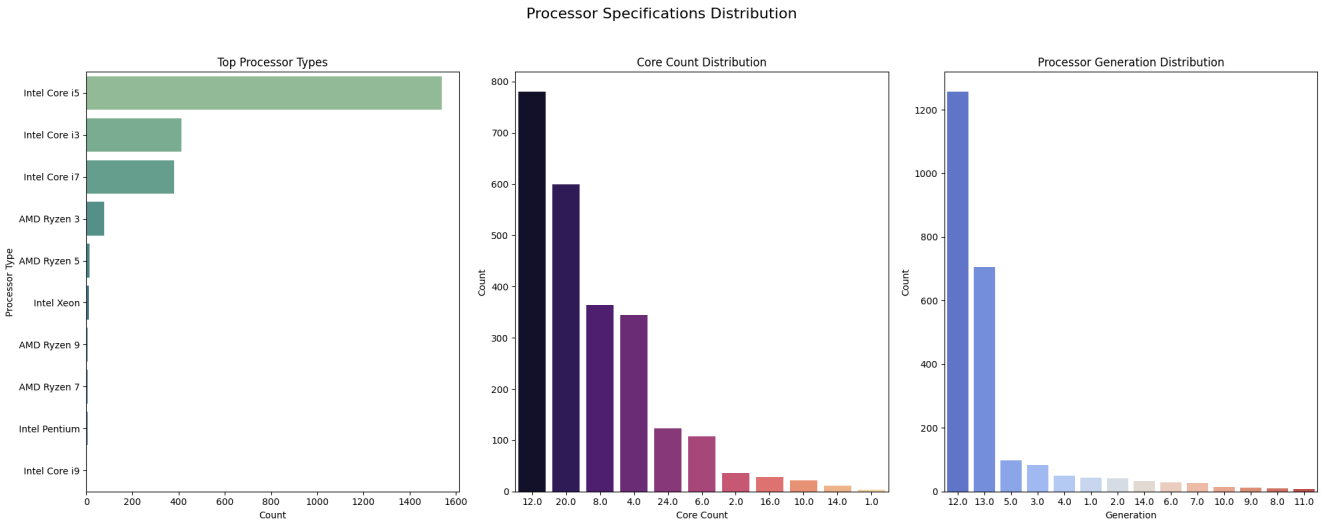

Figure 2.3: Distribution of Top 10 Processors

4. RAM and SSD Capacity

RAM Distribution: The most common RAM size is 16GB, marking it as the preferred choice for modern systems. Usage is also high for 8GB and 32GB, indicating a mix of budget and performance-focused configurations. Very high capacities like 96GB and 128GB are rare, likely restricted to workstations or specialized systems.

SSD Capacity Distribution: 4TB SSDs surprisingly dominate the dataset, suggesting either enterprise usage or a shift toward high-capacity storage. Mainstream capacities like 256GB, 512GB, and 1TB are also widely used, while older or smaller SSDs (e.g., 128GB and below) appear nearly phased out.

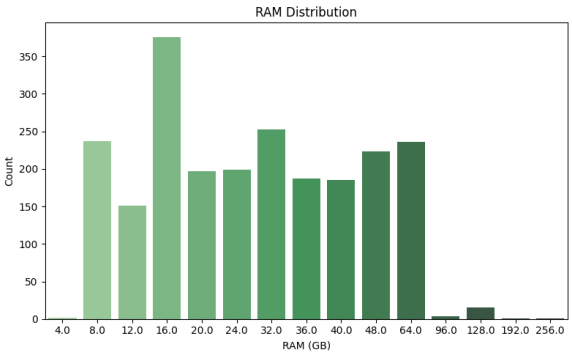

Figure 2.4: Distribution of RAMs

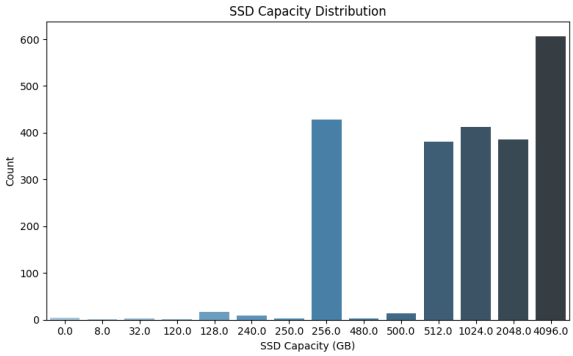

Figure 2.5: Distribution of SSDs

5. Usage Purpose

The dominance of office-oriented configurations suggests bulk procurement of systems, likely driven by the current shift toward computer science and digital operations in modern workplaces.

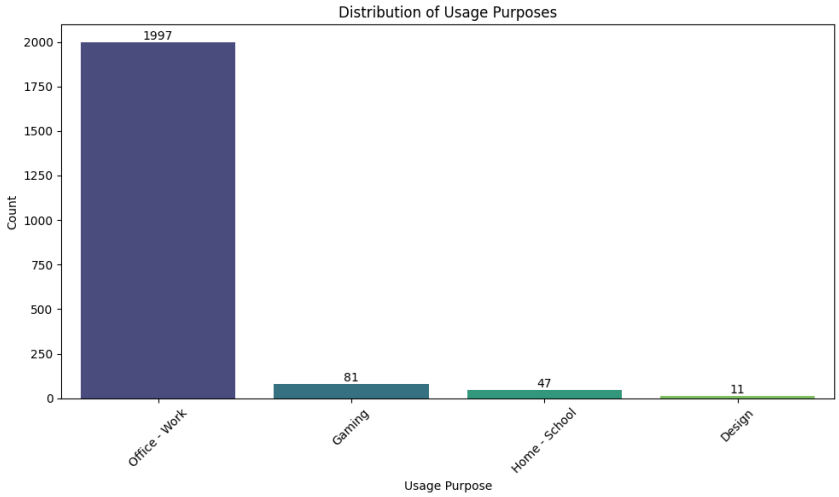

Figure 2.6: Usage Purposes

6. Operating Systems

Windows is the most popular OS followed by Free Dos as these systems are available at a cheaper price than others like Mac OS.

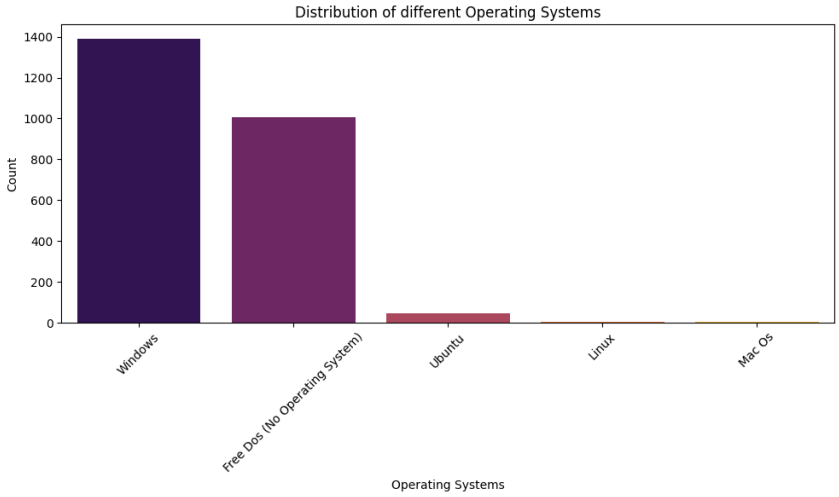

Figure 2.7: Distribution of Operating Systems

2.2.2 Bivariate Analysis

The best pairwise graphs with Average Price as one of the attributes were chosen to explore relationships between two variables effectively:

1. Price vs Brand

Coldpower systems are priced significantly higher than others, likely due to niche targeting or high-performance configurations. Brands like Dell, Redragon, and Revenge show the lowest average prices, aligning with widespread availability and mass adoption for standard business or educational needs. These affordable brands likely dominate in bulk purchases due to cost-efficiency.

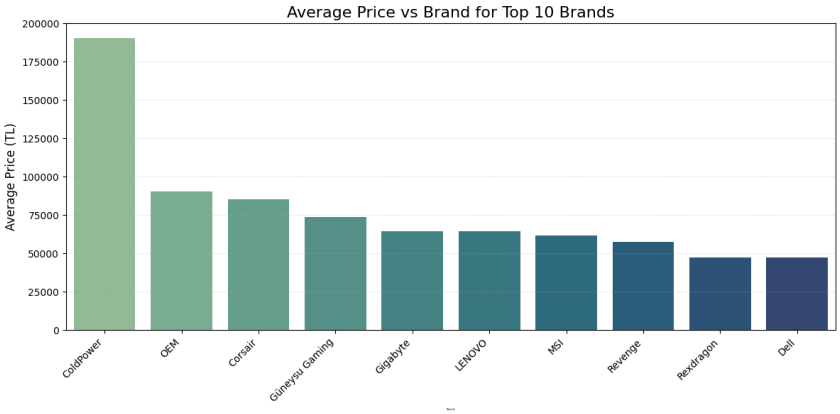

Figure 2.8: Average Price vs Brand distribution

2. Price vs Processor Type

Systems with Intel Core i9, AMD, and Xeon processors top the chart in terms of price, reflecting their high-end performance and suitability for compute-intensive tasks. Lower prices associated with processors like Core i5 and AMD Ryzen 3 indicate their alignment with everyday tasks such as browsing, office work, and education—highlighting their popularity in mass purchases for general use.

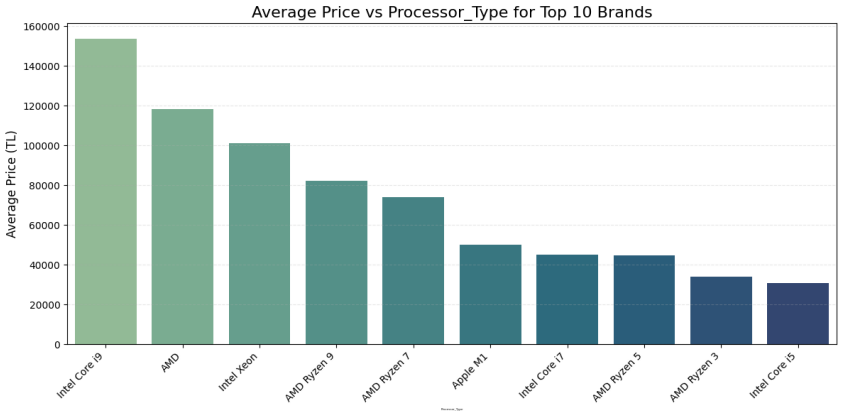

Figure 2.9: Average Price vs Processor Types

3. Price vs Graphics Card

The presence of workstation-class GPUs like the Nvidia RTX A5000 and RTX A4000 drastically increases the average price, indicating their usage in specialized fields such as AI, design, or rendering. Consumer-grade cards like the RTX 4070, 4060, and even 3080 show much lower average prices, suggesting that these systems target gamers or regular users rather than professionals in high-performance computing.

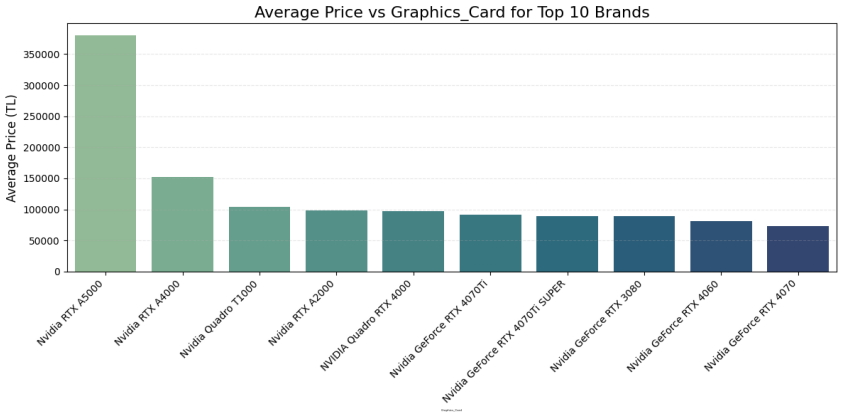

Figure 2.10: Average Price vs Graphics Cards

4. Price vs Usage Purposes

Laptops for Design have the highest average price, followed by Office - Work and Gaming, with Home - School being the most affordable. This reflects the intensive hardware demands of design work.

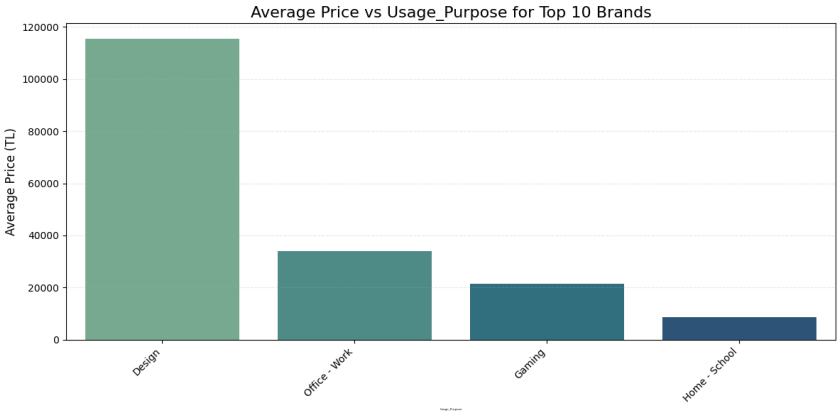

Figure 2.11: Average Price vs Usage Purpose

5. Price vs RAM

As RAM size increases, the average price of laptops generally rises, with a sharp spike observed at 128 GB. This surge is due to high-performance, workstation-grade laptops that typically come with 128 GB RAM, which are built for demanding tasks like 3D rendering, simulations, or data analysis. These machines also include top-tier CPUs, GPUs, and large SSDs, making the overall hardware package significantly more expensive. Thus, the spike isn't solely due to RAM, but the premium configurations that usually accompany it.

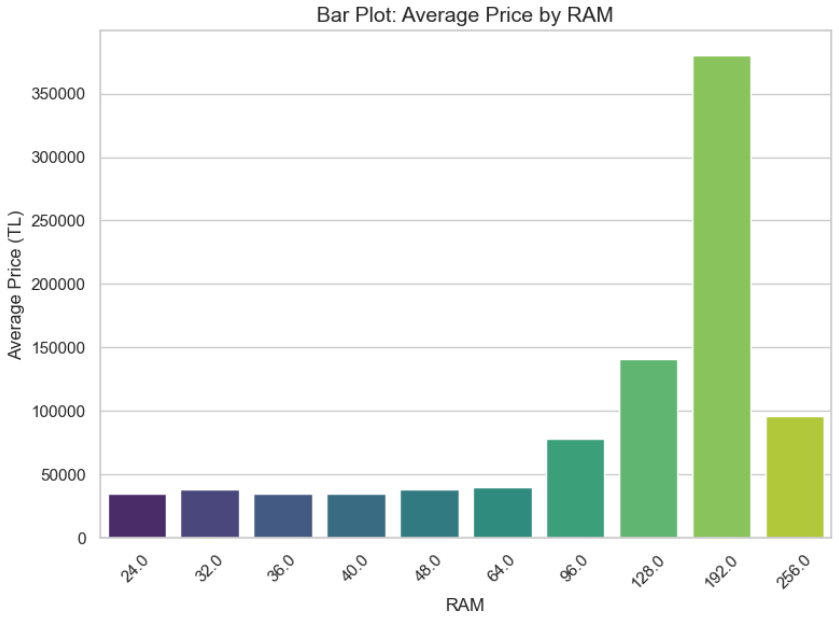

Figure 2.12: Average Price vs RAM

6. Price vs SSD

Generally, SSD prices increase with capacity, reflecting greater storage and performance value. However, the 8 GB capacity shows a sudden and illogical price spike, which is likely an outlier. This could result from a data entry error, a mislabeled product, or a rare, high-end system using a small boot SSD alongside massive secondary storage. Since 8 GB is far below modern SSD norms, the spike doesn't represent actual market trends and should be treated as an anomaly in the dataset.

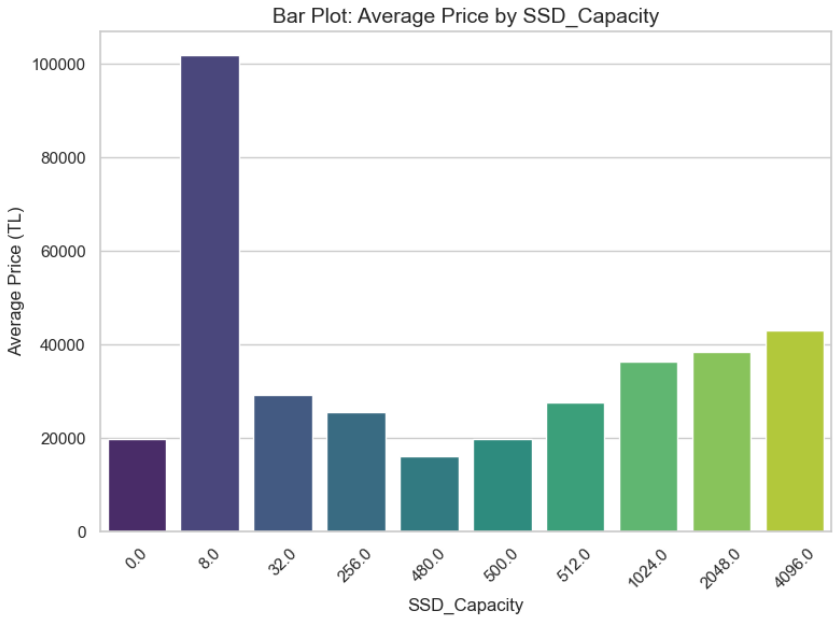

Figure 2.13: Average Price vs SSD

7. Price vs Processor Generation

Prices generally increase with newer processor generations, peaking at 14th gen. The 5th gen processor price spike likely stems from limited availability due to discontinuation, niche demand for compatible legacy systems or specific features, and inflated prices for rare or high-end models sought after by enthusiasts or for specialized uses.

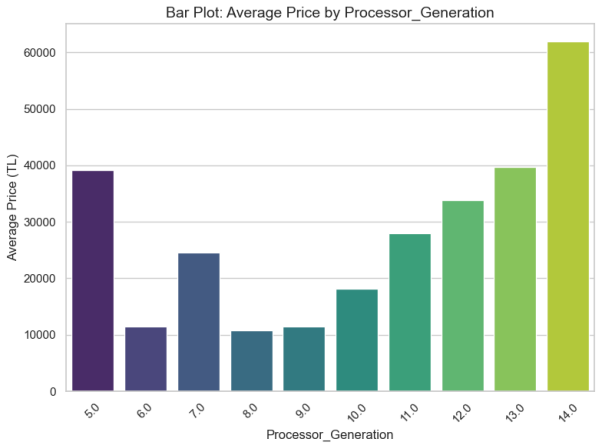

Figure 2.14: Average Price vs Processor Generation

2.2.3 Multivariate Analysis

We leveraged the most informative multivariate visuals to capture patterns across multiple features:

1. Average Price vs RAM vs SSD

The heatmap shows a clear trend where average prices increase with higher RAM and SSD capacities. Devices with 32GB+ RAM and SSDs of 512GB or more have the highest average prices, peaking around 38,338 TL. Conversely, systems with lower RAM (0–8GB)

and smaller SSDs (0–256GB) have the lowest average prices, as low as 15,730 TL. Interestingly, once SSD exceeds 512GB, the price jumps more significantly across all RAM categories. This confirms that both RAM and SSD are strong price drivers, and high-end systems (e.g., for professionals or enthusiasts) are priced substantially higher.

Figure 2.15: Average Price vs RAM vs SSD comparsion on segmented values

2. Radar charts for Brands

The radar charts highlight key brand differences. Dell consistently scores the highest across all specs including price, RAM, processor cores, and SSD, indicating premium features and likely a higher-end market position. HP also performs strongly in core count, RAM, and SSD but is slightly lower in price, possibly offering a balanced value proposition. In contrast, Acer ranks lower in all parameters, suggesting a budget-oriented approach with compromises on specs. Overall, Dell leads in performance and pricing, HP offers a close second with value, and Acer caters to cost-conscious buyers.

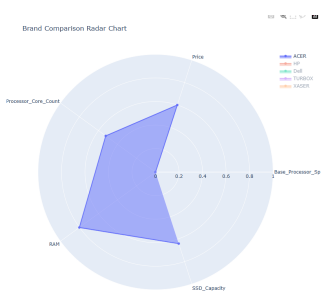

Figure 2.16: ACER

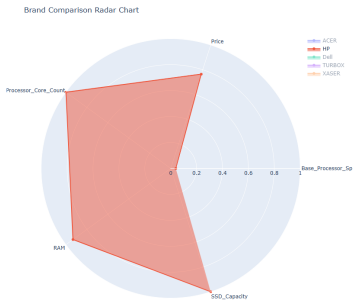

Figure 2.17: HP

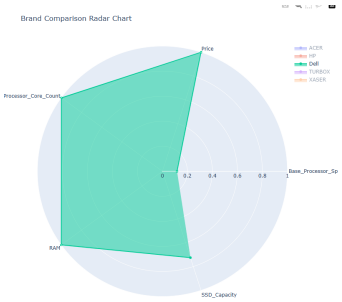

Figure 2.18: DELL

2.3 Data Preprocessing and Feature Engineering

2.3.1 Handling Missing Values

1. Since the **price** and several other numerical features in our **data set are highly skewed**, we used the **median** to fill in the missing values in numerical columns. Unlike the mean, the median is robust to outliers and provides a more reliable central value for skewed distributions like ours.

2. For **categorical features**, we used the **mode**, as it effectively captures the most frequent category without introducing bias.

2.3.2 Outlier Detection

A few outliers were identified and removed based on **visual inspection** and **domain knowledge**. For instance, extremely high prices or suspiciously low RAM values that did not align with realistic product specifications were manually filtered out. Although the IQR (Interquartile Range) method was considered, it was ultimately not used—particularly for features like RAM, where valid values ranged from as low as 1 GB to as high as 4096 GB. The IQR method incorrectly flagged over **66% of the data as outliers**, which would have led to significant data loss. Hence, a more practical, context-driven approach was adopted for outlier handling.

2.3.3 Encoding of Categorical Variables

Given the high cardinality of certain categorical features such as *Brand*, **label encoding** was preferred over one-hot encoding. One-hot encoding creates a separate binary column for each unique category, which can lead to an explosion in the number of features. In our case, this would significantly increase the dimensionality of the dataset, resulting in what is commonly referred to as the **curse of dimensionality**.

This curse not only increases memory usage and computational complexity but also introduces sparsity and multicollinearity, which negatively affect models like Linear and Polynomial Regression. Label encoding, on the other hand, assigns a unique integer to each category, preserving dimensional efficiency and avoiding these pitfalls. Although label encoding may introduce an artificial ordinal relationship, it was deemed acceptable given the trade-offs and model selection in our pipeline.

2.3.4 Correlation Analysis and Feature Reduction

A Pearson correlation matrix was computed for all features in the dataset to identify strongly correlated variables. A threshold of $|\rho| > 0.5$ was used to detect pairs of features with very strong linear relationships.

Results:
The correlation heatmap revealed several pairs of features with high correlations, suggesting redundancy. For example, *RAM* and *Expandable Max Memory* showed a strong correlation. These pairs were flagged for removal.

Feature Pruning:
Based on the correlation analysis, one feature from each strongly correlated pair was dropped. This was done to reduce multicollinearity, which can negatively affect model performance. The final dataset was reduced by removing features with $|\rho| > 0.5$.

This approach helped eliminate redundancy, simplify the dataset, and improve model reliability by addressing multicollinearity.

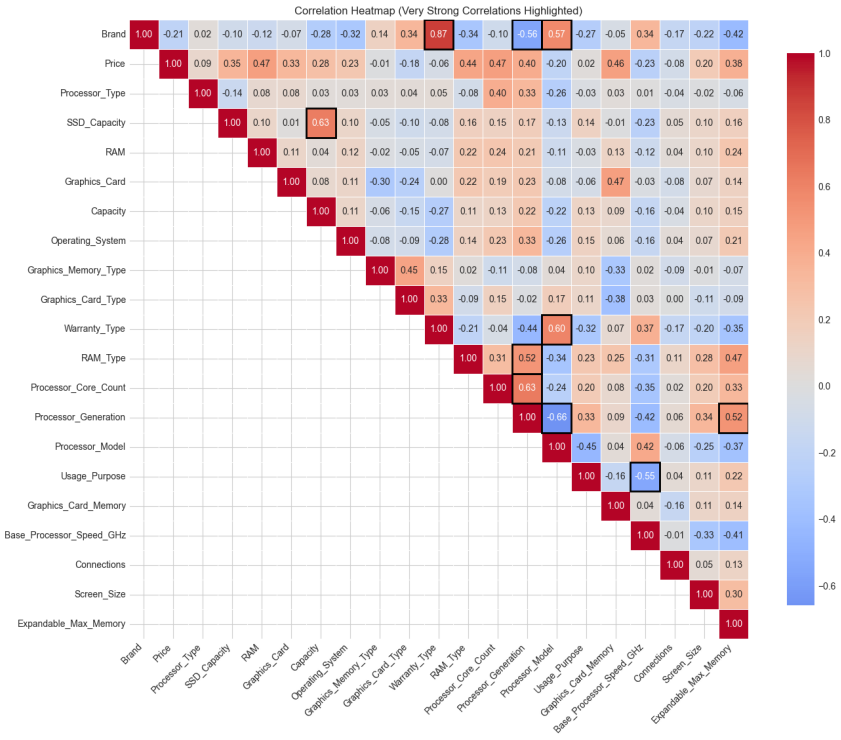

Figure 2.19: Correlation HeatMap

2.3.5 Multicollinearity using VIF

Variance Inflation Factor (VIF) is used to detect and mitigate multicollinearity in regression models. In our analysis, we used VIF to identify and remove features that contribute to multicollinearity, thereby improving model stability and interpretability.

VIF was applied to reduce redundancy in our dataset and ensure that each feature contributes unique information to the model. Features with high VIF values (greater than 10) were removed, improving the accuracy and stability of the regression models by addressing multicollinearity.

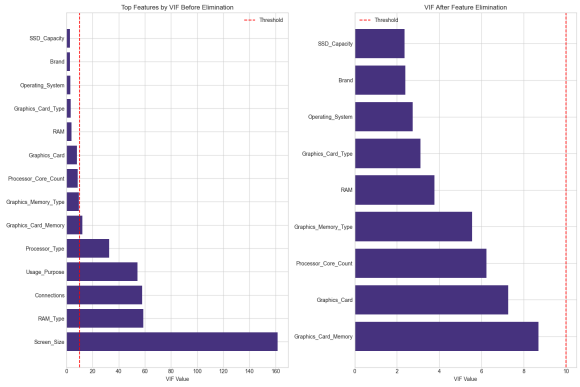

Figure 2.20: VIF of attributes

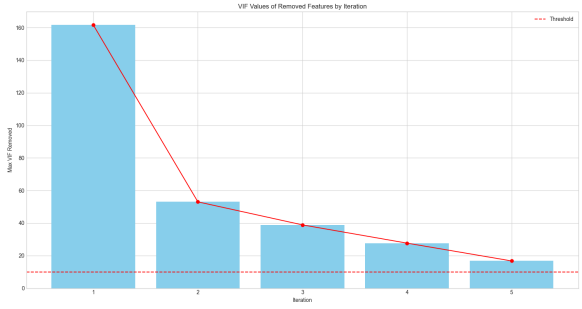

Figure 2.21: VIF with iteration

By removing highly correlated features, we reduced the feature set from 14 to 9, mit-

igating multicollinearity and ensuring that the remaining features provide independent, meaningful information. This process enhances the robustness of the model and improves its generalizability.

2.4 Model Training and Evaluation

Train/Validation/Test Split: We used 60:20:20 split for training , validation and test data in case of hypereparameter tuning whereas 80:20 split for the ones where there are no hyperparameters.
Standardization: The numerical features were standardized using **StandardScaler** to ensure each feature has a mean of 0 and a standard deviation of 1.

Evaluation Metrics

- **R^2 Score (Coefficient of Determination):**
  This metric quantifies the proportion of variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, with higher values indicating a better fit. An R^2 of 0.51 suggests that approximately 51% of the variability in desktop prices is explained by the model.

- **Root Mean Squared Error (RMSE):**
  RMSE measures the average magnitude of the prediction errors. It is calculated as the square root of the average squared differences between predicted and actual values. Lower RMSE values indicate better model performance. For instance, an RMSE of 14,500 TL implies that predictions deviate from actual prices by about 14,500 TL on average.

- **Adjusted R^2 Score:**
  Adjusted R^2 accounts for the number of predictors used in the model and adjusts the R^2 value accordingly. It helps prevent overfitting by penalizing the inclusion of irrelevant variables. It is especially useful when comparing models with different numbers of features.

2.4.1 Linear Regression

Model Fitting
Linear Regression was trained using the `LinearRegression` class from scikit-learn. The model minimizes the sum of squared residuals to find the best-fitting line.

Following the initial Linear Regression model, Ridge Regression was applied to improve performance by addressing overfitting through L2 regularization. Ridge Regression, implemented using `RidgeCV` from scikit-learn, automatically tunes the regularization parameter α using cross-validation.

Evaulation of the model

Model	R^2 Score	RMSE	Adjusted R^2 Score
Linear Regression	0.5107	14623.3538	0.5016
Ridge Regression (Best $\alpha = 100$)	0.5168	14530.8739	0.5078

Table 2.1: Model Evaluation Scores

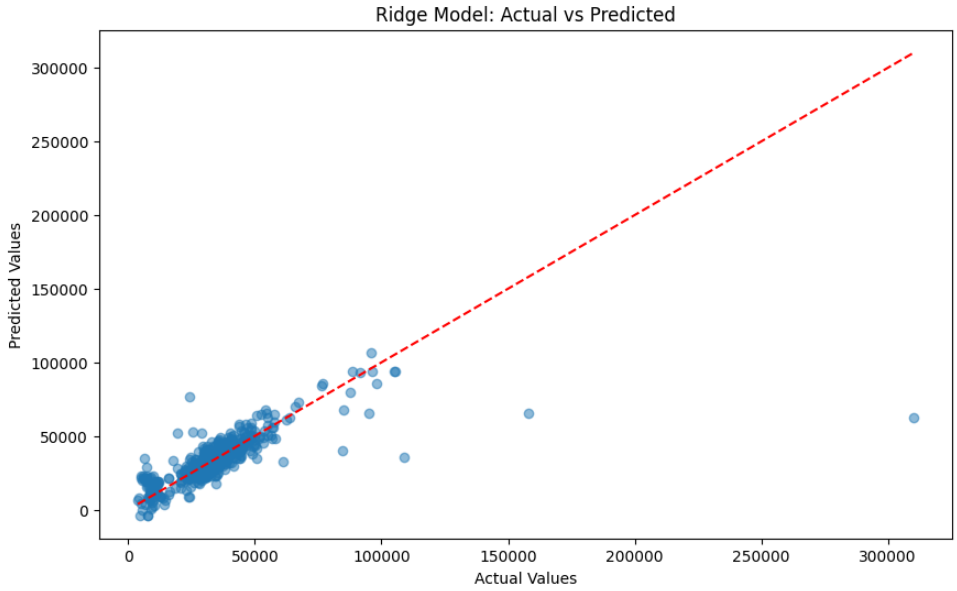

Figure 2.22: Linear Regression: Actual vs Predicted Plot

Inference

Ridge Regression slightly outperformed basic Linear Regression by reducing the RMSE and improving the R^2 scores. However, both models show moderate predictive performance. The relatively low accuracy can be attributed to high feature variability, skewed price distribution, and possible limitations from label encoding, which may have introduced ordinal relationships that don't exist.

2.4.2 Polynomial Regression (Degree 2 to 5)

Model Fitting

Polynomial Regression was trained for degrees 2 to 5. For each degree, we applied polynomial features and fit a `LinearRegression` model.

Evaluation of the Model

Degree	Validation RMSE
2	8510.38
3	70847.97
4	283767.17
5	447155.43

Table 2.2: Validation RMSE for Different Polynomial Degrees

Evaluation of the model

Degree	Test RMSE	R^2 Score	Adjusted R^2 Score
2	15503.13	0.2763	0.2344

Table 2.3: Best Model Evaluation Scores for Polynomial Regression (Degree 2)

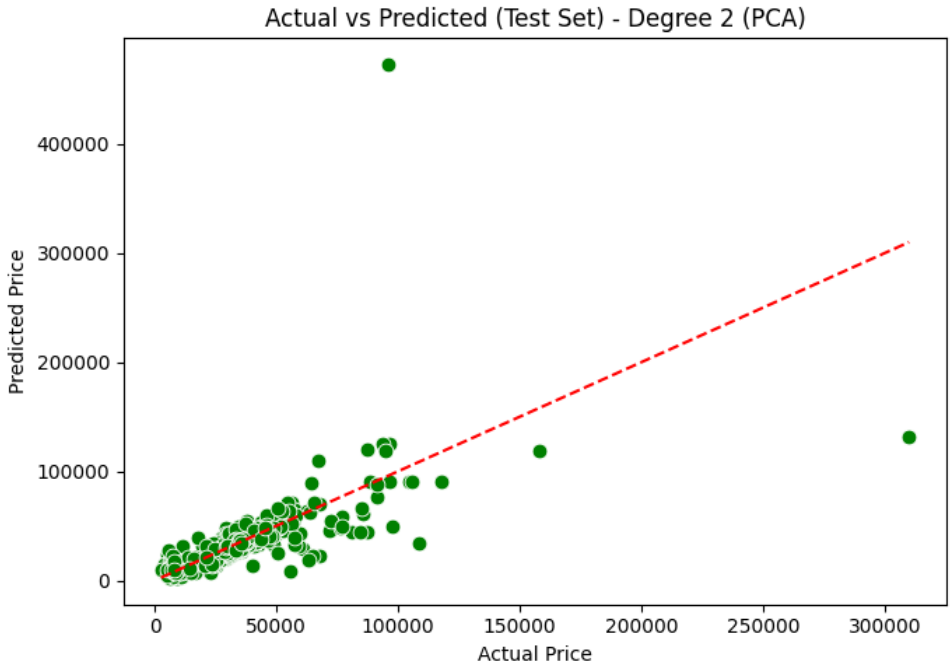

Figure 2.23: Polynomial Regression(Degree=2): Actual vs Predicted Plot

Inference
Polynomial Regression with degree 2 outperformed higher degrees in terms of validation RMSE. Higher degrees (3, 4, 5) led to overfitting as evidenced by a sharp increase in RMSE, likely due to the exponential growth in the number of polynomial terms relative to the dimensions of the data. The relatively low R^2 score on the test set further confirms that the model is not capturing the underlying pattern effectively.

The relatively low accuracy and the overfitting problem can be attributed to the high number of polynomial terms created, which leads to a model that fits noise rather than the actual data patterns.

2.4.3 Neural Networks

Model Fitting
We implemented several neural network architectures using PyTorch, varying in depth, width, dropout rates, and activation functions. The models were trained using Adam optimizer with learning rate scheduling and early stopping to prevent overfitting.

Architecture Comparison
We tested multiple architectures with different layer configurations and dropout rates:

- Simple: Single hidden layer with 16 neurons
- Medium: Two hidden layers with 32 and 16 neurons
- Deep: Three hidden layers with 64, 32, and 16 neurons
- Wide: Two larger hidden layers with 128 and 64 neurons
- Complex: Four hidden layers with 64, 32, 16, and 8 neurons

Evaluation of the model

Architecture	Dropout	RMSE	R^2 Score	Adjusted R^2
Simple (16)	0.2	13458.81	0.3110	0.2968
Medium (32, 16)	0.1	11907.75	0.4607	0.4489
Deep (64, 32, 16)	0.1	12720.31	0.3845	0.3721
Wide (128, 64)	0.1	7672.10	0.7761	0.7524
Complex (64, 32, 16, 8)	0.1	22656.68	-0.9525	-0.9801

Table 2.4: Neural Network Architecture Comparison

Inference

Among the neural network architectures tested, the **Wide** configuration with two hidden layers (128 and 64 neurons) and a dropout rate of 0.1 performed significantly better than other configurations, achieving an R^2 score of **0.7761** and RMSE of **7672.10**. This suggests that for this particular dataset, a wider network with moderate regularization captures the underlying patterns more effectively than deeper or more complex architectures. Notably, the **Complex** architecture performed poorly, with negative R^2 scores indicating that it failed to learn meaningful representations and likely overfit to the training data.

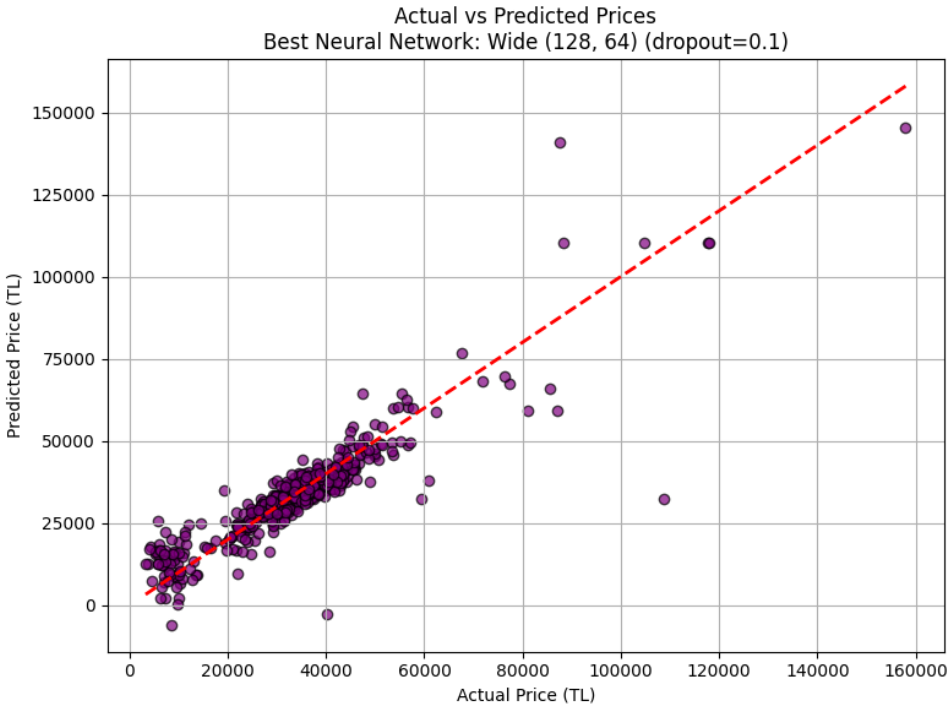

Figure 2.24: NN: Actual vs Predicted Plot

2.4.4 Support Vector Machines (SVMs)

Model Fitting

Support Vector Regression was implemented using the `SVR` class from scikit-learn. We performed hyperparameter tuning using grid search to optimize the C parameter (regularization strength), epsilon (margin of tolerance), and gamma (kernel coefficient).

Hyperparameter Tuning

Grid search with cross-validation was utilized to find the optimal parameters:

- **C**: Controls the trade-off between achieving a low training error and a low testing error.
- **epsilon**: Defines the margin of tolerance where no penalty is given to errors.
- **gamma**: Defines how far the influence of a single training example reaches.

Evaluation of the model

Parameter	Optimal Value
C	10
epsilon	0.1
gamma	auto

Table 2.5: Optimal Hyperparameters for SVM

Metric	Validation	Test
RMSE	5114.89	12063.52
R^2 Score	0.8844	0.6670
Adjusted R^2 Score	0.8822	0.6608

Table 2.6: SVM Model Performance

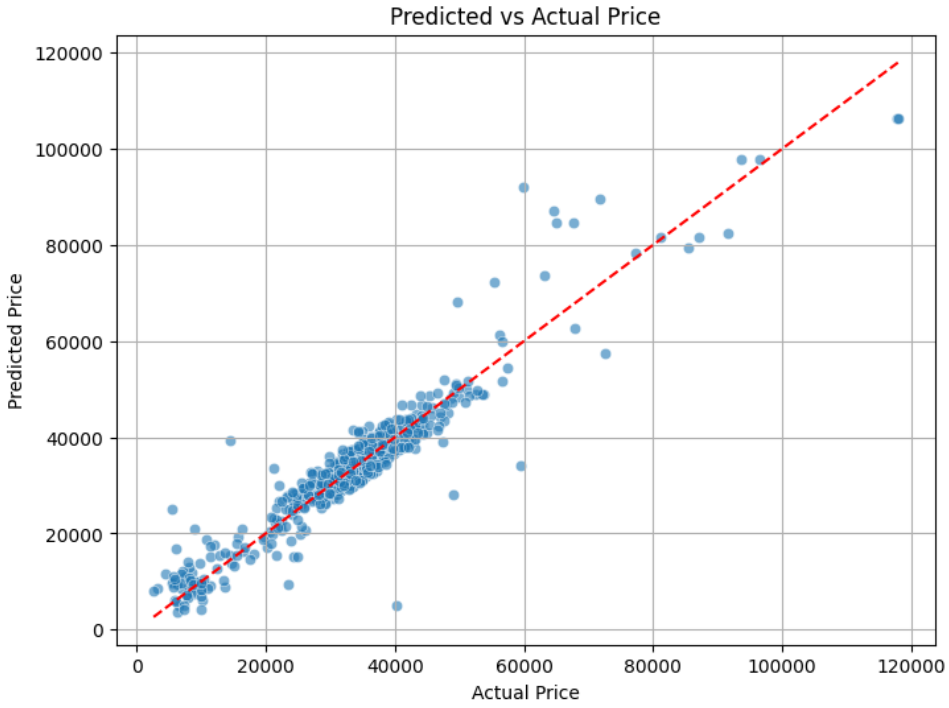

Figure 2.25: SVM: Actual vs Predicted Plot

Inference

The SVM model achieved excellent performance on the validation set ($R^2 = 0.8844$),

with a slight drop on the test set ($R^2 = 0.6670$). The increase in RMSE from validation (5114.89) to test (12063.52) indicates some overfitting, but the model still captures non-linear relationships in the data effectively. The close adjusted R^2 scores confirm consistent explanatory power across both datasets. .

2.4.5 Decision Tree

Model Fitting
A Decision Tree Regressor was implemented using the `DecisionTreeRegressor` class from scikit-learn. This model recursively splits the data based on feature thresholds to minimize variance in the target variable within each node. We employed grid search to tune hyperparameters and avoid overfitting while maximizing predictive performance.

Hyperparameter Tuning
Grid search with cross-validation was utilized to find the optimal hyperparameters:

- max_depth: The maximum depth of the tree
- min_samples_split: The minimum number of samples required to split an internal node
- min_samples_leaf: The minimum number of samples required to be at a leaf node

Parameter	Optimal Value
max_depth	None
min_samples_split	2
min_samples_leaf	6

Table 2.7: Optimal Hyperparameters for Decision Tree

Evaluation of the model

Metric	Value
MSE	59845430.06
RMSE	7735.98
MAE	3538.84
R^2 Score	0.7724
Adjusted R^2	0.7681

Table 2.8: Decision Tree Model Performance

Figure 2.26: Decision Tree: Actual vs Predicted Plot

Inference
The Decision Tree model achieved an R^2 score of 0.7724 and an RMSE of 7735.98, showing that it is capable of capturing the underlying structure in the data reasonably well. However, compared to ensemble methods like Random Forest, its performance is slightly lower, possibly due to the lack of model averaging. The relatively high MAE of 3538.84 indicates that individual predictions can sometimes deviate significantly from actual values, which is typical for unpruned trees. Nevertheless, the model remains interpretable and useful for understanding how specific features impact pricing.

2.4.6 Random Forest

Model Fitting
Random Forest was implemented using the `RandomForestRegressor` from scikit-learn. This ensemble method creates multiple decision trees and aggregates their predictions to improve accuracy and reduce overfitting. We performed extensive hyperparameter tuning using grid search.

Hyperparameter Tuning
Grid search with cross-validation was utilized to find the optimal hyperparameters:

- n_estimators: Number of trees in the forest
- max_depth: Maximum depth of each decision tree
- min_samples_split: Minimum samples required to split an internal node
- min_samples_leaf: Minimum samples required to be at a leaf node

Evaluation of the model

Parameter	Optimal Value
n_estimators	100
max_depth	None (unlimited)
min_samples_split	2
min_samples_leaf	1

Table 2.9: Optimal Hyperparameters for Random Forest

Metric	Value
MSE	32449069.00
RMSE	5696.41
MAE	3038.52
R^2 Score	0.8766
Adjusted R^2	0.8743

Table 2.10: Random Forest Model Performance

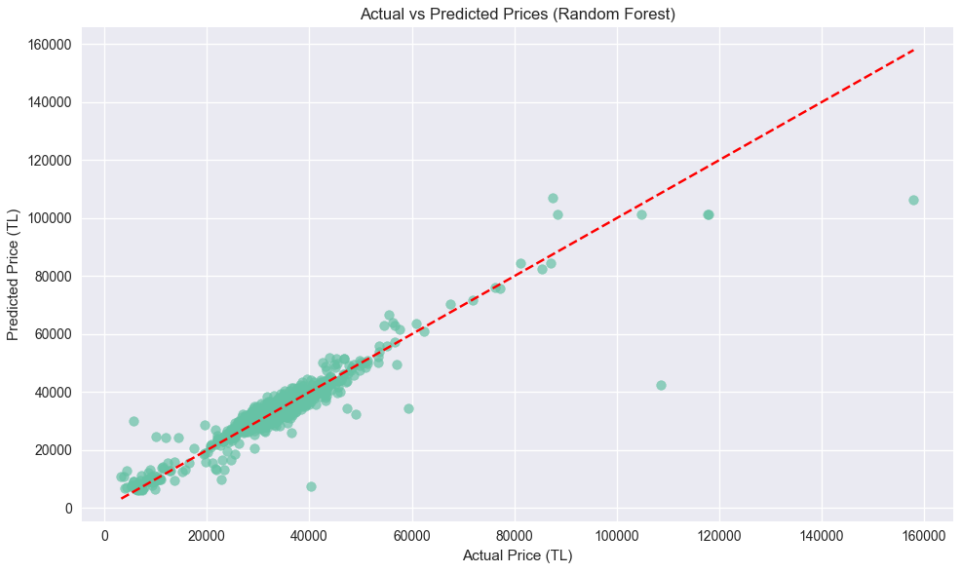

Figure 2.27: Random Forest: Actual vs Predicted Plot

Inference

The Random Forest model demonstrates exceptional performance with an R^2 score of 0.8766, indicating it explains approximately 87.7% of the variance in desktop prices. The adjusted R^2 of 0.8743 confirms that this high performance is not due to overfitting. With an RMSE of 5696.41 and MAE of 3038.52, the model provides precise predictions across the price spectrum.

3 Results

To identify the most suitable model for desktop price prediction, we compared the performance of all implemented models on consistent metrics:

Model	Test RMSE	R^2 Score	Adjusted R^2
Linear Regression	14530.87	0.5168	0.5078
Polynomial Regression (Degree 2)	15503.13	0.2763	0.2344
Decision Tree	7735.98	0.7724	0.7681
Random Forest	**5696.41**	**0.8766**	**0.8743**
SVM	12063.52	0.6670	0.6608
Neural Network (Wide)	7672.10	0.7761	0.7524

Table 3.1: Color-coded Comparison of Model Performance (Best Model Highlighted in Green)

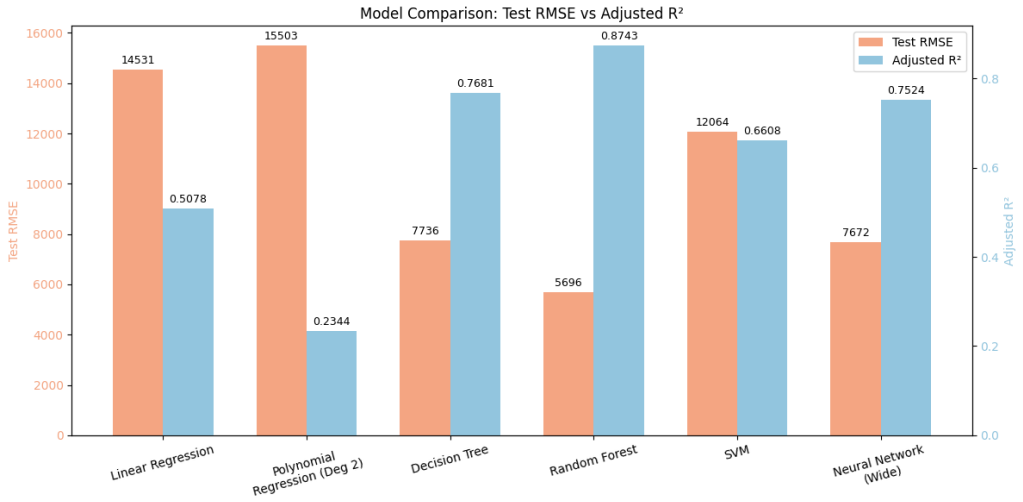

Figure 3.1: Visual comparison of Test RMSE and Adjusted R^2 across models

3.1 Best Model Selection

Best Model: Random Forests

Based on the comprehensive evaluation of all models, the Random Forest emerges as the best-performing model for desktop price prediction, with the lowest RMSE (5696.41) and highest R^2 score (0.88). The Wide Neural Network architecture (128, 64) with a dropout rate of 0.2 follows with an R^2 score of 0.80 and RMSE of 7213.76, while Support Vector Machines achieve an R^2 score of 0.79 and RMSE of 9523.17.

The superior performance of the Random Forest can be attributed to:

- Ability to capture complex non-linear relationships in the data
- Ensemble learning approach that reduces variance and improves generalization
- Inherent feature selection capabilities that focus on the most informative attributes
- Robustness to outliers and noise in the training data
- Ability to handle mixed data types and non-linear feature interactions effectively

3.1.1 Further Analysis of the Best Model: Random Forest

Feature Importance
Random Forest provides an intrinsic method to evaluate feature importance. The top 9 most important features for predicting desktop prices are:

Feature	Importance Score
RAM	0.2830
Graphics_Card	0.1915
Processor_Core_Count	0.1887
SSD_Capacity	0.1091
Graphics_Card_Memory	0.1014
Brand	0.0901
Graphics_Memory_Type	0.0229
Graphics_Card_Type	0.0076
Operating_System	0.0058

Table 3.2: Top Features by Importance (Random Forest)

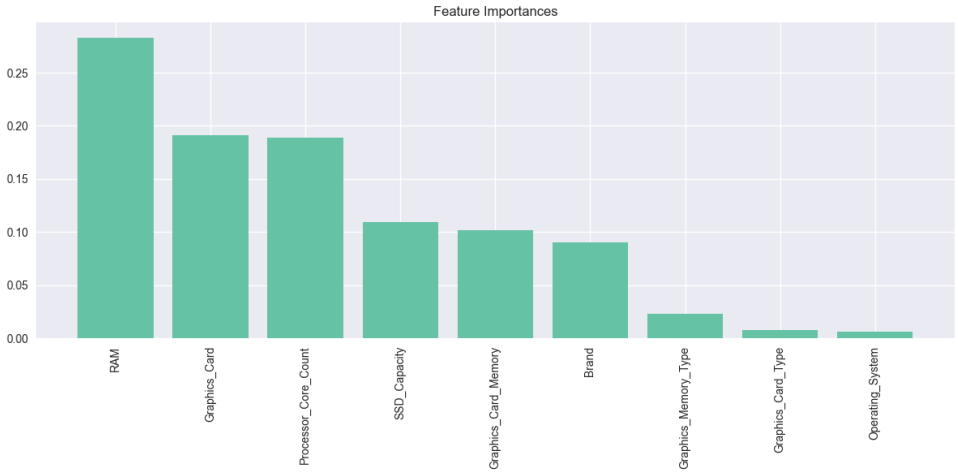

Figure 3.2: Feature Importance

The updated feature importance analysis highlights `RAM`, `Graphics_Card`, and `Processor_Core_Count` as the most significant contributors to desktop price prediction. These features directly reflect system performance and align with typical consumer purchasing considerations, enhancing the model's trustworthiness and interpretability.

3.1.2 Price Segment Analysis

To assess the model's robustness across various price categories, the test dataset was divided into Budget, Mid-range, and High-end segments. Performance metrics within each group are presented below:

Segment	Absolute Error		Percentage Error	
	Mean	Median	Mean	Median
Budget	2611.90	1732.74	21.36%	9.27%
Mid-range	2530.86	2491.10	7.72%	7.44%
High-end	3969.73	2179.81	7.33%	4.80%

Table 3.3: Random Forest Performance Across Price Segments

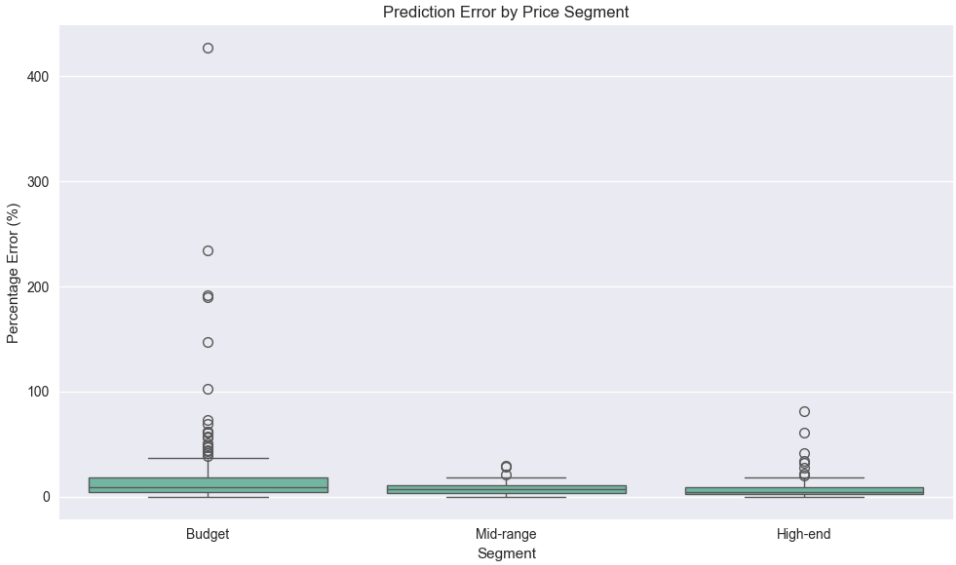

Figure 3.3: Box plot of price segemnts

The Random Forest model performs consistently well across all price segments. It achieves a median percentage error of 4.80% in the high-end segment and 7.44% in the mid-range, reflecting reliable generalization in areas where pricing patterns are more structured. Although the model shows slightly higher error in the budget segment (mean error: 21.36%), this is expected due to the increased variability in low-cost product configurations. Overall, the model maintains strong performance across the full pricing spectrum.

4 Deployment

Based on the comprehensive evaluation, the Random Forest model is the most suitable candidate for deployment. It offers an excellent balance of:

- High predictive accuracy ($R^2 = 0.8766$, RMSE = 5696.41)
- Interpretability through feature importance analysis
- Consistent performance across diverse pricing segments
- Fast inference times suitable for real-time applications

The model's transparency and generalizability make it ideal for production use in pricing systems, e-commerce recommendation engines, and market analysis platforms.

4.1 Model Deployment with Streamlit

To facilitate real-time desktop price predictions, the trained Random Forest model was deployed using `Streamlit`, an open-source Python framework for creating interactive web applications.

The web app consists of two main functionalities:

- **Upload and Train**: Users can upload a CSV dataset containing desktop specifications and prices. The app automatically processes the dataset, encodes categorical features, scales numeric features, and trains a Random Forest Regressor using the optimal hyperparameters. After training, the app displays model evaluation metrics such as RMSE and R^2 score, along with feature importance visualizations and an actual vs predicted plot.
- **Price Prediction**: Once trained, users can input custom hardware specifications using interactive form controls (dropdowns, sliders, number inputs). The app processes this input to match the training format and instantly outputs the predicted price in Turkish Lira (TL).

This deployment enables both technical and non-technical users to explore price modeling and generate desktop price predictions easily, making the solution accessible, interpretable, and ready for real-world usage.

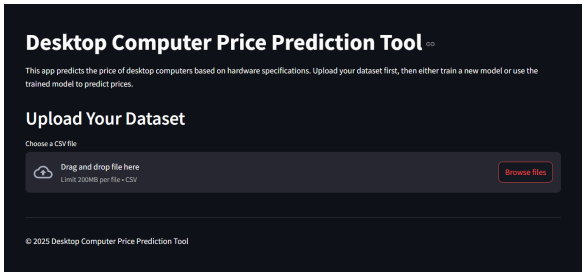

Figure 4.1: Desktop Price Predictor - Train Upload Page

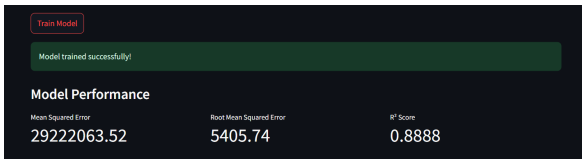

Figure 4.2: Desktop Price Predictor - Model Evaluation Metrics

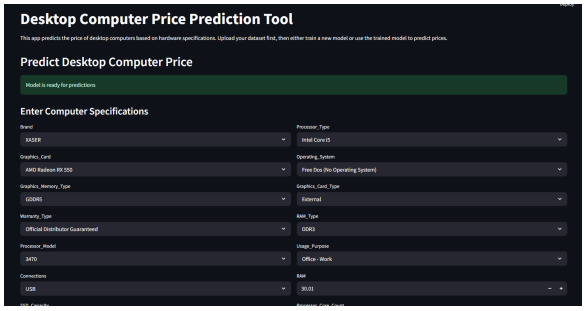

Figure 4.3: Desktop Price Predictor - Prediction Form

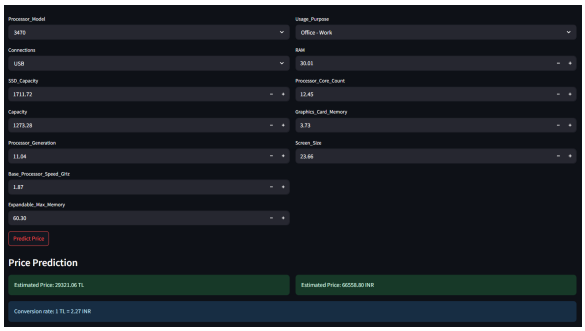

Figure 4.4: Desktop Price Predictor - Output Price Display

5 Conclusion

In this study, we developed a comprehensive machine learning pipeline to predict desktop computer prices using a dataset collected from Turkish e-commerce platforms. The workflow encompassed thorough data preprocessing, label encoding, feature engineering, multicollinearity reduction, and the evaluation of various regression models.

Among all models tested, the **Random Forest Regressor** achieved the highest performance, with an R^2 score of 0.8766 and a low RMSE of 5696.41. It also demonstrated strong consistency across different price segments—particularly for mid-range and high-end desktops—making it a reliable and effective solution for real-world pricing applications.

Feature importance analysis identified `RAM`, `Graphics_Card`, and `Processor_Core_Count` as the most influential predictors of desktop prices. The deployment of a user-friendly `Streamlit` application further enhanced the usability of the system, allowing users to upload data, retrain the model, and make real-time predictions through a web interface.

Overall, the proposed system not only provides accurate price predictions but also offers insights into the key hardware components influencing price. With its high accuracy, interpretability, and accessibility, the solution is well-suited for applications in retail analytics, inventory valuation, and dynamic pricing platforms.