Technical Analysis Report

Data Analysis on Laptop: EDA & Feature Engineering

1. Introduction

This document provides a detailed technical analysis of the Laptop Dataset, covering data cleaning, exploratory data analysis (EDA), and feature engineering. The primary goal is to preprocess the dataset, extract meaningful insights, and prepare it for machine learning models. This document includes an in-depth discussion on data preprocessing, missing value handling, feature transformations, and visualization techniques.

2. Data Loading and Initial Exploration

2.1 Importing Necessary Libraries

To begin with, the essential Python libraries required for data processing, visualization, and machine learning are imported. These libraries play a crucial role in handling large datasets efficiently and deriving meaningful insights.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor,
GradientBoostingRegressor
```

2.2 Loading the Dataset

The dataset is loaded using Pandas, which provides powerful tools to manipulate and analyze data.

```
df = pd.read_csv("laptop_data.csv")
```

2.3 Checking Dataset Shape

```
df.shape
```

This returns the total number of rows and columns in the dataset, giving a preliminary understanding of dataset size.

2.4 Basic Information about the Dataset

```
df.info()
```

This command provides information about column names, data types, and missing values, which is crucial before data preprocessing.

3. Data Cleaning

Data cleaning is a crucial step to ensure that the dataset is free from inconsistencies, missing values, and outliers.

3.1 Handling Missing Values

Missing values can distort the analysis and impact the performance of machine learning models. To identify missing values:

```
df.isnull().sum()
```

If missing values are found, they can be handled using imputation techniques:

```
df.fillna(method='ffill', inplace=True)  # Forward fill for missing values
```

Alternatively, if a column has too many missing values, it can be dropped:

```
df.dropna(axis=1, inplace=True)
```

3.2 Removing Duplicates

Duplicate rows can introduce bias into the dataset. They can be removed as follows:

```
df.drop_duplicates(inplace=True)
```

3.3 Data Type Conversion

Some columns may need conversion to appropriate data types to ensure correct processing:

```
df['Price'] = df['Price'].astype(float)
```

4. Exploratory Data Analysis (EDA)

EDA helps in understanding patterns, distributions, and relationships in the dataset. This step is essential for uncovering key insights before model building.

4.1 Distribution of Target Feature (Price)

Understanding the distribution of laptop prices helps in determining whether the data is skewed.

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Price'], bins=30, kde=True)
plt.title("Distribution of Laptop Prices")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```

4.2 Analyzing Laptop Brands

Laptop brands can significantly impact the price and specifications of the laptops.

```
plt.figure(figsize=(12, 6))
sns.countplot(y=df['Company'], order=df['Company'].value_counts().index)
plt.title("Laptop Brands Distribution")
plt.xlabel("Count")
plt.ylabel("Brand")
plt.show()
```

4.3 Correlation Heatmap

A heatmap helps in identifying relationships between numerical features.

```
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Feature Correlation Heatmap")
plt.show()
```

5. Feature Engineering

Feature engineering helps enhance the dataset by creating new meaningful features, improving model performance.

5.1 Extracting Features from Column Names

If necessary, extracting model-specific details from names:

```
df['Processor'] = df['Name'].apply(lambda x: x.split()[0])
```

5.2 Encoding Categorical Features

Categorical variables need to be converted into numerical representations for machine learning models:

```
df = pd.get_dummies(df, columns=['Company', 'Processor'], drop_first=True)
```

5.3 Standardizing Numeric Features

Standardization ensures that all numerical features have the same scale, improving model performance:

```
scaler = StandardScaler()
df[['Price', 'RAM', 'Weight']] = scaler.fit_transform(df[['Price', 'RAM',
'Weight']])
```

5.4 Feature Selection

Feature selection helps in reducing dimensionality and improving model accuracy by keeping only relevant variables.

```
from sklearn.feature_selection import SelectKBest, f_regression
selector = SelectKBest(score_func=f_regression, k=10)
selected_features = selector.fit_transform(df.drop(columns=['Price']),
df['Price'])
```

6. Model Training

After cleaning and preprocessing, the dataset is ready for machine learning models.

6.1 Splitting the Dataset

```
X = df.drop(columns=['Price'])
y = df['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

6.2 Training Regression Models

A variety of regression models can be trained and evaluated:

```
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(),
    'Random Forest': RandomForestRegressor(),
    'Gradient Boosting': GradientBoostingRegressor()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    print(f'{name} Score: {model.score(X_test, y_test)}')
```

7. Conclusion

This document covered dataset loading, data cleaning, EDA, feature engineering, and model training. The dataset has been processed and prepared for predictive analysis, ensuring high-quality insights for price prediction. Future improvements could involve hyperparameter tuning and advanced feature engineering to further optimize model performance.