**Problem Description**

In this project, the main objective is to build a machine learning model that can predict the price of a laptop in Euros based on its technical specifications and brand. Laptop prices can widely depend on features such as the processor,RAM, GPU, and operating system. Understanding how these features affect the price can be useful for both consumers and retailers. I chose this problem because it reflects a real-world application where predictive modeling can help users make more informed purchasing decisions or guide businesses in pricing strategies.

**Dataset**

The dataset used in this project is from Kaggle and contains information about various laptop models, including their brands, specifications, and prices. There are 15 columns and no missing values in the dataset, which made the data cleaning process relatively straightforward. The target variable is Price (Euro).

For modeling, I selected a subset of features believed to be the most impactful on laptop pricing:

* Company
* Inches
* CPU\_Company
* CPU\_Type
* RAM (GB)
* GPU\_Company
* OpSys
* Weight (kg)

These features were chosen based on their relevance to performance and branding, which are common factors influencing price.

**Modeling Approach**

I use Polynomial Regression as the main modeling approach, implemented through Pipeline using Scikit-learn. The reason for choosing polynomial regression over simple linear regression is that it can model non-linear relationships, which are common in real-world pricing patterns.

The Pipeline included:

* StandardScaler: To normalize numerical features such as screen size, RAM, and weight.
* PolynomialFeatures: To capture non-linear relationships in numerical features.
* OneHotEncoder: To convert categorical variables (like Company and OpSys ) into a suitable numeric format.

I used GridSearchCV to tune the degree of the polynomial features ( from 2 to 5 ) and performed cross-validation ( cv=3 ) to avoid overfitting.

# Define columns

numerical = ['Inches', 'RAM (GB)', 'Weight (kg)']

categorical = ['Company', 'CPU\_Company', 'CPU\_Type', 'GPU\_Company', 'OpSys']

# ColumnTransformer: apply PolynomialFeatures ONLY to numerical

preprocessor = ColumnTransformer(transformers=[

('num', Pipeline(steps=[

('scale', StandardScaler()),

('poly', PolynomialFeatures(include\_bias=False))

]), numerical),

('cat', OneHotEncoder(handle\_unknown='ignore'), categorical)

])

# Final pipeline

pipeline = Pipeline(steps=[

('preprocessor', preprocessor),

('regressor', LinearRegression())

])

# GridSearchCV parameter: match the full nested path

param\_grid = {

'preprocessor\_\_num\_\_poly\_\_degree': [2, 3, 4, 5]

}

# GridSearchCV

poly\_grid = GridSearchCV(pipeline,

param\_grid=param\_grid,

cv=3,

scoring='neg\_mean\_squared\_error',

verbose=1,

error\_score='raise')

Here is my code snippet, I define two columns numerical and categorical to separate the values.

**Evaluation**

To evaluate the model, I used the following metrics: Mean Absolute Error ( MAE ) , Mean Squared Error ( MSE) , R2 Score.

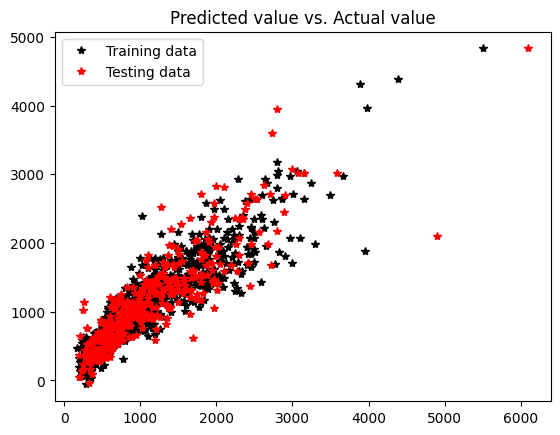
On the training set, the model achieved:

* MAE : 204.0
* MSE : 84864
* R2 Score : 0.82

On the test set:

* MAE : 245
* MSE : 125208
* R2 Score : 0.76

These results indicate that the model fits the training data well and generalizes reasonably to unseen data, although there is some room for improvement.



**Reflection**

This project was a valuable learning experience. One challenge I faced was ensuring the pipeline and parameter tuning were correctly configured, especially when using nested parameters in GridSearchCV. Another issue was that including too many features or too high a polynomial degree significantly slowed down computation and even caused model failures.

From this project, I learned how to properly use pipelines, apply feature scaling and encoding, and evaluate model performance using different metrics. I also gained experience in debugging model configuration errors and understanding the balance between underfitting and overfitting.

If I were to do this project again, I would consider:

* Engineering additional features (like splitting Memory into SSD and HDD).
* Using more detailed data on screen resolution, GPU type, and storage type.

Overall, this project deepened my understanding of regression modeling, preprocessing pipelines, and real-world machine learning workflows.