Machine Learning Project

Car Price Prediction Project Documentation

This document outlines the process, steps, and results of the Car Price Prediction machine learning project, which aims to predict the selling price of a used car based on various features.

1. Introduction and Goal

The primary goal of this project is to build and evaluate machine learning regression models capable of accurately predicting the Selling_Price of a car using a dataset containing car specifications and historical sales information.

2. Data Preprocessing and Cleaning

The initial dataset underwent several cleaning and preparation steps to ensure data quality and suitability for modeling.

A. Missing Value Handling

- Missing values were initially handled by filling all with 0 (This might have been a temporary step).
- Subsequently, a more standard approach was used:
    - **Numerical Columns:** Missing values were filled using the mean of the respective column.
    - **Categorical Columns:** Missing values were filled using the mode (most frequent value) of the respective column.
- **Verification:** After filling, df.isnull().sum() confirmed no remaining missing values.

B. Feature Cleaning (Unit Removal)

- **Mileage:** The units ('km/l' and 'km/kg') were removed, and the column was converted to a float type.
- **Engine:** The unit ('CC') was removed, and the column was converted to a float type.

C. Duplicate Removal

- Duplicates were identified and removed.
- The shape of the dataset changed from the initial number of rows to the final number of unique rows: {Initial Rows} →{Final Rows}.

D. Outlier Detection

- Boxplots were generated for key numerical features (Car_Age, KM_Driven, Mileage, Engine, Selling_Price) to visually inspect the distribution and presence of outliers.

3. Feature Engineering

New features were created to enhance the predictive power of the models.

- **Price_per_KM:** Calculated as {Selling_Price}/{KM_Driven + 1} . This feature represents the value of the car per kilometer driven.
- **Age_Category:** A categorical feature derived from Car_Age:
    - 'New' (Age <= 3)
    - 'Mid' (Age <= 8)
    - 'Old' (Age >= 8)

4. Encoding and Scaling

Categorical features were converted into a numerical format, and numerical features were standardized.

A. Encoding:

- Label Encoding: Applied to ordinal categorical features with a logical order:
    - Owner_Type
    - Age_Category
- One-Hot Encoding: Applied to nominal categorical features with no inherent order, using pd.get_dummies with drop_first=True to avoid multicollinearity (the Dummy Variable Trap):
    - Fuel_Type
    - Transmission

B. Feature Scaling

- StandardScaler (Z-score normalization) was used to scale the following numerical columns:
    - Car_Age, KM_Driven, Mileage, Engine, Selling_Price, Price_per_KM.
- *Note: For the modeling section (7 & 8), the unscaled data (df) was used for X and Y which means the models were trained on features with original scales, except for the encoded categorical features.*

5. Model Building and Evaluation

Two regression models were trained and evaluated on the dataset. The target variable is Selling_Price.

A. Data Splitting

The data was split into training and testing sets:

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

B. Model Performance (Initial)

Model	Metric	Value	Interpretation
Linear Regression	R^2	R^2_{LR}	{Value of R^2_{LR}} (Closer to 1 is better)
	MSE	{MSE}_{LR}	{Value of {MSE}_{LR}} (Lower is better)
	MAE	{MAE}_{LR}	{Value of {MAE}_{LR}} (Lower is better)
Random Forest Regressor	R^2	R^2_{RF}	{Value of R^2_{RF}} (Closer to 1 is better)
	MSE	{MSE}_{RF}	{Value of {MSE}_{RF}} (Lower is better)
	MAE	{MAE}_{RF}	{Value of {MAE}_{RF}} (Lower is better)

The initial results suggest the Random Forest Regressor is a significantly better-performing model, as evidenced by its higher R^2 score.

6. Hyperparameter Tuning and Cross-Validation

The best-performing model, Random Forest Regressor, was optimized using GridSearchCV.

A. Linear Regression Cross-Validation

- Method: 5-fold cross-validation (cv=5) on the training data.
- Metric: R^2 score.

- Results:
    - CV R^2 scores: {CV R2 scores}
    - Mean CV R^2: {Mean CV R2}
    - Test R^2: {LR Test R2}

B. Random Forest Regressor Optimization (GridSearchCV)

- Estimator: RandomForestRegressor(random_state=42)
- Cross-Validation: 3-fold (cv=3)
- Scoring: R^2
- Parameter Grid (param_grid):

Python

```
param_grid = {
  'n_estimators': [100, 200],
  'max_depth': [None, 10, 20],
  'min_samples_split': [2, 5],
  'min_samples_leaf': [1, 2]
}
```

- Best Parameters: {Best RF Parameters}

C. Final Optimized Model Performance

The optimized Random Forest Regressor was evaluated on the test set.

Model	Metric	Value
Optimized Random Forest	Test R^2	{RF Test R2}
	Test MSE	{RF Test MSE}

7. Conclusion

The Random Forest Regressor, particularly after hyperparameter tuning with GridSearchCV, proved to be the most effective model for this car price prediction task, achieving a high R^2 value on the test set. This indicates that the model explains a large proportion of the variance in the car selling prices. The engineered features

(Price_per_KM and Age_Category) and comprehensive data preprocessing steps were critical to the success of the model.