AUTOMOBILE PRICE PREDICTION USING ML MODELS

ANNUAL REPORT

Presented By

UMAR LUQMAN

Report on Machine Learning Workflow for Automobile Price Prediction

1. Introduction

- This report documents the step-by-step process and rationale for creating a machine learning pipeline to predict automobile prices.
- It includes exploratory data analysis (EDA), data preprocessing, model training, and performance evaluation using metrics like Mean Squared Error (MSE) and R^2-score.

2. Data Loading and Overview

Steps:

- Used pandas to load the dataset.
- Initial exploration with .head(), .tail(), .info(), .describe(), and .isnull().sum() to identify data characteristics, missing values, and data types.

3. Exploratory Data Analysis (EDA)

Steps:

- Plotted histograms for numerical features to understand distributions.
- Examined correlations between features and the target variable (price) using a heatmap.
- Identified skewness in the price feature and examined outliers.
- Used count plots for categorical features like make to visualize their distributions.

4. Data Cleaning

Steps:

- Removed duplicates, reducing the dataset size from **30,330** to **25,874** records.
- Identified numerical and categorical columns.
- Replaced incorrect values (e.g., ?) with NaN and imputed missing values
- Numerical Features: Imputed with **mean** or **median** based on distribution and sensitivity to outliers.
- Categorical Features: Imputed missing values with **mode** or added an **'Unknown'** category for infrequent cases.
- Converted data types for features incorrectly stored as objects to float.

5. Feature Engineering

Steps:

- Standardized numerical features to ensure uniform scaling.
- Applied one-hot encoding for categorical variables to handle non-numeric data.
- Rationale:
- Scaling ensures features are comparable for distance-based models like KNN.
- One-hot encoding prevents ordinal relationships in categorical data.

6. Data Splitting

Steps:

- Reserved 20% of the data as test data using train_test_split with a random state for reproducibility.

7. Model Training

Models Used:

- Linear Regression: Baseline for comparison.
- Decision Tree Regressor: Captures non-linear relationships.
- KNN Regressor: Simple, instance-based learning.
- Random Forest Regressor: Reduces overfitting by averaging results of multiple trees.
- Gradient Boosting Regressor: Combines weak learners for better performance.
- Pipeline: Created a unified pipeline for preprocessing and model training, ensuring modularity and reproducibility.

8. Evaluation

Metrics:

- Mean Squared Error (MSE): Measures prediction error magnitude.
- R^2-Score: Indicates goodness-of-fit.

Steps:

- Predicted prices on the test set using each model.
- Calculated and compared MSE and R^2-scores across models.

9. Results and Discussion

Findings:

- Models with higher complexity (e.g., Gradient Boosting) showed better R^2-scores but required more computational resources.
- Simpler models like Linear Regression were faster but less accurate.

Key Insights:

- Feature engineering and preprocessing significantly impact model performance.
- Regularization and hyperparameter tuning can further improve results.

10. Conclusion

- This project demonstrated the complete machine learning workflow, from data preparation to model evaluation, highlighting the importance of each step.
- The use of pipelines ensured a scalable and efficient process.