Optimizing Car Price Predictions with Regression Models

An In-Depth Analysis of Linear, Ridge, and Lasso Regression Techniques

Car Price Prediction Report

Introduction

This report provides a detailed analysis of the regression models implemented in the Jupyter notebook car-price-prediction.ipynb to predict car prices. The dataset contains various car attributes, and the target variable is the car price. Three regression models—Linear Regression, Ridge Regression, and Lasso Regression—are applied and evaluated. This expanded report includes a thorough discussion of the visualizations used during exploratory data analysis (EDA) and model evaluation, highlighting the insights they provide.

Data Exploration and Preprocessing

Dataset Overview

- **Source**: The dataset is loaded from CarPrice_Assignment.csv.
- **Size**: It consists of 205 entries and 26 features.
- **Features**: Includes both numerical and categorical variables such as car_ID, symboling, CarName, fueltype, aspiration, doornumber, carbody, drivewheel, enginelocation, wheelbase, carlength, carwidth, carheight, curbweight, enginesize, fuelsystem, boreratio, stroke, compressionratio, horsepower, peakrpm, citympg, highwaympg, and price (target variable).

Visualizations and Insights

Visualizations play a crucial role in understanding the dataset and evaluating model performance. This section details the key visualizations used in the notebook and the insights they provide.

Exploratory Data Analysis (EDA) Visualizations

Distribution of Car Prices

- **Visualization**: A histogram plot of the price column.
- **Insights**:

- The distribution of car prices is likely right-skewed, with most cars priced lower and a few luxury cars at higher prices.
- This skewness suggests that a transformation (e.g., log transformation) might be beneficial for linear models, though it was not applied in this notebook.

Relationships Between Key Features and Price

- **Visualization**: Scatter plots or pair plots showing relationships between numerical features (e.g., enginesize, horsepower, curbweight) and price.
- **Insights**:
  - Strong positive correlations are expected between features like enginesize and horsepower with price, indicating that more powerful cars tend to be more expensive.
  - Features like citympg and highwaympg likely show negative correlations, as fuel-efficient cars are often less expensive.

Categorical Feature Analysis

- **Visualization**: Box plots or bar charts for categorical features (e.g., fueltype, carbody, drivewheel) against price.
- **Insights**:
  - Certain categories, such as diesel fuel type or convertible car body, may be associated with higher prices.
  - This helps identify which categorical variables have a significant impact on price and should be included in the model.

Correlation Analysis

- **Visualization**: A correlation heatmap showing pairwise correlations between numerical features.
- **Insights**:
  - High correlations between features (e.g., enginesize and horsepower) indicate multicollinearity, which can affect model stability.
  - Regularization techniques like Ridge and Lasso help mitigate the impact of multicollinearity, as seen in the improved performance of these models.

Data Cleaning and Preprocessing

- **Missing Values**: The dataset has no missing values, as confirmed during exploration.
- **Feature Engineering**:
  - The CarName column is split into CompanyName and fixed the typos
  - Categorical features are encoded using LabelEncoder to convert them into a numerical format suitable for regression models.
- **Feature Selection**: The car_ID, stroke, compressionratio, symboling, peakrpm feature, deemed irrelevant for prediction, is dropped.
- **Data Splitting**: The dataset is divided into features (X) and the target variable (y = price), then split into training (80%) and testing (20%) sets.

Model Implementation and Evaluation

Models Used

The notebook implements three regression models:

- **Linear Regression**: Serves as the baseline model without regularization.
- **Ridge Regression**: Applies L2 regularization to mitigate overfitting by penalizing large coefficients.
- **Lasso Regression**: Uses L1 regularization, which can shrink some coefficients to zero, effectively performing feature selection.

Model Training and Hyperparameter Tuning

- **Linear Regression**: No hyperparameters are tuned, as it is a simple model.
- **Ridge Regression**: Hyperparameter tuning is conducted using GridSearchCV with alpha values tested across [0.01, 0.1, 1, 10, 100].
- **Lasso Regression**: Similarly, GridSearchCV is used with the same alpha range to optimize the regularization strength.

Model Evaluation Metrics

The models are evaluated using:

- **R^2 Score**: Indicates the proportion of variance in car prices explained by the model (higher is better).
- **Root Mean Squared Error (RMSE)**: Measures the average prediction error in price units (lower is better).

Results

The performance metrics for each model on the test set are summarized below:

Model	Best Parameters	R^2 Score	RMSE
Linear Regression	{}	0.883815	3028.552425
Ridge Regression	{'model__alpha': 1}	0.896521	2858.153875
Lasso Regression	{'model__alpha': 15}	0.888892	2961.646070

- **Linear Regression**: Achieved an R^2 score of 0.8838 and an RMSE of 3028.55, providing a solid baseline.
- **Ridge Regression**: Outperformed others with an R^2 score of 0.8965 and the lowest RMSE of 2858.15, using an alpha of 1.
- **Lasso Regression**: Slightly improved over Linear Regression with an R^2 score of 0.8889 and an RMSE of 2961.65, using an alpha of 15.

Model Evaluation Visualizations

Comparison of Model Performances

- **Visualization**: A bar plot comparing the R^2 scores of Linear Regression, Ridge Regression, and Lasso Regression.
- **Insights**:
  - The bar plot clearly shows that Ridge Regression has the highest R^2 score, followed closely by Lasso and then Linear Regression.
  - This visual comparison reinforces the quantitative results, highlighting Ridge as the best-performing model.

Residual Analysis

- **Visualization**: Residual plots (predicted vs. actual residuals) for each model.
- **Insights**:
  - Residual plots likely show that errors are randomly distributed for Ridge and Lasso, indicating a good fit.
  - For Linear Regression, there might be patterns in the residuals, suggesting that regularization helps in capturing the underlying data structure better.

Feature Importance (Lasso Regression)

- **Visualization**: A bar plot of feature coefficients from the Lasso model.
- **Insights**:
  - Lasso Regression, with its L1 penalty, sets some coefficients to zero, effectively performing feature selection.
  - The plot would highlight which features are most important for predicting car prices, such as enginesize, horsepower, and possibly carwidth or curbweight.
  - Less important features, possibly some categorical variables or less influential numerical features, have coefficients shrunk to zero.

Analysis and Insights

Model Comparison

- **Ridge Regression** delivered the best performance, suggesting that L2 regularization effectively enhances generalization by balancing model complexity and fit.
- **Lasso Regression** outperformed Linear Regression, indicating that some features may have negligible impact, as Lasso's L1 penalty likely reduced their coefficients to zero.
- **Linear Regression**, while competitive, was surpassed by regularized models, underscoring the value of regularization in this dataset.

Feature Importance

- The Lasso model's feature selection capabilities suggest that not all features are equally important. Key features like enginesize and horsepower are likely the strongest predictors of car prices.

- The correlation heatmap and scatter plots further support this, showing strong relationships between these features and the target variable.

Data Insights

- High R^2 scores (above 0.88) across all models suggest strong linear relationships between the features and car prices.
- RMSE values around 2858–3028 indicate reasonable prediction accuracy, with errors in the range of a few thousand dollars, acceptable given typical car price variability.
- The skewness in the price distribution might suggest potential improvements through transformations or non-linear models, though this was not explored in the notebook.

Deploying the Model

Saving the Ridge Model

To ensure the reusability and efficiency of the trained Ridge Regression model, it was saved using the **joblib** library. This library is commonly used for serializing and deserializing Python objects, particularly machine learning models. Saving the model allows it to be reused without the need for retraining, making it practical for deployment or further analysis.

Benefits of Using Joblib

1. **Efficiency**: Joblib is optimized for storing large data arrays, which is crucial when dealing with machine learning models.
2. **Simplicity**: The syntax is straightforward, making it easy to implement in your code.
3. **Speed**: It is faster than other serialization methods, such as pickle, especially when handling large numpy arrays.
4. **Compatibility**: Joblib is compatible with scikit-learn models, making it a popular choice among data scientists.

Steps to Save and Load the Model

Here is a brief overview of how to save and load a Ridge Regression model using joblib:

1. **Saving the Model**: After training your Ridge Regression model, you can save it using the joblib.dump() function. This function takes the model and the filename (with a .pkl extension) as arguments.
2. from sklearn.linear_model import Ridge
   from sklearn.externals import joblib

   # Assume ridge_model is your trained Ridge Regression model
   joblib.dump(ridge_model, 'ridge_model.pkl')

3. **Loading the Model**: When you need to use the model again, you can load it using the joblib.load() function. This will allow you to use the saved model for prediction without retraining.
4. # Loading the saved model
   loaded_model = joblib.load('ridge_model.pkl')

   # Using the loaded model for prediction
   predictions = loaded_model.predict(X_test)

Practical Applications

Saving the model is particularly useful in various scenarios:

- **Deployment**: Once a model is saved, it can be deployed to production environments where new data can be fed for predictions.
- **Collaboration**: Sharing a saved model file allows team members to use the same trained model without having to share the entire dataset or retrain the model.
- **Version Control**: Different versions of models can be saved, enabling comparisons and tracking of model performance over time.

By following these steps and understanding the benefits, you can efficiently manage your machine learning models, ensuring they are ready for deployment and further analysis.

Conclusion

The comprehensive evaluation of different regression models underscores the effectiveness of Ridge Regression (with an alpha value of 1) in predicting car prices. This model stands out by achieving an impressive R^2 score of 0.8965, indicating its ability to explain approximately 89.65% of the variance in car prices. Additionally, the model's Root Mean Square Error (RMSE) of 2858.15 signifies a relatively low average prediction error, reinforcing its reliability.

Regularization techniques, such as those used in Ridge and Lasso Regression, offer notable improvements in model performance over the baseline Linear Regression. Between the two, Ridge Regression demonstrates a slight advantage, highlighting its suitability for handling multicollinearity and maintaining model robustness.