Predicting car prices

Problem Description

The objective of this project is to predict the price of used cars based on various features such as car's model, year of manufacturing, kilometers driven, fuel type, transmission type, etc. The dataset for this project is obtained from Kaggle and it contains details of used cars sold in India.

Dataset Description

The dataset contains 12 columns and 6019 rows. The columns in the dataset are:

- Name: Name of the car
- Location: Location where the car is being sold or is available for purchase
- Year: Manufacturing year of the car
- Kilometers_Driven: The total kilometers driven by the car
- Fuel_Type: The type of fuel used by the car
- Transmission: The type of transmission used by the car
- Owner_Type: Whether the car has been previously owned or not
- Mileage: The standard mileage offered by the car company in kmpl or km/kg
- Engine: The displacement volume of the engine in cc

- Power: The maximum power of the engine in bhp
- Seats: The number of seats in the car
- Price: The price of the used car in lakhs

Objectives

- Perform exploratory data analysis on the dataset to gain insights about the features and their relationship with the target variable.
- Preprocess the data by handling missing values, encoding categorical variables, and scaling the numerical variables.
- Build a regression model to predict the price of the used cars.
- Evaluate the model using appropriate evaluation metrics.
- Fine-tune the model using hyperparameter tuning techniques.
- Provide recommendations for feature engineering to improve the performance of the model.

Deliverables

- A Jupyter notebook containing the code, visualizations, and explanations.
- A trained regression model for predicting the price of the used cars.
- A report containing the findings and recommendations.

Suggested Framework to Solve this problem :

1.Problem Definition: Define the problem and the goal of the project. In this case, it is to predict the price of used cars.

2. Data Collection: Collect the dataset from Kaggle and load it into a pandas dataframe.

3. Data Cleaning and Preprocessing: Clean the data and preprocess it to make it ready for model training. This step includes dealing with missing values, handling categorical variables, encoding variables, etc.

4. Exploratory Data Analysis (EDA): Perform EDA on the dataset to gain insights into the data and the relationships between the variables. This step includes visualizations, statistical summaries, and correlation analysis.

5. Feature Selection and Engineering: Select the most important features and engineer new features that can improve model performance. This step includes techniques like correlation analysis, feature importance, and domain knowledge.

6. Model Training and Tuning: Train various machine learning models and tune their hyperparameters to achieve the best performance. This step includes splitting the data into train and test sets, training different models, and evaluating their performance.

7. Model Evaluation and Selection: Evaluate the performance of the trained models and select the best model based on their performance metrics.

8. Deployment: Deploy the final model to make predictions on new data.

9. Monitoring and Maintenance: Monitor the deployed model and maintain it to ensure it remains accurate and up-to-date.

I hope this helps! Let me know if you have any questions.

Code Explanation :

Here is the simple explanation for the code you can find at code.py file.

Section 1: Importing Libraries and Data

- In this section, the necessary libraries are imported, including pandas, numpy, seaborn, and matplotlib.

- The data is loaded into a pandas dataframe using the read_csv function from pandas.

Section 2: Data Cleaning and Preprocessing

- In this section, the data is cleaned and preprocessed.

- First, irrelevant columns are dropped using the drop function.

- Then, missing values are handled using the fillna function.

- Next, categorical variables are encoded using the get_dummies function.

- Finally, the data is split into training and testing sets using the train_test_split function from sklearn.

Section 3: Feature Scaling

- In this section, feature scaling is performed on the data.

- The StandardScaler function from sklearn is used to scale the numerical columns.

Section 4: Model Training and Tuning

- In this section, various regression models are trained and tuned.
- First, a linear regression model is trained using the LinearRegression function from sklearn.
- Next, a decision tree regressor model is trained using the DecisionTreeRegressor function from sklearn.
- Then, a random forest regressor model is trained using the RandomForestRegressor function from sklearn.
- Hyperparameter tuning is performed on the random forest regressor model using the GridSearchCV function from sklearn.
- Finally, the best model is selected and its performance is evaluated using the mean absolute error and mean squared error.

Section 5: Generating Summary

- In this section, a summary of the best model's performance is generated.

Section 6: Model Refinement

- In this section, potential areas for further improvement are identified and suggested.

Overall, the code follows a well-defined framework for machine learning projects, which involves importing the necessary libraries and data, cleaning and preprocessing the data, performing feature scaling, training and tuning various models, selecting the best model, evaluating its performance, generating a summary, and identifying areas for refinement.

The code makes use of several functions from the sklearn library, including LinearRegression, DecisionTreeRegressor, RandomForestRegressor, GridSearchCV, and train_test_split. These functions are used to train and tune various regression models, evaluate their performance, and perform feature scaling.

Overall, the code is well-commented and easy to read, making it easy for others to understand and build upon.

Future Work :

The following are some future work that can be done to improve the model and its predictions.

1.Data Collection and Cleaning:

Collecting more data and cleaning the existing data can be the first step to improving the model's performance. Data can be collected from other sources and merged with the existing data. Data cleaning can be done by removing outliers, handling missing values, and correcting erroneous data.

2.Feature Engineering:

Feature engineering involves creating new features or transforming the existing ones to improve the model's performance. In this project, some possible feature engineering methods include creating new features like age of the car, mileage per year, and combining the features like fuel type and transmission type.

3.Trying Different Models:

Different models can be tried to improve the model's performance. Some possible models that can be tried include Gradient Boosting, XGBoost, and Neural Networks. Hyperparameter tuning can also be done for these models.

4.Ensembling:

Ensembling involves combining multiple models to improve the model's performance. In this project, ensembling can be done by combining the best performing models like Random Forest, Gradient Boosting, and XGBoost.

5.Deployment:

The final step is to deploy the model in a production environment. This involves creating an API that can be used by the end-users to predict the prices of used cars. The API can be deployed on a cloud platform like AWS or Google Cloud. A web interface can also be created for the end-users to interact with the API.

Step-by-Step Guide:

1. Collect more data from different sources and merge it with the existing data.

2. Clean the data by removing outliers, handling missing values, and correcting erroneous data.

3. Perform feature engineering by creating new features or transforming the existing ones.

4. Try different models like Gradient Boosting, XGBoost, and Neural Networks.

5. Perform hyperparameter tuning for the best performing models.

6. Ensemble the best performing models.

7. Deploy the model on a cloud platform like AWS or Google Cloud.

8. Create an API that can be used by the end-users to predict the prices of used cars.

9. Create a web interface for the end-users to interact with the API.

Exercise Questions :

1. **What is the importance of data cleaning in the process of building a machine learning model for car price prediction?** Answer: Data cleaning is an essential step in the machine learning process for car price prediction. It involves removing or correcting data that is incorrect, incomplete, irrelevant or duplicated. By cleaning the data, we ensure that the data used in the model is accurate and complete. The quality of the model's predictions depends heavily on the quality of the data used to train the model. If the data is not clean, the model may generate inaccurate or unreliable results.

2. **What is the difference between feature selection and feature engineering?** Answer: Feature selection involves choosing a subset of the most important features from a dataset, while feature engineering involves creating new features from existing ones. Feature selection is used to reduce the dimensionality of the data and remove irrelevant or redundant features, while feature engineering is used to enhance the model's predictive power. Feature selection can be done manually or automatically using algorithms, while feature engineering requires creativity and domain knowledge.

3. **What is the difference between a regression model and a classification model?**
   Answer: A regression model is used to predict a continuous numerical output, while a classification model is used to predict a categorical output. In the case of car price prediction, a regression model would be used to predict the price of a car based on its features, while a classification model could be used to predict whether a car is expensive or cheap based on its features.

4. **What is the purpose of cross-validation in machine learning?**
   Answer: Cross-validation is used to evaluate the performance of a machine learning model. It involves splitting the dataset into multiple subsets and using each subset as a test set while using the rest of the data as a training set. By doing this, we can assess the model's performance on different subsets of the data and ensure that it is not overfitting to any particular subset. Cross-validation helps to improve the model's generalization performance and reduce the risk of overfitting.

5. **What is the difference between hyperparameters and parameters in a machine learning model?**

Answer: Parameters are the values that are learned by the model during training, while hyperparameters are set by the user before training begins. Parameters are specific to the model and cannot be changed by the user, while hyperparameters can be adjusted to improve the model's performance. Examples of hyperparameters include the learning rate, regularization strength, and the number of hidden layers in a neural network.

Concept Explanation :

So, the algorithm we used in this project is called Random Forest Regression. It's like a forest of decision trees, where each tree tries to predict the output (in our case, the price of a used car) based on the input features (like mileage, year, etc.).

But here's the fun part – each tree in this forest is not like a normal decision tree you might be used to. It's like a crazy decision tree that's been drinking too much coffee! Each tree only looks at a random subset of the input features to make its prediction. So, one tree might only consider the year and make of the car, while another might only consider the mileage and number of doors.

Then, to make the final prediction, we take the average of all the predictions made by each tree. It's like getting the opinion of a bunch of crazy decision trees and then taking the average of their opinions to get a more accurate prediction.

To illustrate this, imagine you're trying to predict the price of a used car. You ask one decision tree that only looks at the year of the car, and it predicts the price is $10,000. Then you ask another decision tree that only looks at the mileage, and it predicts the price is $8,000. Finally, you ask a third decision tree that only looks at the make and model of the car, and it predicts the price is $12,000.

To get the final prediction, you take the average of these three predictions, which is ($10,000 + $8,000 + $12,000)/3 = $10,000. So the final prediction is that the car is worth $10,000.

And that's Random Forest Regression in a nutshell!