Predicting Car Price using Machine Learning

Product Model Development

Submitted by
Vinith Ravi

In partial fullfilment for of two Month Internship in FeyNN Labs:

AI for Small Businesses

28-07-2023

Problem Statement for Car price prediction:

The goal of this project is to develop a robust machine learning model that can accurately predict the price of a car based on its features and specifications. The model will be trained on a dataset containing historical data of various car models, including both new and used vehicles, with associated price information.

The main challenges to address in this problem are as follows:

Data Collection and Pre-processing:

Acquire a comprehensive dataset containing car features such as make, model, year, mileage, engine size, fuel type, transmission type, and other relevant attributes.
Handle missing data and outliers to ensure data quality and integrity.
Conduct exploratory data analysis to gain insights into the data distribution and relationships between variables.

Feature Engineering:

Identify and select the most relevant features that significantly influence the car's price.
Transform categorical variables into numerical representations using techniques like one-hot encoding or label encoding.
Scale or normalize numerical features to bring them to a similar range

Model Selection and Training:

Compare and evaluate different regression algorithms suitable for the problem, such as linear regression, decision trees, random forests, gradient boosting, or neural networks.
Split the dataset into training and testing sets for model validation.
Optimize hyper parameters to achieve the best performance of the chosen model.

Overfitting and Generalization:

Implement techniques to prevent overfitting, such as cross-validation, regularization, and feature selection
Ensure the model generalizes well on unseen data to make accurate predictions for new car samples.

Evaluation Metrics:

Select appropriate evaluation metrics, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE), to assess the model's performance.
Aim to minimize the prediction errors and achieve the best possible model accuracy.

Model Interpretability (Optional):

Depending on the complexity of the chosen model, consider using techniques like feature importance to explain which features have the most significant impact on the car's price.

Deployment and User Interface (Optional):

If applicable, create a user-friendly interface where users can input car features and receive a predicted price. The success of this project will be measured by the model's ability to accurately predict car prices on new and unseen data. Additionally, the development of an efficient and reliable prediction model will provide valuable insights to car sellers, buyers, and other stakeholders in the automotive industry.

Consider the scalability and efficiency of the model for real-time or large-scale applications.

Success Metrics:

The success of this project will be measured based on the model's ability to accurately predict car prices on unseen data. The evaluation will be performed using appropriate regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). The primary goal is to minimize prediction errors and create a reliable and useful tool for car buyers, sellers, and other stakeholders in the automotive industry.

Prototype Selection for Predicting Car Price using Machine Learning

Selecting a prototype for predicting car prices using linear regression involves choosing the linear regression algorithm as the primary model for building the predictive system. Linear regression is a simple and interpretable algorithm that can be a good starting point for this problem, especially when examining the linear relationships between car features and prices.

Prototype: Linear Regression
Reasons for Choosing Linear Regression:

Simplicity and Interpretability: Linear regression is a straightforward algorithm that provides interpretable results. It helps understand the direct relationships between individual features and the target variable (car price).

Well-suited for Linear Relationships: Linear regression is effective when the relationships between features and the target variable are approximately linear. While the relationship between car features and price may not be perfectly linear, it can still capture a reasonable amount of variation in the data.

Quick Model Training: Linear regression is computationally efficient and can quickly be trained on large datasets, making it suitable for prototyping and initial exploration.

Baseline Model: Linear regression can serve as a baseline model against which more complex algorithms can be compared. If linear regression performs well, it may be sufficient for the task, and if not, more advanced algorithms like Random Forest or Gradient Boosting can be explored.

Steps for Model Development:

1. **Data Preparation:** Collect and preprocess the dataset, ensuring it is clean, free of missing values, and properly formatted for linear regression.

2. **Feature Selection:** Select relevant features that are likely to have a significant impact on car prices. Remove any irrelevant or redundant attributes

3. **Data Split:** Divide the dataset into training and testing sets, typically using an 80-20 or 70-30 split, respectively. The training set will be used to train the linear regression model, while the testing set will assess its performance on unseen data.

4. **Model Training:** Train the linear regression model on the training data. The model will estimate the coefficients (slopes) for each feature, indicating the strength and direction of their influence on the car price.

5. **Evaluation:** Evaluate the model's performance on the testing dataset using appropriate regression metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE).

6. **Interpretation:** Analyze the coefficients to understand the impact of each feature on car prices. Positive coefficients indicate a positive correlation, while negative coefficients suggest a negative correlation.

7. **Refinement and Comparison:** If linear regression performs well, it can be used as the final model. Otherwise, more sophisticated algorithms can be explored and compared to improve prediction accuracy.

8. **Deployment:** If the linear regression model meets the desired accuracy and interpretability, it can be deployed as an API or a user-friendly interface for car price predictions.

Remember that linear regression may have limitations when dealing with complex and nonlinear relationships in the data. In such cases, more advanced algorithms like Random Forest, Gradient Boosting, or Neural Networks can be considered for improved predictive performance.

Business/Financial Modeling Objectives:

Market Segmentation and Targeting:

a) Utilize the predictive model to segment the car market based on price ranges and specific car features.
b) Identify target market segments for different car models and price categories.

Pricing Strategy Optimization:

a) Develop pricing strategies based on the model's insights to optimize car prices for maximizing profitability and competitiveness.
b) Analyze price elasticity to understand how changes in prices affect demand and revenue.

Inventory Management:

a) Implement a demand forecasting model using historical car price data to optimize inventory management and prevent overstocking or stockouts.

Competitor Analysis:

Compare predicted prices with competitors' prices to evaluate pricing competitiveness.

a) Identify opportunities to adjust pricing based on competitors' actions and market trends.

Financial Projections:

a. Develop financial projections and scenarios based on the model's predictions to assess potential revenue and profitability outcomes.

Prototype Deployment:

Upon successful development and evaluation of the linear regression model, consider deploying the prototype as an API or a user-friendly interface for car price predictions. The predictive model and business/financial insights can be utilized by car dealerships, manufacturers, and other stakeholders in the automotive industry to make informed pricing decisions and optimize business strategies.

Rationale for Selection of Prototype Development and Business/Financial Modeling on Car Price Prediction:

Linear Regression as Prototype:

1. **Simplicity and Interpretability**: Linear regression is a straightforward algorithm that is easy to understand and interpret. This is crucial for business stakeholders who may not have an in-depth understanding of machine learning techniques but need to comprehend the model's results and insights.
2. **Baseline Model:** Linear regression serves as an excellent starting point for predictive modeling. It provides a baseline against which more complex algorithms can be compared. If linear regression yields satisfactory results, there may be no need for more computationally intensive models.
3. **Feature Importance**: Linear regression allows us to analyze the coefficients of each feature, providing information on their impact on car prices. This feature importance analysis aids in understanding the factors driving car prices in the market.

4. **Linear Regression Suitability**: Linear regression is widely used in various industries and is well-suited for scenarios where the relationships between variables are approximately linear. Since car prices can be influenced by multiple factors in a complex but potentially linear manner, linear regression is an appropriate choice.
5. **Business Impact:** The combination of predictive modeling and financial/business analysis provides actionable insights that can directly impact car dealerships, manufacturers, and other stakeholders. Optimized pricing and inventory management can lead to increased revenue and profitability.
6. **User-Friendly Interface:** By deploying the prototype as an API or user-friendly interface, businesses can easily access and utilize the predictive model and its insights without requiring extensive technical expertise.
7. **Scalability:** The prototype's implementation can be scaled to handle large datasets and real-time data, enabling businesses to adapt to dynamic market conditions effectively.
8. Overall, the chosen prototype development using linear regression and business/financial modeling on car price prediction offers a practical and interpretable approach to address the specific problem of estimating car prices. The combination of machine learning and business insights empowers stakeholders to make data-driven decisions that can improve their competitiveness and profitability in the automotive market.

The solution is divided into the following sections:
1. Data understanding and exploration
2. Data cleaning
3. Data preparation
4. Model building and evaluation

Applicable Regulations for car price prediction

When developing a car price prediction system or any machine learning application, it is essential to consider applicable regulations to ensure compliance, data privacy, and ethical practices. While specific regulations may vary based on the region or country, here are some common regulations that might be relevant for car price prediction:

Data Privacy and Protection:

- General Data Protection Regulation (GDPR) in the European Union or similar data privacy laws in other regions may apply to the collection, storage, and processing of personal data, including customer information used in car price prediction.
- Ensure that proper consent is obtained from individuals whose data is used, and implement measures to secure and protect the data from unauthorized access or breaches.

Fair Credit Reporting Act (FCRA):

- If the car price prediction system uses credit data or credit scores to assess a buyer's ability to pay, it must comply with FCRA regulations, including providing consumers with accurate and fair credit information and obtaining appropriate authorization.

Consumer Protection Laws:

- Various consumer protection laws may govern pricing transparency, advertising practices, and the accuracy of information provided to potential car buyers. The prediction system should adhere to these regulations to avoid misleading consumers.

Antitrust Laws:

- In some regions, antitrust laws may regulate price-fixing and collusion among competitors. Ensure that the car price prediction system does not engage in any anticompetitive behavior or contribute to unfair market practices.

Use of Personal Data for Profiling:

- If the prediction system uses personal data for profiling or decision-making, it must comply with relevant regulations that govern automated decision-making, such as the EU's General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA).

Intellectual Property and Copyright:

- Be mindful of intellectual property rights when using data sources or models from third parties. Obtain proper permissions and licenses to use copyrighted materials, and avoid infringing on any proprietary rights.

Bias and Fairness:

- Ensure that the car price prediction system does not perpetuate biases related to race, gender, ethnicity, or any other protected characteristic. Addressing bias and promoting fairness is crucial to maintaining ethical practices in machine learning.

Industry-Specific Regulations:

- In the automotive industry, there may be specific regulations regarding pricing practices, advertising, and disclosure requirements that the car price prediction system must adhere to.
- It is essential to consult with legal experts and regulatory authorities in the relevant jurisdiction to ensure full compliance with all applicable regulations. Additionally, implement transparency measures to inform users about how their data is used and ensure that the car price prediction system operates in a fair and ethical manner.

Applicable Constraints for car price prediction

When developing a car price prediction system, there are several constraints that need to be considered. These constraints may affect the design, implementation, and deployment of the system.
Some of the common constraints for car price prediction include:

Data Availability and Quality:

- Availability of quality data is critical for building an accurate prediction model. Constraints may arise due to limited or incomplete data, missing values, or data that is not representative of the entire market.
- Ensuring data quality and addressing any biases in the dataset is essential to avoid misleading predictions.

Model Complexity and Training Time:

- More complex machine learning models may require longer training times, making it challenging to develop and deploy the model in real-time applications.
- Balancing model complexity with prediction accuracy and efficiency is crucial, especially if the system is intended for real-time pricing decisions.

Interpretability and Explainability:

- Complex machine learning models like neural networks may provide better predictive performance but can be challenging to interpret and explain.
- However, in certain contexts, interpretability is essential for regulatory compliance or building trust with users.

Regulatory and Legal Constraints:

- The system must comply with applicable regulations, such as data privacy laws, consumer protection laws, and fair credit reporting regulations, depending on the region and the data used.

Hardware and Computational Resources:

- Running prediction models may require significant computational resources, especially for large datasets or complex algorithms. Constraints on hardware and computing power might affect model choices and deployment options.

Online vs. Offline Prediction:

- In real-time applications where car prices need to be predicted quickly, the system must be capable of handling real-time data and providing predictions promptly.
- Offline predictions might be suitable for certain use cases where the pricing decision is not time-sensitive.

Model Accuracy and Error Tolerance:

- There will always be some level of error in predictive models. Determining an acceptable level of prediction error is essential, especially when making critical business decisions based on the predictions.

Overfitting and Generalization:

- Overfitting can occur when the model performs exceptionally well on the training data but poorly on new, unseen data. Ensuring the model generalizes well to new data is essential for accurate predictions.

User Interface and User Experience:

- For applications with user interfaces, constraints related to the design, responsiveness, and user experience need to be considered to ensure the system is user-friendly and accessible.

Ethical Considerations:

- The car price prediction system must avoid bias and discrimination, ensuring fairness and transparency in pricing predictions.
- Balancing these constraints is essential for building a car price prediction system that meets the specific needs of the stakeholders while providing accurate and reliable predictions. Regular monitoring and updates to the system may be necessary to adapt to changing data and market dynamics.

Business Opportunity for Car Price Prediction

Car price prediction presents several lucrative business opportunities for various stakeholders in the automotive industry and related sectors. Some of the key business opportunities include:

Online Car Marketplaces:

- Online platforms that facilitate car buying and selling can integrate car price prediction models to offer users more accurate and data-driven pricing information.
- By providing transparent and reliable price estimates, these marketplaces can attract more potential buyers and sellers, enhancing user engagement and increasing platform usage.

Dealerships and Car Sellers:

- Car dealerships and sellers can utilize car price prediction models to optimize their pricing strategies. This includes setting competitive prices for their inventory and understanding market demand for specific car models and features.
- Accurate price predictions can help dealerships maximize profits and minimize the time cars spend on the lot, resulting in improved inventory turnover.

Car Manufacturers and OEMs:

- Car manufacturers can leverage price prediction models to estimate the value of their vehicles throughout their lifecycle, from launch to the end of production.
- Understanding how different features and specifications influence car prices can guide manufacturers in product design and marketing decisions.

Financial Institutions and Insurance Companies:

- Banks and financial institutions can use car price prediction models to assess the value of vehicles for loan approval and collateral evaluation.
- Insurance companies can incorporate price predictions to determine appropriate coverage and premiums based on the car's estimated value.

Car Rental Companies:

- Car rental businesses can benefit from price prediction to optimize rental rates based on market demand and seasonal fluctuations.
- Accurate pricing can lead to increased customer satisfaction and improved rental utilization.

Auction Houses:

- Auction houses that deal with used cars can leverage price prediction models to estimate reserve prices for auctioned vehicles.
- This can attract more potential buyers to auctions and ensure fair pricing for both buyers and sellers

Car Valuation Services:

- Companies offering car valuation services can enhance their offerings by incorporating data-driven price predictions for more accurate and up-to-date valuations.
- This can attract more customers seeking reliable and trustworthy valuations for their vehicles.

Automotive Market Research:

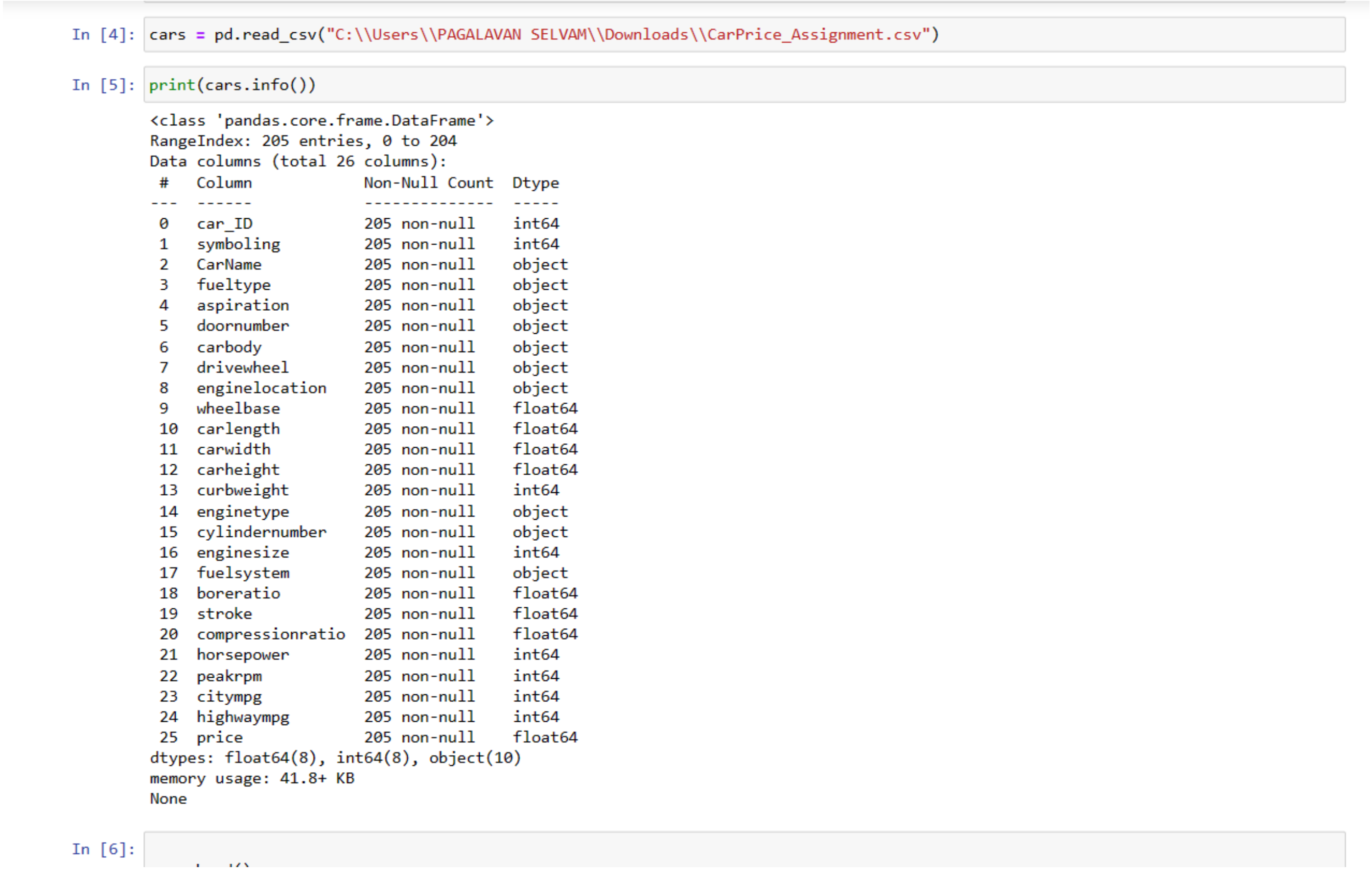
- Market research firms can use price prediction models to analyze car price trends and consumer preferences, providing valuable insights to automakers, investors, and policymakers.
- Overall, integrating car price prediction models into various aspects of the automotive industry offers an opportunity to make data-driven and informed decisions, enhance customer satisfaction, and optimize business operations. The predictive capabilities can lead to a competitive advantage for businesses and improve overall market efficiency in the automotive sector.
- Based on various market surveys, the consulting firm has gathered a large dataset of different types of cars across the American market.

Business Objectives:

We required to apply some data science techniques for the price of cars with the available independent variables. That should help the management to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain price levels.

Data understanding and exploration:

Summary of data: 205 rows, 26 columns, no null values



```
In [4]: cars = pd.read_csv("C:\\Users\\PAGALAVAN SELVAM\\Downloads\\CarPrice_Assignment.csv")

In [5]: print(cars.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
None

In [6]:
```

The column"**Price**" is the target variable and rest of the columns are independent variables.

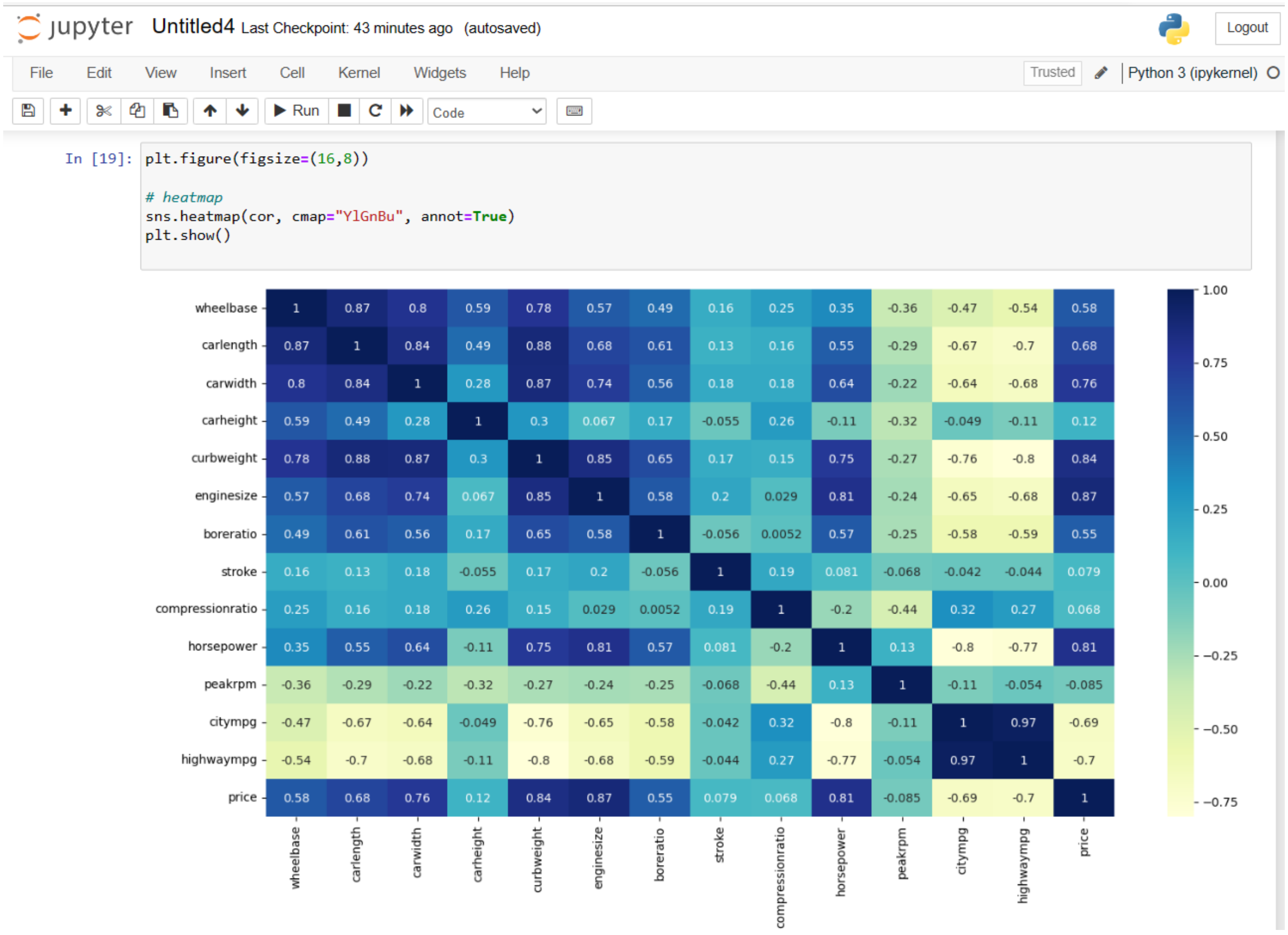
The independent variables are again divided into Categorical and Numerical variables.

Numerical variables: ['wheelbase', 'carlength', 'carwidth', 'carheight', 'curbweight', 'enginesize', 'boreratio', 'stroke', 'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg']

Categorical variables: ['symboling', 'fueltype', 'aspiration', 'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'enginetype', 'cylindernumber', 'fuelsystem' 'car_name']

Heatmap to show correlation of Numerical and Target variable:

Now let's plot Heatmap which is pretty useful to visualise multiple correlations among numerical variables. We have also used the Target variable "**Price**" to understand the correlation of numerical variables with it.



```
In [19]: plt.figure(figsize=(16,8))

         # heatmap
         sns.heatmap(cor, cmap="YlGnBu", annot=True)
         plt.show()
```

	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg	price
wheelbase	1	0.87	0.8	0.59	0.78	0.57	0.49	0.16	0.25	0.35	-0.36	-0.47	-0.54	0.58
carlength	0.87	1	0.84	0.49	0.88	0.68	0.61	0.13	0.16	0.55	-0.29	-0.67	-0.7	0.68
carwidth	0.8	0.84	1	0.28	0.87	0.74	0.56	0.18	0.18	0.64	-0.22	-0.64	-0.68	0.76
carheight	0.59	0.49	0.28	1	0.3	0.067	0.17	-0.055	0.26	-0.11	-0.32	-0.049	-0.11	0.12
curbweight	0.78	0.88	0.87	0.3	1	0.85	0.65	0.17	0.15	0.75	-0.27	-0.76	-0.8	0.84
enginesize	0.57	0.68	0.74	0.067	0.85	1	0.58	0.2	0.029	0.81	-0.24	-0.65	-0.68	0.87
boreratio	0.49	0.61	0.56	0.17	0.65	0.58	1	-0.056	0.0052	0.57	-0.25	-0.58	-0.59	0.55
stroke	0.16	0.13	0.18	-0.055	0.17	0.2	-0.056	1	0.19	0.081	-0.068	-0.042	-0.044	0.079
compressionratio	0.25	0.16	0.18	0.26	0.15	0.029	0.0052	0.19	1	-0.2	-0.44	0.32	0.27	0.068
horsepower	0.35	0.55	0.64	-0.11	0.75	0.81	0.57	0.081	-0.2	1	0.13	-0.8	-0.77	0.81
peakrpm	-0.36	-0.29	-0.22	-0.32	-0.27	-0.24	-0.25	-0.068	-0.44	0.13	1	-0.11	-0.054	-0.085
citympg	-0.47	-0.67	-0.64	-0.049	-0.76	-0.65	-0.58	-0.042	0.32	-0.8	-0.11	1	0.97	-0.69
highwaympg	-0.54	-0.7	-0.68	-0.11	-0.8	-0.68	-0.59	-0.044	0.27	-0.77	-0.054	0.97	1	-0.7
price	0.58	0.68	0.76	0.12	0.84	0.87	0.55	0.079	0.068	0.81	-0.085	-0.69	-0.7	1

The heatmap shows some useful insights:

Correlation of **target variable "Price"** with **independent variables**:\

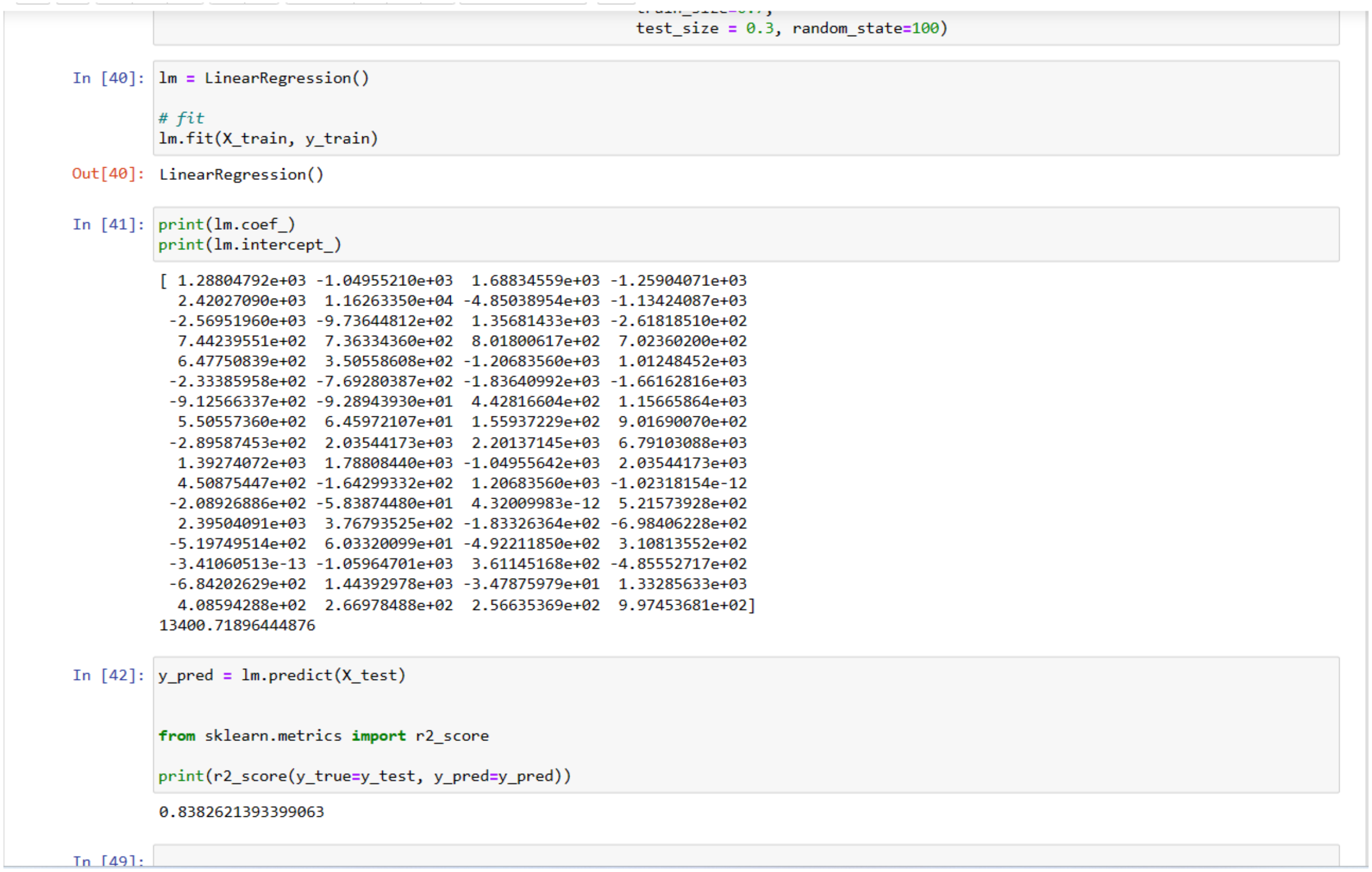
Price is highly (positively) correlated with wheelbase, carlength, carwidth, curbweight, enginesize, horsepower (notice how all of these variables represent the size/weight/engine power of the car)

Price is negatively correlated to ‘citympg’ and ‘highwaympg’ (-0.70 approximately). This suggest that cars having high mileage may fall in the ‘economy’ cars category, and are priced lower (think Maruti Alto/Swift type of cars, which are designed to be affordable by the middle class, who value mileage more than horsepower/size of car etc.)

Correlation among independent variables:
Many independent variables are highly correlated (look at the top-left part of matrix): wheelbase, carlength, curbweight, enginesize etc. are all measures of ‘size/weight’, and are positively correlated Thus, while building the model, we’ll have to pay attention to multicollinearity (especially linear models, such as linear and logistic regression, suffer more from multicollinearity).

Data Cleaning:
We’ve seen that there are no missing values in the dataset.
We’ve also seen that variables are in the correct format, except “symboling”, which should rather be a categorical variable (so that dummy variable are created for the categories).
We have also done data preprocessing on The variable “CarName” and created a new variable called as “car_company.

\Data Preparation:
Let’s now prepare the data for model building.
Split the data into X and y.



```
                                                           test_size = 0.3, random_state=100)

In [40]: lm = LinearRegression()

         # fit
         lm.fit(X_train, y_train)

Out[40]: LinearRegression()

In [41]: print(lm.coef_)
         print(lm.intercept_)

         [ 1.28804792e+03 -1.04955210e+03  1.68834559e+03 -1.25904071e+03
           2.42027090e+03  1.16263350e+04 -4.85038954e+03 -1.13424087e+03
          -2.56951960e+03 -9.73644812e+02  1.35681433e+03 -2.61818510e+02
           7.44239551e+02  7.36334360e+02  8.01800617e+02  7.02360200e+02
           6.47750839e+02  3.50558608e+02 -1.20683560e+03  1.01248452e+03
          -2.33385958e+02 -7.69280387e+02 -1.83640992e+03 -1.66162816e+03
          -9.12566337e+02 -9.28943930e+01  4.42816604e+02  1.15665864e+03
           5.50557360e+02  6.45972107e+01  1.55937229e+02  9.01690070e+02
          -2.89587453e+02  2.03544173e+03  2.20137145e+03  6.79103088e+03
           1.39274072e+03  1.78808440e+03 -1.04955642e+03  2.03544173e+03
           4.50875447e+02 -1.64299332e+02  1.20683560e+03 -1.02318154e-12
          -2.08926886e+02 -5.83874480e+01  4.32009983e-12  5.21573928e+02
           2.39504091e+03  3.76793525e+02 -1.83326364e+02 -6.98406228e+02
          -5.19749514e+02  6.03320099e+01 -4.92211850e+02  3.10813552e+02
          -3.41060513e-13 -1.05964701e+03  3.61145168e+02 -4.85552717e+02
          -6.84202629e+02  1.44392978e+03 -3.47875979e+01  1.33285633e+03
           4.08594288e+02  2.66978488e+02  2.56635369e+02  9.97453681e+02]
         13400.71896444876

In [42]: y_pred = lm.predict(X_test)


         from sklearn.metrics import r2_score

         print(r2_score(y_true=y_test, y_pred=y_pred))

         0.8382621393399063

In [49]:
```

Building the first model with all the features

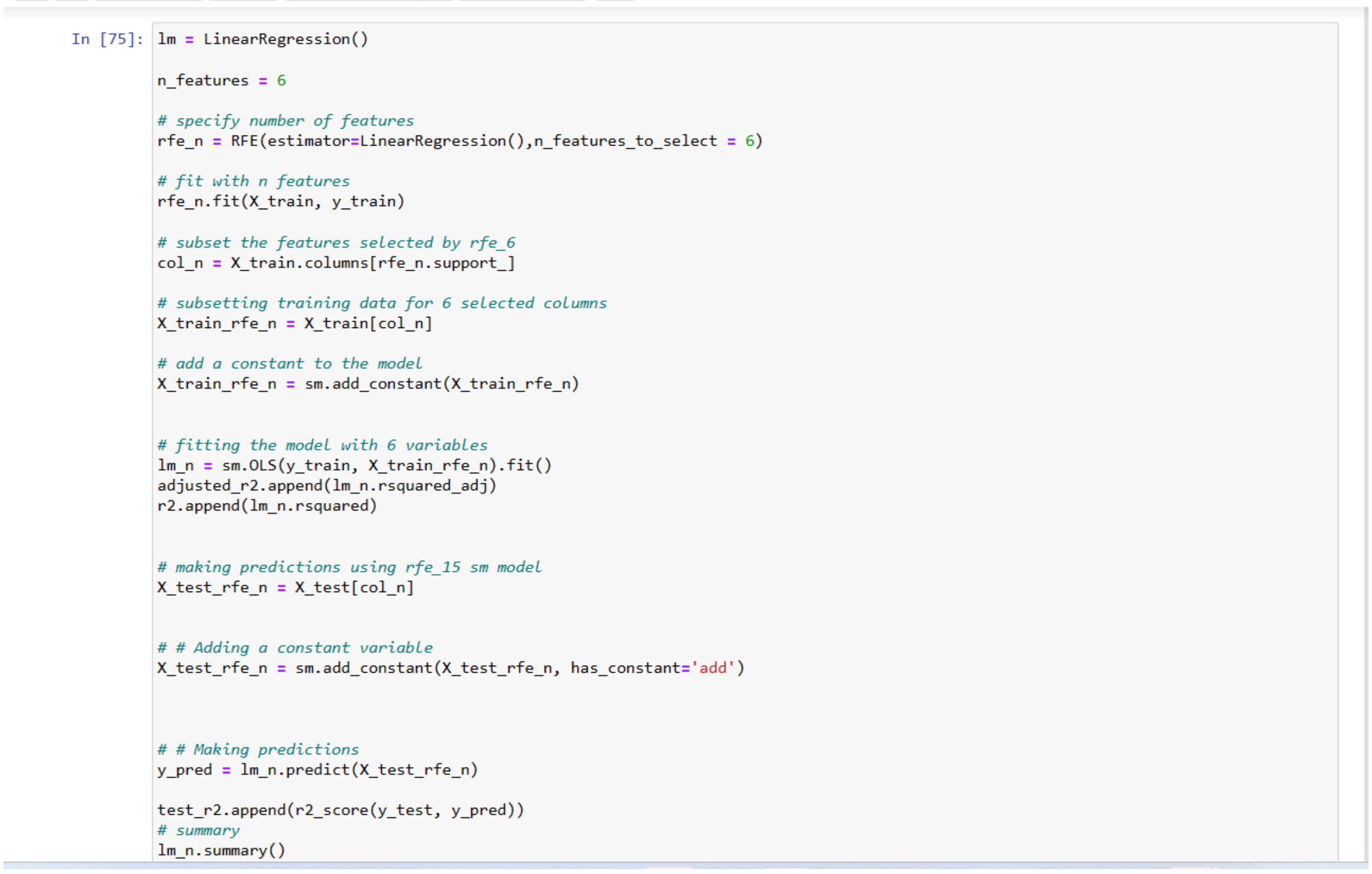
R-squared = 0.83826213934

Not bad, we are getting approx. 83% r-squared with all the variables. Let's see how much we can get with lesser features.

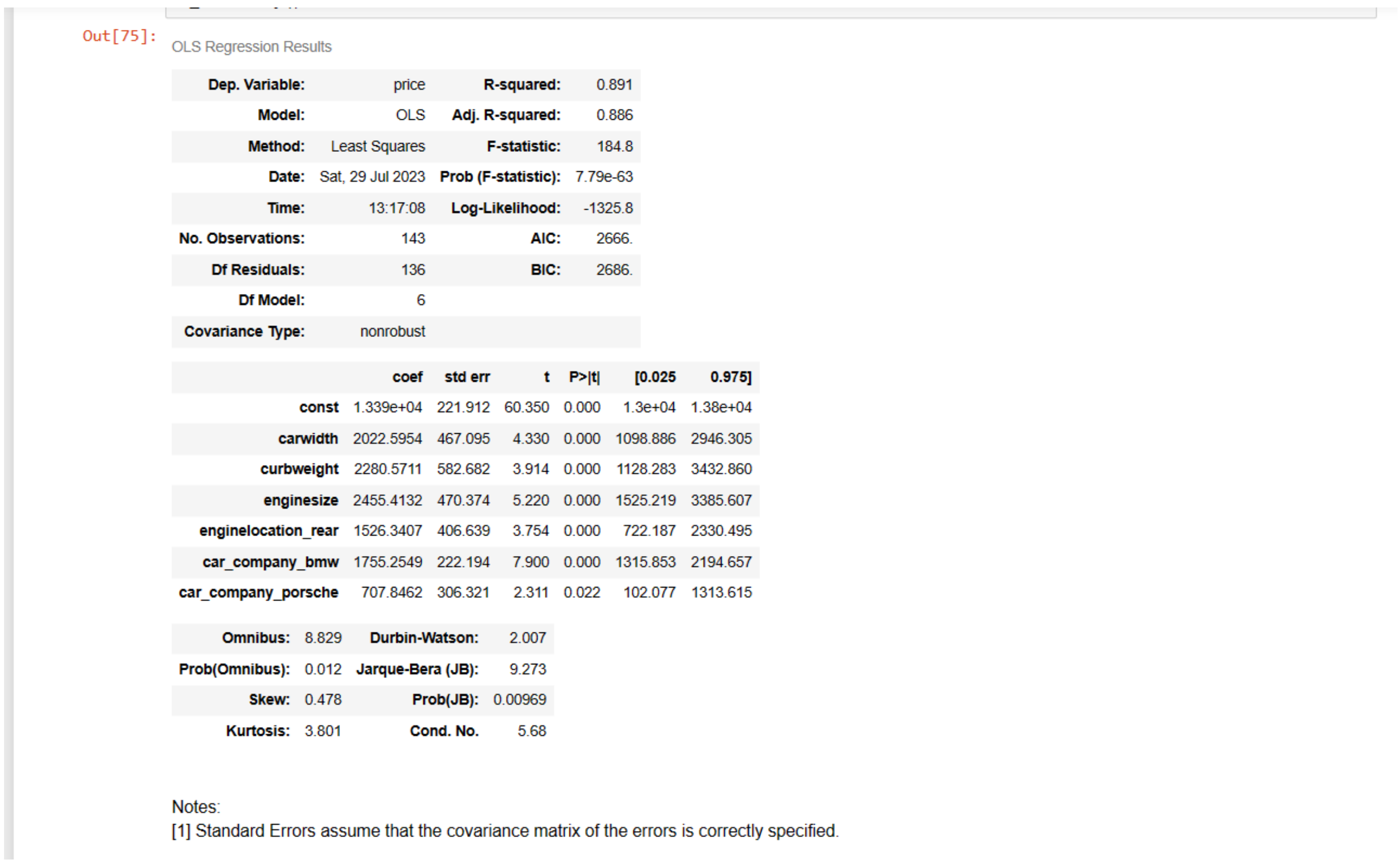
Let's now build a model using recursive feature elimination to select features. We'll first start off with an arbitrary number of features, and then use the "**statsmodels"** library to build models using the shortlisted features (this is because sklearn doesn't have adjusted r-squared but statsmodels has).

Choosing the optimal number of features for Model building:

One way to choose the optimal number of features is to make a plot between number of features (n_features) vs adjusted r-squared, and then choose the best value of n_features.



```
In [75]: lm = LinearRegression()

         n_features = 6

         # specify number of features
         rfe_n = RFE(estimator=LinearRegression(),n_features_to_select = 6)

         # fit with n features
         rfe_n.fit(X_train, y_train)

         # subset the features selected by rfe_6
         col_n = X_train.columns[rfe_n.support_]

         # subsetting training data for 6 selected columns
         X_train_rfe_n = X_train[col_n]

         # add a constant to the model
         X_train_rfe_n = sm.add_constant(X_train_rfe_n)


         # fitting the model with 6 variables
         lm_n = sm.OLS(y_train, X_train_rfe_n).fit()
         adjusted_r2.append(lm_n.rsquared_adj)
         r2.append(lm_n.rsquared)


         # making predictions using rfe_15 sm model
         X_test_rfe_n = X_test[col_n]


         # # Adding a constant variable
         X_test_rfe_n = sm.add_constant(X_test_rfe_n, has_constant='add')



         # # Making predictions
         y_pred = lm_n.predict(X_test_rfe_n)

         test_r2.append(r2_score(y_test, y_pred))
         # summary
         lm_n.summary()
```


Out[75]:

OLS Regression Results

Dep. Variable:	price	**R-squared:**	0.891
Model:	OLS	**Adj. R-squared:**	0.886
Method:	Least Squares	**F-statistic:**	184.8
Date:	Sat, 29 Jul 2023	**Prob (F-statistic):**	7.79e-63
Time:	13:17:08	**Log-Likelihood:**	-1325.8
No. Observations:	143	**AIC:**	2666.
Df Residuals:	136	**BIC:**	2686.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.339e+04	221.912	60.350	0.000	1.3e+04	1.38e+04
carwidth	2022.5954	467.095	4.330	0.000	1098.886	2946.305
curbweight	2280.5711	582.682	3.914	0.000	1128.283	3432.860
enginesize	2455.4132	470.374	5.220	0.000	1525.219	3385.607
enginelocation_rear	1526.3407	406.639	3.754	0.000	722.187	2330.495
car_company_bmw	1755.2549	222.194	7.900	0.000	1315.853	2194.657
car_company_porsche	707.8462	306.321	2.311	0.022	102.077	1313.615

Omnibus:	8.829	**Durbin-Watson:**	2.007
Prob(Omnibus):	0.012	**Jarque-Bera (JB):**	9.273
Skew:	0.478	**Prob(JB):**	0.00969
Kurtosis:	3.801	**Cond. No.**	5.68

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Based on the plot, we can choose the number of features considering the r2_score we are looking for. Note that there are a few caveats in this approach, and there are more sophisticated techniques to choose the optimal number of features:

Cross-validation: In this case, we have considered only one train-test split of the dataset; the values of r-squared and adjusted r-squared will vary with train-test split. Thus, cross-validation is a more commonly used technique (you divide the data into multiple train-test splits into 'folds', and then compute average metrics such as r-squared across the 'folds'.

The values of r-squared and adjusted r-squared are computed based on the training set, though we must always look at metrics computed on the test set. For e.g. in this case, the test r2 actually goes down with increasing n — this phenomenon is called 'overfitting', where the performance on training set is good because the model has in some way 'memorised' the dataset, and thus the performance on test set is worse.

Thus, we can choose anything between 4 and 12 features, since beyond 12, the test r2 goes down; and at lesser than 4, the r2_score is too less.

In fact, the test_r2 score doesn't increase much anyway from n=6 to n=12. It is thus wiser to choose a simpler model, and so let's choose n=6.

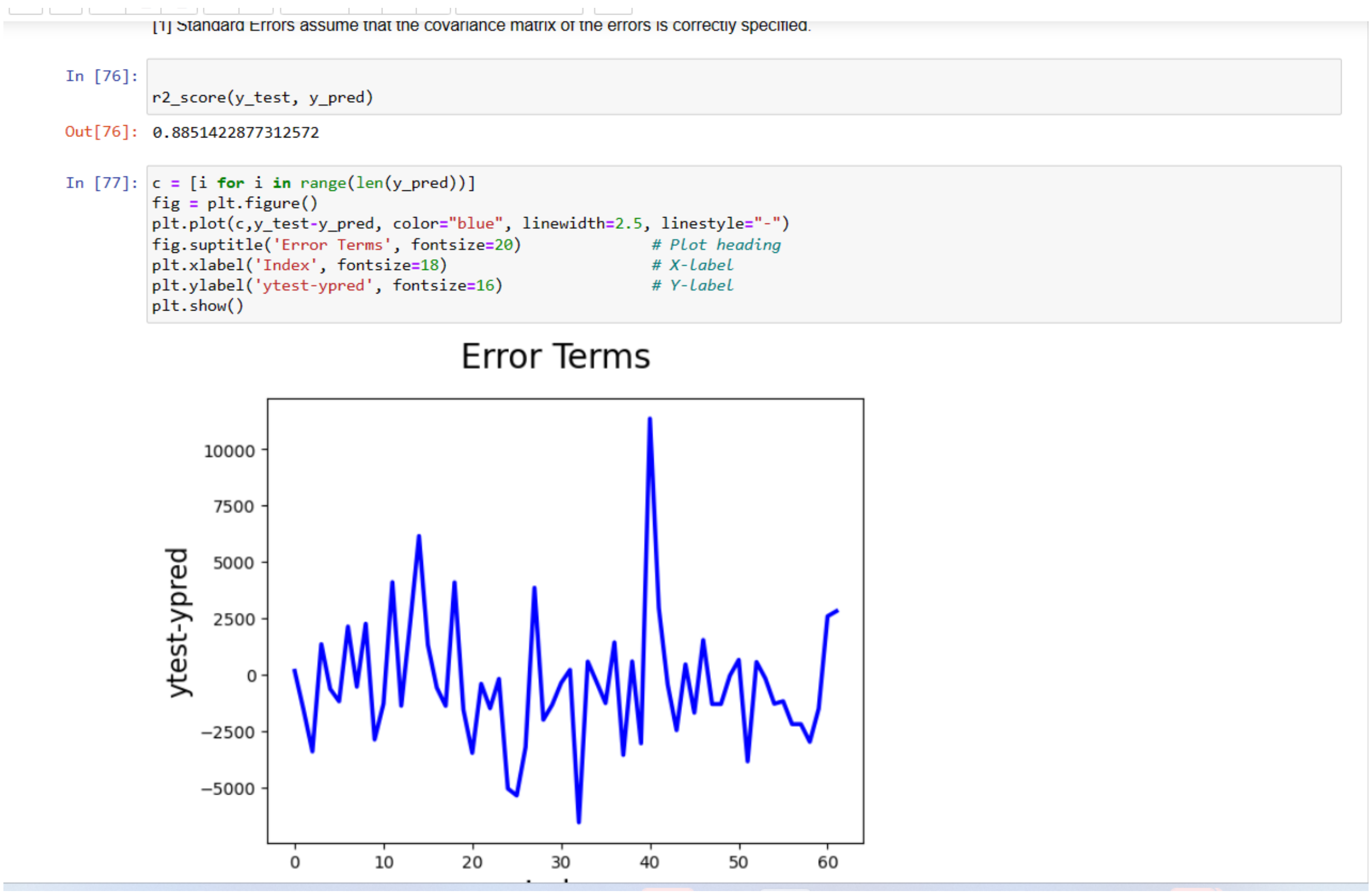
r2-squared = 0.88514228773125714

So the model has accuracy of 88.51% on test data which is good. There are other ways of model evaluation as well, let's see those points.

Final Model Evaluation:

Let's now evaluate the model in terms of its assumptions. We should test that:

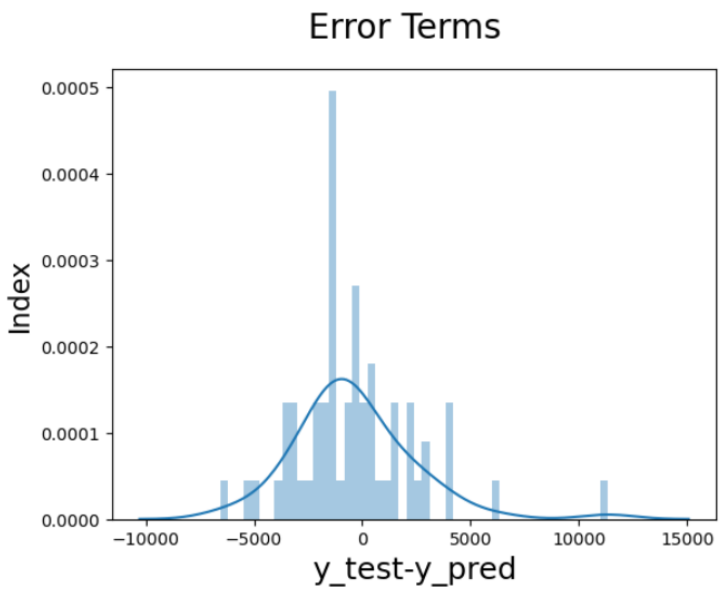
- The error terms are normally distributed with mean approximately 0.
- There is little correlation between the predictors.
- Homoscedasticity, i.e. the 'spread' or 'variance' of the error term (y_true-y_pred) is constant.



```
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [76]: r2_score(y_test, y_pred)
Out[76]: 0.8851422877312572

In [77]: c = [i for i in range(len(y_pred))]
         fig = plt.figure()
         plt.plot(c,y_test-y_pred, color="blue", linewidth=2.5, linestyle="-")
         fig.suptitle('Error Terms', fontsize=20)              # Plot heading
         plt.xlabel('Index', fontsize=18)                      # X-label
         plt.ylabel('ytest-ypred', fontsize=16)                # Y-label
         plt.show()
```

```
In [78]: fig = plt.figure()
         sns.distplot((y_test-y_pred),bins=50)
         fig.suptitle('Error Terms', fontsize=20)                  # Plot heading
         plt.xlabel('y_test-y_pred', fontsize=18)                  # X-label
         plt.ylabel('Index', fontsize=16)                          # Y-label
         plt.show()
```

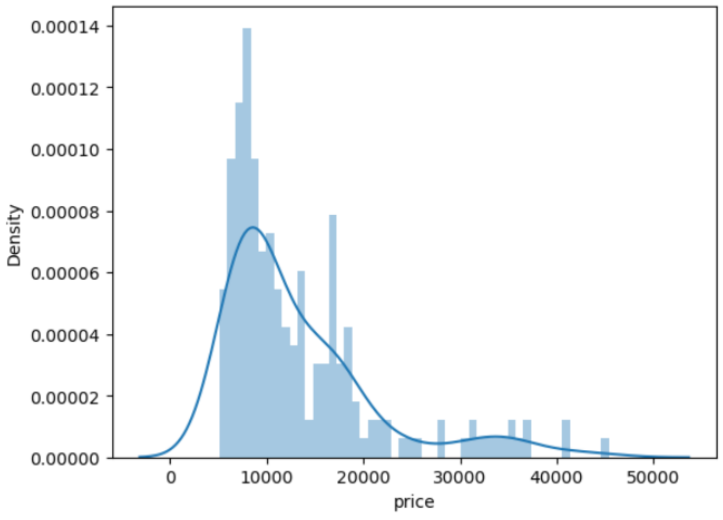
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function an
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



```
Out[79]: -382.7363245027597
```

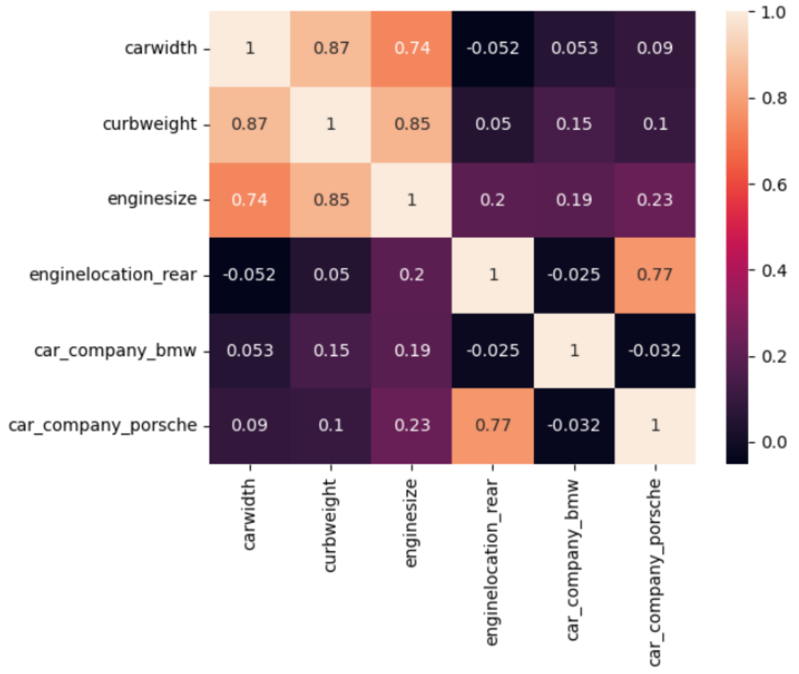
```
In [80]: sns.distplot(cars['price'],bins=50)
         plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function an
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



```
In [81]: predictors = ['carwidth', 'curbweight', 'enginesize',
                       'enginelocation_rear', 'car_company_bmw', 'car_company_porsche']

         cors = X.loc[:, list(predictors)].corr()
         sns.heatmap(cors, annot=True)
```

```
In [81]: predictors = ['carwidth', 'curbweight', 'enginesize',
                       'enginelocation_rear', 'car_company_bmw', 'car_company_porsche']

         cors = X.loc[:, list(predictors)].corr()
         sns.heatmap(cors, annot=True)
         plt.show()
```



Conclusion:

Though this is the most simple model we've built till now, the final predictors still seem to have high correlations. One can go ahead and remove some of these features, though that will affect the adjusted-r2 score significantly (you should try doing that).

Thus, for now, the final model consists of the 6 variables mentioned above.