Text

Description automatically generated

Toronto, Canada

**A Report on**

Module 2 Midweek Project

Predictive Analytics

(ALY 6020)

Guided by:

Prof. Dr. Harpreet Sharma

|  |  |  |
| --- | --- | --- |
| Name of Student | Submitted By:  NUID | Date of submission |
| Parth Shah | 002956963 | 22nd January, 2023 |

**Table of Figure**

Figure 1: Importing Libraries…………………………………………………………………….03

Figure 2: Importing Dataset………………………………………………………………………04

Figure 3: Info of dataset…………………………………………………………………………..05

Figure 4: Describe about dataset………………………………………………………………….06

Figure 5: Dropping Duplicate Values…………………………………………………………….06

Figure 6: Dropping Columns……………………………………………………………………..07

Figure 7: Correlation Plot………………………………………………………………………...08

Figure 8: Top variables that influences car price………………………………………………….09

Figure 9: Determining Categorical variables……………………………………………………..10

Figure 10: Creating dummy variable……………………………………………………………..11

Figure 11: Concatenating dummy variables into DataFrame……………………………………12

Figure 12: Splitting dataset and scaling…………………………………………………………..13

Figure 13: OLS Regression Result……………………………………………………………….14

Figure 14: Accuracy of Model……………………………………………………………………15

Figure 15: Actual Vs Predicted Price……………………………………………………………16

**Introduction**

* In addition to the price of each automobile, a car dataset with a price variable often includes details on the make, model, year, engine size, fuel type, and other characteristics of the many types of cars.
* This kind of dataset is frequently used in the automobile sector for research and analysis, including forecasting and pricing, as well as for the creation of prediction models for vehicle sales and other related applications.

**Data Analysis**

Text, letter

Description automatically generated

**Figure 1: Importing Libraries**

* The above code imports a number of libraries and modules that are commonly used in data analysis and machine learning.
* numpy and pandas are used for data manipulation and handling.
* matplotlib and seaborn are used for data visualization.
* statsmodels.api is used for statistical modeling and hypothesis testing.
* sklearn is a library that provides a variety of machine learning models, including linear regression, and preprocessing methods such as StandardScaler and MinMaxScaler
* train\_test\_split is used for dividing the data into training and test sets.
* linear\_model and LinearRegression are used for fitting a linear regression model.
* mean\_absolute\_error, mean\_squared\_error, r2\_score is used for evaluating the performance of the model.
* The line warnings.filterwarnings('ignore') is used to suppress warning messages in the output, which can help make the output more readable.

A picture containing graphical user interface

Description automatically generated

**Figure 2: Importing Dataset**

* The above code reads in a CSV file called "CarPrice\_Assignment.csv" located in the directory "D:/PARTH SHAH/R/ALY 6020/Module 2 Midweek Project/" and assigns it to the variable df. The pd.read\_csv() function from the pandas library is used to read in the data and create a DataFrame.
* The df.head(5) function is used to display the first 5 rows of the DataFrame. This can be useful for quickly previewing the data and ensuring that it has been imported correctly. It will show the first 5 records of the dataframe which will help to understand the structure of data.

Table

Description automatically generated

**Figure 3: Info of dataset**

* The df.info() method is used to display details about the DataFrame, including its size in memory use, the number of rows and columns, and the data types for each column.
* It will provide a summary of the dataframe, including its size, number of columns, data types, non-null values, and memory utilization.
* Understanding the dataframe's structure will make it easier to spot any issues with data types or missing information.

Graphical user interface, application

Description automatically generated with medium confidence

**Figure 4: Describe about dataset.**

* The df.describe() function is used to display a summary statistics of the DataFrame. It will give the count, mean, standard deviation, minimum, 25th percentile, 50th percentile (Median), 75th percentile, and Maximum of the numerical columns.
* This can be useful for getting a quick understanding of the distribution of the data and identifying any potential outliers or errors.
* It will give a statistical summary of the DataFrame. It will also help to identify any missing values as count of missing values will be zero.

Graphical user interface

Description automatically generated with medium confidence

**Figure 5: Dropping Duplicate Values**

* To delete duplicate rows from a DataFrame, use the df.drop duplicates() method. By default, it evaluates all columns while looking for duplicate rows and only maintains the first instance.
* It will deliver a fresh DataFrame with no repeated entries. When the data is messy and contains duplicate values, it is helpful. It will aid in reducing DataFrame size and enhancing analysis performance.

Graphical user interface

Description automatically generated with medium confidence

**Figure 6: Dropping Columns**

* The df.drop() method is used in the code above to remove certain columns from the DataFrame. The names of the columns to be discarded are specified using the columns option. The columns that are being removed include "CarName," "car ID," "enginetype," "fuel system," "engine location," "aspiration," "fuel type," "stroke," "compression ratio," and "peak rpm."
* To specify that columns, not rows, should be discarded, use the axis=1 argument. Without generating a new DataFrame, modifications to the existing DataFrame are made using the inplace=True argument.

Table

Description automatically generated

**Figure 7: Correlation Plot**

* The above code creates a correlation matrix for the DataFrame df and visualizes it using a heatmap.
* The plt.figure(figsize=(25,20)) sets the size of the plot.
* The corr = df.corr() computes the pairwise correlation of all columns in the DataFrame.
* The sns.heatmap(corr, annot=True, cmap="YlGnBu") creates a heatmap of the correlation matrix using the seaborn library. The annot=True parameter displays the correlation coefficients on the heatmap and cmap is the color map used to show the correlation.
* The plt.show() function is used to display the plot.
* This will create a heatmap where the color of the cells represents the correlation coefficient between two variables. The darker the color, the higher the correlation. It will help to identify the relationship between different variables and how much they are correlated. It will also help to identify any multicollinearity issues, which can affect the accuracy of the model.
* Price, engine size, curb weight, and horsepower are the three key factors.

Graphical user interface, text, application

Description automatically generated

**Figure 8: Top variables that influences car price.**

* The above code creates a correlation matrix for the DataFrame df and extracts the top 4 correlated variables with the target variable 'price'.
* The corr\_matrix = df.corr() computes the pairwise correlation of all columns in the DataFrame.
* The top\_corr\_vars = corr\_matrix['price'].sort\_values(ascending=False).head(4) sorts the correlation matrix by the correlation with the target variable 'price' in descending order and selects the top 4 correlated variables.
* The above code will print the top 4 most correlated variables with the target variable 'price' in descending order. It will help to identify the most important variables that are affecting the target variable and will be useful in building a predictive model.
* The factors that have the biggest impact on vehicle prices are engine size, curb weight, and horsepower.

Table

Description automatically generated

**Figure 9: Determining Categorical variables.**

* The above code is used to extract the categorical variables from the DataFrame df.
* The categorical\_variables = df.select\_dtypes(include=['object']) selects all columns that have the data type 'object', which is often used to represent categorical variables.
* The categorical\_variables.head() function is used to display the first 5 rows of the extracted categorical variables.
* It will return the categorical variables present in the DataFrame. This is useful when we want to perform some analysis on categorical variables or when we want to convert categorical variables into numerical variables for building a model.

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 10: Creating dummy variable.**

* The above code is used to convert the categorical variables into dummy variables. Dummy variables are binary variables that are used to represent a categorical variable with more than 2 categories.
* The pd.get\_dummies(categorical\_variables,drop\_first=True) function from the pandas library is used to convert the categorical variables into dummy variables. The drop\_first=True parameter is used to drop the first level of each categorical variable, to avoid the issue of multicollinearity.
* The dummies.head() function is used to display the first 5 rows of the dummy variables.
* It will create a new dataframe with the dummy variables and drop the first level of each categorical variable. This is useful when we want to build a model using categorical variables as it converts them into numerical variables. It's also useful when we want to analyze the effect of different categories on the target variable.

A picture containing graphical user interface

Description automatically generated

**Figure 11: Concatenating dummy variables into DataFrame.**

* The above code is used to concatenate the dummy variables dataframe with the main dataframe and drop the original categorical variables.
* The pd.concat([df,dummies],axis=1) function is used to concatenate the dummy variables dataframe with the main dataframe along the columns (axis=1).
* The df.drop(columns=categorical\_variables.columns,axis=1,inplace=True) function is used to drop the original categorical variables from the main dataframe. The columns parameter is used to specify the names of the columns to be dropped. The axis=1 parameter is used to specify that columns, not rows, should be dropped. The inplace=True parameter is used to make the changes to the DataFrame in place, without creating a new DataFrame.
* The df.head(5) function is used to display the first 5 rows of the dataframe after concatenation and dropping the original categorical variables.
* This will update the main DataFrame with the dummy variables and drop the original categorical variables. This is useful when we want to build a model using categorical variables as it converts them into numerical variables. It's also useful when we want to analyze the effect of different categories on the target variable.

Graphical user interface, text, application, email

Description automatically generated

**Figure 12: Splitting dataset and scaling.**

* This code is defining two variables, x and y, by dropping the 'price' column from a DataFrame called df\_train and storing it in x, and then storing the 'price' column in y.
* Then, it splits the data into training and testing sets using the train\_test\_split function from the sklearn.model\_selection module, with the test set size being 25% and a fixed random state of 42.
* Finally, it applies scaling on the training set using the StandardScaler method from the sklearn.preprocessing module.

Table

Description automatically generated

**Figure 13: OLS Regression Result**

* This code is creating a linear regression model using the LinearRegression class from the sklearn.linear\_model module, and fitting it to the training data using the fit() method. Then it is using the model to make predictions on the test set and storing them in the variable y\_pred.
* Then, it is using the statsmodel library to add a constant to the x variable and then creating an OLS (ordinary least squares) model using the sm.OLS(y,X) and fitting the model using the fit() method. Finally, it prints the summary of the model using the summary() method.
* The summary of the model includes the R-squared value(0.880), which represents the proportion of the variance in the dependent variable that is predictable from the independent variable and some other statistics like coefficients, standard error, t-values, p-values, etc. which can be used to determine the significance of each feature in the model.

Graphical user interface, text

Description automatically generated

**Figure 14: Accuracy of Model**

* The score () method in scikit-learn returns the coefficient of determination R^2 of the prediction. R^2 is a statistical measure of how well the regression line approximates the real data points. An R^2 of 1 indicates that the regression line perfectly fits the data. An R^2 of 0 means that the line does not fit the data at all.
* In this case, the lin\_reg object is being used to fit a linear regression model to the x\_train and y\_train data, and the score () method is being used to evaluate the accuracy of that model using the same training data.
* Here, the linear regression model's accuracy is 90%.

Chart, scatter chart

Description automatically generated

**Figure 15: Actual Vs Predicted Price**

* The code you've provided is creating a scatter plot using the seaborn library's regplot() function. The x-axis of the plot represents the predicted price, which is obtained by using the predict() method on the lin\_reg object and passing in the x\_test data as the input. The y-axis represents the actual price, which is the corresponding target variable in the y\_test data.
* The xlabel(), ylabel(), and title() functions are used to add labels to the x and y axis and title of the plot respectively. The show () function is used to display the plot.
* This plot will show how well the linear regression model is able to predict the actual price of the test data. Ideally, the points in the plot should be close to the 45-degree line, indicating that the predicted price is very close to the actual price.

**Conclusion**

* Based on the code provided and the plots generated, it appears that a linear regression model was created and trained using a car dataset. The model's accuracy was evaluated using the R^2 score and a scatter plot was generated to visualize the relationship between the predicted prices and the actual prices.
* In conclusion, linear regression is a simple and fast technique for predicting a continuous variable. It can be used to model a linear relationship between the dependent and independent variables. However, it is important to note that linear regression assumes that the relationship between the variables is linear and that the errors are normally distributed and have constant variance. If this assumption is not met, the model may not be the best fit for the data.

**Reference**

* Create a correlation Matrix using Python - GeeksforGeeks. (2020, November 9). GeeksforGeeks. Retrieved January 22, 2023, from <https://www.geeksforgeeks.org/create-a-correlation-matrix-using-python/>
* Z., & posts by Zach, V. A. (2022, August 26). How to Perform OLS Regression in Python (With Example) - Statology. Statology. Retrieved January 22, 2023, from <https://www.statology.org/ols-regression-python/>
* Li, L. (2019, February 5). Introduction to Linear Regression in Python. Medium. Retrieved January 22, 2023, from <https://towardsdatascience.com/introduction-to-linear-regression-in-python-c12a072bedf0>