SM1 PROJECT - REGRESSION

-Charvi Dave(J015), Divya Dhulipala (J017), Khyaati Nayak (J044), Calvin Pinto (J048), Riya Vaze(J064)

PROBLEM STATEMENT

- An automobile company aspires to branch out and enter the US market by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts.
- They have contracted an automobile consulting company to understand the factors on which the pricing of cars depends. Specifically, they want to understand the factors affecting the pricing of cars in the American market, since those may be very different from the other country market.

WHAT DOES THE COMPANY WANT TO KNOW?

- Which variables are highly significant in estimating the price of the car.
- How well those variables describe the price of a car.

WHAT IS OUR GOAL?

- We are required to model the price of cars with the available independent variables.
- It will be used by the company 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.
- Further, the model will be a good way for the company to understand the pricing dynamics of a new market.

DATA CLEANING

- Process of preparing data for analysis by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated or improperly formatted.
- We have implemented data cleaning in the following ways-
    - > Dropping unnecessary columns.
    - > Finding and removing null values.
    - > Finding and removing missing values.
    - > Analyzing outliers.
    - > Identifying errors in spellings/names.
    - > Checking for duplicate values.
    - > Changing incorrect data types.
    - > Segregation of numerical and categorical values.

DATA VISUALISATION

- Process of cleaning and transforming raw data prior to processing and analysis.
- It is an important step prior to processing and often involves reformatting data, making corrections to data and the combining of data sets to enrich data.

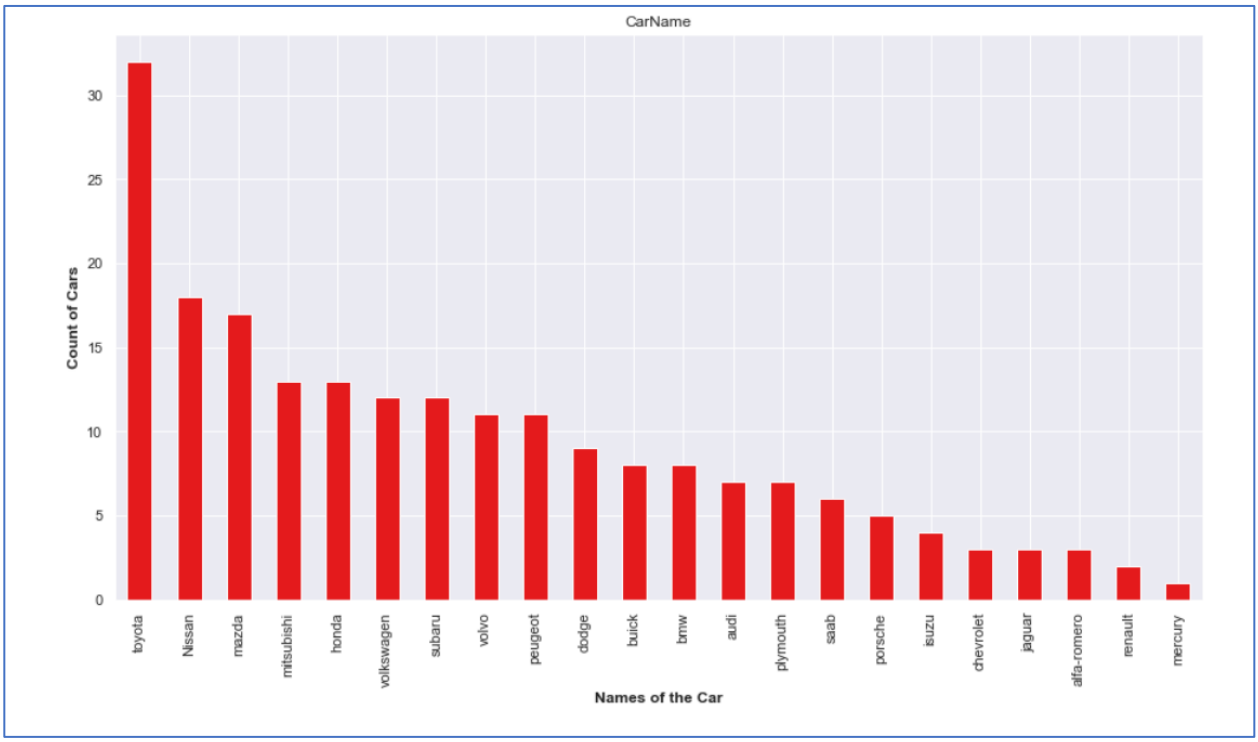


Fig: Name of cars vs count of cars

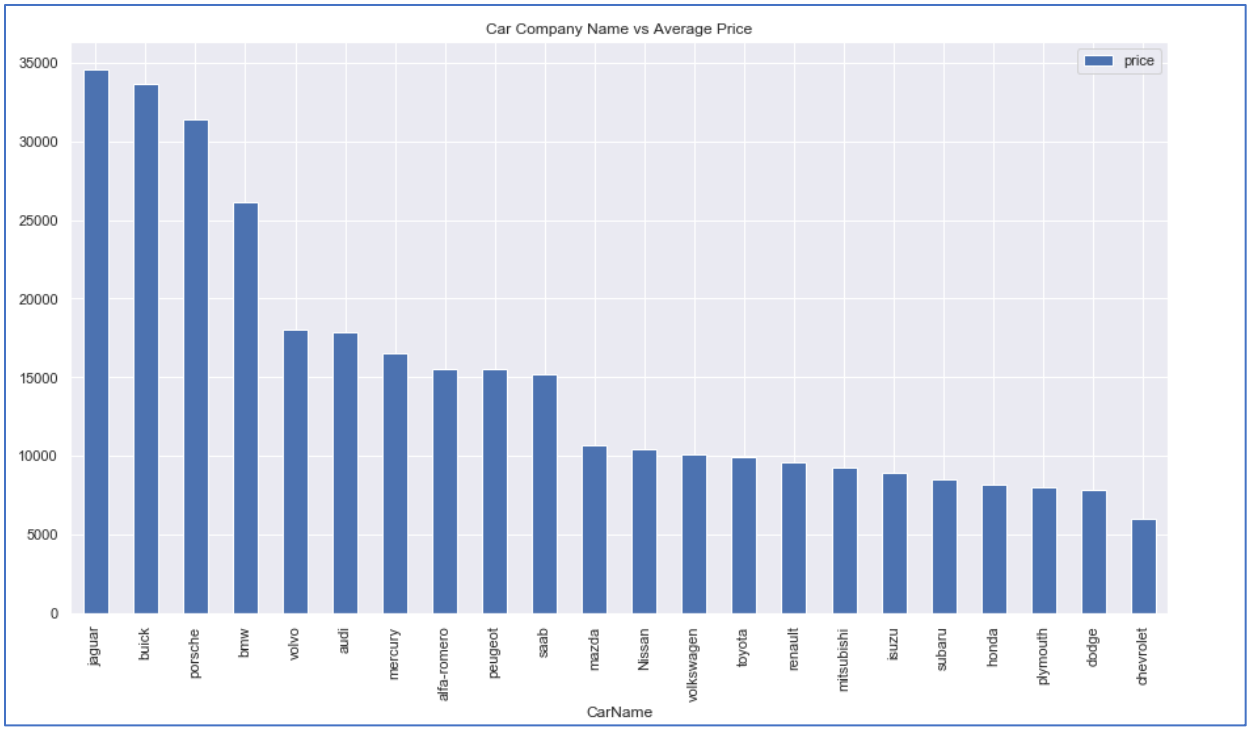


Fig: Car Name vs Average Price

DATA PREPARATION

Data preparation is the process of cleaning and transforming raw data prior to processing and analysis.

We need to get the dummy variables for the categorical features and store it in new variables. Then we add the results to the original data frame and drop the original categorical variables (since we now have the dummy variables).

FEATURE SCALING

```
In [38]: scaler = preprocessing.StandardScaler()

In [39]: sig_num_col = ['wheelbase','carlength','carwidth','curbweight','enginesize','boreratio','horsepower','citympg','highwaympg','pric

In [40]: # Apply scaler() to all the columns except the 'dummy' variables
         import warnings
         warnings.filterwarnings("ignore")

         df_train[sig_num_col] = scaler.fit_transform(df_train[sig_num_col])
```

SPLITTING DATA SET INTO TRAINING AND TESTING SETS

- The dataset is divided into training and testing sets.
- The training set to test set ratio is 70:30.
- The linear regression model is trained on the training set and then it is tested on the testing set on which it is used to predict the values.

```
In [36]: # We specify this so that the train and test data set always have the same rows, respectively
         # We divide the df into 70/30 ratio

         np.random.seed(0)
         df_train, df_test = train_test_split(df_auto, train_size = 0.7, test_size = 0.3, random_state = 100)
```

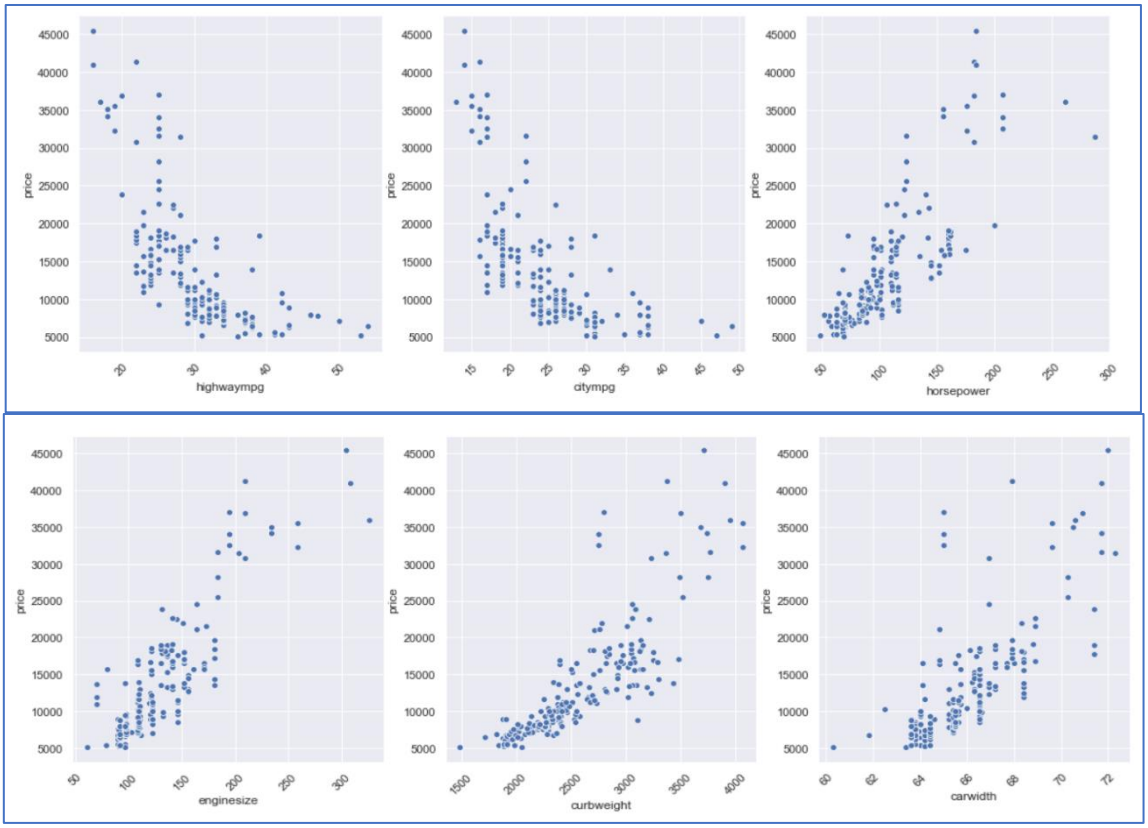


Fig: Scatter plots of few dependent variables with the independent variable (price)

FEATURE SELECTION

- Process of identifying and selecting those features in your data that contribute most to the prediction variable or output in which you are interested.
- Having too many irrelevant features in your data can decrease the accuracy of the models.

>RECURSIVE FEATURE ELIMINATION (RFE)

- It works by recursively removing attributes and building a model on those attributes that remain.
- It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.
- It is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached.
- Features are ranked by the model's attributes, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model.

>BACKWARD SELECTION

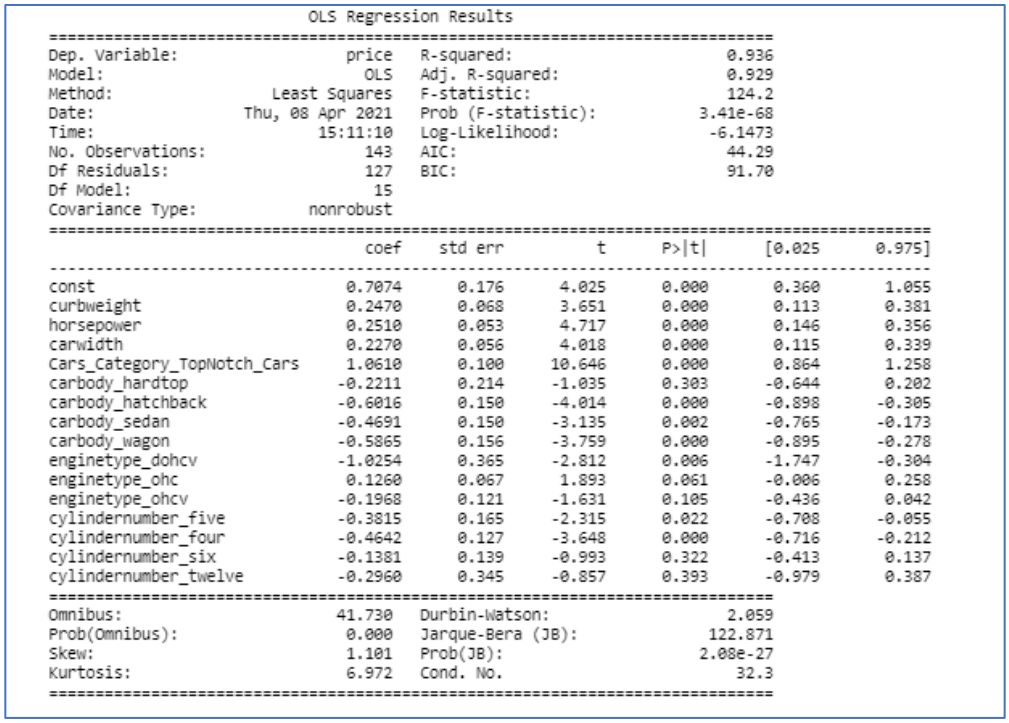
- Backward Selection is the method to fit the full model and then remove terms one at a time, starting with the term which has the highest p-value. After removing a feature, the model is fit again (the fit statistic and the p-values change) and the process is repeated again.

BUILDING A MULTIPLE REGRESSION MODEL USING RFE AND BACKWARD SELECTION

In Recursive Feature Elimination we eliminate the weakest variable(s) (curbweight, carbody_hardtop, enginetype_ohc, enginetype_ohcv, cylindernumber_five, cylindernumber_four, cylindernumber_six, cylindernumber_twelve) from a set of 24 independent variables using the Variance Inflation Factor (VIF) and the P-value test.

```
In [46]: # Running RFE with the output number of the variable equal to 15
         lm = LinearRegression()
         lm.fit(X_train, y_train)

         rfe = RFE(lm, 15)
         rfe = rfe.fit(X_train, y_train)

In [47]: list(zip(X_train.columns,rfe.support_,rfe.ranking_))

Out[47]: [('wheelbase', False, 5),
          ('curbweight', True, 1),
          ('enginesize', False, 11),
          ('boreratio', False, 8),
          ('horsepower', True, 1),
          ('citympg', False, 17),
          ('highwaympg', False, 4),
          ('carlength', False, 12),
          ('carwidth', True, 1),
          ('Cars_Category_Medium_Range', False, 3),
          ('Cars_Category_TopNotch_Cars', True, 1),
          ('fueltype_gas', False, 9),
          ('aspiration_turbo', False, 13),
          ('carbody_hardtop', True, 1),
          ('carbody_hatchback', True, 1),
          ('carbody_sedan', True, 1),
          ('carbody_wagon', True, 1),
          ('drivewheel_fwd', False, 6),
          ('drivewheel_rwd', False, 10),
          ('enginetype_dohcv', True, 1),
          ('enginetype_l', False, 15),
          ('enginetype_ohc', True, 1),
          ('enginetype_ohcf', False, 2),
          ('enginetype_ohcv', True, 1),
          ('enginetype_rotor', False, 16),
          ('cylindernumber_five', True, 1),
          ('cylindernumber_four', True, 1),
          ('cylindernumber_six', True, 1),
          ('cylindernumber_three', False, 7),
          ('cylindernumber_twelve', True, 1),
          ('cylindernumber_two', False, 14)]
```

```
In [48]: # Selecting the variables which are in support

         col_sup = X_train.columns[rfe.support_]
         col_sup

Out[48]: Index(['curbweight', 'horsepower', 'carwidth', 'Cars_Category_TopNotch_Cars',
                'carbody_hardtop', 'carbody_hatchback', 'carbody_sedan',
                'carbody_wagon', 'enginetype_dohcv', 'enginetype_ohc',
                'enginetype_ohcv', 'cylindernumber_five', 'cylindernumber_four',
                'cylindernumber_six', 'cylindernumber_twelve'],
               dtype='object')
```



OLS Regression Results

Dep. Variable:	price	R-squared:	0.936
Model:	OLS	Adj. R-squared:	0.929
Method:	Least Squares	F-statistic:	124.2
Date:	Thu, 08 Apr 2021	Prob (F-statistic):	3.41e-68
Time:	15:11:10	Log-Likelihood:	-6.1473
No. Observations:	143	AIC:	44.29
Df Residuals:	127	BIC:	91.70
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.7074	0.176	4.025	0.000	0.360	1.055
curbweight	0.2470	0.068	3.651	0.000	0.113	0.381
horsepower	0.2510	0.053	4.717	0.000	0.146	0.356
carwidth	0.2270	0.056	4.018	0.000	0.115	0.339
Cars_Category_TopNotch_Cars	1.0610	0.100	10.646	0.000	0.864	1.258
carbody_hardtop	-0.2211	0.214	-1.035	0.303	-0.644	0.202
carbody_hatchback	-0.6016	0.150	-4.014	0.000	-0.898	-0.305
carbody_sedan	-0.4691	0.150	-3.135	0.002	-0.765	-0.173
carbody_wagon	-0.5865	0.156	-3.759	0.000	-0.895	-0.278
enginetype_dohcv	-1.0254	0.365	-2.812	0.006	-1.747	-0.304
enginetype_ohc	0.1260	0.067	1.893	0.061	-0.006	0.258
enginetype_ohcv	-0.1968	0.121	-1.631	0.105	-0.436	0.042
cylindernumber_five	-0.3815	0.165	-2.315	0.022	-0.708	-0.055
cylindernumber_four	-0.4642	0.127	-3.648	0.000	-0.716	-0.212
cylindernumber_six	-0.1381	0.139	-0.993	0.322	-0.413	0.137
cylindernumber_twelve	-0.2960	0.345	-0.857	0.393	-0.979	0.387

Omnibus:	41.730	Durbin-Watson:	2.059
Prob(Omnibus):	0.000	Jarque-Bera (JB):	122.871
Skew:	1.101	Prob(JB):	2.08e-27
Kurtosis:	6.972	Cond. No.	32.3

	Features	VIF
12	cylindernumber_four	15.9200
6	carbody_sedan	9.7300
0	curbweight	9.0600
2	carwidth	6.2400
5	carbody_hatchback	6.1500
9	enginetype_ohc	5.9700
1	horsepower	5.6100
13	cylindernumber_six	4.7300
7	carbody_wagon	3.4800
11	cylindernumber_five	2.8200
3	Cars_Category_TopNotch_Cars	2.1700
8	enginetype_dohcv	1.8400
14	cylindernumber_twelve	1.6600
10	enginetype_ohcv	1.6300
4	carbody_hardtop	1.4500

If VIF is greater than 5 then we remove it from the model

- Dropping cylindernumber_twelve beacuse its p-value is 0.393 and we want p-value less than 0.05, hence rebuilding the model. Similarly, we have dropped cylindernumber_six, carbody_hardtop, enginetype_oh, cylindernumber_five, enginetype_ohcv, curbweight since each has a p-value greater than 0.05 and we are only including p-values less than 0.05. We drop cylindernumber_four beacuse its VIF is 5.66 and we want VIF less than 5 and hence rebuilding the model.

FINAL MODEL

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.918
Model:                            OLS   Adj. R-squared:                  0.914
Method:                 Least Squares   F-statistic:                     215.9
Date:                Wed, 07 Apr 2021   Prob (F-statistic):           4.70e-70
Time:                        23:37:50   Log-Likelihood:                -24.089
No. Observations:                 143   AIC:                             64.18
Df Residuals:                     135   BIC:                             87.88
Df Model:                           7
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
const                            0.2440      0.116      2.096      0.038       0.014       0.474
horsepower                       0.3599      0.039      9.228      0.000       0.283       0.437
carwidth                         0.3652      0.037      9.944      0.000       0.293       0.438
Cars_Category_TopNotch_Cars      1.2895      0.095     13.559      0.000       1.101       1.478
carbody_hatchback               -0.4859      0.122     -3.976      0.000      -0.728      -0.244
carbody_sedan                   -0.3518      0.121     -2.896      0.004      -0.592      -0.112
carbody_wagon                   -0.4023      0.135     -2.974      0.003      -0.670      -0.135
enginetype_dohcv                -1.4450      0.326     -4.435      0.000      -2.089      -0.801
==============================================================================
Omnibus:                       43.937   Durbin-Watson:                   2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              127.746
Skew:                           1.171   Prob(JB):                     1.82e-28
Kurtosis:                       6.995   Cond. No.                         17.5
==============================================================================
```

	Features	VIF
0	horsepower	2.4500
1	carwidth	2.1200
2	Cars_Category_TopNotch_Cars	1.7000
4	carbody_sedan	1.2200
6	enginetype_dohcv	1.2200
3	carbody_hatchback	1.1000
5	carbody_wagon	1.0200

ASSUMPTION TESTING

Multiple linear regression analysis makes several key assumptions:

- There must be a linear relationship between the outcome variable and the independent variables.
- Multivariate Normality–Multiple regression assumes that the residuals are normally distributed.
- No Multicollinearity—Multiple regression assumes that the independent variables are not highly correlated with each other.
- Homoscedasticity–This assumption states that the variance of error terms are similar across the values of the independent variables

>NORMALITY TESTING (ASSUMPTION OF NORMALITY)

>> ANDERSON-DARLING TEST AND HISTOGRAM

- It is a statistical test of whether or not a dataset comes from a certain probability distribution.
- The test involves calculating the Anderson-Darling statistic.
- The two hypotheses for the Anderson-Darling test for the normal distribution are given below:

  H0 : The data follows the normal distribution

  H1 : The data does not follow the normal distribution
- In many cases we can determine a p-value for the Anderson-Darling statistic and use that value to help you determine if the test is significant are not.
- If the p value is low (e.g., <=0.05), you conclude that the data do not follow the normal distribution.

In [69]:
```
def normal_errors_assumption(model, features, label, p_value_thresh=0.05):
    from statsmodels.stats.diagnostic import normal_ad
    print('Assumption 1: The error terms are normally distributed', '\n')
    # Calculating residuals for the Anderson-Darling test
    df_results = calculate_residuals(model, features, label)
    print('Using the Anderson-Darling test for normal distribution')
    # Performing the test on the residuals
    p_value = normal_ad(df_results['Residuals'])[1]
    print('p-value from the test (below 0.05 generally means non-normal) :', p_value)
    # Reporting the normality of the residuals
    if p_value < p_value_thresh:
        print('Residuals are not normally distributed')
    else:
        print('Residuals are normally distributed')
    # Plotting the residuals distribution
    plt.subplots(figsize=(12, 6))
    plt.title('Distribution of Residuals')
    sns.distplot(df_results['Residuals'],bins=20)
    plt.show()
    print()
    if p_value > p_value_thresh:
        print('Assumption satisfied')
    else:
        print('Assumption not satisfied')
        print()
        print('Confidence intervals will likely be affected')
        print('Try performing nonlinear transformations on variables')
```

In [70]:
```
normal_errors_assumption(lm_rfe8,X_train_rfe8c,y_train)
```

```
Assumption 1: The error terms are normally distributed

Using the Anderson-Darling test for normal distribution
p-value from the test (below 0.05 generally means non-normal) : 0.003280088073501484
Residuals are not normally distributed
```

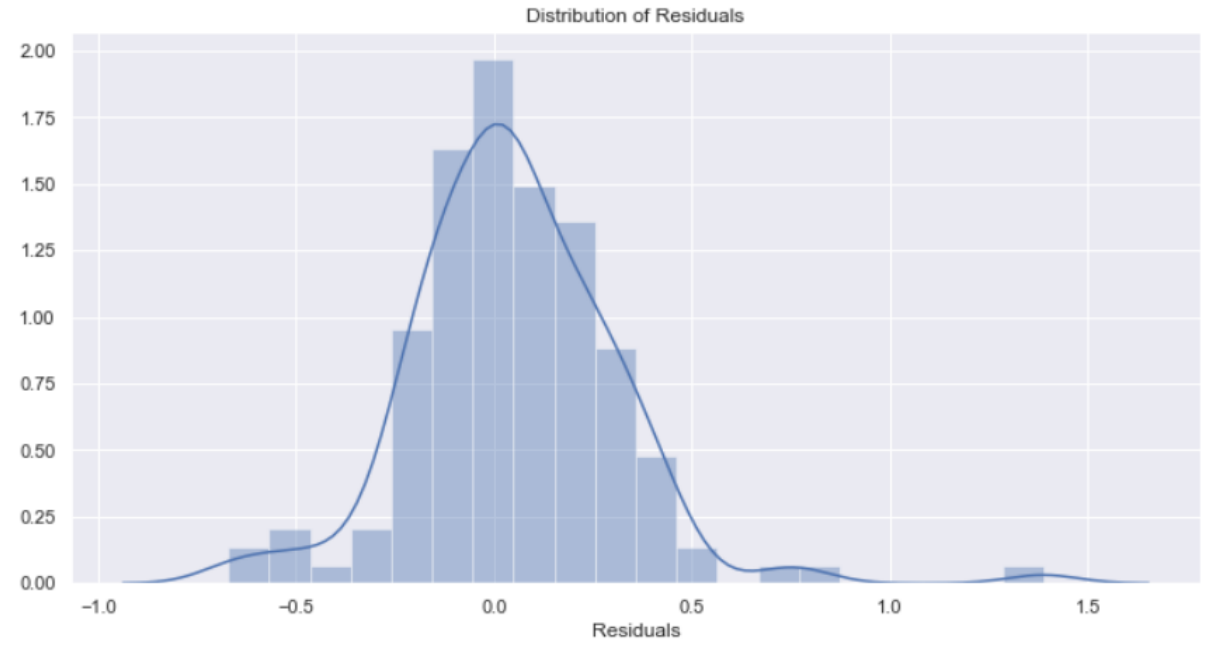


Fig: Anderson Darling Test and Histogram

>>SHAPIRO- WILK TEST

- The Shapiro-Wilk test is a way to tell if a random sample comes from a normal distribution.
- H0: distribution is normal.
  H1: distribution is not normal.
- The test gives you a W value; small values indicate your sample is not normally distributed (you can reject the null hypothesis that your population is normally distributed if your values are under a certain threshold)

```
In [94]: df_results = calculate_residuals(lm_rfe8,X_train_rfe8c,y_train)
         # Shapiro-Wilk Test
         from scipy.stats import shapiro
         stat, p = shapiro(df_results['Residuals'])
         print("SHAPIRO-WILK TEST")
         print('Statistic = ',stat,'\np-value = ', p)
         if p>0.05: print('Probably Normal')
         else: print('Probably Not Normal')

         SHAPIRO-WILK TEST
         Statistic =  0.9400341510772705
         p-value =  8.515859008184634e-06
         Probably Not Normal
```

Fig: Shapiro Wilk test

>>Q-Q PLOT

- Q-Q Plots (Quantile-Quantile plots) are plots of two quantiles against each other.
- The purpose of Q-Q plots is to find out if two sets of data come from the same distribution.
- A 45 degree angle is plotted on the Q-Q plot; if the two data sets come from a common distribution, the points will fall on that reference line.

Q-Q Plot

```
In [119]: ## Q-Q PLOT
from scipy import stats

sm.ProbPlot(df_results['Residuals']).qqplot(line='s');
plt.title('Q-Q plot');
```

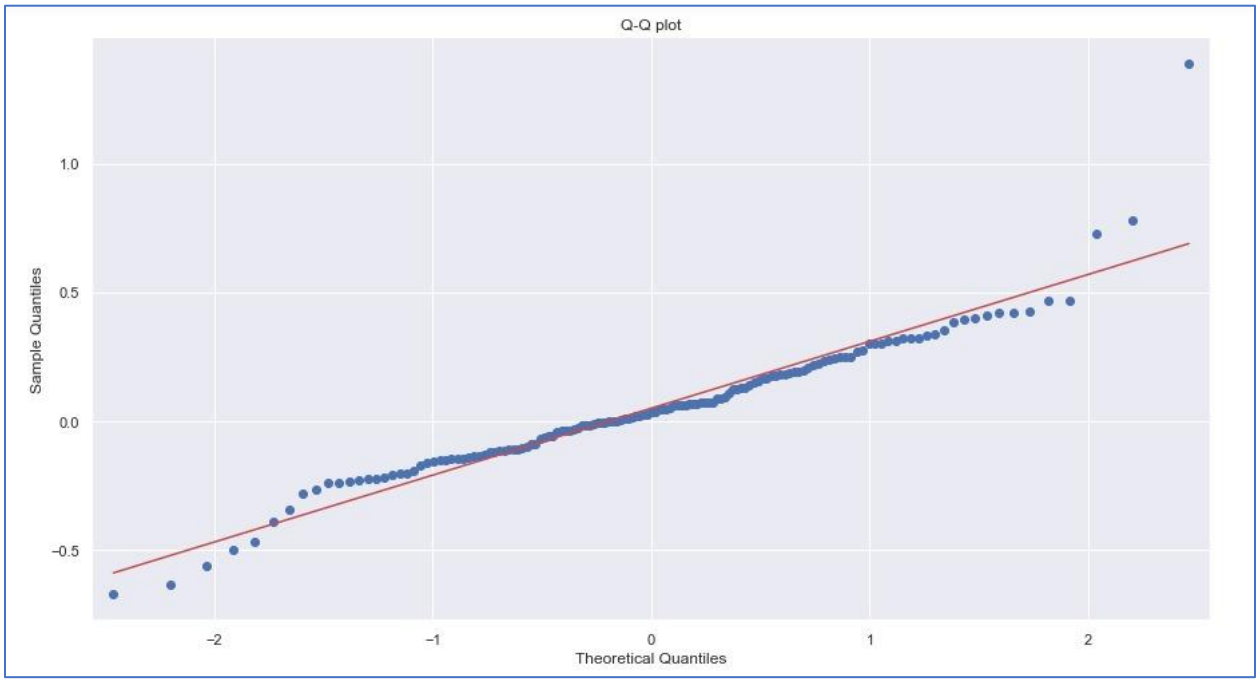


Fig: Q-Q Plot

>HOMOSCEDASTICITY OF RESIDUALS (ASSUMPTION OF HOMOSCEDASTICITY)

>>RESIDUAL PLOT

In [73]:
```
def homoscedasticity_assumption(model, features, label):
    """
    Homoscedasticity: Assumes that the errors exhibit constant variance
    """
    print('Assumption 2: Homoscedasticity of Error Terms', '\n')
    print('Residuals should have relative constant variance')
    # Calculating residuals for the plot
    df_results = calculate_residuals(model, features, label)
    # Plotting the residuals
    plt.subplots(figsize=(12, 6))
    ax = plt.subplot(111)  # To remove spines
    plt.scatter(x=df_results.index, y=df_results.Residuals, alpha=0.5)
    plt.plot(np.repeat(0, df_results.index.max()), color='darkorange', linestyle='--')
    ax.spines['right'].set_visible(False)  # Removing the right spine
    ax.spines['top'].set_visible(False)  # Removing the top spine
    plt.title('Residuals')
    plt.show()
```

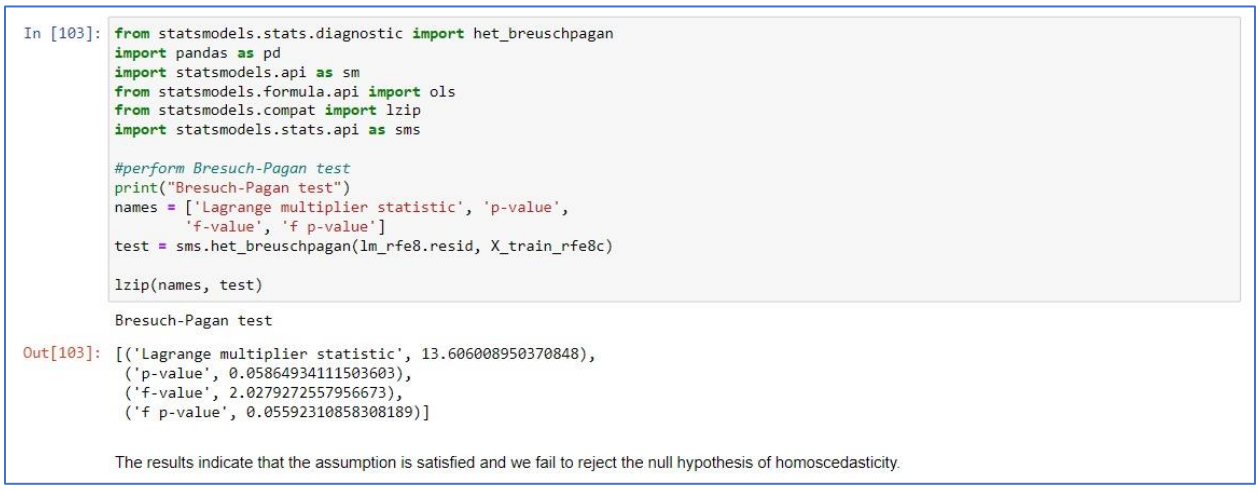
In [74]:
```
homoscedasticity_assumption(lm_rfe8,X_train_rfe8c,y_train)
```

```
Assumption 2: Homoscedasticity of Error Terms

Residuals should have relative constant variance
```

Fig: Residual Plot

>> BREUSCH-PAGAN TEST

- The Breusch-Pagan-Godfrey Test (or the Breusch-Pagan test) is a test for heteroscedasticity of errors in regression. Heteroscedasticity means “differently scattered”.
- Homoscedasticity in regression is an important assumption if the assumption is violated, you won’t be able to use regression analysis.
- Breusch-Pagan test measures how errors increase across the explanatory variable, Y. The test assumes the error variances are due to a linear function of one or more explanatory variables in the model. That means heteroscedasticity could still be present in your regression model, but those errors (if present) are not correlated with the Y-values.



```
In [103]: from statsmodels.stats.diagnostic import het_breuschpagan
          import pandas as pd
          import statsmodels.api as sm
          from statsmodels.formula.api import ols
          from statsmodels.compat import lzip
          import statsmodels.stats.api as sms

          #perform Bresuch-Pagan test
          print("Bresuch-Pagan test")
          names = ['Lagrange multiplier statistic', 'p-value',
                   'f-value', 'f p-value']
          test = sms.het_breuschpagan(lm_rfe8.resid, X_train_rfe8c)

          lzip(names, test)

          Bresuch-Pagan test

Out[103]: [('Lagrange multiplier statistic', 13.606008950370848),
           ('p-value', 0.05864934111503603),
           ('f-value', 2.0279272557956673),
           ('f p-value', 0.05592310858308189)]

          The results indicate that the assumption is satisfied and we fail to reject the null hypothesis of homoscedasticity.
```

Fig: Breusch-Pagan Test

>LINEARITY TESTS (ASSUMPTION OF LINEARITY)

>> SCATTER PLOT

```
In [77]: def linear_assumption(model, features, label):
             print('Assumption 3: Linear Relationship between the Target and the Feature', '\n')
             print('Checking with a scatter plot of actual vs. predicted.',
                   'Predictions should follow the diagonal line.')
             # Calculating residuals for the plot
             df_results = calculate_residuals(model, features, label)
             # Plotting the actual vs predicted values
             sns.lmplot(x='Actual', y='Predicted', data=df_results, fit_reg=False, size=7)
             # Plotting the diagonal line
             line_coords = np.arange(df_results.min().min(), df_results.max().max())
             plt.plot(line_coords, line_coords,  # X and y points
                      color='darkorange', linestyle='--')
             plt.title('Actual vs. Predicted')
             plt.show()

In [78]: linear_assumption(lm_rfe8,X_train_rfe8c,y_train)
```

```
Assumption 3: Linear Relationship between the Target and the Feature

Checking with a scatter plot of actual vs. predicted. Predictions should follow the diagonal line.
```

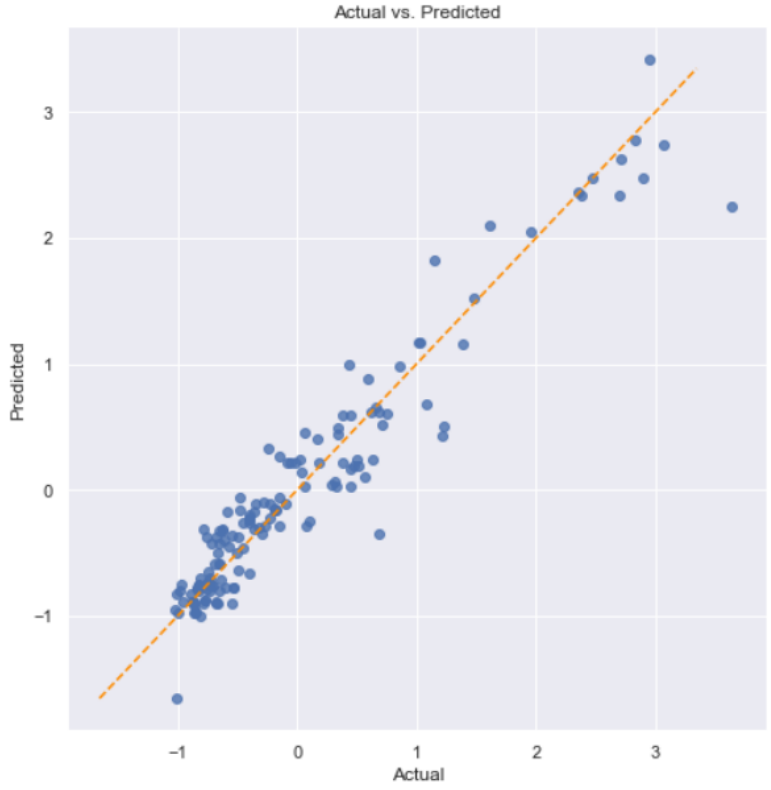


Fig: Scatter Plot

>> RAMSEY RESET TEST

- The Ramsey Reset test is a way of testing whether there exists some significant non linear relationships when you have built a Linear Regression Model.
- After you run a simple linear regression model , this test runs another regression model with adding powers of the predicted Y as independent variables with the original x variables .
- Now it uses the F-Test to compare the two models.

In [79]:
```
import numpy as np
import pandas as pd
import statsmodels.regression.linear_model as rg
import statsmodels.tools.tools as ct
import statsmodels.stats.diagnostic as dg

reset = dg.linear_reset(lm_rfe8, power = 2, test_type='exog', use_f = True)

print("Ramsey Reset test: f stat:", np.round(reset.fvalue,6))
print("Ramsey Reset test: p value:", np.round(reset.pvalue,6))
```

```
Ramsey Reset test: f stat: [[2.653512]]
Ramsey Reset test: p value: 0.07413
```

p>0.05. Hence no non-linearity present.

Fig: Ramsey RESET Test

>MULTICOLLINEARITY

>> VARIANCE INFLATION FACTOR TEST (VIF TEST)

- Variance inflation factor (VIF) is a measure of the amount of multicollinearity in a set of multiple regression variables.
- VIF for a regression model variable is equal to the ratio of the overall model variance to the variance of a model that includes only that single independent variable. This ratio is calculated for each independent variable.
- A high VIF indicates that the associated independent variable is highly collinear with the other variables in the model.

In [80]:
```
print('Assumption 4: Little to no multicollinearity among predictors')

# Create a dataframe that will contain the names of all the feature variables and their respective VIFs
vif = pd.DataFrame()
vif['Features'] = X_train_rfe8.columns
vif['VIF'] = [variance_inflation_factor(X_train_rfe8.values, i) for i in range(X_train_rfe8.shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

```
Assumption 4: Little to no multicollinearity among predictors
```

Out[80]:

	Features	VIF
0	horsepower	2.4500
1	carwidth	2.1200
2	Cars_Category_TopNotch_Cars	1.7000
4	carbody_sedan	1.2200
6	enginetype_dohcv	1.2200
3	carbody_hatchback	1.1000
5	carbody_wagon	1.0200

Fig: VIF Test

>> HEAT MAP

```
In [81]: # Heat Map
         plt.figure(figsize = (20, 20))
         sns.heatmap(X_train_rfe8.corr(), cmap="coolwarm")
         plt.show()
```

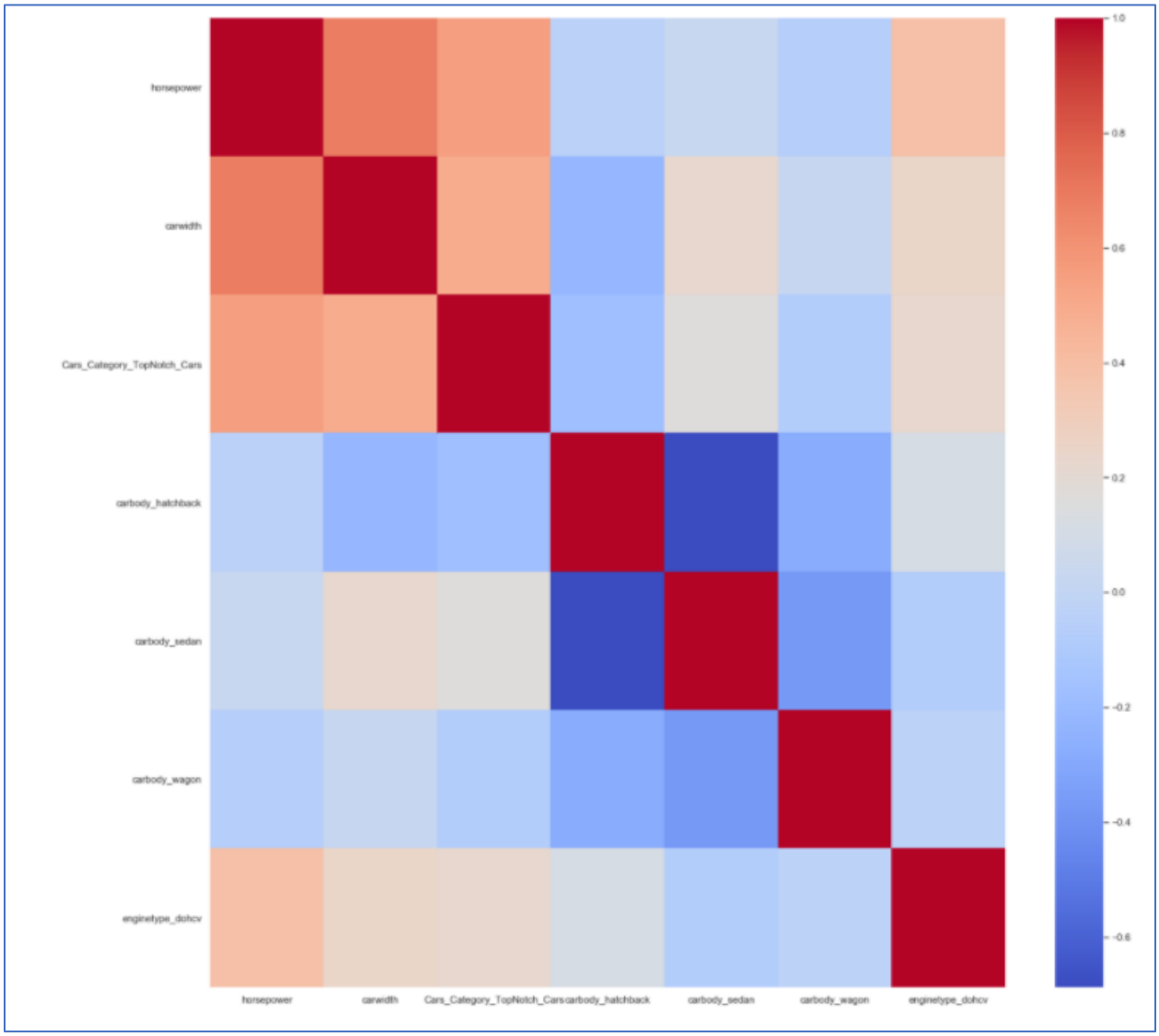


Fig: Heat Map

>> CORRELATION MATRIX

In [82]:
```
## Regressor Correlation Matrix
corr = X_train_rfe8.corr()
corr
```

Out[82]:

	horsepower	carwidth	Cars_Category_TopNotch_Cars	carbody_hatchback	carbody_sedan	carbody_wagon	enginetype_dohcv
horsepower	1.0000	0.6851	0.5544	-0.0354	0.0267	-0.0632	0.3932
carwidth	0.6851	1.0000	0.4967	-0.2241	0.2227	0.0191	0.2457
Cars_Category_TopNotch_Cars	0.5544	0.4967	1.0000	-0.1741	0.1694	-0.0801	0.2285
carbody_hatchback	-0.0354	-0.2241	-0.1741	1.0000	-0.6875	-0.2826	0.1162
carbody_sedan	0.0267	0.2227	0.1694	-0.6875	1.0000	-0.3727	-0.0799
carbody_wagon	-0.0632	0.0191	-0.0801	-0.2826	-0.3727	1.0000	-0.0328
enginetype_dohcv	0.3932	0.2457	0.2285	0.1162	-0.0799	-0.0328	1.0000

Assupmtion 4 satisfied.

Fig: Correlation Matrix

MODEL EVALUATION

In [90]:
```
# Plotting y_test and y_pred to understand the spread.
fig = plt.figure()
plt.scatter(y_test,y_pred2)
fig.suptitle('y_test vs y_pred2', fontsize=20)
plt.xlabel('y_test ', fontsize=18)
plt.ylabel('y_pred2', fontsize=16)
```

Out[90]: Text(0, 0.5, 'y_pred2')

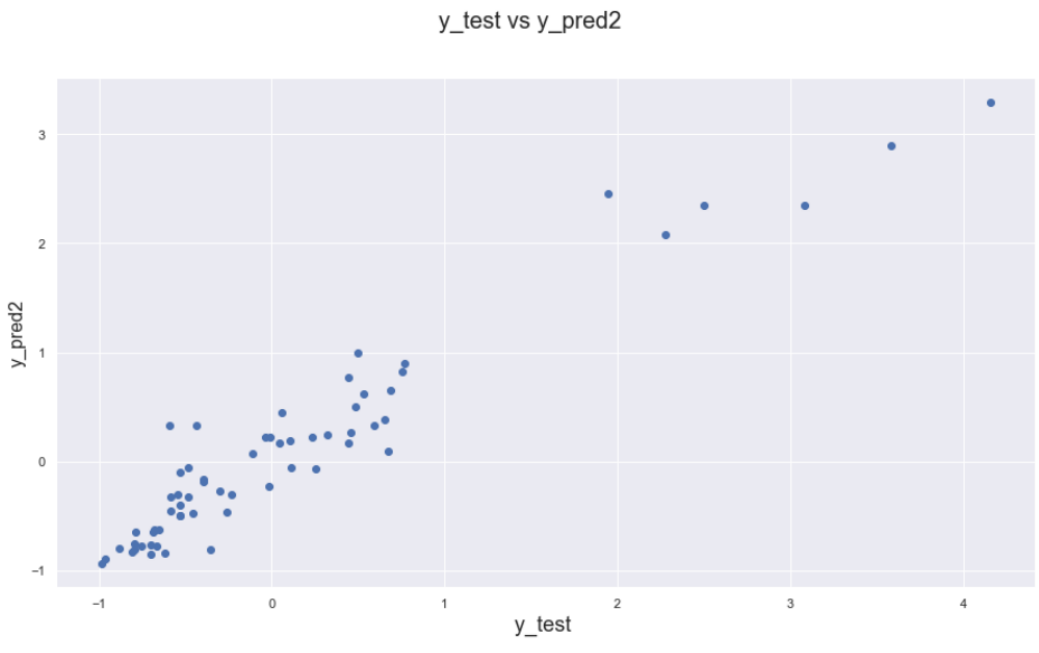


Fig: Scatter Plot for y_test vs y_pred2

R-SQUARED SCORE

```
In [91]: r2_score(y_test, y_pred2)
Out[91]: 0.915385372286651
```

The R^2 score of Training set is 0.918 and Test set is 0.915 which is quite close. Hence, we can say that our model is good enough to predict the Car prices using below predictor variables.

- Horsepower
- Carwidth
- Cars_Category_TopNotch_Cars
- Carbody_hatchback
- Enginetype_dohcv
- Carbody_sedan
- Carbody_wagon

EQUATION OF LINE TO PREDICT THE CAR PRICES VALUES

CAR PRICES = 0.2440 + 0.3599x(horsepower) + 0.3652x(carwidth) + 1.2895x(cars_category_top_notch_cars) – 0.4859x(car_body_hachback) – 1.4450x(engine_type_dohcv) – 0.3518x(car_body_sedan) – 0.4023x(car_body_wagon)

MODEL CONCLUSIONS

- R-squared and Adjusted R-squared - 0.918 and 0.915 - 90% variance explained.
- F-stats and Prob(F-stats) (overall model fit) - 215.9 and 4.70e-70(approx. 0.0) - Model fit is significant and explained 90% variance is just not by chance.

- P-values for all the coefficients seem to be less than the significance level of 0.05. - meaning that all the predictors are statistically significant.
- CLOSING STATEMENT: Model is good enough to predict the car prices which explains the variance of data upto 90% and the model is significant.