Data Science II with python (Class notes)

STAT 303-2

Arvind Krishna

1/3/23

Table of contents

Preface

These are class notes for the course STAT303-2. This is not the course text-book. You are required to read the relevant sections of the book as mentioned on the course website.

The course notes are currently being written, and will continue to being developed as the course progresses (just like the course textbook last quarter). Please report any typos / mistakes / inconsistencies / issues with the class notes / class presentations in your comments here. Thank you!

Part I

Linear regression

1 Simple Linear Regression

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
```

Develop a simple linear regression model that predicts car price based on engine size. Datasets to be used: *Car_features_train.csv, Car_prices_train.csv*

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

```
#Using the ols function to create an ols object. 'ols' stands for 'Ordinary least squares'
ols_object = smf.ols(formula = 'price~engineSize', data = train)
```

```
#Using the fit() function of the 'ols' class to fit the model
model = ols_object.fit()
```

```
#Printing model summary which contains among other things, the model coefficients
model.summary()
```

Table 1.2: OLS Regression Results

Dep. Variable:	price	R-squared:	0.390
Model:	OLS	Adj. R-squared:	0.390
Method:	Least Squares	F-statistic:	3177.
Date:	Thu, 19 Jan 2023	Prob (F-statistic):	0.00
Time:	16:44:04	Log-Likelihood:	-53949.
No. Observations:	4960	AIC:	1.079e+05
Df Residuals:	4958	BIC:	1.079e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4122.0357	522.260	-7.893	0.000	-5145.896	-3098.176
engineSize	1.299e+04	230.450	56.361	0.000	1.25e+04	1.34e+04

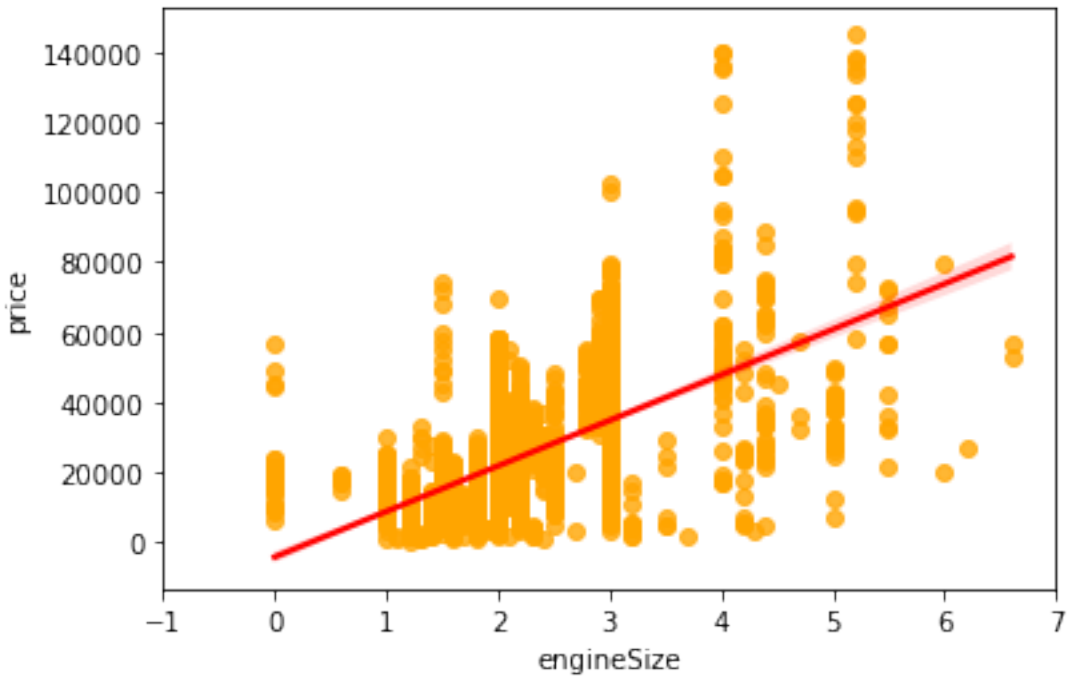
Omnibus:	1271.986	Durbin-Watson:	0.517
Prob(Omnibus):	0.000	Jarque-Bera (JB):	6490.719
Skew:	1.137	Prob(JB):	0.00
Kurtosis:	8.122	Cond. No.	7.64

The model equation is: car price = -4122.0357 + 12990 * engineSize

Visualize the regression line

```
sns.regplot(x = 'engineSize', y = 'price', data = train, color = 'orange',line_kws={"color
plt.xlim(-1,7)
#Note that some of the engineSize values are 0. They are incorrect, and should ideally be
```

```
(-1.0, 7.0)
```



Predict the car price for the cars in the test dataset. Datasets to be used: *Car_features_test.csv, Car_prices_test.csv*

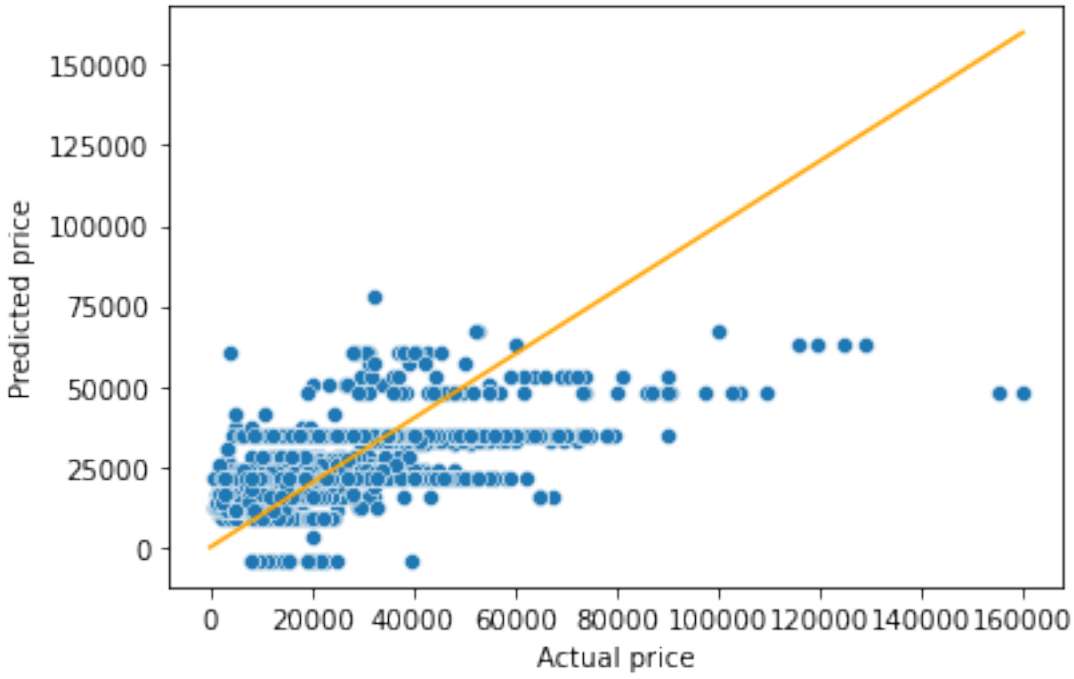
```
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
```

```
#Using the predict() function associated with the 'model' object to make predictions of ca
pred_price = model.predict(testf)#Note that the predict() function finds the predictor 'en
```

Make a visualization that compares the predicted car prices with the actual car prices

```
sns.scatterplot(x = testp.price, y = pred_price)
#In case of a perfect prediction, all the points must lie on the line x = y.
sns.lineplot(x = [0,testp.price.max()], y = [0,testp.price.max()],color='orange') #Plottin
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
```

```
Text(0, 0.5, 'Predicted price')
```



The prediction doesn't look too good. This is because we are just using one predictor - engine size. We can probably improve the model by adding more predictors when we learn multiple linear regression.

What is the RMSE of the predicted car price?

```
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
12995.1064515487
```

The root mean squared error in predicting car price is around $13k.

What is the residual standard error based on the training data?

```
np.sqrt(model.mse_resid)
```

```
12810.109175214136
```

The residual standard error on the training data is close to the RMSE on the test data. This shows that the performance of the model on unknown data is comparable to its performance

on known data. This implies that the model is not overfitting, which is good! In case we overfit a model on the training data, its performance on unknown data is likely to be worse than that on the training data.

Find the confidence and prediction intervals of the predicted car price

```
#Using the get_prediction() function associated with the 'model' object to get the interva
intervals = model.get_prediction(testf)
```

```
#The function requires specifying alpha (probability of Type 1 error) instead of the confi
intervals.summary_frame(alpha=0.05)
```

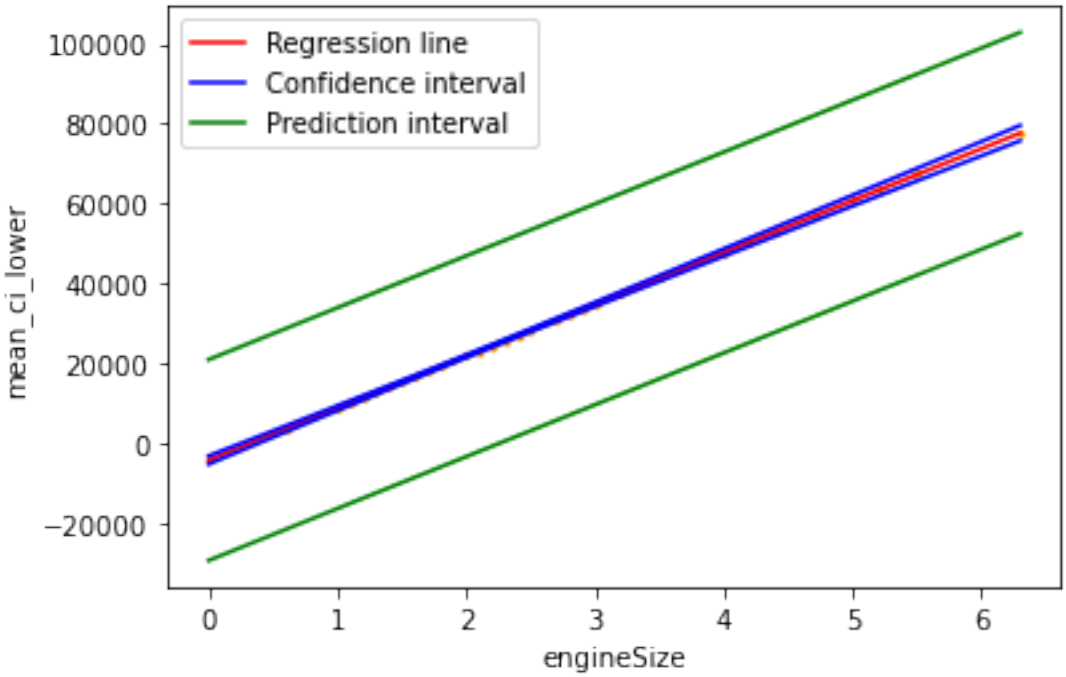
	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
1	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
2	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
3	8866.245277	316.580850	8245.606701	9486.883853	-16254.905974	33987.396528
4	47831.088340	468.949360	46911.740050	48750.436631	22700.782946	72961.393735
...	...	...	...	...	...	...
2667	47831.088340	468.949360	46911.740050	48750.436631	22700.782946	72961.393735
2668	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
2669	8866.245277	316.580850	8245.606701	9486.883853	-16254.905974	33987.396528
2670	21854.526298	184.135754	21493.538727	22215.513869	-3261.551421	46970.604017
2671	21854.526298	184.135754	21493.538727	22215.513869	-3261.551421	46970.604017

Show the regression line predicting car price based on engine size for test data. Also show the confidence and prediction intervals for the car price.

```
interval_table = intervals.summary_frame(alpha=0.05)
```

```
sns.scatterplot(x = testf.engineSize, y = pred_price,color = 'orange', s = 10)
sns.lineplot(x = testf.engineSize, y = pred_price, color = 'red')
sns.lineplot(x = testf.engineSize, y = interval_table.mean_ci_lower, color = 'blue')
sns.lineplot(x = testf.engineSize, y = interval_table.mean_ci_upper, color = 'blue',label=
sns.lineplot(x = testf.engineSize, y = interval_table.obs_ci_lower, color = 'green')
sns.lineplot(x = testf.engineSize, y = interval_table.obs_ci_upper, color = 'green')
plt.legend(labels=["Regression line","Confidence interval", "Prediction interval"])
```

```
<matplotlib.legend.Legend at 0x26a3a32c550>
```


Regression line
Confidence interval
Prediction interval
100000
80000
60000
40000
20000
0
−20000
mean_ci_lower
0
1
2
3
4
5
6
engineSize

2 Multiple Linear Regression

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
```

Develop a multiple linear regression model that predicts car price based on engine size, year, mileage, and mpg. Datasets to be used: *Car_features_train.csv, Car_prices_train.csv*

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

```
#Using the ols function to create an ols object. 'ols' stands for 'Ordinary least squares'
ols_object = smf.ols(formula = 'price~year+mileage+mpg+engineSize', data = train)
model = ols_object.fit()
model.summary()
```

Table 2.2: OLS Regression Results

Dep. Variable:	price	R-squared:	0.660
Model:	OLS	Adj. R-squared:	0.660
Method:	Least Squares	F-statistic:	2410.

Table 2.2: OLS Regression Results

Date:	Tue, 27 Dec 2022	Prob (F-statistic):	0.00
Time:	01:07:25	Log-Likelihood:	-52497.
No. Observations:	4960	AIC:	1.050e+05
Df Residuals:	4955	BIC:	1.050e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.661e+06	1.49e+05	-24.593	0.000	-3.95e+06	-3.37e+06
year	1817.7366	73.751	24.647	0.000	1673.151	1962.322
mileage	-0.1474	0.009	-16.817	0.000	-0.165	-0.130
mpg	-79.3126	9.338	-8.493	0.000	-97.620	-61.006
engineSize	1.218e+04	189.969	64.107	0.000	1.18e+04	1.26e+04

Omnibus:	2450.973	Durbin-Watson:	0.541
Prob(Omnibus):	0.000	Jarque-Bera (JB):	31060.548
Skew:	2.045	Prob(JB):	0.00
Kurtosis:	14.557	Cond. No.	3.83e+07

The model equation is: estimated car price = -3.661e6 + 1818 * year -0.15 * mileage - 79.31 * mpg + 12180 * engineSize

Predict the car price for the cars in the test dataset. Datasets to be used: *Car_features_test.csv, Car_prices_test.csv*

```
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
```

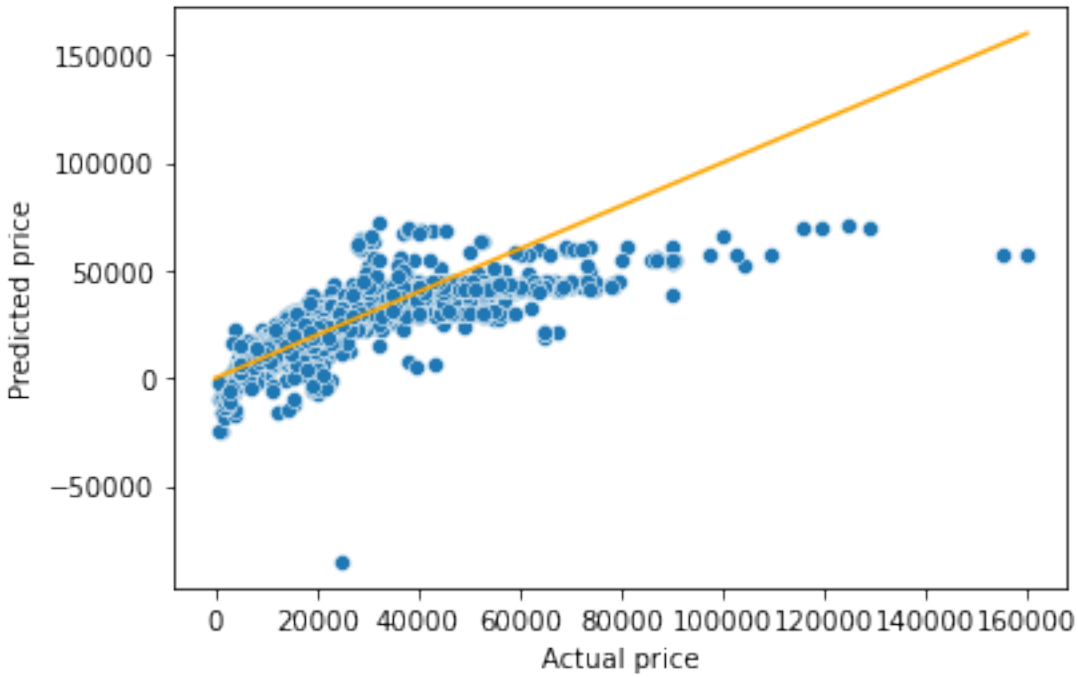
```
#Using the predict() function associated with the 'model' object to make predictions of ca
pred_price = model.predict(testf)#Note that the predict() function finds the predictor 'en
```

Make a visualization that compares the predicted car prices with the actual car prices

```
sns.scatterplot(x = testp.price, y = pred_price)
#In case of a perfect prediction, all the points must lie on the line x = y.
sns.lineplot(x = [0,testp.price.max()], y = [0,testp.price.max()],color='orange') #Plottin
```

```
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
```

```
Text(0, 0.5, 'Predicted price')
```



The prediction looks better as compared to the one with simple linear regression. This is because we have four predictors to help explain the variation in car price, instead of just one in the case of simple linear regression. Also, all the predictors have a significant relationship with price as evident from their p-values. Thus, all four of them are contributing in explaining the variation. Note the higher values of R2 as compared to the one in the case of simple linear regression.

What is the RMSE of the predicted car price?

```
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
9956.82497993548
```

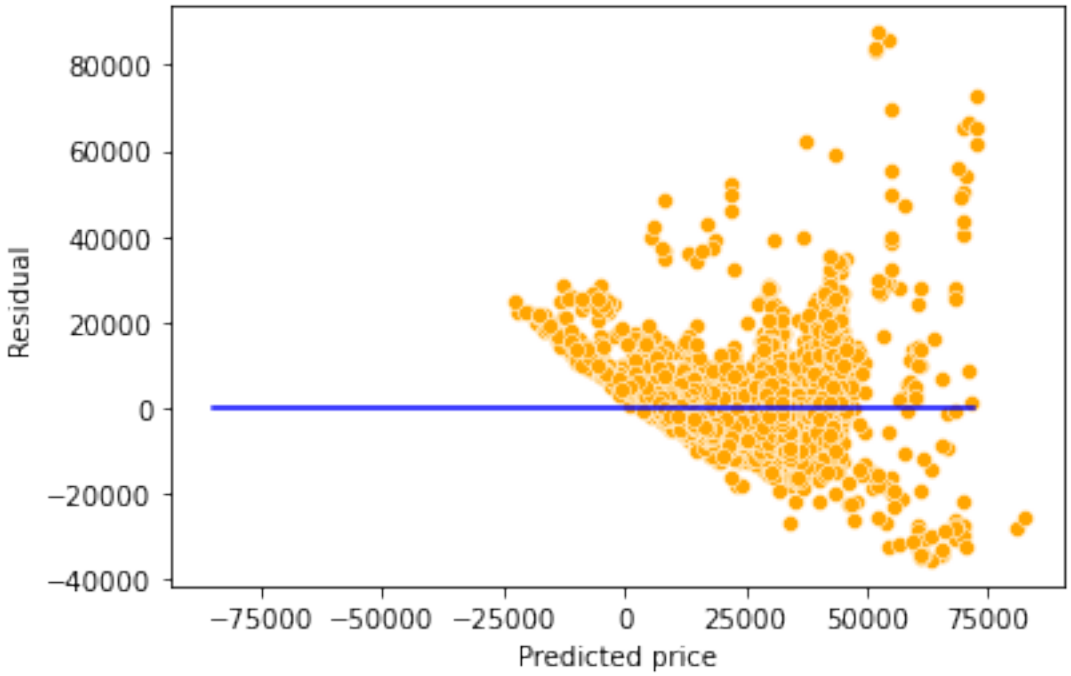
What is the residual standard error based on the training data?

```
np.sqrt(model.mse_resid)
```

```
9563.74782917604
```

```
sns.scatterplot(x = model.fittedvalues, y=model.resid,color = 'orange')
sns.lineplot(x = [pred_price.min(),pred_price.max()],y = [0,0],color = 'blue')
plt.xlabel('Predicted price')
plt.ylabel('Residual')
```

```
Text(0, 0.5, 'Residual')
```



Will the explained variation (R-squared) in car price always increase if we add a variable?

Should we keep on adding variables as long as the explained variation (R-squared) is increasing?

```
#Using the ols function to create an ols object. 'ols' stands for 'Ordinary least squares'
np.random.seed(1)
```

```
train['rand_col'] = np.random.rand(train.shape[0])
ols_object = smf.ols(formula = 'price~year+mileage+mpg+engineSize+rand_col', data = train)
model = ols_object.fit()
model.summary()
```

Table 2.5: OLS Regression Results

Dep. Variable:	price	R-squared:	0.661
Model:	OLS	Adj. R-squared:	0.660
Method:	Least Squares	F-statistic:	1928.
Date:	Tue, 27 Dec 2022	Prob (F-statistic):	0.00
Time:	01:07:38	Log-Likelihood:	-52497.
No. Observations:	4960	AIC:	1.050e+05
Df Residuals:	4954	BIC:	1.050e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.662e+06	1.49e+05	-24.600	0.000	-3.95e+06	-3.37e+06
year	1818.1672	73.753	24.652	0.000	1673.578	1962.756
mileage	-0.1474	0.009	-16.809	0.000	-0.165	-0.130
mpg	-79.2837	9.338	-8.490	0.000	-97.591	-60.976
engineSize	1.218e+04	189.972	64.109	0.000	1.18e+04	1.26e+04
rand_col	451.1226	471.897	0.956	0.339	-474.004	1376.249

Omnibus:	2451.728	Durbin-Watson:	0.541
Prob(Omnibus):	0.000	Jarque-Bera (JB):	31040.331
Skew:	2.046	Prob(JB):	0.00
Kurtosis:	14.552	Cond. No.	3.83e+07

Adding a variable with random values to the model (*rand_col*) increased the explained variation (R-squared). This is because the model has one more parameter to tune to reduce the residual squared error (RSS). However, the p-value of *rand_col* suggests that its coefficient is zero. Thus, using the model with *rand_col* may give poorer performance on unknown data, as compared to the model without *rand_col*. This implies that it is not a good idea to blindly add variables in the model to increase R-squared.

3 Variable interactions and transformations

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
```

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

Until now, we have assumed that the association between a predictor X_j and response Y does not depend on the value of other predictors. For example, the multiple linear regression model that we developed in Chapter 2 assumes that the average increase in price associated with a unit increase in engineSize is always $12,180, regardless of the value of other predictors. However, this assumption may be incorrect.

3.0.1 Variable interaction between continuous predictors

We can relax this assumption by considering another predictor, called an interaction term. Let us assume that the average increase in `price` associated with a one-unit increase in `engineSize` depends on the model `year` of the car. In other words, there is an interaction between `engineSize` and `year`. This interaction can be included as a predictor, which is the

product of `engineSize` and `year`. *Note that there are several possible interactions that we can consider. Here the interaction between* ***`engineSize`*** *and* ***`year`*** *is just an example.*

```
#Considering interaction between engineSize and year
ols_object = smf.ols(formula = 'price~year*engineSize+mileage+mpg', data = train)
model = ols_object.fit()
model.summary()
```

Table 3.2: OLS Regression Results

Dep. Variable:	price	R-squared:	0.682
Model:	OLS	Adj. R-squared:	0.681
Method:	Least Squares	F-statistic:	2121.
Date:	Tue, 24 Jan 2023	Prob (F-statistic):	0.00
Time:	15:28:11	Log-Likelihood:	-52338.
No. Observations:	4960	AIC:	1.047e+05
Df Residuals:	4954	BIC:	1.047e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	5.606e+05	2.74e+05	2.048	0.041	2.4e+04	1.1e+06
year	-275.3833	135.695	-2.029	0.042	-541.405	-9.361
engineSize	-1.796e+06	9.97e+04	-18.019	0.000	-1.99e+06	-1.6e+06
year:engineSize	896.7687	49.431	18.142	0.000	799.861	993.676
mileage	-0.1525	0.008	-17.954	0.000	-0.169	-0.136
mpg	-84.3417	9.048	-9.322	0.000	-102.079	-66.604

Omnibus:	2330.413	Durbin-Watson:	0.524
Prob(Omnibus):	0.000	Jarque-Bera (JB):	29977.437
Skew:	1.908	Prob(JB):	0.00
Kurtosis:	14.423	Cond. No.	7.66e+07

Note that the R-squared has increased as compared to the model in Chapter 2 since we added a predictor.

The model equation is:

$$price = \beta_0 + \beta_1 * year + \beta_2 * engineSize + \beta_3 * (year * engineSize) + \beta 4 * mileage + \beta_5 * mpg, \quad (3.1)$$

or

$$price = \beta_0 + \beta_1 * year + (\beta_2 + \beta_3 * year) * engineSize + \beta 4 * mileage + \beta_5 * mpg, \quad (3.2)$$

or

$$price = \beta_0 + \beta_1 * year + \tilde{\beta} * engineSize + \beta 4 * mileage + \beta_5 * mpg, \quad (3.3)$$

Since $\tilde{\beta}$ is a function of `year`, the association between `engineSize` and `price` is no longer a constant. A change in the value of `year` will change the association between `price` and `engineSize`.

Substituting the values of the coefficients:

$$price = 5.606e5 - 275.3833 * year + (-1.796e6 + 896.7687 * year) * engineSize - 0.1525 * mileage - 84.3417 * mpg \quad (3.4)$$

Thus, for cars launched in the year 2010, the average increase in price for one liter increase in engine size is -1.796e6 + 896.7687 * 2010 ≈ \$6,500, assuming all the other predictors are constant. However, for cars launched in the year 2020, the average increase in price for one liter increase in engine size is -1.796e6 + 896.7687*2020 ≈ \$15,500 , assuming all the other predictors are constant.

Similarly, the equation can be re-arranged as:

$$price = 5.606e5 + (-275.3833 + 896.7687 * engineSize) * year - 1.796e6 * engineSize - 0.1525 * mileage - 84.3417 * mp \quad (3.5)$$

Thus, for cars with an engine size of 2 litres, the average increase in price for a one year newer model is -275.3833+896.7687 * 2 ≈ \$1500, assuming all the other predictors are constant. However, for cars with an engine size of 3 litres, the average increase in price for a one year newer model is -275.3833+896.7687 * 3 ≈ \$2400, assuming all the other predictors are constant.

```
#Computing the RMSE of the model with the interaction term
pred_price = model.predict(testf)
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
9423.598872501092
```

Note that the RMSE is lower than that of the model in Chapter 2. This is because the interaction term between `engineSize` and `year` is significant and relaxes the assumption of constant association between price and engine size, and between price and year. This added flexibility makes the model better fit the data. Caution: Too much flexibility may lead to overfitting!

Note that interaction terms corresponding to other variable pairs, and higher order interaction terms (such as those containing 3 or 4 variables) may also be significant and improve the model fit & thereby the prediction accuracy of the model.

3.0.2 Including qualitative predictors in the model

Let us develop a model for predicting `price` based on `engineSize` and the qualitative predictor `transmission`.

```
#checking the distribution of values of transmission
train.transmission.value_counts()
```

```
Manual       1948
Automatic    1660
Semi-Auto    1351
Other           1
Name: transmission, dtype: int64
```

Note that the *Other* category of the variable *transmission* contains only a single observation, which is likely to be insufficient to train the model. We'll remove that observation from the training data. Another option may be to combine the observation in the *Other* category with the nearest category, and keep it in the data.

```
train_updated = train[train.transmission!='Other']
```

```
ols_object = smf.ols(formula = 'price~engineSize+transmission', data = train_updated)
model = ols_object.fit()
model.summary()
```

Table 3.5: OLS Regression Results

Dep. Variable:	price	R-squared:	0.459
Model:	OLS	Adj. R-squared:	0.458
Method:	Least Squares	F-statistic:	1400.

Table 3.5: OLS Regression Results

Date:	Tue, 24 Jan 2023	Prob (F-statistic):	0.00
Time:	15:28:21	Log-Likelihood:	-53644.
No. Observations:	4959	AIC:	1.073e+05
Df Residuals:	4955	BIC:	1.073e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3042.6765	661.190	4.602	0.000	1746.451	4338.902
transmission[T.Manual]	-6770.6165	442.116	-15.314	0.000	-7637.360	-5903.873
transmission[T.Semi-Auto]	4994.3112	442.989	11.274	0.000	4125.857	5862.765
engineSize	1.023e+04	247.485	41.323	0.000	9741.581	1.07e+04

Omnibus:	1575.518	Durbin-Watson:	0.579
Prob(Omnibus):	0.000	Jarque-Bera (JB):	11006.609
Skew:	1.334	Prob(JB):	0.00
Kurtosis:	9.793	Cond. No.	11.4

Note that there is no coefficient for the *Automatic* level of the variable `Transmission`. If a car doesn't have *Manual* or *Semi-Automatic* transmission, then it has an *Automatic* transmission. Thus, the coefficient of *Automatic* will be redundant, and the dummy variable corresponding to *Automatic* transmission is dropped from the model.

The level of the categorical variable that is dropped from the model is called the baseline level. Here *Automatic* transmission is the baseline level. The coefficients of other levels of `transmission` should be interpreted with respect to the baseline level.

Q: Interpret the intercept term

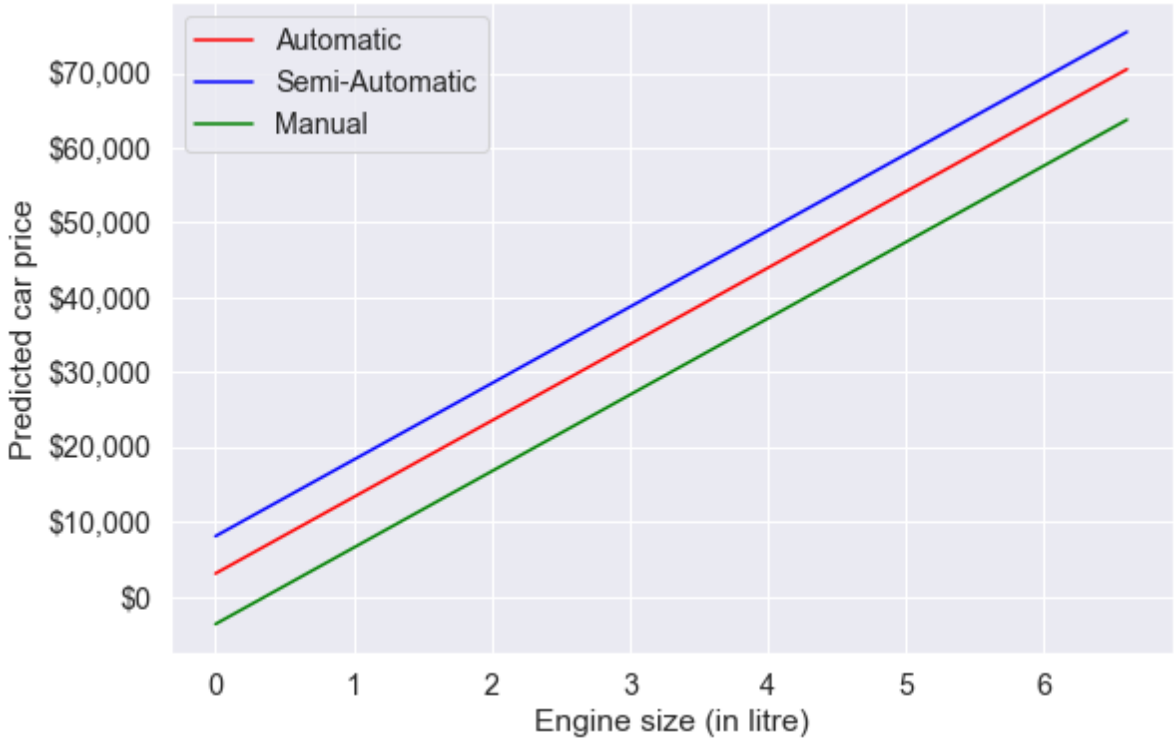
Ans: For the hypothetical scenario of a car with zero engine size and *Automatic* transmission, the estimated mean car price is $\approx$ \$3042.

Q: Interpret the coefficient of `transmission[T.Manual]`

Ans: The estimated mean price of a car with manual transmission is $\approx$ \$6770 less than that of a car with *Automatic* transmission.

Let us visualize the developed model.

```
#Visualizing the developed model
plt.rcParams["figure.figsize"] = (9,6)
sns.set(font_scale = 1.3)
x = np.linspace(train_updated.engineSize.min(),train_updated.engineSize.max(),100)
ax = sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept'], color
sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept']+model.param
sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept']+model.param
plt.legend(labels=["Automatic","Semi-Automatic", "Manual"])
plt.xlabel('Engine size (in litre)')
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
```



Based on the developed model, for a given engine size, the car with a semi-automatic transmission is estimated to be the most expensive on average, while the car with a manual transmission is estimated to be the least expensive on average.

Changing the baseline level: By default, the baseline level is chosen as the one that comes first if the levels are arranged in alphabetical order. However, you can change the baseline level by specifying one explicitly.

Internally, statsmodels uses the patsy package to convert formulas and data to the matrices that are used in model fitting. You may refer to this section in the patsy documentation to specify a particular level of the categorical variable as the baseline.

For example, suppose we wish to change the baseline level to *Manual* transmission. We can specify this in the formula as follows:

```
ols_object = smf.ols(formula = 'price~engineSize+C(transmission, Treatment("Manual"))', da
model = ols_object.fit()
model.summary()
```

Table 3.8: OLS Regression Results

Dep. Variable:	price	R-squared:	0.459
Model:	OLS	Adj. R-squared:	0.458
Method:	Least Squares	F-statistic:	1400.
Date:	Tue, 24 Jan 2023	Prob (F-statistic):	0.00
Time:	15:28:39	Log-Likelihood:	-53644.
No. Observations:	4959	AIC:	1.073e+05
Df Residuals:	4955	BIC:	1.073e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.9
Intercept	-3727.9400	492.917	-7.563	0.000	-4694.275	-27
C(transmission, Treatment("Manual"))[T.Automatic]	6770.6165	442.116	15.314	0.000	5903.873	763
C(transmission, Treatment("Manual"))[T.Semi-Auto]	1.176e+04	473.110	24.867	0.000	1.08e+04	1.2
engineSize	1.023e+04	247.485	41.323	0.000	9741.581	1.0

Omnibus:	1575.518	Durbin-Watson:	0.579
Prob(Omnibus):	0.000	Jarque-Bera (JB):	11006.609
Skew:	1.334	Prob(JB):	0.00
Kurtosis:	9.793	Cond. No.	8.62

3.0.3 Including qualitative predictors and their interaction with continuous predictors in the model

Note that the qualitative predictor leads to fitting 3 parallel lines to the data, as there are 3 categories.

However, note that we have made the constant association assumption. The fact that the lines are parallel means that the average increase in car price for one litre increase in engine size does not depend on the type of transmission. This represents a potentially serious limitation of the model, since in fact a change in engine size may have a very different association on the price of an automatic car versus a semi-automatic or manual car.

This limitation can be addressed by adding an interaction variable, which is the product of `engineSize` and the dummy variables for semi-automatic and manual transmissions.

```
#Using the ols function to create an ols object. 'ols' stands for 'Ordinary least squares'
ols_object = smf.ols(formula = 'price~engineSize*transmission', data = train_updated)
model = ols_object.fit()
model.summary()
```

Table 3.11: OLS Regression Results

Dep. Variable:	price	R-squared:	0.479
Model:	OLS	Adj. R-squared:	0.478
Method:	Least Squares	F-statistic:	909.9
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	22:55:55	Log-Likelihood:	-53550.
No. Observations:	4959	AIC:	1.071e+05
Df Residuals:	4953	BIC:	1.072e+05
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3754.7238	895.221	4.194	0.000	1999.695	5509.753
transmission[T.Manual]	1768.5856	1294.071	1.367	0.172	-768.366	4305.538
transmission[T.Semi-Auto]	-5282.7164	1416.472	-3.729	0.000	-8059.628	-2505.805
engineSize	9928.6082	354.511	28.006	0.000	9233.610	1.06e+04
engineSize:transmission[T.Manual]	-5285.9059	646.175	-8.180	0.000	-6552.695	-4019.117
engineSize:transmission[T.Semi-Auto]	4162.2428	552.597	7.532	0.000	3078.908	5245.578

Omnibus:	1379.846	Durbin-Watson:	0.622
Prob(Omnibus):	0.000	Jarque-Bera (JB):	9799.471
Skew:	1.139	Prob(JB):	0.00
Kurtosis:	9.499	Cond. No.	30.8

The model equation for the model with interactions is:

Automatic transmission: $price = 3754.7238 + 9928.6082 * engineSize$,
Semi-Automatic transmission: $price = 3754.7238 + 9928.6082 * engineSize + (-5282.7164 + 4162.2428 * engineSize)$,
Manual transmission: $price = 3754.7238 + 9928.6082 * engineSize + (1768.5856 - 5285.9059 * engineSize)$, or

Automatic transmission: $price = 3754.7238 + 9928.6082 * engineSize$,
Semi-Automatic transmission: $price = -1527 + 7046 * engineSize$,
Manual transmission: $price = 5523 + 4642 * engineSize$,

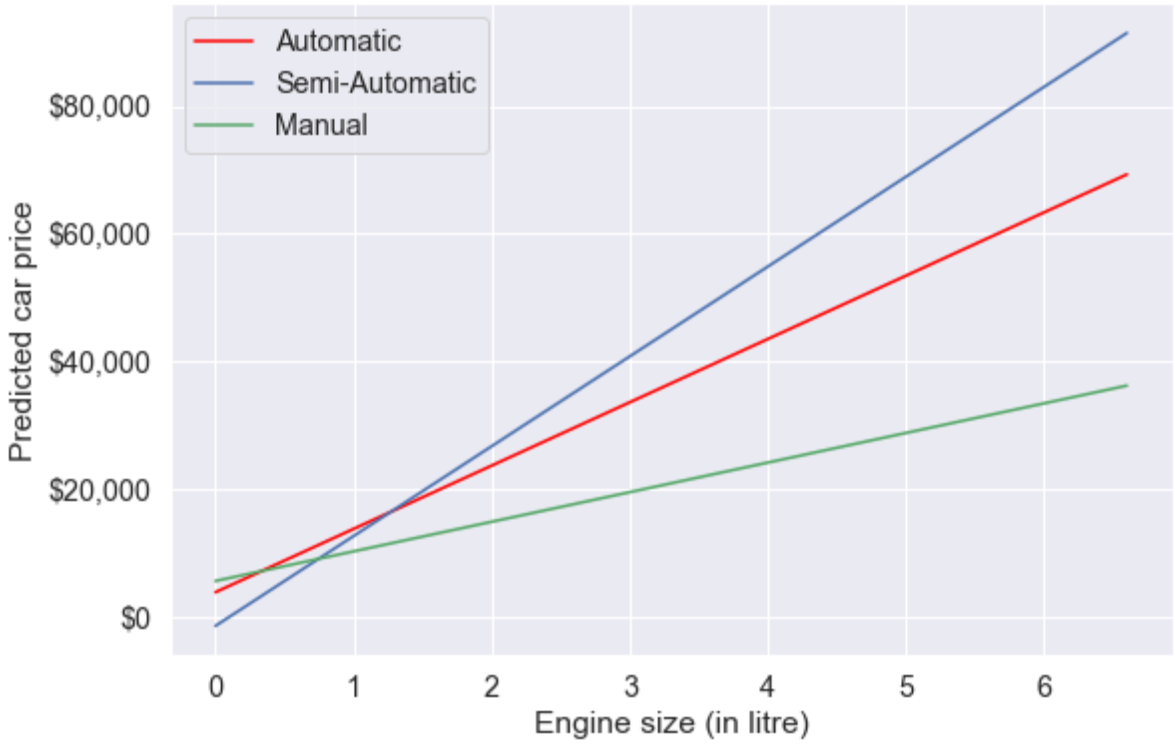
Q: Interpret the coefficient of manual tranmission, i.e., the coefficient of `transmission[T.Manual]`.

A: For a given engine size, the estimated mean `price` of a car with *Manual* transmission is $\approx$ \$1768 more than the estimated mean `price` of a car with *Automatic* transmission.

Q: Interpret the coefficient of the interaction between engine size and manual transmission, i.e., the coefficient of `engineSize:transmission[T.Manual]`.

A: For a unit (or a litre) increase in `engineSize` , the increase in estimated mean `price` of a car with *Manual* transmission is $\approx$ \$5285 less than the increase in estimated mean `price` of a car with *Automatic* transmission.

```
#Visualizing the developed model with interaction terms
plt.rcParams["figure.figsize"] = (9,6)
sns.set(font_scale = 1.3)
x = np.linspace(train_updated.engineSize.min(),train_updated.engineSize.max(),100)
ax = sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept'], label
plt.plot(x, (model.params['engineSize']+model.params['engineSize:transmission[T.Semi-Auto]
plt.plot(x, (model.params['engineSize']+model.params['engineSize:transmission[T.Manual]'])
plt.legend(loc='upper left')
plt.xlabel('Engine size (in litre)')
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
```



Note the interaction term adds flexibility to the model.

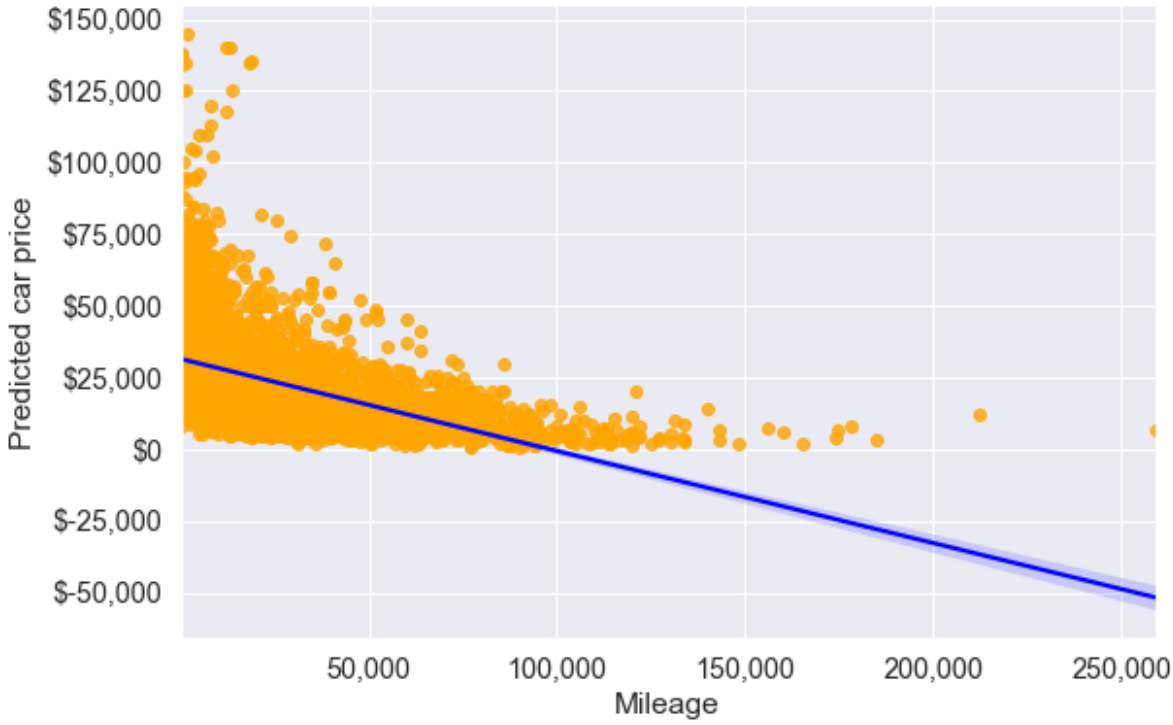
The slope of the regression line for semi-automatic cars is the largest. This suggests that increase in engine size is associated with a higher increase in car price for semi-automatic cars, as compared to other cars.

3.1 Variable transformations

So far we have considered only a linear relationship between the predictors and the response. However, the relationship may be non-linear.

Consider the regression plot of `price` on `mileage`.

```
ax = sns.regplot(x = train_updated.mileage, y =train_updated.price,color = 'orange', line_
plt.xlabel('Mileage')
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('{x:,.0f}')
```




```
#R-squared of the model with just mileage
model = smf.ols('price~mileage', data = train_updated).fit()
model.rsquared
```

```
0.22928048993376182
```

From the first scatterplot, we see that the relationship between `price` and `mileage` doesn't seem to be linear, as the points do not lie on a straight line. Also, we see the regression line (or the curve), which is the best fit line doesn't seem to fit the points well. However, `price` on average seems to decrease with `mileage`, albeit in a non-linear manner.

3.1.1 Quadratic transformation

So, we guess that if we model price as a quadratic function of `mileage`, the model may better fit the points (or the curve may better fit the points). Let us transform the predictor `mileage` to include $mileage^2$ (i.e., perform a quadratic transformation on the predictor).

```
#Including mileage squared as a predictor and developing the model
ols_object = smf.ols(formula = 'price~mileage+I(mileage**2)', data = train_updated)
model = ols_object.fit()
model.summary()
```

Table 3.14: OLS Regression Results

Dep. Variable:	price	R-squared:	0.271
Model:	OLS	Adj. R-squared:	0.271
Method:	Least Squares	F-statistic:	920.6
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	23:26:05	Log-Likelihood:	-54382.
No. Observations:	4959	AIC:	1.088e+05
Df Residuals:	4956	BIC:	1.088e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.44e+04	332.710	103.382	0.000	3.37e+04	3.5e+04
mileage	-0.5662	0.017	-33.940	0.000	-0.599	-0.534
I(mileage ** 2)	2.629e-06	1.56e-07	16.813	0.000	2.32e-06	2.94e-06

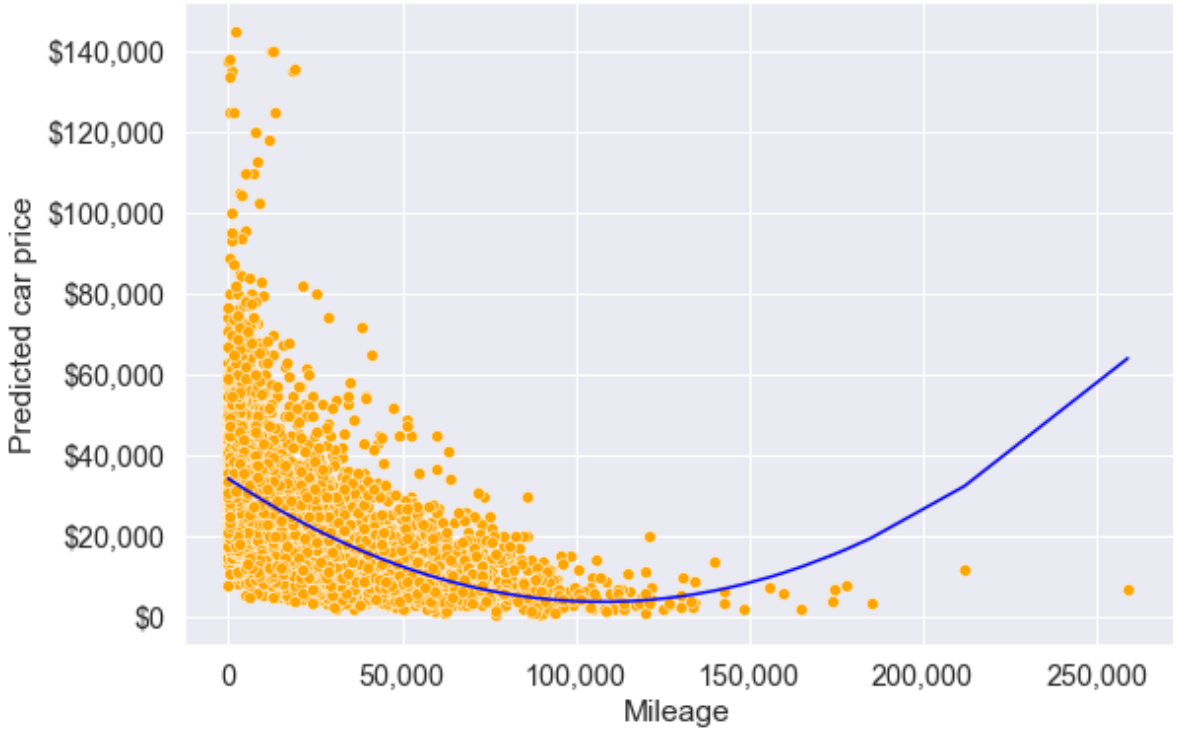
Omnibus:	2362.973	Durbin-Watson:	0.325
Prob(Omnibus):	0.000	Jarque-Bera (JB):	22427.952
Skew:	2.052	Prob(JB):	0.00
Kurtosis:	12.576	Cond. No.	4.81e+09

Note that in the formula specified within the `ols()` function, the `I()` operator isolates or insulates the contents within I(…) from the regular formula operators. Without the `I()` operator, `mileage**2` will be treated as the interaction of `mileage` with itself, which is `mileage`. Thus, to add the square of `mileage` as a separate predictor, we need to use the `I()` operator.

Let us visualize the model fit with the quadratic transformation of the predictor - `mileage`.

```
#Visualizing the regression line with the model consisting of the quadratic transformation
pred_price = model.predict(train_updated)
ax = sns.scatterplot(x = 'mileage', y = 'price', data = train_updated, color = 'orange')
sns.lineplot(x = train_updated.mileage, y = pred_price, color = 'blue')
plt.xlabel('Mileage')
```

```
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('{x:,.0f}')
```



The above model seems to better fit the data (as compared to the model without transformation) at least upto mileage around 125,000. The R^2 of the model with the quadratic transformation of `mileage` is also higher than that of the model without transformation indicating a better fit.

3.1.2 Cubic transformation

Let us see if a cubic transformation of `mileage` can further improve the model fit.

```
#Including mileage squared and mileage cube as predictors and developing the model
ols_object = smf.ols(formula = 'price~mileage+I(mileage**2)+I(mileage**3)', data = train_u
model = ols_object.fit()
model.summary()
```

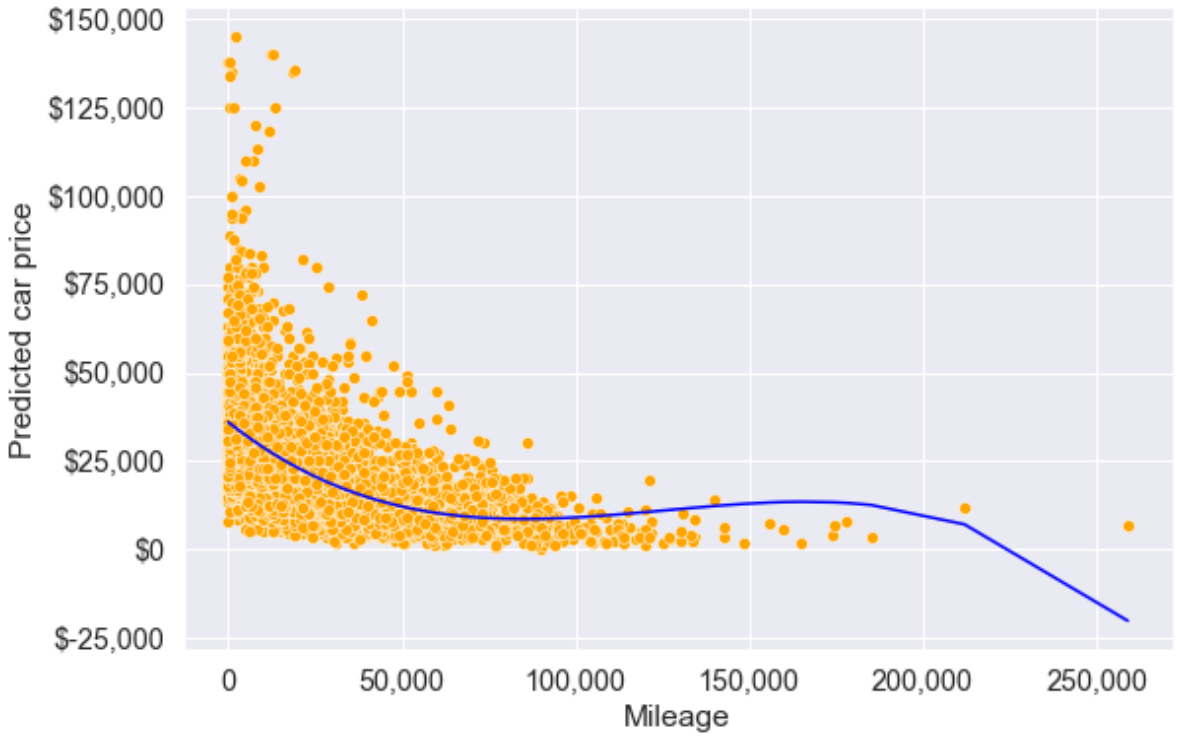
Table 3.17: OLS Regression Results

Dep. Variable:	price	R-squared:	0.283
Model:	OLS	Adj. R-squared:	0.283
Method:	Least Squares	F-statistic:	652.3
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	23:33:27	Log-Likelihood:	-54340.
No. Observations:	4959	AIC:	1.087e+05
Df Residuals:	4955	BIC:	1.087e+05
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	3.598e+04	371.926	96.727	0.000	3.52e+04	3.67e+04
mileage	-0.7742	0.028	-27.634	0.000	-0.829	-0.719
I(mileage ** 2)	6.875e-06	4.87e-07	14.119	0.000	5.92e-06	7.83e-06
I(mileage ** 3)	-1.823e-11	1.98e-12	-9.199	0.000	-2.21e-11	-1.43e-11

Omnibus:	2380.788	Durbin-Watson:	0.321
Prob(Omnibus):	0.000	Jarque-Bera (JB):	23039.307
Skew:	2.065	Prob(JB):	0.00
Kurtosis:	12.719	Cond. No.	7.73e+14

```
#Visualizing the model with the cubic transformation of mileage
pred_price = model.predict(train_updated)
ax = sns.scatterplot(x = 'mileage', y = 'price', data = train_updated, color = 'orange')
sns.lineplot(x = train_updated.mileage, y = pred_price, color = 'blue')
plt.xlabel('Mileage')
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('{x:,.0f}')
```



Note that the model fit with the cubic transformation of `mileage` seems slightly better as compared to the models with the quadratic transformation, and no transformation of mileage, for mileage up to 180k. However, the model should not be used to predict car prices of cars with a mileage higher than 180k.

Let's update the model created earlier (in the beginning of this chapter) to include the transformed predictor.

```
#Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'price~year*engineSize+mileage+mpg+I(mileage**2)', data = t
model = ols_object.fit()
model.summary()
```

Table 3.20: OLS Regression Results

Dep. Variable:	price	R-squared:	0.702
Model:	OLS	Adj. R-squared:	0.702
Method:	Least Squares	F-statistic:	1947.
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	23:42:13	Log-Likelihood:	-52162.

Table 3.20: OLS Regression Results

No. Observations:	4959	AIC:	1.043e+05
Df Residuals:	4952	BIC:	1.044e+05
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.53e+06	2.7e+05	5.671	0.000	1e+06	2.06e+06
year	-755.7419	133.791	-5.649	0.000	-1018.031	-493.453
engineSize	-2.022e+06	9.72e+04	-20.803	0.000	-2.21e+06	-1.83e+06
year:engineSize	1008.6993	48.196	20.929	0.000	914.215	1103.184
mileage	-0.3548	0.014	-25.973	0.000	-0.382	-0.328
mpg	-54.7450	8.896	-6.154	0.000	-72.185	-37.305
I(mileage ** 2)	1.926e-06	1.04e-07	18.536	0.000	1.72e-06	2.13e-06

Omnibus:	2355.448	Durbin-Watson:	0.562
Prob(Omnibus):	0.000	Jarque-Bera (JB):	38317.404
Skew:	1.857	Prob(JB):	0.00
Kurtosis:	16.101	Cond. No.	6.40e+12

Note that the R-squared has increased as compared to the model with just the interaction term.

```
#Computing RMSE on test data
pred_price = model.predict(testf)
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
9074.494088619422
```

Note that the prediction accuracy of the model has further increased, as the RMSE has reduced. The transformed predictor is statistically significant and provides additional flexibility to better capture the trend in the data, leading to an increase in prediction accuracy.

4 Model assumptions

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

Consider the model with interactions and transformation developed previously.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
```

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

```
ols_object = smf.ols(formula = 'price~(year+engineSize+mileage+mpg)**2+I(mileage**2)', dat
model = ols_object.fit()
model.summary()
```

Table 4.2: OLS Regression Results

Dep. Variable:	price	R-squared:	0.732
Model:	OLS	Adj. R-squared:	0.731
Method:	Least Squares	F-statistic:	1229.
Date:	Wed, 25 Jan 2023	Prob (F-statistic):	0.00
Time:	11:36:00	Log-Likelihood:	-51911.
No. Observations:	4960	AIC:	1.038e+05
Df Residuals:	4948	BIC:	1.039e+05
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.282e+06	7.14e+05	-1.795	0.073	-2.68e+06	1.18e+05
year	632.3954	353.865	1.787	0.074	-61.338	1326.128
engineSize	-1.465e+06	1.61e+05	-9.129	0.000	-1.78e+06	-1.15e+06
mileage	56.4581	3.811	14.815	0.000	48.987	63.929
mpg	-2.951e+04	9550.775	-3.089	0.002	-4.82e+04	-1.08e+04
year:engineSize	735.8074	79.532	9.252	0.000	579.890	891.725
year:mileage	-0.0281	0.002	-14.898	0.000	-0.032	-0.024
year:mpg	14.6915	4.731	3.105	0.002	5.417	23.966
engineSize:mileage	-0.0808	0.011	-7.143	0.000	-0.103	-0.059
engineSize:mpg	-120.5780	11.384	-10.592	0.000	-142.896	-98.260
mileage:mpg	0.0026	0.000	5.173	0.000	0.002	0.004
I(mileage ** 2)	3.495e-07	1.56e-07	2.236	0.025	4.31e-08	6.56e-07

Omnibus:	1958.631	Durbin-Watson:	0.542
Prob(Omnibus):	0.000	Jarque-Bera (JB):	44560.042
Skew:	1.349	Prob(JB):	0.00
Kurtosis:	17.434	Cond. No.	1.73e+13

```
np.sqrt(model.mse_resid)
```

```
8502.851955843495
```

```
pred_price = model.predict(testf)
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
8708.676318160937
```

```
#Computing MAE on test data
pred_price = model.predict(testf)
(np.abs(testp.price - pred_price)).mean()
```

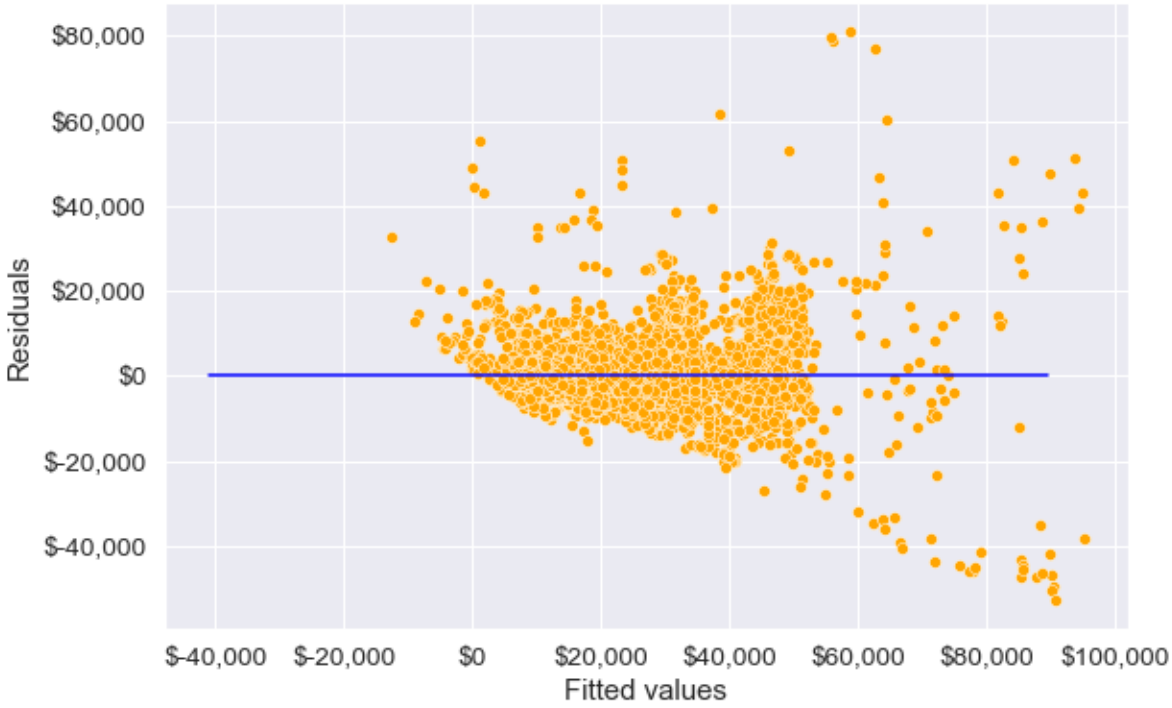
```
5395.006622253402
```

Let us check if this model satisfies the assumptions of the linear regression model

4.1 Non-linearity of data

We have assumed that there is a linear relationship between the predictors and the response. Residual plots, which are scatter plots of residuals vs fitted values, can be used to identify non-linearity. Fitted values are the values estimated by the model on training data, denoted by $\hat{y}_i$, and residuals are given by $e_i = y_i - \hat{y}_i$.

```
#Plotting residuals vs fitted values
plt.rcParams["figure.figsize"] = (9,6)
sns.set(font_scale=1.25)
ax = sns.scatterplot(x = model.fittedvalues, y=model.resid,color = 'orange')
sns.lineplot(x = [pred_price.min(),pred_price.max()],y = [0,0],color = 'blue')
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('${x:,.0f}')
```



The model seems to satisfy this assumption, as we do not observe a strong pattern in the residuals around the line Residuals = 0. Residuals are distributed more or less in a similar manner on both sides of the blue line for all fitted values.

For the model to satisfy the linearity assumption perfectly, the points above the line *(Residuals = 0)*, should be mirror age of the points below the line, i.e., the blue line in the above plot should act as a mirror.

What to do if there is non-linear association (page 94 of book): If the residual plot indicates that there are non-linear associations in the data, then a simple approach is to use non-linear transformations of the predictors, such as $logX, \sqrt{X}$, and X^2, in the regression model.

4.2 Non-constant variance of error terms

The variance of the error terms is assumed to be constant, i.e., $Var(\epsilon_i) = \sigma^2$, and this assumption is used while deriving the standard errors of the regression coefficients. The standard errors in turn are used to test the significant of the predictors, and obtain their confidence interval. Thus, violation of this assumption may lead to incorrect inference. Non-constant variance of error terms, or violation of the constant variance assumption, is called *heteroscedasticity.*

This assumption can be checked by plotting the residuals against fitted values.

```
#Plotting residuals vs fitted values
ax = sns.scatterplot(x = model.fittedvalues, y=model.resid,color = 'orange')
sns.lineplot(x = [pred_price.min(),pred_price.max()],y = [0,0],color = 'blue')
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('${x:,.0f}')
```

We see that the variance of errors seems to increase with increase in the fitted values. In such a case a log transformation of the response can resolve the issue to some extent. This is because a log transform will result in a higher shrinkage of larger values.

```
#Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
model_log.summary()
```

Table 4.5: OLS Regression Results

Dep. Variable:	np.log(price)	R-squared:	0.803
Model:	OLS	Adj. R-squared:	0.803
Method:	Least Squares	F-statistic:	1834.
Date:	Wed, 25 Jan 2023	Prob (F-statistic):	0.00
Time:	11:37:55	Log-Likelihood:	-1173.8
No. Observations:	4960	AIC:	2372.
Df Residuals:	4948	BIC:	2450.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-238.2125	25.790	-9.237	0.000	-288.773	-187.652
year	0.1227	0.013	9.608	0.000	0.098	0.148
engineSize	13.8349	5.795	2.387	0.017	2.475	25.195
mileage	0.0005	0.000	3.837	0.000	0.000	0.001
mpg	-1.2446	0.345	-3.610	0.000	-1.921	-0.569
year:engineSize	-0.0067	0.003	-2.324	0.020	-0.012	-0.001
year:mileage	-2.67e-07	6.8e-08	-3.923	0.000	-4e-07	-1.34e-07
year:mpg	0.0006	0.000	3.591	0.000	0.000	0.001
engineSize:mileage	-2.668e-07	4.08e-07	-0.654	0.513	-1.07e-06	5.33e-07
engineSize:mpg	0.0028	0.000	6.842	0.000	0.002	0.004
mileage:mpg	7.235e-08	1.79e-08	4.036	0.000	3.72e-08	1.08e-07
I(mileage ** 2)	1.828e-11	5.64e-12	3.240	0.001	7.22e-12	2.93e-11

Omnibus:	711.515	Durbin-Watson:	0.498
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2545.807
Skew:	0.699	Prob(JB):	0.00
Kurtosis:	6.220	Cond. No.	1.73e+13

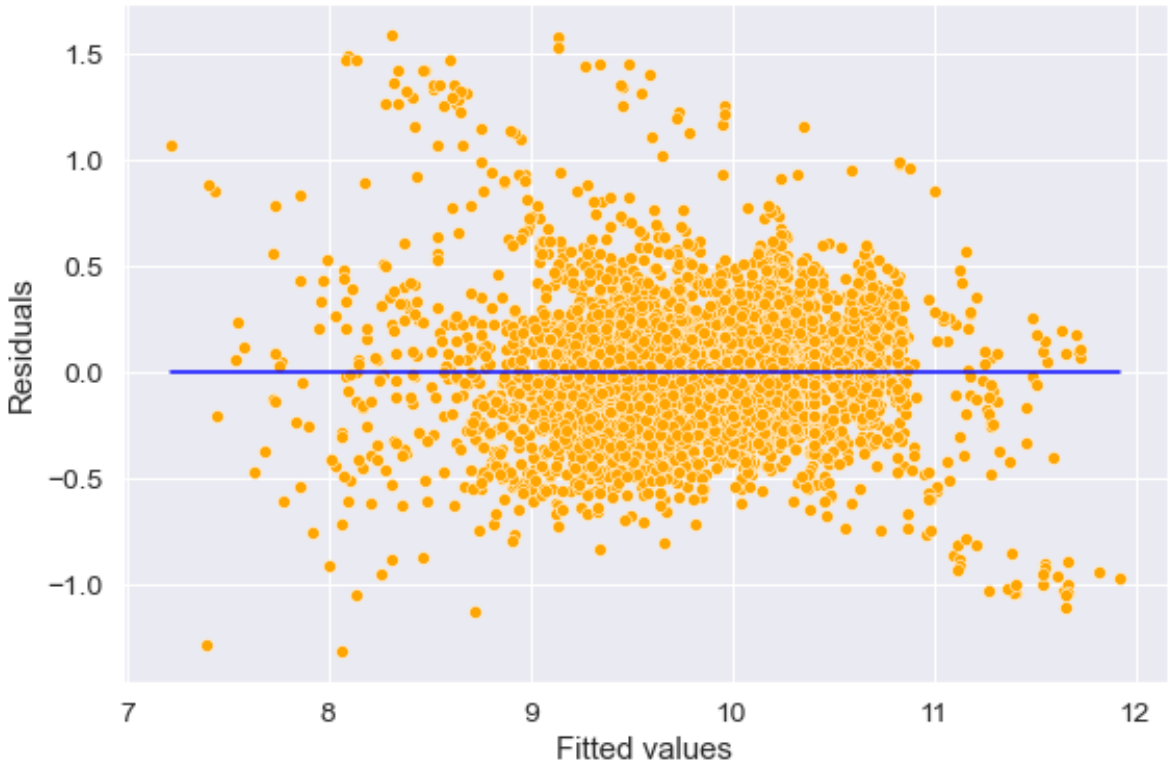
Note that the coefficient of *year* turns out to be significant (at 5% significance level), unlike in the previous model. Intuitively, the coefficient of *year* should have been significant, as *year* has the highest linear correlation of 50% with car *price*.

Although the R-squared has increased as compared to the previous model, violation of this assumption does not cause bias in the regression coefficients. Thus, there may not be a large improvement in the model fit, unless we add predictor(s) to address heteroscedasticity.

Let us check the constant variance assumption again.

```
#Plotting residuals vs fitted values
sns.scatterplot(x = (model_log.fittedvalues), y=(model_log.resid),color = 'orange')
sns.lineplot(x = [model_log.fittedvalues.min(),model_log.fittedvalues.max()],y = [0,0],col
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
```

```
Text(0, 0.5, 'Residuals')
```



Now we observe that the constant variance assumption is satisfied. Let us see the RMSE of this model on test data.

```
#Computing RMSE on test data
pred_price_log = model_log.predict(testf)
np.sqrt(((testp.price - np.exp(pred_price_log))**2).mean())
```

```
9094.209503063496
```

Note that the RMSE of the log-transformed model has increased as compared to the model without transformation. Does it mean the log-transformed model is less accurate?

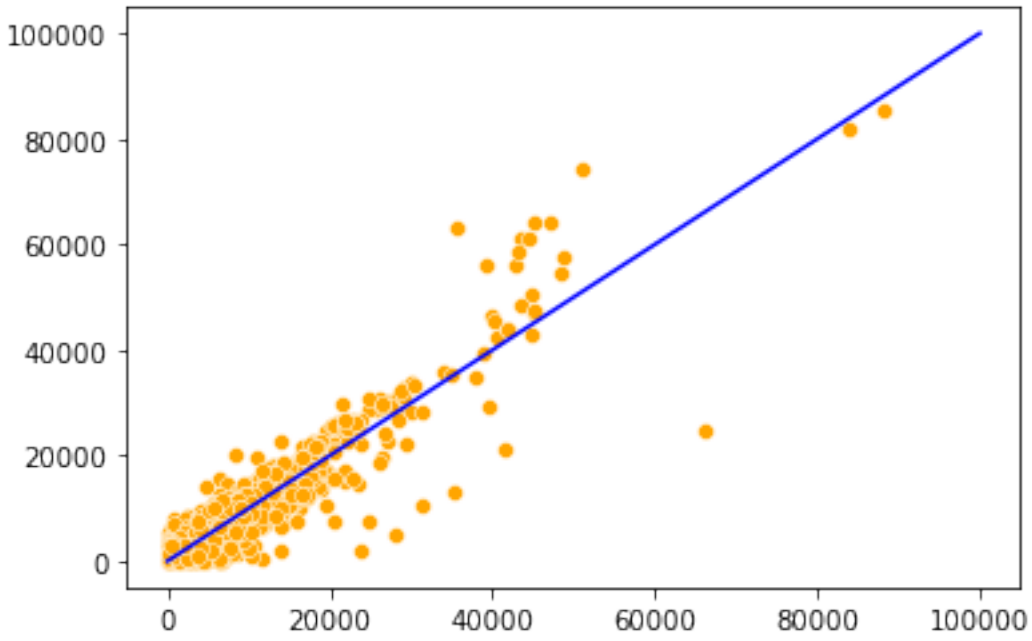
```
#Computing MAE on test data
pred_price_log = model_log.predict(testf)
((np.abs(testp.price - np.exp(pred_price_log))).mean())
```

```
5268.398904745121
```

Although the RMSE has increased a bit for the log-transformed model, the MAE has reduced. This means the log-transformed model does a bit worse on reducing relatively large errors, but does better in reducing the absolute errors on an average.

```
#Comparing errors of the log-transformed model with the previous model
err = np.abs(testp.price - pred_price)
err_log = np.abs(testp.price - np.exp(pred_price_log))
sns.scatterplot(x = err,y = err_log, color = 'orange')
sns.lineplot(x = [0,100000], y = [0,100000], color = 'blue')
np.sum(err_log<err)/len(err)
```

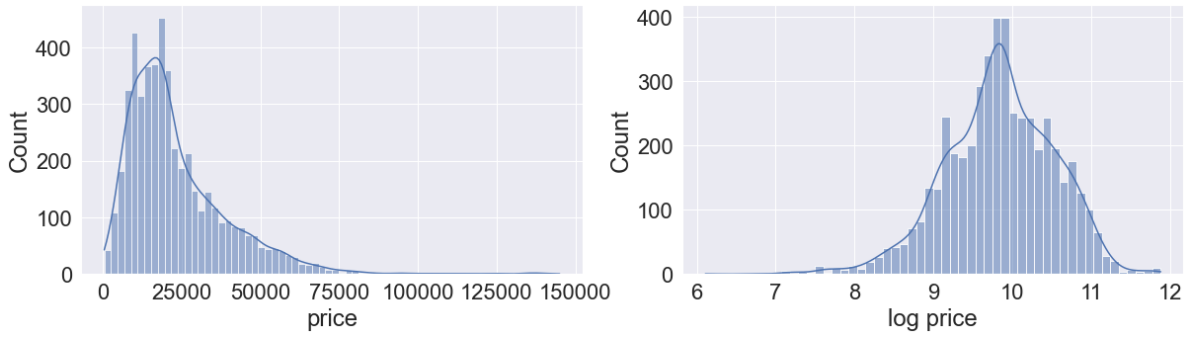
```
0.5572604790419161
```



For 56% of the cars, the log transformed makes a more accurate prediction than the previous model, which is another criterion based on which the log-transformed model is more accurate. However, the conclusion based on RMSE is different. This is because RMSE can be influenced by a few large errors. Thus, RMSE, though sometimes appropriate than other criteria, should not be used as the sole measure to compare the accuracy of models.

```
#Visualizing the distribution of price and log(price)
fig = plt.figure()
fig.subplots_adjust(hspace=0.4, wspace=0.2)
sns.set(rc = {'figure.figsize':(20,12)})
sns.set(font_scale = 2)
ax = fig.add_subplot(2, 2, 1)
sns.histplot(train.price,kde=True)
ax.set(xlabel='price', ylabel='Count')
ax = fig.add_subplot(2, 2, 2)
sns.histplot(np.log(train.price),kde=True)
ax.set(xlabel='log price', ylabel='Count')
```

```
[Text(0.5, 0, 'log price'), Text(0, 0.5, 'Count')]
```



We can see that the log transformation shrinked the higher values of price, making its distribution closer to normal.

Note that heterscedasticity can also occur due to model misspecification, i.e., in case of missing predictor(s). Some of the cars are too expensive, which makes the *price* distribution skewed. Perhaps, the price of expensive cars be better explained by the car *model*, a predictor that is missing in the current model.

5 Potential issues

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

Let us continue with the car price prediction example from the previous chapter.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

```
# Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
model_log.summary()
```

Table 5.2: OLS Regression Results

Dep. Variable:	np.log(price)	R-squared:	0.803
Model:	OLS	Adj. R-squared:	0.803
Method:	Least Squares	F-statistic:	1834.
Date:	Sun, 29 Jan 2023	Prob (F-statistic):	0.00
Time:	00:18:20	Log-Likelihood:	-1173.8
No. Observations:	4960	AIC:	2372.
Df Residuals:	4948	BIC:	2450.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-238.2125	25.790	-9.237	0.000	-288.773	-187.652
year	0.1227	0.013	9.608	0.000	0.098	0.148
engineSize	13.8349	5.795	2.387	0.017	2.475	25.195
mileage	0.0005	0.000	3.837	0.000	0.000	0.001
mpg	-1.2446	0.345	-3.610	0.000	-1.921	-0.569
year:engineSize	-0.0067	0.003	-2.324	0.020	-0.012	-0.001
year:mileage	-2.67e-07	6.8e-08	-3.923	0.000	-4e-07	-1.34e-07
year:mpg	0.0006	0.000	3.591	0.000	0.000	0.001
engineSize:mileage	-2.668e-07	4.08e-07	-0.654	0.513	-1.07e-06	5.33e-07
engineSize:mpg	0.0028	0.000	6.842	0.000	0.002	0.004
mileage:mpg	7.235e-08	1.79e-08	4.036	0.000	3.72e-08	1.08e-07
I(mileage ** 2)	1.828e-11	5.64e-12	3.240	0.001	7.22e-12	2.93e-11

Omnibus:	711.515	Durbin-Watson:	0.498
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2545.807
Skew:	0.699	Prob(JB):	0.00
Kurtosis:	6.220	Cond. No.	1.73e+13

5.1 Outliers

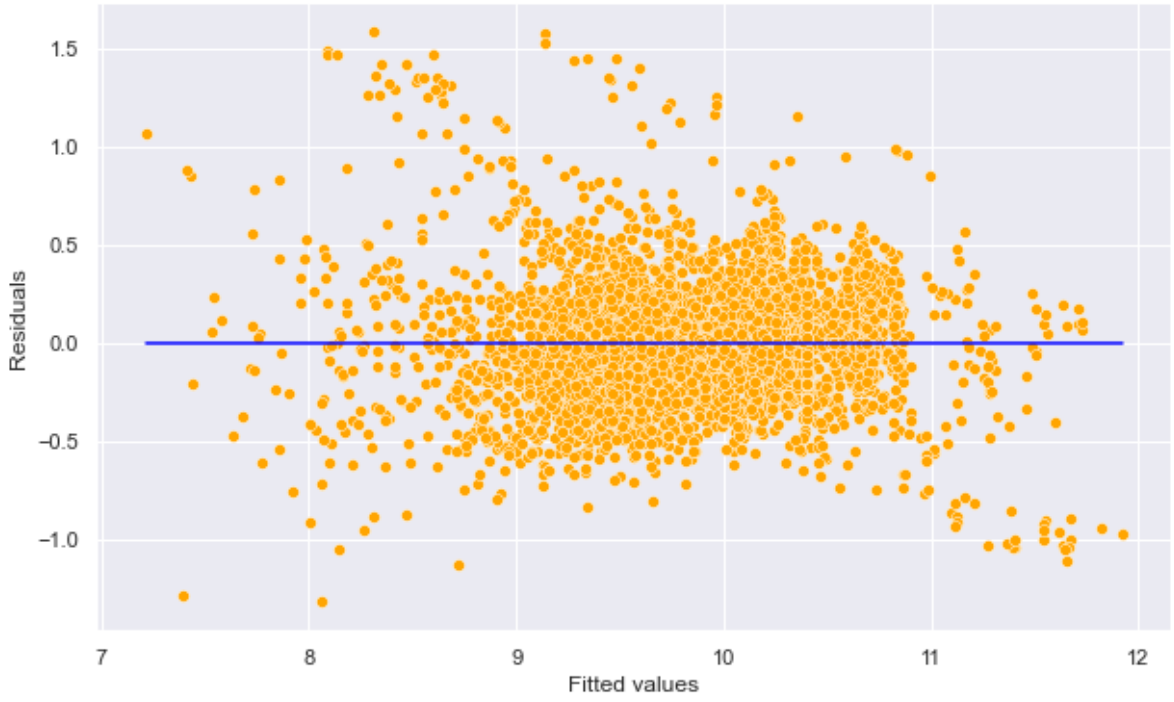
An outlier is a point for which the true response (y_i) is far from the value predicted by the model. Residual plots can be used to identify outliers.

If the the response at the i^{th} observation is y_i, the prediction is $\widehat{y}_i$, then the residual e_i is:

$$e_i = y_i - \widehat{y}_i$$

```
#Plotting residuals vs fitted values
sns.set(rc={'figure.figsize':(10,6)})
sns.scatterplot(x = (model_log.fittedvalues), y=(model_log.resid),color = 'orange')
sns.lineplot(x = [model_log.fittedvalues.min(),model_log.fittedvalues.max()],y = [0,0],col
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
```

```
Text(0, 0.5, 'Residuals')
```



Some of the errors may be high. However, it is difficult to decide how large a residual needs to be before we can consider a point to be an outlier. To address this problem, we have standardized residuals, which are defined as:

$$r_i = \frac{e_i}{RSE(\sqrt{1-h_{ii}})},$$

where r_i is the standardized residual, RSE is the residual standard error, and h_{ii} is the leverage *(introduced in the next section)* of the i^{th} observation.

Standardized residuals, allow the residuals to be compared on a *standard scale*.

Issue with standardized residuals:, If the observation corresponding to the standardized residual has a high leverage, then it will drag the regression line / plane / hyperplane towards it, thereby influencing the estimate of the residual itself.

Studentized residuals: To address the issue with standardized residuals, studentized residual for the i^{th} observation is computed as the standardized residual, but with the RSE (residual standard error) computed after removing the i^{th} observation from the data. Studentized residual, t_i for the i^{th} observation is given as:

$$t_i = \frac{e_i}{RSE_i(\sqrt{1-h_{ii}})},$$

where RSE_i is the residual standard error of the model developed on the data without the i^{th} observation.

Studentized residuals follow a t distribution with (n–p–2) degrees of freedom. Thus, in general, observations whose studentized residuals have a magnitude higher than 3 are potential outliers.

Let us find the studentized residuals in our car `price` prediction model.

```
#Studentized residuals
out = model_log.outlier_test()
out
```

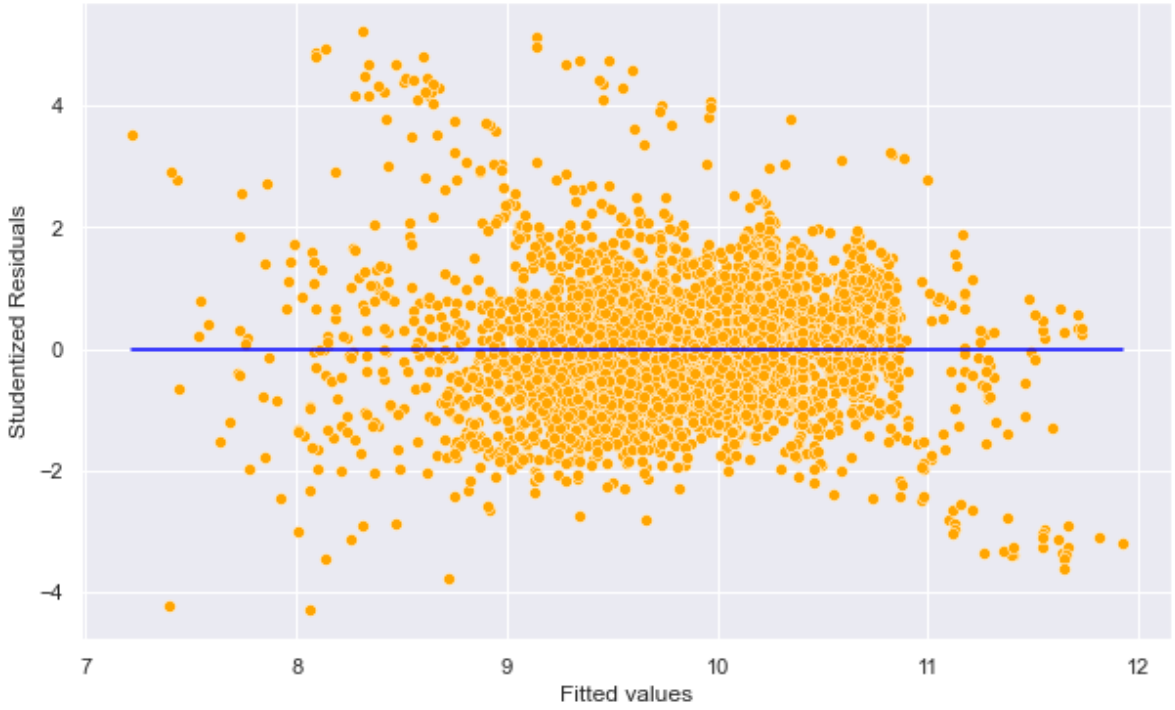
	student_resid	unadj_p	bonf(p)
0	-1.164204	0.244398	1.0
1	-0.801879	0.422661	1.0
2	-1.263820	0.206354	1.0
3	-0.614171	0.539130	1.0
4	0.027930	0.977719	1.0
...	...	...	...
4955	-0.523361	0.600747	1.0
4956	-0.509539	0.610397	1.0
4957	-1.718802	0.085713	1.0
4958	-0.077595	0.938153	1.0
4959	-0.482388	0.629551	1.0

Studentized residuals are in the first column of the above table.

```
#Plotting studentized residuals vs fitted values
sns.scatterplot(x = (model_log.fittedvalues), y=(out.student_resid),color = 'orange')
sns.lineplot(x = [model_log.fittedvalues.min(),model_log.fittedvalues.max()],y = [0,0],col
```

```
plt.xlabel('Fitted values')
plt.ylabel('Studentized Residuals')
```

```
Text(0, 0.5, 'Studentized Residuals')
```



Potential outliers: Observations whose studentized residuals have a magnitude greater than 3.

Impact of outliers: Outliers do not have a large impact on the OLS line / plane / hyperplane. However, outliers do inflate the residual standard error (RSE). RSE in turn is used to compute the standard errors of regression coefficients. As a result, statistically significant variables may appear to be insignificant, and R^2 may appear to be lower.

```
#Number of points with absolute studentized residuals greater than 3
np.sum((np.abs(out.student_resid)>3))
```

```
86
```

Are there outliers in our example?: In the above plot, there are 86 points with absolute studentized residuals larger than 3. However, most of the predictors are significant and R-squared has a relatively high value of 80%. Thus, even if there are outliers, there is no need to remove them as it is unlikely to change the significance of individual variables. Furthermore, looking into the data, we find that the price of some of the luxury cars such as Mercedez G-class is actually much higher than average. So, the potential outliers in the data do not seem to be due to incorrect data. The high studentized residuals may be due to some deficiency in the model, such as missing predictor(s) (like car `model`), rather than incorrect data. Thus, we should not remove any data that has an outlying value of *log(price)*.

Since `model` seems to be a variable that can explain the price of overly expensive cars, let us include it in the regression model.

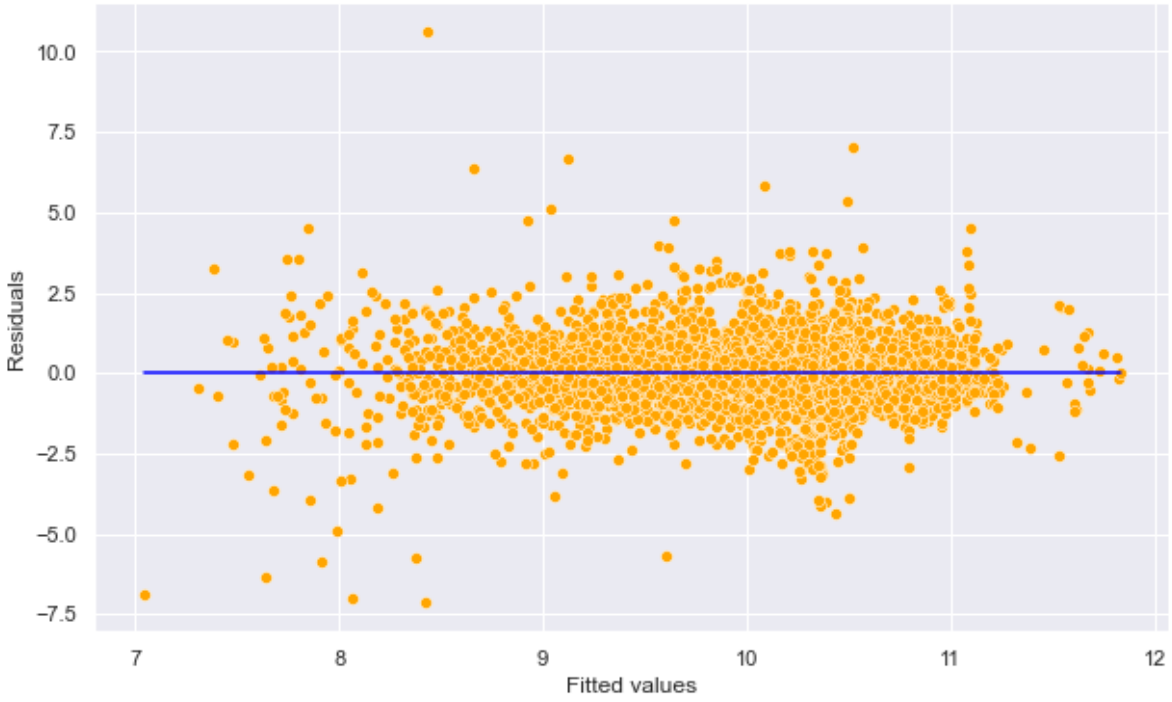
```
#Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
#Model summary not printed to save space
#model_log.summary()
```

```
#Computing RMSE on test data with car 'model' as one of the predictors
pred_price_log2 = model_log.predict(testf)
np.sqrt(((testp.price - np.exp(pred_price_log2))**2).mean())
```

```
4252.20045604376
```

```
#Plotting studentized residuals vs fitted values for the model with car 'model' as one of
out = model_log.outlier_test()
sns.scatterplot(x = (model_log.fittedvalues), y=(out.student_resid),color = 'orange')
sns.lineplot(x = [model_log.fittedvalues.min(),model_log.fittedvalues.max()],y = [0,0],col
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
```

```
Text(0, 0.5, 'Residuals')
```




```
#Number of points with absolute studentized residuals greater than 3
np.sum((np.abs(out.student_resid)>3))
```

```
69
```

Note the RMSE has reduced to almost half of its value corresponding to the regression model without the predictor - `model`. Car model does help better explain the variation in price of cars! The number of points with absolute studentized residuals greater than 3 has also reduced to 69 from 86.

5.2 High leverage points

High leverage points are those with an unsual value of the predictor(s). They have a relatively higher impact on the OLS line / plane / hyperplane, as compared to the outliers.

Leverage statistic (page 99 of the book): In order to quantify an observation's leverage, we compute the leverage statistic. A large value of this statistic indicates an observation with

high leverage. For simple linear regression,

$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2}. \tag{5.1}$$

It is clear from this equation that h_i increases with the distance of x_i from $\bar{x}$.The leverage statistic h_i is always between $1/n$ and 1, and the average leverage for all the observations is always equal to $(p+1)/n$. So if a given observation has a leverage statistic that greatly exceeds $(p+1)/n$, then we may suspect that the corresponding point has high leverage.

Influential points: Note that if a high leverage point falls in line with the regression line, then it will not effect the regression line. However, it may inflate R-squared and increase the significance of predictors. If a high leverage point falls away from the regression line, then it is also an outlier, and will effect the regression line. The points whose presence significantly effects the regression line are called influential points. A point that is both a high leverage point and an outlier is likely to be an influential point. However, a high leverage point is not necessarily an influential point.

Source for influential points: https://online.stat.psu.edu/stat501/book/export/html/973

Let us see if there are any high leverage points in our regression model without the predictor - `model`.

```
#Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
model_log.summary()
```

Table 5.6: OLS Regression Results

Dep. Variable:	np.log(price)	R-squared:	0.803
Model:	OLS	Adj. R-squared:	0.803
Method:	Least Squares	F-statistic:	1834.
Date:	Sun, 29 Jan 2023	Prob (F-statistic):	0.00
Time:	01:25:04	Log-Likelihood:	-1173.8
No. Observations:	4960	AIC:	2372.
Df Residuals:	4948	BIC:	2450.
Df Model:	11		
Covariance Type:	nonrobust		

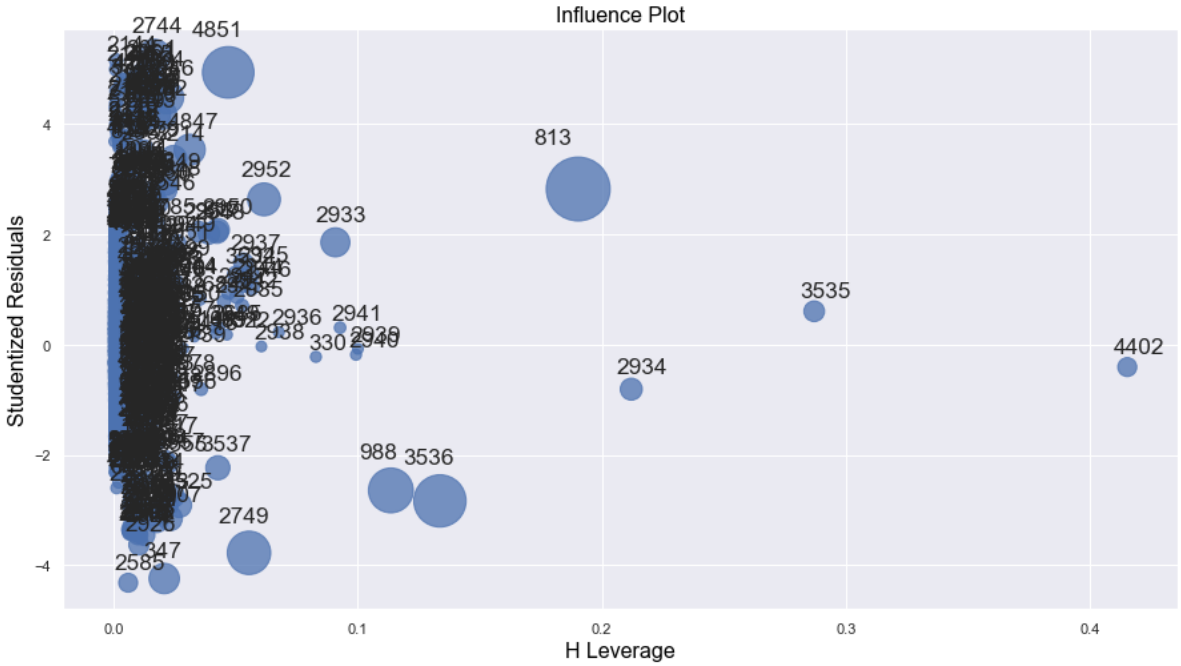
	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-238.2125	25.790	-9.237	0.000	-288.773	-187.652

year	0.1227	0.013	9.608	0.000	0.098	0.148
engineSize	13.8349	5.795	2.387	0.017	2.475	25.195
mileage	0.0005	0.000	3.837	0.000	0.000	0.001
mpg	-1.2446	0.345	-3.610	0.000	-1.921	-0.569
year:engineSize	-0.0067	0.003	-2.324	0.020	-0.012	-0.001
year:mileage	-2.67e-07	6.8e-08	-3.923	0.000	-4e-07	-1.34e-07
year:mpg	0.0006	0.000	3.591	0.000	0.000	0.001
engineSize:mileage	-2.668e-07	4.08e-07	-0.654	0.513	-1.07e-06	5.33e-07
engineSize:mpg	0.0028	0.000	6.842	0.000	0.002	0.004
mileage:mpg	7.235e-08	1.79e-08	4.036	0.000	3.72e-08	1.08e-07
I(mileage ** 2)	1.828e-11	5.64e-12	3.240	0.001	7.22e-12	2.93e-11

Omnibus:	711.515	Durbin-Watson:	0.498
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2545.807
Skew:	0.699	Prob(JB):	0.00
Kurtosis:	6.220	Cond. No.	1.73e+13

```
#Computing the leverage statistic for each observation
influence = model_log.get_influence()
leverage = influence.hat_matrix_diag
```

```
#Visualizng leverage against studentized residuals
sns.set(rc={'figure.figsize':(15,8)})
sm.graphics.influence_plot(model_log);
```



Let us identify the high leverage points in the data, as they may be effecting the regression line if they are outliers as well, i.e., if they are influential points. Note that there is no defined threshold for a point to be classified as a high leverage point. Some statisticians consider points having twice the average leverage as high leverage points, some consider points having thrice the average leverage as high leverage points, and so on.

```
out = model_log.outlier_test()
```

```
#Average leverage of points
(model_log.df_model+1)/model_log.nobs
```

```
0.0024193548387096775
```

Let us consider points having four times the average leverage as high leverage points.

```
#We will remove all observations that have leverage higher than the threshold value.
high_leverage_threshold = 4*(model_log.df_model+1)/model_log.nobs
```

```
#Number of high leverage points in the dataset
np.sum(leverage>high_leverage_threshold)
```

```
197
```

Observations that are both high leverage points and outliers are influential points that may effect the regression line. Let's remove these influential points from the data and see if it improves the model prediction accuracy on test data.

```
#Dropping influential points from data
train_filtered = train.drop(np.intersect1d(np.where(np.abs(out.student_resid)>3)[0],
                                           (np.where(leverage>high_leverage_threshold)[0])
```

```
train_filtered.shape
```

```
(4921, 11)
```

```
#Number of points removed as they were influential
train.shape[0]-train_filtered.shape[0]
```

```
39
```

We removed 39 influential data points from the training data.

```
#Model the model after removing the high leverage observations
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
model_log.summary()
```

Table 5.9: OLS Regression Results

Dep. Variable:	np.log(price)	R-squared:	0.830
Model:	OLS	Adj. R-squared:	0.829
Method:	Least Squares	F-statistic:	2173.
Date:	Sun, 29 Jan 2023	Prob (F-statistic):	0.00
Time:	01:26:25	Log-Likelihood:	-775.51
No. Observations:	4921	AIC:	1575.
Df Residuals:	4909	BIC:	1653.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-262.7743	24.455	-10.745	0.000	-310.717	-214.832
year	0.1350	0.012	11.148	0.000	0.111	0.159
engineSize	16.6645	5.482	3.040	0.002	5.917	27.412
mileage	0.0008	0.000	5.945	0.000	0.001	0.001
mpg	-1.1217	0.324	-3.458	0.001	-1.758	-0.486
year:engineSize	-0.0081	0.003	-2.997	0.003	-0.013	-0.003
year:mileage	-3.927e-07	6.5e-08	-6.037	0.000	-5.2e-07	-2.65e-07
year:mpg	0.0005	0.000	3.411	0.001	0.000	0.001
engineSize:mileage	-4.566e-07	3.86e-07	-1.183	0.237	-1.21e-06	3e-07
engineSize:mpg	0.0071	0.000	16.202	0.000	0.006	0.008
mileage:mpg	7.29e-08	1.68e-08	4.349	0.000	4e-08	1.06e-07
I(mileage ** 2)	1.418e-11	5.29e-12	2.683	0.007	3.82e-12	2.46e-11

Omnibus:	631.414	Durbin-Watson:	0.553
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1851.015
Skew:	0.682	Prob(JB):	0.00
Kurtosis:	5.677	Cond. No.	1.73e+13

Note that we obtain a higher R-sqauared value of 83% as compared to 80% with the complete data. Removing the influential points helped obtain a better model fit. However, that may also happen just by reducing observations.

```
#Computing RMSE on test data
pred_price_log = model_log.predict(testf)
np.sqrt(((testp.price - np.exp(pred_price_log))**2).mean())
```

```
8820.685844070766
```

The RMSE on test data has also reduced. This shows that some of the influential points were impacting the regression line. With those points removed, the model better captures the general trend in the data.

5.3 Collinearity

Collinearity refers to the situation when two or more predictor variables have a high linear association. Linear association between a pair of variables can be measured by the correlation coefficient. Thus the correlation matrix can indicate some potential collinearity problems.

Why and how is collinearity a problem (page 100-101 of book): The presence of collinearity can pose problems in the regression context, since it can be difficult to separate out the individual effects of collinear variables on the response.

Since collinearity reduces the accuracy of the estimates of the regression coefficients, it causes the standard error for $\hat{\beta}_j$ to grow. Recall that the t-statistic for each predictor is calculated by dividing $\hat{\beta}_j$ by its standard error. Consequently, collinearity results in a decline in the t-statistic. As a result, in the presence of collinearity, we may fail to reject $H_0 : \beta_j = 0$. This means that the power of the hypothesis test—the probability of correctly detecting a non-zero coefficient—is reduced by collinearity.

How to measure collinearity/multicollinearity (page 102 of book): Unfortunately, not all collinearity problems can be detected by inspection of the correlation matrix: it is possible for collinearity to exist between three or more variables even if no pair of variables has a particularly high correlation. We call this situation multicollinearity. Instead of inspecting the correlation matrix, a better way to assess multi- collinearity is to compute the variance inflation factor (VIF). The VIF is variance inflation factor the ratio of the variance of $\hat{\beta}_j$ when fitting the full model divided by the variance of $\hat{\beta}_j$ if fit on its own. The smallest possible value for VIF is 1, which indicates the complete absence of collinearity. Typically in practice there is a small amount of collinearity among the predictors. As a rule of thumb, a VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity. The VIF for each variable can be computed using the formula:

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}} \tag{5.2}$$

```
#Correlation matrix
train.corr()
```

	carID	year	mileage	tax	mpg	engineSize	price
carID	1.000000	0.006251	-0.001320	0.023806	-0.010774	0.011365	0.012129
year	0.006251	1.000000	-0.768058	-0.205902	-0.057093	0.014623	0.501296
mileage	-0.001320	-0.768058	1.000000	0.133744	0.125376	-0.006459	-0.478705
tax	0.023806	-0.205902	0.133744	1.000000	-0.488002	0.465282	0.144652
mpg	-0.010774	-0.057093	0.125376	-0.488002	1.000000	-0.419417	-0.369919
engineSize	0.011365	0.014623	-0.006459	0.465282	-0.419417	1.000000	0.624899
price	0.012129	0.501296	-0.478705	0.144652	-0.369919	0.624899	1.000000

Let us compute the Variance Inflation Factor (VIF) for the four predictors.

```
X = train[['mpg','year','mileage','engineSize']]
```

```
X.columns[1:]
```

```
Index(['mpg', 'year', 'mileage', 'engineSize'], dtype='object')
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
X = add_constant(X)
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns

# calculating VIF for each feature
#vif_data["VIF"] = [variance_inflation_factor(X.values, i)
#                          for i in range(len(X.columns))]

for i in range(len(X.columns)):
    vif_data.loc[i,'VIF'] = variance_inflation_factor(X.values, i)

print(vif_data)
```

```
      feature           VIF
0       const  1.201579e+06
1         mpg  1.243040e+00
2        year  2.452891e+00
3     mileage  2.490210e+00
4  engineSize  1.219170e+00
```

As all the values of VIF are close to one, we do not have the problem of multicollinearity in the model. Note that the VIF of *year* and *mileage* is relatively high as they are the most correlated.

```
#Manually computing the VIF for year
ols_object = smf.ols(formula = 'year~(mpg+engineSize+mileage)', data = train)
model_log = ols_object.fit()
model_log.summary()
```

Table 5.13: OLS Regression Results

Dep. Variable:	year	R-squared:	0.592
Model:	OLS	Adj. R-squared:	0.592
Method:	Least Squares	F-statistic:	2400.
Date:	Tue, 01 Feb 2022	Prob (F-statistic):	0.00
Time:	01:24:20	Log-Likelihood:	-10066.
No. Observations:	4960	AIC:	2.014e+04
Df Residuals:	4956	BIC:	2.017e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2018.3135	0.140	1.44e+04	0.000	2018.039	2018.588
mpg	0.0095	0.002	5.301	0.000	0.006	0.013
engineSize	0.1171	0.037	3.203	0.001	0.045	0.189
mileage	-9.139e-05	1.08e-06	-84.615	0.000	-9.35e-05	-8.93e-05

Omnibus:	2949.664	Durbin-Watson:	1.161
Prob(Omnibus):	0.000	Jarque-Bera (JB):	63773.271
Skew:	-2.426	Prob(JB):	0.00
Kurtosis:	19.883	Cond. No.	1.91e+05

```
#VIF for year
1/(1-0.592)
```

```
2.4509803921568625
```

Note that year and mileage have a high linear correlation. Removing one of them should decrease the standard error of the coefficient of the other, without significanty decrease R-squared.

```
ols_object = smf.ols(formula = '(price)~(mpg+engineSize+mileage+year)', data = train)
model_log = ols_object.fit()
model_log.summary()
```

Table 5.16: OLS Regression Results

Dep. Variable:	price	R-squared:	0.660
Model:	OLS	Adj. R-squared:	0.660
Method:	Least Squares	F-statistic:	2410.
Date:	Sun, 23 Jan 2022	Prob (F-statistic):	0.00
Time:	01:45:55	Log-Likelihood:	-52497.
No. Observations:	4960	AIC:	1.050e+05
Df Residuals:	4955	BIC:	1.050e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.661e+06	1.49e+05	-24.593	0.000	-3.95e+06	-3.37e+06
mpg	-79.3126	9.338	-8.493	0.000	-97.620	-61.006
engineSize	1.218e+04	189.969	64.107	0.000	1.18e+04	1.26e+04
mileage	-0.1474	0.009	-16.817	0.000	-0.165	-0.130
year	1817.7366	73.751	24.647	0.000	1673.151	1962.322

Omnibus:	2450.973	Durbin-Watson:	0.541
Prob(Omnibus):	0.000	Jarque-Bera (JB):	31060.548
Skew:	2.045	Prob(JB):	0.00
Kurtosis:	14.557	Cond. No.	3.83e+07

Removing mileage from the above regression.

```
ols_object = smf.ols(formula = '(price)~(mpg+engineSize+year)', data = train)
model_log = ols_object.fit()
model_log.summary()
```

Table 5.19: OLS Regression Results

Dep. Variable:	price	R-squared:	0.641
Model:	OLS	Adj. R-squared:	0.641
Method:	Least Squares	F-statistic:	2951.
Date:	Mon, 24 Jan 2022	Prob (F-statistic):	0.00
Time:	00:04:28	Log-Likelihood:	-52635.
No. Observations:	4960	AIC:	1.053e+05
Df Residuals:	4956	BIC:	1.053e+05

Table 5.19: OLS Regression Results

Df Model:	3
Covariance Type:	nonrobust

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-5.586e+06	9.78e+04	-57.098	0.000	-5.78e+06	-5.39e+06
mpg	-101.9120	9.500	-10.727	0.000	-120.536	-83.288
engineSize	1.196e+04	194.848	61.392	0.000	1.16e+04	1.23e+04
year	2771.1844	48.492	57.147	0.000	2676.118	2866.251

Omnibus:	2389.075	Durbin-Watson:	0.528
Prob(Omnibus):	0.000	Jarque-Bera (JB):	26920.051
Skew:	2.018	Prob(JB):	0.00
Kurtosis:	13.675	Cond. No.	1.41e+06

```
ols_object = smf.ols(formula = '(price)~(mpg+engineSize+year+mileage)', data = train)
model_log = ols_object.fit()
model_log.summary()
```

Table 5.22: OLS Regression Results

Dep. Variable:	price	R-squared:	0.660
Model:	OLS	Adj. R-squared:	0.660
Method:	Least Squares	F-statistic:	2410.
Date:	Mon, 31 Jan 2022	Prob (F-statistic):	0.00
Time:	12:10:59	Log-Likelihood:	-52497.
No. Observations:	4960	AIC:	1.050e+05
Df Residuals:	4955	BIC:	1.050e+05
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3.661e+06	1.49e+05	-24.593	0.000	-3.95e+06	-3.37e+06
mpg	-79.3126	9.338	-8.493	0.000	-97.620	-61.006
engineSize	1.218e+04	189.969	64.107	0.000	1.18e+04	1.26e+04
year	1817.7366	73.751	24.647	0.000	1673.151	1962.322

mileage	-0.1474	0.009	-16.817	0.000	-0.165	-0.130

Omnibus:	2450.973	Durbin-Watson:	0.541
Prob(Omnibus):	0.000	Jarque-Bera (JB):	31060.548
Skew:	2.045	Prob(JB):	0.00
Kurtosis:	14.557	Cond. No.	3.83e+07

```
ols_object = smf.ols(formula = '(price)~(year)', data = train)
model_log = ols_object.fit()
model_log.summary()
```

Table 5.25: OLS Regression Results

Dep. Variable:	price	R-squared:	0.251
Model:	OLS	Adj. R-squared:	0.251
Method:	Least Squares	F-statistic:	1664.
Date:	Mon, 31 Jan 2022	Prob (F-statistic):	5.84e-314
Time:	12:11:12	Log-Likelihood:	-54459.
No. Observations:	4960	AIC:	1.089e+05
Df Residuals:	4958	BIC:	1.089e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-5.728e+06	1.41e+05	-40.627	0.000	-6e+06	-5.45e+06
year	2851.7747	69.907	40.794	0.000	2714.725	2988.824

Omnibus:	2541.815	Durbin-Watson:	0.228
Prob(Omnibus):	0.000	Jarque-Bera (JB):	23237.973
Skew:	2.270	Prob(JB):	0.00
Kurtosis:	12.583	Cond. No.	1.41e+06

```
73.751/69.907
```

```
1.0549873403235728
```

```
1.05**2
```

```
1.1025
```

Note that the standard error of the coefficient of *year* has reduced from 73 to 48, without any large reduction in R-squared.

A Assignment A

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells and your answer in the *Markdown* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. The assignment is worth 100 points, and is due on **Tuesday, 17th January 2023 at 11:59 pm**.
6. There is a **bonus** question worth 5 points.
7. **Five points are for properly formatting the assignment**. The breakdown is as follows:

- Must be an HTML file rendered using Quarto (1 pt); *If you have a Quarto issue, you must mention the issue & quote the error you get when rendering using Quarto in the comments section of Canvas, and submit the ipynb file.*
- No name can be written on the assignment, nor can there be any indicator of the student's identity—e.g., printouts of the working directory should not be included in the final submission (1 pt).
- There aren't excessively long outputs of extraneous information (e.g. no printouts of entire data frames without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt).
- Final answers of each question are written in Markdown cells (1 pt).
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt).

8. The maximum possible score in the assigment is 95 + 5 (formatting) + 5 (bonus question) = 105 out of 100. There is no partial credit for the bonus question.

A.1 Regression vs Classification; Prediction vs Inference

Explain (1) whether each scenario is a classification or regression problem, and (2) whether we are most interested in inference or prediction. Answers to both parts must be supported by a justification.

A.1.1

Consider a company that is interested in conducting a marketing campaign. The goal is to identify individuals who are likely to respond positively to a marketing campaign, based on observations of demographic variables *(such as age, gender, income, etc.)* measured on each individual.

(2+2 points)

A.1.2

Consider that the company mentioned in the previous question is interested in understanding the impact of advertising promotions in different media types on the company sales. For example, the company is interested in the question, *'how large of an increase in sales is associated with a given increase in radio vis-a-vis TV advertising?'*

(2+2 points)

A.1.3

Consider a company selling furniture is interested in the finding the association between demographic characterisitcs of customers (such as age, gender, income, etc.) and their probability of purchase of a particular company product.

(2+2 points)

A.1.4

We are interested in predicting the % change in the USD/Euro exchange rate in relation to the weekly changes in the world stock markets. Hence we collect weekly data for all of 2022. For each week we record the % change in the USD/Euro, the % change in the US market, the % change in the British market, and the % change in the German market.

(2+2 points)

A.2 RMSE vs MAE

A.2.1

Describe a regression problem, where it will be more appropriate to assess the model accuracy using the root mean squared error (RMSE) metric as compared to the mean absolute error (MAE) metric.

Note: Don't use the examples presented in class

(4 points)

A.2.2

Describe a regression problem, where it will be more appropriate to assess the model accuracy using the mean absolute error (MAE) metric as compared to the root mean squared error (RMSE) metric.

Note: Don't use the examples presented in class

(4 points)

A.3 FNR vs FPR

A.3.1

A classification model is developed to predict those customers who will respond positively to a company's tele-marketing campaign. All those customers that are predicted to respond positively to the campaign will be called by phone to buy the product being marketed. If the customer being called purchases the product ($y = 1$), the company will get a profit of \$100. On the other hand, if they are called and they don't purchase ($y = 0$), the company will have a loss of \$1. Among FPR (False positive rate) and FNR (False negative rate), which metric is more important to be minimized to reduce the loss associated with misclassification? Justify your answer.

In your justification, you must clearly interpret False Negatives (FN) and False Postives (FP) first.

Assumption: Assume that based on the past marketing campaigns, around 50% of the customers will actually respond positively to the campaign.

(4 points)

A.3.2

Can the answer to the previous question change if the assumption stated in the question is false? Justify your answer.

(6 points)

A.4 Petrol consumption

Read the dataset *petrol_consumption_train.csv*. It contains the following five columns:

`Petrol_tax`: Petrol tax (cents per gallon)

`Per_capita_income`: Average income (dollars)

`Paved_highways`: Paved Highways (miles)

`Prop_license`: Proportion of population with driver's licenses

`Petrol_consumption`: Consumption of petrol (millions of gallons)

A.4.1

Make a pairwise plot of all the variables in the dataset. Which variable seems to have the highest linear correlation with `Petrol_consumption`? Let this variable be predictor P. *Note: If you cannot figure out P by looking at the visualization, you may find the pairwise linear correlation coefficient to identify P.*

(4 points)

A.4.2

Fit a simple linear regression model to predict `Petrol_consumption` based on predictor P (identified in the previous part). Print the model summary.

(4 points)

A.4.3

Interpret the coefficient of P. What is the increase in petrol consumption for an increase of 0.05 in P?

(2+2 points)

A.4.4

Does petrol consumption have a statistically significant relationship with the predictor P? Justify your answer.

(4 points)

A.4.5

What is the R-squared? Interpret its value.

(4 points)

A.4.6

Use the model developed above to estimate the petrol consumption for a state in which 50% of the population has a driver's license. What are the confidence and prediction intervals for your estimate? Which interval includes the irreducible error?

(4+3+3+2 = 12 points)

A.4.7

Use the model developed above to estimate the petrol consumption for a state in which 10% of the population has a driver's license. Are you getting a reasonable estimate? Why or why not?

(5 points)

A.4.8

What is the residual standard error of the model?

(4 points)

A.4.9

Using the model developed above, predict the petrol consumption for the observations in *petrol_consumption_test.csv*. Find the RMSE (Root mean squared error). Include the units of RMSE in your answer.

(5 points)

A.4.10

Based on the answers to the previous two questions, do you think the model is overfitting? Justify your answer.

(4 points)

Make a scatterplot of `Petrol_consumption` vs `Prop_license` using *petrol_consumption_test.csv*. Over the scatterplot, plot the regression line, the prediction interval, and the confidence interval. Distinguish the regression line, prediction interval lines, and confidence interval lines with the following colors. Include the legend as well.

- Regression line: red
- Confidence interval lines: blue
- Prediction interval lines: green

(4 points)

Among the confidence and prediction intervals, which interval is wider, and why?

(1+2 points)

A.4.11

Find the correlation between `Petrol_consumption` and the rest of the variables in *petrol_consumption_train.csv*. Based on the correlations, a simple linear regression model with which predictor will have the least R-squared value for predicting `Petrol_consumption`. Don't develop any linear regression models.

(4 points)

Bonus point question

(5 points - no partial credit)

A.4.12

Fit a simple linear regression model to predict `Petrol_consumption` based on predictor P, but without an intercept term.

(you must answer this correctly to qualify for earning bonus points)

A.4.13

Estimate the petrol consumption for the observations in *petrol_consumption_test.csv* using the model in developed in the previous question. Find the RMSE.

(you must answer this correctly to qualify for earning bonus points)

A.4.14

The RMSE for the models with and without the intercept are similar, which indicates that both models are almost equally good. However, the R-squared for the model without intercept is much higher than the R-squared for the model with the intercept. Why? Justify your answer.

(5 points)

B Assignment B

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells and your answer in the *Markdown* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. The assignment is worth 100 points, and is due on **Thursday, 26th January 2023 at 11:59 pm**.
6. **Five points are properly formatting the assignment**. The breakdown is as follows:

- Must be an HTML file rendered using Quarto (1 pt). *If you have a Quarto issue, you must mention the issue & quote the error you get when rendering using Quarto in the comments section of Canvas, and submit the ipynb file.*
- No name can be written on the assignment, nor can there be any indicator of the student's identity—e.g. printouts of the working directory should not be included in the final submission (1 pt)
- There aren't excessively long outputs of extraneous information (e.g. no printouts of entire data frames without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- Final answers of each question are written in Markdown cells (1 pt).
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)

B.1 Multiple linear regression

A study was conducted on 97 men with prostate cancer who were due to receive a radical prostatectomy. The dataset *prostate.csv* contains data on 9 measurements made on these 97 men. The description of variables can be found here:

B.1.1 Training MLR

Fit a linear regression model with `lpsa` as the response and all the other variables as predictors. Write down the equation to predict `lpsa` based on the other eight variables.

(2+2 points)

B.1.2 Model significance

Is the overall regression significant at 5% level? Justify your answer.

(2 points)

B.1.3 Coefficient interpretation

Interpret the coefficient of `svi`.

(2 points)

B.1.4 Variable significance

Report the p-values for `gleason` and `age`. What do you conclude about the significance of these variables?

(2+2 points)

B.1.5 Variable significance from confidence interval

What is the 95% confidence interval for the coefficient of `age`? Can you conclude anything about its significance based on the confidence interval?

(2+2 points)

B.1.6 p-value

Fit a simple linear regression on `lpsa` against `gleason`. What is the p-value for `gleason`?

(1+1 points)

B.1.7 Predictor significance in presence / absence of other predictors

Is the predictor `gleason` statistically significant in the model developed in the previous question *(B.1.6)*?

Was `gleason` statistically significant in the model developed in the first question *(B.1.1)* with multiple predictors?

Did the statistical significance of `gleason` change in the absence of other predictors? Why or why not?

(1+1+4 points)

B.1.8 Prediction

Predict `lpsa` of a 65-year old man with `lcavol` = 1.35, `lweight` = 3.65, `lbph` = 0.1, `svi` = 0.22, `lcp` = -0.18, `gleason` = 6.75, and `pgg45` = 25 and find 95% prediction intervals.

(2 points)

B.1.9 Variable selection

Find the largest subset of predictors in the model developed in the first question *(B.1.1)*, such that their coefficients are zero, i.e., none of the predictors in the subset are statistically significant.

Does the model R-squared change a lot if you remove the set of predictors identifed above from the model in the first question *(B.1.1)*?

Hint: You may use the `f_test()` method to test hypotheses.

(4+1 points)

B.2 Using MLR coefficients and variable transformation

The dataset *infmort.csv* gives the infant mortality of different countries in the world. The column `mortality` contains the infant mortality in deaths per 1000 births.

B.2.1 Data visualisation

Make the following plots:

1. a boxplot of log(`mortality`) against `region` *(note that a plot of log(`mortality`) against `region` better distinguishes the mortality among regions as compared to a plot of `mortality` against `region`*,
2. a boxplot of `income` against `region`, and
3. a scatter plot of `mortality` against `income`.

What trends do you see in these plots? *Mention the trend separately for each plot.*

(3+2 points)

B.2.2 Removing effect of predictor from response

Europe seems to have the lowest infant mortality, but it also has the highest per capita annual income. We want to see if Europe still has the lowest mortality if we remove the effect of income from the mortality. We will answer this question with the following steps.

B.2.2.1 Variable transformation

Plot:

1. `mortality` against `income`,
2. log(`mortality`) against `income`,
3. `mortality` against log(`income`), and
4. log(`mortality`) against log(`income`).

Based on the plots, postulate an appropriate model to predict mortality as a function of income. *Print the model summary.*

(2+4 points)

B.2.2.2 Model update

Update the model developed in the previous question by adding `region` as a predictor. Print the model summary.

(2 points)

Use the model developed in the previous question to compute `adjusted_mortality` for each observation in the data, where adjusted mortality is the mortality after removing the estimated effect of income. Make a boxplot of log(`adjusted_mortality`) against `region`.

(4+2 points)

B.2.3 Data visualisation after removing effect of predictor from response

From the plot in the previous question:

1. Does Europe still seem to have the lowest mortality as compared to other regions after removing the effect of income from mortality?
2. After adjusting for income, is there any change in the mortality comparison among different regions. Compare the plot developed in the previous question to the plot of `log(mortality)` against `region` developed earlier *(B.2.1)* to answer this question.

Hint: Do any African / Asian / American countries seem to do better than all the European countries with regard to mortality after adjusting for income?

(1+3 points)

B.3 Variable transformations and interactions

The dataset *soc_ind.csv* contains the GDP per capita of some countries along with several social indicators.

B.3.1 Training SLR

For a simple linear regression model predicting `gdpPerCapita`. Which predictor will provide the best model fit *(ignore categorical predictors)*? Let that predictor be P.

(2 points)

B.3.2 Linearity in relationship

Make a scatterplot of `gdpPerCapita` vs P. Does the relationship between `gdpPerCapita` and P seem linear or non-linear?

(1 + 2 points)

B.3.3 Variable transformation

If the relationship identified in the previous question is non-linear, identify and include transformation(s) of the predictor P in the model to improve the model fit.

Mention the predictors of the transformed model, and report the change in the R-squared value of the transformed model as compared to the simple linear regression model with only P.

(4+4 points)

B.3.4 Model visualisation with transformed predictor

Plot the regression curve of the transformed model *(developed in the previous question)* over the scatterplot in (b) to visualize model fit. Also make the regression line of the simple linear regression model with only P on the same plot.

(3 + 1 points)

B.3.5 Training MLR with qualitative predictor

Develop a model to predict `gdpPerCapita` with P and `continent` as predictors.

1. Interpert the intercept term.
2. For a given value of P, are there any continents that **do not** have a signficant difference between their mean `gdpPerCapita` and that of Africa? If yes, then which ones, and why? If no, then why not? Consider a significance level of 5%.

(4 + 4 points)

B.3.6 Variable interaction

The model developed in the previous question has a limitation. It assumes that the increase in mean `gdpPerCapita` with a unit increase in P does not depend on the `continent`.

1. Eliminate this limitation by including interaction of `continent` with P in the model developed in the previous question. Print the model summary of the model with interactions.
2. Interpret the coefficient of any one of the interaction terms.

(4 + 4 points)

B.3.7 Model visualisation with qualitative predictor

Use the model developed in the previous question to plot the regression lines for Africa, Asia, and Europe. Put `gdpPerCapita` on the vertical axis and P on the horizontal axis. Use a legend to distinguish among the regression lines of the three continents.

(4 points)

B.3.8 Model interpretation

Based on the plot develop in the previous question, which continent has the highest increase in mean `gdpPerCapita` for a unit increase in P, and which one has the least? Justify your answer.

(2+2 points)

C Datasets, assignment and project files

Datasets used in the book, assignment files, and project files can be found here

References