

Data Science II with python (Class notes)

STAT 303-2

Arvind Krishna

1/3/23

Table of contents

Preface	3
I Linear regression	4
1 Simple Linear Regression	5
2 Multiple Linear Regression	11
3 Variable interactions and transformations	16
3.0.1 Variable interaction between continuous predictors	16
3.0.2 Including qualitative predictors in the model	19
3.0.3 Including qualitative predictors and its interaction with continuous predictor in the model	21
3.1 Variable transformations	23
Appendices	27
A Assignment A	28
A.1 Regression vs Classification; Prediction vs Inference	29
A.2 RMSE vs MAE	30
A.3 FNR vs FPR	30
A.4 Petrol consumption	31
B Datasets, assignment and project files	35
References	36

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:	Tue, 27 Dec 2022	Prob (F-statistic):	0.00
Time:	01:06:43	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

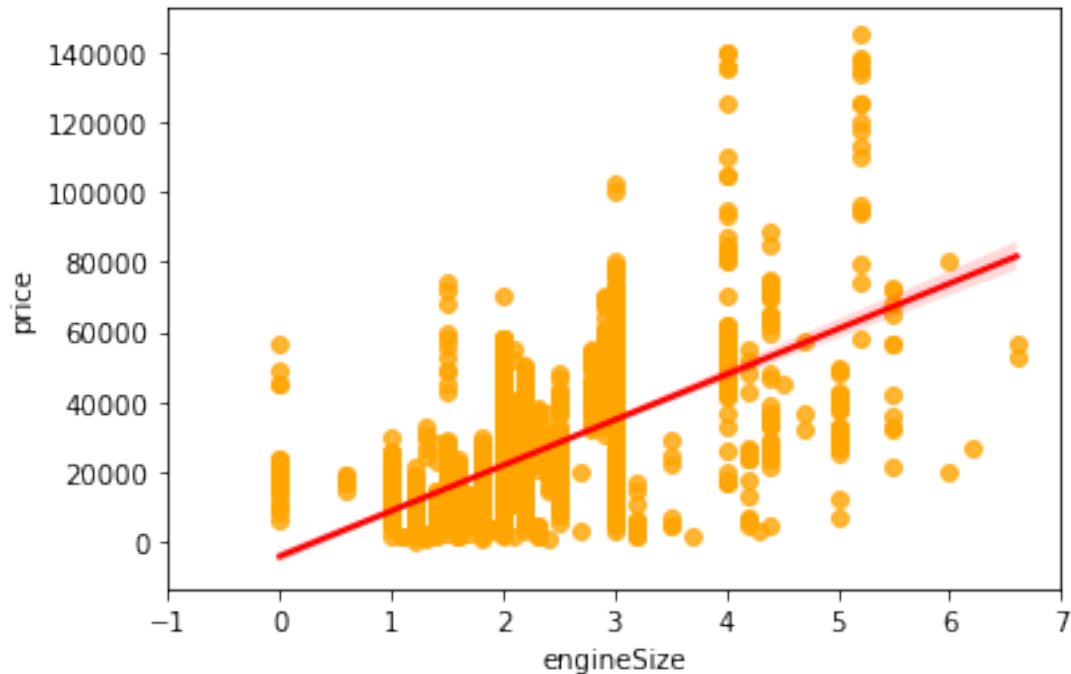
Omnibus:	1271.986	Durbin-Watson:	0.517
Prob(Omnibus):	0.000	Jarque-Bera (JB):	6490.719
Skew:	1.137	Prob(JB):	0.00
Kurtosis:	8.122	Cond. No.	7.64

The model equation is: $\text{car price} = -4122.0357 + 12990 * \text{engineSize}$

Visualize the regression line

```
sns.regplot(x = 'engineSize', y = 'price', data = train, color = 'orange', line_kws={"color": "red", "dash": [5, 5]})
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*

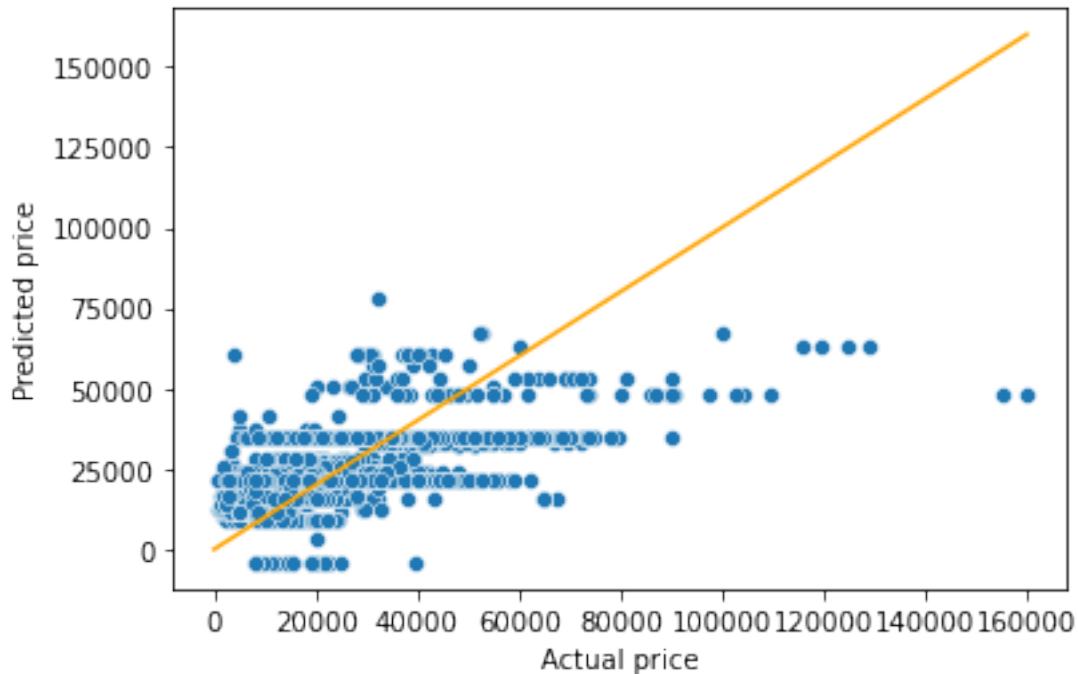
```
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
```

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

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') #Plotting the line x = y
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, it's 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 intervals
intervals = model.get_prediction(testf)

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

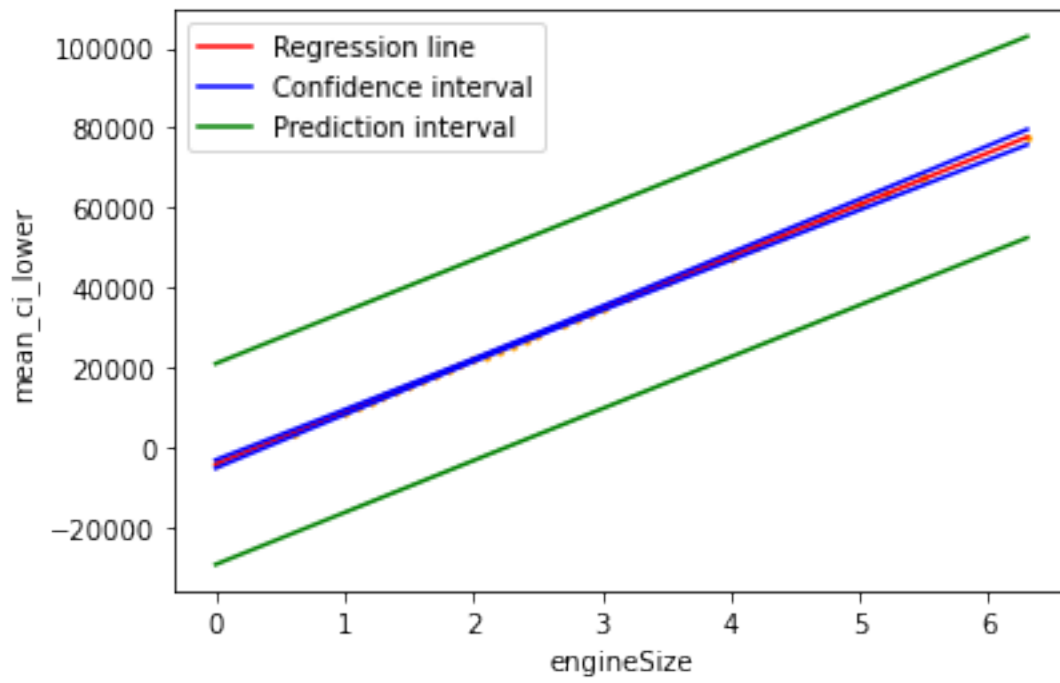
	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
1	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
2	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
3	8866.245277	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='Confidence interval')
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 0x27c6cfd1070>



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 car price
pred_price = model.predict(testf)#Note that the predict() function finds the predictor 'engineSize'
```

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') #Plotting the line x = y
```

```
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

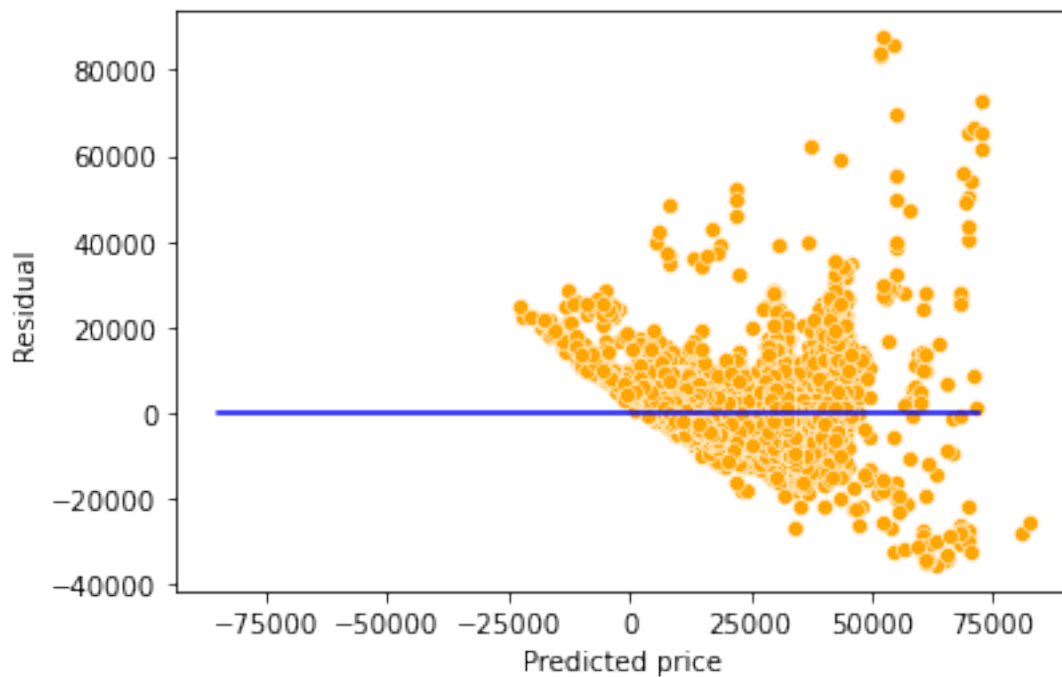
What is the residual standard error based on the training data?

```
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, 17 Jan 2023	Prob (F-statistic):	0.00
Time:	02:19:05	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 $-1.796e6 + 896.7687 * 2010 \approx \$6,500$, 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 $-1.796e6 + 896.7687 * 2020 \approx \$15,500$, assuming all the other predictors are constant.

Similarly, the equation can be re-arranged as:

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

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 \approx \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 \approx \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 reduced as compared to 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

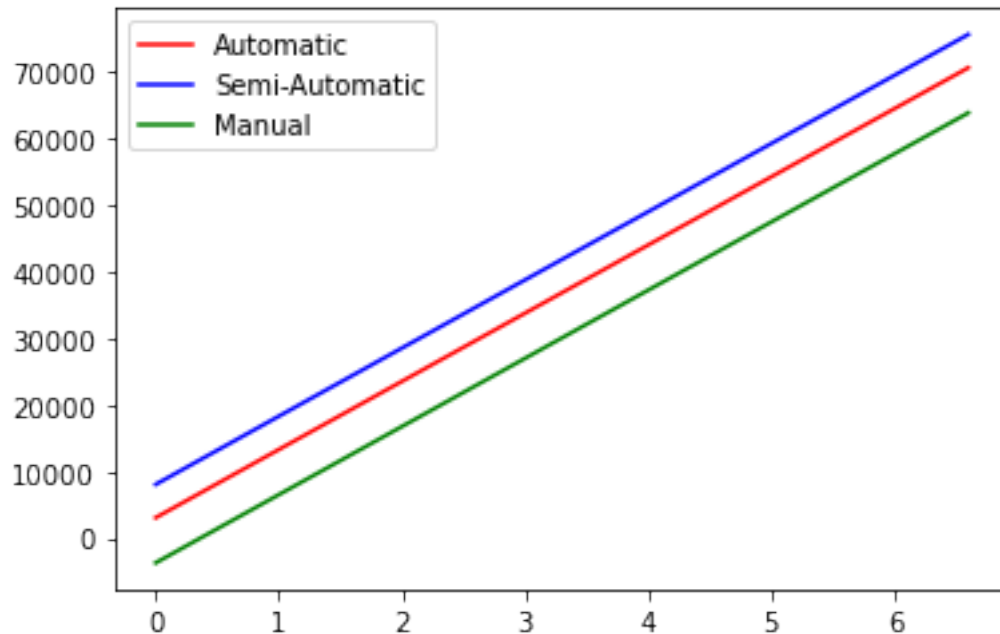
Date:	Tue, 17 Jan 2023	Prob (F-statistic):	0.00
Time:	10:10:15	Log-Likelihood:	-53644.
No. Observations:	4959	AIC:	1.073e+05
Df Residuals:	4955	BIC:	1.073e+05
Df Model:	3		
Covariance Type:	nonrobust		

	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

```
#Visualizing the developed model with interaction terms
x = np.linspace(train.engineSize.min(),train.engineSize.max(),100)
sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept'], color = 'r')
sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept']+model.params['transmission[T.Manual]'], color = 'b')
sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept']+model.params['transmission[T.Semi-Auto]'], color = 'g')
plt.legend(labels=["Automatic","Semi-Automatic", "Manual"])
```

<matplotlib.legend.Legend at 0x1cabb8deac0>



3.0.3 Including qualitative predictors and its interaction with continuous predictor 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 by multiplying `engineSize` with 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 = train2)
model = ols_object.fit()
model.summary()
```

Table 3.8: OLS Regression Results

Dep. Variable:	price	R-squared:	0.479
----------------	-------	------------	-------

Table 3.8: OLS Regression Results

Model:	OLS	Adj. R-squared:	0.478
Method:	Least Squares	F-statistic:	909.9
Date:	Tue, 17 Jan 2023	Prob (F-statistic):	0.00
Time:	01:32:37	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

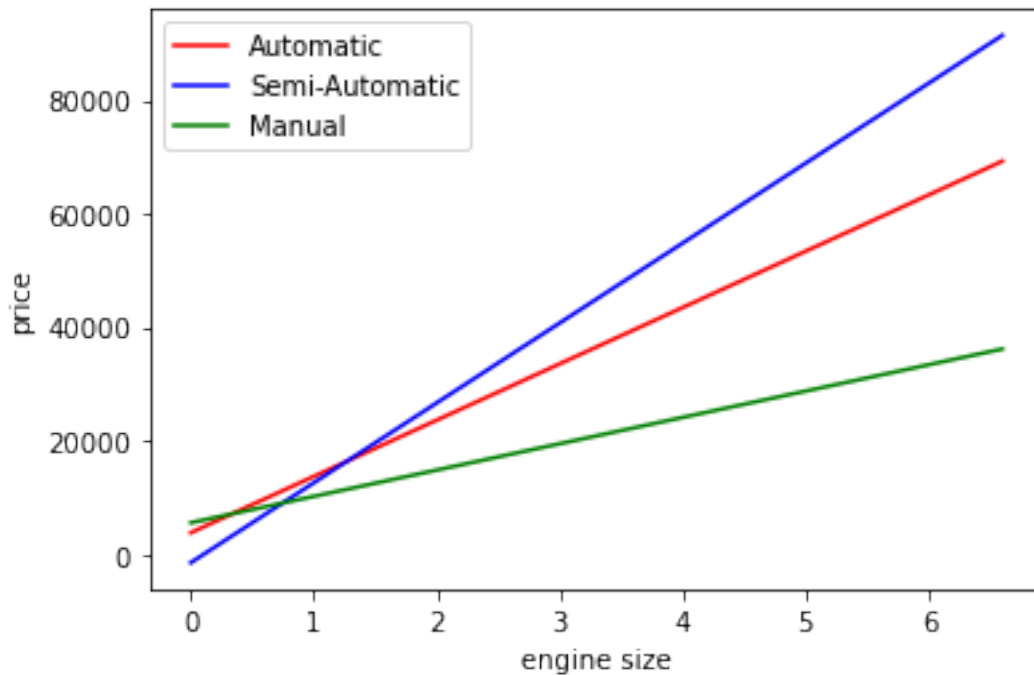
Automatic transmission: $price = 3754.7238 + 9928.6082 * engineSize$,

Semi-Automatic transmission: $price = -1527 + 7046 * engineSize$,

Manual transmission: $price = 5523 + 4642 * engineSize$,

```
#Visualizing the developed model with interaction terms
x = np.linspace(train.engineSize.min(),train.engineSize.max(),100)
plt.plot(x, model.params['engineSize']*x+model.params['Intercept'], '-r', label='Automatic')
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')
plt.ylabel('price')
plt.show()
```



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.

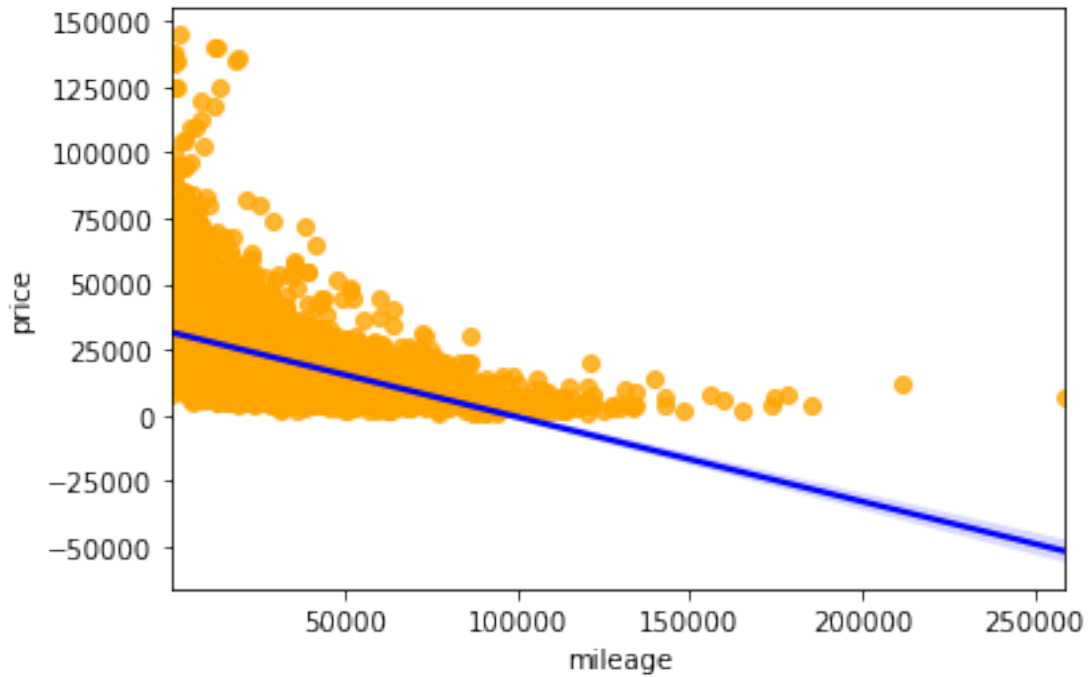
3.1 Variable transformations

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*

```
sns.regplot(x = train2.mileage, y =train2.price,color = 'orange', line_kws = {'color':'blue', 'dash':[5, 5]})
```

```
<AxesSubplot:xlabel='mileage', ylabel='price'>
```



It seems like a quadratic curve may better fit the points.

```
#Including mileage squared as a predictor and developing the model
ols_object = smf.ols(formula = 'price~mileage+I(mileage**2)', data = train2)
model = ols_object.fit()
model.summary()
```

Table 3.11: OLS Regression Results

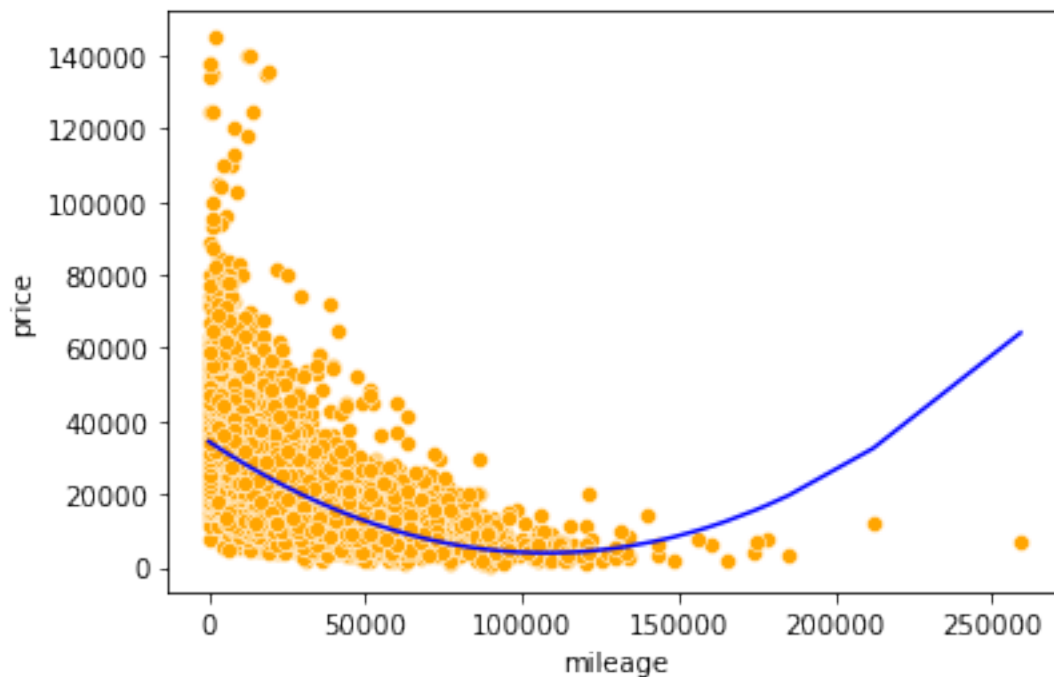
Dep. Variable:	price	R-squared:	0.271
Model:	OLS	Adj. R-squared:	0.271
Method:	Least Squares	F-statistic:	920.6
Date:	Tue, 17 Jan 2023	Prob (F-statistic):	0.00
Time:	01:32:40	Log-Likelihood:	-54382.
No. Observations:	4959	AIC:	1.088e+05
Df Residuals:	4956	BIC:	1.088e+05
Df Model:	2		
Covariance Type:	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

```
#Visualizing the regression line
pred_price = model.predict(train2)
sns.scatterplot(x = 'mileage', y = 'price', data = train2, color = 'orange')
sns.lineplot(x = train2.mileage, y = pred_price, color = 'blue')
```

<AxesSubplot:xlabel='mileage', ylabel='price'>



```
#Including mileage squared as a predictor and developing the model
ols_object = smf.ols(formula = 'price~mileage+I(mileage**2)+I(mileage**3)', data = train2)
model = ols_object.fit()
model.summary()
```

Table 3.14: 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:	Tue, 17 Jan 2023	Prob (F-statistic):	0.00
Time:	01:32:50	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

Note that the fit seems slightly better for mileage less than 150k. The model should not be used to predict car prices of cars with a mileage higher than 150k.

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 3.17: OLS Regression Results

Dep. Variable:	price	R-squared:	0.702
Model:	OLS	Adj. R-squared:	0.702
Method:	Least Squares	F-statistic:	1947.
Date:	Tue, 17 Jan 2023	Prob (F-statistic):	0.00
Time:	01:32:51	Log-Likelihood:	-52172.
No. Observations:	4960	AIC:	1.044e+05
Df Residuals:	4953	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	-755.6269	133.764	-5.649	0.000	-1017.864	-493.390
engineSize	-2.022e+06	9.72e+04	-20.807	0.000	-2.21e+06	-1.83e+06
year:engineSize	1008.6542	48.185	20.933	0.000	914.190	1103.118
mileage	-0.3548	0.014	-25.977	0.000	-0.382	-0.328
mpg	-54.7489	8.895	-6.155	0.000	-72.186	-37.311
I(mileage ** 2)	1.926e-06	1.04e-07	18.539	0.000	1.72e-06	2.13e-06

Omnibus:	2356.205	Durbin-Watson:	0.562
Prob(Omnibus):	0.000	Jarque-Bera (JB):	38343.274
Skew:	1.858	Prob(JB):	0.00
Kurtosis:	16.105	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(((testf.price - pred_price)**2).mean())
```

9074.485626191494

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.

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 assignment 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 characteristics 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 Positives (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 `Prop_license`. 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 Datasets, assignment and project files

Datasets used in the book, assignment files, and project files can be found [here](#)

References