Data Science II with python (Class notes)

STAT 303-2

Arvind Krishna

9/20/22

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.

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()
```

```
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:	01:06:42	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

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

Table 1.5: 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

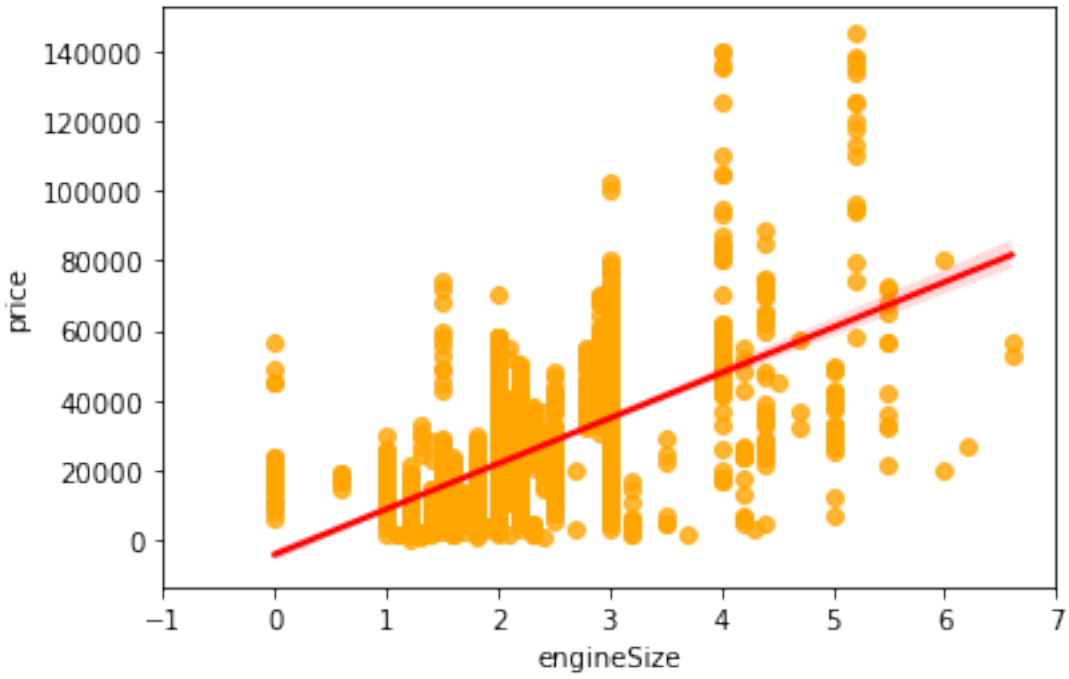
Omnibus:	1271.986	Durbin-Watson:	0.517
Prob(Omnibus):	0.000	Jarque-Bera (JB):	6490.719
Skew:	1.137	Prob(JB):	0.00
Kurtosis:	8.122	Cond. No.	7.64

The model equation is: 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*

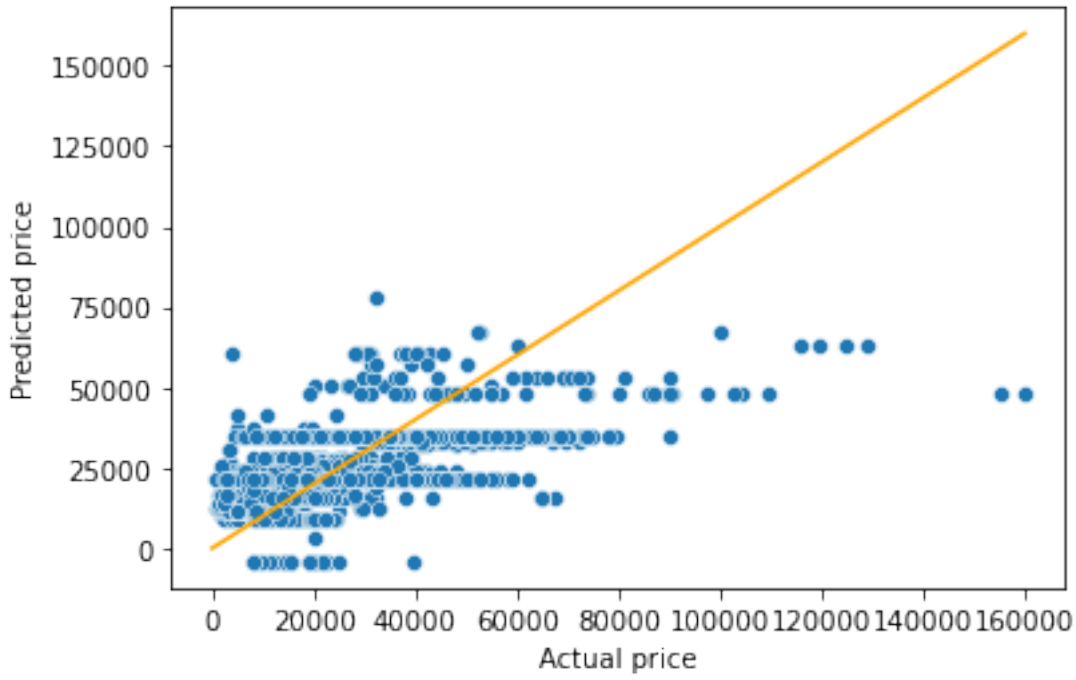
```
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
```

```
#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, it's performance on unknonwn 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() fucntion 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

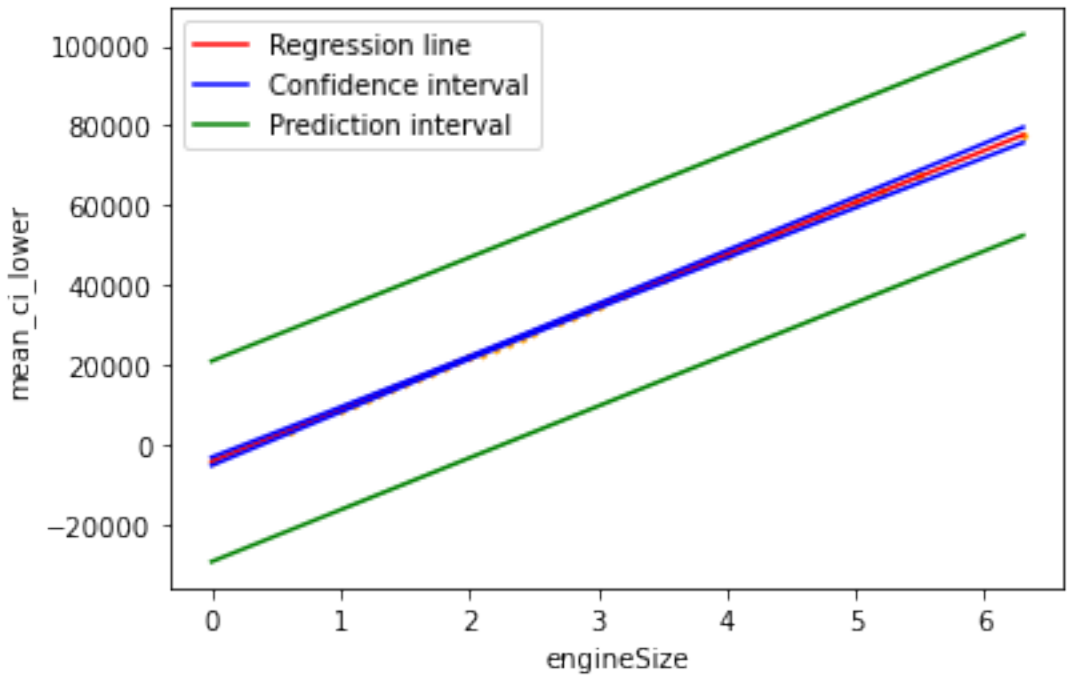
	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
2668	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
2669	8866.245277	316.580850	8245.606701	9486.883853	-16254.905974	33987.396528
2670	21854.526298	184.135754	21493.538727	22215.513869	-3261.551421	46970.604017
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 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')
```

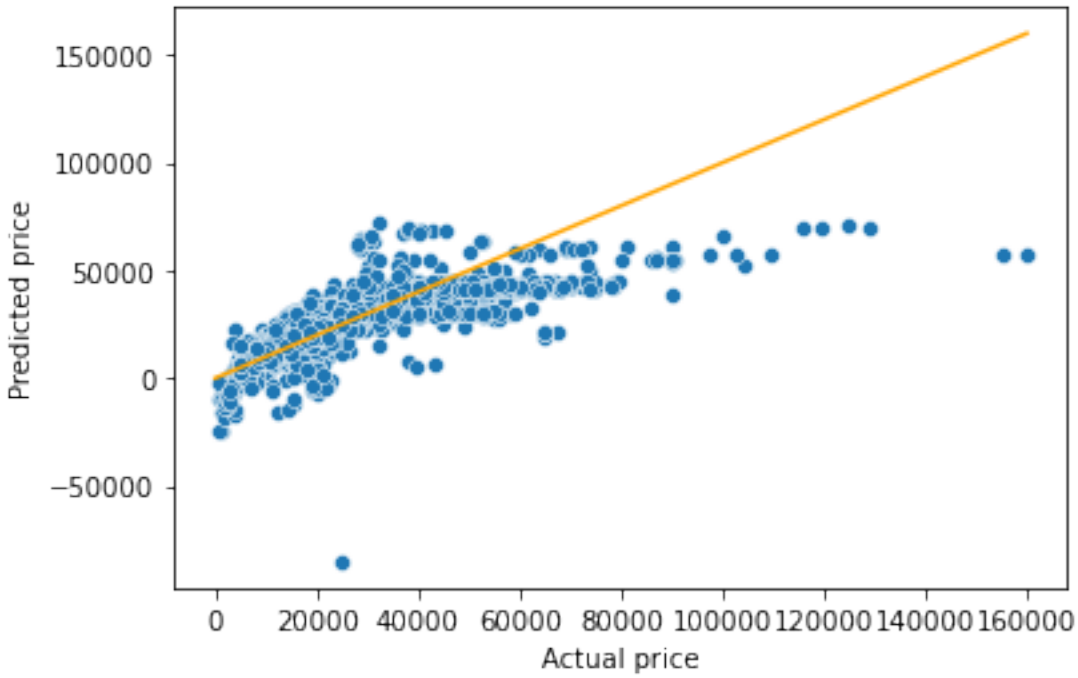
```
#Using the predict() function associated with the 'model' object to make predictions of ca
pred_price = model.predict(testf)#Note that the predict() function finds the predictor 'en
```

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
```

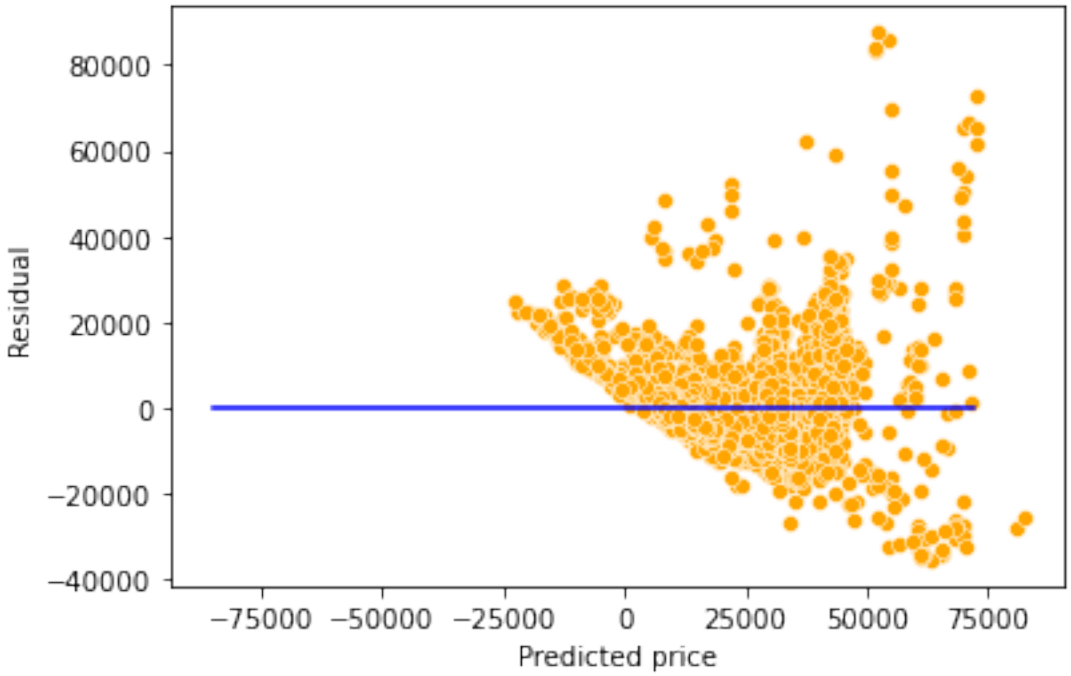
What is the residual standard error based on the training data?

```
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.

A Assignment 1 (Control flow)

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 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. There are 5 points for clealiness and organization. The code should be commented and clearly written with intuitive variable names. For example, use variable names such as number_input, factor, hours, instead of a,b,xyz, etc.
6. The assignment is worth 100 points, and is due on **5th October 2022 at 11:59 pm**.

B 1

B.1 1(a)

You look at the clock and it is exactly 2pm. You set an alarm to go off in 510 hours. At what time does the alarm go off? If the answer is say, 4 pm, then your code should print - "The alarm goes off at 4 pm".

(5 points)

B.2 1(b)

Write a program to solve the general version of the above problem. Ask the user for - (1) the time now (in hours), and (2) the number of hours for the alarm to go off. Your program should output the time at which the alarm goes off. Both the user inputs must be in {0, 1, 2..., 22, 23}. If the answer is, say 14:00 hours, then your program should print - "The alarm goes off at 14:00 hours.

Show the output of your program when the user inputs 7 as the current time, and 95 as the number of hours for the alarm to go off.

(10 points)

C 2

C.1 2(a)

Write a program that checks if a positive integer is prime or not. Show the output when the program is used to check if 89 is prime or not.

(5 points)

C.2 2(b)

Prompt the user to input a positive integer. Write a program that prints the **factors** of the positive integer input by the user. Show the output of the program if the user inputs 190.

(10 points)

C.3 2(c)

Prompt the user to input a positive integer. Update the program in 2(b) to print the **prime factors** of the positive integer input by the user. Show the output of the program if the user inputs 190.

(15 points)

C.4 2(d)

Update the program in 2(c), so that it prints "Incorrect input, please enter positive integer" if the user does not enter a positive integer, and then prompts the user to input a positive integer. The program should continue to prompt the user to enter a positive integer until the user successfully enters a positive integer. Show the output of the program if the user enters "seventy" in the first attempt, "#70" in the second attempt, and 70 in the third attempt.

(30 points)

C.5 3

Prompt the user to input an english sentence. Write a program that counts and prints the number of words in the sentence input by the user. The program should continue to run until the user inputs the sentence - “end program”

(20 points)

D Assignment 2 (Functions)

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. There are 5 points for clealiness and organization. The code should be commented and clearly written with intuitive variable names. For example, use variable names such as number_input, factor, hours, instead of a,b,xyz, etc.
6. The assignment is worth 100 points, and is due on **14th October 2022 at 11:59 pm**.

D.1 1(a)

Write a function that accepts a word, and a sentence as arguments, and returns the number of times the word occurs in the sentence.

Call the function, and print the returned value if the word is "*sea*", and the sentence is "*She sells sea shells on the sea shore when the sea is calm.*" Note that this is just an example to check your function. Your function should work for any word and sentence.

(10 points)

D.2 1(b)

Ask the user to input a sentence. Use the function in 1(a) to find the word that occurs the maximum number of times in the sentence. Print the word and its number of occurences. If multiple words occur the maximum number of times, then you can print any one of them.

Check your program when the user inputs the sentence, "*She sells sea shells on the sea shore when the sea is calm.*". Your program must print, "*The word with the maximum number of occurences is 'sea' and it occurs 3 times.*" Note that this is just an example to check your program. Your program must work for any sentence"

(20 points)

D.3 2(a)

Write a function that checks if an integer is prime. The function must accept the integer as an argument, and return `True` if the integer is prime, otherwise it must return `False`.

Call your function with the argument as `197`.

(4 points)

D.4 2(b)

Write a function that checks if an integer is a factor of another integer. The function must accept both the integers as arguments, and return `True` if the integer is a factor, otherwise it must return `False`.

Call your function with the arguments as `(19,85)`.

(3 points)

D.5 2(c)

Prompt the user to input a positive integer. Use the functions in 2(a) and 2(b) to print the prime factors of the integer. Your program should be no more than 4 lines (excluding the comments)

Check your program is the user inputs 190

(8 points)

D.6 3(a)

The tuple below named as `tuple_of_words` consists of words. Write a function that accepts a word, say `word_to_search` and the `tuple_of_words` as arguments, and finds if the `word_to_search` occurs in the `tuple_of_words` or not. This is very simple to do with the code `word_to_search in tuple_of_words`. However, this code is unfortunately very slow.

As the words in the `tuple_of_words` are already sorted in alphabetical order, we can search using a faster way, called binary search. To implement binary search in a function, start by comparing `word_to_search` with the middle entry in the `tuple_of_words`. If they are equal, then you are done and the function should return `True`. On the other hand, if the `word_to_search` comes before the middle entry, then search the first half of `tuple_of_words`. If it comes after the middle entry, then search the second half of `tuple_of_words`. Then repeat the process on the appropriate half of the `tuple_of_words` and continue until the word is found or there is nothing left to search, in which case the function short return `False`. The `<` and `>` operators can be used to alphabetically compare two strings.

You may write just one function or multiple functions to solve this problem.

Check your function if the `word_to_search` is:

1. `'rocket'`
2. `'rest'`

(25 points)

D.7 3(b)

Update the function in 3(a) to also print the number of iterations it took to find the `word_to_search` or fail to find the `word_to_search`.

Check your function if the `word_to_search` is:

1. `'rocket'`
2. `'rest'`

(10 points)

D.8 3(c)

Update the function in 3(b) to also print the index of `word_to_search` in `tuple_of_words` if the word is found in the tuple. For example, the index of 'abacus' is 0, the index of 'abdomen' is 1, and so on.

Check your function if the `'word_to_search'` is:

1. `'rocket'`
2. `'rest'`

(10 points)

D.9 3(d)

What is the maximum number of iterations it may take for your function to search the `word_to_search`. You may either write a program to answer this question, or answer it analytically.

(5 points)

E Assignment 3 (Data structures)

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. There are 5 points for clealiness and organization. The code should be commented and clearly written with intuitive variable names. For example, use variable names such as number_input, factor, hours, instead of a,b,xyz, etc.
6. The assignment is worth 100 points, and is due on **21st October 2022 at 11:59 pm**.

F 1 GDP of The USA

USA's GDP per capita from 1960 to 2021 is given by the tuple `T` in the code cell below. The values are arranged in ascending order of the year, i.e., the first value is for 1960, the second value is for 1961, and so on.

```
T = (3007, 3067, 3244, 3375,3574, 3828, 4146, 4336, 4696, 5032,5234,5609,6094,6726,7226,78
```

F.1 1(a)

F.1.1 1(a)(i)

Use list comprehension to produce a list of the gaps between consecutive entries in `T`, i.e, the increase in GDP per capita with respect to the previous year. The list with gaps should look like: [60, 177, ...].

(6 points)

F.1.2 1(a)(ii)

Use the list developed in *1(a)(i)* to find the maximum gap size, i.e, the maximum increase in GDP per capita.

(2 points)

F.1.3 1(a)(iii)

Using list comprehension with the list developed in *1(a)(i)*, find the percentage of gaps that have size greater than $1000.

(6 points)

F.2 1(b)

F.2.1 1(b)(i)

Create a dictionary `D`, where the `key` is the year, and `value` for the `key` is the increase in GDP per capita in that year with respect to the previous year, i.e., the gaps computed in *1(a)(i)*.

(6 points)

F.2.2 1(b)(ii)

Use the dictionary `D` to find the year when the GDP per capita increase was the maximum as compared to the previous year. Use the list comprehension method.

(6 points)

Hint: […… for …. in D.items() if ……]

F.2.3 1(b)(iii)

Use the dictionary `D` to find the years when the GDP per capita decreased with respect to the previous year. Use the list comprehension method.

(6 points)

G 2 Ted Talks

G.1 2(a)

Read the file *TED_Talks.json* on ted talks using the code below. You will get the data in the object `TED_Talks_data`. Just look at the data structure of `TED_Talks_data`. You will need to know how the data is structured in lists/dictionaries to answer the questions below.

(2 points)

```
import json
with open("TED_Talks.json", "r") as file:
    TED_Talks_data=json.load(file)
```

G.2 2(b)

Find the number of talks in the dataset.

(2 points)

G.3 2(c)

Find the `headline`, `speaker` and `year_filmed` of the talk with the highest number of `views`.

(6 points)

G.4 2(d)

What are the mean and median number of `views` for a talk? Can we say that the majority of talks (i.e., more than 50% of the talks) have less `views` than the average number of `views` for a talk? Justify your answer.

(6 points)

G.5 2(e)

Do at least 25% of the talks have more `views` than the average number of `views` for a talk? Justify your answer.

(4 points)

G.6 2(f)

Find the `headline` of the talk that received the highest number of votes in the *Confusing* category.

(8 points)

G.7 2(g)

Find the `headline` and the `year_filmed` of the talk that received the highest percentage of votes in the *Fascinating* category.

$$\text{Percentage of } \textit{Fascinating} \text{ votes for a ted talk} = \frac{Number\ of\ votes\ in\ the\ Fascinating\ category}{Total\ votes\ in\ all\ categories}$$

(10 points)

H 3 Poker

The object `deck` defined below corresponds to a deck of cards. Estimate the probability that a five card hand will be:

1. Straight
2. Three-of-a-kind
3. Two-pair
4. One-pair
5. High card

You may check the meaning of the above terms here.

(25 points)

Hint:

Estimate these probabilities as follows.

1. Write a function that accepts a hand of 5 cards as argument, and returns relevant characterisitics of a hand, such as the number of distinct card values, maximum occurences of a value etc. Using the values returned by this function (may be in a dictionary), you can compute if the hand is of any of the above types *(Straight / Three-of-a-kind / two-pair / one-pair / high card)*.
2. Randomly pull a hand of 5 cards from the `deck`. Call the function developed in (1) to get the relevant characteristics of the hand. Use those characteristics to determine if the hand is one of the five mentioned types *(Straight / Three-of-a-kind / two-pair / one-pair / high card)*.
3. Repeat (2) 10,000 times.
4. Estimate the probability of the hand being of the above five mentioned types *(Straight / Three-of-a-kind / two-pair / one-pair / high card)* from the results of the 10,000 simulations.

You may use the function `shuffle()` from the library `random` to shuffle the deck everytime before pulling a hand of 5 cards.

You don't need to stick to the hint if you feel you have a better way to do it. In case you have a better way, you can claim 10 bonus points for this assignment.

```
deck = [{'value':i, 'suit':c}
for c in ['spades', 'clubs', 'hearts', 'diamonds']
for i in range(2,15)]
```

I Assignment 4 (Classes and Objects)

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. There are 6 points for clealiness and organization. The code should be commented and clearly written with intuitive variable names. For example, use variable names such as number_input, factor, hours, instead of a,b,xyz, etc.
6. The assignment is worth 100 points, and is due on **5th November 2022 at 11:59 pm**.
7. There is a Bonus question worth 25 points. You can score 125 out of 100 in the assignment if all is correct. However, there is no partial credit for the bonus question.

I.1 Password Manager

Define a class named `PasswordManagerUpdated` that inherits the class `PasswordManager` defined in Practice exercise 1. The class `PasswordManagerUpdated` should have two methods, other than the constructor:

The method `set_password(new_password)` that sets a new password. The new password must only be accepted if it does not have any punctuations in it, and if it is not the same as one of the old passwords. If the new password is not acceptable, then one of the appropriate messages should be printed - (a) Cannot have punctuation in password, try again, or (b) Old password cannot be reused, try again. The new password is a parameter to this method.

The method `suggest_password()` that randomly sets and returns a password as a string comprising of 15 randomly chosen letters of the English language. Letters may be repeated as well.

```
past_passwords = ['titanic1911','ibiza1972','montecarlo799']
```

Hint:

1. For generating letters, use str.lowercase()
2. For randomly selecting letters, use random.randint()
3. For iterating through punctuations, use `from string import punctuation`.

After defining the class, instantiate an object of this class with `past_passwords`. Then:

I.1.1

Check the attribute `old_passwords`.

(2 points)

I.1.2

Check the method `get_password()`

(2 points)

I.1.3

Try re-setting the password to 'ibiza1972', and then check the current password.

(4 points)

I.1.4

Try re-setting the password to 'oktoberfest%2022', and then check the current password.

(6 points)

I.1.5

Try re-setting the password to 'oktoberfest2022', and then check the current password.

(2 points)

I.1.6

Use the method `suggest_password()` to set a new password.

(6 points)

I.1.7

Check the `is_correct()` method by entering the correct password as set in the previous question.

(2 points)

I.2 SMS store manager

Create a new class, named as *SMS_store_manager*. This class will be used to store and manage SMSs of a person's cellphone. An object of this class, say *my_inbox*, will be initialized with a list of existing messages. The class attribute `messages` will store the existing messages during instantiation:

```
my_inbox(existing_messages)
```

Each message in the list will be represented as a dictionary. A sample message is shown below:

```
{'has_been_viewed':True, 'from_number':9348593356, time_arrived='19:50', 'date':'2022-10-2
```

The class should provide these methods:

```
my_inbox.add_new_arrival(from_number, time_arrived, text_of_SMS)
#Makes new SMS dictionary, inserts it after other messages
#in the store, i.e, in the list of messages. When creating this message, its
#'has_been_viewed' status is set False.

my_inbox.message_count()
#Returns the number of sms messages in my_inbox

my_inbox.get_unread_messages()
#Returns unread messages, i.e., messages with 'has_been_viewed' status as False
#Also changes the status of 'has_been_viewed' to True for all messages returned
#While returning unread messages, the 'has_been_viewed' status must not be returned
```

```
my_inbox.delete(i)     # Delete the message at index i
my_inbox.clear()       # Delete all messages from inbox
```

Once you define the class, instantiate an object of this class, and call it *harry_messages*. Initialize the object with the existing messages below.

```
existing_messages = [{'has_been_viewed':False, 'from_number':8769038451, 'time':'09:30','d
                      {'has_been_viewed':False, 'from_number':9579038373, 'time':'19:30'
                      {'has_been_viewed':True, 'from_number':8639568726, 'time':'10:30',
                      {'has_been_viewed':False, 'from_number':4567653456, 'time':'11:50'
                      {'has_been_viewed':False, 'from_number':5646786643, 'time':'18:50'
                      {'has_been_viewed':False, 'from_number':9845543492, 'time':'17:10'
                      {'has_been_viewed':True, 'from_number':8793450987, 'time':'13:20',
                      {'has_been_viewed':True, 'from_number':874556445, 'time':'07:20','
                      {'has_been_viewed':True, 'from_number':998456435, 'time':'07:20','
                      {'has_been_viewed':True, 'from_number':8769038451, 'time':'07:20',
                      {'has_been_viewed':True, 'from_number':7739984533, 'time':'07:20',
                      {'has_been_viewed':True, 'from_number':3443498738, 'time':'07:20',
```

Use the object *harry_messages* to:

I.2.1

Add a new message below:

```
from_number=8749373884;
time='07:25';
date='2022-10-29'
text_of_SMS='Hey, I want my bike back.'
```

(4 points)

I.2.2

Count the number of messages.

(2 points)

I.2.3

Print all the unread messages.

(6 points)

I.2.4

Try printing all the unread messages again *(Nothing should show up as all messages were read in in the previous question).*

(4 points)

I.2.5

Clear the inbox, and then count the number of messages.

(4 points)

I.3 Personalized SMS store manager

Inherit the class developed in the previous question to create a new class *SMS_personalized_store_manager*. This class will inherit all the methods of the class *SMS_store_manager*. However, it will have the following differences as compared to the parent class:

1. Instantiation

    a. During instantiation, it will initialize two additional attributes - `spam_words`, and `update_words`, along with the `messages` attribute which stores existing messages. Use the lists below to initialize the attributes of the class.

    b. Each message will be tagged as `spam`, `update`, or `personal` as follows. If the message contains any word / phrase in the list `spam_words`, then it will be tagged as `spam`. If the messege is not tagged as spam, and it contains any word / phrase in the list `update_words`, then it will be tagged as `update`. If the message has not been tagged as `spam` or `update`, then it will be tagged as `personal`. The tag will appear as an additional key-value pair in the dictionary of each message, where the key will be `tag`, and the value will be `'spam'`, `'update'` or `'personal'`.

A message after tagging may look like:

```
{'has_been_viewed':True, 'from_number':9348593356, time_arrived='19:50', 'date':'2022-10-2
```

```
#Lists for initializing the attributes of the class
existing_messages = [{'has_been_viewed':False, 'from_number':8769038451, 'time':'09:30','d
                     {'has_been_viewed':False, 'from_number':9579038373, 'time':'19:30'
                     {'has_been_viewed':True, 'from_number':8639568726, 'time':'10:30',
                     {'has_been_viewed':False, 'from_number':4567653456, 'time':'11:50'
                     {'has_been_viewed':False, 'from_number':5646786643, 'time':'18:50'
                     {'has_been_viewed':False, 'from_number':9845543492, 'time':'17:10'
                     {'has_been_viewed':True, 'from_number':8793450987, 'time':'13:20',
                     {'has_been_viewed':True, 'from_number':874556445, 'time':'07:20','
                     {'has_been_viewed':True, 'from_number':998456435, 'time':'07:20','
                     {'has_been_viewed':True, 'from_number':8769038451, 'time':'07:20',
                     {'has_been_viewed':True, 'from_number':7739984533, 'time':'07:20',
                     {'has_been_viewed':True, 'from_number':3443498738, 'time':'07:20',
spam_words=['100% more', '100% free', '100% satisfied', 'Additional income', 'Be your own
update_words = ['Your order', 'appointment', 'Reservation confirmed', 'Card Not Present',
```

2. The class will have two additional methods:

    a. `get_unread_messages_by_category()`: This method will return all the **unread** messages of a particular category, i.e, `'spam'`, `'update'` or `'personal'`. The method will accept the category as an argument. If no argument is specified by the user for the category, then all the **unread** messages must be displayed. Once unread messages are returned, they will be marked as read. While returning unread messages, the `has_been_viewed` status must be changed to `True`, but the status itself must not be returned.

    b. `get_messages_by_category()`: This method will return all the messages (both read or unread) of a particular category, i.e, `'spam'`, `'update'` or `'personal'`. The method will accept the category as an argument. If no argument is specified by the user for the category, then all the messages must be returned.

3. The class will modify the method `add_new_arrival(self,from_number, time, date, text_of_SMS)` of the parent class to tag a new message as `'spam'`, `'update'` or `'personal'`.

Once you define the class, instantiate an object of this class with the lists `existing_messages`, `spam_words` and `update_words`, and call it *ron_messages*:

I.3.1

Add a new message below:

```
from_number=8749373884;
time='07:25';
date='2022-10-29'
text_of_SMS='Hey, I want my bike back.'
```

(6 points)

I.3.2

Print all the unread messages tagged as `'personal'`. Use the method `get_unread_messages_by_category()`.

(10 points)

I.3.3

Print all the unread messages tagged as `'update'`. Use the method `get_unread_messages_by_category()`.

(10 points)

I.3.4

Print all the unread messages. Use the method `get_unread_messages()`.

(10 points)

I.3.5

Print all the messages tagged as `'spam'`. Use the method `get_messages_by_category()`.

(7 points)

I.3.6

Print all the messages tagged as `'update'`. Use the method `get_messages_by_category()`.

(7 points)

I.4 Bonus question: Creating a new datatype - list of dictionaries

Read *movie_data* with the code below.

```
import json
with open("movies.json", encoding="utf8") as file:
    movie_data=json.load(file)
```

Inherit the in-built python class `list()` to create a new class `list_dict()`. This class will be used for objects that are a list of dictionaries, where all the dictionaries in the list have the same keys. Add a method in this class, named as `sort_by_dict_value()` that sorts the dictionaries of the list based on the values of the desired key in the dictionaries. Sorting can be done in ascending or descending order depending on the user. The key to be used for sorting and the order (ascending / descending) will be parameters to the method `sort_by_dict_value()`. If the sorting order is unspecified, use ascending as default.

Instantiate an object of the class `list_dict()` with `movie_data`.

If the name of the object is `mov`, then the method `sort_by_dict_value()` may be called as: mov.sort_by_dict_value(movie_parameter, ascending = True)

where `movie_parameter` can be any key of the dictionaries, using the values of which the list of dictionaries has to be sorted.

I.4.1

Use the method `sort_by_dict_value()` to sort the list of dictionaries in increasing order of `Production Budget`. What is the name of the 45th movie in the sorted list of dictionaries?

I.4.2

Use the method `sort_by_dict_value()` to sort the list of dictionaries in decreasing order of `Worldwide Gross`. What is the name of the 2nd movie in the sorted list of dictionaries?

(25 points - No partial credit)

J Datasets

Datasets used in the book can be found here