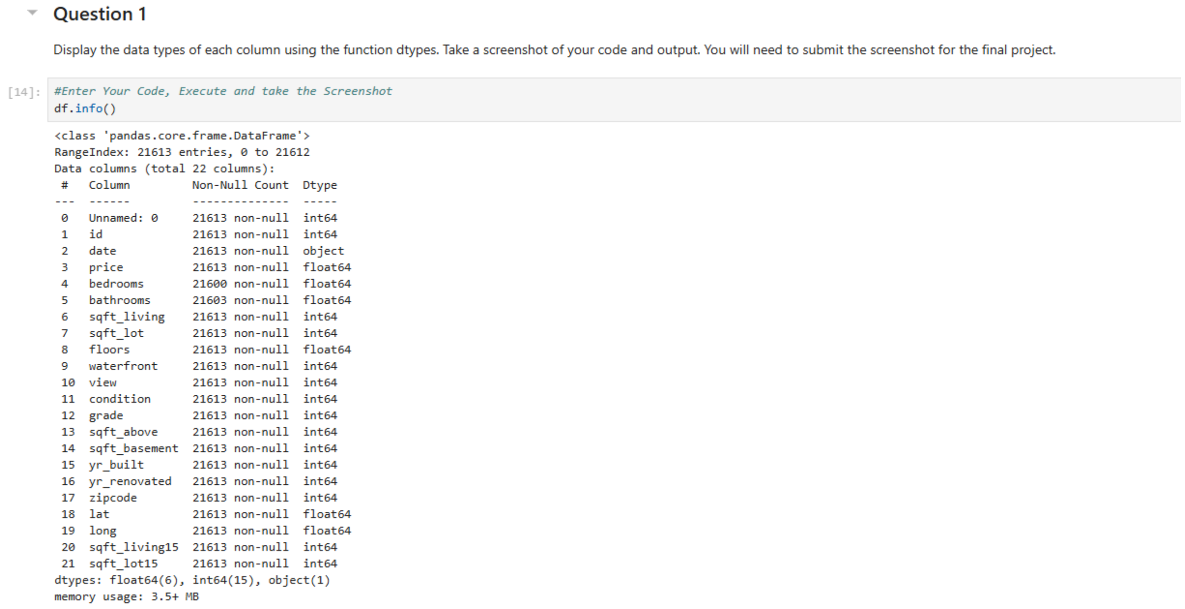
QUESTION 1



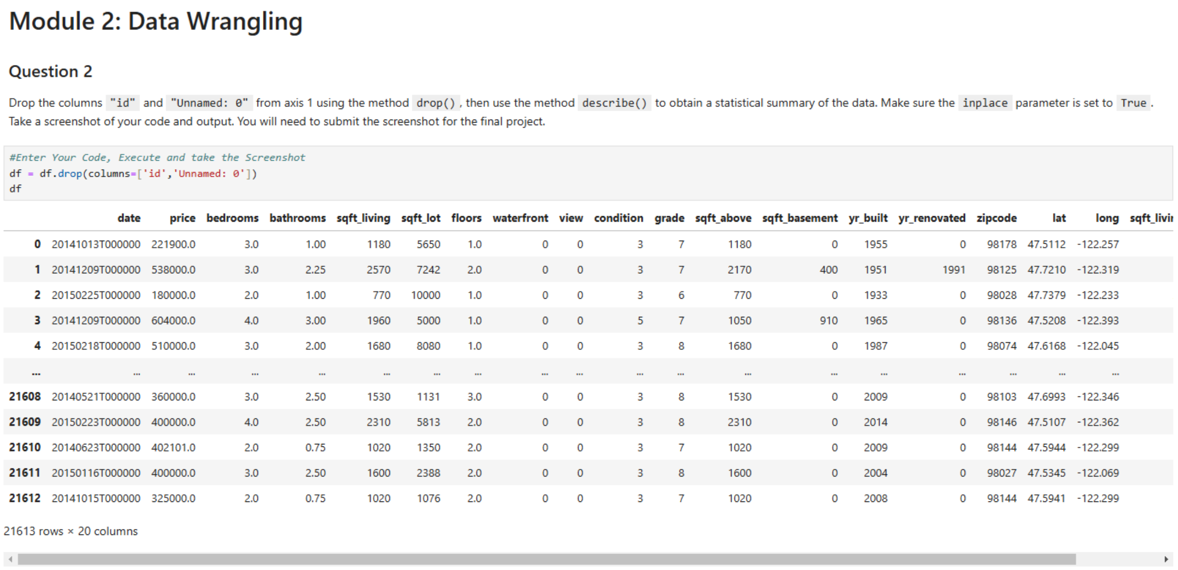
Question 1

Display the data types of each column using the function dtypes. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
[14]: #Enter Your Code, Execute and take the Screenshot
      df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Unnamed: 0     21613 non-null  int64  
 1   id             21613 non-null  int64  
 2   date           21613 non-null  object 
 3   price          21613 non-null  float64
 4   bedrooms       21600 non-null  float64
 5   bathrooms      21603 non-null  float64
 6   sqft_living    21613 non-null  int64  
 7   sqft_lot       21613 non-null  int64  
 8   floors         21613 non-null  float64
 9   waterfront     21613 non-null  int64  
 10  view           21613 non-null  int64  
 11  condition      21613 non-null  int64  
 12  grade          21613 non-null  int64  
 13  sqft_above     21613 non-null  int64  
 14  sqft_basement  21613 non-null  int64  
 15  yr_built       21613 non-null  int64  
 16  yr_renovated   21613 non-null  int64  
 17  zipcode        21613 non-null  int64  
 18  lat            21613 non-null  float64
 19  long           21613 non-null  float64
 20  sqft_living15  21613 non-null  int64  
 21  sqft_lot15     21613 non-null  int64  
dtypes: float64(6), int64(15), object(1)
memory usage: 3.5+ MB
```

QUESTION 2



Module 2: Data Wrangling

Question 2

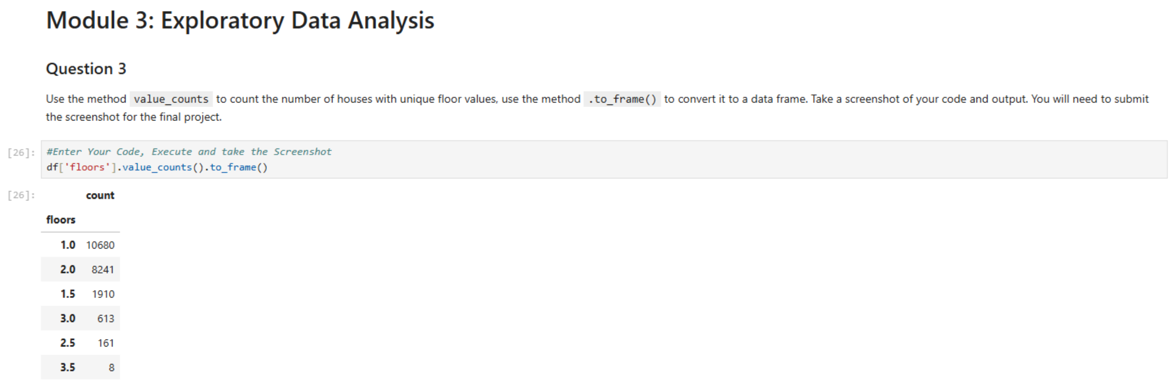
Drop the columns `"id"` and `"Unnamed: 0"` from axis 1 using the method `drop()`, then use the method `describe()` to obtain a statistical summary of the data. Make sure the `inplace` parameter is set to `True`. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
#Enter Your Code, Execute and take the Screenshot
df = df.drop(columns=['id','Unnamed: 0'])
df
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_livi
0	20141013T000000	221900.0	3.0	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955	0	98178	47.5112	-122.257	
1	20141209T000000	538000.0	3.0	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951	1991	98125	47.7210	-122.319	
2	20150225T000000	180000.0	2.0	1.00	770	10000	1.0	0	0	3	6	770	0	1933	0	98028	47.7379	-122.233	
3	20141209T000000	604000.0	4.0	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965	0	98136	47.5208	-122.393	
4	20150218T000000	510000.0	3.0	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987	0	98074	47.6168	-122.045	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21608	20140521T000000	360000.0	3.0	2.50	1530	1131	3.0	0	0	3	8	1530	0	2009	0	98103	47.6993	-122.346	
21609	20150223T000000	400000.0	4.0	2.50	2310	5813	2.0	0	0	3	8	2310	0	2014	0	98146	47.5107	-122.362	
21610	20140623T000000	402101.0	2.0	0.75	1020	1350	2.0	0	0	3	7	1020	0	2009	0	98144	47.5944	-122.299	
21611	20150116T000000	400000.0	3.0	2.50	1600	2388	2.0	0	0	3	8	1600	0	2004	0	98027	47.5345	-122.069	
21612	20141015T000000	325000.0	2.0	0.75	1020	1076	2.0	0	0	3	7	1020	0	2008	0	98144	47.5941	-122.299	

21613 rows × 20 columns

QUESTION 3



Module 3: Exploratory Data Analysis

Question 3

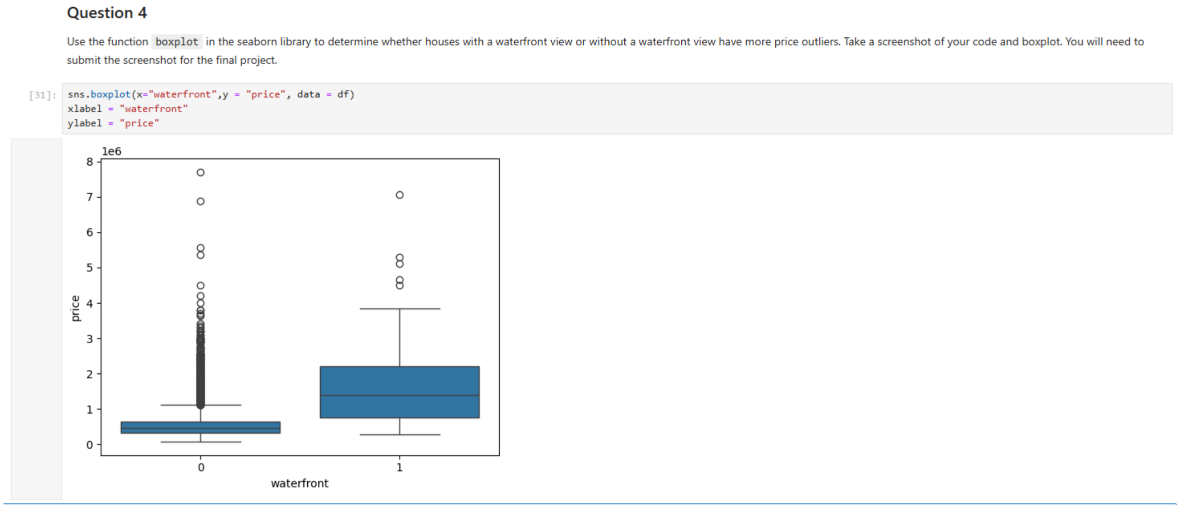
Use the method `value_counts` to count the number of houses with unique floor values, use the method `.to_frame()` to convert it to a data frame. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
[26]: #Enter Your Code, Execute and take the Screenshot
      df['floors'].value_counts().to_frame()
```

[26]:

	count
floors	
1.0	10680
2.0	8241
1.5	1910
3.0	613
2.5	161
3.5	8

QUESTION 4

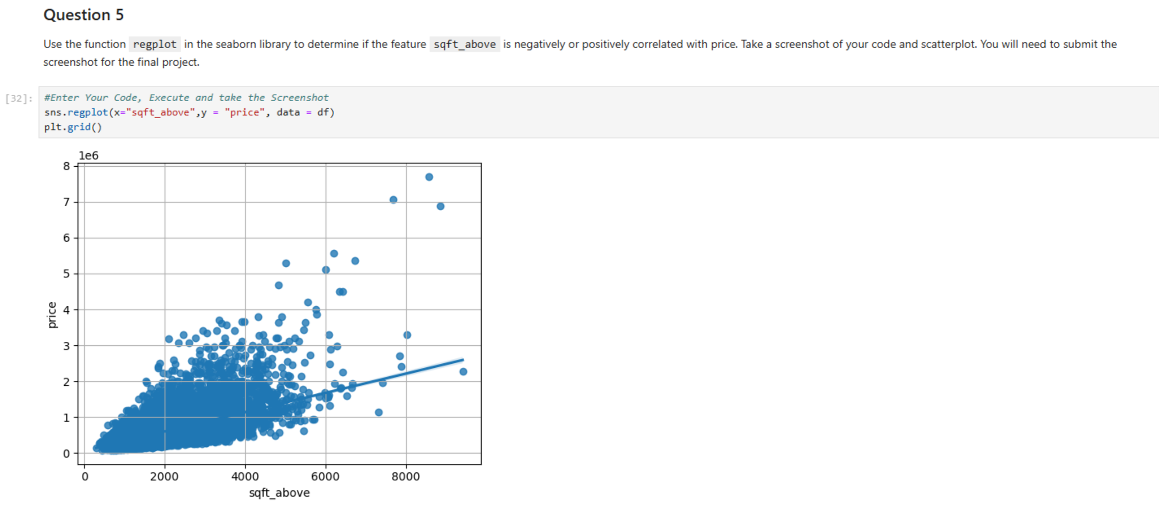


Question 4

Use the function `boxplot` in the seaborn library to determine whether houses with a waterfront view or without a waterfront view have more price outliers. Take a screenshot of your code and boxplot. You will need to submit the screenshot for the final project.

```
[31]: sns.boxplot(x="waterfront",y = "price", data = df)
      xlabel = "waterfront"
      ylabel = "price"
```

QUESTION 5

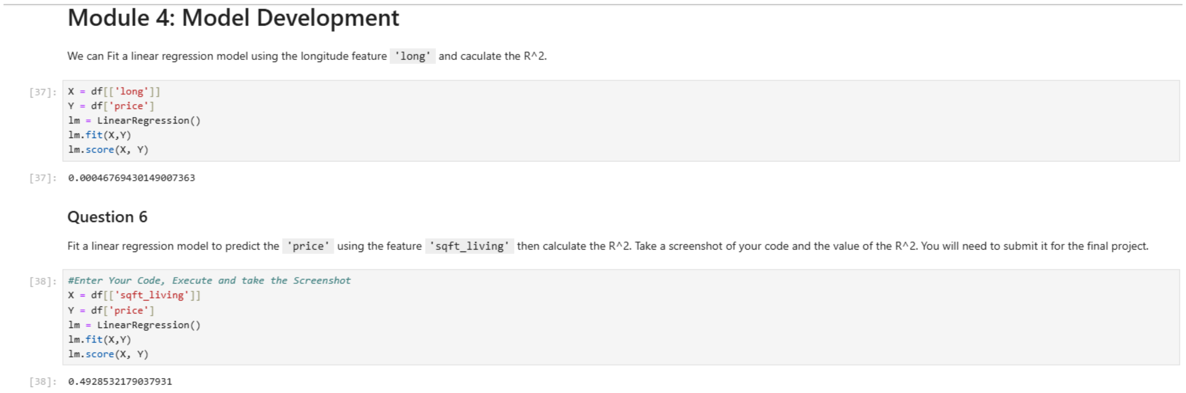


Question 5

Use the function `regplot` in the seaborn library to determine if the feature `sqft_above` is negatively or positively correlated with price. Take a screenshot of your code and scatterplot. You will need to submit the screenshot for the final project.

```
[32]: #Enter Your Code, Execute and take the Screenshot
      sns.regplot(x="sqft_above",y = "price", data = df)
      plt.grid()
```

QUESTION 6



Module 4: Model Development

We can Fit a linear regression model using the longitude feature `'long'` and caculate the R^2.

```
[37]: X = df[['long']]
      Y = df['price']
      lm = LinearRegression()
      lm.fit(X,Y)
      lm.score(X, Y)

[37]: 0.00046769430149007363
```

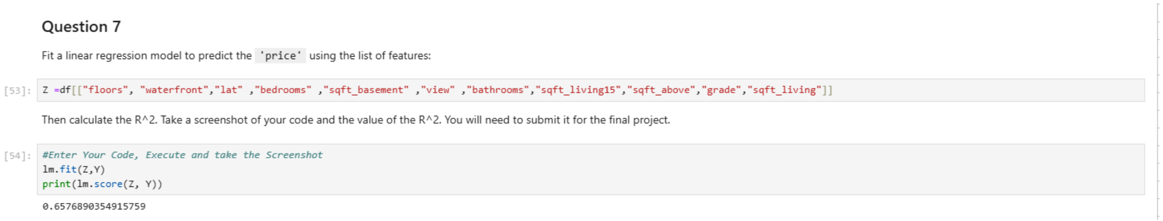
Question 6

Fit a linear regression model to predict the `'price'` using the feature `'sqft_living'` then calculate the R^2. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
[38]: #Enter Your Code, Execute and take the Screenshot
      X = df[['sqft_living']]
      Y = df['price']
      lm = LinearRegression()
      lm.fit(X,Y)
      lm.score(X, Y)

[38]: 0.4928532179037931
```

QUESTION 7



Question 7

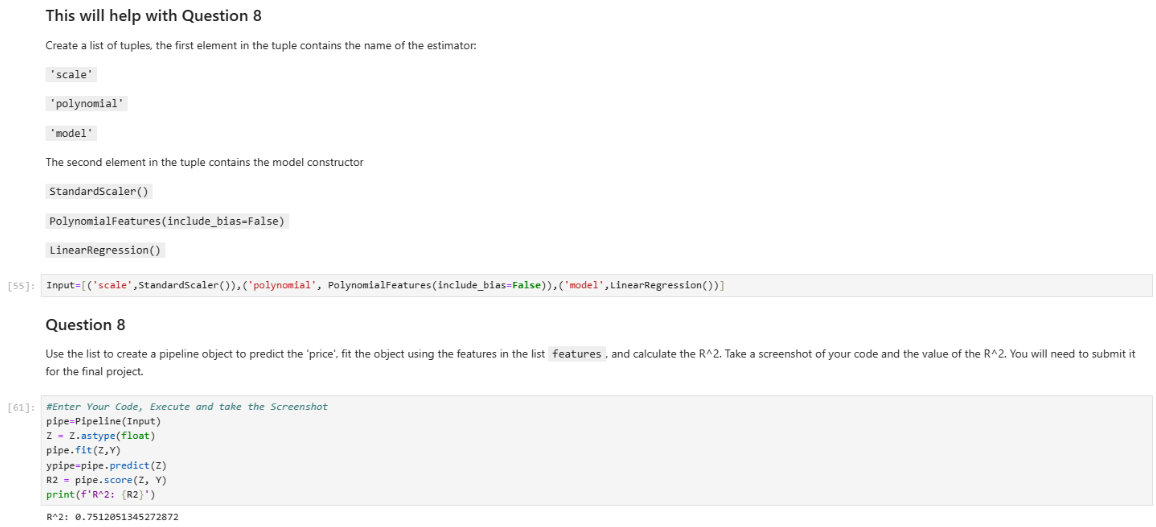
Fit a linear regression model to predict the `'price'` using the list of features:

```
[53]: Z =df[["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]]
```

Then calculate the R^2. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
[54]: #Enter Your Code, Execute and take the Screenshot
      lm.fit(Z,Y)
      print(lm.score(Z, Y))
      0.6576890354915759
```

QUESTION 8



This will help with Question 8

Create a list of tuples, the first element in the tuple contains the name of the estimator:

`'scale'`

`'polynomial'`

`'model'`

The second element in the tuple contains the model constructor

`StandardScaler()`

`PolynomialFeatures(include_bias=False)`

`LinearRegression()`

```
[55]: Input=[('scale',StandardScaler()),('polynomial', PolynomialFeatures(include_bias=False)),('model',LinearRegression())]
```

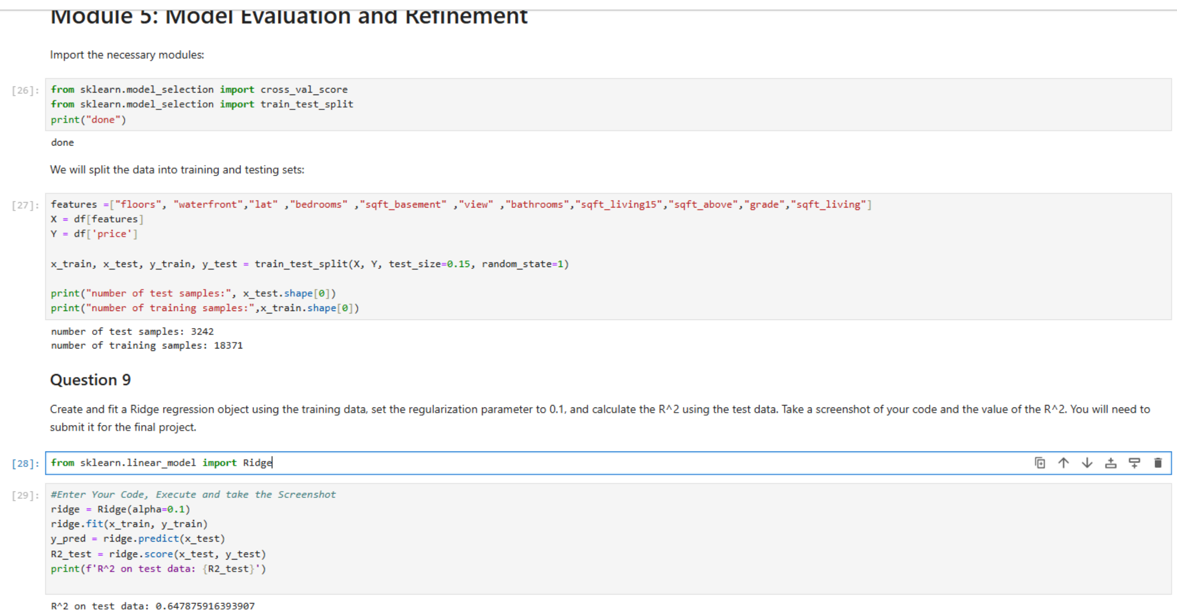
Question 8

Use the list to create a pipeline object to predict the 'price', fit the object using the features in the list `features`, and calculate the R^2. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
[61]: #Enter Your Code, Execute and take the Screenshot
      pipe=Pipeline(Input)
      Z = Z.astype(float)
      pipe.fit(Z,Y)
      ypipe=pipe.predict(Z)
      R2 = pipe.score(Z, Y)
      print(f'R^2: {R2}')

      R^2: 0.7512051345272872
```

QUESTION 9



Module 5: Model Evaluation and Refinement

Import the necessary modules:

```
[26]: from sklearn.model_selection import cross_val_score
      from sklearn.model_selection import train_test_split
      print("done")

      done
```

We will split the data into training and testing sets:

```
[27]: features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","sqft_living15","sqft_above","grade","sqft_living"]
      X = df[features]
      Y = df['price']

      x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.15, random_state=1)

      print("number of test samples:", x_test.shape[0])
      print("number of training samples:",x_train.shape[0])

      number of test samples: 3242
      number of training samples: 18371
```

Question 9

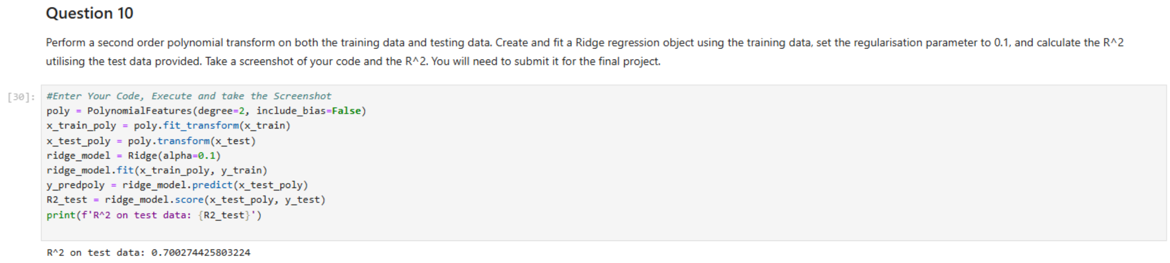
Create and fit a Ridge regression object using the training data, set the regularization parameter to 0.1, and calculate the R^2 using the test data. Take a screenshot of your code and the value of the R^2. You will need to submit it for the final project.

```
[28]: from sklearn.linear_model import Ridge
```

```
[29]: #Enter Your Code, Execute and take the Screenshot
      ridge = Ridge(alpha=0.1)
      ridge.fit(x_train, y_train)
      y_pred = ridge.predict(x_test)
      R2_test = ridge.score(x_test, y_test)
      print(f'R^2 on test data: {R2_test}')

      R^2 on test data: 0.647875916393907
```

QUESTION 10



Question 10

Perform a second order polynomial transform on both the training data and testing data. Create and fit a Ridge regression object using the training data, set the regularisation parameter to 0.1, and calculate the R^2 utilising the test data provided. Take a screenshot of your code and the R^2. You will need to submit it for the final project.

[30]:

```
#Enter Your Code, Execute and take the Screenshot
poly = PolynomialFeatures(degree=2, include_bias=False)
x_train_poly = poly.fit_transform(x_train)
x_test_poly = poly.transform(x_test)
ridge_model = Ridge(alpha=0.1)
ridge_model.fit(x_train_poly, y_train)
y_predpoly = ridge_model.predict(x_test_poly)
R2_test = ridge_model.score(x_test_poly, y_test)
print(f'R^2 on test data: {R2_test}')
```

```
R^2 on test data: 0.700274425803224
```