Name: Rahul Vijayvargiya,
Student ID: 245784

Task:
Real estate valuation data set Link for dataset:

UCI Machine Learning Repository: Real estate valuation data set Data Set

The dataset contains historical data, from the Taiwanese property market.
The dataset allows learning a model that will estimate housing prices based on the data describing them. The collection is designed for the regression task.

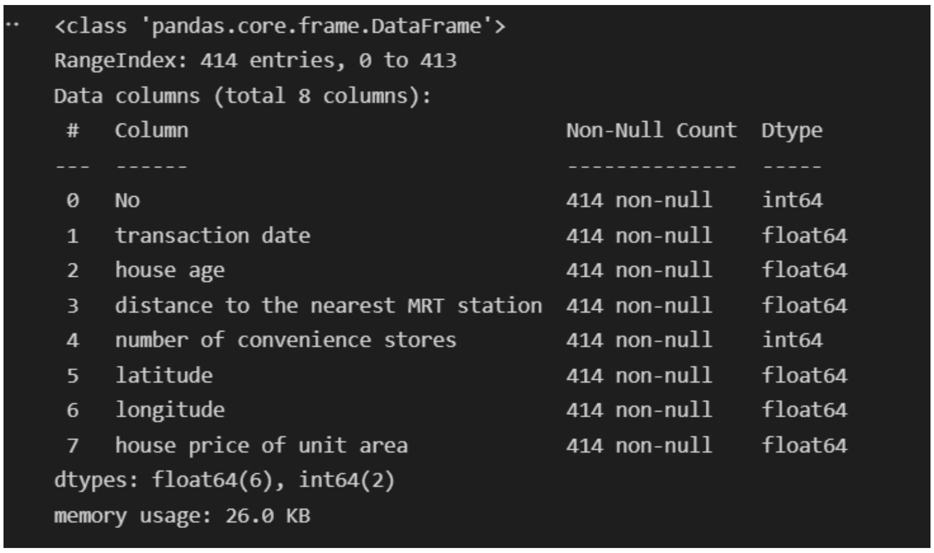
Sol:

Columns names:

['No',
'transaction date', 'house age',
'distance to the nearest MRT station',
'number of convenience stores',
'latitude', 'longitude',
'house price of unit area'] are the columns in our dataset,

Which tells us about index no.
Transaction date, house age,
Distance from the metro-station, no. of convenience stores around, coordinates and in that area
What are the price as per unit area

1.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                               Non-Null Count  Dtype
---  ------                               --------------  -----
 0   No                                   414 non-null    int64
 1   transaction date                     414 non-null    float64
 2   house age                            414 non-null    float64
 3   distance to the nearest MRT station  414 non-null    float64
 4   number of convenience stores         414 non-null    int64
 5   latitude                             414 non-null    float64
 6   longitude                            414 non-null    float64
 7   house price of unit area             414 non-null    float64
dtypes: float64(6), int64(2)
memory usage: 26.0 KB
```

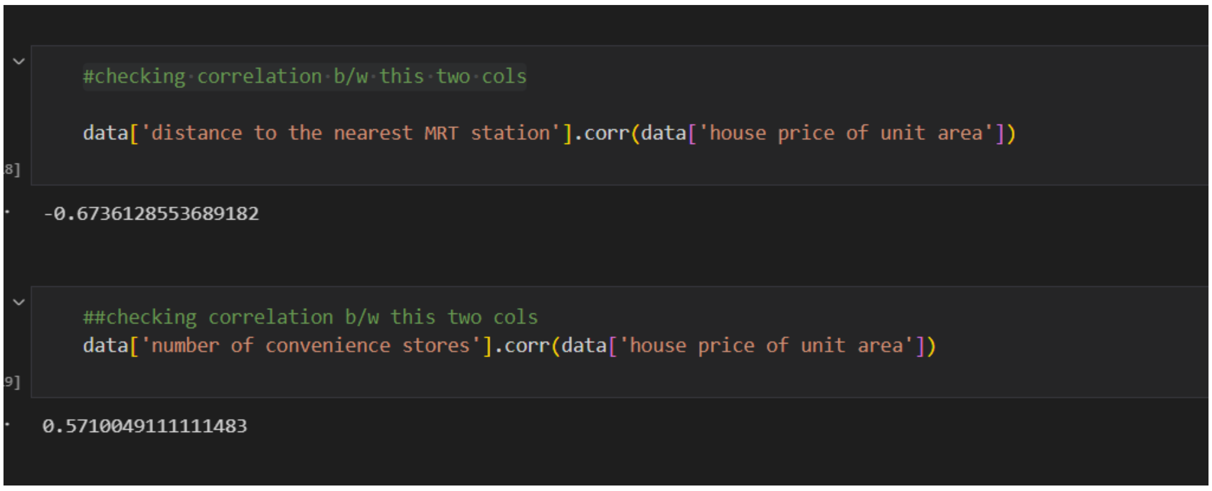
- Dataset info function, tells us about non-null count and datatype of col value start with a checking a correlation with columns of dataset,

Correlation is a statistical measure that expresses the extent to which two variables are linearly related or not

A correlation coefficient of +1 indicates a perfect positive correlation. As variable x increases, variable y increases. As variable x decreases, variable y decreases. A correlation coefficient of -1 indicates a perfect negative correlation.

I had two columns in my mind to take as a dependent variable

2.



```
#checking correlation b/w this two cols

data['distance to the nearest MRT station'].corr(data['house price of unit area'])
```

```
-0.6736128553689182
```

```
##checking correlation b/w this two cols
data['number of convenience stores'].corr(data['house price of unit area'])
```

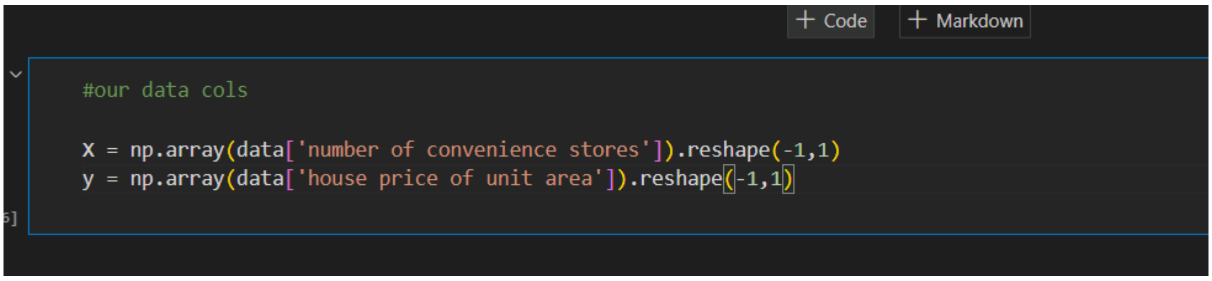
```
0.5710049111111483
```

As you can see here in above picture,

So i decided to take convenience stores as my dependent variable,

So we gonna predict price of unit area as per no. of convenience store in the area,

3.

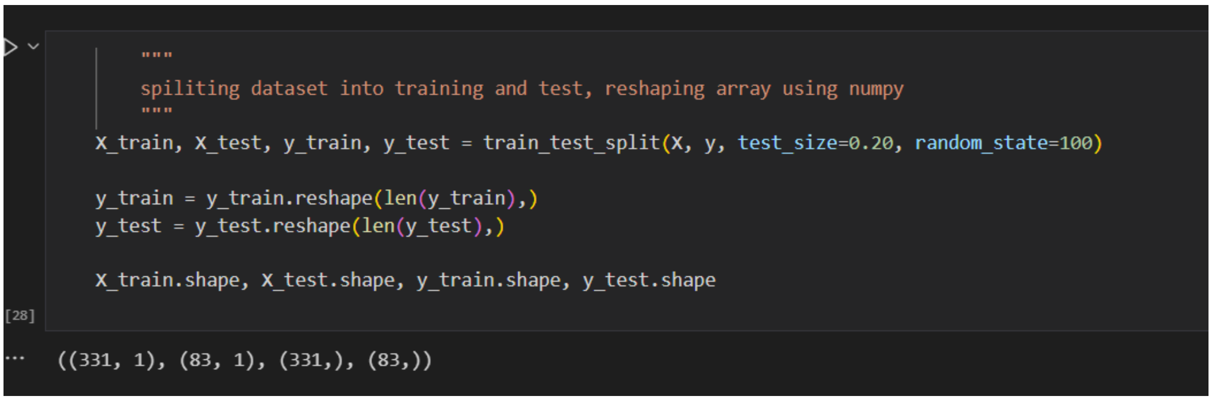


```
#our data cols

X = np.array(data['number of convenience stores']).reshape(-1,1)
y = np.array(data['house price of unit area']).reshape(-1,1)
```

As you can see above picture, we are reshaping an array and converting it into 1D array

4.



```
"""
spiliting dataset into training and test, reshaping array using numpy
"""
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=100)

y_train = y_train.reshape(len(y_train),)
y_test = y_test.reshape(len(y_test),)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((331, 1), (83, 1), (331,), (83,))
```

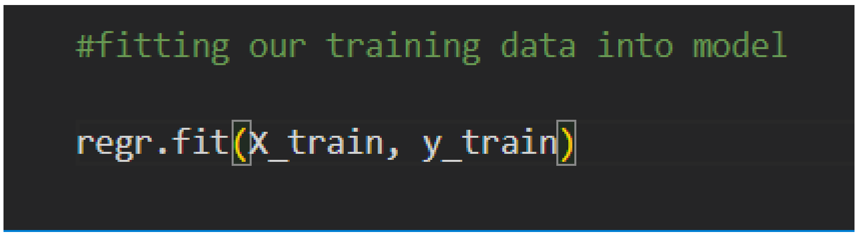
Finally here, we are converting our 1D Array into train and test dataset, with 80% Data into Training and 20% data for the test

Linear Regression:

a linear approach for modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables).

We are using sklearn from python, which has regression model inside,
We are calling and fitting our training data into it

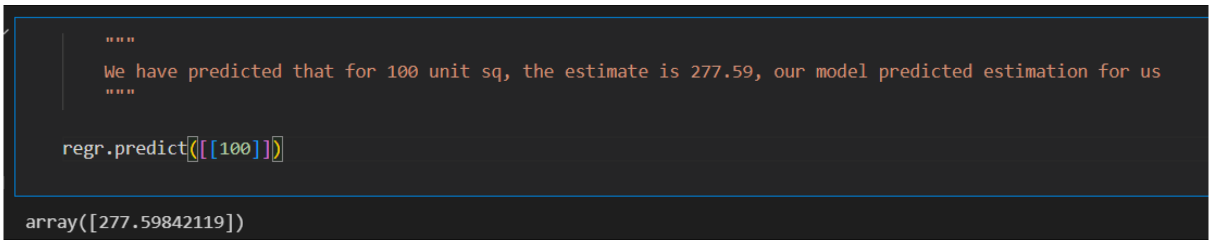
5.



```
#fitting our training data into model

regr.fit(X_train, y_train)
```

6.



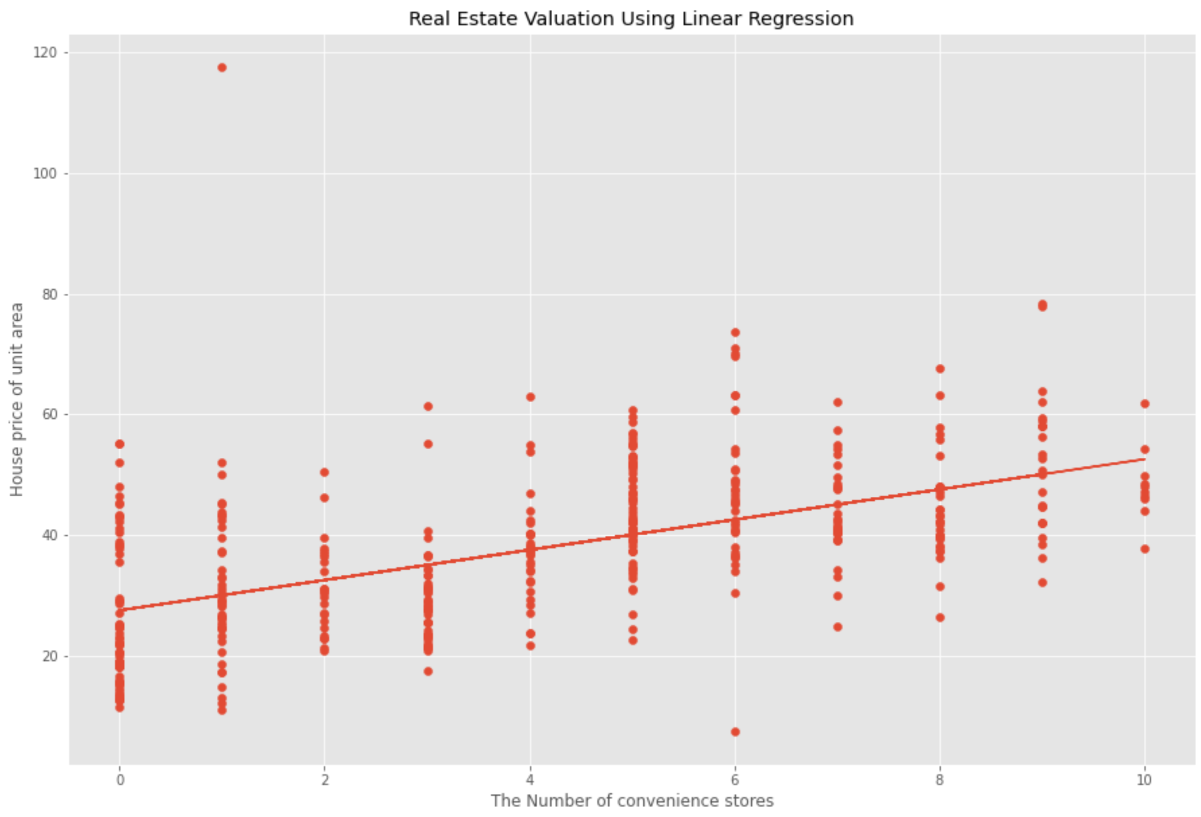
```
"""
We have predicted that for 100 unit sq, the estimate is 277.59, our model predicted estimation for us
"""

regr.predict([[100]])
```

```
array([277.59842119])
```

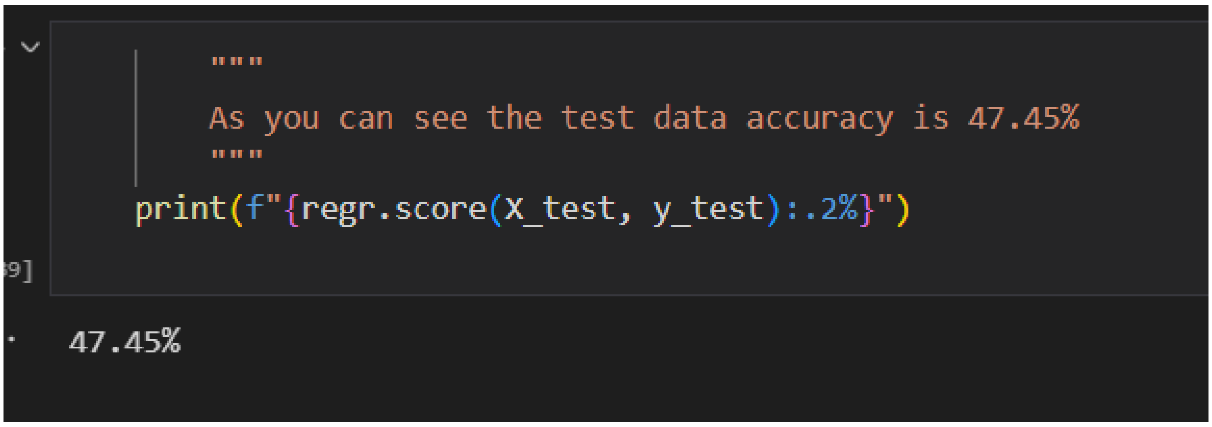
So far, using simple linear regression we have predicted for 100 unit sq. price would be around 277.59

7.



As you can see a regression line has been plotted using simple linear regression.

8.



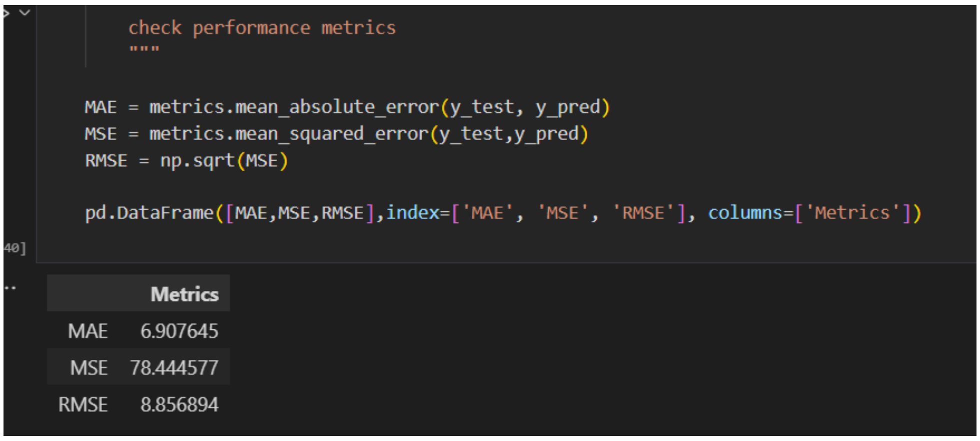
```
"""
As you can see the test data accuracy is 47.45%
"""
print(f"{regr.score(X_test, y_test):.2%}")
```

```
47.45%
```

We have model accuracy on test data is 47.45%

Furthermore checking,

9.



```
    check performance metrics
    """

MAE = metrics.mean_absolute_error(y_test, y_pred)
MSE = metrics.mean_squared_error(y_test,y_pred)
RMSE = np.sqrt(MSE)

pd.DataFrame([MAE,MSE,RMSE],index=['MAE', 'MSE', 'RMSE'], columns=['Metrics'])
```

```
       Metrics
MAE   6.907645
MSE  78.444577
RMSE  8.856894
```

As you can see in pic above, mean absolute error, mean square error, and root mean square error

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

The Mean squared error (MSE) tells you how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the “errors”) and squaring them. The squaring is necessary to remove any negative signs.

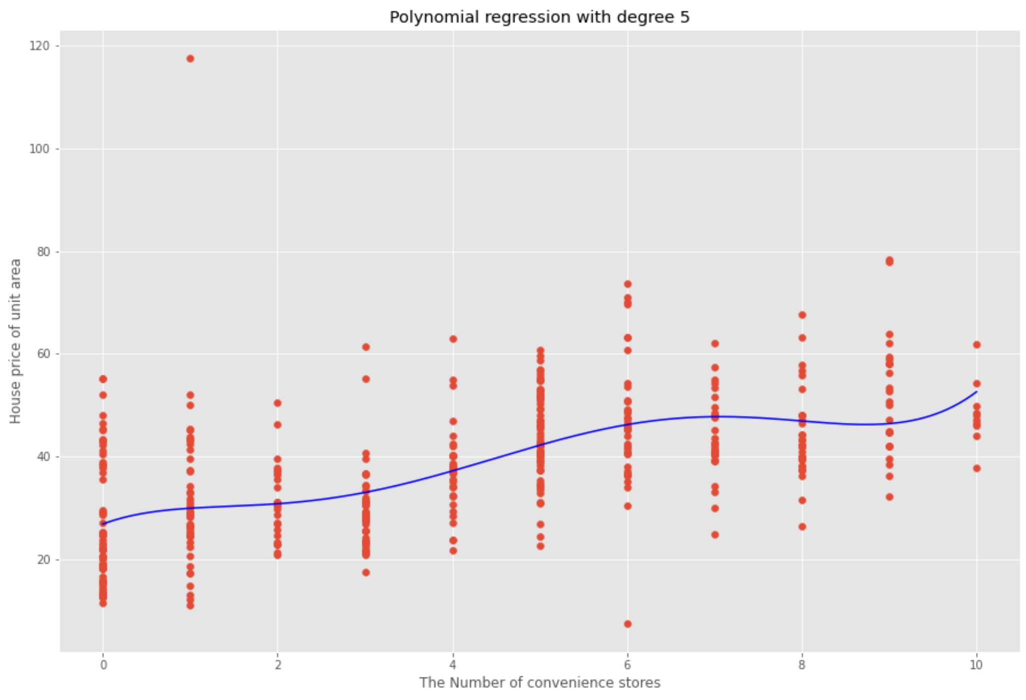
The Mean of the absolute error(MAE) tells us about the absolute value of the difference between the forecasted value and the actual value. MAE tells us how big of an error we can expect from the forecast on average.

Polynomial Regression and Plot:

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an nth degree polynomial in x.

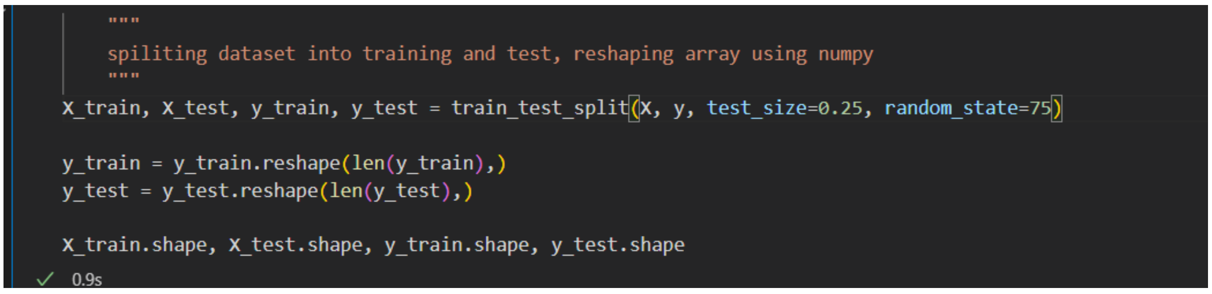
Here were using 5 degree in our model Fit a polynomial of degree 4 to the 5 points. In general, for n points, you can fit a polynomial of degree n-1 to exactly pass through the points. p = polyfit(x,y,4); Evaluate the original function and the polynomial fit on a finer grid of points between 0 and 2.

10. Here we are,



Experiments:

Changing the test size and random state of our training and test data



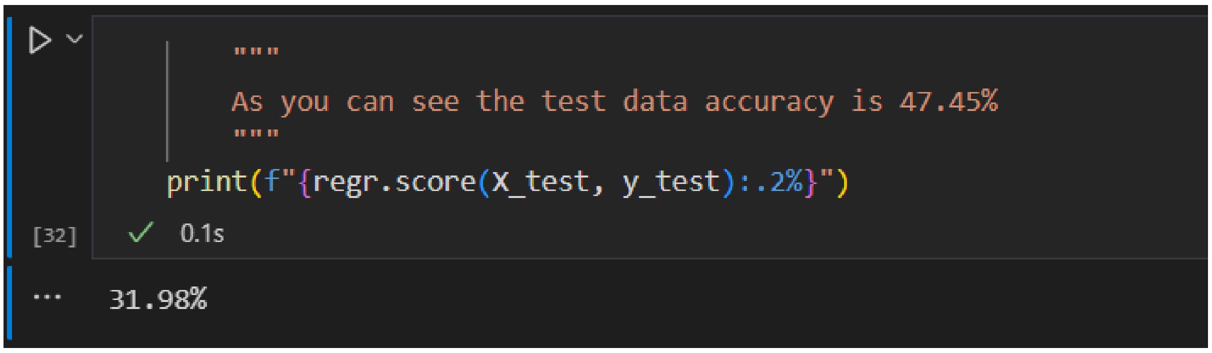
```
"""
spiliting dataset into training and test, reshaping array using numpy
"""
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=75)

y_train = y_train.reshape(len(y_train),)
y_test = y_test.reshape(len(y_test),)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

✓ 0.9s

It seems the performance of our model has decrease

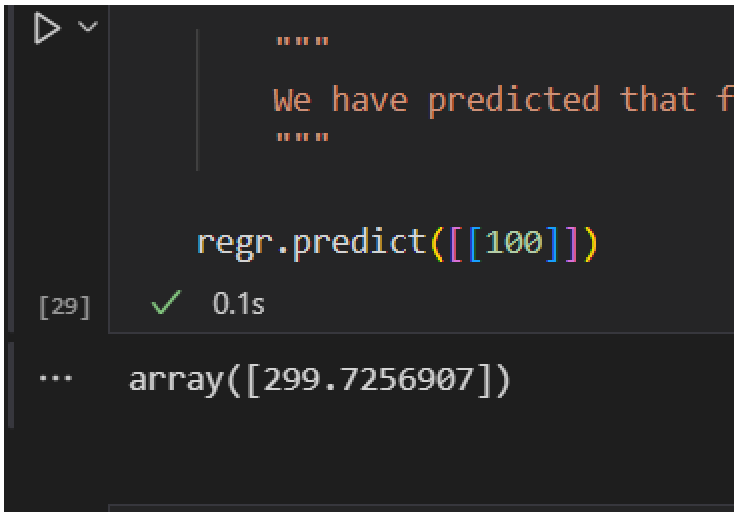


```
"""
As you can see the test data accuracy is 47.45%
"""
print(f"{regr.score(X_test, y_test):.2%}")
```

[32] ✓ 0.1s

```
31.98%
```

From previously price it’s predicted for 100 unit sq is increase



```
"""
We have predicted that f
"""

regr.predict([[100]])
```

```
[29] ✓ 0.1s
```

```
array([299.7256907])
```