

House Price Prediction

Dataset - 3

Description:

Accurately predicting house prices can be a daunting task. The buyers are just not concerned about the size(square feet) of the house and there are various other factors that play a key role to decide the price of a house/property. It can be extremely difficult to figure out the right set of attributes that are contributing to understanding the buyer's behavior as such. This dataset has been collected across various property aggregators across.

Date set:

We divided into 3 datasets each dataset containing 7 columns and 1000 rows with csv extension. The data contains the following columns :

'Avg. Area Income' – Avg. The income of the householder of the city house is located. 'Avg. Area House Age' – Avg. Age of Houses in the same city. 'Avg. Area Number of Rooms' – Avg. Number of Rooms for Houses in the same city. 'Avg. Area Number of Bedrooms' – Avg. Number of Bedrooms for Houses in the same city. 'Area Population' – Population of the city. 'Price' – Price that the house sold at. 'Address' – Address of the houses.

visualization:

Pair plot: Pair plot is used to understand the best set of features to explain a relationship between two variables or to form the most separated clusters. Dist plot: Distplot is a combination of a histogram with a line on it. Heat map: A heat map represents these coefficients to visualize the strength of correlation among variables. Scatter plot: Scatter plots shows how much one variable is affected by another or the relationship between them with the help of dots in two dimensions.

Python packages:

Numpy: is a library used to add support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Pandas: it is used to work with datasets, for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Seaborn: is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.

Matplotlib: is a comprehensive library for creating static, animated, and interactive visualizations in Python

Import Libraries

```
In [1]: import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
%matplotlib inline
```

Importing Data and Checking out.

```
In [2]: HouseDF = pd.read_csv('USA_Housing_3.csv')
```

```
In [3]: HouseDF.head()
```

Out[3]:	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	55813.29647	8.318992	7.726407	6.04	22779.36995	1.164497e+06	65858 Simmons Coves Apt. 430\nEast Garyshire, ...
1	75947.47173	5.329159	7.093562	6.23	37455.31756	1.375737e+06	2198 Green Pike\nSouth Nicole, KY 29768
2	58615.14560	4.791109	5.712521	3.43	40187.32733	6.365060e+05	5802 Hudson Crest\nElliottstad, UT 03084
3	56868.10768	6.474246	6.589028	3.32	31060.20473	7.759577e+05	7086 Sheila Curve\nNorth Amandaburgh, DE 08791
4	70586.59589	4.775051	7.310313	4.22	31798.68515	9.765401e+05	287 Fields Falls Suite 212\nWellsside, WY 48741

In [4]: `HouseDF.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Avg. Area Income    1000 non-null   float64
 1   Avg. Area House Age 1000 non-null   float64
 2   Avg. Area Number of Rooms 1000 non-null   float64
 3   Avg. Area Number of Bedrooms 1000 non-null   float64
 4   Area Population     1000 non-null   float64
 5   Price               1000 non-null   float64
 6   Address             1000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 54.8+ KB
```

In [5]: `HouseDF.describe()`

Out[5]:

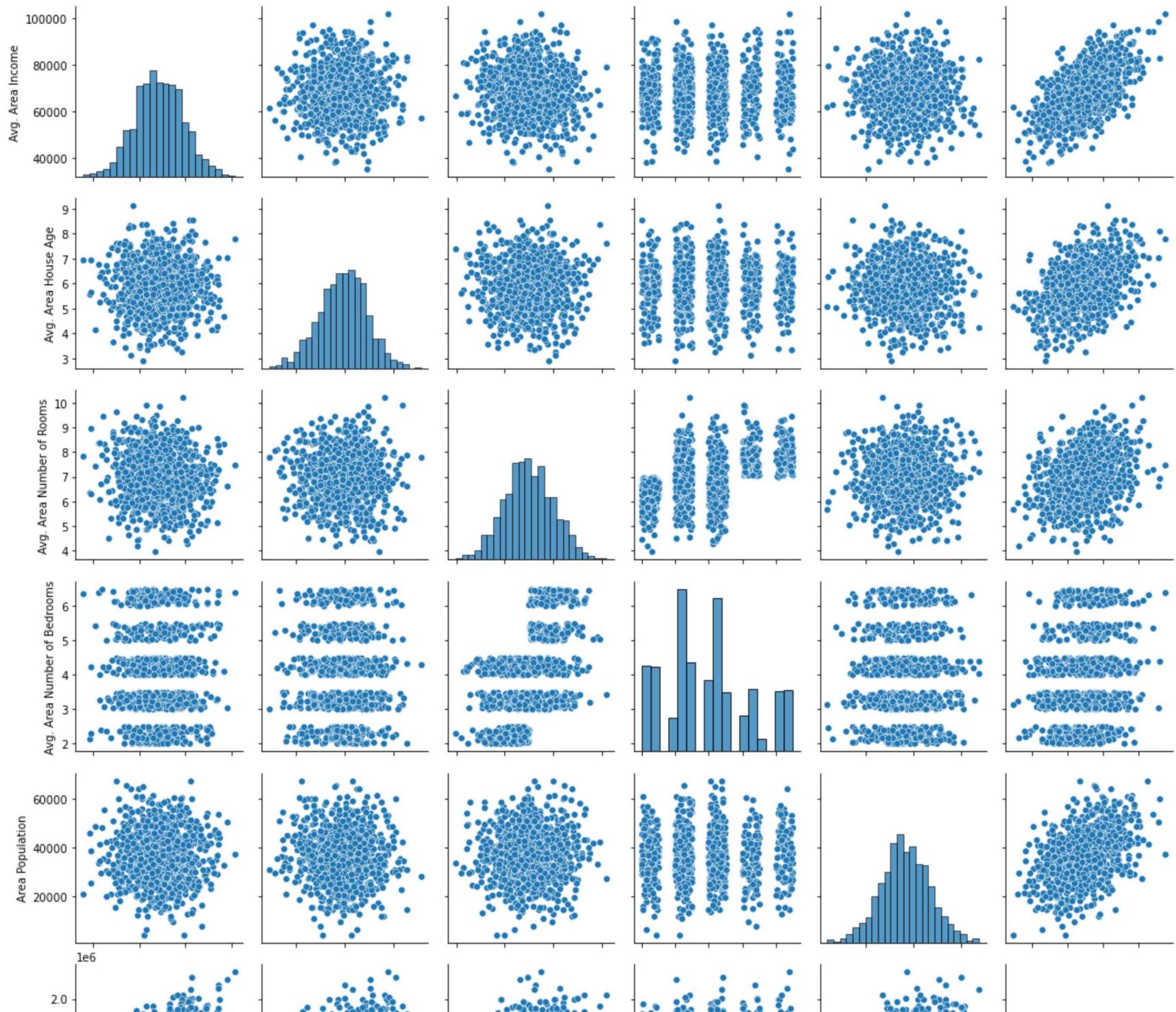
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03
mean	68585.065504	5.930137	6.981274	3.965970	36654.079178	1.233484e+06
std	10420.981531	0.970635	1.018742	1.249522	10301.603069	3.412742e+05
min	35608.986240	2.922736	3.950225	2.000000	3883.448164	2.311898e+05
25%	61366.098858	5.304363	6.308540	3.120000	30108.396160	1.008599e+06
50%	68384.321295	5.956386	6.970312	4.040000	36329.393050	1.240684e+06
75%	75866.944905	6.583477	7.662061	4.490000	43651.945155	1.454699e+06
max	101599.670600	9.125283	10.219902	6.500000	67701.649800	2.370231e+06

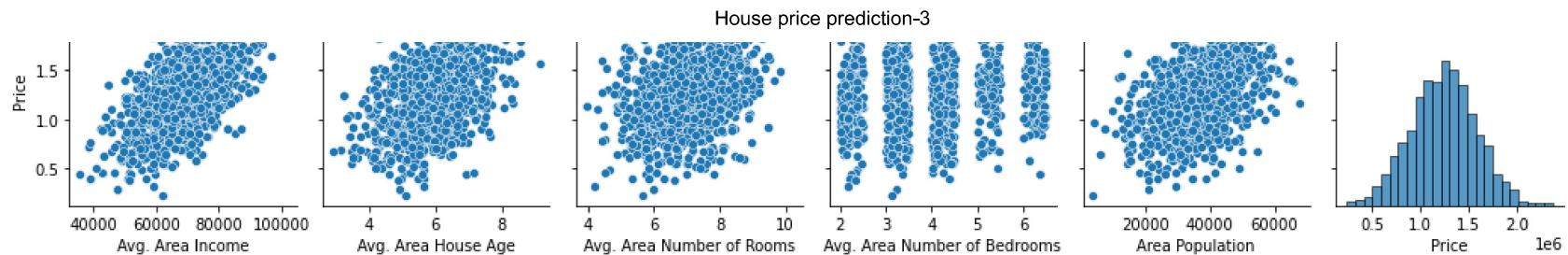
In [6]: `HouseDF.columns`Out[6]: `Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'], dtype='object')`

Exploratory Data Analysis for House Price Prediction

In [7]: `sns.pairplot(HouseDF)`Out[7]: `<seaborn.axisgrid.PairGrid at 0x174fa1e0970>`

House price prediction-3



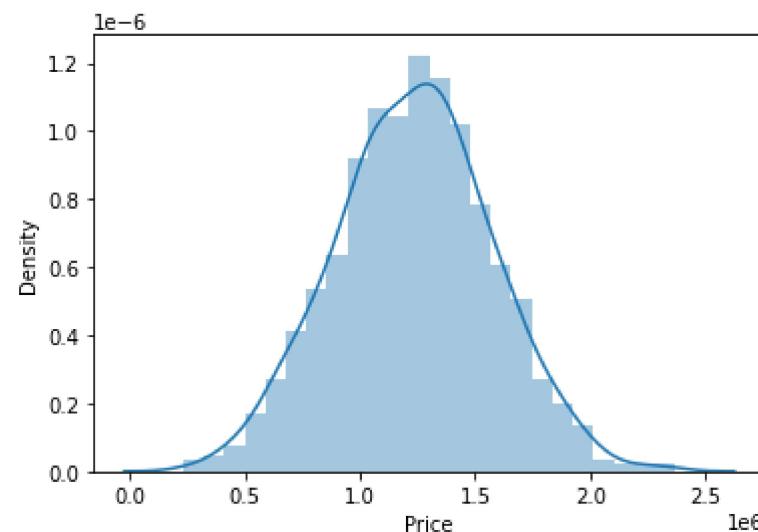


```
In [8]: sns.distplot(HouseDF['Price'])
```

C:\Users\DELL\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

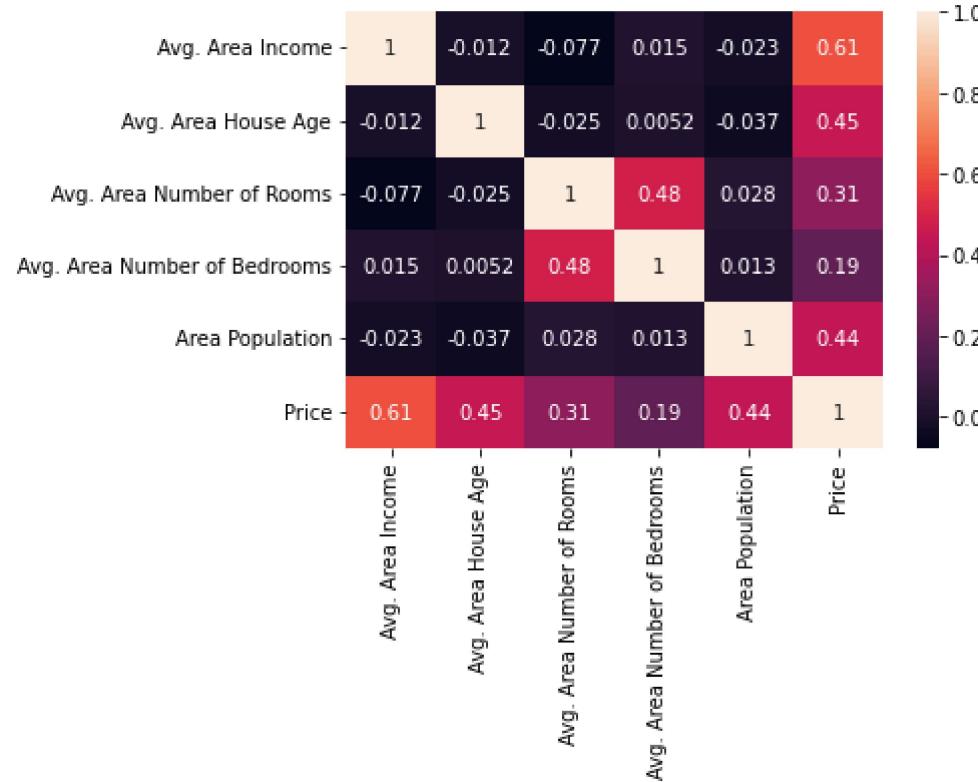
```
    warnings.warn(msg, FutureWarning)
```

```
Out[8]: <AxesSubplot:xlabel='Price', ylabel='Density'>
```



```
In [9]: sns.heatmap(HouseDF.corr(), annot=True)
```

```
Out[9]: <AxesSubplot:>
```



Training a Linear Regression Model

X and y List

```
In [10]: X = HouseDF[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]

y = HouseDF['Price']
```

Split Data into Train, Test

```
In [11]: from sklearn.model_selection import train_test_split
```

```
In [12]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

Creating and Training the LinearRegression Model

```
In [13]: from sklearn.linear_model import LinearRegression
```

```
In [14]: lm = LinearRegression()
```

```
In [15]: lm.fit(X_train,y_train)
```

```
Out[15]: LinearRegression()
```

LinearRegression Model Evaluation

```
In [16]: print(lm.intercept_)
```

```
-2660687.7107224436
```

```
In [17]: coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=[ 'Coefficient'])  
coeff_df
```

	Coefficient
Avg. Area Income	22.146793
Avg. Area House Age	167364.223838
Avg. Area Number of Rooms	116662.640912
Avg. Area Number of Bedrooms	1569.504875
Area Population	15.363966

Predictions from our Linear Regression Model

```
In [18]: predictions = lm.predict(X_test)
```

```
In [19]: #print(predictions)
```

```
from sklearn.metrics import accuracy_score
```

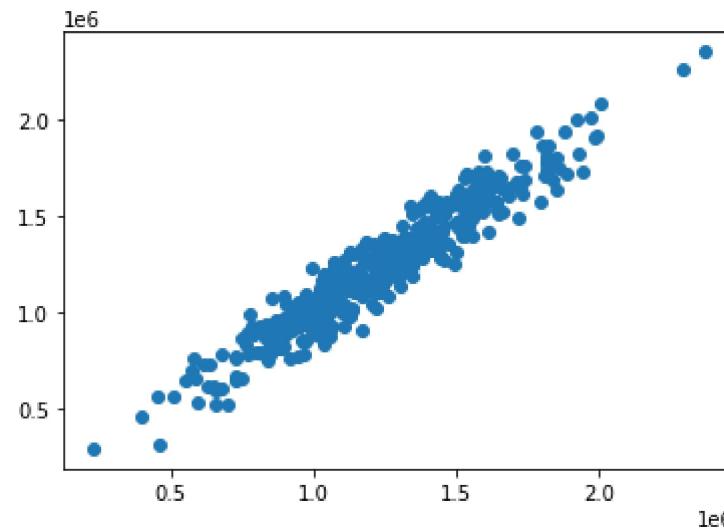
```
#print(accuracy_score(y_test,predictions))
```

```
In [20]: lm.score(X_test, y_test)
```

```
Out[20]: 0.910916896804859
```

```
In [21]: plt.scatter(y_test,predictions)
```

```
Out[21]: <matplotlib.collections.PathCollection at 0x174fdc22bb0>
```

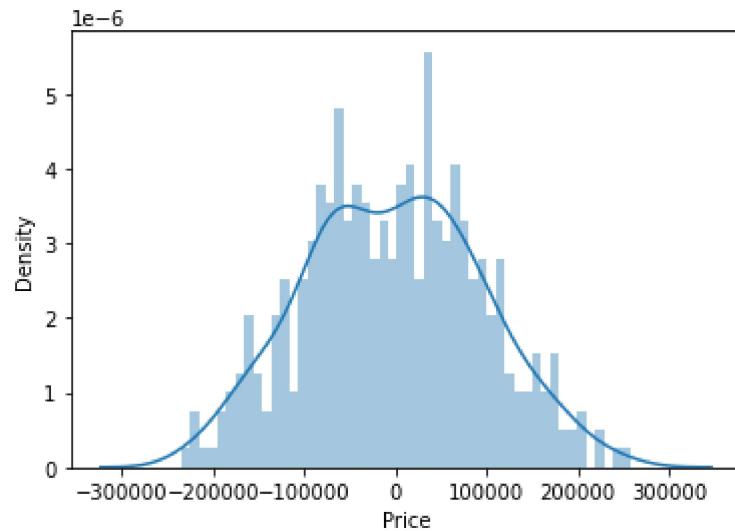


In the above scatter plot, we see data is in line shape, which means our model has done good predictions.

```
In [22]: sns.distplot((y_test-predictions),bins=50);
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```

```
warnings.warn(msg, FutureWarning)
```



In the above histogram plot, we see data is in bell shape (Normally Distributed), which means our model has done good predictions.

Regression Evaluation Metrics

```
In [23]: from sklearn import metrics
```

```
In [24]: print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 79692.43730505006
MSE: 9390250756.171417
RMSE: 96903.30621899037
```

SVR

```
In [25]: from sklearn.svm import SVR
svr = SVR(C=100000)
svr.fit(X_train, y_train)
predictions = svr.predict(X_test)
```

```
In [26]: svr.score(X_test, y_test)
```

```
Out[26]: 0.5254069935162614
```

KNN

```
In [27]: from sklearn.neighbors import KNeighborsRegressor  
  
knn_model=KNeighborsRegressor()  
knn_model.fit(X_train, y_train)  
knn_model.score(X_test,y_test)
```

```
Out[27]: 0.4281786385326768
```

Desicion tree

```
In [28]: from sklearn.tree import DecisionTreeRegressor  
  
dtr_model=DecisionTreeRegressor(max_depth=5)  
dtr_model.fit(X_train, y_train)  
dtr_model.score(X_test,y_test)
```

```
Out[28]: 0.5985371468954082
```

```
In [ ]:
```