Homework I: Predicting Home Prices

The [Ames Housing Dataset] was introduced by Professor Dean De Cock in 2011 for use in data science education. It contains 2,919 observations of housing sales in Ames, Iowa between 2006 and 2010. There are a total of 79 features describing each house's size, quality, area, age, and other miscellaneous attributes.

From Kaggle:

> Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

Please follow the instructions in this notebook and prepare the data for home price prediction. **Submit your solutions as a PDF file to Blackboard by Wednesday, March 2nd at 11:59PM.**

Author: Diego Freire

1. Overall Understanding of the Data

In this section, you will need to complete the following tasks:

- Load the dataset as a pandas data frame.
- Display key information of the data.
- Handle missing values.

1.1 In the cell below, import the `pandas` library and load file `train.csv` from the Ames housing dataset as a data frame.

```
import pandas as pd

df = pd.read_csv('train.csv')
```

1.2 Display the first 5 rows of the data frame.

```
df.head()
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl

5 rows × 81 columns

1.3 Display the shape of the data frame and list all column names.

```
print("Shape:", df.shape)

df.columns
```

```
Shape: (1460, 62)
Index(['Id', 'MSSubClass', 'MSZoning', 'LotArea', 'Street', 'LotShape',
       'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood',
       'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'OverallQual',
       'OverallCond', 'YearBuilt', 'YearRemodAdd', 'RoofStyle', 'RoofMatl',
       'Exterior1st', 'Exterior2nd', 'ExterQual', 'ExterCond', 'Foundation',
       'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF',
       'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath',
       'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
       'Functional', 'Fireplaces', 'GarageCars', 'GarageArea', 'PavedDrive',
       'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
       'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

1.4 Display the number of missing values in each column.

```
df.isna().sum()
```

```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     259
LotArea           0
               ...
MoSold            0
YrSold            0
SaleType          0
```

```
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

1.5 Remove all the columns that contain missing values.

```
df = df.dropna(axis='columns')
```

```
df
```

	Id	MSSubClass	MSZoning	LotArea	Street	LotShape	LandContour	Utilities	Lo
0	1	60	RL	8450	Pave	Reg	Lvl	AllPub	
1	2	20	RL	9600	Pave	Reg	Lvl	AllPub	
2	3	60	RL	11250	Pave	IR1	Lvl	AllPub	
3	4	70	RL	9550	Pave	IR1	Lvl	AllPub	
4	5	60	RL	14260	Pave	IR1	Lvl	AllPub	
...	...	...	...	...	...	...	...	...	
1455	1456	60	RL	7917	Pave	Reg	Lvl	AllPub	
1456	1457	20	RL	13175	Pave	Reg	Lvl	AllPub	
1457	1458	70	RL	9042	Pave	Reg	Lvl	AllPub	
1458	1459	20	RL	9717	Pave	Reg	Lvl	AllPub	
1459	1460	20	RL	9937	Pave	Reg	Lvl	AllPub	

1460 rows × 62 columns

2. Study Key Features

The total number of features seems overwhelming, so let's start with a few features that we know are definitely relevant:

1. `OverallQual` : Overall material and finish quality
2. `YearBuilt` : Original construction date
3. `TotalBsmtSF` : Total basement area in square feet
4. `GrLivArea` : Above ground living area in square feet

and don't forget `SalePrice` .

For each of these 5 features, please find:

- Descriptive statistics
- Graphical representation of their distribution
- Check for outliers
- Study correlations

2.1 **Descriptive statistics**: For each of the 5 features, find its minimum, maximum, mean, and standard deviation.

```
df[['OverallQual','YearBuilt','TotalBsmtSF','GrLivArea','SalePrice']].describe()
```

	OverallQual	YearBuilt	TotalBsmtSF	GrLivArea	SalePrice
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	6.099315	1971.267808	1057.429452	1515.463699	180921.195890
std	1.382997	30.202904	438.705324	525.480383	79442.502883
min	1.000000	1872.000000	0.000000	334.000000	34900.000000
25%	5.000000	1954.000000	795.750000	1129.500000	129975.000000
50%	6.000000	1973.000000	991.500000	1464.000000	163000.000000
75%	7.000000	2000.000000	1298.250000	1776.750000	214000.000000
max	10.000000	2010.000000	6110.000000	5642.000000	755000.000000

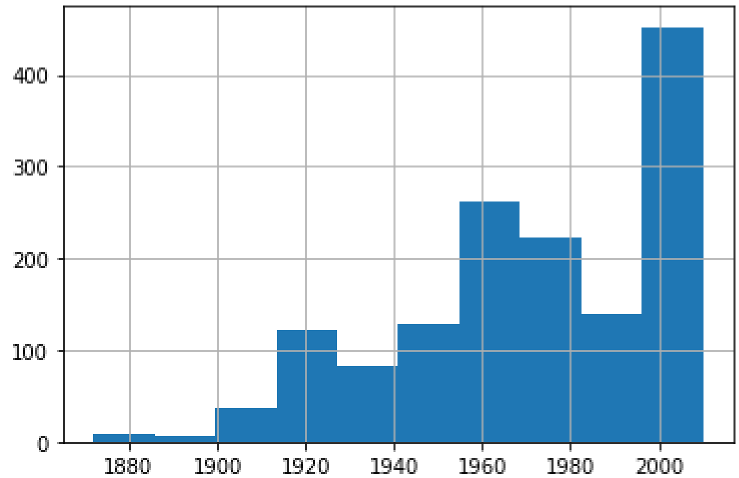
2.2 **Distribution**: For each of the 5 features, generate a histogram. Choose the number of bins properly.

```
df['OverallQual'].hist(bins = 20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f175ba91390>
```

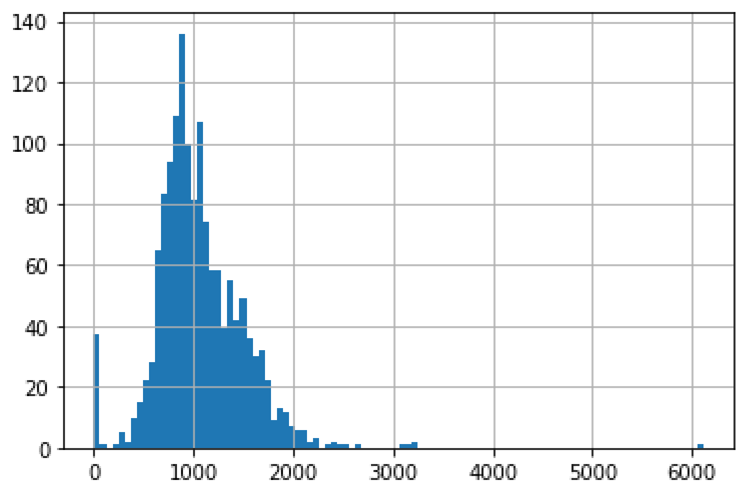
```
df['YearBuilt'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f175b917910>
```



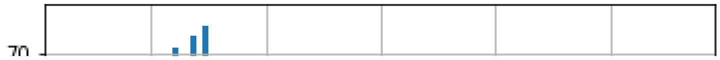

```
df['TotalBsmtSF'].hist(bins = 100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f175b467490>
```



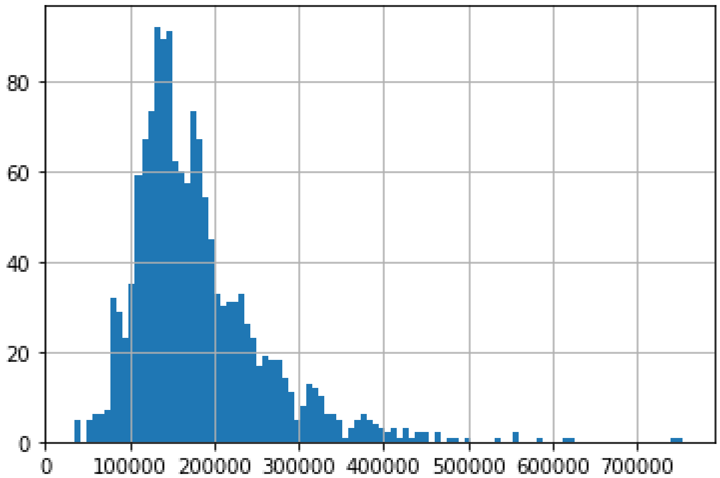

```
df['GrLivArea'].hist(bins = 100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f175b293910>
```



```
df['SalePrice'].hist(bins = 100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f175b130b10>
```



2.3 **Outliers**: An **outlier** is a value that is located far away from the vast majority of the data. Remove those rows that contain outliers.

```
# Need to find best way to do this
import numpy as np

Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

df_out = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

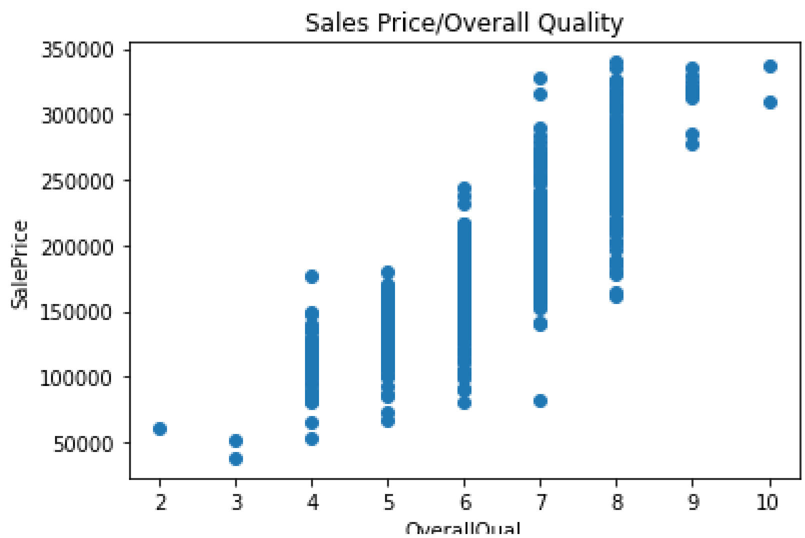
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: FutureWarning: Automatic
```

2.4 **Correlation with sale price**: For each of the 4 chosen predictive features, draw a scatter plot of this feature and `SalePrice`. Set the title, axis label of the graph properly.

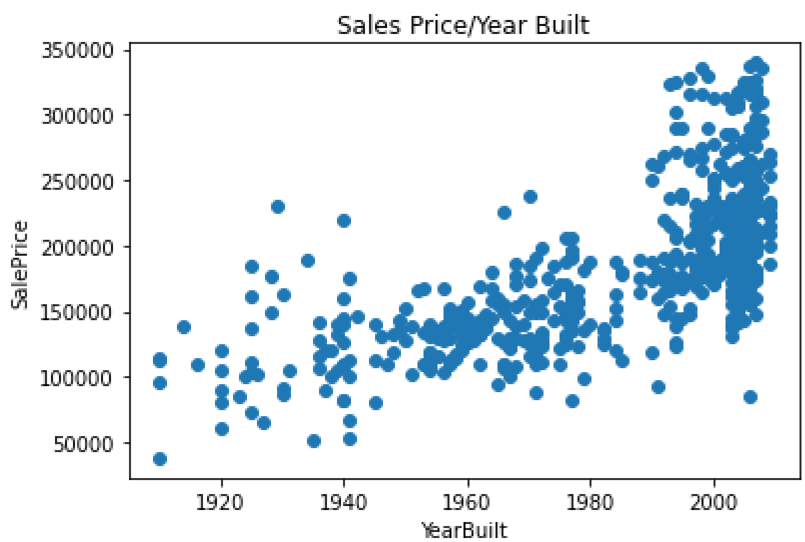
```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.scatter(df_out['OverallQual'],df_out['SalePrice'])
plt.title("Sales Price/Overall Quality")
plt.xlabel("OverallQual")
```

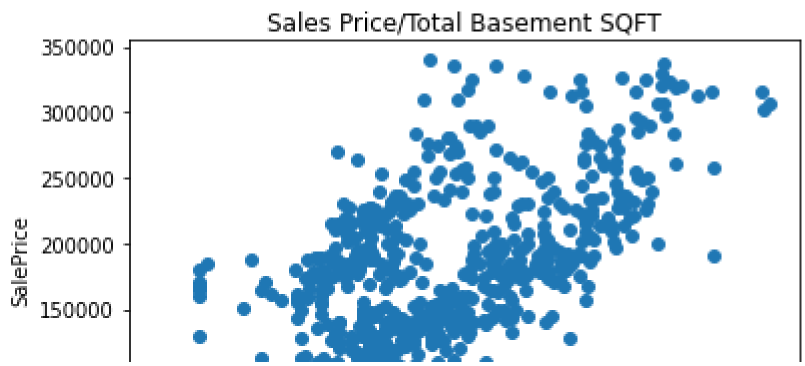
```
plt.ylabel("SalePrice")
plt.show()
```



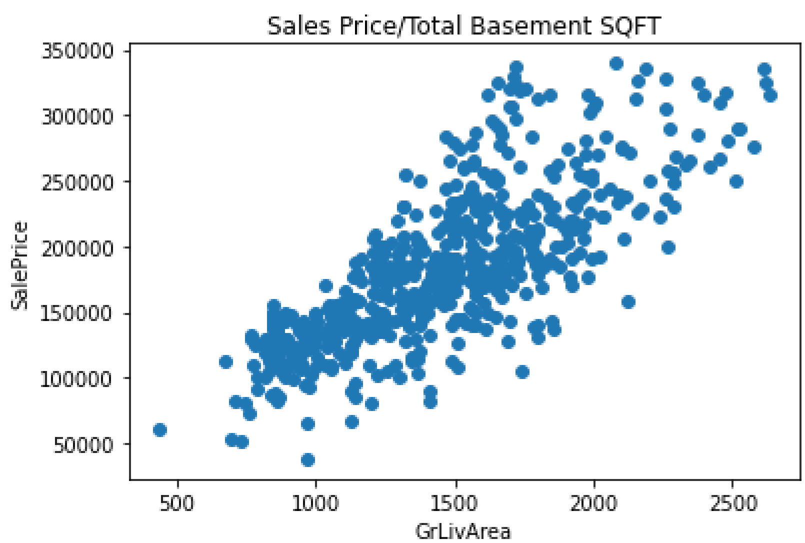

```
plt.scatter(df_out['YearBuilt'],df_out['SalePrice'])
plt.title("Sales Price/Year Built")
plt.xlabel("YearBuilt")
plt.ylabel("SalePrice")
plt.show()
```




```
plt.scatter(df_out['TotalBsmtSF'],df_out['SalePrice'])
plt.title("Sales Price/Total Basement SQFT")
plt.xlabel("TotalBsmtSF")
plt.ylabel("SalePrice")
plt.show()
```




```
plt.scatter(df_out['GrLivArea'],df_out['SalePrice'])
plt.title("Sales Price/Total Basement SQFT")
plt.xlabel("GrLivArea")
plt.ylabel("SalePrice")
plt.show()
```



Describe the correlation between each predictive feature and `SalePrice`. Is there a positive correclation, a negative correlation, or no correlation?

```
df_out[['OverallQual','YearBuilt','TotalBsmtSF','GrLivArea','SalePrice']].corr()
```

	OverallQual	YearBuilt	TotalBsmtSF	GrLivArea	SalePrice
OverallQual	1.000000	0.672065	0.491408	0.662874	0.841090
YearBuilt	0.672065	1.000000	0.413285	0.407915	0.677442
TotalBsmtSF	0.491408	0.413285	1.000000	0.268436	0.576487
GrLivArea	0.662874	0.407915	0.268436	1.000000	0.772132
SalePrice	0.841090	0.677442	0.576487	0.772132	1.000000

3. Identify Additional Predictive Feature

Let’s find out if other features are helpful to the price prediction. Additional features can be identified in the following ways:

- Calculate correlation coefficient between `SalePrice` and an existing feature.
- Create new features from existing features.

3.1 Calculate the correlation coefficient of each feature with `SalePrice` (excluding `SalePrice` itself). Identify the feature (other than the 4 features studied in the previous section) that has the strongest correlation with the sale prices.

```
#finding correlation for features not including 'OverallQual','YearBuilt','TotalBsmtSF','GrLi

df_out.corr()['SalePrice'].sort_values(ascending=False).drop(['OverallQual','YearBuilt','Tota
```

```
SalePrice        1.000000
GarageCars       0.719309
GarageArea       0.703012
FullBath         0.671137
YearRemodAdd     0.595790
TotRmsAbvGrd     0.583123
1stFlrSF         0.533746
OpenPorchSF      0.407853
Fireplaces       0.389852
2ndFlrSF         0.366058
WoodDeckSF       0.345703
LotArea          0.333735
HalfBath         0.293952
BsmtFinSF1       0.243154
BedroomAbvGr     0.224650
BsmtFullBath     0.221886
BsmtUnfSF        0.178867
MSSubClass       0.151670
MoSold           0.075577
YrSold           0.017785
Id              -0.024665
OverallCond     -0.352801
BsmtFinSF2            NaN
LowQualFinSF          NaN
BsmtHalfBath          NaN
KitchenAbvGr          NaN
EnclosedPorch         NaN
3SsnPorch             NaN
ScreenPorch           NaN
PoolArea              NaN
MiscVal               NaN
Name: SalePrice, dtype: float64
```

3.2 **Feature engineering**: Based on our experience, the total area of the house and the average area per room should also be important factors in determining the price. Please create these two columns using the following formula:

1. total area = total area above ground (“GrLivArea”) + total basement area (“TotalBsmtSF”)
2. area per room = total area above ground (“GrLivArea”) / number of rooms (“TotRmsAbvGrd”)

```
df_out['TotalArea'] = df_out['GrLivArea'] + df_out['TotalBsmtSF']
df_out['areaPerRoom'] = df_out['GrLivArea'] / df_out['TotRmsAbvGrd']
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
```

Up to this point, you should have obtained 7 features that are helpful to predict the sale price: `OverallQual`, `YearBuilt`, `TotalBasmtSF`, `GrLivArea`, `TotalArea`, `AreaPerRoom`, and a feature selected in 3.1. Create a new data frame with `SalePrice` and these 7 features only. Save the data as a CSV file named `HousingData_processed.csv` on your computer.

```
data = df_out[['OverallQual','YearBuilt','TotalBsmtSF','GrLivArea','TotalArea','areaPerRoom',
```

```
data.head()
```

	OverallQual	YearBuilt	TotalBsmtSF	GrLivArea	TotalArea	areaPerRoom	GarageCars
0	7	2003	856	1710	2566	213.750000	2
2	7	2001	920	1786	2706	297.666667	2
4	8	2000	1145	2198	3343	244.222222	3
6	8	2004	1686	1694	3380	242.000000	2
10	5	1965	1040	1040	2080	208.000000	1

4. Calculate Feature Statistics

Let's apply the **k-nearest-neighbor method** to this dataset and estimate the price of a house in the test set:

- OverallQual: 5
- YearBuilt: 1961
- TotalBsmtSF: 882
- GrLivArea: 896

Additional information about this house is on the first row of `test.csv`. The ID of this house in the data set is 1461.

The core idea of the k-nearest-neighbor method is to find existing houses that are most similar to the house with unknown price. Since similar houses should be priced similarly, their average price can be used as a good estimate on the price of the new house.

In order to conduct this estimation, we need to normalize the columns using the mean value and the standard deviation of each of the seven predictive features. These features include `OverallQual`, `YearBuilt`, `TotalBsmtSF`, `GrLivArea`, `TotalArea`, `AreaPerRoom`, and the feature you selected using correlation coefficient.

Transform each column with the following formula:

$$normalized\ value = \frac{original\ value - mean}{}$$

```
data = (data - data.mean())/data.std()
```

5. Measure Difference

For each house in the data frame, measure its difference to the target house by summing up the squared difference on each predictive feature. Write this value in a new column named `Diff`.

Display the difference for the first 5 houses below:

```
df1 = pd.read_csv('test.csv')

df1['TotalArea'] = df1['GrLivArea'] + df1['TotalBsmtSF']
df1['areaPerRoom'] = df1['GrLivArea'] / df1['TotRmsAbvGrd']

test_data = df1[['OverallQual','YearBuilt','TotalBsmtSF','GrLivArea','TotalArea','areaPerRoom


test_data
```

	OverallQual	YearBuilt	TotalBsmtSF	GrLivArea	TotalArea	areaPerRoom	GarageCar
0	5	1961	882.0	896	1778.0	179.200000	1.
1	6	1958	1329.0	1329	2658.0	221.500000	1.
2	5	1997	928.0	1629	2557.0	271.500000	2.
3	6	1998	926.0	1604	2530.0	229.142857	2.
4	8	1992	1280.0	1280	2560.0	256.000000	2.
...	...	...	...	...	...	...	.
1454	4	1970	546.0	1092	1638.0	218.400000	0.
1455	4	1970	546.0	1092	1638.0	182.000000	1.
1456	5	1960	1224.0	1224	2448.0	174.857143	2.

```
"""
OverallQual: 5
YearBuilt: 1961
TotalBsmtSF: 882
GrLivArea: 896
"""

# some rows in test_data had null values

#test_data.isnull().sum()

# I replaced the null values with the mean of each column

test_data.fillna(df.mean())
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: FutureWarning: Dropping
```

	OverallQual	YearBuilt	TotalBsmtSF	GrLivArea	TotalArea	areaPerRoom	GarageCar
0	5	1961	882.0	896	1778.0	179.200000	1.
1	6	1958	1329.0	1329	2658.0	221.500000	1.
1455	4	1970	546.0	1092	1638.0	182.000000	1.

```
np.array(test_data)[0]
A = np.array(test_data)[0]

B = np.array(data)

Diff = pow(abs(B - A),2).sum(axis=1)


data['Diff'] = Diff
```

```
data['Diff'].head(5)
```

```
0     8.616617e+06
2     8.614570e+06
4     8.608178e+06
6     8.606483e+06
10    8.627614e+06
Name: Diff, dtype: float64
```

6. Find Nearest Neighbors

Find 5 houses that are the most similar to the target house.

List their prices below.

```
k =5

sorted_data = data.sort_values(by='Diff', ascending=True)

target_index = []

# make a list of the k neighbors' targets
for i in range(k): # k
  index = sorted_data.index[i]
  target_index.append(index)
```

```
target_index
```

```
[261, 423, 305, 1359, 1105]
```

7. Make Predictions

The prediction on the price of the new house is the average price of the 5 houses listed above. Display the predicted price below.

```
df.loc[target_index][['OverallQual','YearBuilt','TotalBsmtSF','GrLivArea','SalePrice']]
```

	OverallQual	YearBuilt	TotalBsmtSF	GrLivArea	SalePrice
261	8	2007	1482	2574	276000
423	8	1998	1470	2630	315000
305	8	2004	2000	2000	305900
1359	9	2004	1980	1980	315000
1105	8	1994	1463	2622	325000