Importing the Libraries

```
import numpy as np
import pandas as pd
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

```
# Loading the data using excel format
data = pd.read_excel("/content/DS - Assignment Part 1 data set.xlsx")
```

Unmistakable Insights of the Dataset

```
data.head(10)
```

	Transaction date	House Age	Distance from nearest Metro station (km)	Number of convenience stores	latitude	longitude	Number of bedrooms	House size (sqft)	House price of unit area
0	2012.916667	32.0	84.87882	10	24.98298	121.54024	1	575	37.9
1	2012.916667	19.5	306.59470	9	24.98034	121.53951	2	1240	42.2

```
data.info
```

```
<bound method DataFrame.info of      Transaction date  House Age  Distance from nearest Metro station (km)  \
0         2012.916667       32.0                                   84.87882
1         2012.916667       19.5                                  306.59470
2         2013.583333       13.3                                  561.98450
3         2013.500000       13.3                                  561.98450
4         2012.833333        5.0                                  390.56840
..                ...        ...                                        ...
409       2013.000000       13.7                                 4082.01500
410       2012.666667        5.6                                   90.45606
411       2013.250000       18.8                                  390.96960
412       2013.000000        8.1                                  104.81010
413       2013.500000        6.5                                   90.45606

     Number of convenience stores  latitude  longitude  Number of bedrooms  \
0                              10  24.98298  121.54024                   1
1                               9  24.98034  121.53951                   2
2                               5  24.98746  121.54391                   3
3                               5  24.98746  121.54391                   2
4                               5  24.97937  121.54245                   1
..                            ...       ...        ...                 ...
409                             0  24.94155  121.50381                   3
410                             9  24.97433  121.54310                   2
411                             7  24.97923  121.53986                   1
412                             5  24.96674  121.54067                   1
413                             9  24.97433  121.54310                   2

     House size (sqft)  House price of unit area
0                  575                      37.9
1                 1240                      42.2
2                 1060                      47.3
3                  875                      54.8
```

```
4                491                43.1
..               ...                 ...
409              803                15.4
410             1278                50.0
411              503                40.6
412              597                52.5
413             1097                63.9

[414 rows x 9 columns]>
```

```
data.isnull()
```

	Transaction date	House Age	Distance from nearest Metro station (km)	Number of convenience stores	latitude	longitude	Number of bedrooms	House size (sqft)	House price of unit area
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
409	False	False	False	False	False	False	False	False	False
410	False	False	False	False	False	False	False	False	False
411	False	False	False	False	False	False	False	False	False
412	False	False	False	False	False	False	False	False	False
413	False	False	False	False	False	False	False	False	False

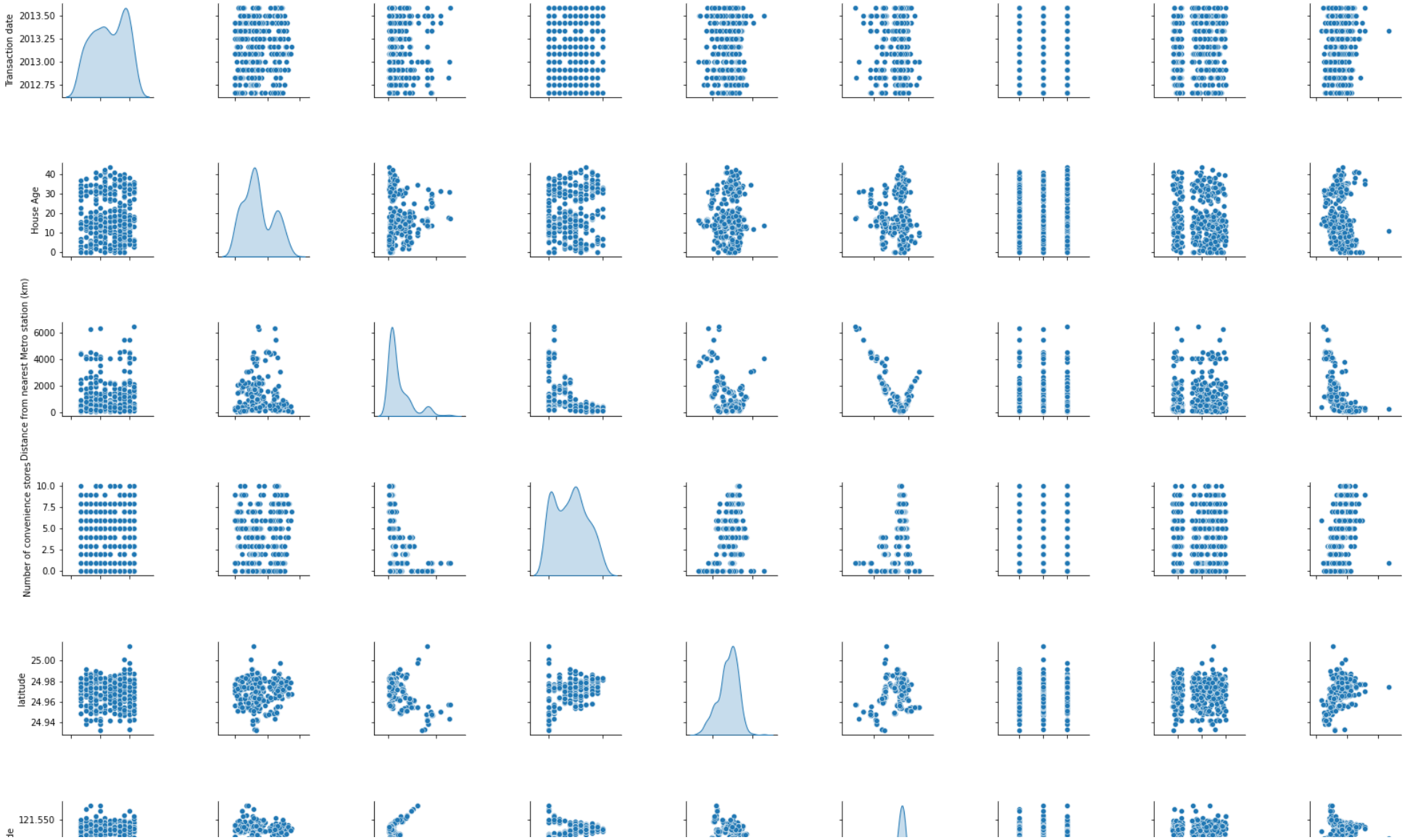
414 rows × 9 columns

Feature Selection

We will involve channel technique for highlight determination. In this strategy, sifting is finished utilizing relationship network and it is most regularly done utilizing corrleation and VIF.

```
sb.pairplot(data,diag_kind="kde")
```

```
<seaborn.axisgrid.PairGrid at 0x7f307e603fd0>
```



Missing Values.

```
# Definig the function to find the missing Values
```

```
df_miss = pd.DataFrame(100*data.isnull().sum()/data.shape[0]).reset_index()
```

```
df_miss
```

	index	0
0	Transaction date	0.0
1	House Age	0.0
2	Distance from nearest Metro station (km)	0.0
3	Number of convenience stores	0.0
4	latitude	0.0
5	longitude	0.0
6	Number of bedrooms	0.0
7	House size (sqft)	0.0
8	House price of unit area	0.0

Characterizing Data sources and the Objectives

```
X=data.drop('House price of unit area', axis=1)
y= data['House price of unit area']
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=365)
```

```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
reg = LinearRegression()
```

```
reg.fit(x_train,y_train)
y_hat = reg.predict(x_train)

reg.score(x_train,y_train).round(3)
```

```
0.581
```

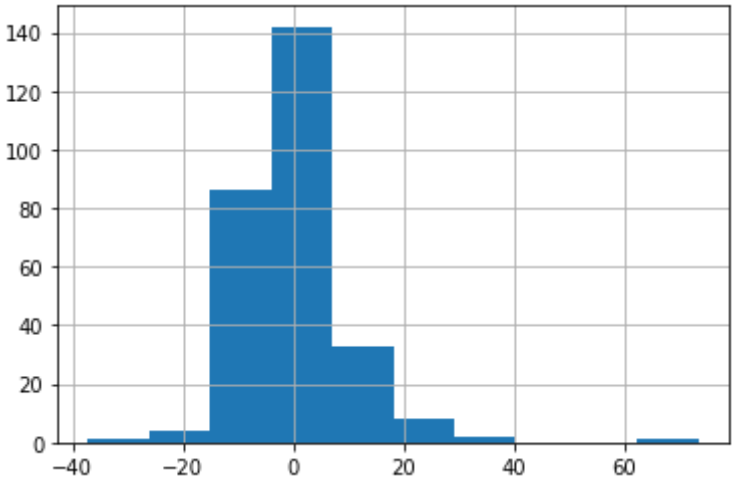
58.10% of the information fit the relapse model.

```
y_resid = y_train - y_hat

fig, ax = plt.subplots(1,2, constrained_layout = True)

sb.regplot(x=y_hat, y=y_train, ax=ax[0], line_kws={'color': 'red'})
ax[0].set_title('Observed vs. Predicted Values', fontsize=10)
ax[0].set(xlabel='Predicted', ylabel='Observed')

sb.regplot(x=y_hat, y=y_resid, ax=ax[1], line_kws={'color': 'red'})
ax[1].set_title('Residuals vs. Predicted Values', fontsize=10)
ax[1].set(xlabel='Predicted', ylabel='Residuals')
```

```
[Text(0, 0.5, 'Residuals'), Text(0.5, 0, 'Predicted')]
```

The focuses are not evenly conveyed around the slanting line neither around the level line. Thus, there is infringement of the linearity and homogeneity presumption.

```
y_resid.hist()
plt.show()
```



The residuals are slanted on the right or we can say the circulation of residuals is non-typical. Thus, it further abuses the OLS suspicions

Disregarding these infringement, I will test the model.

Finding the Weights and Bias

```
intercept = reg.intercept_.round(3)
```

```
coeff = reg.coef_.round(3)
coeff
```

```
array([ 5.37000e+00, -2.29000e-01, -5.00000e-03,  1.11700e+00,
        2.29063e+02, -3.04170e+01, -1.23000e+00,  3.00000e-03])
```

Testing

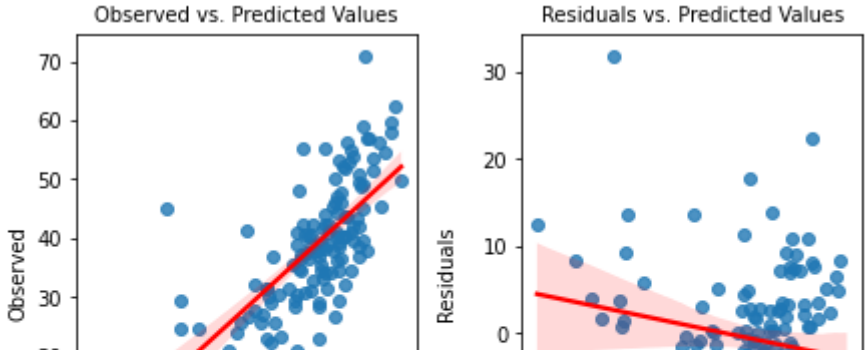
```
y_hat_test = reg.predict(x_test)
```

```
reg.score(x_test,y_test).round(3)
```

```
0.571
```

```
y_resid_test = y_test - y_hat_test

fig, ax = plt.subplots(1,2, constrained_layout = True)

sb.regplot(x=y_hat_test, y=y_test, ax=ax[0], line_kws={'color': 'red'})
ax[0].set_title('Observed vs. Predicted Values', fontsize=10)
ax[0].set(xlabel='Predicted', ylabel='Observed')

sb.regplot(x=y_hat_test, y=y_resid_test, ax=ax[1], line_kws={'color': 'red'})
ax[1].set_title('Residuals vs. Predicted Values', fontsize=10)
ax[1].set(xlabel='Predicted', ylabel='Residuals')
```

```
[Text(0, 0.5, 'Residuals'), Text(0.5, 0, 'Predicted')]
```



Our straight model isn't proper as it disregards the OLS suppositions of Ordinariness and homogeneity. There are ways of fixing these issues which requires definite concentrate on my part as I'm a fledgling in ML. Up until this point I delighted in playing out this examination, and my comprehension on relapse procedure has expanded. You advance by doing.

Predicted Predicted

Random Forest Regression

An irregular woods is a meta assessor that fits various grouping choice trees on different sub-tests of the dataset and utilizes averaging to work on the prescient exactness and command over-fitting.

```
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
```

Instantiate a fresh instance of RandomForestRegressor

```
num_pipeline = Pipeline([('std_scaler', StandardScaler())])
prepared_data = num_pipeline.fit_transform(X)
```

```
forest_reg = RandomForestRegressor()
forest_reg.fit(prepared_data, y)
```

```
RandomForestRegressor()
```

```
from sklearn.ensemble import RandomForestRegressor
forest_reg = RandomForestRegressor()
forest_reg.fit(prepared_data, y)
```

```
RandomForestRegressor()
```

Now we can evaluate the loaded model to see if its predictions line up with the predictions made prior to saving.Disregarding these infringement, I will test the model.

```
dataset_predictions = forest_reg.predict(prepared_data)
forest_mse = mean_squared_error(y, dataset_predictions)
forest_rmse = np.sqrt(forest_mse)
forest_rmse
```

```
2.756487845347026
```

Prediction computed with out-of-bag estimate on the training set.

```
def display_scores(scores):
    print('Scores:', scores)
    print('Mean:', scores.mean())
    print('Standard deviation:', scores.std())
```

```
forest_scores = cross_val_score(forest_reg, prepared_data, y,
                            scoring='neg_mean_squared_error', cv=10)
```

```
forest_rmse_scores = np.sqrt(-forest_scores)
display_scores(forest_rmse_scores)
```

```
Scores: [ 6.22444252  6.56749552  7.50794688  8.08345404  5.25486614  7.04208826
 13.03373362  7.65729715  7.17983522  4.85065906]
Mean: 7.340181840785587
Standard deviation: 2.1354261176265
```

This mean squared error result is lower than our base model which is great to see but overall, I'd still consider this performance inadequate. A root mean of 7.369 means that the average error per estimate.

Using Neural Networks

```
inputs=X.to_numpy()
s_input=StandardScaler()
inputs=s_input.fit_transform(inputs)
```

benchmark base network model for the relapse issue. Beginning with the required capabilities in general and items.

```
target=y.to_numpy()
```

```
np.savez('tf_data',inputs=inputs,targets=target)
```

```
datatf=np.load('tf_data.npz')
inputss=datatf['inputs']
targets=datatf['targets']
print(targets.shape)
output_size=1
input_size=inputss.shape[1]
print(input_size)
```

```
(414,)
8
```

```
import tensorflow as tf
model=tf.keras.Sequential([tf.keras.layers.Dense(8,input_shape=(8,),activation='relu'),tf.keras.layers.Dense(8,activation='relu'),tf

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.1),loss='mean_squared_error')
```

We have 8 highlights, we should embed 8 neurons as a beginning, 4 secret layers, and 1 result layer due to anticipated house costs.

Likewise, ADAM streamlining calculation is utilized for upgrading misfortune capability (Mean squared blunder

```
model.fit(inputss,targets,epochs=1500,verbose=1)
weights=model.layers[0].get_weights()[0]
bias=model.layers[0].get_weights()[1]
```

```
Epoch 1/1500
13/13 [==============================] - 2s 2ms/step - loss: 700.4231
Epoch 2/1500
13/13 [==============================] - 0s 3ms/step - loss: 142.6370
Epoch 3/1500
13/13 [==============================] - 0s 2ms/step - loss: 83.7754
Epoch 4/1500
13/13 [==============================] - 0s 2ms/step - loss: 68.8874
Epoch 5/1500
13/13 [==============================] - 0s 3ms/step - loss: 65.4075
Epoch 6/1500
13/13 [==============================] - 0s 2ms/step - loss: 63.8021
Epoch 7/1500
13/13 [==============================] - 0s 2ms/step - loss: 61.4761
Epoch 8/1500
13/13 [==============================] - 0s 2ms/step - loss: 59.4630
Epoch 9/1500
13/13 [==============================] - 0s 2ms/step - loss: 57.8450
Epoch 10/1500
13/13 [==============================] - 0s 2ms/step - loss: 68.7764
Epoch 11/1500
13/13 [==============================] - 0s 2ms/step - loss: 61.6667
```

```
Epoch 12/1500
13/13 [==============================] - 0s 2ms/step - loss: 57.7122
Epoch 13/1500
13/13 [==============================] - 0s 2ms/step - loss: 57.7072
Epoch 14/1500
13/13 [==============================] - 0s 2ms/step - loss: 64.0765
Epoch 15/1500
13/13 [==============================] - 0s 2ms/step - loss: 58.5327
Epoch 16/1500
13/13 [==============================] - 0s 3ms/step - loss: 58.2511
Epoch 17/1500
13/13 [==============================] - 0s 2ms/step - loss: 55.8149
Epoch 18/1500
13/13 [==============================] - 0s 2ms/step - loss: 59.2729
Epoch 19/1500
13/13 [==============================] - 0s 2ms/step - loss: 57.5307
Epoch 20/1500
13/13 [==============================] - 0s 2ms/step - loss: 57.6623
Epoch 21/1500
13/13 [==============================] - 0s 2ms/step - loss: 60.6311
Epoch 22/1500
13/13 [==============================] - 0s 2ms/step - loss: 54.3479
Epoch 23/1500
13/13 [==============================] - 0s 2ms/step - loss: 58.8424
Epoch 24/1500
13/13 [==============================] - 0s 2ms/step - loss: 59.3245
Epoch 25/1500
13/13 [==============================] - 0s 2ms/step - loss: 60.2881
Epoch 26/1500
13/13 [==============================] - 0s 3ms/step - loss: 58.2622
Epoch 27/1500
13/13 [==============================] - 0s 2ms/step - loss: 55.2858
Epoch 28/1500
13/13 [==============================] - 0s 3ms/step - loss: 58.9957
Epoch 29/1500
13/13 [==============================] - 0s 3ms/step - loss: 58.3310
```

Then, train the model for auto epochs, recording the preparation and approval exactness in the set of experiences each time. The model will run both training and testing information while working out the misfortune capability to monitor how well the model performs for each age.

```
prediction=model.predict(inputss)
```

```
13/13 [==============================] - 0s 3ms/step
```

```
prediction
```

```
array([[41.490845],
       [28.455763],
       [16.378826],
       [15.593117],
       [16.378826],
       [39.991222],
       [46.796112],
       [27.543499],
       [37.79697 ],
       [34.92038 ],
       [34.614   ],
       [40.38417 ],
       [61.225655],
       [40.85304 ],
       [35.321617],
       [53.649834],
       [41.49328 ],
       [41.38197 ],
       [49.13858 ],
       [30.061031],
       [16.378826],
       [53.887672],
       [48.64689 ],
       [38.35095 ],
       [29.693966],
       [24.848036],
       [32.46547 ],
       [27.24708 ],
       [57.240524],
       [41.79246 ],
       [48.676453],
       [51.890137],
       [27.24708 ],
       [44.473526],
       [50.23957 ],
```

```
[50.23957 ],
[40.407192],
[49.604652],
[36.492683],
[26.564402],
[27.24708 ],
[31.16143 ],
[48.03907 ],
[28.399807],
[45.039642],
[52.298958],
[40.220665],
[63.19677 ],
[52.19725 ],
[27.314878],
[29.353588],
[51.99122 ],
[38.375477],
[59.634552],
[52.224625],
[54.135612],
[27.24708 ],
[35.94049 ],
[16.378826],
[46.787857],
```

shock, this score can be further developed through highlight determination or utilizing other relapse models.

```
df = pd.DataFrame (prediction) # converting in to dataframe of all predicitons.
```

```
csv = pd.read_excel("/content/DS - Assignment Part 1 data set.xlsx") # reading the xlsx.
```

```
csv['prediction']=df #stored in as a new parameter.
```

```
csv.to_csv('output.csv')
```

The focuses are not evenly disseminated around the slanting line neither around the flat line. Subsequently, there is infringement of the linearity and homogeneity presumption.

Colab paid products - Cancel contracts here

✓ 0s completed at 6:27 PM