Predicting House Prices using Machine Learning

Phase 4: Development Part 2

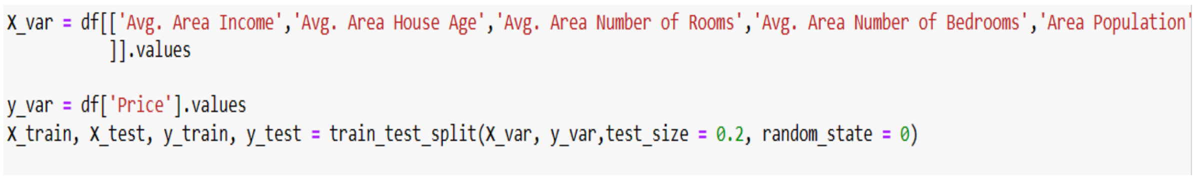
Continue building the house price prediction model by feature selection, model training and evaluation.

Coding:

Model Training and evaluation:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
```

#Train and test the data



```
X_var = df[['Avg. Area Income','Avg. Area House Age','Avg. Area Number of Rooms','Avg. Area Number of Bedrooms','Area Population'
        ]].values

y_var = df['Price'].values
X_train, X_test, y_train, y_test = train_test_split(X_var, y_var,test_size = 0.2, random_state = 0)
```

print(y_train)

```
[1305210.26495953 1400961.27878845 1048639.78949745 ... 1102641.11402327
 865099.52337233 2108376.16580335]
```

Linear Regression:

#Mean Squared error

```
#mse linear
model=LinearRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
print("Mean Squared Error:",mse)
```

Out []:

Mean Squared Error: 10549721686.160307

#R2 Score

```
#r-squared linear
model=LinearRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
r_squared=r2_score(y_test,y_pred)
print("R2 Score:",r_squared)
```

Out []:

```
R2 Score: 0.9146454505137986
```

#Mean Absolute error

```
#mae linear
model=LinearRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
mae=mean_absolute_error(y_test,y_pred)
print("Mean Absolute error:",mae)
```

Out []:

```
Mean Absolute error: 82657.9460589243
```

Lasso Regression:

#Mean Squared error

```
#mse lasso
model=Lasso(alpha=0.5)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
print("Mean Squared Error:",mse)
```

Out []:

Mean Squared Error: 10549719672.37874

#R2 Score

```
#r-squared lasso
model=Lasso(alpha=0.5)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
r_squared=r2_score(y_test,y_pred)
print("R2 Score:",r_squared)
```

Out []:

R2 Score: 0.9146454668066851

#Mean Absolute error

```
#mae lasso
model=Lasso(alpha=0.5)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
mae=mean_absolute_error(y_test,y_pred)
print("Mean Absolute Error:",mae)
```

Out []:

Mean Absolute Error: 82657.94633403154

Support Vector Machines:

#Mean Squared error

```
#mse svr
svr=SVR(C=100000)
svr.fit(X_train,y_train)
y_pred=svr.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
print("Mean Squared Error:",mse)
```

Out []:

Mean Squared Error: 51764160088.32387

#R2 Score

```
#r-squared svr
svr=SVR(C=100000)
svr.fit(X_train,y_train)
y_pred=svr.predict(X_test)
r_squared=r2_score(y_test,y_pred)
print("R2 Score:",r_squared)
```

Out []:

R2 Score: 0.5811921209574027

#Mean Absolute error

```
#mae lasso
model=Lasso(alpha=0.5)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
mae=mean_absolute_error(y_test,y_pred)
print("Mean Absolute Error:",mae)
```

Out []:

```
Mean Absolute error: 183596.37376356052
```

Random Forest Regressor:

#Mean Squared error

```
#mse randomforest
random_forest=RandomForestRegressor(n_estimators=100)
random_forest.fit(X_train,y_train)
y_pred=random_forest.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
print("Mean Squared Error:",mse)
```

Out []:

```
Mean Squared Error: 15029091690.717777
```

#R2 Score

```
#r-squared randomforest

random_forest=RandomForestRegressor(n_estimators=100)
random_forest.fit(X_train,y_train)
y_pred=random_forest.predict(X_test)
r_squared=r2_score(y_test,y_pred)
print("R2 Score:",r_squared)
```

Out []:

```
R2 Score: 0.8792930559273248
```

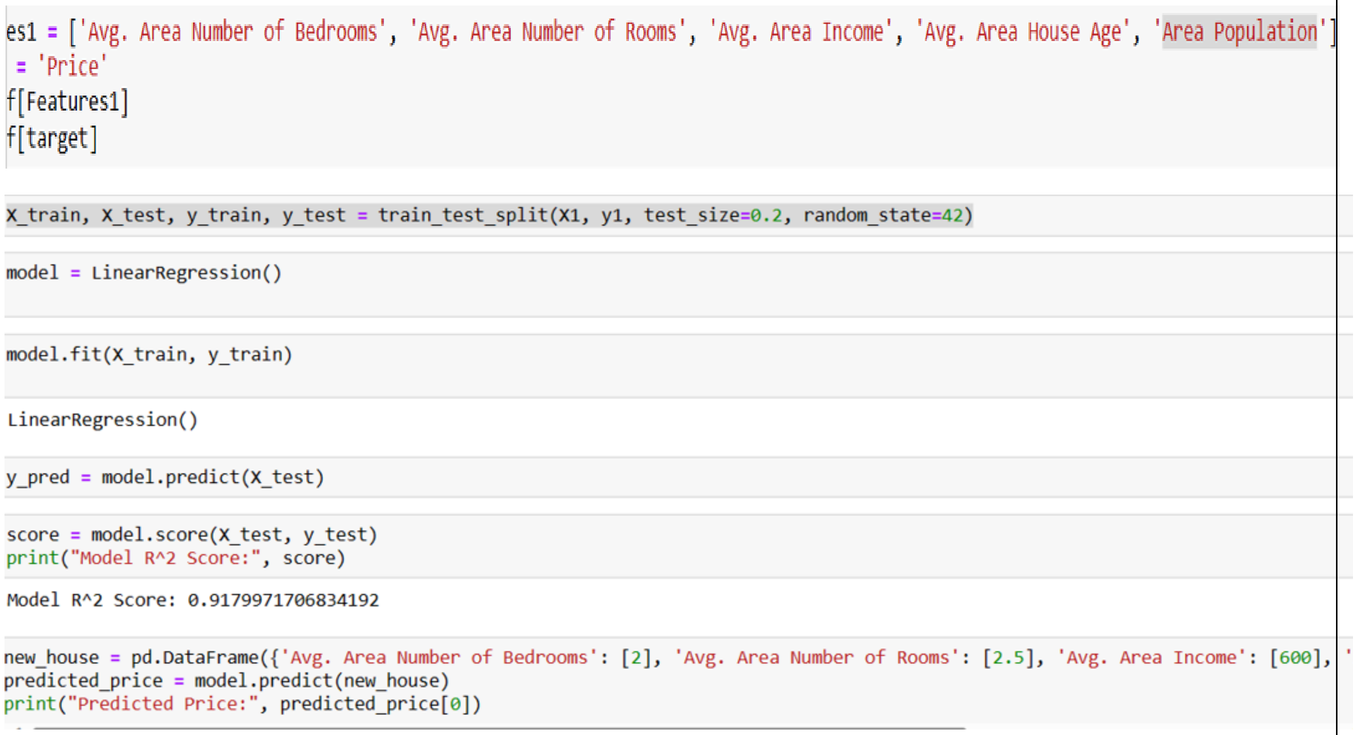
#Mean Absolute error

```
#mae randomforest
random_forest=RandomForestRegressor(n_estimators=100)
random_forest.fit(X_train,y_train)
y_pred=random_forest.predict(X_test)
mae=mean_absolute_error(y_test,y_pred)
print("Mean Absolute error:",mae)
```

Out []:

Mean Absolute error: 97483.63407534623

Evaluation of Predicted Price using Linear Regression:



```
es1 = ['Avg. Area Number of Bedrooms', 'Avg. Area Number of Rooms', 'Avg. Area Income', 'Avg. Area House Age', 'Area Population']
 = 'Price'
f[Features1]
f[target]
```

```
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.2, random_state=42)
```

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
y_pred = model.predict(X_test)
```

```
score = model.score(X_test, y_test)
print("Model R^2 Score:", score)
```

```
Model R^2 Score: 0.9179971706834192
```

```
new_house = pd.DataFrame({'Avg. Area Number of Bedrooms': [2], 'Avg. Area Number of Rooms': [2.5], 'Avg. Area Income': [600], '
predicted_price = model.predict(new_house)
print("Predicted Price:", predicted_price[0])
```

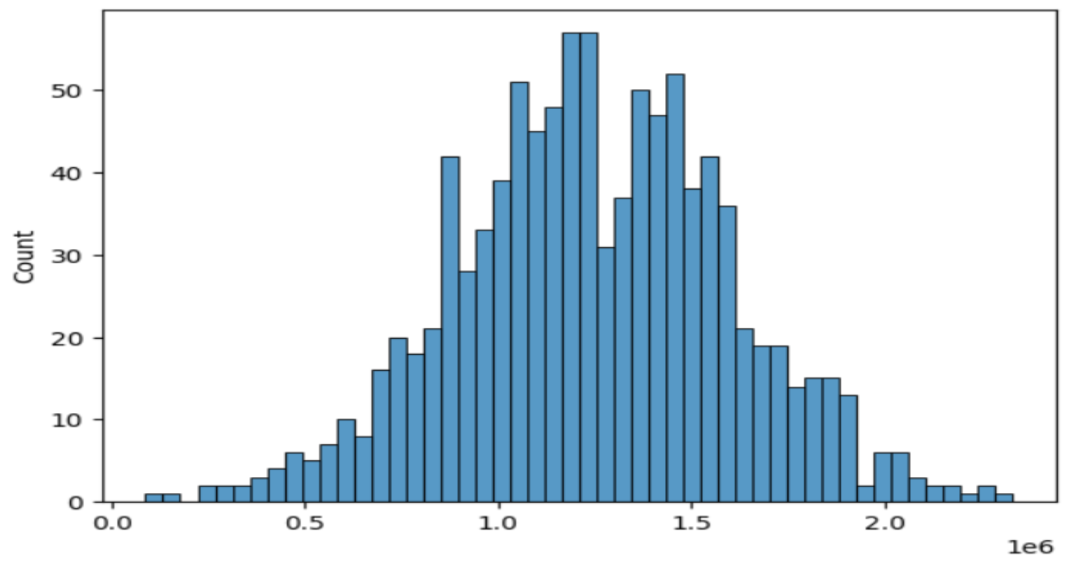
Out []:

```
Predicted Price: 96481778.18208618
```

sns.histplot((y_test),bins=50)

out []:

```
<AxesSubplot:ylabel='Count'>
```



Split the data:

```
print("\n Data Spliting")
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.2, random_state=42)
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: { y_test.shape}")
```

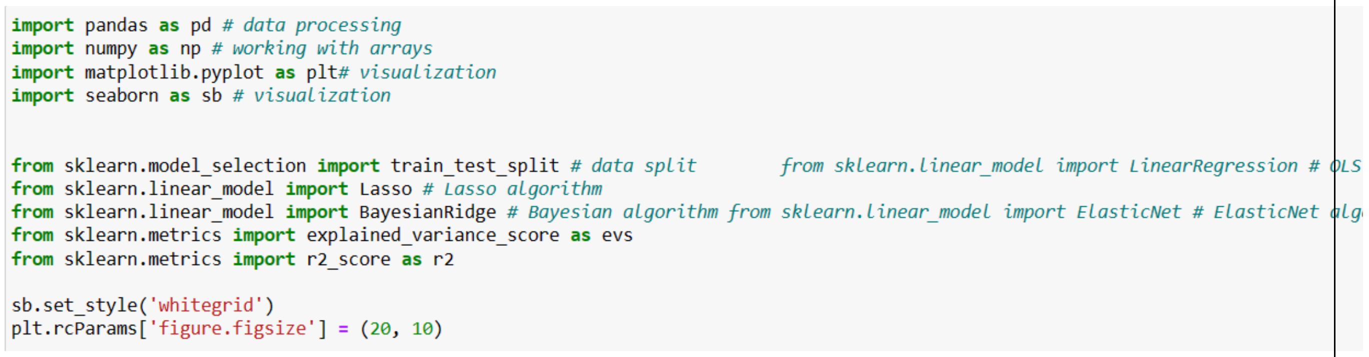
```
 Data Spliting
X_train shape: (4000, 5)
X_test shape: (1000, 5)
y_train shape: (4000,)
y_test shape: (1000,)
```

Train and Test Samples:

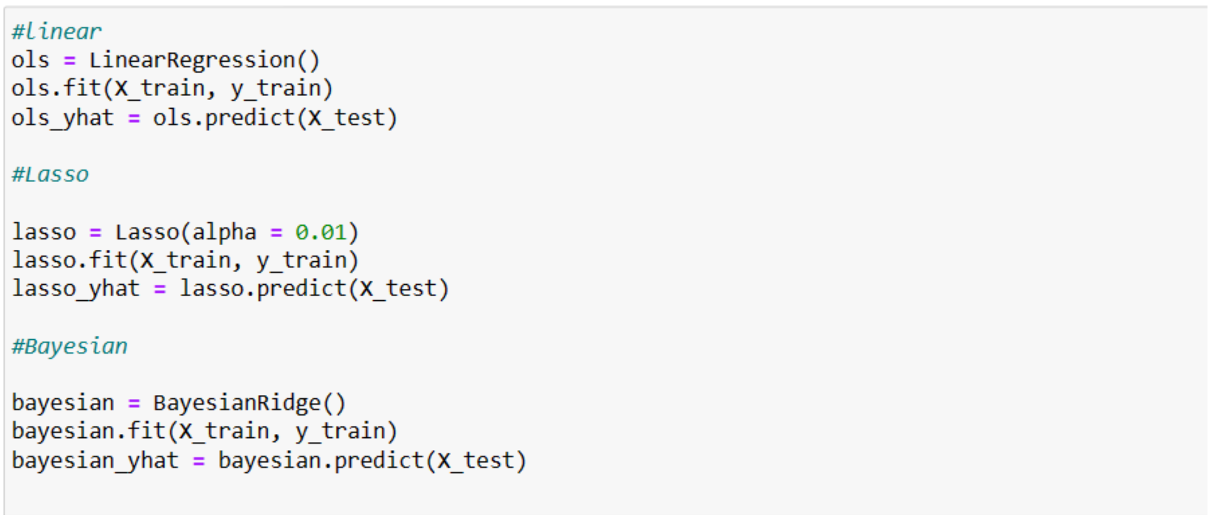
```
X_var = df[['Avg. Area Income','Avg. Area House Age','Avg. Area Number of Rooms','Avg. Area Number of Bedrooms','Area Population'
           ]].values

y_var = df['Price'].values
X_train, X_test, y_train, y_test = train_test_split(X_var, y_var,test_size = 0.2, random_state = 0)

print(('X_train samples : '), X_train[0:5])
print(('X_test samples : '), X_test[0:5])
print(('y_train samples : '), y_train[0:5])
print(('y_test samples : '), y_test[0:5])
```

```
X_train samples :  [[8.01962423e+04 6.67569707e+00 7.27519280e+00 3.17000000e+00
  4.86948641e+04]
 [7.41306063e+04 6.91966289e+00 8.26699440e+00 3.24000000e+00
  4.99585810e+04]
 [6.73840004e+04 7.22428091e+00 7.80991877e+00 6.43000000e+00
  4.89180554e+04]
 [5.95695373e+04 6.27953689e+00 7.32537954e+00 4.24000000e+00
  3.12946525e+04]
 [5.83852154e+04 7.58855882e+00 6.40611758e+00 2.30000000e+00
  4.19303750e+04]]
X_test samples :  [[6.12007262e+04 5.29969400e+00 6.23461464e+00 4.23000000e+00
  4.27896922e+04]
 [6.33808147e+04 5.34466404e+00 6.00157433e+00 2.45000000e+00
  4.02173336e+04]
 [7.12082693e+04 5.30032605e+00 6.07798886e+00 4.01000000e+00
  2.56963617e+04]
 [5.03437635e+04 6.02746799e+00 5.16023950e+00 4.35000000e+00
  2.74458767e+04]
 [5.45354537e+04 5.27806515e+00 6.87103756e+00 4.41000000e+00
  3.08522070e+04]]
y_train samples :  [1616937.01868768 1881075.43794209 1930344.44890875  885920.55324936
 1266209.75320911]
y_test samples :  [894251.06863578 932979.36062132 920747.91128789 691854.921027
 732733.23629305]
```

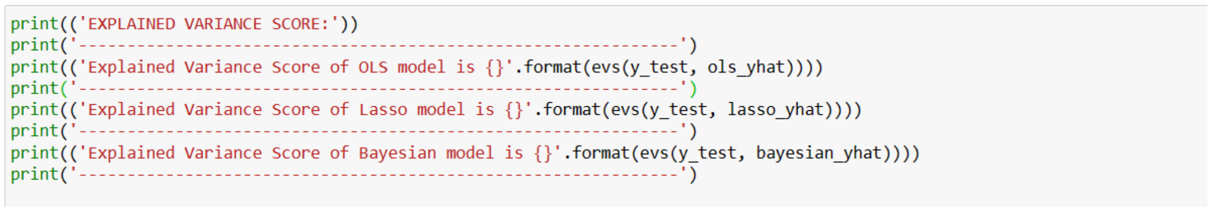
Explained variance score:



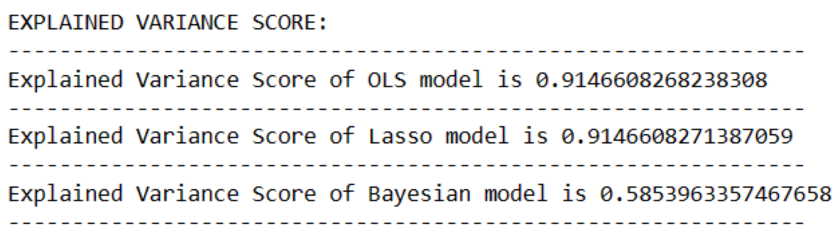
```
import pandas as pd # data processing
import numpy as np # working with arrays
import matplotlib.pyplot as plt# visualization
import seaborn as sb # visualization


from sklearn.model_selection import train_test_split # data split        from sklearn.linear_model import LinearRegression # OLS
from sklearn.linear_model import Lasso # Lasso algorithm
from sklearn.linear_model import BayesianRidge # Bayesian algorithm from sklearn.linear_model import ElasticNet # ElasticNet alg
from sklearn.metrics import explained_variance_score as evs
from sklearn.metrics import r2_score as r2

sb.set_style('whitegrid')
plt.rcParams['figure.figsize'] = (20, 10)
```



```
#linear
ols = LinearRegression()
ols.fit(X_train, y_train)
ols_yhat = ols.predict(X_test)

#Lasso

lasso = Lasso(alpha = 0.01)
lasso.fit(X_train, y_train)
lasso_yhat = lasso.predict(X_test)

#Bayesian

bayesian = BayesianRidge()
bayesian.fit(X_train, y_train)
bayesian_yhat = bayesian.predict(X_test)
```



```
print(('EXPLAINED VARIANCE SCORE:'))
print('-------------------------------------------------------------------')
print(('Explained Variance Score of OLS model is {}'.format(evs(y_test, ols_yhat))))
print('-------------------------------------------------------------------')
print(('Explained Variance Score of Lasso model is {}'.format(evs(y_test, lasso_yhat))))
print('-------------------------------------------------------------------')
print(('Explained Variance Score of Bayesian model is {}'.format(evs(y_test, bayesian_yhat))))
print('-------------------------------------------------------------------')
```



```
EXPLAINED VARIANCE SCORE:
-------------------------------------------------------------------
Explained Variance Score of OLS model is 0.9146608268238308
-------------------------------------------------------------------
Explained Variance Score of Lasso model is 0.9146608271387059
-------------------------------------------------------------------
Explained Variance Score of Bayesian model is 0.5853963357467658
-------------------------------------------------------------------
```