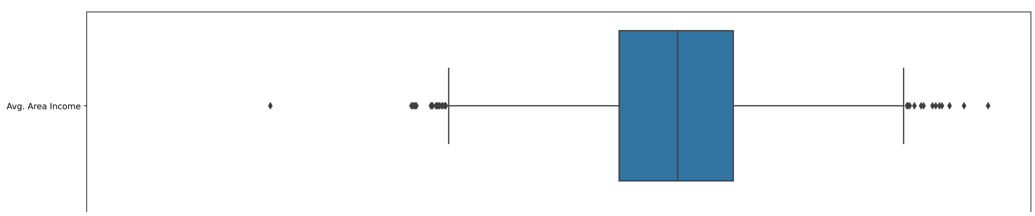
```
import pandas as pd
import numpy as np
import pickle

#visualisation
import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import seaborn as sns

#sns.set_style('whitegrid')

# Model
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, SGDRegressor
from sklearn.linear_model import ElasticNet

#perfomance
from sklearn.metrics import mean_squared_error,r2_score

# Load the dataset
df= pd.read_csv("USA_Housing.csv")


df.head()
```

```
df.drop('Address',axis=1,inplace=True)
```

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

```
Avg. Area Income                0
Avg. Area House Age             0
Avg. Area Number of Rooms       0
Avg. Area Number of Bedrooms    0
Area Population                 0
Price                           0
dtype: int64
```

```
plt.figure(figsize=(20,8),dpi=400)
sns.boxplot(data=df[[ 'Avg. Area Income', 'Area Population']],orient='h')
plt.show()
```




```
# Split the data into training and testing sets
X = df.drop("Price", axis=1)  # Features
y = df["Price"]  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

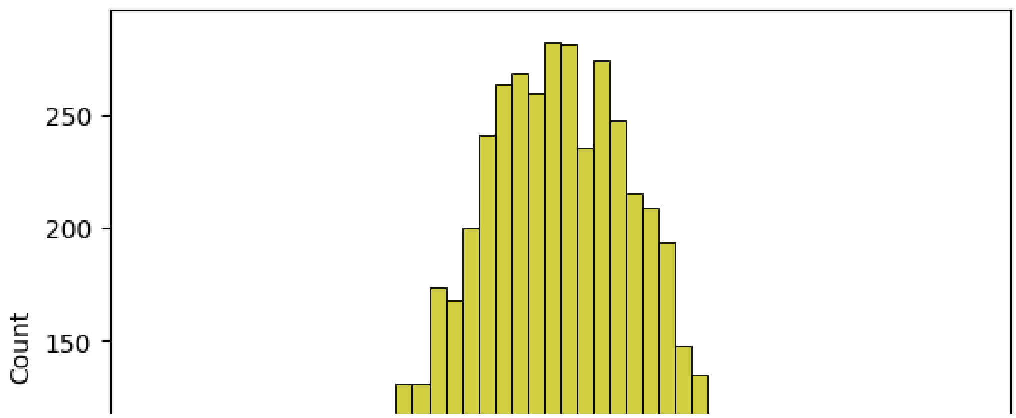
```
X = df.drop(['Price'],axis=1)
y = df['Price']
print(X.shape)
```

```
(5000, 5)
```

** DATA VISUALIZATION**

```
sns.histplot(df, x='Price', bins=50, color='y')
```

```
<Axes: xlabel='Price', ylabel='Count'>
```



Dividing Dataset in to features and target variable

```
X = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
Y = df['Price']
```

Using Train Test Split

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=101)
```

```
Y_train.head()
```

```
3413    1.305210e+06
1610    1.400961e+06
3459    1.048640e+06
4293    1.231157e+06
1039    1.391233e+06
Name: Price, dtype: float64
```

Standardizing the data

```
sc = StandardScaler()
X_train_scal = sc.fit_transform(X_train)
X_test_scal = sc.fit_transform(X_test)
```

Model Building and Evaluation

```
model_lr=LinearRegression()
```

```
model_lr.fit(X_train_scal, Y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Predicting Prices

```
Prediction1 = model_lr.predict(X_test_scal)
```

```
model_svr = SVR()
```

```
model_svr.fit(X_train_scal, Y_train)
```

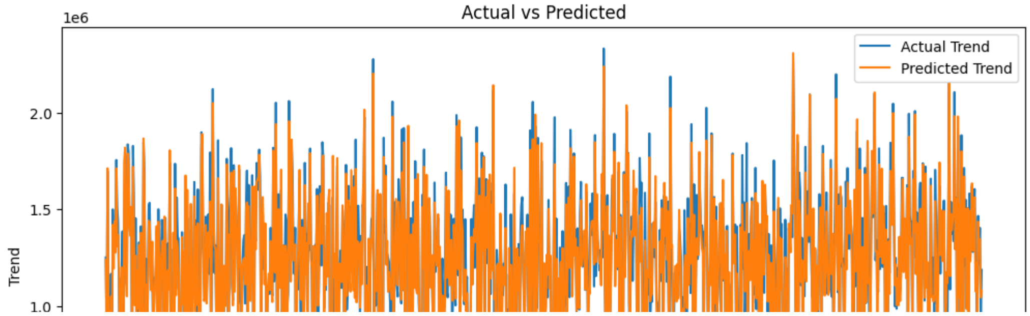
```
Prediction2 = model_svr.predict(X_test_scal)
```

Evaluation of Predicted Data

```
plt.figure(figsize=(12,6))
```

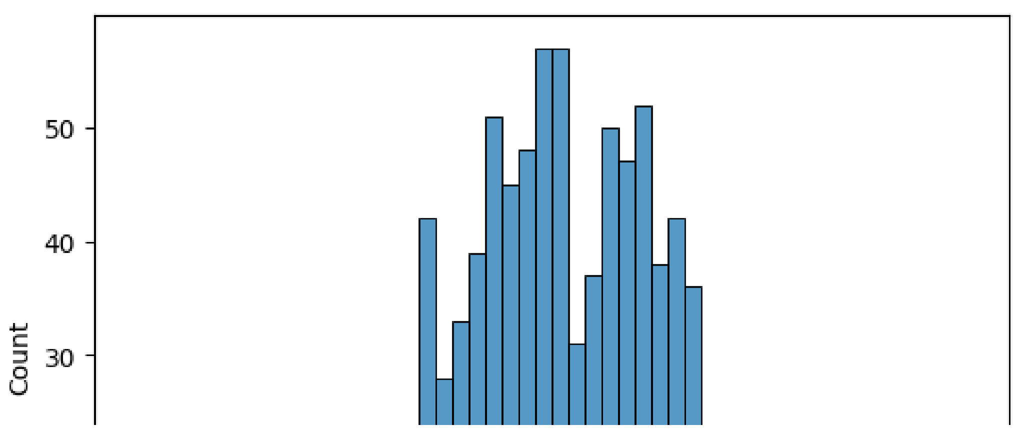
```
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction1, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```



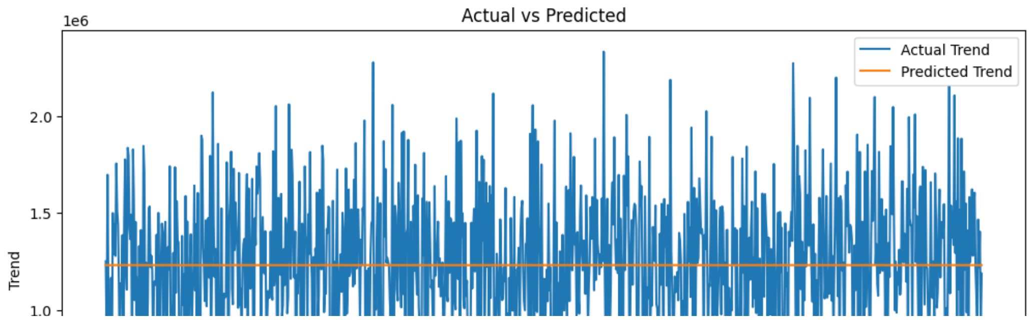

```
sns.histplot((Y_test-Prediction2), bins=50)
```

```
<Axes: xlabel='Price', ylabel='Count'>
```



```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction2, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```



```
print(r2_score(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

```
-0.0006222175925689744
128209033251.4034
128209033251.4034
```