House price prediction using SVR

In [1]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import warnings
warnings.filterwarnings('ignore')
```

Reading Dataset

In [2]:
```
df = pd.read_csv('data/USA_Housing.csv')
df.head()
```

Out[2]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

In [3]:
```
df.shape
```

Out[3]:
```
(5000, 7)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
```

```
 6   Address                        5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

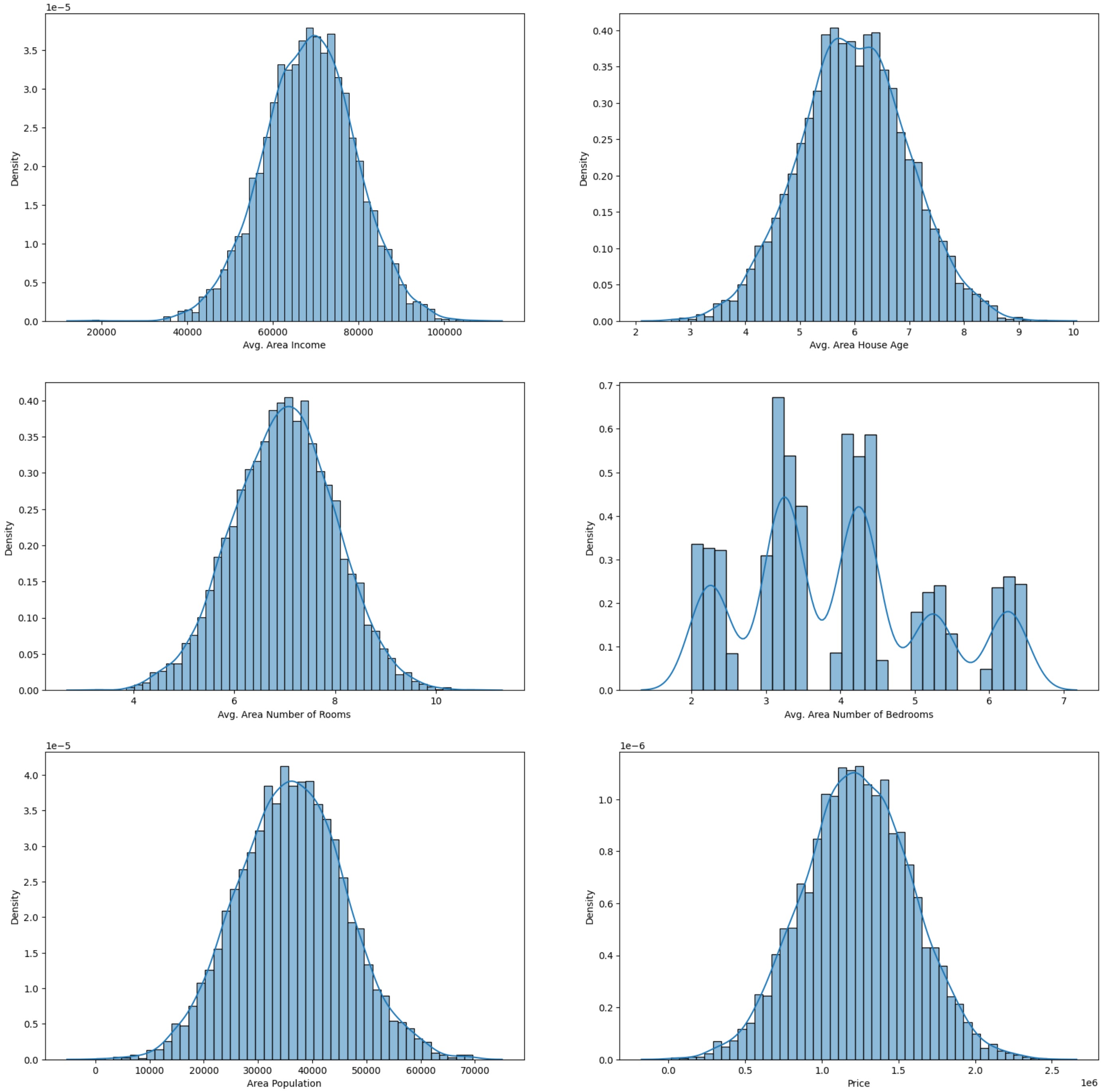
Exploratory Data Analysis

In [5]:
```python
df.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	
Avg. Area Income	5000.0	6.858311e+04	10657.991214	17796.631190	61480.562388	6.880429e+04	7.578334e+04	1.077
Avg. Area House Age	5000.0	5.977222e+00	0.991456	2.644304	5.322283	5.970429e+00	6.650808e+00	9.519
Avg. Area Number of Rooms	5000.0	6.987792e+00	1.005833	3.236194	6.299250	7.002902e+00	7.665871e+00	1.075
Avg. Area Number of Bedrooms	5000.0	3.981330e+00	1.234137	2.000000	3.140000	4.050000e+00	4.490000e+00	6.500
Area Population	5000.0	3.616352e+04	9925.650114	172.610686	29403.928702	3.619941e+04	4.286129e+04	6.962
Price	5000.0	1.232073e+06	353117.626581	15938.657923	997577.135049	1.232669e+06	1.471210e+06	2.469

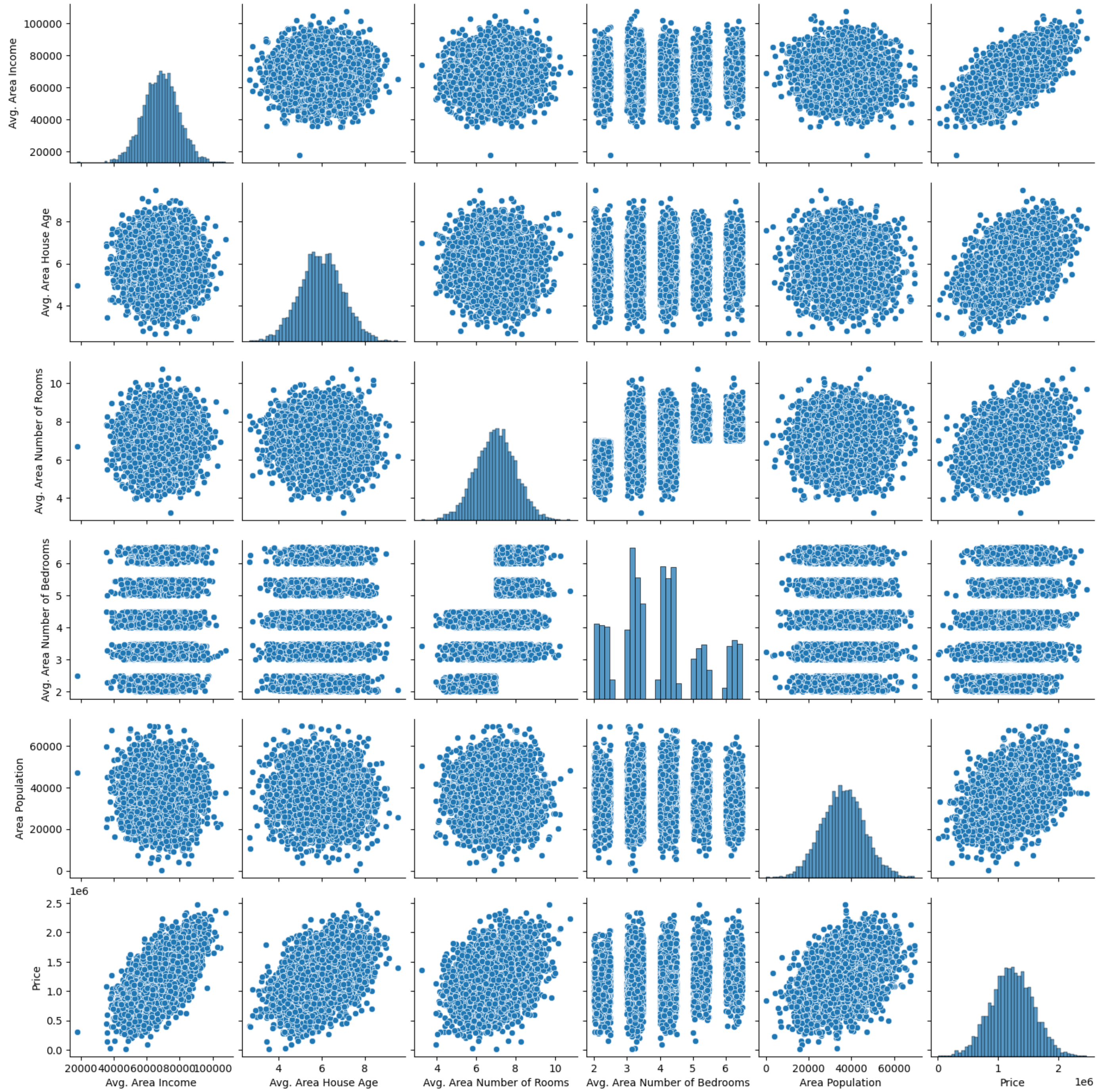
In [6]:
```python
#plotting distributions of features

num_cols = df.columns[:-1]

plt.figure(figsize=(20,20))
for i, cols in enumerate(num_cols):
    plt.subplot(3, 2, i+1)
    sns.histplot(df[cols], kde=True, stat='density', kde_kws=dict(cut=3))
    plt.title = cols + 'Distribution'
```



- Features other than Avg. Area Number of Bedrooms follow normal distribution
- Target price also follows normal distribution

In [7]:
```
#pairplot
sns.pairplot(df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x22e2045f430>



- The target price has a linear relationship with Avg. Area income
- Avg age, number of rooms and area populations show weak realtionship with price

In [8]:
```
#correlation heatmap

sns.heatmap(df.corr(), annot=True)
```

Out[8]: <AxesSubplot: >

- Correlation matrix confirms that Avg. Area income is the one feature that has higher correlation with price
- Avg. age, number of rooms and area poplulations show weaker correlation
- Avg. Area number of Bedrooms show no to little correlation with price
- There are no multicollenearity between the features
- Avg number of rooms and Avg number or bedrooms show a postive correlation and it is obvious

Data preprocessing

In [9]:
```
#checking for null values
df.isnull().sum()
```

Out[9]:
```
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
Address                         0
dtype: int64
```

In [10]:
```
#checking for duplicate values
```

```
df.duplicated().sum()
```

Out[10]:

```
0
```

In [11]:

```
#dropping Address columns because it seems irrelevent
df = df.drop('Address', axis=1)
```

In [12]:

```
#creating feature matrix and target vector
X = df.drop('Price', axis=1)
y = df['Price']
```

In [13]:

```
#splitting dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

In [14]:

```
#feature scaling
scaler = StandardScaler()
scaler = scaler.fit(X_train)
X_train_scl = scaler.transform(X_train)
X_test_scl = scaler. transform(X_test)
```

Model building

1. Linear regression

In [15]:

```
#training the model
lr = LinearRegression()
lr.fit(X_train_scl, y_train)
```

Out[15]:

```
▾ LinearRegression
LinearRegression()
```

In [16]:

```
#model evaluatoin
y_pred = lr.predict(X_test_scl)

print('r2 score of Linear regression :', r2_score(y_test, y_pred))
print('MAE of Linear regression :', mean_absolute_error(y_test, y_pred))
print('MSE score of Linear regression :', mean_squared_error(y_test, y_pred))
print('RMSE score of Linear regression :', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
r2 score of Linear regression : 0.9215935236936268
MAE of Linear regression : 82494.73770125784
MSE score of Linear regression : 10543597313.625992
RMSE score of Linear regression : 102682.02040097376
```

2. Support Vector Regression

In [17]:

```
# #finding best hyperparametes for svr
# from sklearn.model_selection import GridSearchCV

# params = {'C': [0.1, 1, 10, 100, 1000, 10000, 100000, 1000000],
#                'gamma': [1, 0.1, 0.01],
#                'kernel': ['rbf', 'linear', 'poly'],
#                 'epsilon': [ 0.01, 0.001, 0.0001]}

# svrCV = GridSearchCV(SVR(), params, cv=5, scoring='neg_mean_squared_error')

# svrCV.fit(X_train_scl, y_train)
```

In [32]:
```
svr = SVR(kernel='rbf', C=1000000)
svr.fit(X_train_scl, y_train)

y_pred = svr.predict(X_test_scl)

print('r2 score of Linear regression :', r2_score(y_test, y_pred))
print('MAE of Linear regression :', mean_absolute_error(y_test, y_pred))
print('MSE score of Linear regression :', mean_squared_error(y_test, y_pred))
print('RMSE score of Linear regression :', np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
r2 score of Linear regression : 0.9109467608056634
MAE of Linear regression : 87242.84408855016
MSE score of Linear regression : 11975305328.990774
RMSE score of Linear regression : 109431.73821607136
```

Conclusion

Linear regression model gives slightly better scores

In [ ]:
```
!jupyter nbconvert --to webpdf --allow-chromium-download profit_estimation_of_companies.
```