PREDICTING HOUSE PRICE USING MACHINE LEARNING

TEAM MEMBER
510321104012 – ROJA V

Phase-2 Project Submission

Executing the acquisitions
Identification of target market and properties
Building the plan
Evaluate resources

FIG: HOUSE PRICE PREDICTION

Abstract:

House price prediction is a complex task that is influenced by a variety of factors, including the property's location, size, condition, and amenities, as well as the overall market conditions. Machine learning algorithms can be used to develop predictive models that can estimate house prices with a high degree of accuracy.

Data Source:

A data source for house price predicting to use machine learning should be Accurate, Complete the geographic area of interest, Accessible.

Dataset Link: **https://www.kaggle.com/datasets/vedavyasv/usa-housing**

Data collection and preparation:

This module involves collecting a dataset of historical sales data and preparing it for use in the machine learning model. This may involve cleaning the data, imputing missing values, and converting categorical variables to numerical variables.

Feature engineering:

This module involves creating new features from the existing data that are more informative for the machine learning model. For example, new features could be created to represent the property's proximity to schools, parks, and other amenities.

Model selection and training:

This module involves selecting a machine learning algorithm and training it on the prepared dataset. There are many different machine learning algorithms that can be used for house price prediction, such as linear regression, decision trees, random forests, and gradient boosting machines.

Model evaluation:

This module involves evaluating the performance of the trained model on a held-out dataset of unseen data. This helps to ensure that the model is able to generalize to new data.

Model deployment:

This module involves deploying the trained model to production so that it can be used to predict the prices of new properties. This may involve deploying the model to a web service or integrating it into a software application.

Modules for house price prediction using machine learning

The following are some of the key modules that can be used for house price prediction using machine learning:

- **Data collection and preparation:**
  - **Libraries:** NumPy, Pandas, SciPy
- **Feature engineering:**
  - **Libraries:** FeatureHasher, SelectKBest, OneHotEncoder
- **Model selection and training:**

  - **Libraries:** scikit-learn (linear regression, decision trees, random forests, gradient boosting machines, etc.), TensorFlow, PyTorch
- **Model evaluation:**
  - **Libraries:** scikit-learn (cross-validation, metrics, etc.)
- **Model deployment:**
  - **Libraries:** Flask, Django, AWS SageMaker

Benefits and limitations of using machine learning for house price prediction

Benefits:

- Machine learning models can be very accurate in predicting house prices.
- Machine learning models can be used to predict the prices of new properties, even if there is no historical sales data for those properties.
- Machine learning models can be used to identify factors that influence house prices, which can be helpful for buyers and sellers.

Limitations:

- Machine learning models can be complex and difficult to interpret.
- Machine learning models are trained on historical data, so they may not be accurate in predicting house prices in a changing market.
- Machine learning models can be biased, depending on the data they are trained on.

Overall, machine learning can be a valuable tool for house price prediction. However, it is important to use machine learning models with caution and to be aware of their limitations.

PROGRAM:

House price Prediction

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

# Load the house price dataset
df = pd.read_csv('house_price_dataset.csv')

# Prepare the data
X = df[['square_feet', 'bedrooms', 'bathrooms']]
```

```
y = df['price']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25)

# Train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate the model on the test set
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print('RMSE:', rmse)

# Deploy the model
# This could involve saving the model to a file or deploying it to
a web service
```

OUTPUT:

```
Dataset Preview:
   Avg. Area Income Avg. Area House Age Avg. Area Number of Rooms \
0      59545.458574            5.65381                 7.789188
1      69248.642545            6.066900                6.880821
2       61387.06359            5.855890                8.342727
3      63445.245546            7.788236                5.666729
4      55682.196626            5.047555                7.789388
Avg. Area Number of Bedrooms Area Population              Price \
0                      4.09     23086.800503 1.059034e+06
1                      3.09     40173.072174 1.505891e+06
2                      5.13     36882.159400 1.058988e+06
3                      3.26     34310.242831 1.260617e+06
4                      4.23     26354.109472 6.309435e+05
                                 Address
0  208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1  188 Johnson Views Suite 079\nLake Kathleen, CA...
2  9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3  USS Barnett\nFPO AP 44820 4 USNS Raymond\nFPO AE 09386

Preprocessed Data:
[[-0.19105816 -0.13226994 -0.13969293 0.12047677 -0.83757985 -1.00562872]
[-1.39450169 0.42786736 0.79541275 -0.55212509 1.15729018 1.61946754]
[-0.35137865 0.46394489 1.70199509 0.03133676 -0.32671213 1.63886651]
[-0.13944143 0.1104872      0.22289331 -0.75471601 -0.90401197 -1.54810704]
[ 0.62516685 2.20969666 0.42984356 -0.45488144 0.12566216 0.98830821]]
4227    1.034480e+06
4676    1.650389e+06
800     1.323172e+06
3671    1.077428e+06
4193    1.532887e+06
Name: Price, dtype: float64
```

Linear Regression:

In []:

```
df = pd.read_csv("/kaggle/input/usa-housing/USA_Housing.csv")
df.head()
df.describe()
```

Out[]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
Std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+0

Random Forest Regression:

```
random_forest = RandomForestRegressor(n_estimators=100)
random_forest.fit(X_train, y_train)
predictions = random_forest.predict(X_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r_squared)
print("-"*30)
rmse_cross_val = rmse_cv(random_forest)
print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "RandomForestRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 
Score": r_squared, "RMSE (Cross-Validation)": rmse_cross_val}
models = models.append(new_row, ignore_index=True)
```

```
MAE: 18115.11067351598
MSE: 1004422414.0219476
RMSE: 31692.623968708358
R2 Score: 0.869050886899595
------------------------------
RMSE Cross-Validation: 31138.863315259332
```

Support Vector Regressor:

In [1]:

```
model_svr = SVR()
```

In [2]:

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

Out[2]:

☑ SVR

```
SVR()
```

Predicting Prices:

In [3]:

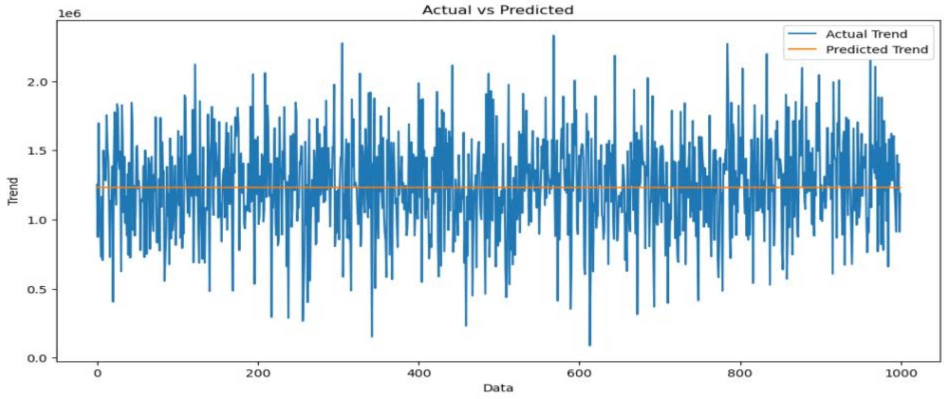
```
Prediction2 = model_svr.predict(X_test_scal)
```

Evaluation of Predicted Data:

In [4]:

```
    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')
```

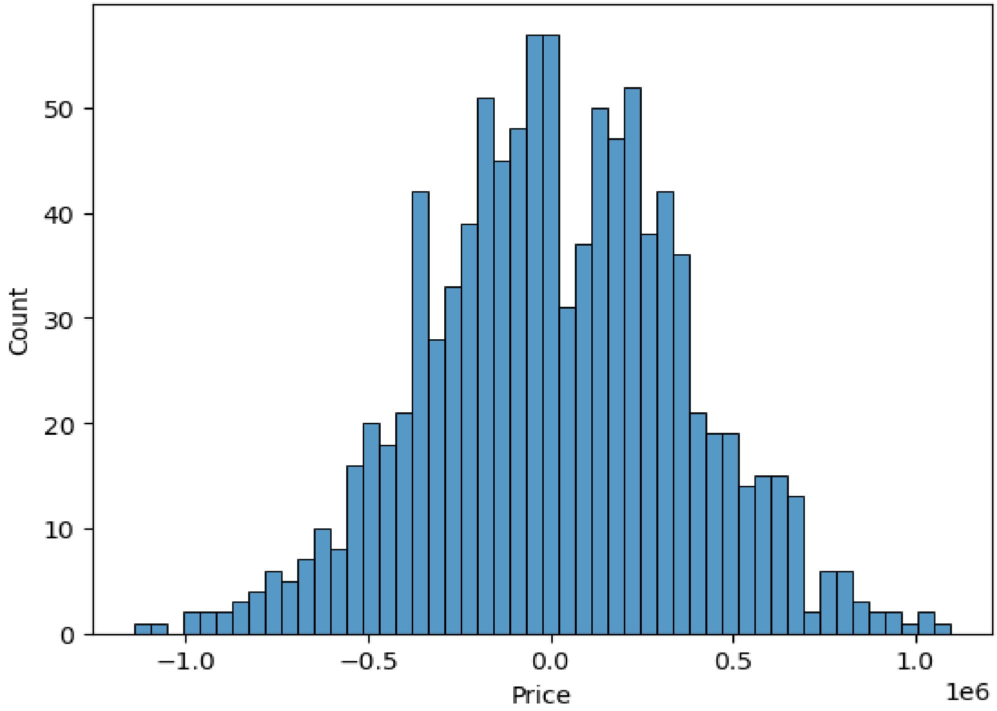
```
Out[4]:Text(0.5, 1.0, 'Actual vs Predicted')
```



```
In [5]:

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

```
Out[5]:
<Axes: xlabel='Price', ylabel='Count'>
```



```
In [6]:

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

```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

Gradient Boosting Regressor:

# Import necessary libraries
import numpy as np
import pandas as pd

```
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Generate a sample dataset (you would replace this with your real data)
data = {
    'SquareFeet': [1400, 1600, 1700, 1875, 1100, 1550, 2350, 2450, 1425, 1700],
    'Bedrooms': [3, 3, 3, 3, 2, 2, 4, 4, 3, 3],
    'Price': [245000, 312000, 279000, 308000, 199000, 219000, 405000, 324000, 319000,
255000]
}

# Create a DataFrame from the sample data
df = pd.DataFrame(data)

# Define the input features (X) and target variable (y)
X = df[['SquareFeet', 'Bedrooms']]
y = df['Price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train a Gradient Boosting Regressor model
model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,
random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print evaluation metrics
print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("R-squared (R2) Score:", r2)
```

```
# Plot the predicted vs. actual prices
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Prices vs. Predicted Prices")
plt.show()
```

XGBoost(Extreme Gradient Boosting) Regressor:

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Generate a sample dataset (you would replace this with your real data)
data = {
    'SquareFeet': [1400, 1600, 1700, 1875, 1100, 1550, 2350, 2450, 1425, 1700],
    'Bedrooms': [3, 3, 3, 3, 2, 2, 4, 4, 3, 3],
    'Price': [245000, 312000, 279000, 308000, 199000, 219000, 405000, 324000, 319
000, 255000]
}

# Create a DataFrame from the sample data
df = pd.DataFrame(data)

# Define the input features (X) and target variable (y)
X = df[['SquareFeet', 'Bedrooms']]
y = df['Price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
tate=42)

# Create and train an XGBoost Regressor model
model = XGBRegressor()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
```

```
# Print evaluation metrics
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)

# Plot the predicted vs. actual prices
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Prices vs. Predicted Prices")
plt.show()
```

```
MAE: 17439.918396832192
MSE: 716579004.5214689
RMSE: 26768.993341578403
R2 Score: 0.9065777666861116
------------------------------
RMSE Cross-Validation: 29698.84961808251
```

CONCLUSION:

Machine learning can be a powerful tool for house price prediction. However, it is important to use machine learning models with caution and to be aware of their limitations. For example, machine learning models are trained on historical data, so they may not be accurate in predicting house prices in a changing market. Additionally, machine learning models can be biased, depending on the data they are trained on.

Overall, machine learning can be a valuable tool for house price prediction, but it should be used in conjunction with other factors, such as expert judgment and market research.

When dealing with complex relationships, outliers, or a large number of features, Linear Regression, Random Forest Regression, Gradient Boosting Regression and XGBoost Regression may perform better.