PREDICTING HOUSE PRICES USING MACHINE LEARNING:

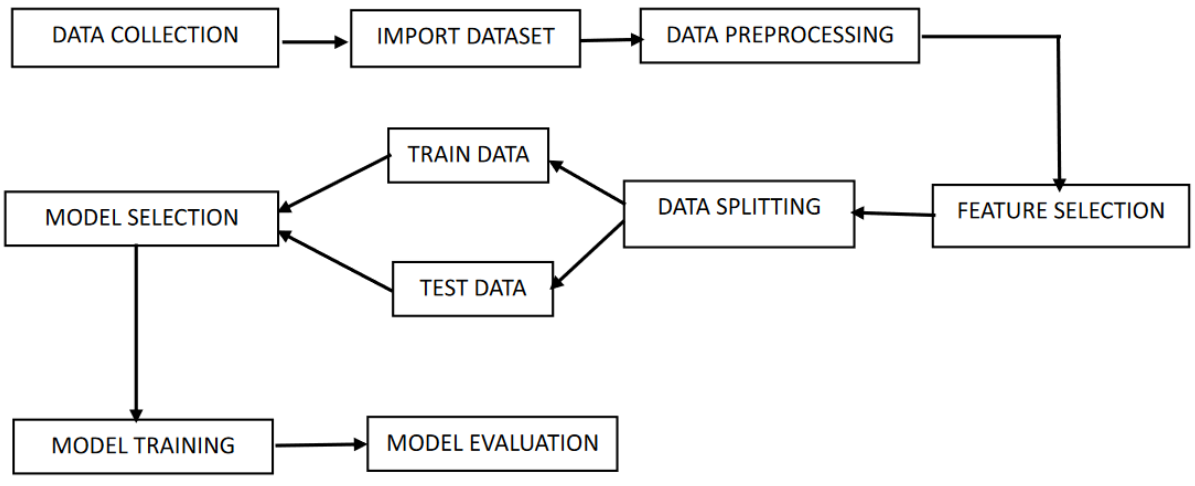
PHASE 5 – PROJECT DOCUMENTATION & SUBMISSION

PROBLEM STATEMENT:

The project's goal is to leverage machine learning techniques for accurately predicting house prices. This is crucial as buying a house represents a significant lifetime investment for many individuals and families. By considering factors like location, square footage, bedrooms, bathrooms, and other relevant variables, the project aims to provide valuable insights for informed decisions in the housing market.

DESIGN THINKING:

Flow diagram:



Data Source Selection:

- ✓ The first step is to import the housing prices dataset downloaded from
- ✓ Kaggle.

- ✓ The dataset contains features such as avg. Area Income, avg. Area House Age, avg. Area Number of Rooms, avg. Area Number of Bedrooms, area Population, price, and address

Data Preprocessing:

- ✓ Data cleaning: Fill missing values, remove duplicates, and correct errors.
- ✓ Feature processing: Select relevant features, encode categorical data, standardize numerical features, and handle outliers.
- ✓ Data splitting and transformation: Divide the data into training, validation, and test sets, address skewed data, and apply necessary transformations for machine learning.

Feature Exploration:

- ✓ Correlation: Keep highly correlated features, discard low correlations.
- ✓ Feature Importance: Prioritize features with high importance scores in tree-based models.
- ✓ Cross-Validation: Opt for the feature subset that yields the best model performance metrics.

Model Selection:

Considering regression algorithms like Linear Regression for a simple and interpretable baseline model. Exploring decision tree-based models like Random Forest and Gradient Boosting for handling non-linear relationships effectively.

Model Training:

- ✓ Train the selected model using the preprocessed data.
- ✓ Train the selected model on the training data using the features to predict house prices.
- ✓ The model will learn the relationships between the features and target variable

Evaluation:

Assess the model's performance using key metrics including Mean Absolute Error (MAE) for average absolute prediction errors, Root Mean Squared Error (RMSE) to penalize larger errors, and R-squared to measure goodness of fit. Lower MAE and RMSE values signify better accuracy, while a higher R-squared indicates a better fit of the model. These metrics help gauge the model's effectiveness in house price prediction.

PHASES OF DEVELOPMENT:

1. Phase 1:

In phase 1, the problem statement and design thinking steps are

clearly defined.

2. Phase 2:

In this phase 2, consider innovative techniques such as ensemble methods and deep learning architectures to enhance the prediction system's accuracy and robustness. Explore advanced regression techniques like Gradient Boosting or XG Boost to improve prediction performance.

3. Phase 3:

In phase 3, the housing prices dataset is loaded and pre-processed.

4. Phase 4:

In phase 4, building the house price prediction model by Feature selection, Model training, Evaluation.

ABOUT THE DATASET USED:

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

- Avg. Area Income: The average income of residents in the area where the property is located.
- Avg. Area House Age: The average age of houses in the area.
- Avg. Area Number of Rooms: The average number of rooms in the houses in the area.
- Avg. Area Number of Bedrooms: The average number of bedrooms in the houses in the area.
- Area Population: The population of the area where the property is located.
- Price: The price of the property.
- Address: The address of the property, including street, city, state, and postal code.

USA Housing Prices

# Avg. Area Income	# Avg. Area House ...	# Avg. Area Numbe...	# Avg. Area Numbe...	# Area Population	# Price
17.8k 108k	2.64 9.52	3.24 10.8	2 6.5	173 69.6k	15.9k
79545.45857431678	5.682861321615587	7.009188142792237	4.09	23086.800502686456	105903
79248.64245482568	6.0028998082752425	6.730821019094919	3.09	40173.07217364482	150589
61287.067178656784	5.865889840310001	8.512727430375099	5.13	36882.15939970458	105898
63345.24004622798	7.1882360945186425	5.586728664827653	3.26	34310.24283090706	126061
59982.197225708034	5.040554523106283	7.839387785120487	4.23	26354.109472103148	630943
80175.7541594853	4.9884077575337145	6.104512439428879	4.04	26748.428424689715	106813

DATA PRE-PROCCESSING AND MODEL TRAINING:

```
#PREPROCESSING DATASET
#1.importing libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
```

Handling missing values:

```
imputer = SimpleImputer(strategy='mean')
data[['Avg. Area Number of Bedrooms']] = imputer.fit_transform(data[['Avg. Area Number of Bedrooms']])
```

Encoding categorical variables:

```
encoder = OneHotEncoder(sparse=False, drop='first')
encoded_address = encoder.fit_transform(data[['Address']])
encoded_df = pd.DataFrame(encoded_address, columns=encoder.get_feature_names_out(['Address']))
data = pd.concat([data, encoded_df], axis=1)
data.drop(['Address'], axis=1, inplace=True)
print(data.head())
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and
  warnings.warn(
   Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
0      79545.458574             5.682861                   7.009188
1      79248.642455             6.002900                   6.730821
2      61287.067179             5.865890                   8.512727
3      63345.240046             7.188236                   5.586729
4      59982.197226             5.040555                   7.839388

   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_000 Todd Pines\nAshleyberg, KY 90207-1179  \
0                                                0.0
1                                                0.0
2                                                0.0
3                                                0.0
4                                                0.0

   Address_001 Steve Plaza\nJessicastad, UT 25190  \
0                                             0.0
1                                             0.0
2                                             0.0
3                                             0.0
4                                             0.0

   Address_0010 Gregory Loaf\nSouth Ericfort, VA 34651-0718  \
0                                                0.0
1                                                0.0
2                                                0.0
3                                                0.0
4                                                0.0
```

Separate Features and Target Variable:

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

Split the Dataset:

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

standardize feature value:

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
#MODEL TRAINING:
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
```

Create a linear regression model:

```
# Creating a linear regression model
model = LinearRegression()
```

Fitting the model on the training data:

```
# Fit the model on the training data
model.fit(X_train_selected, y_train)
```

Making predictions on the test data:

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

REGRESSION ALGORITHM:

Random Forest Regression:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Random Forest Regressor
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)

# Make predictions
y_pred = rf_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 13614968426.227232
```

XG-Boost Regressor:

```
import xgboost as xgb
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error

# XGBoost Regressor
xgb_regressor = XGBRegressor(n_estimators=100, random_state=42)
xgb_regressor.fit(X_train, y_train)

# Make predictions
y_pred = xgb_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 13799788036.573933
```

EVALUATION METRICS:

```
#EVALUATION:
from sklearn.metrics import mean_squared_error, r2_score

# Calculating Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

#calculating root mean Squared Error
rmse = np.sqrt(mse)

# Calculating R-squared (R2) Score
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared (R2) Score:", r2)
```

```
Mean Squared Error (MSE): 10089009300.894518
Root Mean Squared Error: 100444.06055558745
R-squared (R2) Score: 0.9179971706834289
```

ASSIGNMENT NOTEBOOK:

- The entire project is developed in the Google Colab Notebook.

- All the code files, including the data preprocessing, model training and evaluation steps are compiled in prior.

COLAB NOTEBOOK lINK:
https://colab.research.google.com/drive/1pUQLaYgPhDibbDu7v9exr3wOrX4a_gsM?authuser=1

CONCLUSION:

In this project, we have done effective data preprocessing and feature selection techniques significantly improved the model's accuracy in predicting house prices.

README FILE:

House Price Prediction README

This project aims to predict house prices using machine learning techniques. Below are the steps to run the code successfully:

Steps:

1.Clone the Repository:

git clone https://github.com/your_username/house-price-prediction.git

2.Install Dependencies:

We need Python 3.x and the following Python libraries:

pip install pandas numpy scikit-learn matplotlib seaborn

3.Dataset:

Place your dataset file in the root folder with the name dataset.csv.csv.

4.Run the Code:

Open a terminal or command prompt and navigate to the project directory, then run:

python house_price_prediction.py

5.Results:

The code will generate a prediction model for house prices and display evaluation metrics.

Dependencies:

- Pandas
- Numpy
- scikit-learn
- matplotlib
- seaborn