HOUSE PRICE PREDICTION WITH MACHINE LEARNING

Name: Sai Deepthi Malugu

Student Id: 23070109

GitHub Link: https://github.com/saideepthimalugu/machine-learning.git

INTRODUCTION: Forecasting house prices is a significant application of machine learning in real estate. By examining various factors like house age, median earnings, and geographical location, we can set up a model to evaluate housing prices correctly. In this prediction, we are using the California housing dataset and applying a Random Forest Regressor to predict housing prices.

DATASET OVERVIEW:

We make use of the California housing dataset which covers the following attributes:

- House age: average age of houses
- Average income of households in the area
- Average bedrooms
- Average rooms
- Total population in the area
- Average occupants per house
- Geographical latitude
- Geographical longitude

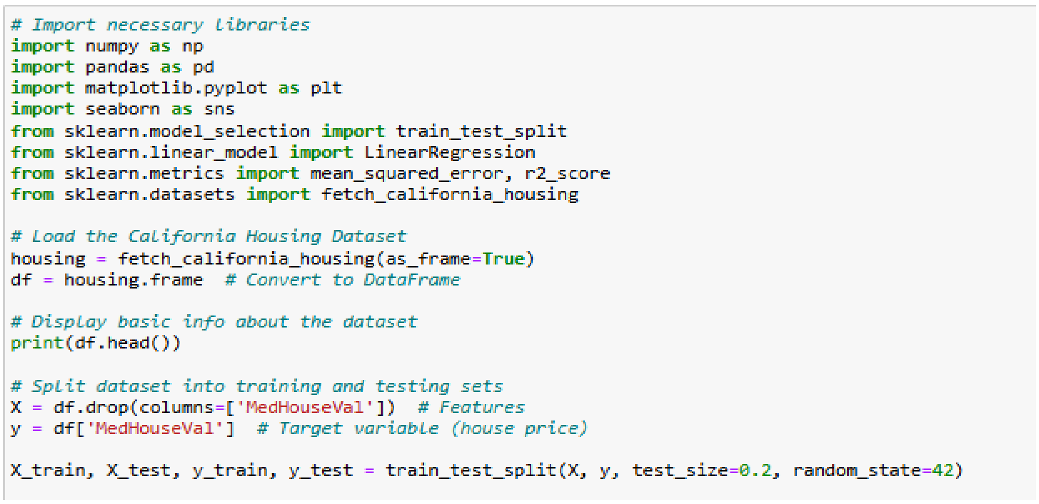
Why this Dataset?

This dataset is popularly used for Regression tasks and allows real-world housing price data that lend a hand to recognise how several factors impact house prices.

DATA PREPROCESSING:

Before we train our model we have to perform data cleaning such as handling missing values, dealing with duplicates, and feature scaling which is standardized by the numeric attributes to refine model performance. Splitting the dataset into input features(X) and target variable (y) followed by splitting the data into training sets (80%) and testing (20%)

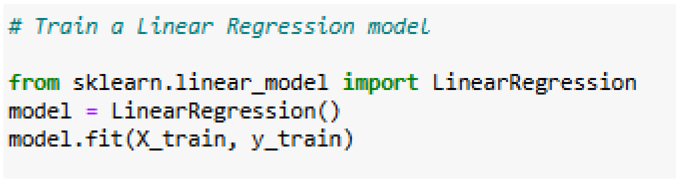
DATASET LOADING:



```python
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.datasets import fetch_california_housing

# Load the California Housing Dataset
housing = fetch_california_housing(as_frame=True)
df = housing.frame  # Convert to DataFrame

# Display basic info about the dataset
print(df.head())

# Split dataset into training and testing sets
X = df.drop(columns=['MedHouseVal'])  # Features
y = df['MedHouseVal']  # Target variable (house price)

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

MODEL TRAINING:

By using the dataset a linear regression model is trained. The model acquires a knowledge of the relationship between the independent variables(attributes) and target variables (house price).

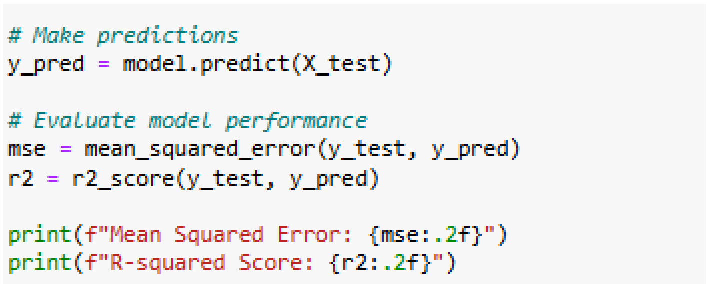


```python
# Train a Linear Regression model

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

MODEL EVALUATION:

The model’s performance was assessed using:

- Mean Squared Error (MSE): This measures the average squared distinctness between actual and predicted values.
- R² Score: Regulate how ably the model describes the variability in house prices.



```python
# Make predictions
y_pred = model.predict(X_test)

# Evaluate model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared Score: {r2:.2f}")
```

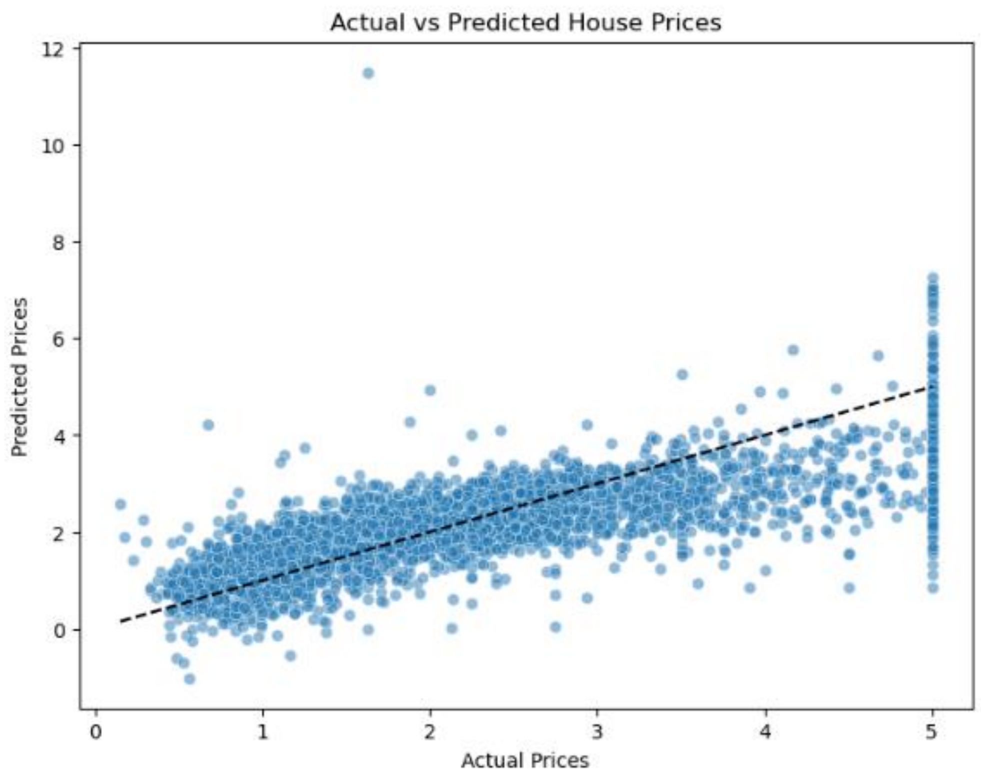
RESULTS:

- **Linear Regression MSE: 0.26**

- **Linear Regression R^2 Score: 0.81**

These values specify that the model describes around 81% of the variance in house prices, making it a suitable fit.

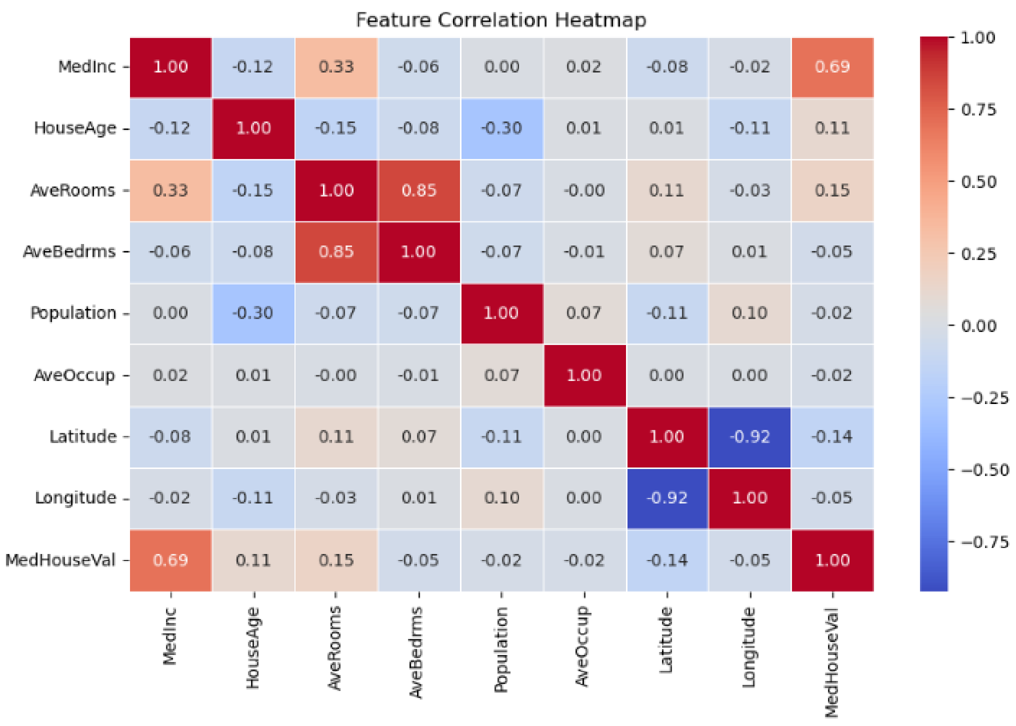
VISUALIZATION: We used two visualizations heat map and a scatter plot to analyze the data and model performance.



Scatter plot:

- If the spots in the scatter plot are closely arranged in line with the black diagonal line, it signifies the mode is creating exact predictions.
- If the spots are scattered a long away from the diagonal line, it specifies that the model has a big error and has difficulty predicting exact house prices.
- We can observe some outliers are present, where the model overestimates house prices notably.

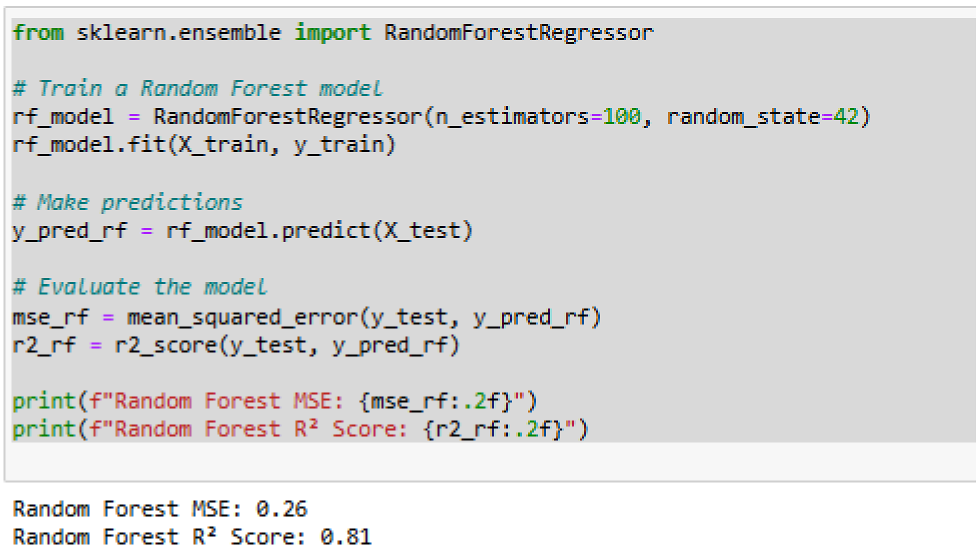
Heat map:



From the above heat map, we can observe some key relationships between attributes. The strongest correlation with (y) target variable median(average) house value is with median income, this shows an important positive correlation. This indicates that higher income levels are liable to lead to high house prices. The other attribute is average rooms (AveRooms) and average occupancy (Aveoccup) have visible some positive correlation with house prices but not as strong as median income (MedInc) notably, House age has a weaker correlation with house prices, shows that age of households possibly not outstandingly work on the house price. The heat map also tells some inter-feature correlations, high correlation between average bedrooms and average rooms they both are liable to increase together which consists as big houses normally have more rooms and more bedrooms. Mainly this heat map helps to recognize the relationship between different attributes and house prices. Through this analysis, we can make decisions during feature selection and model training to make better our predictions.

ADVANCED TECHNIQUES:

To get accurate prediction also used advanced techniques of Random Forest Regression and Hyperparameter Tuning, Random forests normally exceed single decision trees and other basic models by reducing overfitting and increasing generalization.

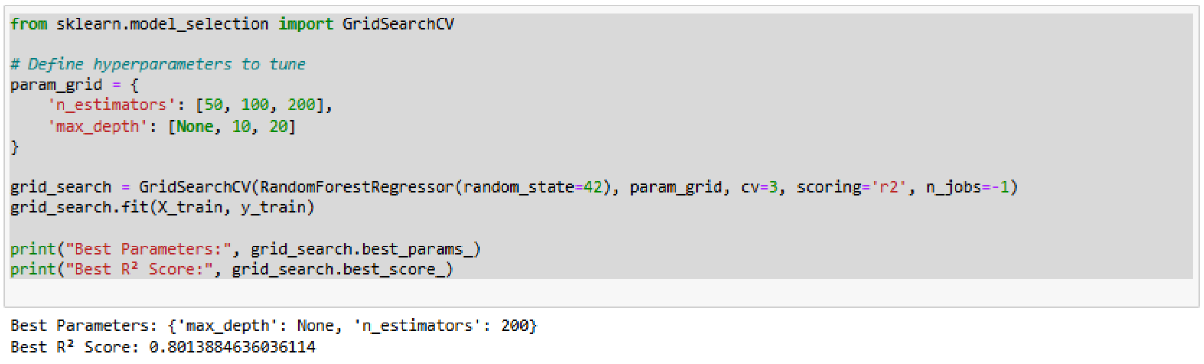


```
from sklearn.ensemble import RandomForestRegressor

# Train a Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions
y_pred_rf = rf_model.predict(X_test)

# Evaluate the model
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print(f"Random Forest MSE: {mse_rf:.2f}")
print(f"Random Forest R² Score: {r2_rf:.2f}")
```

```
Random Forest MSE: 0.26
Random Forest R² Score: 0.81
```

To improve the random forest model, I execute the hyperparameter tuning using GridsearchCV. This technique allows to search through multiple combinations of hyperparameters, the hyperparameter tune is:

- **n_estimators**: The number of trees in the forest (more trees typically lead to better performance, but also more calculation cost).
- **max_depth**: The maximum depth of each tree. restrict the depth can help to stop overfitting.



```
from sklearn.model_selection import GridSearchCV

# Define hyperparameters to tune
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20]
}

grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_grid, cv=3, scoring='r2', n_jobs=-1)
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best R² Score:", grid_search.best_score_)
```

```
Best Parameters: {'max_depth': None, 'n_estimators': 200}
Best R² Score: 0.8013884636036114
```

Conclusion:

This analysis provides an introduction to house pricing prediction using linear regression, it offsets the dataset analysis, preprocessing, model training, model evaluation and visualization. The predictions reveal that linear regression is a simple yet successful method for this analysis. Although we also made with advanced techniques.

REFERENCES:

- **Chen, Y., & Zhang, J. (2014). Machine learning in real estate**

- **James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). A book Named Introducation to statistical this book gives more information on Linera regression and other advanced techniques, including model evaluation.**
- **Dataset from kaagle website (kaagle house price prediction dataset)**