NICMAR UNIVERSITY

NICMAR BUSINESS SCHOOL

MBA GENERAL

AI, ML, AND KNOWLEDGE BASE

(ICT 621)

DR. PRIYA DESHPANDE

MACHINE LEARNING MODEL ASSIGNMENT

"Housing Price Prediction"

Prepared By: –

Sandeep Sharma (P2373022)

Semester – 3rd

NICMAR University, Pune

2023-2025

Introduction

Housing price prediction is a significant area of study in the fields of data science and machine learning, given the complex factors that influence real estate markets. Accurate prediction of housing prices helps buyers, sellers, real estate agents, and policy makers make informed decisions. In this project, we will use a dataset of California housing prices, which contains key features such as location, population, median income, and housing characteristics, to develop a predictive model. By leveraging machine learning algorithms in Python, we aim to build a model that can forecast housing prices with high accuracy.

The dataset used in this project is the California Housing Dataset, which contains information from the 1990 U.S. Census. It includes demographic and geographical data, which are crucial to predicting housing prices in various regions of California. This dataset has been widely used in machine learning tutorials, making it a valuable resource for understanding real-world housing prediction tasks.

The goal of this project is to explore various machine learning techniques, preprocess the data, and evaluate the performance of different models to predict housing prices based on the available features.

Problem Statement

In the real estate market, accurately predicting housing prices is a key challenge that influences various stakeholders, including homebuyers, real estate agents, investors, and policymakers. Predicting housing prices requires an understanding of the relationship between various factors such as income levels, geographical location, demographic characteristics, and housing features. This project focuses on building a predictive model to forecast the median house values in California based on a variety of attributes such as the median income, the age of the houses, the number of rooms, and the population of each district.
The dataset used for this project, known as the California Housing Dataset, contains information from the 1990 U.S. Census and includes demographic and geographical features for different districts in California. The challenge is to build a model that can predict the median house value for each district, given its various features. This problem is framed as a regression task, where the goal is to predict a continuous value—the median house value—based on input features.

AIM & OBJECTIVE

- People looking to buy a new home tend to be more conservative with their budgets and market strategies.
- This project aims to analyse various parameters like average income, average area etc. and predict the house price accordingly.
- This application will help customers to invest in an estate without approaching an agent.
- To provide a better and fast way of performing operations.
- To provide proper house price to the customers.
- To eliminate need of real estate agent to gain information regarding house prices.
- To provide best price to user without getting cheated.
- House prices increase every year, so there is a need for a system to predict house prices in the future.
- House price prediction can help the developer determine the selling price of a
- house and can help the customer to arrange the right time to purchase a house.
- We use linear regression algorithm in machine learning for predicting the house price trends.

ABOUT DATASET

This dataset is used in the second chapter of Aurélien Géron's recent book 'Hands-On Machine Learning with Scikit-Learn and TensorFlow'. It contains information from the 1990 California census and serves as an excellent introduction to implementing machine learning algorithms. The dataset includes various features related to housing and demographics within a block in California.

- Longitude: A measure of how far west a house is; a higher value is farther west
- Latitude: A measure of how far north a house is; a higher value is farther north
- Housing Median Age: Median age of a house within a block; a lower number is a newer building
- Total Rooms: Total number of rooms within a block
- Total Bedrooms: Total number of bedrooms within a block
- Population: Total number of people residing within a block
- Households: Total number of households, a group of people residing within a home unit, for a block
- Median Income: Median income for households within a block of houses (measured in tens of thousands of US Dollars)
- Median House Value: Median house value for households within a block (measured in US Dollars)
- Ocean Proximity: Location of the house w.r.t ocean/sea.

Data Info: -

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):

Link - https://drive.google.com/drive/folders/1c8eYOHtHzVKKOlvAeu1ujLmmAkxi-b3S?usp=drive_link

Methodology

The methodology for the housing price prediction project follows a structured approach, from data exploration to model evaluation. Here's a concise overview:

Data Collection: - The project uses the California Housing Dataset, which includes features like median income, total rooms, population, and geographical information of districts in California. The target variable is the median house value for each district.

Data Preprocessing

Exploratory Data Analysis (EDA) is performed to understand the dataset and check for missing values or inconsistencies.

Feature Scaling is applied using Standard Scaler to normalize the data, especially for models like Linear Regression.
The dataset is split into training (80%) and testing (20%) sets to ensure the model is trained on one subset and evaluated on another.

Model Selection

Linear Regression is initially used as a baseline model to predict house prices based on the features.
A more advanced model, Random Forest Regressor, is also employed to handle non-linear relationships and capture more complex patterns in the data.

Model Training

Both models are trained on the scaled training data. The Random Forest Regressor is further fine-tuned using GridSearchCV to optimize hyperparameters like the number of trees and maximum depth.

Model Evaluation

Model performance is evaluated using metrics like Mean Squared Error (MSE)**,** Root Mean Squared Error (RMSE)**,** and R-Squared (R^2**)**. The goal is to measure the model's accuracy in predicting house prices.

Results Visualization

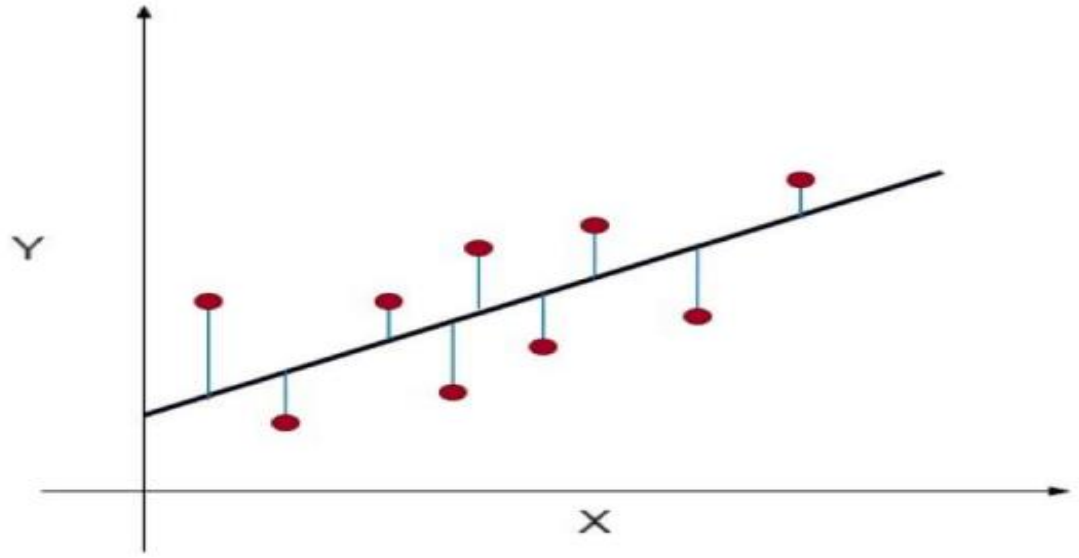
A plot of Actual vs. Predicted house values is created to visually assess the model's prediction accuracy.

Linear Regression

Linear Regression is a supervised machine learning model that attempts to model a linear relationship between dependent variables (Y) and independent variables (X). Every evaluated observation witha model, the target (Y)'s actual value is compared to the target (Y)'s predicted value, and the major differences in these values are called residuals. The Linear Regression model aims to minimize the sum of all squared residuals. Here is the mathematical representation of thelinear regression:

$$Y= a0+a1X+ \varepsilon$$

The values of X and Y variables are training datasets for the model representation of linear regression. When a user implements a linear regression, algorithms start to find the best fit line using a0 and a1. In such a way, it becomes more accurate to actual data points; since we recognize the value of a0 and a1, we can use a model for predicting. The values of X and Y variables are training datasets for the model representation of linear regression. When a user implements a linear regression, algorithms start to find the best fit line using a0 and a1. In such a way, it becomes more accurate to actual data points; since we recognize the value of a0 and a1, we can use a model for predicting the response.

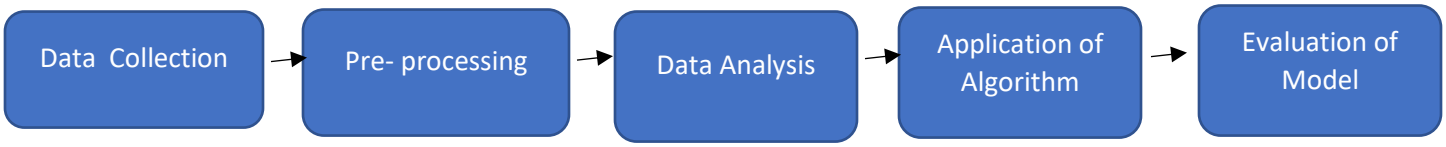


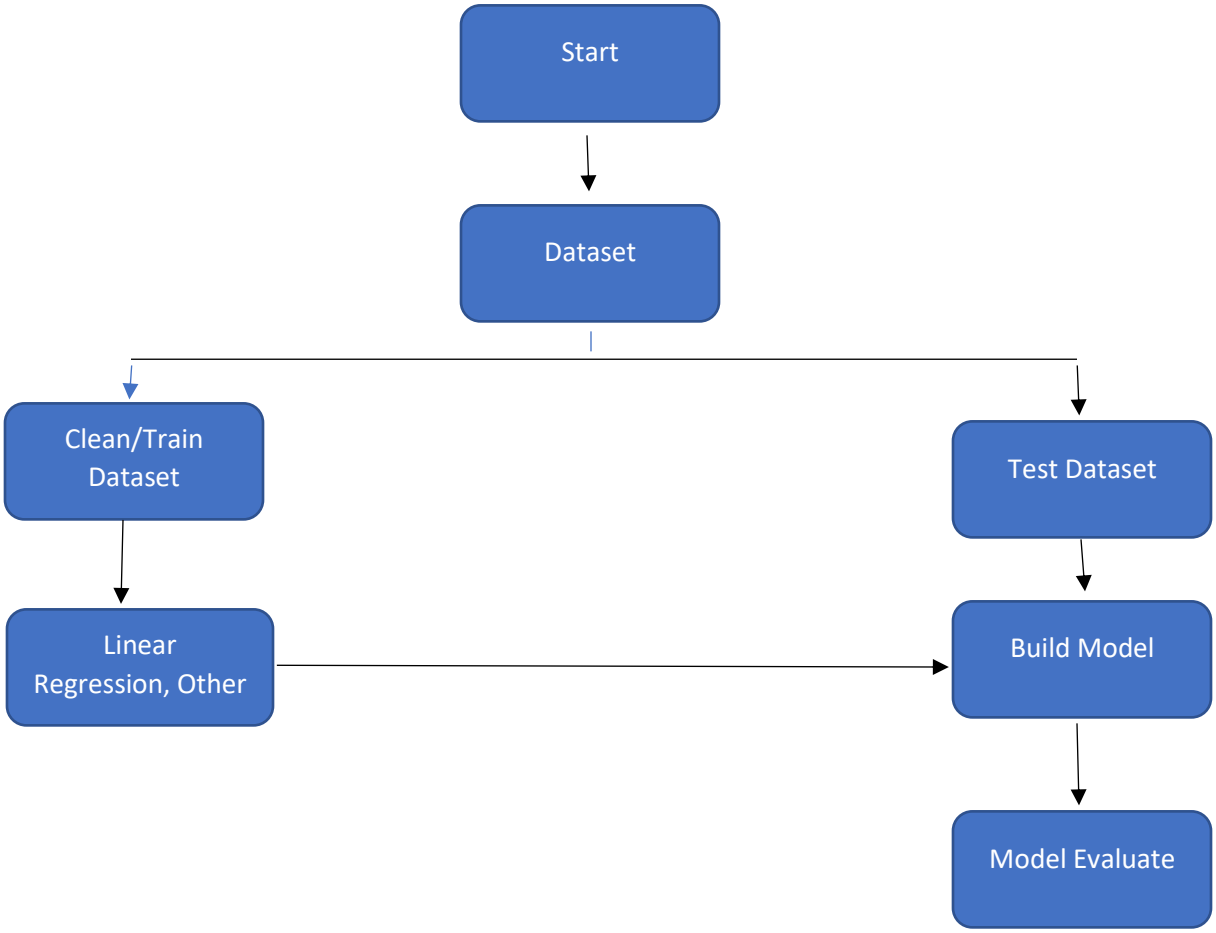
Source: - Google

As you can see in the above diagram, the red dots are observed values for both X and Y.

- The black line, which is called a line of best fit, minimizes a sum of a squared error.
- The blue lines represent the errors; it is a distance between the line of best fit and observed values.
- The value of the a1is the slope of the black line.

BLOCK DIAGRAM

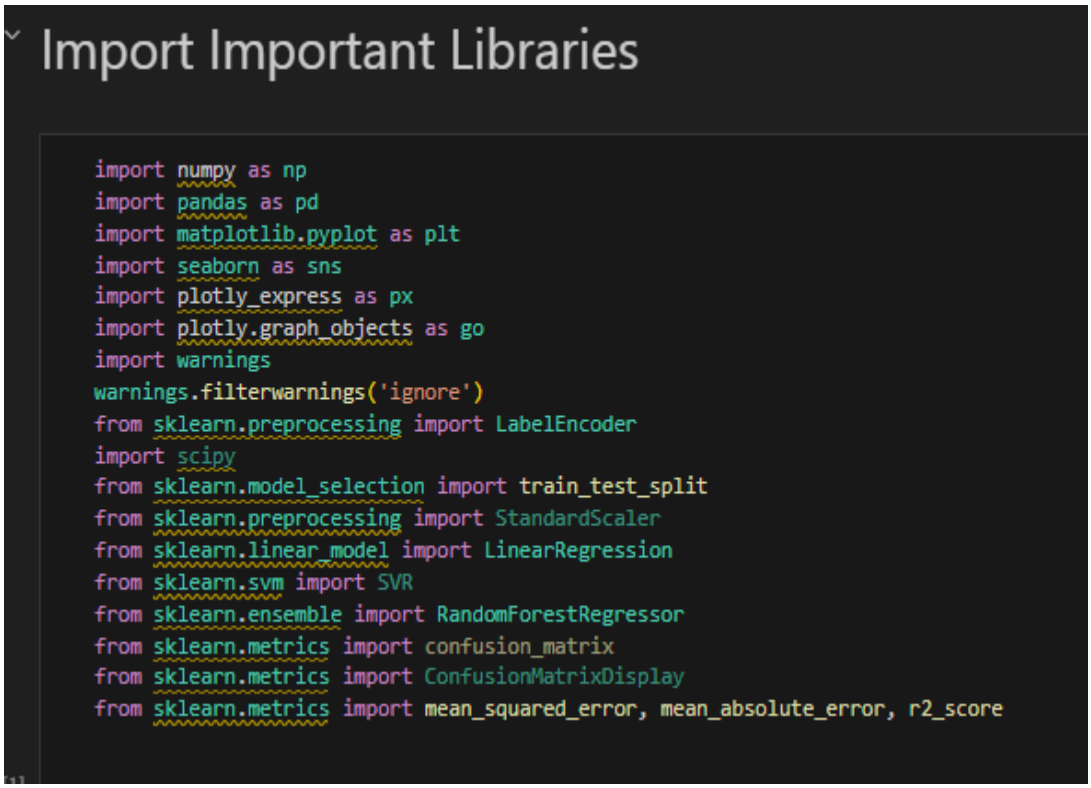




Technical Use

- Programming Language: Python
- Data Manipulation and Analysis: Pandas, NumPy
- Python Visualization Library: Matplotlib, Seaborn (optional)
- Machine Learning Framework: Scikit-learn
- Modeling Algorithms: Linear Regression, Random Forest Regressor
- Model Evaluation: MSE, RMSE, R^2

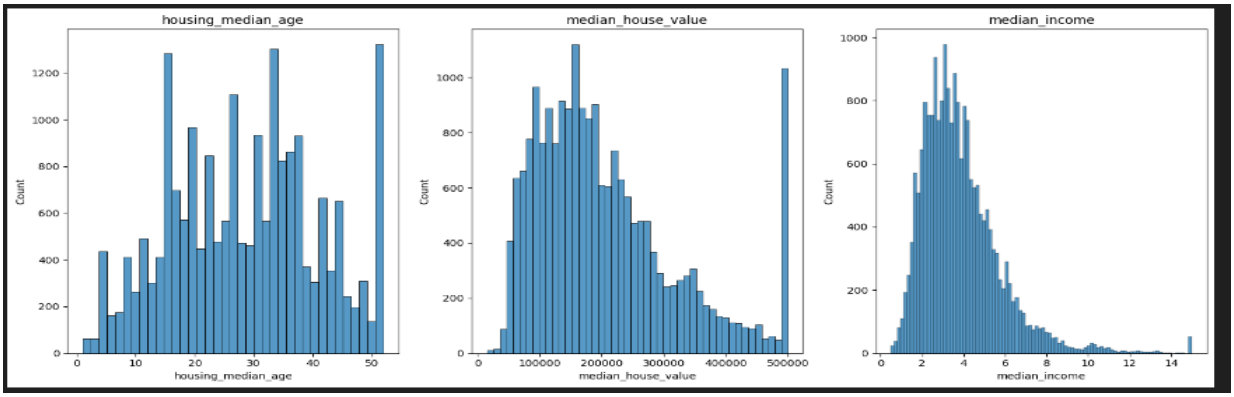
Python Code Glance



Explore data

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```



Data Cleaning

```
data[data['total_bedrooms'].isna()]
```

```
data['total_bedrooms'].fillna(data['total_bedrooms'].mean(),inplace=True)
```

```
d_null(data)
```

```
Series([], dtype: int64)
```

Split Data Target & Features

```
x=data.drop('median_house_value',axis=1)
y=data['median_house_value']
```

Split Train & Test data

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

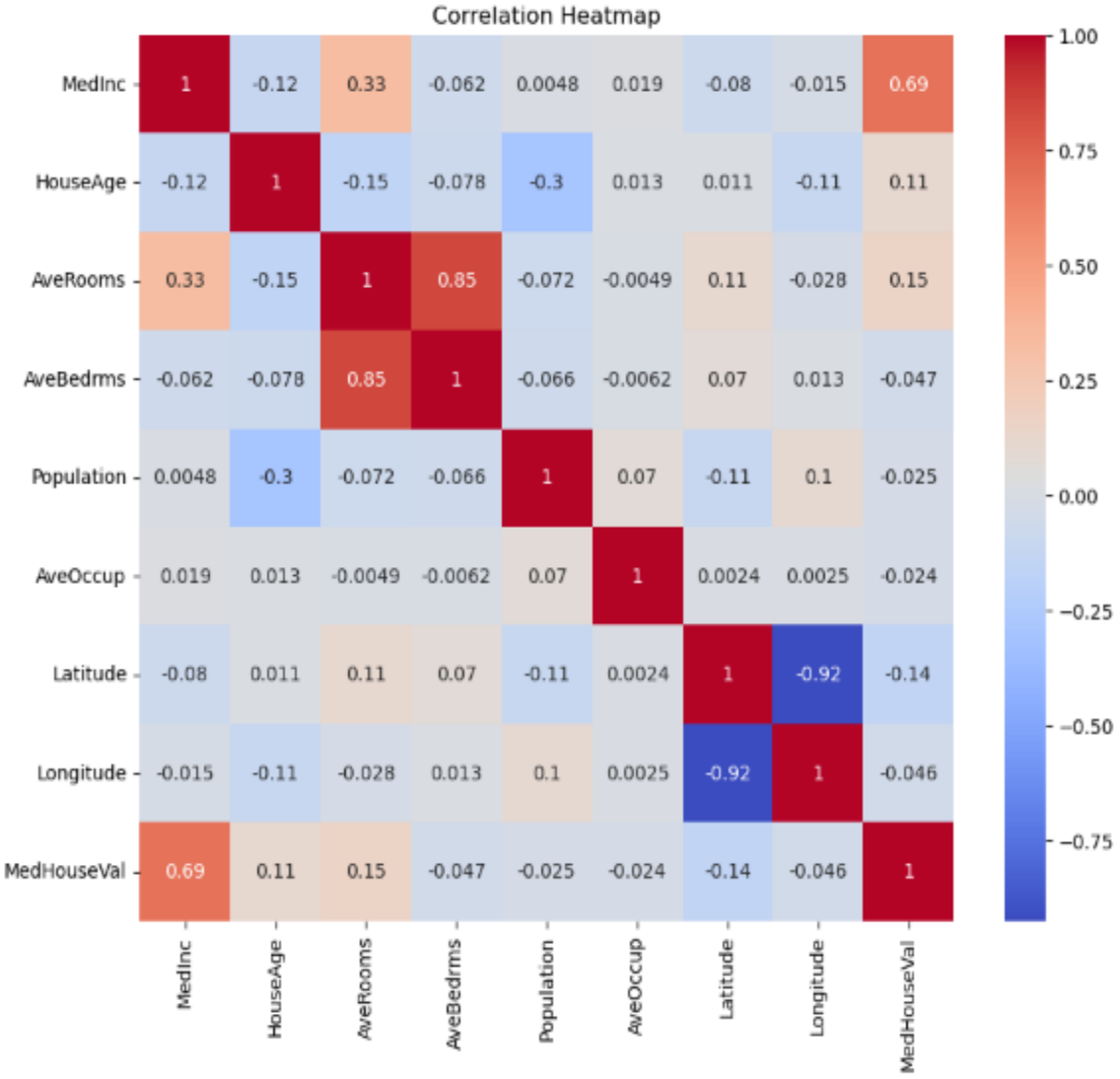
```
((16512, 9), (4128, 9), (16512,), (4128,))
```

Build Model

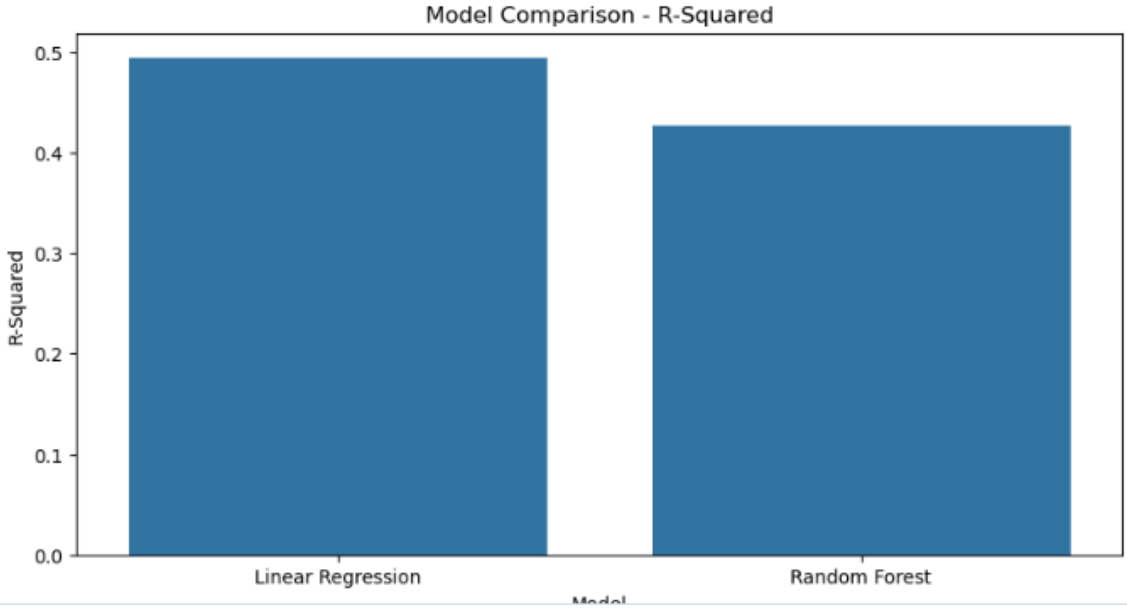
```
lin_reg=LinearRegression()
rf_reg=RandomForestRegressor()
```

```
columns=['LinearRegression','RandomForestRegressor']
train_score=[]
rec_score=[]
mae_val=[]
mse_val=[]
```

```
def all(model):
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    accuracy_train=model.score(x_train,y_train)*100
    recall_result=r2_score(y_pred,y_test)*100
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    train_score.append(accuracy_train)
    rec_score.append(recall_result)
    mse_val.append(mse)
    mae_val.append(mae)
```



```
Comparison of Models:
               Model  R-Squared  Mean Squared Error
0  Linear Regression   0.494061        6.629874e+09
1      Random Forest   0.427052        7.507956e+09
```



[160]:
```
linear_model = LinearRegression()
rf_model = RandomForestRegressor(random_state=42)
linear_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)
```

[160]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

2]:
```
y_pred_linear = linear_model.predict(X_test)
mse_linear = mean_squared_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)
y_pred_rf = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
```

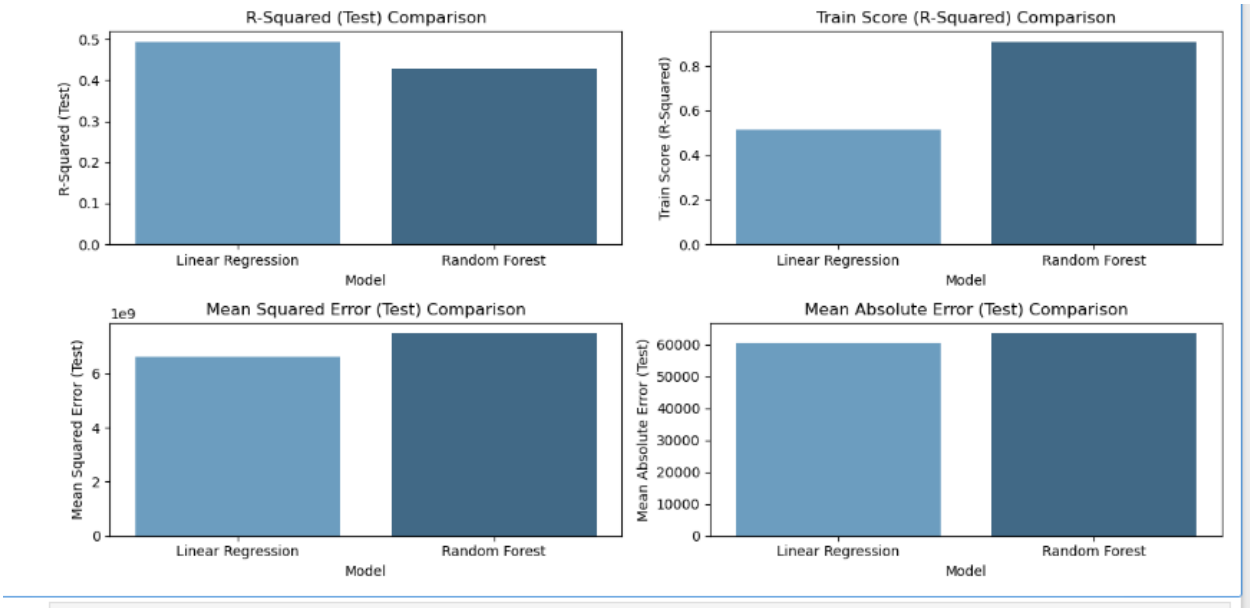
3]:
```
print("Model Testing Results (20% test data):")
```

```
Model Testing Results (20% test data):
```

4]:
```
print("Linear Regression - MSE:", mse_linear, ", R^2:", r2_linear)
print("Random Forest - MSE:", mse_rf, ", R^2:", r2_rf)
```

```
Linear Regression - MSE: 6629874283.048178 , R^2: 0.4940606792889837
Random Forest - MSE: 7507955792.177114 , R^2: 0.427052476223428
```

Comparison with different method




```
y_pred_rf_train = rf_model.predict(X_train)
y_pred_rf_test = rf_model.predict(X_test)
r2_rf_train = r2_score(y_train, y_pred_rf_train)
r2_rf_test = r2_score(y_test, y_pred_rf_test)
mse_rf = mean_squared_error(y_test, y_pred_rf_test)
mae_rf = mean_absolute_error(y_test, y_pred_rf_test)
results = pd.DataFrame({
    "Model": ["Linear Regression", "Random Forest"],
    "R-Squared (Test)": [r2_linear_test, r2_rf_test],
    "Train Score (R-Squared)": [r2_linear_train, r2_rf_train],
    "Mean Squared Error (Test)": [mse_linear, mse_rf],
    "Mean Absolute Error (Test)": [mae_linear, mae_rf]
})

# Display the table
print("Model Evaluation Metrics Table:")
print(results)
```

```
Model Evaluation Metrics Table:
               Model  R-Squared (Test)  Train Score (R-Squared)  \
0  Linear Regression          0.494061                 0.512779
1      Random Forest          0.427052                 0.909559

   Mean Squared Error (Test)  Mean Absolute Error (Test)
0               6.629874e+09                60597.800519
1               7.507956e+09                63646.556471
```

Random Forest Regressor vs. Linear Regression

Random Forest Regressor Outperforms Linear Regression:

- Train Score: Random Forest Regressor performs better
- Recall Score: Random Forest Regressor performs better
- Mean Squared Error (MSE): Random Forest Regressor performs better (lower MSE is preferred)
- Mean Absolute Error (MAE): Random Forest Regressor performs better (lower MAE is preferred)

Outcome of the Housing Price Prediction Project

The outcome of the project revolves around building an effective predictive model that can estimate the median house values in California based on various features such as median income**,** number of rooms**,** population, and geographical location. Below are the key outcomes:

Linear Regression achieved better performance on the test set compared to Random Forest **Regression** in terms of both R2 and error metrics (MSE and MAE).

- **R-Squared (Test)**: Linear Regression achieved an R2 of **0.494**, indicating it explained around 49.4% of the variance in housing prices in the test data.
- **Train-Test Consistency**: Linear Regression showed consistency between training and test scores, with a training R2 of **0.513**, indicating minimal overfitting.

Random Forest Regression showed signs of overfitting, with a significantly higher training R2 (0.910) compared to its test R2 (0.427).

- **Error Metrics**: Random Forest had a higher Mean Squared Error (MSE) and Mean Absolute Error (MAE) on the test set than Linear Regression, suggesting it had larger prediction errors.

Conclusion

- The **Linear Regression** model demonstrated a more reliable performance on the test set, providing a good balance between simplicity and predictive accuracy. With an R2R^2R2 of **0.494**, it explained nearly 49.4% of the variance in housing prices.
- **Random Forest**, despite its higher training score, appeared to be overfitting to the training data, as evidenced by its lower test score and higher error metrics. With further hyperparameter tuning or the use of regularization techniques, it could potentially improve.
- Overall, **Linear Regression** emerged as the better model in this context, offering a simpler and more consistent approach to predicting housing prices.