**House price prediction**

**1. Understand the Data:**

- Obtain a dataset with features such as the area, number of bedrooms, location, age of the property, and other relevant information.

- Understand the distribution of data, identify missing values, and handle outliers if necessary.

2. **Feature Selection Techniques:**

- **Univariate Selection:**

Select the features with the strongest relationship with the target variable. For instance, you can use statistical tests like ANOVA F-test or chi-squared test.

**- Feature Importance:**

Use techniques like decision trees or random forests to determine the importance of each feature in predicting the target variable.

- **Correlation Matrix with Heatmap:**

Analyze the correlation between features to identify redundant or highly correlated variables.

- **Recursive Feature Elimination:**

Use models that recursively remove the least significant features, based on the coefficients, until the specified number of features is reached.

**- L1 Regularization:**

Apply L1 regularization techniques like Lasso regression to automatically select the relevant features while penalizing the irrelevant ones.

3. **Data Preprocessing:**

- Handle missing values using techniques like mean or median imputation, or use advanced techniques like KNN imputation.

- Standardize or normalize the data to bring all the features to a common scale.

4. **Model Building:**

- Choose an appropriate regression algorithm like Linear Regression, Ridge Regression, Lasso Regression, Decision Trees, Random Forests, or Gradient Boosting Machines.

- Split the dataset into training and testing sets.

- Train the model on the training data and evaluate its performance on the testing data.

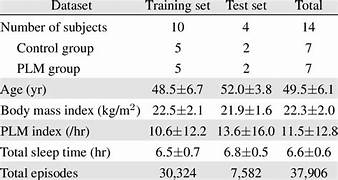
5. **Model Evaluation:**

- Use metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared to evaluate the model's performance.

- Perform cross-validation to ensure the model's robustness.

6. **Iterate and Refine:**

- Based on the evaluation results, fine-tune the feature selection process or try different models to achieve the best possible prediction accuracy.



* **Coding for feature selection**

# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.feature\_selection import SelectKBest, f\_regression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error, r2\_score

# Load the dataset

data = pd.read\_csv('house\_data.csv') # replace 'house\_data.csv' with your dataset

# Perform data preprocessing

# Handle missing values

data = data.dropna()

# Define the features and target

X = data.drop('Price', axis=1)

y = data['Price']

# Feature selection using SelectKBest

selector = SelectKBest(score\_func=f\_regression, k=5) # select the top 5 features

X\_new = selector.fit\_transform(X, y)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_new, y, test\_size=0.2, random\_state=42)

# Standardize the data

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Train the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Model evaluation

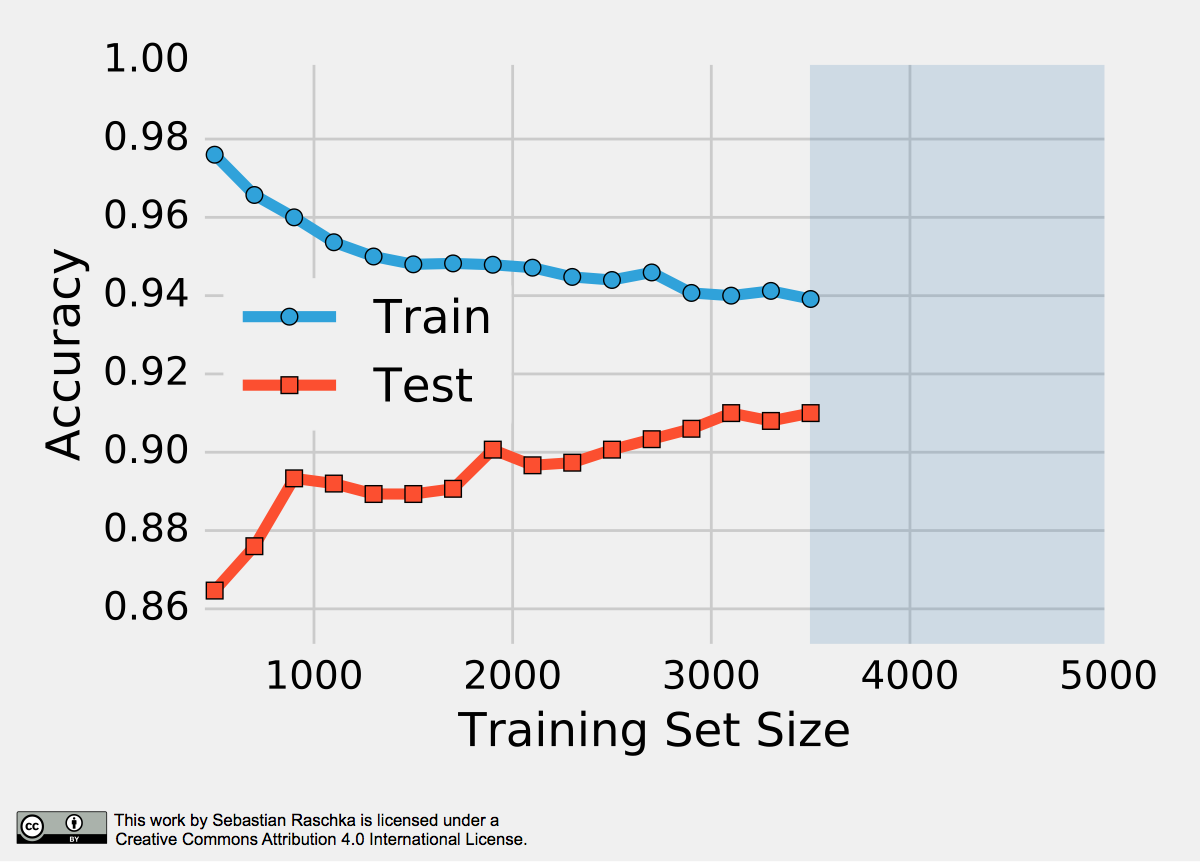
mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

# Print the evaluation metrics

print('Mean Squared Error:', mse)

print('R-squared:', r2)



Coding for Model Training:

# Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error, r2\_score

# Load the dataset

data = pd.read\_csv('house\_data.csv') # Replace 'house\_data.csv' with your dataset

# Perform data preprocessing

# Handle missing values

data = data.dropna()

# Define the features and target

X = data.drop('Price', axis=1)

y = data['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)

# Standardize the data

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Train the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Model evaluation

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

# Print the evaluation metrics

print('Mean Squared Error:', mse)

print('R-squared:', r2)

