```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import StandardScaler
```

Use pandas library to read our data from the California_Houses.csv file

```python
df = pd.read_csv(r'D:\term 7\machine learning\ML- Assignment 1\ML-
Assignment 1\California_Houses.csv')
df.head()
```

```
   Median_House_Value  Median_Income  Median_Age  Tot_Rooms
Tot_Bedrooms  \
0            452600.0         8.3252          41        880
129
1            358500.0         8.3014          21       7099
1106
2            352100.0         7.2574          52       1467
190
3            341300.0         5.6431          52       1274
235
4            342200.0         3.8462          52       1627
280

   Population  Households  Latitude  Longitude  Distance_to_coast  \
0         322         126     37.88    -122.23        9263.040773
1        2401        1138     37.86    -122.22       10225.733072
2         496         177     37.85    -122.24        8259.085109
3         558         219     37.85    -122.25        7768.086571
4         565         259     37.85    -122.25        7768.086571

   Distance_to_LA  Distance_to_SanDiego  Distance_to_SanJose  \
0   556529.158342         735501.806984         67432.517001
1   554279.850069         733236.884360         65049.908574
2   554610.717069         733525.682937         64867.289833
3   555194.266086         734095.290744         65287.138412
4   555194.266086         734095.290744         65287.138412

   Distance_to_SanFrancisco
0              21250.213767
1              20880.600400
2              18811.487450
3              18031.047568
4              18031.047568
```

next cell split data to target that want to be predicted and feature that will be used to predict the target

```
target = df["Median_House_Value"]
feature =
df[["Median_Income","Median_Age","Tot_Rooms","Tot_Bedrooms","Population","Households","Latitude","Longitude","Distance_to_coast","Distance_to_LA","Distance_to_SanDiego","Distance_to_SanJose","Distance_to_SanFrancisco"]]
```

Split the data into Train, Test, and Validation sets This train_test_split data 70% train data and 30% temp data that will be split This split temp data to two equal parts part for val and other for testing

```
x_train,x_test_temp,y_train,y_test_temp = train_test_split(feature,
target, test_size= 0.3, random_state= 42)
x_test,x_val,y_test,y_val = train_test_split(x_test_temp, y_test_temp,
test_size=0.5, random_state=42)

scaler = StandardScaler()
x_train_norm = scaler.fit_transform(x_train)
x_test_norm = scaler.transform(x_test)
x_val_norm = scaler.transform(x_val)
```

StandardScaler is an object that will normalize the features by removing the mean and scaling each feature to unit variance Normalization or standardization is done to bring all features onto a similar scale

```
#1-linear regression

linear = LinearRegression()
linear.fit(x_train_norm, y_train)
linear_pred_val = linear.predict(x_val_norm)

linear_pred_test = linear.predict(x_test_norm)

#calculate Mean Square Error and Mean Absolute Error for validate data
mse_linear_val = mean_squared_error(y_val, linear_pred_val)
mae_linear_val = mean_absolute_error(y_val, linear_pred_val)

#calculate Mean Square Error and Mean Absolute Error
mse_linear_test = mean_squared_error(y_test, linear_pred_test)
mae_linear_test = mean_absolute_error(y_test, linear_pred_test)
print(f"Linear Regression")
print("Mean Squared error and Mean Absolute Error using validate
data:")
print(f"Mean Squared Error: {mse_linear_val:.4f}\nMean Absolute Error:
{mae_linear_val:.4f}")
print("Mean Squared error and Mean Absolute Error using test data:")
print(f"Mean Squared Error: {mse_linear_test:.4f}\nMean Absolute
Error: {mae_linear_test:.4f}")
```

```
Linear Regression
Mean Squared error and Mean Absolute Error using validate data:
Mean Squared Error: 4400953150.6137
Mean Absolute Error: 48782.0311
Mean Squared error and Mean Absolute Error using test data:
Mean Squared Error: 4907211997.3748
Mean Absolute Error: 50790.0603
```

Here, we're creating an instance of the LinearRegression model from scikit-learn Then We're fitting the linear model to the training data After the model is trained, it is used to make predictions on the validation data and test It generates predicted values for the median house prices based on the features Mean Squared Error (MSE) and Mean Absolute Error (MAE) are calculated to evaluate the model's performance

```
#2-lasso regression

# Define a range of alpha values to try
alpha_values = [0.1, 1, 50, 45]
best_alpha = None
best_mse_val = float('inf')
best_model = None

# Loop through each alpha value
for alpha in alpha_values:
    # Create a Lasso model with the current alpha value and increased
max_iter
    lasso = Lasso(alpha=alpha, max_iter=5000)
    lasso.fit(x_train_norm, y_train)

    # Predict on the validation set
    lasso_pred_val = lasso.predict(x_val_norm)

    # Calculate the Mean Squared Error on the validation set
    mse_val = mean_squared_error(y_val, lasso_pred_val)

    # Update best alpha if this one has the lowest validation MSE
    if mse_val < best_mse_val:
        best_mse_val = mse_val
        best_alpha = alpha
        best_model = lasso  # Save the best model

# Print the best alpha and validation MSE
print(f"Best alpha found: {best_alpha}")
print(f"Validation Mean Squared Error (MSE): {best_mse_val:.4f}")

# Use the best model to make predictions on the test set
lasso_pred_test = best_model.predict(x_test_norm)
mse_lasso_test = mean_squared_error(y_test, lasso_pred_test)
mae_lasso_test = mean_absolute_error(y_test, lasso_pred_test)
```

```
# Print results for test set
print("Test Mean Squared Error (MSE):", mse_lasso_test)
print("Test Mean Absolute Error (MAE):", mae_lasso_test)
```

```
Best alpha found: 45
Validation Mean Squared Error (MSE): 4398910243.3335
Test Mean Squared Error (MSE): 4910977739.600413
Test Mean Absolute Error (MAE): 50836.47618953052
```

Here, we create an instance of the Lasso regression model from scikit-learn The alpha parameter is calculated based on the minimum mse when applied on the validation dataset The lasso model is trained on the training data After training, the model is used to predict the target variable values for the validation data and test data It generates predicted values for the median house prices based on the features Mean Squared Error (MSE) and Mean Absolute Error (MAE) are calculated to evaluate the performance of the Lasso model

```
#3-ridge regression

# Define a range of alpha values to try
alpha_values = [0.1, 1, 50, 45]
best_alpha = None
best_mse_val = float('inf')
best_model = None

# Loop through each alpha value
for alpha in alpha_values:
    # Create a Ridge model with the current alpha value
    ridge = Ridge(alpha=alpha)
    ridge.fit(x_train_norm, y_train)

    # Predict on the validation set
    ridge_pred_val = ridge.predict(x_val_norm)

    # Calculate the Mean Squared Error on the validation set
    mse_val = mean_squared_error(y_val, ridge_pred_val)

    # Update best alpha if this one has the lowest validation MSE
    if mse_val < best_mse_val:
        best_mse_val = mse_val
        best_alpha = alpha
        best_model = ridge  # Save the best model

# Print the best alpha and validation MSE
print(f"Best alpha found: {best_alpha}")
print(f"Validation Mean Squared Error (MSE): {best_mse_val:.4f}")

# Use the best model to make predictions on the test set
ridge_pred_test = best_model.predict(x_test_norm)
```

```
mse_ridge_test = mean_squared_error(y_test, ridge_pred_test)
mae_ridge_test = mean_absolute_error(y_test, ridge_pred_test)

# Print results for test set
print("Test Mean Squared Error (MSE):", mse_ridge_test)
print("Test Mean Absolute Error (MAE):", mae_ridge_test)
```

```
Best alpha found: 1
Validation Mean Squared Error (MSE): 4400540039.5975
Test Mean Squared Error (MSE): 4907281049.444644
Test Mean Absolute Error (MAE): 50793.61026819888
```

Here, we're creating an instance of the Ridge regression model from scikit-learn The alpha parameter is calculated based on the minimum mse when applied on the validation dataset the ridge model is trained on the training data After training, the model is used to make predictions for the validation data and test data It generates predicted values for the median house prices based on the features

```
print("Mean Squared error and Mean Absolute Error using test data:")
print(f"Linear Regression\nMean Squared Error: {mse_linear_test:.4f}\
nMean Absolute Error: {mae_linear_test:.4f}")
print(f"Lasso Regression:\nMean Squared Error: {mse_lasso_test:.4f}\
nMean Absolute Error:{mae_lasso_test:.4f}")
print(f"Ridge Regression:\nMean Squared Error:{mse_ridge_test:.4f}\
nMean Absolute Error:{mae_ridge_test:.4f}")
```

```
Mean Squared error and Mean Absolute Error using test data:
Linear Regression
Mean Squared Error: 4907211997.3748
Mean Absolute Error: 50790.0603
Lasso Regression:
Mean Squared Error: 4910977739.6004
Mean Absolute Error:50836.4762
Ridge Regression:
Mean Squared Error:4907281049.4446
Mean Absolute Error:50793.6103
```

Linear Regression showed the lowest mean squared error and mean absolute error Lasso Regression produced a higher mean squared error and mean absolute error Ridge Regression produced a lower mean squared error than lasso conclusion: Linear Regression is performing well both Lasso and Ridge regressions did not provide a noticeable advantage in predictive accuracy