In [ ]:
```
import pandas as pd
import numpy as np
```

Reading Dataset

In [ ]:
```
df = pd.read_csv('California_Houses.csv')
df.head()
```

Out[ ]:

	Median_House_Value	Median_Income	Median_Age	Tot_Rooms	Tot_Bedrooms	Populati
0	452600.0	8.3252	41	880	129	3
1	358500.0	8.3014	21	7099	1106	24
2	352100.0	7.2574	52	1467	190	4
3	341300.0	5.6431	52	1274	235	5
4	342200.0	3.8462	52	1627	280	5

In [ ]:
```
X = df.drop('Median_House_Value', axis=1)
y = df['Median_House_Value']
```

Data Split

- Split the data into training, validation, and testing sets.
- 'X_train' and 'y_train' will be used for training your machine learning model with size 70% of the data.
- 'X_val' and 'y_val' will be used for validating and fine-tuning your model with size 15% of the data.
- 'X_test' and 'y_test' will be used for evaluating the final model's performance with size 15% of the data.

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_sta
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, rand
```

Feature Scaling

- The 'StandardScaler' performs a specific type of feature scaling called standardization or (Z-Normalization). Standardization transforms the data such that it has a mean of 0 and a standard deviation of 1. It's done by subtracting the mean and dividing by the standard deviation for each feature.

In [ ]:
```
from sklearn.discriminant_analysis import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_val = sc.transform(X_val)
X_test = sc.transform(X_test)
```

Linear Regression

In [ ]:
```
from sklearn.linear_model import LinearRegression


linear_regression = LinearRegression()
linear_regression.fit(X_train, y_train)

y_predict_linear = linear_regression.predict(X_val)
```

Linear regression : (MSE & MAE)

In [ ]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

print("Linear Regression:")
mse_linear = mean_squared_error(y_val, y_predict_linear)
print(f'Mean Squared Error : {mse_linear:.3f}')

mae_linear = mean_absolute_error(y_val, y_predict_linear)
print(f"Mean Absolute Error : {mae_linear:.3f}")
```

```
Linear Regression:
Mean Squared Error : 4494277636.747
Mean Absolute Error : 48984.680
```

Kfold Crossvalidation

- using kfold to get the best alpha.

In [ ]:
```
from sklearn.model_selection import RepeatedKFold

# cross validation
cross_validation = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)
alpha_values = np.arange(0.01, 1.01, 0.01)
alpha_values = np.append(alpha_values, [0.001, 0.003])
alphas = np.logspace(-4, 4, 9)
```

Ridge Regression

In [ ]:
```
from sklearn.linear_model import RidgeCV

ridge_regression = RidgeCV(alphas = alpha_values, cv=cross_validation)
ridge_regression.fit(X_train, y_train)

y_predict_ridge = ridge_regression.predict(X_val)
```

In [ ]:
```
print(ridge_regression.alpha_)
```

```
1.0
```

Ridge regression : (MSE & MAE)

In [ ]:
```
print("Ridge Regression:")
mse_ridge = mean_squared_error(y_val, y_predict_ridge)
print(f'Mean Squared Error : {mse_ridge:.3f}')

mae_ridge = mean_absolute_error(y_val, y_predict_ridge)
print(f"Mean Absolute Error : {mae_ridge:.3f}")
```

```
Ridge Regression:
Mean Squared Error : 4493471097.641
Mean Absolute Error : 48985.216
```

Lasso regression

In [ ]:
```
from sklearn.linear_model import LassoCV

lasso_regression = LassoCV(alphas=alpha_values, cv=cross_validation , max_iter=1000
lasso_regression.fit(X_train , y_train)

y_predict_lasso = lasso_regression.predict(X_val)
```

In [ ]:
```
print(lasso_regression.alpha_)
```

```
1.0
```

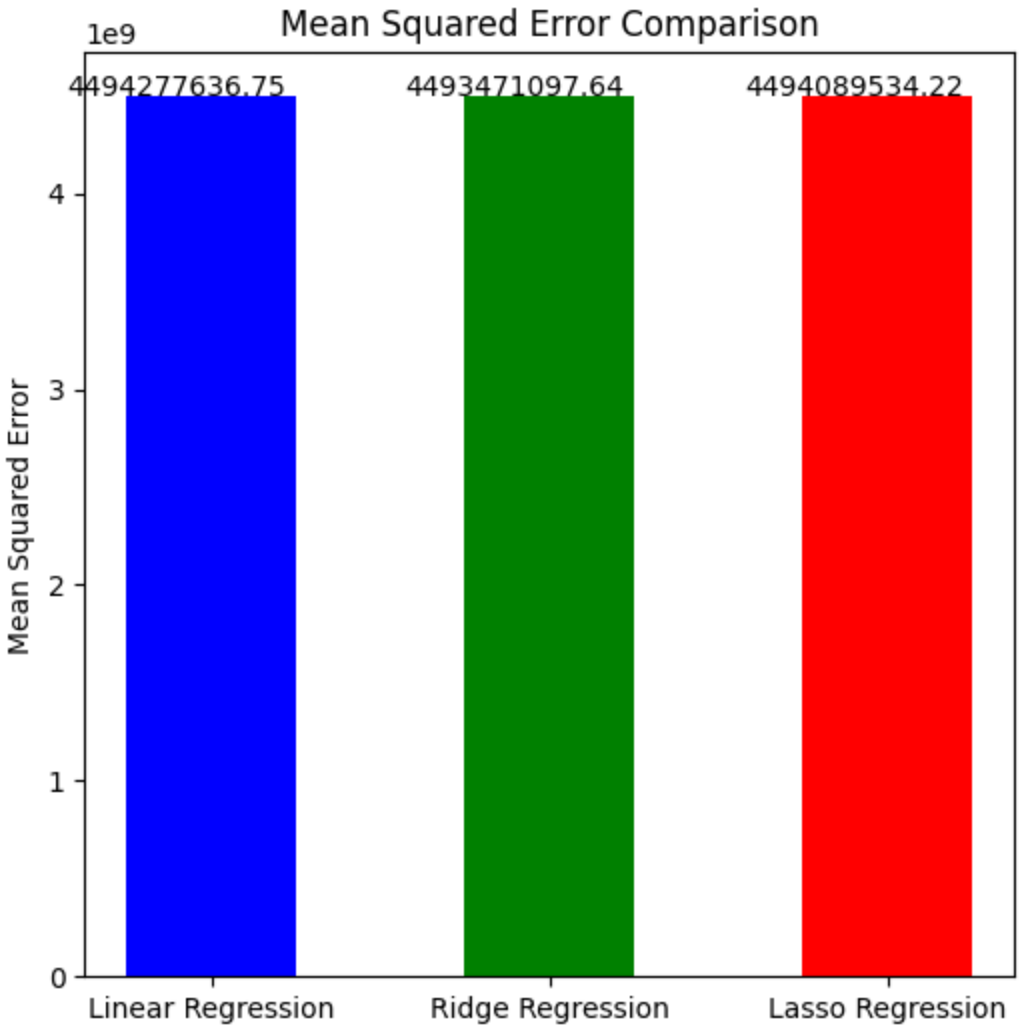
Lasso regression : (MSE & MAE)

In [ ]:
```
print("Lasso Regression:")
mse_lasso = mean_squared_error(y_val, y_predict_lasso)
print(f'Mean Squared Error : {mse_lasso:.3f}')

mae_lasso = mean_absolute_error(y_val, y_predict_lasso)
print(f"Mean Absolute Error : {mae_lasso:.3f}")
```

```
Lasso Regression:
Mean Squared Error : 4494089534.220
Mean Absolute Error : 48984.725
```

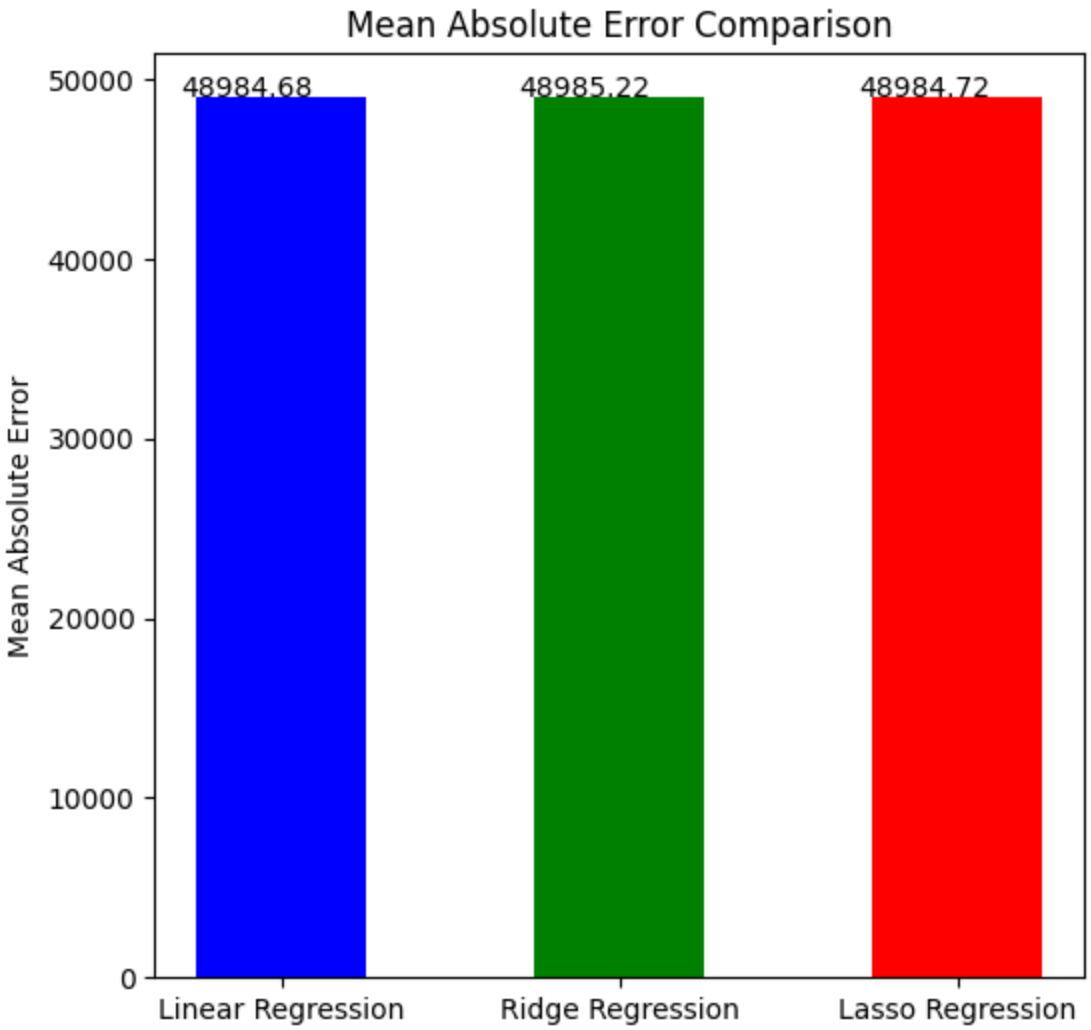
In [ ]:
```
import matplotlib.pyplot as plt

# Define the regression methods and their respective MSE values
```

```
methods = ["Linear Regression", "Ridge Regression", "Lasso Regression"]
mse_values = [mse_linear, mse_ridge, mse_lasso]

# Create a bar chart to compare MSE values
plt.figure(figsize=(6, 6))
bars = plt.bar(methods, mse_values, color=['blue', 'green', 'red'],width = 0.5)

# Annotate the bars with their respective MSE values
for bar, mse in zip(bars, mse_values):
    plt.text(bar.get_x() + bar.get_width() / 2 - 0.1, mse + 0.001, f'{mse:.2f}', ha

plt.title("Mean Squared Error Comparison")
plt.ylabel("Mean Squared Error")
plt.show()
```



In [ ]:
```
# Define the regression methods and their respective MAE values
methods = ["Linear Regression", "Ridge Regression", "Lasso Regression"]
mae_values = [mae_linear, mae_ridge, mae_lasso]  # Replace with your MAE values

# Create a bar chart to compare MAE values
plt.figure(figsize=(6, 6))
bars = plt.bar(methods, mae_values, color=['blue', 'green', 'red'], width =0.5)

# Annotate the bars with their respective MAE values
```

```
for bar, mae in zip(bars, mae_values):
    plt.text(bar.get_x() + bar.get_width() / 2 - 0.1, mae + 0.001, f'{mae:.2f}', ha

plt.title("Mean Absolute Error Comparison")
plt.ylabel("Mean Absolute Error")
plt.show()
```



Comment on the outputs

- The output is different in each model.
- The ridge and lasso has a more bias and less variance than linear regression.

Testing

In [ ]:
```
y_test_linear = linear_regression.predict(X_test)
y_test_ridge = ridge_regression.predict(X_test)
y_test_lasso = lasso_regression.predict(X_test)
```

In [ ]:
```
mse_linear_test = mean_squared_error(y_test, y_test_linear)
mse_ridge_test = mean_squared_error(y_test, y_test_ridge)
```

```
mse_lasso_test = mean_squared_error(y_test, y_test_lasso)
```

In [ ]:
```
mae_linear_test = mean_absolute_error(y_test, y_test_linear)
mae_ridge_test = mean_absolute_error(y_test, y_test_ridge)
mae_lasso_test = mean_absolute_error(y_test, y_test_lasso)
```

In [ ]:
```
print('Mean Squared Error in testing :')
print(f'Linear Regression : {mse_linear_test:.3f}')
print(f'Ridge Regression : {mse_ridge_test:.3f}')
print(f'Lasso Regression : {mse_lasso_test:.3f}')
```

```
Mean Squared Error in testing :
Linear Regression : 4917930970.874
Ridge Regression : 4917403467.016
Lasso Regression : 4917766703.280
```

In [ ]:
```
print('Mean Absolute Error in testing :')
print(f'Linear Regression : {mae_linear_test:.3f}')
print(f'Ridge Regression : {mae_ridge_test:.3f}')
print(f'Lasso Regression : {mae_lasso_test:.3f}')
```

```
Mean Absolute Error in testing :
Linear Regression : 50822.632
Ridge Regression : 50827.220
Lasso Regression : 50823.274
```