Linear Regression*

Outcome: Understanding of mathematical working behind the code

Importing neccesary libraries

```
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
```

Note: Do not import the libraries thar you are not using in the model, which may increase the complexity

```
housing = fetch_california_housing()
```

Note: This Dataset is used for example purposes, can decreses the R2 score as the **Linear Regression** model would never a best fit for this dataset

```
X = pd.DataFrame(housing.data, columns=housing.feature_names)[['MedInc']]
y = pd.Series(housing.target, name = "price")
```

Here X (independent variables) are always be in a 2D array suitable with sklearn

Here y (dependent variable) is always be in 1D array suitable with sklearn

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

random_state=42 This is the value for shuffling the data. the shuffling value should not be higher.

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

1. .fit() : this is actually the function which chose the best fit line for the dataset. It uses the **Gradient Descent** to compare the cost function with different lines and chose the line for which the cost function is minimum.
2. .fit() actually stores the coefficient and the intercept of the best fit line to predict the values.
3. Mathematical formula used:

- `(X^T X)^-1 (X^T Y)`
- this formula gives the 2X1 matrix where:
- X11 = intercept
- X21 = slope

4. .coef_ (the slope m): This tells us how much the Price (y) is expected to change for a one-unit increase in Median Income (x).
5. .intercept_ (the intercept b): This is the "starting point." It's the predicted Price (y) when the Median Income (x) is 0.

```
print("Slope (m):", model.coef_)
print("Intercept (b):", model.intercept_)
```

```
Slope (m): [0.41933849]
Intercept (b): 0.4445972916907879
```

```
m = model.coef_[0]
b = model.intercept_
print(f"Slope (m): {m:.4f}")
print(f"Intercept (b): {b:.4f}")

print(f"\nOur model's equation is: Price = {m:.4f} * MedInc + {b:.4f}")
```

```
Slope (m): 0.4193
Intercept (b): 0.4446

Our model's equation is: Price = 0.4193 * MedInc + 0.4446
```

```
y_pred = model.predict(X_test)
print(y_pred[:5])
print(y_test.head().values)
```

```
[1.14958917 1.50606882 1.90393718 2.85059383 2.00663318]
[0.477   0.458   5.00001 2.186   2.78   ]
```

1. .predict(): This function uses the equation of **Linear Regression** using the coefficient and intercept that .fit() generated.

- The eqation:
- `y = m x + c`
- `y = slope x + intercept`
- this eq. of y is used to predict the values

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MAE: {mae:.2f}")
print(f"MSE: {mse:.2f}")
print(f"RMSE: {np.sqrt(mse):.2f}")
print(f"R2 Score: {r2:.2f}")
```

```
MAE: 0.63
MSE: 0.71
RMSE: 0.84
R2 Score: 0.46
```