Check for Missing Values

```
import pandas as pd

# Sample data with missing values
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'Diana', None],
    'Age': [25, 30, None, 22, 28],
    'Score': [85, 90, 88, None, 95]
}

df = pd.DataFrame(data)

# Check for missing values
print(df.isnull().sum())

# Option 1: Drop rows with missing values
df_cleaned = df.dropna()

# Option 2: Fill missing values (e.g., with mean)
df_filled = df.fillna(df.mean(numeric_only=True))
```

```
Name     1
Age      1
Score    1
dtype: int64
```

Choose the Right Columns

```
# Show all columns
print(df.columns)

# Drop an irrelevant column (e.g., Name)
df_features = df.drop(columns=['Name'])

# Or manually select columns
selected_features = df[['Age', 'Score']]
```

```
Index(['Name', 'Age', 'Score'], dtype='object')
```

💻 Let's Train a Simple Model!

```
from sklearn.linear_model import LinearRegression

X = [[1], [2], [3], [4]]
y = [2, 4, 6, 8]

model = LinearRegression()
model.fit(X, y)

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

```
m: [2.]
b: 0.0
```

Making Predictions – Using the Model

```
model.predict([[5]])
```

```
array([10.])
```

Understanding Error Metrics – How Good Are Our Predictions?

Let’s measure how well the model fits the data.

```
from sklearn.metrics import mean_squared_error

y_true = [2, 4, 6, 8]
y_pred = model.predict([[1], [2], [3], [4]])
mean_squared_error(y_true, y_pred)
```

```
0.0
```

Let’s Try New Data – How Does the Line Change?

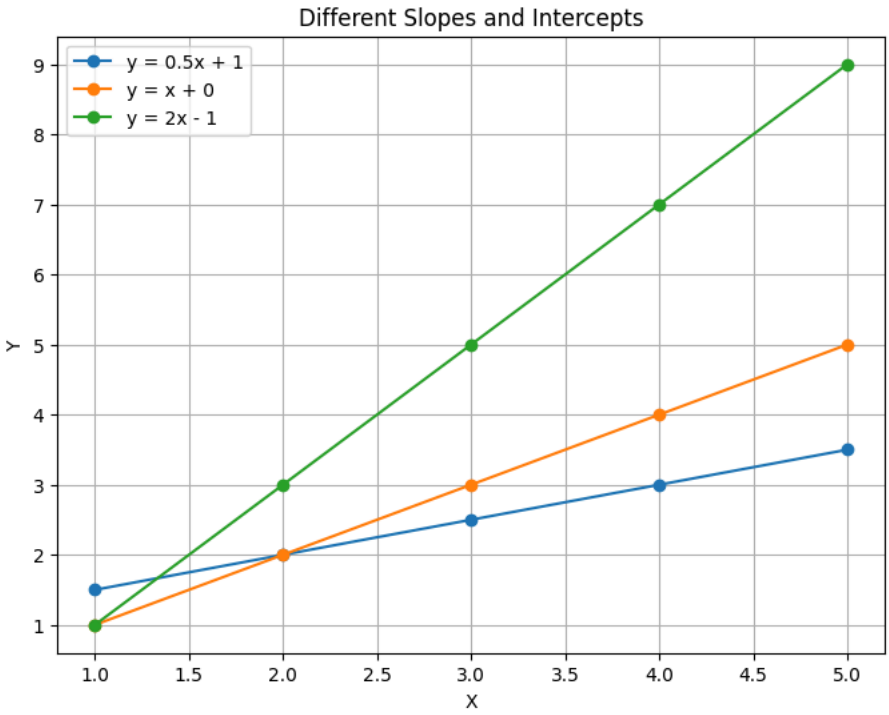
🎯 Goal: What happens if we change the data? Let’s explore!

```
from sklearn.linear_model import LinearRegression
import numpy as np

X = [[1], [2], [3], [4]]
y = [3, 5, 7, 9]

model = LinearRegression()
model.fit(X, y)

print("m (slope):", model.coef_)
print("b (intercept):", model.intercept_)
```

```
m (slope): [2.]
b (intercept): 1.0
```

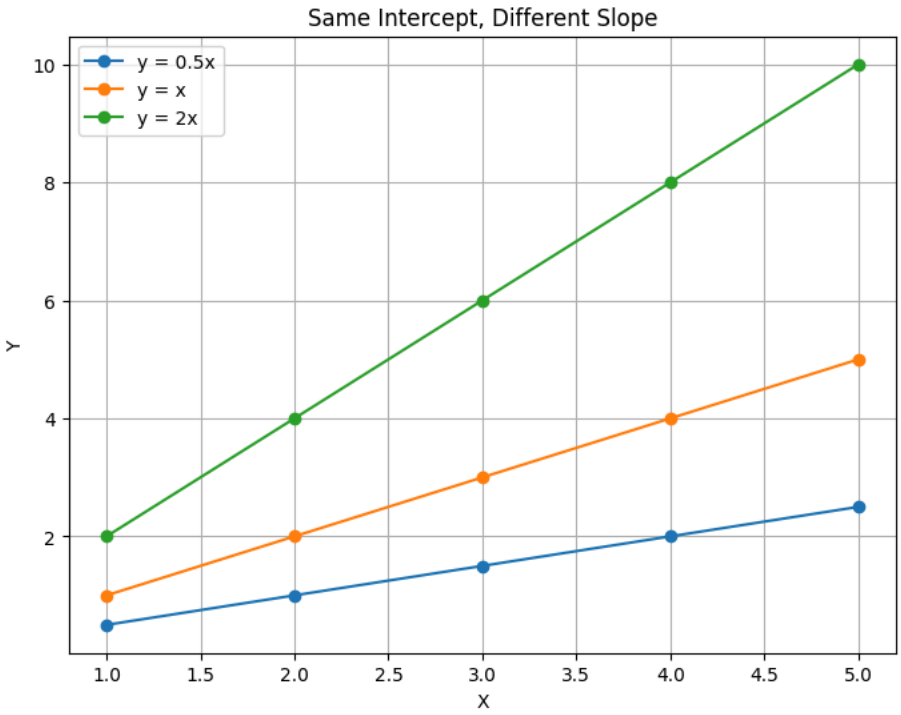
Different Slope and Intercept

```
import numpy as np
import matplotlib.pyplot as plt

# X values
X = np.array([1, 2, 3, 4, 5])

# Define lines with different slopes and intercepts
lines = {
    'y = 0.5x + 1': 0.5 * X + 1,
    'y = x + 0': X,
    'y = 2x - 1': 2 * X - 1
}

# Plotting the lines
plt.figure(figsize=(8, 6))
for label, y_values in lines.items():
    plt.plot(X, y_values, marker='o', label=label)

plt.title('Different Slopes and Intercepts')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.show()
```



Double-click (or enter) to edit

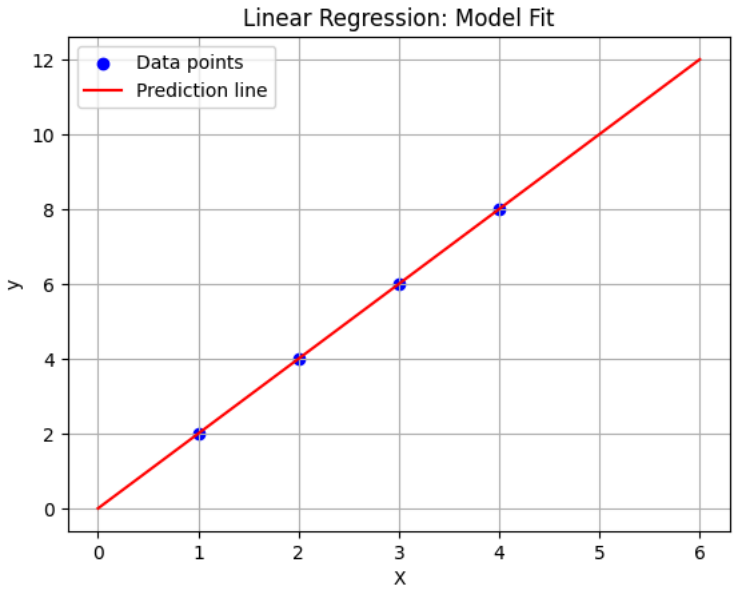
```
import numpy as np
import matplotlib.pyplot as plt

# X values
X = np.array([1, 2, 3, 4, 5])

# Lines with same intercept (0) but different slopes
lines = {
    'y = 0.5x': 0.5 * X,
    'y = x': X,
    'y = 2x': 2 * X
}

# Plotting the lines
plt.figure(figsize=(8, 6))
for label, y_values in lines.items():
    plt.plot(X, y_values, marker='o', label=label)

plt.title('Same Intercept, Different Slope')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.show()
```



Mini Project: Predict House Prices

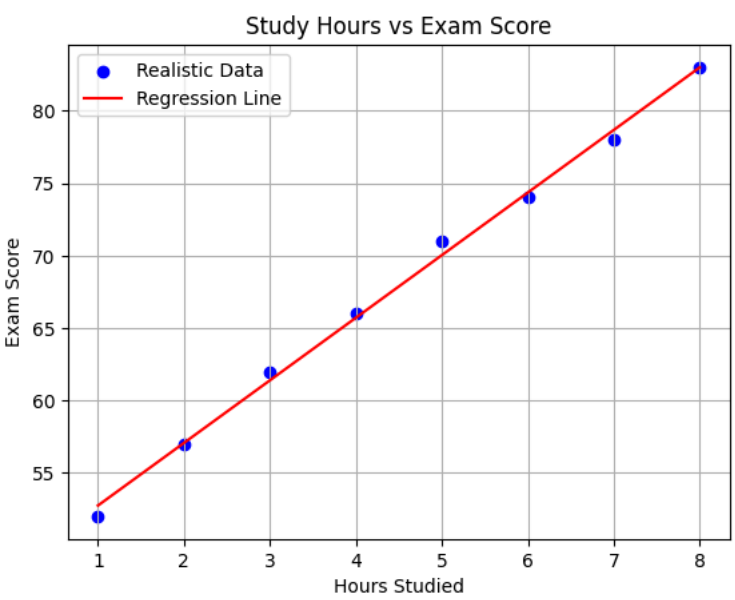
Visualizing the Model – Line of Best Fit

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression

# Data
X = np.array([[1], [2], [3], [4]])
y = np.array([2, 4, 6, 8])

# Train the model
model = LinearRegression()
model.fit(X, y)

# Generate line for prediction
X_line = np.linspace(0, 6, 100).reshape(-1, 1)
y_line = model.predict(X_line)

# Plot
plt.scatter(X, y, color='blue', label='Data points')
plt.plot(X_line, y_line, color='red', label='Prediction line')
plt.title('Linear Regression: Model Fit')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```



Visualising the Model – Real-World Data

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import numpy as np

# Simulated real-world-like data: Study hours vs Exam scores
X = np.array([[1], [2], [3], [4], [5], [6], [7], [8]])
y = np.array([52, 57, 62, 66, 71, 74, 78, 83])  # more realistic exam scores

model = LinearRegression()
model.fit(X, y)

# Plot
plt.scatter(X, y, color='blue', label='Realistic Data')
plt.plot(X, model.predict(X), color='red', label='Regression Line')
plt.xlabel("Hours Studied")
plt.ylabel("Exam Score")
plt.title("Study Hours vs Exam Score")
plt.legend()
plt.grid(True)
plt.show()
```



Evaluating the Model – How Good Are the Predictions?

Let's measure how accurate our model really is!

✅ 1. Veri Setini Eğitim ve Test Olarak Ayır:

```
from sklearn.model_selection import train_test_split

# Örnek veri seti:
X = [[1], [2], [3], [4], [5], [6], [7], [8]]
y = [2, 4, 6, 8, 10, 12, 14, 16]

# Veriyi eğitim ve test seti olarak ayır
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

✅ 2. Modeli Eğit:

```
from sklearn.linear_model import LinearRegression

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

```
▾ LinearRegression ⓘ ?
LinearRegression()
```

✅ 3. Modeli Değerlendir:

```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test)

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

print("Mean Squared Error:", mse)
print("R² Score:", r2)
```

```
Mean Squared Error: 0.0
R² Score: 1.0
```

All code

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Data
X = [[1], [2], [3], [4], [5], [6], [7], [8]]
y = [2, 4, 6, 8, 10, 12, 14, 16]

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R² Score:", r2)
```

```
Mean Squared Error: 0.0
R² Score: 1.0
```

Using Real-World Data for Linear Regression

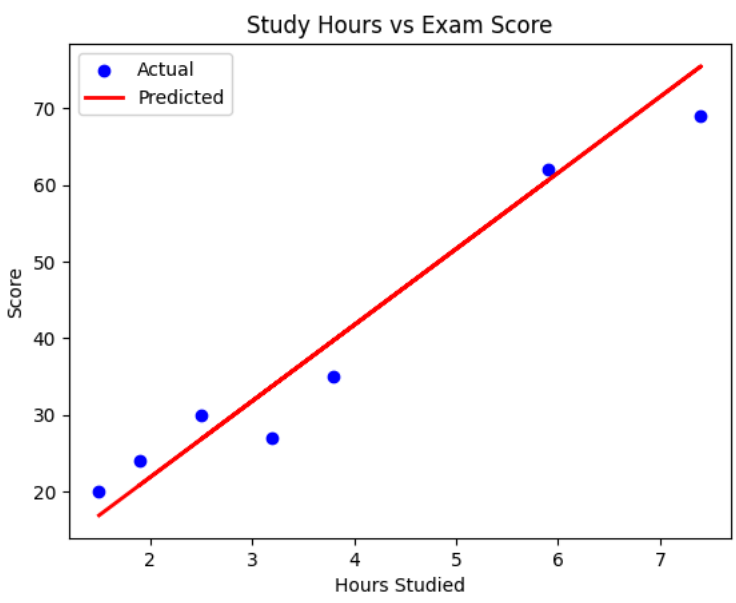
Predicting Exam Scores from Study Hours

Source of dataset

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Load data
url = "http://bit.ly/w-data"
data = pd.read_csv(url)

# Split features and labels
X = data[['Hours']]
y = data['Scores']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R² Score:", r2)
```

```
Mean Squared Error: 20.33292367497996
R² Score: 0.9367661043365056
```

```
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Predicted')
plt.xlabel("Hours Studied")
plt.ylabel("Score")
plt.title("Study Hours vs Exam Score")
plt.legend()
plt.show()
```



Let's Predict: What If You Study 9.25 Hours?

```
import pandas as pd
```

```
hours_df = pd.DataFrame({'Hours': [9.25]})
predicted_score = model.predict(hours_df)
print(f"Predicted Score: {predicted_score[0]:.2f}")
```

```
Predicted Score: 93.89
```

```
import matplotlib.pyplot as plt

# Plot the actual data points (scatter plot)
```