```
x = np.array([1, 2, 3, 4, 5, 6, 7])
y = np.array([2, 3, 5, 4, 6, 8, 9])
slope,intercept = np.polyfit(x,y,1)
slope,intercept
cov_xx = np.cov(x,y)[0][0]
cov_xy = np.cov(x,y)[0][1]
cov_yy = np.cov(x,y)[1][1]
m = cov_xy/cov_xx
b = np.mean(y)-m*np.mean(x)
y_line = m*x + b
plt.plot(x,y_line)
plt.scatter(x,y)
```

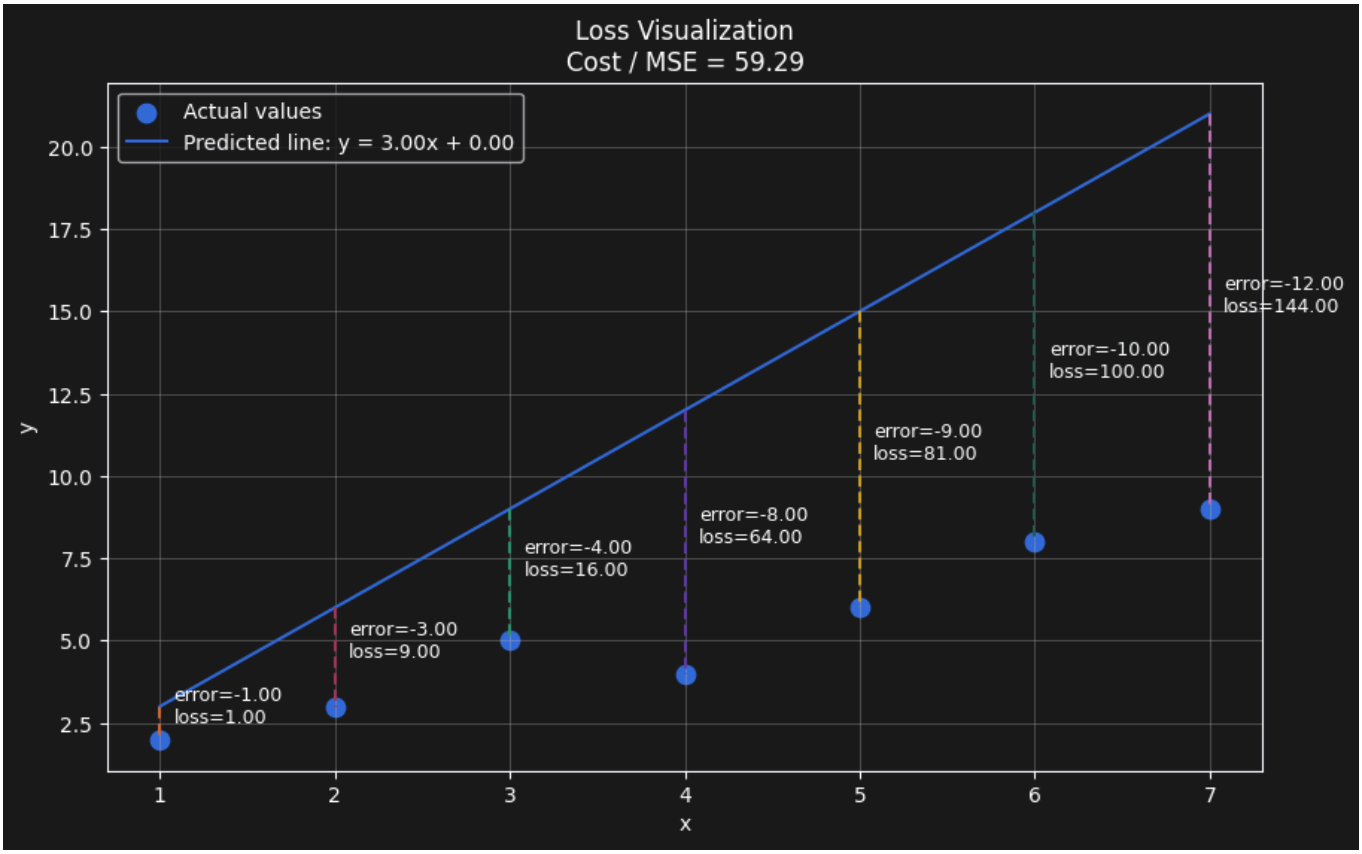
Show with different m and b

and then use apt m ,b

then use good m ,b

```
# A simple prediction line: y_pred = mx + b
m = 3
b = 0
y_pred = m * x + b
# Error and squared loss
errors = y - y_pred
squared_losses = errors ** 2
# Cost function: Mean Squared Error
cost = np.mean(squared_losses)
plt.figure(figsize=(10, 6))
# Actual data points
plt.scatter(x, y, s=80, label="Actual values")
# Predicted line
plt.plot(x, y_pred, label=f"Predicted line: y = {m:.2f}x + {b:.2f}")
# Draw vertical loss/error lines from predicted value to actual value
for xi, actual, pred, error, loss in zip(x, y, y_pred, errors,
squared_losses):
    plt.plot([xi, xi], [pred, actual], linestyle="--")
    mid_y = (actual + pred) / 2
    plt.text(
    xi + 0.08,
    mid_y,
    f"error={error:.2f}\nloss={loss:.2f}",
    fontsize=9
```

```
    )
plt.title(f"Loss Visualization\nCost / MSE = {cost:.2f}")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.legend()
plt.show()
```




```
import matplotlib.pyplot as plt
m = 8000
b = 25000
predicted_salary = m * experience + b
mse = np.mean((actual_salary - predicted_salary) ** 2)
plt.scatter(experience, actual_salary, label='Actual Salary')
plt.plot(experience, predicted_salary, marker='*',c='r', label='Predicted
Salary')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Linear Regression Prediction')
plt.legend()
plt.grid(True)
plt.show()
print("Predicted Salary:", predicted_salary)
print("Cost using MSE:", mse)
```

#Finding best m ,b

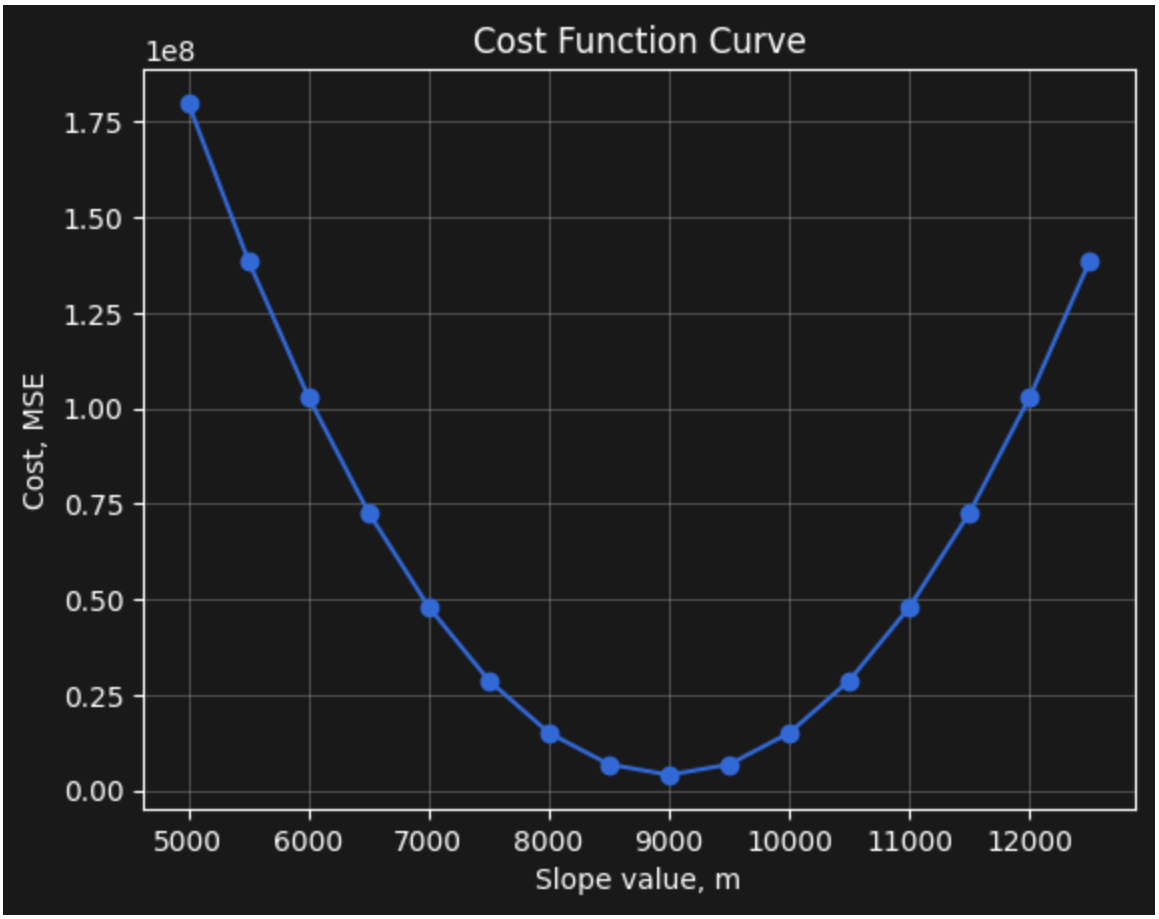
```
# sklearn expects X to be 2D
X = experience.reshape(-1, 1)
y = actual_salary

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

m = model.coef_[0]
b = model.intercept_
m,b
```

```
m,b = np.polyfit(experience,actual_salary,1)
m,b
```

```
experience = np.array([1, 2, 3, 4, 5])
actual_salary = np.array([30000, 35000, 50000, 55000, 65000])
cov_xy = np.cov(experience,actual_salary)[0][1]
cov_xx = np.cov(experience,actual_salary)[0][0]
m = cov_xy/cov_xx
m
```

```
import numpy as np
import matplotlib.pyplot as plt
experience = np.array([1, 2, 3, 4, 5])
actual_salary = np.array([30000, 35000, 50000, 55000, 65000])
b = 20000
m_values = np.arange(5000, 13000, 500)
costs = []
for m in m_values:
    predicted_salary = m * experience + b
    mse = np.mean((actual_salary - predicted_salary) ** 2)
    costs.append(mse)
print(costs)
plt.plot(m_values, costs, marker='o')
plt.xlabel('Slope value, m')
plt.ylabel('Cost, MSE')
plt.title('Cost Function Curve')
plt.grid(True)
plt.show()
```

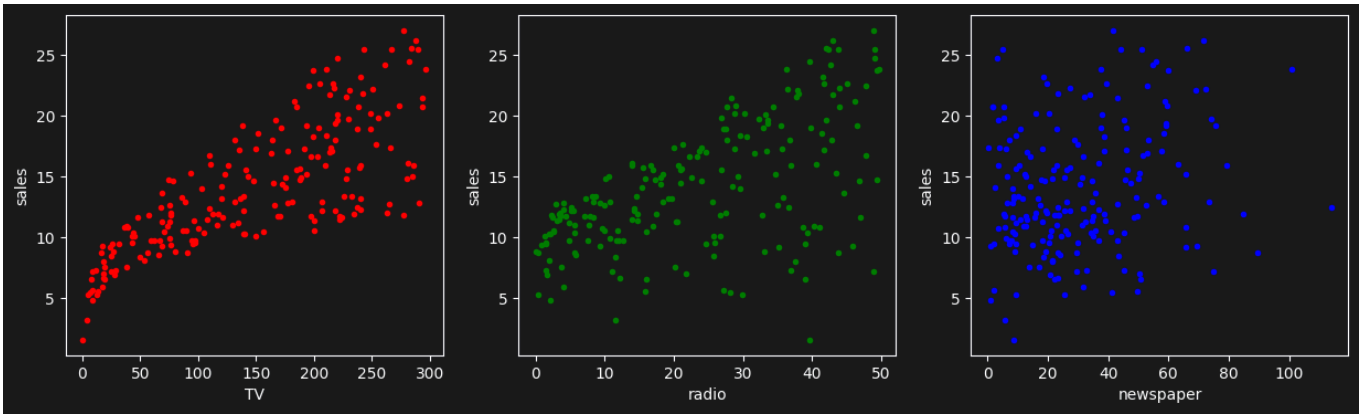
```
best_index = np.argmin(costs)
print("Best m:", m_values[best_index])
print("Lowest cost:", costs[best_index])
```



Sales

```
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn import linear_model
import pandas as pd

data_path = "https://www.statlearning.com/s/Advertising.csv"

# Read the CSV data from the link
data_df = pd.read_csv(data_path, index_col=0)

# Print out first 5 samples from the DataFrame
data_df.head()
fig = plt.figure(figsize=(15,4))
gs = mpl.gridspec.GridSpec(1,3)

# Plot of sales vs TV
ax = fig.add_subplot(gs[0])
```

```
ax.scatter(data_df["TV"], data_df["sales"], color="red", marker=".")
ax.set_xlabel("TV")
ax.set_ylabel("sales")

# Plot of sales vs radio
ax = fig.add_subplot(gs[1])
ax.scatter(data_df["radio"], data_df["sales"], color="green", marker=".")
ax.set_xlabel("radio")
ax.set_ylabel("sales")

# Plot of sales vs newspaper
ax = fig.add_subplot(gs[2])
ax.scatter(data_df["newspaper"], data_df["sales"], color="blue", marker=".")
ax.set_xlabel("newspaper")
ax.set_ylabel("sales")

plt.show()
```




```
import matplotlib.pyplot as plt
from sklearn import linear_model
import pandas as pd

data_path = "https://www.statlearning.com/s/Advertising.csv"
data_df = pd.read_csv(data_path, index_col=0)

features = ["TV", "radio", "newspaper"]
colors = ["red", "green", "blue"]

fig, axes = plt.subplots(1, 3, figsize=(15, 4))

for ax, feature, color in zip(axes, features, colors):
    X = data_df[[feature]]  # 2D input for sklearn
    Y = data_df["sales"]  # target

    reg = linear_model.LinearRegression()
```

```
    reg.fit(X, Y)

    m = reg.coef_[0]
    c = reg.intercept_
    y_hat = m * data_df[feature] + c

    ax.scatter(data_df[feature], data_df["sales"], color=color, marker=".")
    ax.plot(data_df[feature], y_hat, color="black")

    ax.set_xlabel(feature)
    ax.set_ylabel("sales")
    ax.set_title(f"sales vs {feature} with {m.round(2)}*x+{c.round(2)}")

plt.tight_layout()
plt.show()
```