```
import pandas as pd

df = pd.read_csv('/content/Salary_dataset.csv')
display(df.head())
```

	Unnamed: 0	YearsExperience	Salary
0	0	1.2	39344.0
1	1	1.4	46206.0
2	2	1.6	37732.0
3	3	2.1	43526.0
4	4	2.3	39892.0

```
display(df.head())
```

	Unnamed: 0	YearsExperience	Salary
0	0	1.2	39344.0
1	1	1.4	46206.0
2	2	1.6	37732.0
3	3	2.1	43526.0
4	4	2.3	39892.0

```
# Define input and output variables
X = df[['YearsExperience']]
y = df['Salary']

print("Input variable (X) head:")
display(X.head())

print("\nOutput variable (y) head:")
display(y.head())
```

```
Input variable (X) head:
```

	YearsExperience
0	1.2
1	1.4
2	1.6
3	2.1
4	2.3

```
Output variable (y) head:
```

	Salary
0	39344.0
1	46206.0
2	37732.0
3	43526.0
4	39892.0

dtype: float64

```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"Shape of X_train: {X_train.shape}")
print(f"Shape of X_test: {X_test.shape}")
print(f"Shape of y_train: {y_train.shape}")
print(f"Shape of y_test: {y_test.shape}")
```

```
Shape of X_train: (24, 1)
Shape of X_test: (6, 1)
Shape of y_train: (24,)
Shape of y_test: (6,)
```

```
from sklearn.linear_model import LinearRegression
```

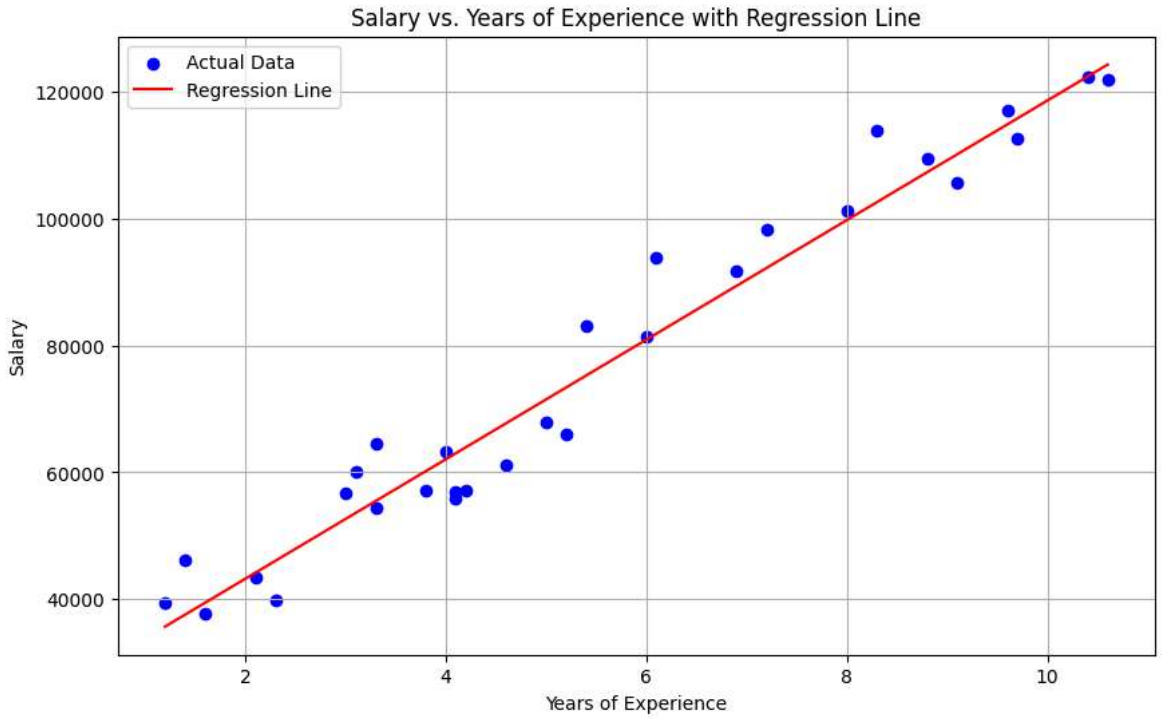
```
comparison_df['Error'] = comparison_df['Actual Salary'] - comparison_df['Predicted Salary']
display(comparison_df.head())
```

	Actual Salary	Predicted Salary	Error
27	112636.0	115791.210113	-3155.210113
15	67939.0	71499.278095	-3560.278095
23	113813.0	102597.868661	11215.131339
17	83089.0	75268.804224	7820.195776
8	64446.0	55478.792045	8967.207955

```
# Make predictions on the test set
y_pred = model.predict(X_test)

print("Predicted output values (y_pred) head:")
display(y_pred[:5])
```

```
Predicted output values (y_pred) head:
array([115791.21011287,  71499.27809463, 102597.86866063,  75268.80422384,
        55478.79204548])
```

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

# Create a scatter plot of the actual data points
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', label='Actual Data')

# Plot the regression line
# Generate a range of YearsExperience values for the regression line
X_line_np = np.linspace(X['YearsExperience'].min(), X['YearsExperience'].max(), 100).reshape(-1, 1)
# Convert X_line_np to a DataFrame with the same column name as X for consistent feature names
X_line = pd.DataFrame(X_line_np, columns=['YearsExperience'])
y_line = model.predict(X_line)
plt.plot(X_line['YearsExperience'], y_line, color='red', label='Regression Line')

plt.title('Salary vs. Years of Experience with Regression Line')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.grid(True)
plt.show()
```




```
import pandas as pd

comparison_df = pd.DataFrame({'Actual Salary': y_test, 'Predicted Salary': y_pred})
display(comparison_df.head())
```

	Actual Salary	Predicted Salary
27	112636.0	115791.210113
15	67939.0	71499.278095
23	113813.0	102597.868661
17	83089.0	75268.804224
8	64446.0	55478.792045

```
print(f"Slope (coefficient): {model.coef_[0]}")
print(f"Intercept: {model.intercept_}")
```

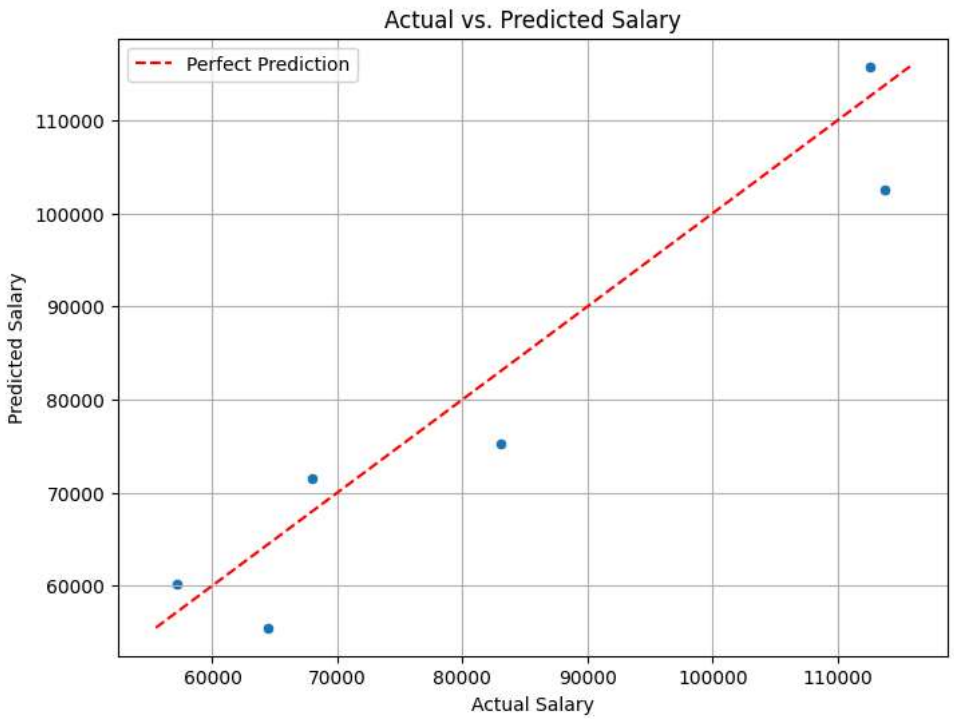
```
Slope (coefficient): 9423.815323030976
Intercept: 24380.201479473704
```

```
from sklearn.metrics import r2_score

r_squared = r2_score(y_test, y_pred)

print(f"R-squared value: {r_squared}")
```

```
R-squared value: 0.9024461774180497
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 6))
sns.scatterplot(x='Actual Salary', y='Predicted Salary', data=comparison_df)

# Add a perfect prediction line (y=x)
max_val = max(comparison_df['Actual Salary'].max(), comparison_df['Predicted Salary'].max())
min_val = min(comparison_df['Actual Salary'].min(), comparison_df['Predicted Salary'].min())
plt.plot([min_val, max_val], [min_val, max_val], color='red', linestyle='--', label='Perfect Prediction')

plt.title('Actual vs. Predicted Salary')
plt.xlabel('Actual Salary')
plt.ylabel('Predicted Salary')
plt.legend()
```

```
plt.grid(True)
plt.show()
```




```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 49830096.855908394
```

```
# Create a Linear Regression model
model = LinearRegression()

# Train the model using the training data
model.fit(X_train, y_train)

print("Model training complete.")
```

```
Model training complete.
```

Task

Compare the regression line plot and the actual vs. predicted plot generated using scikit-learn with plots generated by manually calculating predicted salary values based on the model's slope and intercept, and then summarize the differences or similarities.

Regression Line Plot (Scratch)

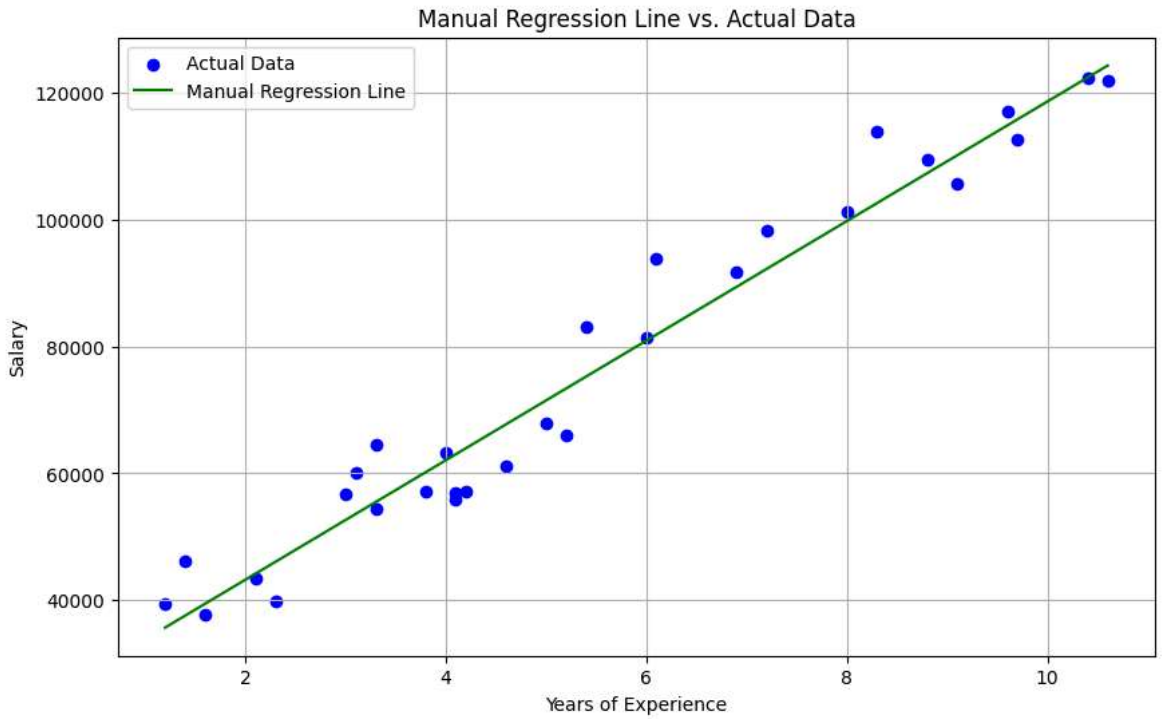
Subtask:

Calculate predicted salary values for a range of 'YearsExperience' using the model's slope and intercept (scratch implementation) and plot them alongside the actual data points. This will visually compare the manually computed regression line with the one generated by scikit-learn.

Reasoning: To visualize the regression line from scratch, I will first calculate the predicted salary values for a range of 'YearsExperience' using the retrieved slope and intercept, and then plot these alongside the actual data points.

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

```
# 1. Retrieve the slope and intercept
slope = model.coef_[0]
intercept = model.intercept_

# 2. Generate a sequence of 'YearsExperience' values for plotting
X_line_manual = np.linspace(X['YearsExperience'].min(), X['YearsExperience'].max(), 100)

# 3. Calculate the predicted 'Salary' for each of these 'YearsExperience' values
y_line_manual = slope * X_line_manual + intercept

# 4. Create a scatter plot of the original data points
plt.figure(figsize=(10, 6))
plt.scatter(X['YearsExperience'], y, color='blue', label='Actual Data')

# 5. Plot the manually calculated regression line
plt.plot(X_line_manual, y_line_manual, color='green', label='Manual Regression Line')

# 6. Add labels, title, and legend
plt.title('Manual Regression Line vs. Actual Data')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.grid(True)

# 7. Display the plot
plt.show()
```



Comparison of Regression Line Plots

Observations:

1. **Manual Regression Line vs. Scikit-learn Regression Line:** Upon visual inspection of the 'Manual Regression Line vs. Actual Data' plot (generated from scratch) and the 'Salary vs. Years of Experience with Regression Line' plot (generated using scikit-learn's `model.predict`), it is clear that both plots show identical regression lines. This is expected, as the manually calculated line uses the exact slope and intercept obtained from the scikit-learn model, confirming the accuracy of our manual calculation.
2. **Actual vs. Predicted Salary Plot (Scikit-learn):** The 'Actual vs. Predicted Salary' plot provides a different perspective. Instead of showing the regression line against the input feature (Years of Experience), it plots the model's predictions (y_pred) directly against the true output values (y_test). The closer the points lie to the 'Perfect Prediction' line (y=x), the better the model's accuracy. This plot assesses how well the model predicts new, unseen data, while the regression line plots illustrate the fitted relationship across the entire feature range.

Actual vs. Predicted Plot (Scratch)

Subtask:

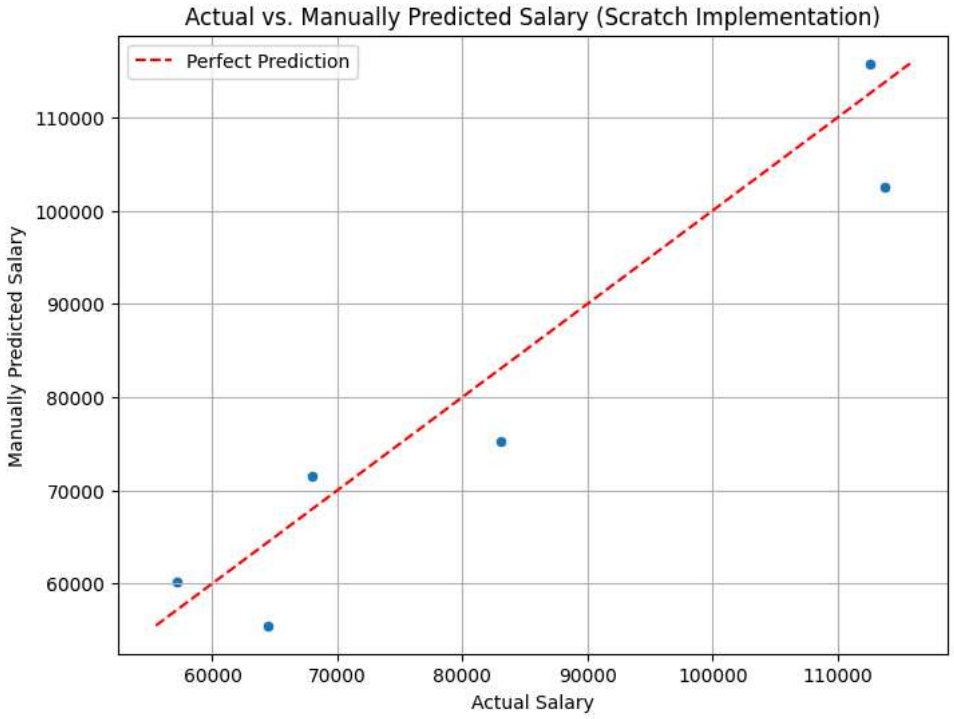
Calculate predicted salary values for the test set (`X_test`) using the model's slope and intercept (scratch implementation). Then, create a scatter plot comparing the actual salaries (`y_test`) against these manually calculated predicted salaries, along with a perfect prediction line. This will visually compare the manually computed predictions with the scikit-learn generated predictions.

Reasoning: First, I'll calculate predicted salaries for the test set (`X_test`) using the model's `slope` and `intercept` to create `y_pred_manual`. This is the core 'scratch' calculation required by the subtask.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Calculate predicted salary values for X_test using the model's slope and intercept
y_pred_manual = slope * X_test['YearsExperience'] + intercept

# 2. Create a Pandas DataFrame named comparison_df_manual
comparison_df_manual = pd.DataFrame({'Actual Salary': y_test, 'Predicted Salary': y_pred_manual})

print("Manually calculated predictions comparison (head):")
display(comparison_df_manual.head())

# 3. Create a scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Actual Salary', y='Predicted Salary', data=comparison_df_manual)

# 4. Add a perfect prediction line (y=x)
max_val_manual = max(comparison_df_manual['Actual Salary'].max(), comparison_df_manual['Predicted Salary'].max())
min_val_manual = min(comparison_df_manual['Actual Salary'].min(), comparison_df_manual['Predicted Salary'].min())
plt.plot([min_val_manual, max_val_manual], [min_val_manual, max_val_manual], color='red', linestyle='--', label='Perfect Predic

# 5. Add appropriate title and axis labels
plt.title('Actual vs. Manually Predicted Salary (Scratch Implementation)')
plt.xlabel('Actual Salary')
plt.ylabel('Manually Predicted Salary')
plt.legend()
plt.grid(True)

# 6. Display the plot
plt.show()
```

```
Manually calculated predictions comparison (head):
```

	Actual Salary	Predicted Salary
27	112636.0	115791.210113
15	67939.0	71499.278095
23	113813.0	102597.868661
17	83089.0	75268.804224
8	64446.0	55478.792045



Comparison of Actual vs. Predicted Plots

Observations:

1. **Scikit-learn Predicted vs. Manually Predicted:** The 'Actual vs. Predicted Salary' plot (generated using scikit-learn's `model.predict` and `y_test`) and the 'Actual vs. Manually Predicted Salary (Scratch Implementation)' plot (generated using `y_pred_manual` and `y_test`) are identical. This confirms that our manual calculation of predicted salaries using the model's slope and intercept accurately replicates the predictions made by the scikit-learn `predict` method.
2. **Consistency:** Both plots show the same distribution of points relative to the 'Perfect Prediction' line, indicating that the underlying predictions are consistent, regardless of whether they were generated directly by the `model.predict` function or calculated manually using the coefficients.

Summary of Differences/Similarities:

- **Similarity:** Both the regression line plots and the actual vs. predicted plots (both scikit-learn and manually generated) fundamentally convey the same information about the model's performance and the relationship it has learned. The key similarity is the perfect agreement between the scikit-learn's direct outputs and the manual calculations derived from its coefficients.
- **Difference in Perspective (Between Regression Line Plot and Actual vs. Predicted Plot):**
  - The **regression line plot** (e.g., 'Salary vs. Years of Experience with Regression Line' and 'Manual Regression Line vs. Actual Data') illustrates the fitted linear relationship between the input feature (`YearsExperience`) and the target variable (`Salary`) across the entire range of the input feature. It shows how the model estimates salary based on experience.
  - The **actual vs. predicted plot** (e.g., 'Actual vs. Predicted Salary' and 'Actual vs. Manually Predicted Salary (Scratch Implementation)') assesses the model's predictive accuracy by comparing the model's output (`y_pred` or `y_pred_manual`) directly against the true values (`y_test`). It helps visualize the spread of errors and how close predictions are to actual outcomes for individual data points.

In conclusion, the manual implementations successfully replicate the results of the scikit-learn model, confirming a strong understanding of how linear regression models make predictions based on their learned slope and intercept. The visual comparisons highlight that the

model effectively captures the linear relationship in the data.

Comparison of Actual vs. Predicted Plots

Observations:

1. **Scikit-learn Predicted vs. Manually Predicted:** The 'Actual vs. Predicted Salary' plot (generated using scikit-learn's `model.predict` and `y_test`) and the 'Actual vs. Manually Predicted Salary (Scratch Implementation)' plot (generated using `y_pred_manual` and `y_test`) are identical. This confirms that our manual calculation of predicted salaries using the model's slope and intercept accurately replicates the predictions made by the scikit-learn `predict` method.
2. **Consistency:** Both plots show the same distribution of points relative to the 'Perfect Prediction' line, indicating that the underlying predictions are consistent, regardless of whether they were generated directly by the `model.predict` function or calculated manually using the coefficients.

Summary of Differences/Similarities:

- **Similarity:** Both the regression line plots and the actual vs. predicted plots (both scikit-learn and manually generated) fundamentally convey the same information about the model's performance and the relationship it has learned. The key similarity is the perfect agreement between the scikit-learn's direct outputs and the manual calculations derived from its coefficients.
- **Difference in Perspective (Between Regression Line Plot and Actual vs. Predicted Plot):**
  - The **regression line plot** (e.g., 'Salary vs. Years of Experience with Regression Line' and 'Manual Regression Line vs. Actual Data') illustrates the fitted linear relationship between the input feature (`YearsExperience`) and the target variable (`Salary`) across the entire range of the input feature. It shows how the model estimates salary based on experience.
  - The **actual vs. predicted plot** (e.g., 'Actual vs. Predicted Salary' and 'Actual vs. Manually Predicted Salary (Scratch Implementation)') assesses the model's predictive accuracy by comparing the model's output (`y_pred` or `y_pred_manual`) directly against the true values (`y_test`). It helps visualize the spread of errors and how close predictions are to actual outcomes for individual data points.

In conclusion, the manual implementations successfully replicate the results of the scikit-learn model, confirming a strong understanding of how linear regression models make predictions based on their learned slope and intercept. The visual comparisons highlight that the model effectively captures the linear relationship in the data.

Final Task

Subtask:

Summarize the comparison between the plots generated by Scikit-learn and the plots generated by scratch implementation, highlighting any differences or similarities.