```
import os
os.environ['KAGGLE_USERNAME'] = 'sreeyadaripelli'
os.environ['KAGGLE_KEY'] = '5e0752429f497134d5284d136fd300fb'
!kaggle datasets download -d abhishek14398/salary-dataset-simple-linear-regress
!unzip salary-dataset-simple-linear-regression.zip
```

```
Dataset URL: https://www.kaggle.com/datasets/abhishek14398/salary-dataset-simple
License(s): CC0-1.0
Downloading salary-dataset-simple-linear-regression.zip to /content
  0% 0.00/457 [00:00<?, ?B/s]
100% 457/457 [00:00<00:00, 1.28MB/s]
```

```
: os
: contents of the /content directory to find the correct path for Salary_Data.csv
: /content/
```

```
salary-dataset-simple-linear-regression.zip  sample_data/
```

```
import os

# List the contents of the zip file to understand its structure
!unzip -l /content/salary-dataset-simple-linear-regression.zip
```

```
Archive:  /content/salary-dataset-simple-linear-regression.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
      664  2023-01-10 03:55   Salary_dataset.csv
---------                     -------
      664                     1 file
```

Task

Perform a simple linear regression analysis on the "Salary_dataset.csv" dataset to predict salary based on years of experience. This analysis should include loading the data, preparing it for modeling, training a linear regression model, making predictions, evaluating model performance using MSE and R-squared, visualizing the regression results, and interpreting the model's accuracy and key findings.

Load Dataset

Subtask:

Load the 'Salary_dataset.csv' dataset into a pandas DataFrame.

Reasoning: The previous attempts to load the CSV file failed because the file was not found directly in the `/content/` directory, despite the zip file being downloaded there.

To ensure the `Salary_dataset.csv` file is correctly extracted and available for loading, I will explicitly run the `unzip` command again, directing it to extract the contents into the `/content/` directory. After successful extraction, I will proceed to import pandas, load the CSV into a DataFrame, and display its head and columns as requested by the subtask.

```
import pandas as pd
import os

# Step 1: Extract the zip file explicitly to /content/
!unzip -o /content/salary-dataset-simple-linear-regression.zip -d /content/

# Step 3: Load the 'Salary_dataset.csv' file from the '/content/' directory int
df_salary = pd.read_csv('/content/Salary_dataset.csv')

# Step 4: Display the first 5 rows of the df_salary DataFrame
print("First 5 rows of the DataFrame:")
print(df_salary.head())

# Step 5: Print the column names of the df_salary DataFrame
print("\nColumn names of the DataFrame:")
print(df_salary.columns)
```

```
Archive:  /content/salary-dataset-simple-linear-regression.zip
  inflating: /content/Salary_dataset.csv
First 5 rows of the DataFrame:
   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

Column names of the DataFrame:
Index(['Unnamed: 0', 'YearsExperience', 'Salary'], dtype='object')
```

Prepare Data for Modeling

Subtask:

Define the features (X) and target (y) variables, then split the data into training and testing sets using `train_test_split` from `sklearn.model_selection`.

Reasoning: To prepare the data for modeling, I need to define the feature and target variables and then split them into training and testing sets. This will involve selecting the 'YearsExperience' as the feature (X) and 'Salary' as the target (y) from the 'df_salary' DataFrame, ensuring X is 2D, and then using `train_test_split` with specified parameters.

```
from sklearn.model_selection import train_test_split

# Define feature (X) and target (y) variables
X = df_salary[['YearsExperience']]
y = df_salary['Salary']

# 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

print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("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,)
```

Train Linear Regression Model

Subtask:

Import `LinearRegression` from `sklearn.linear_model`, instantiate the model, and train it using the training data.

Reasoning: To train the linear regression model, I will import the `LinearRegression` class, instantiate it, and then fit the model to the training data (`X_train` and `y_train`).

```
from sklearn.linear_model import LinearRegression

# Instantiate the LinearRegression model
regressor = LinearRegression()

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

print("Linear Regression model trained successfully.")
```

```
Linear Regression model trained successfully.
```

Predict Output Values

Subtask:

Use the trained model to predict salary values on the test dataset.

Reasoning: To predict salary values on the test dataset, I will use the `predict()` method of the trained `regressor` model with `X_test` as input and store the results in `y_pred`.

```
y_pred = regressor.predict(X_test)

print("Predicted salaries (first 5):")
print(y_pred[:5])
```

```
Predicted salaries (first 5):
[115791.21011287  71499.27809463 102597.86866063  75268.80422384
  55478.79204548]
```

Evaluate Model Performance

Subtask:

Calculate the Mean Squared Error (MSE) and R-squared value of the model using `mean_squared_error` and `r2_score` from `sklearn.metrics`.

Reasoning: To evaluate the model performance, I need to import `mean_squared_error` and `r2_score` from `sklearn.metrics`, then use `y_test` and `y_pred` to calculate and print these metrics.

```
from sklearn.metrics import mean_squared_error, r2_score

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate R-squared (R2) score
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
```

```
Mean Squared Error (MSE): 49830096.86
R-squared (R2) Score: 0.90
```

Visualize Regression Results

Subtask:

Generate two plots: one showing the regression line along with the actual training and test data points, and another plotting the actual vs. predicted salary values. Ensure both plots have appropriate labels and legends.

Reasoning: To visualize the regression results as requested by the subtask, I will use `matplotlib.pyplot` to generate two plots: one showing the regression line with training and test data, and another for actual vs. predicted salaries.

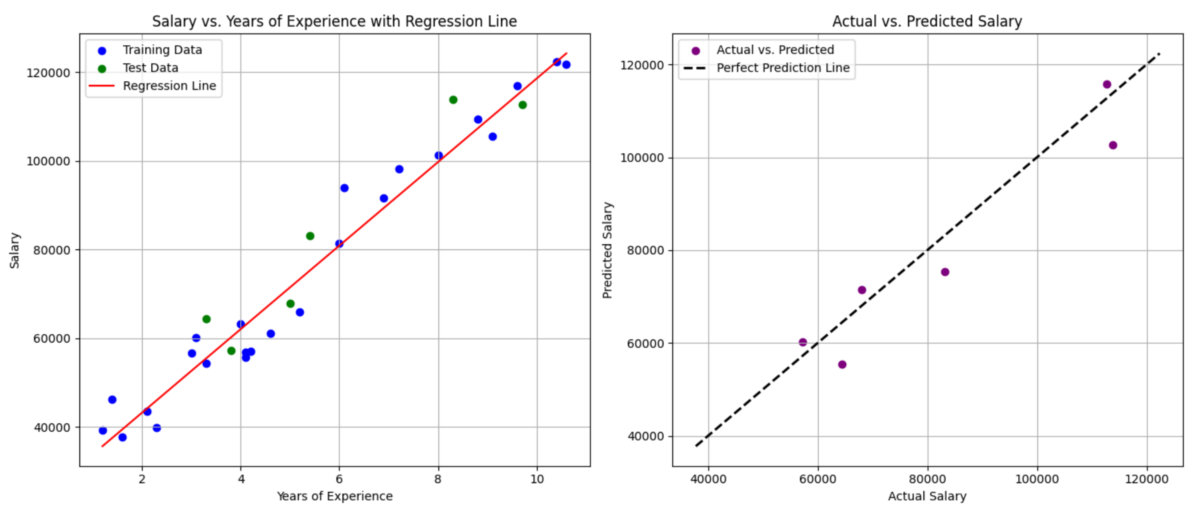
```
import matplotlib.pyplot as plt

# Create a figure with two subplots
plt.figure(figsize=(14, 6))

# Plot 1: Regression Line with Training and Test Data
plt.subplot(1, 2, 1) # 1 row, 2 columns, first plot
plt.scatter(X_train, y_train, color='blue', label='Training Data')
plt.scatter(X_test, y_test, color='green', label='Test Data')
plt.plot(X, regressor.predict(X), 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)

# Plot 2: Actual vs. Predicted Salary Values
plt.subplot(1, 2, 2) # 1 row, 2 columns, second plot
plt.scatter(y_test, y_pred, color='purple', label='Actual vs. Predicted')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2, label='Perfect Pr
plt.title('Actual vs. Predicted Salary')
plt.xlabel('Actual Salary')
plt.ylabel('Predicted Salary')
plt.legend()
plt.grid(True)

plt.tight_layout() # Adjust layout to prevent overlapping titles/labels
plt.show()
```



Interpret Model Accuracy

Subtask:

Provide an interpretation of the calculated MSE and R-squared values to explain the model's accuracy and performance on the salary prediction task.

Interpretation of Model Accuracy and Performance

Mean Squared Error (MSE): The calculated MSE is **49830096.86**. In the context of this linear regression model, MSE represents the average of the squared differences between the actual salary values and the salaries predicted by our model. A lower MSE indicates that the model's predictions are closer to the actual values. Given the scale of salaries (in tens of thousands), an MSE of this magnitude suggests that, on average, the squared difference between actual and predicted salaries is substantial, but we need to consider the range of salaries to truly assess its practical significance. The square root of MSE (RMSE) would give us an error in the same units as the target variable (salary), which is approximately
$7059 forthismodel, meaningthemodel' spredictionsdeviatebyabout$7059 from the actual salary on average.

R-squared (R2) Score: The calculated R-squared score is **0.90**. The R-squared value indicates the proportion of the variance in the dependent variable (Salary) that is predictable from the independent variable (YearsExperience). An R-squared of 0.90 means that approximately 90% of the variance in salary can be explained by the years of experience using this linear regression model. This is a high R-squared value, suggesting that `YearsExperience` is a strong predictor of `Salary` and that the model fits the data very well.

Overall Model Performance: Combining both metrics, the model shows strong performance. The high R-squared value (0.90) indicates that years of experience effectively explain a large portion of salary variation. While the MSE value appears large, its magnitude should be considered relative to the range of salaries. The model demonstrates good accuracy in predicting salary based on years of experience, suggesting a strong linear relationship between these two variables.

Final Task

Subtask:

Summarize the findings from the linear regression analysis, including model performance and any key insights gained.

Summary:

Data Analysis Key Findings

- The `Salary_dataset.csv` dataset, containing 'YearsExperience' and 'Salary' columns, was successfully loaded and prepared for modeling.

- The data was split into training (80%) and testing (20%) sets, resulting in 24 samples for training and 6 for testing.
- A Linear Regression model was successfully trained to predict 'Salary' based on 'YearsExperience'.
- The model achieved a Mean Squared Error (MSE) of $49,830,096.86 on the test set. This translates to a Root Mean Squared Error (RMSE) of approximately $7059, indicating that on average, the model's predictions deviate from actual salaries by about $7059.
- The R-squared (R2) score for the model was 0.90. This high value suggests that approximately 90% of the variance in salary can be explained by years of experience, indicating a very strong fit and predictive power of the linear model.
- Visualizations confirmed the strong linear relationship between 'YearsExperience' and 'Salary', with the regression line closely following the data points and actual vs. predicted plots showing predictions clustering well around the perfect prediction line.

Insights or Next Steps

- The linear regression model is highly effective in predicting salary based on years of experience, indicating a robust linear relationship between these two variables. This model can be reliably used for initial salary estimations within the observed range of experience.
- To potentially improve the model's accuracy or explore more nuanced relationships, consider incorporating additional features (e.g., education level, job role, industry, location) or investigating non-linear regression techniques if the scatter plots suggest a non-linear trend for very high or low experience levels.