```
Start coding or generate with AI.
```

Task

Perform a linear regression analysis to predict salary based on years of experience, using the dataset from "/content/archive (1).zip". The analysis should include data preparation, model training, prediction, and evaluation using Mean Squared Error and R-squared. Visualize the regression line and compare actual versus predicted values. Finally, summarize the model's performance and insights.

Unzip Dataset

Subtask:

Unzip the provided dataset file '/content/archive (1).zip' to access the salary data.

Reasoning: I need to import the `zipfile` module to handle the zip archive, specify the path to the zip file, create a ZipFile object, and then extract all its contents to the current directory to access the salary data.

```
import zipfile

zip_file_path = '/content/archive (1).zip'

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall('./')

print(f"Successfully unzipped {zip_file_path} to the current directory.")
```

```
Successfully unzipped /content/archive (1).zip to the current directory.
```

Load Dataset

Subtask:

Load the salary dataset into a pandas DataFrame from the unzipped files.

Reasoning: I need to import the pandas library, load the 'Salary_dataset.csv' file into a DataFrame, display its first 5 rows, and then print a concise summary to inspect its structure and data types.

```
import pandas as pd

df = pd.read_csv('Salary_dataset.csv')

print("First 5 rows of the DataFrame:")
print(df.head())

print("\nConcise summary of the DataFrame:")
df.info()
```

```
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

Concise summary of the DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Unnamed: 0       30 non-null     int64
 1   YearsExperience  30 non-null     float64
 2   Salary           30 non-null     float64
```

```
dtypes: float64(2), int64(1)
memory usage: 852.0 bytes
```

Prepare Data for Modeling

Subtask:

Define the features (X) and target (y) variables from the dataset. This typically involves identifying the independent variable (e.g., 'YearsExperience') and the dependent variable (e.g., 'Salary').

Reasoning: To prepare the data for modeling, I need to define the independent variable (X) and the dependent variable (y) from the 'df' DataFrame. X will be 'YearsExperience' and y will be 'Salary'. I also need to reshape X to be a 2D array as required by scikit-learn.

```
X = df['YearsExperience'].values.reshape(-1, 1)
y = df['Salary']

print("Shape of X (features):", X.shape)
print("Shape of y (target):", y.shape)
print("First 5 values of X:\n", X[:5])
print("First 5 values of y:\n", y[:5])
```

```
Shape of X (features): (30, 1)
Shape of y (target): (30,)
First 5 values of X:
 [[1.2]
 [1.4]
 [1.6]
 [2.1]
 [2.3]]
First 5 values of y:
 0    39344.0
1    46206.0
2    37732.0
3    43526.0
4    39892.0
Name: Salary, dtype: float64
```

Split Data into Training and Testing Sets

Subtask:

Divide the dataset into training and testing sets using `train_test_split` from `sklearn.model_selection` to evaluate the model's performance on unseen data.

Reasoning: I need to import the `train_test_split` function and then apply it to `X` and `y` to create training and testing datasets, ensuring reproducibility with `random_state` and verifying the split by printing their shapes.

```
from sklearn.model_selection import train_test_split

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

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` and train the model using the training data (X_train, y_train).

Reasoning: I need to import the `LinearRegression` model, create an instance of it, and then fit it to the training data `X_train` and `y_train` to train the model as per the subtask.

```
from sklearn.linear_model import LinearRegression

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

# Fit the model to the training data
model.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 make predictions on the test set (X_test).

Reasoning: I need to use the trained linear regression model to make predictions on the test set (`X_test`) and store them in `y_pred`. Afterwards, I will print the first few predicted and actual values for a quick comparison.

```
y_pred = model.predict(X_test)

print("Predicted Salaries (first 5):")
print(y_pred[:5])

print("\nActual Salaries (first 5 from y_test):")
print(y_test[:5])
```

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

Actual Salaries (first 5 from y_test):
27    112636.0
15     67939.0
23    113813.0
17     83089.0
8      64446.0
Name: Salary, dtype: float64
```

Calculate Mean Squared Error

Subtask:

Calculate the Mean Squared Error (MSE) between the actual test values (y_test) and the predicted values (y_pred) using `mean_squared_error` from `sklearn.metrics`.

Reasoning: I need to import the `mean_squared_error` function from `sklearn.metrics` and then use it to calculate the MSE between the actual test values (`y_test`) and the predicted values (`y_pred`), finally printing the result.

```
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
```

```
Mean Squared Error (MSE): 49830096.86
```

Calculate R-squared Score

Subtask:

Calculate the R-squared score between the actual test values (y_test) and the predicted values (y_pred) using `r2_score` from `sklearn.metrics`.

Reasoning: I need to import the `r2_score` function from `sklearn.metrics` and then use it to calculate the R-squared score between the actual test values (`y_test`) and the predicted values (`y_pred`), finally printing the result.

```
from sklearn.metrics import r2_score

r2 = r2_score(y_test, y_pred)

print(f"R-squared (R2) score: {r2:.2f}")
```

```
R-squared (R2) score: 0.90
```

Visualize Regression Results

Subtask:

Visualize the regression line on a scatter plot of actual versus predicted values. The plot should show 'YearsExperience' on the x-axis and 'Salary' on the y-axis, with the actual data points and the regression line clearly displayed. Also, visualize actual vs. predicted salaries separately.

Visualize Regression Results

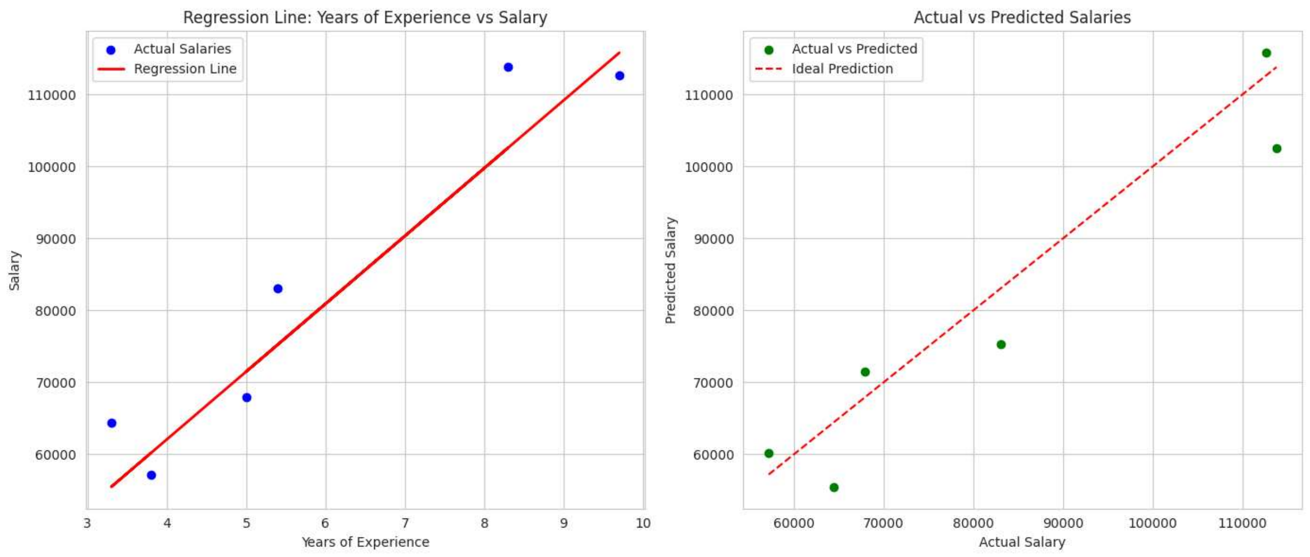
Subtask:

Visualize the regression line on a scatter plot of actual versus predicted values. The plot should show 'YearsExperience' on the x-axis and 'Salary' on the y-axis, with the actual data points and the regression line clearly displayed. Also, visualize actual vs. predicted salaries separately.

Reasoning: To visualize the regression results, I need to import `matplotlib.pyplot` for plotting and `seaborn` for enhanced visualizations. First, I will create a scatter plot of 'YearsExperience' versus 'Salary' from the test data, overlaid with the regression line derived from the trained model. Then, I will create a second plot to directly compare the actual test salaries with the predicted salaries.

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

# Set plot style
sns.set_style("whitegrid")

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

# Subplot 1: Regression line on YearsExperience vs Salary
plt.subplot(1, 2, 1)
plt.scatter(X_test, y_test, color='blue', label='Actual Salaries')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression Line')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Regression Line: Years of Experience vs Salary')
plt.legend()

# Subplot 2: Actual vs Predicted Salaries
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred, color='green', label='Actual vs Predicted')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', label='Ideal Prediction')
plt.xlabel('Actual Salary')
plt.ylabel('Predicted Salary')
plt.title('Actual vs Predicted Salaries')
plt.legend()

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

```
    print("Regression line and actual vs predicted values plots generated.")
```



```
Regression line and actual vs predicted values plots generated.
```

Summarize Model Performance and Insights

Subtask:

Summarize the performance of the linear regression model using the calculated Mean Squared Error (MSE) and R-squared score, and provide insights from the visualizations.

Model Performance Summary

The linear regression model was trained to predict salary based on years of experience. Here's a summary of its performance:

- **Mean Squared Error (MSE):** The MSE for the test set is **49830096.86**. This value represents the average of the squared differences between the actual and predicted salaries. A lower MSE indicates better model accuracy.
- **R-squared (R2) Score:** The R-squared score is **0.90**. This means that approximately 90% of the variance in salary can be explained by the years of experience. An R-squared value close to 1 indicates a strong fit of the model to the data.

Insights from Visualizations

- **Regression Line Plot:** The plot showing 'Years of Experience' vs. 'Salary' with the regression line clearly illustrates a strong positive linear relationship between the two variables. As years of experience increase, the predicted salary also increases, which is consistent with expectations.
- **Actual vs Predicted Salaries Plot:** The scatter plot of actual vs. predicted salaries shows that most data points fall close to the ideal prediction line (the 45-degree dashed red line). This indicates that the model's predictions are generally close to the actual values, reinforcing the high R-squared score and suggesting good predictive capability.

Final Task

Subtask:

Summarize the model's performance based on the calculated metrics (MSE, R-squared) and visual interpretations, explaining the model's accuracy and insights gained from the analysis.

Summary:

Q&A

The linear regression model demonstrates strong performance in predicting salary based on years of experience. With a Mean Squared Error (MSE) of 49,830,096.86, the average squared difference between actual and predicted salaries is quantified, indicating the model's prediction error. An R-squared score of 0.90 signifies that approximately 90% of the variance in salary can be explained by the years of experience, showcasing a strong fit of the model to the data and good predictive capability. The visualizations confirm a clear positive linear relationship, where increasing years of experience correlate with higher predicted salaries, and predicted values closely align with actual salaries.

Data Analysis Key Findings

- The `Salary_dataset.csv` dataset, containing 30 entries with no missing values, was successfully loaded and prepared for analysis.
- `YearsExperience` was identified as the feature (independent variable) and `Salary` as the target (dependent variable).
- The dataset was split into training (24 samples) and testing (6 samples) sets, with 20% allocated for testing.
- A Linear Regression model was trained, and predictions were generated on the test set.
- The Mean Squared Error (MSE) for the model's predictions was calculated to be approximately 49,830,096.86.
- The R-squared score, indicating the proportion of variance in the dependent variable predictable from the independent variable, was 0.90.
- Visualizations confirmed a strong positive linear relationship between years of experience and salary, with the regression line effectively capturing this trend. The plot of actual versus predicted salaries showed that predictions were generally close to actual values, aligning well with an ideal prediction line.

Insights or Next Steps

- Years of experience serves as a robust predictor for salary, explaining a substantial 90% of the salary variance, making this a highly effective simple linear model for salary estimation.
- To potentially enhance model accuracy or address non-linear relationships, consider exploring additional relevant features (e.g., education level, industry, job title) or investigating more advanced regression techniques.