```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
housing = pd.read_csv("/content/canada_per_capita_income.csv")
housing = housing.rename(columns={'per capita income': 'per_capita_income'})
housing.columns = ['year', 'per_capita_income']

housing.head(100)
housing.info()
housing.describe()
```

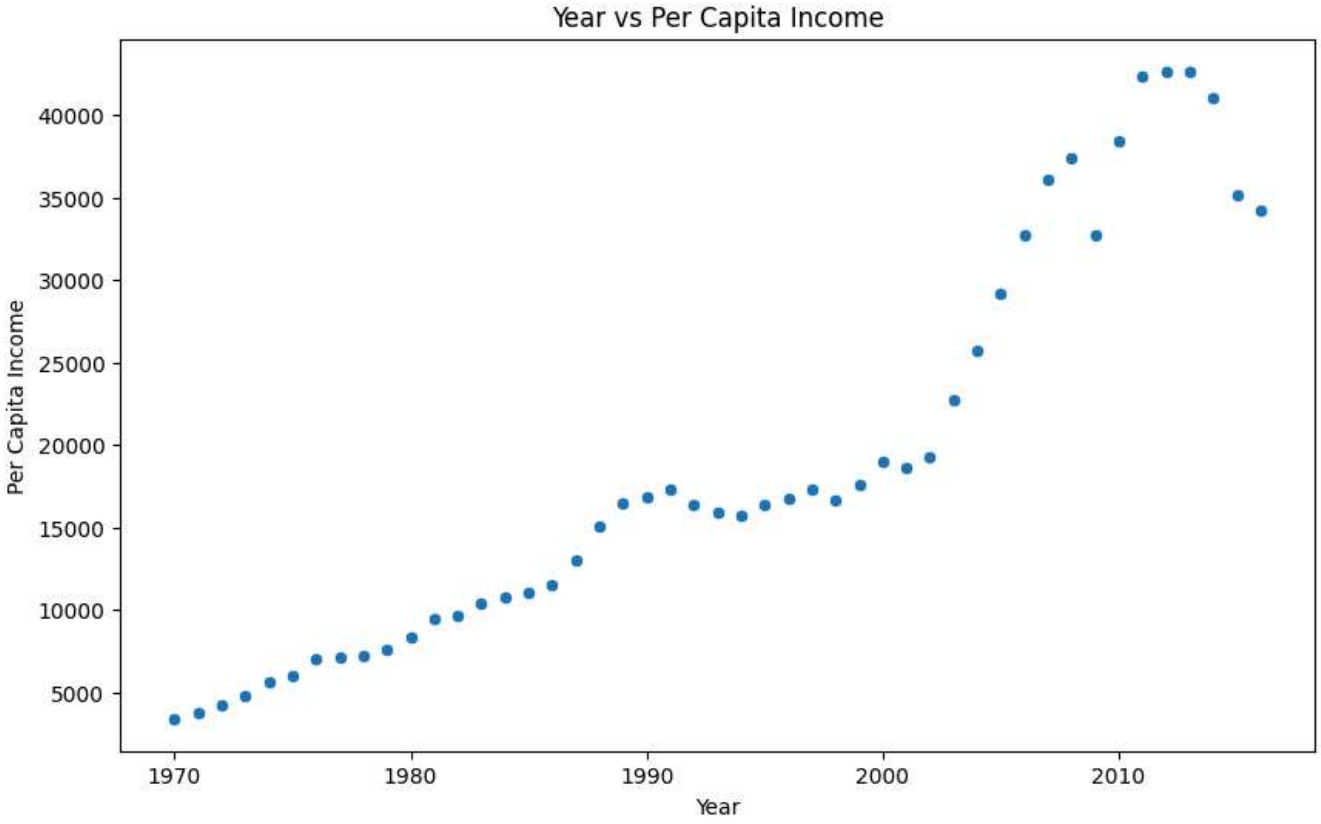
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   year               47 non-null     int64  
 1   per_capita_income  47 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 884.0 bytes
```

	year	per_capita_income
count	47.000000	47.000000
mean	1993.000000	18920.137063
std	13.711309	12034.679438
min	1970.000000	3399.299037
25%	1981.500000	9526.914515
50%	1993.000000	16426.725480
75%	2004.500000	27458.601420
max	2016.000000	42676.468370

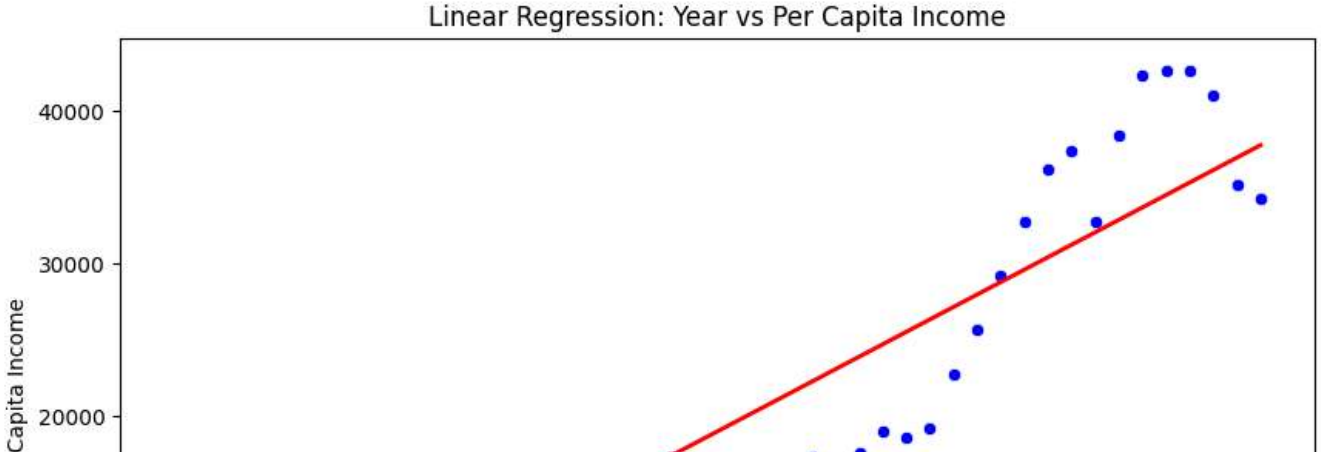
```
#canada's per capita income

# Step 1: Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Step 2: Load the dataset
# Assuming your CSV file is named 'canada_per_capita_income.csv' and it's uploaded in Colab
url = '/content/canada_per_capita_income.csv'  # Replace with actual file path or URL
data = pd.read_csv(url)

# Step 3: Rename columns for easier access (optional)
data.columns = ['year', 'per_capita_income']  # Renaming to remove spaces and special characters

# Step 4: Inspect the dataset
```

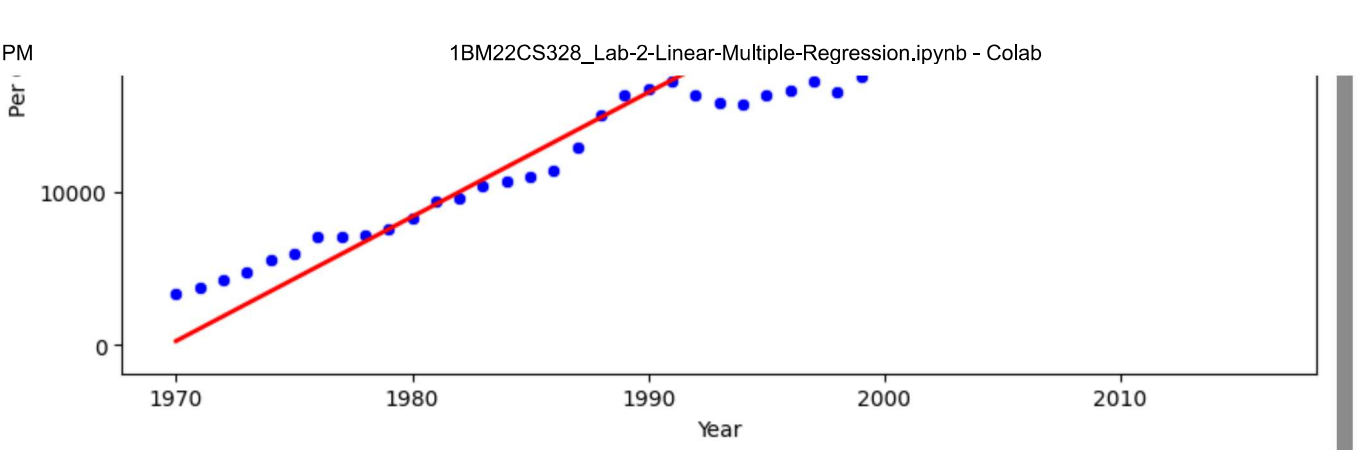
```
print(data.head())
print(data.info())

# Step 5: Visualize the data to understand the relationship between Year and Per Capita Income
plt.figure(figsize=(10, 6))
sns.scatterplot(x='year', y='per_capita_income', data=data)
plt.title('Year vs Per Capita Income')
plt.xlabel('Year')
plt.ylabel('Per Capita Income')
plt.show()

# Step 6: Prepare data for model
# 'year' as feature and 'per_capita_income' as target
X = data[['year']]  # Feature: Year
y = data['per_capita_income']  # Target: Per Capita Income

# Step 7: Split the data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 8: Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 9: Make predictions on the test set
y_pred = model.predict(X_test)

# Step 10: Evaluate the model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# Step 11: Predict per capita income for the year 2020
year_2020 = np.array([[2020]])
income_2020 = model.predict(year_2020)

print(f'Predicted Per Capita Income for the year 2020: {income_2020[0]}')

# Step 12: Visualize the linear regression model
plt.figure(figsize=(10, 6))
sns.scatterplot(x='year', y='per_capita_income', data=data, color='blue')
plt.plot(data['year'], model.predict(data[['year']]), color='red', linewidth=2)
plt.title('Linear Regression: Year vs Per Capita Income')
plt.xlabel('Year')
plt.ylabel('Per Capita Income')
plt.show()
```

```
   year  per_capita_income
0  1970        3399.299037
1  1971        3768.297935
2  1972        4251.175484
3  1973        4804.463248
4  1974        5576.514583
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   year               47 non-null     int64
 1   per_capita_income  47 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 884.0 bytes
None
```




```
Mean Squared Error: 15147815.5477862
R-squared: 0.8751771396846304
Predicted Per Capita Income for the year 2020: 41027.67748165317
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not
  warnings.warn(
```





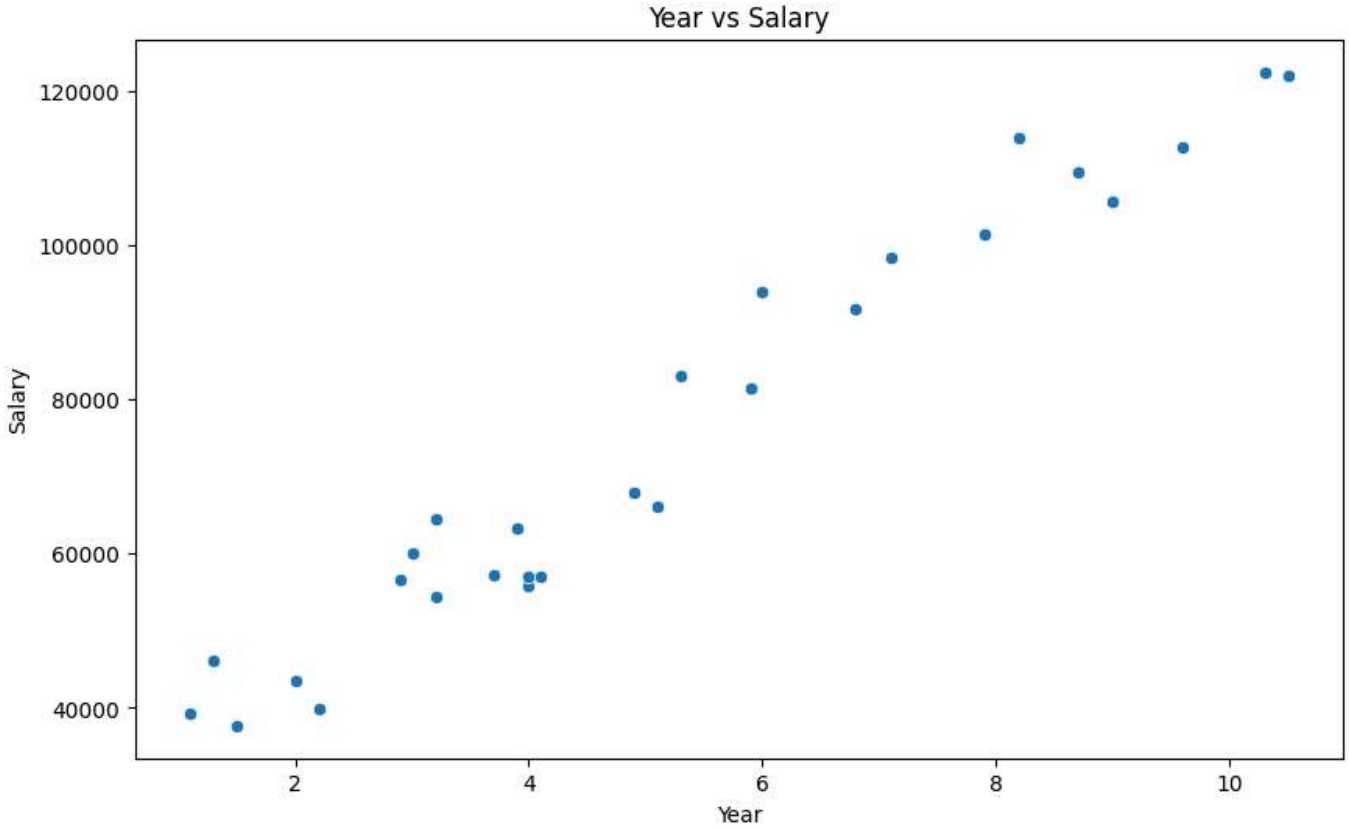

```
salary = pd.read_csv("/content/salary.csv")

salary = salary.dropna()
salary.head(100)
salary.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 28 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  28 non-null     float64
 1   Salary           28 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 672.0 bytes
```

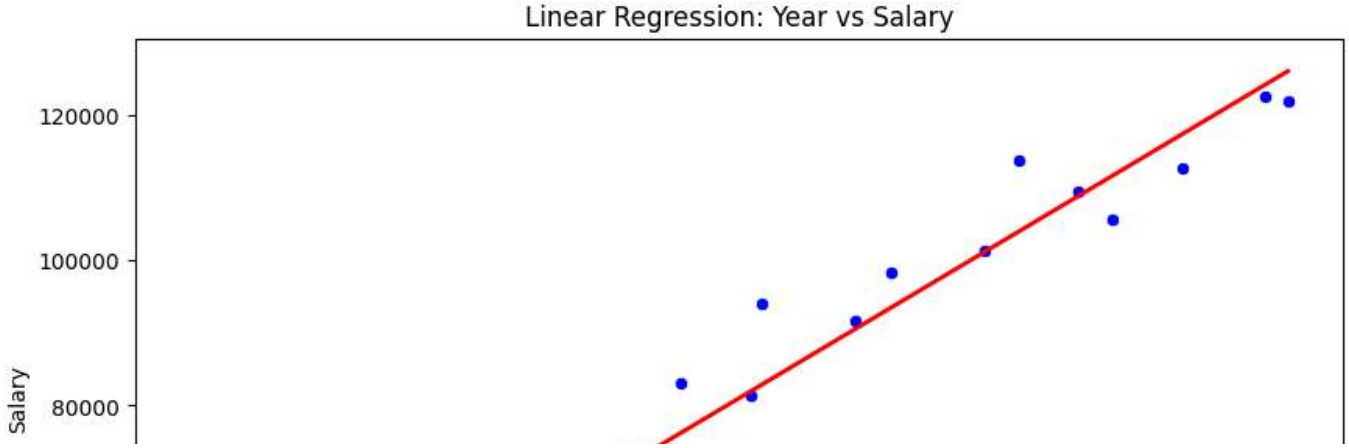
```
#Salary of the employee

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Step 1: Load the dataset
data = pd.read_csv('salary.csv')

# Step 2: Data Preprocessing
# Remove rows with missing values (if any)
data = data.dropna()

# Rename columns to remove spaces or special characters (if necessary)
data.columns = ['year', 'salary']

# Step 3: Inspect the dataset
print(data.head())
print(data.info())
```

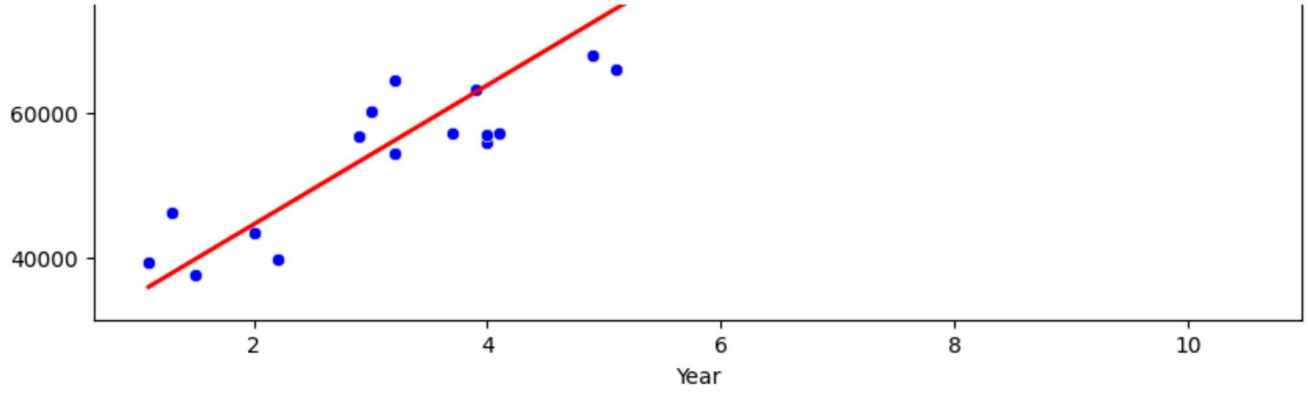
```
# Step 4: Visualize the data to understand the relationship between Year and Salary
plt.figure(figsize=(10, 6))
sns.scatterplot(x='year', y='salary', data=data)
plt.title('Year vs Salary')
plt.xlabel('Year')
plt.ylabel('Salary')
plt.show()

# Step 5: Prepare data for model
# 'year' as feature and 'salary' as target
X = data[['year']]  # Feature: Year
y = data['salary']  # Target: Salary

# Step 6: Split the data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 7: Train a linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Step 8: Make predictions on the test set
y_pred = model.predict(X_test)

# Step 9: Evaluate the model performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')

# Step 10: Predict salary for 12 years of experience
years_of_experience = np.array([[12]])  # 12 years of experience
predicted_salary = model.predict(years_of_experience)

print(f'Predicted Salary for 12 years of experience: {predicted_salary[0]}')

# Step 11: Visualize the linear regression model
plt.figure(figsize=(10, 6))
sns.scatterplot(x='year', y='salary', data=data, color='blue')
plt.plot(data['year'], model.predict(data[['year']]), color='red', linewidth=2)
plt.title('Linear Regression: Year vs Salary')
plt.xlabel('Year')
plt.ylabel('Salary')
plt.show()
```

```
   year  salary
0   1.1   39343
1   1.3   46205
2   1.5   37731
3   2.0   43525
4   2.2   39891
<class 'pandas.core.frame.DataFrame'>
Index: 28 entries, 0 to 29
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   year    28 non-null     float64
 1   salary  28 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 672.0 bytes
None
```




```
Mean Squared Error: 27180506.800821673
R-squared: 0.960019091624879
Predicted Salary for 12 years of experience: 140337.54125839565
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not ha
  warnings.warn(
```






```
hiring = pd.read_csv("/content/hiring.csv")
hiring = hiring.rename(columns={'test_score(out of 10)': 'test_score', 'interview_score(out of 10)': 'in
hiring
```

	experience	test_score	interview_score	salary
0	NaN	8.0	9	50000
1	NaN	8.0	6	45000
2	five	6.0	7	60000
3	two	10.0	10	65000
4	seven	9.0	6	70000
5	three	7.0	10	62000
6	ten	NaN	7	72000
7	eleven	7.0	8	80000

Next steps: Generate code with hiring | View recommended plots | New interactive sheet

```
#Hiring

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Step 1: Load the dataset
data = pd.read_csv('hiring.csv')

# Step 2: Data Preprocessing

# Convert the 'experience' column to numeric values:
experience_mapping = {
    'NaN': np.nan,
    'one': 1,
    'two': 2
```

```
        'two': 2,
        'three': 3,
        'four': 4,
        'five': 5,
        'six': 6,
        'seven': 7,
        'eight': 8,
        'nine': 9,
        'ten': 10,
        'eleven': 11
    }

    # Replace the string values with the corresponding numeric values
    data['experience'] = data['experience'].map(experience_mapping)
    data.columns = ['experience', 'test_score' , 'interview_score' , 'salary']
    # Check for missing values
    print("Missing values in the data:")
    print(data.isnull().sum())

    # Handle missing values:
    # Fill the missing experience values with the median of the experience column
    data['experience'] = data['experience'].fillna(data['experience'].median())

    # If the test score has missing values, fill them with the mean of the test scores
    data['test_score'] = data['test_score'].fillna(data['test_score'].mean())

    # Step 3: Inspect the data
    print("Preprocessed data:")
    print(data.head(10))

    # Step 4: Prepare data for the model
    # Features: 'experience', 'test_score', 'interview_score'
    # Target: 'salary'
    X = data[['experience', 'test_score', 'interview_score']]  # Features
    y = data['salary']  # Target

    # Step 5: Split the data into training and testing sets (80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Step 6: Train the Multiple Linear Regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Step 7: Make predictions on the test set
    y_pred = model.predict(X_test)

    # Step 8: Evaluate the model performance
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f'Mean Squared Error: {mse}')
    print(f'R-squared: {r2}')

    # Step 9: Predict salary for given candidates
    # Candidate 1: 2 years experience, 9 test score, 6 interview score
    candidate_1 = np.array([[2, 9, 6]])

    # Candidate 2: 12 years experience, 10 test score, 10 interview score
    candidate_2 = np.array([[12, 10, 10]])
```

```
# Predict the salary for both candidates
predicted_salary_1 = model.predict(candidate_1)
predicted_salary_2 = model.predict(candidate_2)

print(f'Predicted Salary for Candidate 1 (2 years experience, 9 test score, 6 interview score): {predicted
print(f'Predicted Salary for Candidate 2 (12 years experience, 10 test score, 10 interview score): {predic
```

```
Missing values in the data:
experience         2
test_score         1
interview_score    0
salary             0
dtype: int64
Preprocessed data:
   experience  test_score  interview_score  salary
0         6.0    8.000000                9   50000
1         6.0    8.000000                6   45000
2         5.0    6.000000                7   60000
3         2.0   10.000000               10   65000
4         7.0    9.000000                6   70000
5         3.0    7.000000               10   62000
6        10.0    7.857143                7   72000
7        11.0    7.000000                8   80000
Mean Squared Error: 257414883.55789393
R-squared: -2.5628357585867674
Predicted Salary for Candidate 1 (2 years experience, 9 test score, 6 interview score): 59414.229728
Predicted Salary for Candidate 2 (12 years experience, 10 test score, 10 interview score): 81409.979
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not ha
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not ha
  warnings.warn(
```

```
#1000_companies

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder

# Step 1: Load the companies dataset
df_companies = pd.read_csv('/content/1000_Companies.csv')

# Check the data to understand its structure
print(df_companies.head())

# Step 2: Preprocess the 'State' column (encode categorical values)
label_encoder = LabelEncoder()
df_companies['State'] = label_encoder.fit_transform(df_companies['State'])

# Handle missing values if there are any
df_companies = df_companies.dropna()

# Step 3: Prepare the data for regression
X_companies = df_companies[['R&D Spend', 'Administration', 'Marketing Spend', 'State']]  # Features
y_companies = df_companies['Profit']  # Target variable: Profit
```

```
# Step 4: Create and train the regression model
reg_companies = linear_model.LinearRegression()
reg_companies.fit(X_companies, y_companies)

# Step 5: Model coefficients and intercept
print(f"Model Coefficients: {reg_companies.coef_}")
print(f"Model Intercept: {reg_companies.intercept_}")

# Step 6: Predict profit for a new company
# New company data: 91694.48 R&D Spend, 515841.3 Administration, 11931.24 Marketing Spend, Florida State
new_company_data = np.array([[91694.48, 515841.3, 11931.24, label_encoder.transform(['Florida'])[0]]])
predicted_profit = reg_companies.predict(new_company_data)
print(f"Predicted profit for the new company: ${predicted_profit[0]:.2f}")
```

```
   R&D Spend  Administration  Marketing Spend       State     Profit
0  165349.20       136897.80        471784.10    New York  192261.83
1  162597.70       151377.59        443898.53  California  191792.06
2  153441.51       101145.55        407934.54     Florida  191050.39
3  144372.41       118671.85        383199.62    New York  182901.99
4  142107.34        91391.77        366168.42     Florida  166187.94
Model Coefficients: [ 0.55389023  1.02672443  0.08058525 46.62387015]
Model Intercept: -70214.44175560221
Predicted profit for the new company: $511209.20
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not ha
  warnings.warn(
```

Start coding or generate with AI.