```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Load the dataset
dataset = pd.read_csv(r"C:\Users\A3MAX SOFTWARE TECH\Desktop\WORK\1. KODI
WORK\1. NARESH\1. MORNING BATCH\N_Batch -- 8.30AM _ AUG25\3. Apr' 25\21st-
SLR\SIMPLE LINEAR REGRESSION\Salary_Data.csv")
# Check the shape of the dataset
print("Dataset Shape:", dataset.shape)  # (30, 2)

# Feature selection (independent variable X and dependent variable y)
x = dataset.iloc[:, :-1]  # Years of experience (Independent variable)
y = dataset.iloc[:, -1]   # Salary (Dependent variable)

# Split the dataset into training and testing sets (80% training, 20% testing)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.20,random_state=0)

# Reshape x_train and x_test into 2D arrays if they are single feature columns
x_train = x_train.values.reshape(-1, 1)
x_test = x_test.values.reshape(-1, 1)

# You don't need to reshape y_train, as it's the target variable
# Fit the Linear Regression model to the training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train, y_train)

# Predicting the results for the test set
y_pred = regressor.predict(x_test)

# Visualizing the Training set results
plt.scatter(x_train, y_train, color = 'red')  # Real salary data (training)
plt.plot(x_train, regressor.predict(x_train), color = 'blue')  # Predicted
regression line
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

# Visualizing the Test set results
plt.scatter(x_test, y_test, color = 'red')  # Real salary data (testing)
plt.plot(x_train, regressor.predict(x_train), color = 'blue')  # Regression line
from training set
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

# Optional: Output the coefficients of the linear model
print(f"Intercept: {regressor.intercept_}")
print(f"Coefficient: {regressor.coef_}")


bias = regressor.score(x_train, y_train)
print(bias)

variance = regressor.score(x_test,y_test)
print(variance)
```

```
# stats for ml


# Compare predicted and actual salaries from the test set
comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(comparison)

#STATISTICS FOR MACHINE LEARNING

dataset.mean()

dataset['Salary'].mean()

dataset.median()

dataset['Salary'].mode()

dataset.describe()

dataset.var()

dataset.std()

dataset.corr()


# ssr
y_mean = np.mean(y)
SSR = np.sum((y_pred-y_mean)**2)
print(SSR)

#sse
y = y[0:6]
SSE = np.sum((y-y_pred)**2)
print(SSE)

#sst
mean_total = np.mean(dataset.values) # here df.to_numpy()will convert pandas
Dataframe to Nump
SST = np.sum((dataset.values-mean_total)**2)
print(SST)

#r2
r_square = 1 - SSR/SST
print(r_square)


bias = regressor.score(x_train, y_train)
print(bias)

variance = regressor.score(x_test, y_test)
print(variance)
```