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

# Load the dataset
dataset = pd.read_csv(r'D:\Samsom - All Data\Naresh IT Institute\New folder\Salary_Data.csv')

# Check the shape of the dataset
print("Dataset Shape:", dataset.shape) # (30, 2)

# Feature selection (independent variable x and dependent variable)
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)
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=0)

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

# Predicting the results for the test set
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

```
y_pred = regressor.predict(x_test)

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

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

m_slope = regressor.coef_
print(m_slope)

c_intercept = regressor.intercept_
print(c_intercept)

y_12 = m_slope*12+c_intercept
print(y_12)

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

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

```
# statistic concept need to add on the code


dataset.mean() # this will give mean of entire dataframe


dataset['Salary'].mean() # this will give us mean of that particular column


# Median
dataset.median() # this will give median of entire dataframe


dataset['Salary'].median() # this will give us median of that particular column


# Mode
dataset['Salary'].mode() # this will give us mode of that particular column


# Variance


dataset.var() # this will give variance of entire dataframe


dataset['Salary'].var() # this give us variance of that particular column


# Standard deviation
dataset.std() # this will give standard deviation of entire dataframe


dataset['Salary'].std() # this will give us standard deviation of that particular column


# Coefficient of variation(cv)
# for calculating cv we have to import a library  first
from scipy.stats import variation
variation(dataset.values) # this will give cv of entire dataframe

variation(dataset['Salary']) # this will give us cv of that particular column

# Correlation
dataset.corr() # this will give correlation of entire dataframe

dataset['Salary'].corr(dataset['YearsExperience']) # this will give us correlation between these

# Skewness
dataset.skew() # this will give skewness of entire dataframe

dataset['Salary'].skew() # this will give us skewness of that particular column

# Standard Error
dataset.sem() # this will give standard error of entire dataframe

dataset['Salary'].sem() # this will give us standard error of that particular column

# Z-score
# for calculating Z-score we have to import a library first
import scipy.stats as stats

dataset.apply(stats.zscore) # this will give Z-score of entire dataframe
```

```
stats.zscore(dataset['Salary']) # this will give us Z-score of that particular column

# Degree of Freedom
a = dataset.shape[0] # this will gives us no.of rows
b = dataset.shape[1] # this will give us no.of columns

degree_of_freedom = a-b
print(degree_of_freedom) # this will give us degree of freedom for entire dataset

# sum of squer regresso (SSR)
#First we have to separate dependent and independent variables
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_nump()will
SST = np.sum((dataset.values-mean_total)**2)
print(SST)

# R2 SQUER
r_square = 1 - (SSR/SST)
r_square
```

```
# Save the trained model to disk
import pickle
filename = 'linear_regression_model.pk1'
with open(filename, 'wb') as file:
    pickle.dump(regressor, file)
print("Model has been pickled and saved as linear_regression_model.pk1")


import os
print(os.getcwd())
```