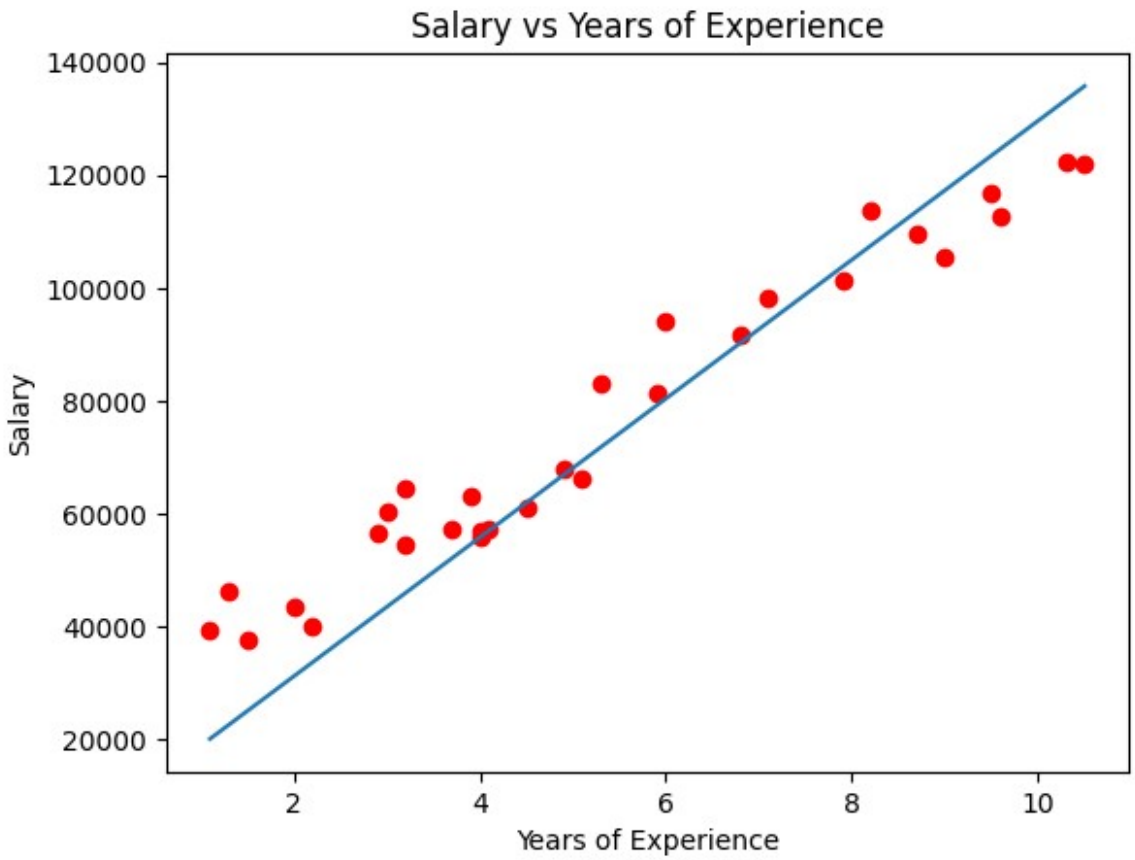
1. A program for linear regression model using ML

```
#Dataset
import pandas as pd
df = pd.read_csv('../salary_data.csv')

df.head()
```

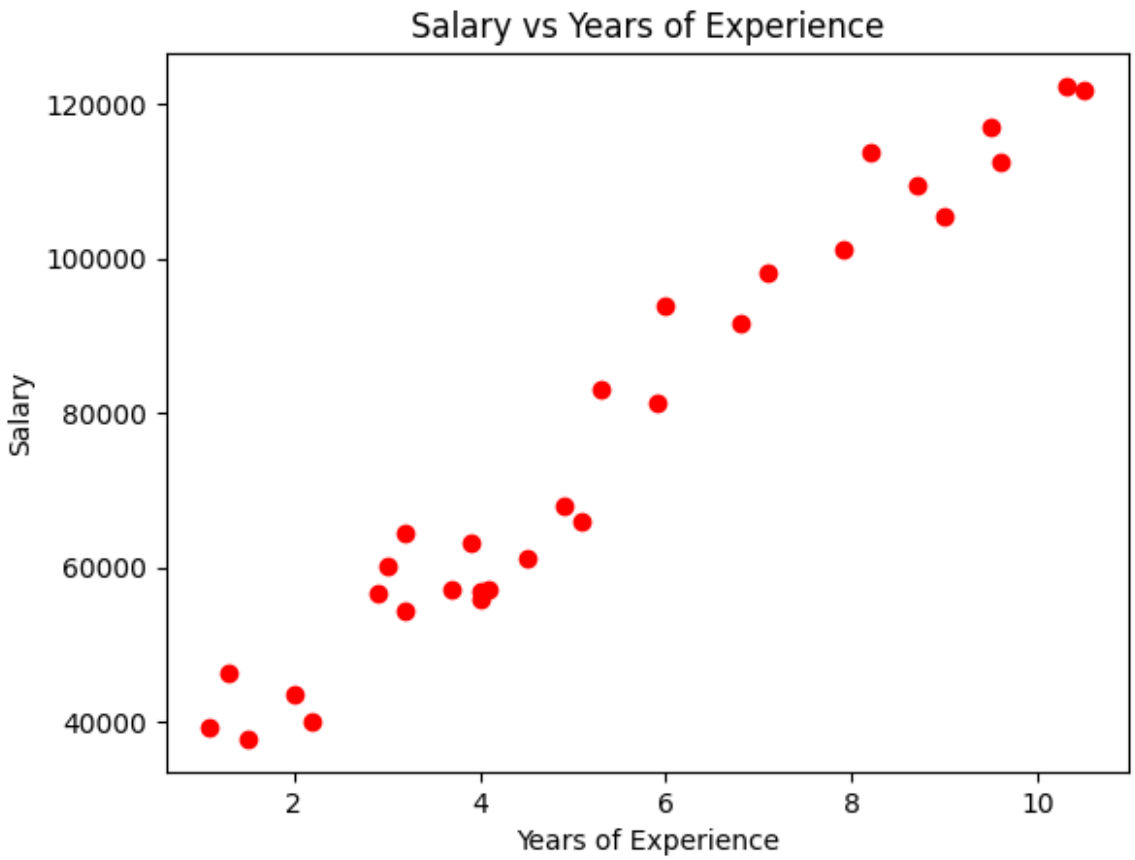
	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
import numpy as np

class SimpleLinearRegression:
    def __init__(self):
        self.Theta = None
    def fit(self,X,y,theta = [],learning_rate = 1e-2,iterations =
100):
        m,n = X.shape
        X_bias = np.c_[np.ones((m,1)),X]
        theta = np.zeros(n+1)
        inverse_X = X_bias.T
        for i in range(iterations):
            predictions = np.dot(X_bias,theta)
            errors = predictions - y
            gradient = (1 / m) * X_bias.T.dot(errors)
            theta -= learning_rate * gradient
        self.Theta = theta

    def predict(self,X):
        return np.dot(np.c_[np.ones((len(X),1)),X],self.Theta) if
self.Theta is not None else "Use fit method first"


SLP = SimpleLinearRegression()
SLP.fit(X,Y)
Y_pred = SLP.predict(X)

plt.plot(X,Y_pred)
plt.scatter(X,Y, color='red')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Salary vs Years of Experience')
plt.show()
```

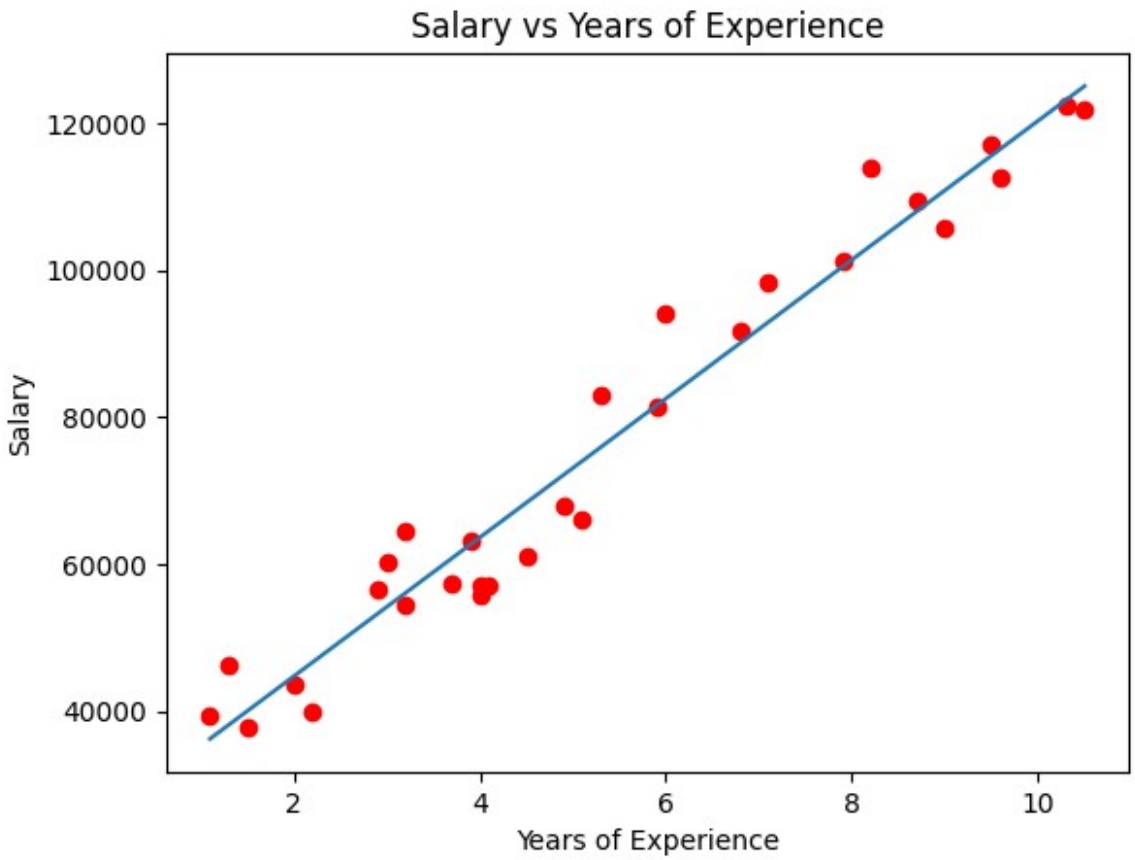


2. A Program for Linear Regression model using scikit-learn but no machine learning

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

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

reg = LinearRegression()
reg.fit(X, Y)

Y_pred = reg.predict(X)

import matplotlib.pyplot as plt
plt.scatter(X,Y, color='red')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Salary vs Years of Experience')
plt.show()
```



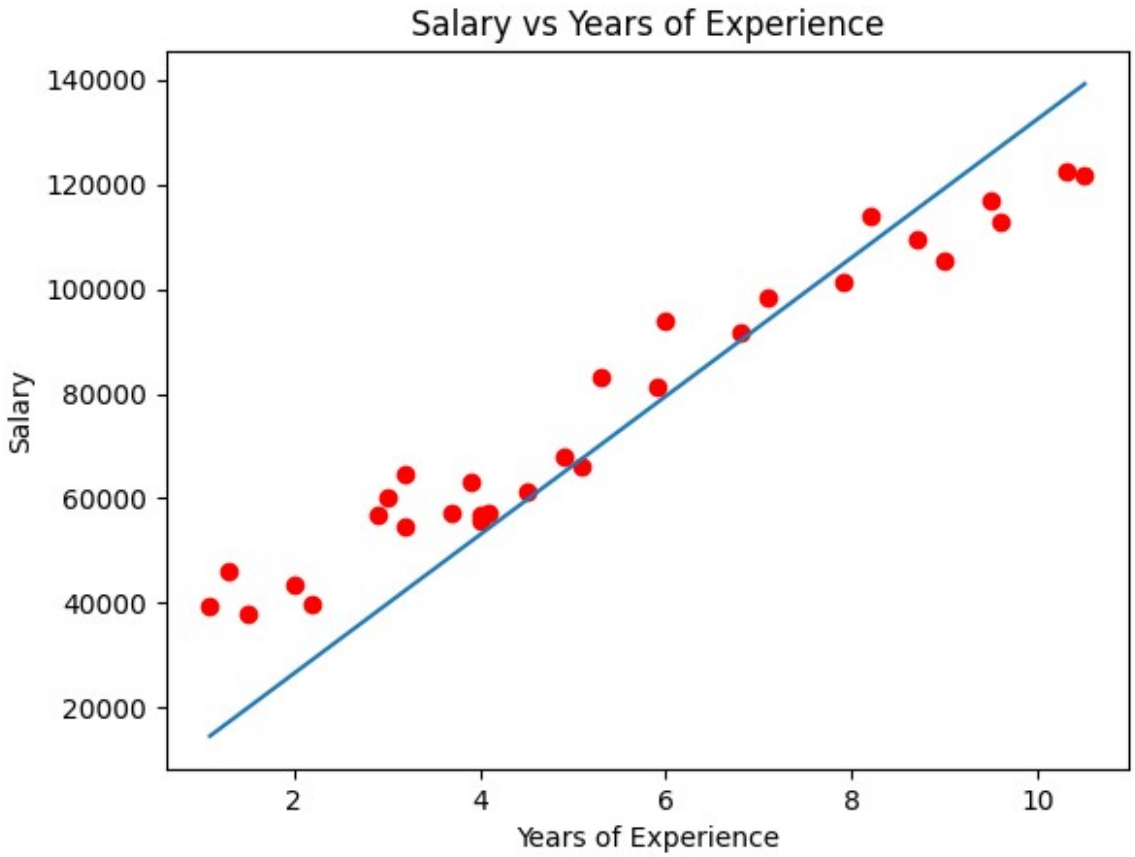

```
import matplotlib.pyplot as plt
plt.plot(X,Y_pred)
plt.scatter(X,Y, color='red')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Salary vs Years of Experience')
plt.show()
```



3. A program without using scikit-learn or machine learning for Linear regression

```
class OLS:
    def __init__(self):
        self.OLS = None

    def fit(self,X,y):
        inverse_X = X.T
        product = np.linalg.inv(np.matmul(inverse_X,X))
        prod = np.matmul(product,inverse_X)
        self.OLS = np.matmul(prod,y)

    def predict(self,X):
        return np.matmul(X,self.OLS) if self.OLS is not None else "Use
fit first"

model = OLS()
model.fit(X,Y)
Y_pred = model.predict(X)

plt.plot(X,Y_pred)
plt.scatter(X,Y, color='red')
```

```
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Salary vs Years of Experience')
plt.show()
```



What do the Coefficients of Regression signify? Research and Answer in not more than 100 words.

The coefficients in a regression model tell us how much the dependent variable (what we're trying to predict) is expected to change when an independent variable (one of the predictors) changes by one unit. For example, if we're looking at the impact of study hours on test scores, the coefficient for study hours would show how much the test score is expected to increase for each additional hour of studying, assuming other factors remain the same. The intercept is the predicted value when all predictors are zero.