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

import necessary libraries and packages

```
[1] import  numpy as np
```

```
[2] from sklearn.linear_model import LinearRegression
```

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

Dataset of Student's Regular on average Reading time in Hour(s) against his Exam Performance

```
[19] # X = np.array( [ [1], [2], [3], [4], [5] ])
     X = np.array ( [ [2.4 ], [5.5], [4.1], [7.3] ])
```

```
[20] # y = np.array ( [ [35], [45], [55], [65], [80] ] )
     y = np.array ([ [ 24 ], [41], [ 33 ], [68]   ] )
```

OVERFITTING Dataset of Student's Regular on average Reading time in Hour(s) against his Exam Performance

```
# X = np.array( [ [2], [3], [4], [5], [6] ,[7], [8], [9], [10], [11], [12], [13], [14], [15] ])
# y = np.array ( [  [4], [6], [8], [10], [12], [14], [16], [18], [20], [22], [24], [26], [28], [30] ] )
```

UNDERFITTING Dataset of Student's Regular on average Reading time in Hour(s) against his Exam Performance

```
# X = np.array( [ [2], [3], [4], [5], [6] ,[7], [8], [9], [10], [11], [12], [13], [14], [15] ])
# y = np.array ( [  [5435345], [4564566], [4564568], [56765], [67867], [867867], [65756], [567], [8768678], [6786], [67867867], [678678], [7689768678678], [89879789879] ] )
```

initialize the Liner Regression Model

```
model = LinearRegression()
```

Train the model with student's on average Hourly Reading time against his/her result

```
model.fit(X,y)
```

```
▾ LinearRegression
LinearRegression()
```

Making the predictions

```
y_pred = model.predict(X)
```

```
print(y_pred)
```

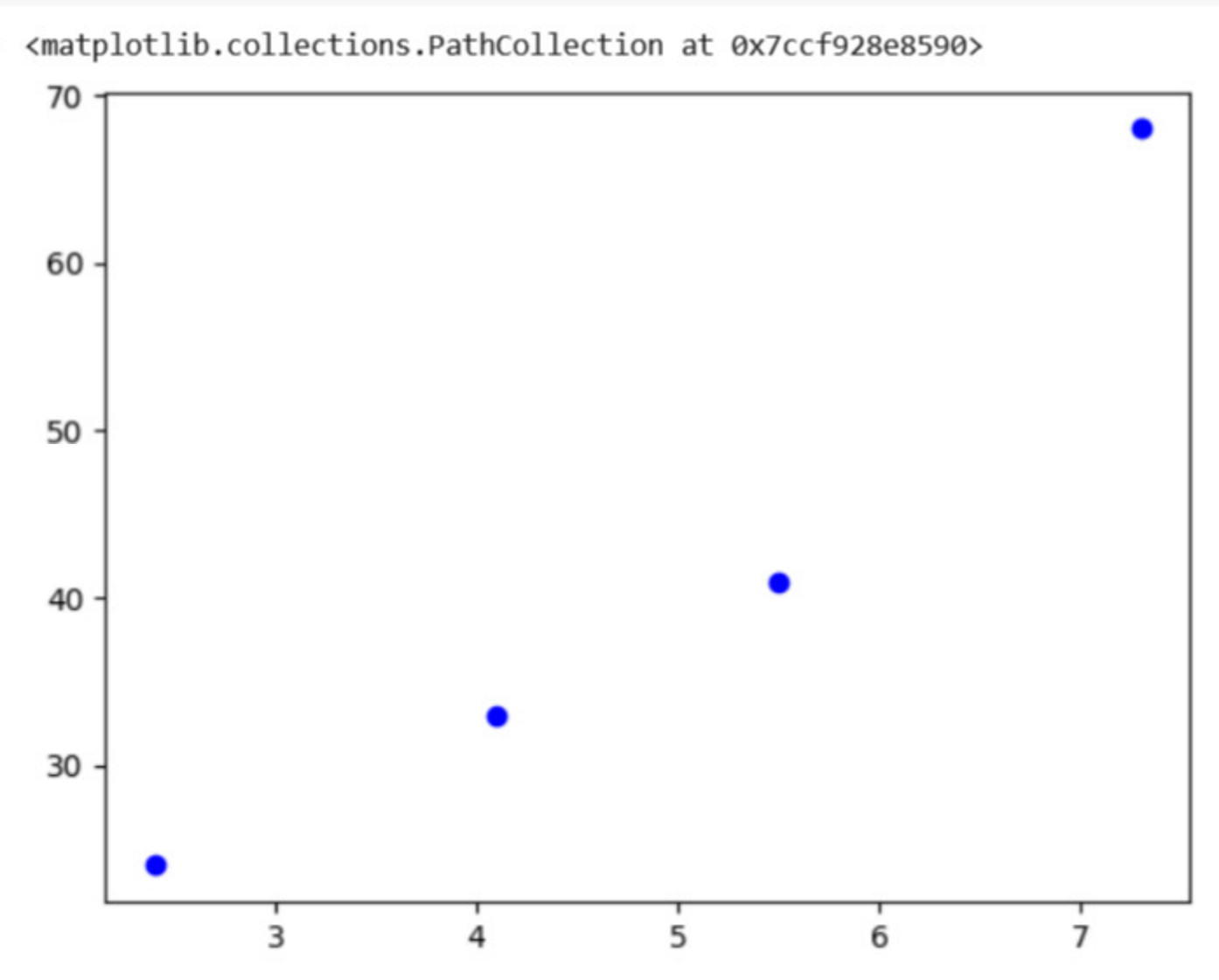
```
[[20.24215592]
 [47.41713186]
 [35.14456208]
 [63.19615014]]
```

plot for the linear Regression with the custom Student data

```
import matplotlib.pyplot as plt
```

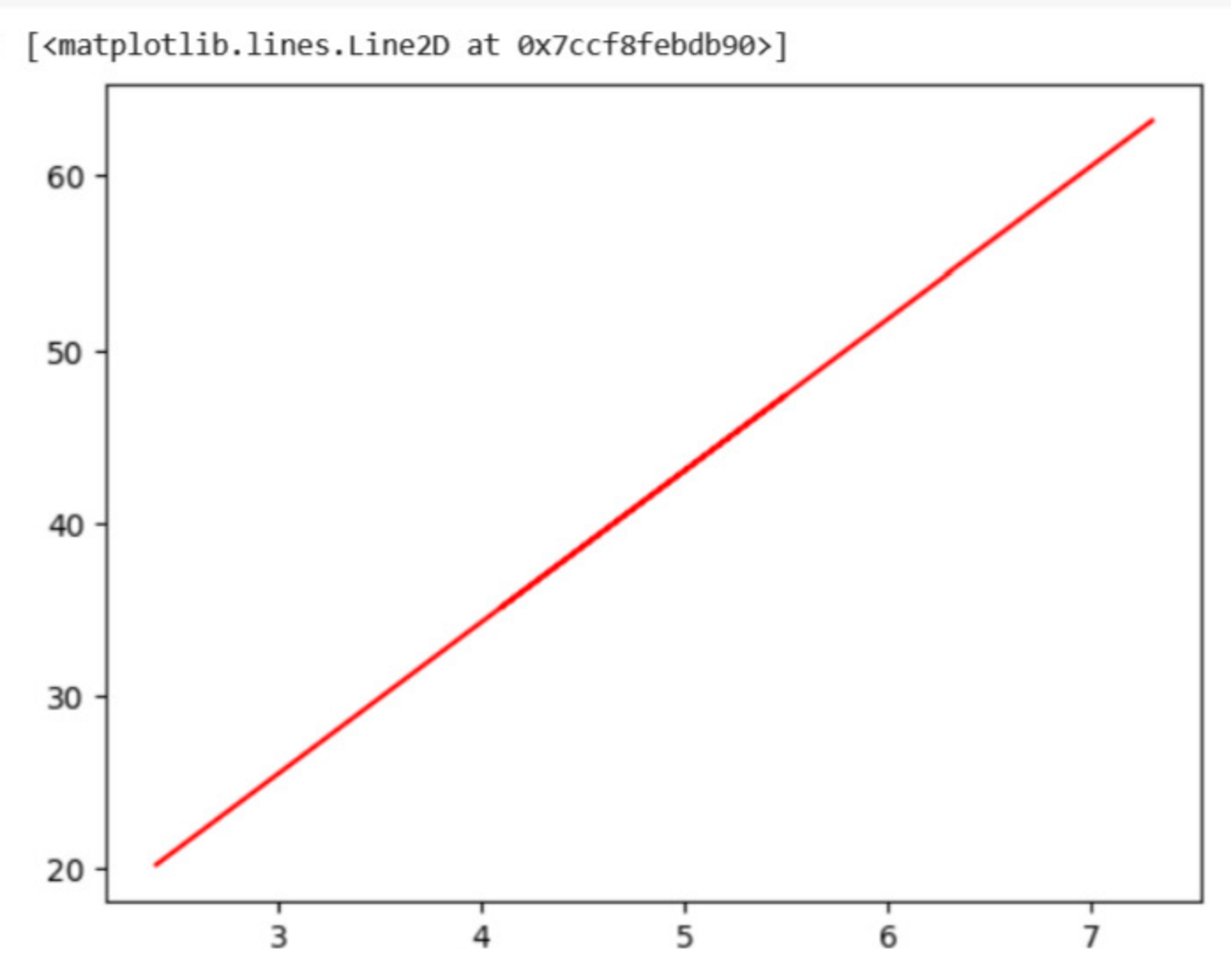
```
plt.scatter(X, y, color='blue', label= 'original data')
```

```
<matplotlib.collections.PathCollection at 0x7ccf928e8590>
```

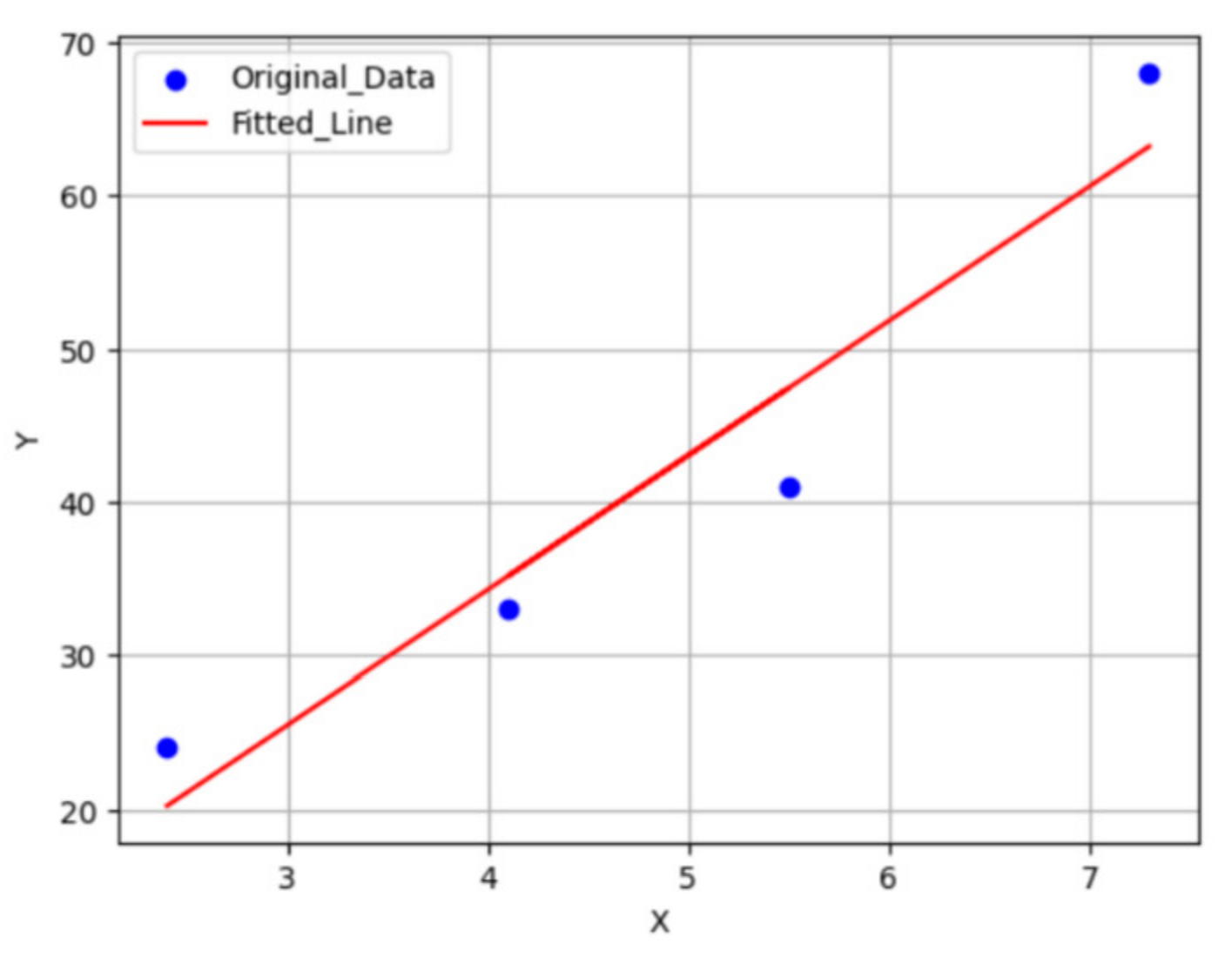


```
plt.plot(X, y_pred, color= 'red', label= 'fitted Line')
```

```
[<matplotlib.lines.Line2D at 0x7ccf8febdb90>]
```



```
plt.scatter(X, y, color='blue', label= 'Original_Data')
plt.plot(X, y_pred, color='red', label = 'Fitted_Line')
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.grid(True)
plt.show()
```




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

```
model.predict([[48]])
```

```
array([[419.97728585]])
```

In Depth *Mathmatical* Explanation behind the scene

```
X = np.array( [ [1], [2], [3], [4], [5] ])
y = np.array ( [ [35], [45], [55], [65], [80] ] )
```

```
X_mean =np.mean(X)
y_mean = np.mean(y)
```

```
numerator = np.sum( (X - X_mean) * (y - y_mean) )
denominator = np.sum( (X - X_mean) **2)
```

```
m = numerator/denominator
b = y_mean - m * X_mean
```

```
print('slope(m)', m)
print('intercept(b)', b)
```

```
slope(m) 11.0
intercept(b) 23.0
```

```
X_new = 7
y_pred = m* X_new + b
print('predicted value: ', y_pred)
```

```
predicted value:  100.0
```

next topic >>

Double-click (or enter) to edit

Double-click (or enter) to edit

Ridge Regression (L2 Penalty)

```
from sklearn.linear_model import Ridge
```

Dataset

```
# X = np.array( [ [1], [2], [3], [4], [5] ])
# y = np.array ( [ [35], [45], [55], [65], [80] ] )

X = np.array( [ [2], [3], [4], [5], [6] ,[7], [8], [9], [10], [11], [12], [13], [14], [15] ])
y = np.array ( [  [4], [6], [8], [10], [12], [14], [16], [18], [20], [22], [24], [26], [28], [30] ] )
```

initialize the Ridge Regression Model

```
ridge = Ridge()
```

fit the data with the Ridge Regression Model

```
ridge.fit(X, y)
```

```
▾ Ridge
Ridge()
```

predict for new value using trained Ridge Regression Model

```
y_predict_ridge = ridge.predict([[6.5]])
print(y_predict_ridge)

test_data = np.array( [ [6.5], [7.6], [8.9], [11.4] , [34.7] ] )
y_predict_ridge_new = ridge.predict(test_data)
print(y_predict_ridge_new)
```

```
[13.01750547]
[13.01750547 15.20787746 17.79649891 22.77461707 69.17067834]
```

Lasso Regression (L1 Penalty)

```
from sklearn.linear_model import Lasso
```

initialize Lasso (l1) model

```
lasso = Lasso()
```

fit the model with data

```
lasso.fit(X, y)
```

```
▾ Lasso
Lasso()
```

predict for new value with trained model

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

Double-click (or enter) to edit

Let's Visualize the Ridge and Lasso Model together

```
y_predict_lasso = lasso.predict([[6.5]])
print(y_predict_lasso)

test_data = np.array( [ [6.5], [7.6], [8.9], [11.4] , [34.7] ] )
y_predict_lasso_new = lasso.predict(test_data)
print(y_predict_lasso_new)
```

```
[13.12307692]
[13.12307692 15.25538462 17.77538462 22.62153846 67.78769231]
```

Double-click (or enter) to edit

Comparison of L2 VS L1 with same test data :

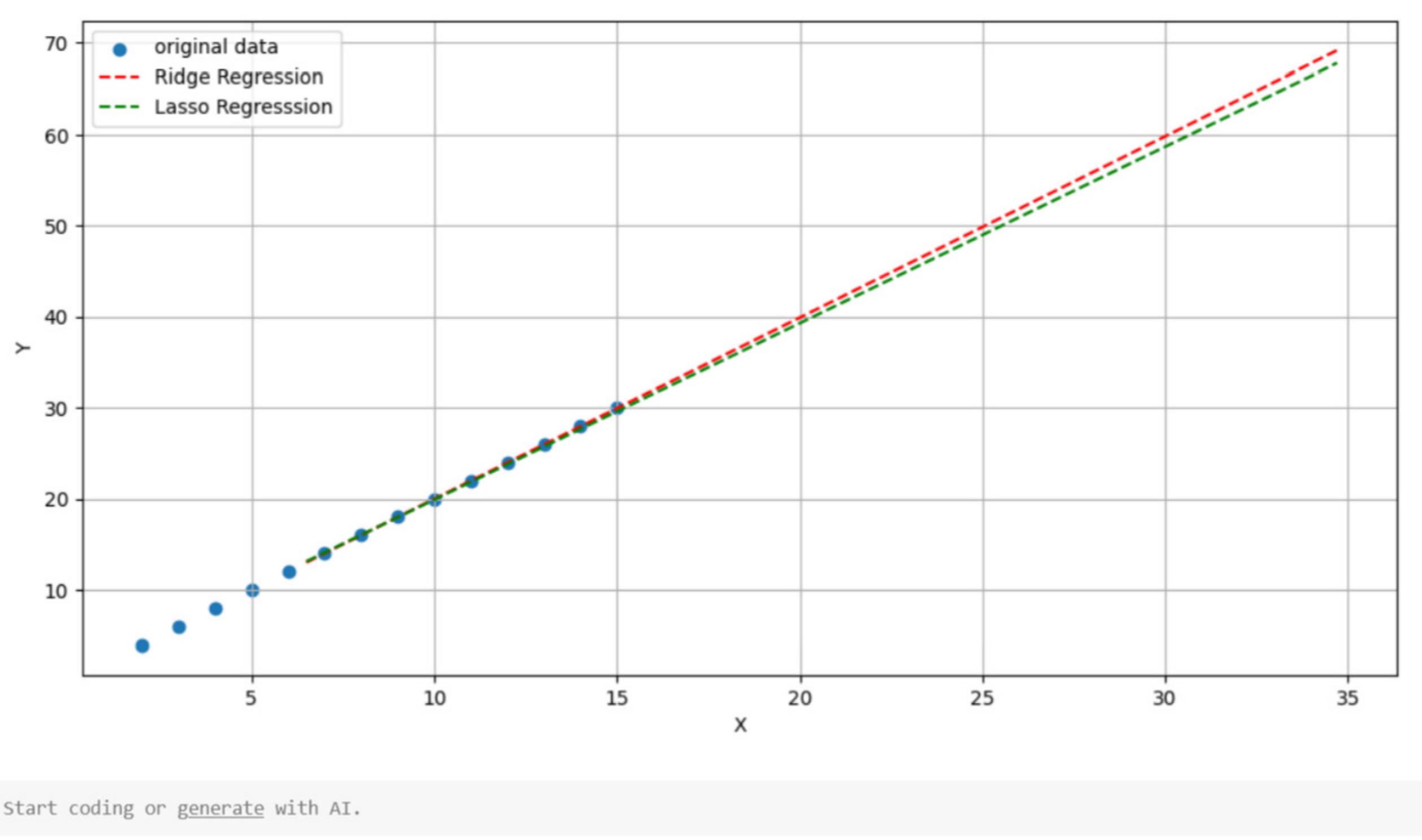
[91.] [91. 102. 115. 140. 373.]

[92.75] [92.75 104.3 117.95 144.2 388.85]

Double-click (or enter) to edit

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

```
plt.figure(figsize=(12, 6))
plt.scatter(X, y, label="original data")
plt.plot(test_data,y_predict_ridge_new, 'r--',  label= 'Ridge Regression')
plt.plot(test_data, y_predict_lasso_new, 'g--',  label = 'Lasso Regresssion')
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.grid(True)
plt.show()
```




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