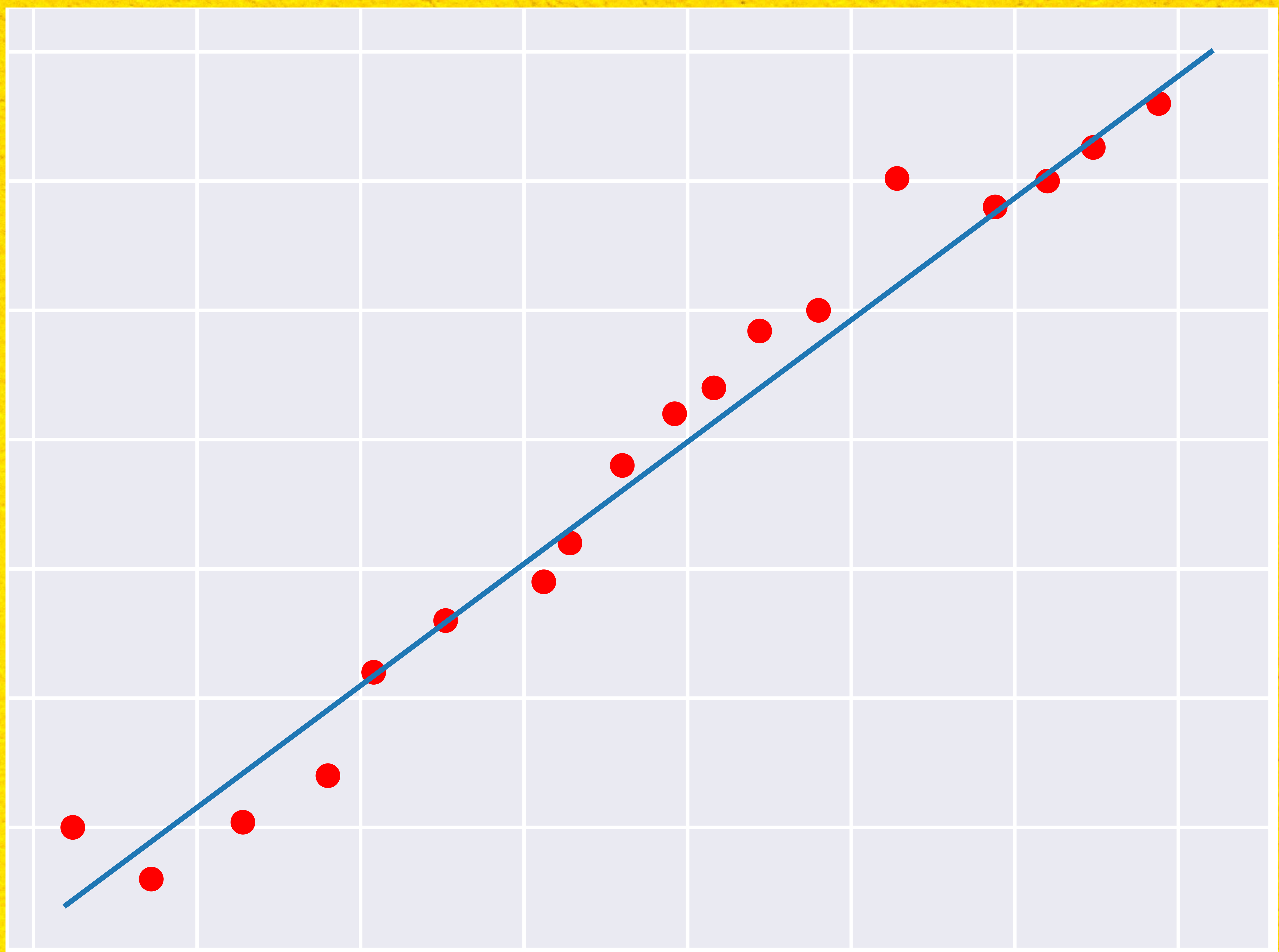
01

SIMPLE INTRO ON LINEAR REGRESSION

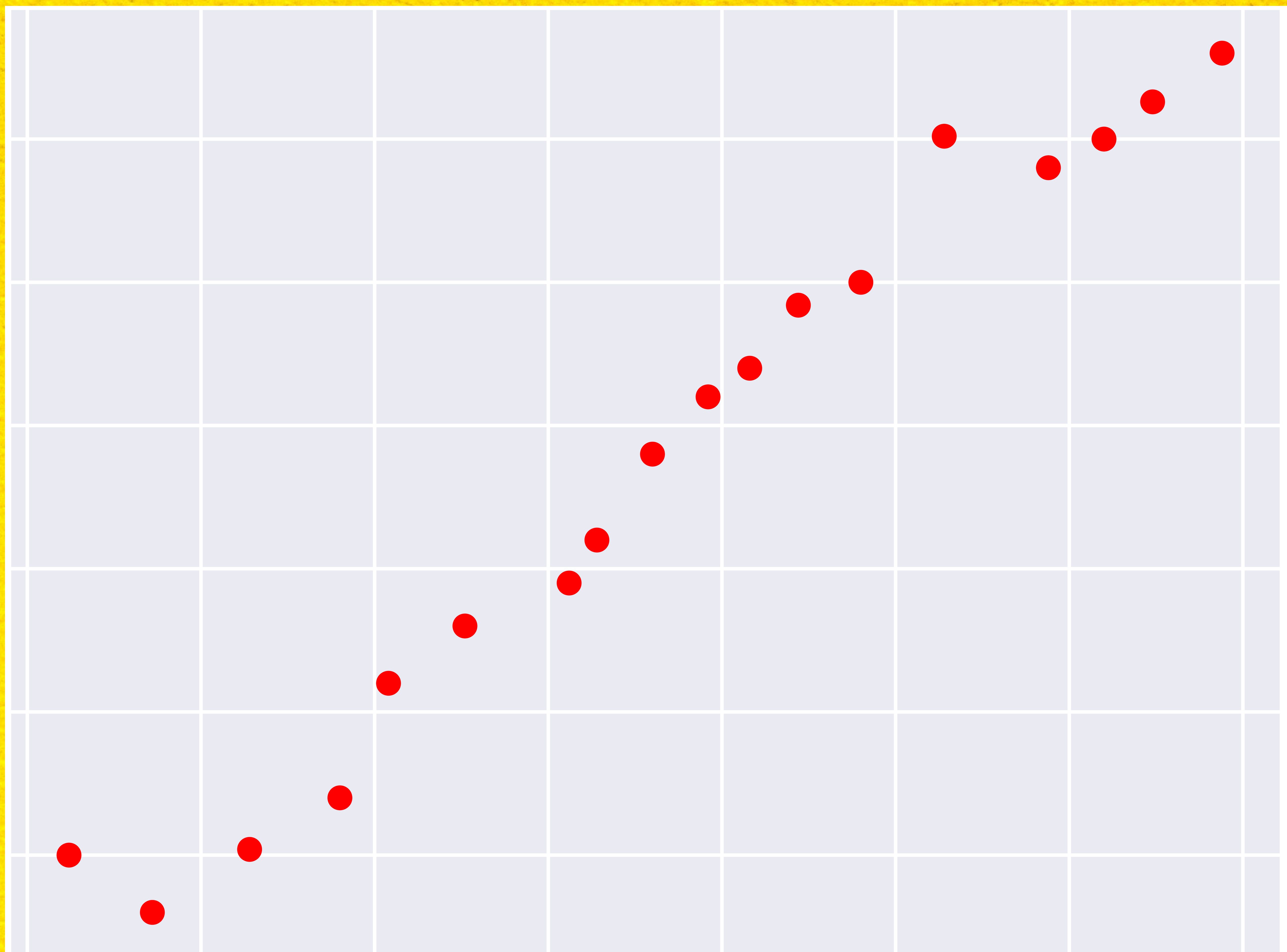
@learn.machinelearning

code >>>



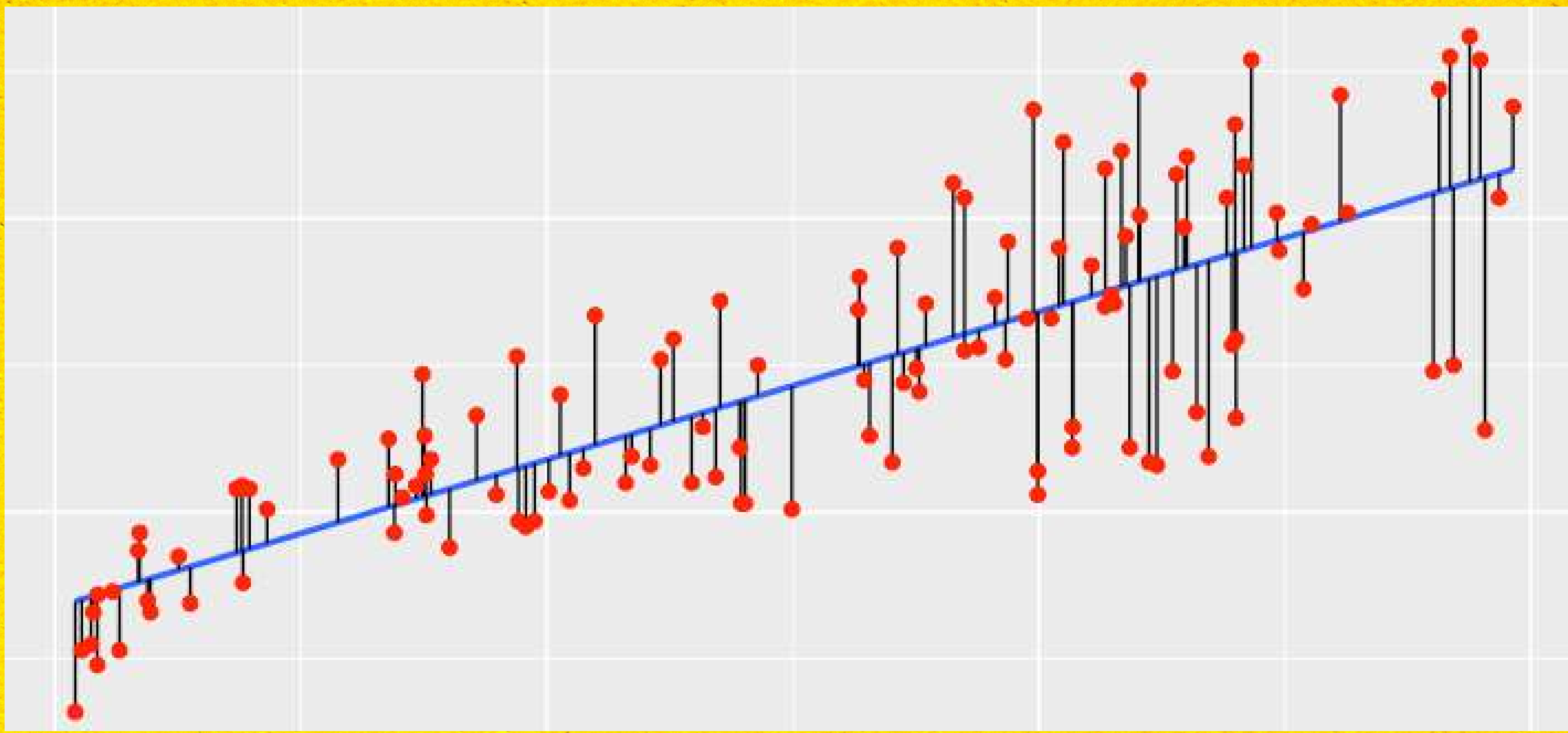
WHAT IS LINEAR REGRESSION?

- This algorithm is used to find the relationship between 2 variables(dependent and independent)
- It is a linear model which assumes a linear relationship between input and output variables.
- If we have single input variable then we call it as simple linear regression, If we have multiple we call it as multiple linear regression.



GOAL OF LINEAR REGRESSION

- The core idea is to obtain a line that best fits the data.
- And the equation of the line is (simple linear regre)
    - Y = m*X + b
- where y is the output variable we want to predict.
- x is the input variable and m & b are coefficients that we need to estimate.
- m = slope and b = bias.



HOW DOES IT WORK??

- Ultimate goal is to find the best fit line which minimizes the error (distance between the line and the data point)
- The values m and b must be chosen so that they minimize the error.
- So the algorithm will try multiple m and b values and calculates the error.
- Finally it takes the best m and b which has low error

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

HOW TO CALCULATE ERROR?

- We can calculate the error/loss in multiple ways like by using mean squared error formula as loss function.
  - MSE = (1/n) sum(yi - yi^)^2
  - n = total no of samples
  - y = actual value
  - yi = predicted value
- This helps us to evaluate the performace of the model.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

HOW TO ESTIMATE COEFFICIENTS?

- We have multiple methods to find m and b values like statistical methods or ordinary least squares or gradient descent.
- We will be using gradient descent where it starts with random m and b values and at every iteration we will calculate the loss and based on the loss it adjusts the m and b values and finally it stops when it converges and gives the best m and b values which gives low error rate.
- Check my previous posts to understad gradient descent.

ADVANTAGES

- Linear Regression is simple to implement and easier to interpret the output coefficients.
- High Performance on linearly seperable datasets
- Linear Regression is susceptible to over-fitting but it can be avoided using some dimensionality reduction techniques, regularization (L1 and L2) techniques and cross-validation.

DISADVANTAGES

- Prone to underfitting
- Sensitive to outliers
- Linear Regression assumes that the data is independent

SAMPLE CODE USING SCIKIT LEARN

Source - scikit learn documentation

```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error

%23 Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

%23 Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]

%23 Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

%23 Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]

%23 Create linear regression object
regr = linear_model.LinearRegression()
```

SAMPLE CODE USING SCIKIT LEARN

Source - scikit learn documentation

```
# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print('Mean squared error: %.2f'
      % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'
      % r2_score(diabetes_y_test, diabetes_y_pred))

#output
Coefficients: [938.23786125]
Mean squared error: 2548.07
Coefficient of determination: 0.47
```

RESOURCES

CLICK THE LINKS TO GET RESOURCES

- Linear regression - 1
- Linear regression - 2
- Linear regression - 3