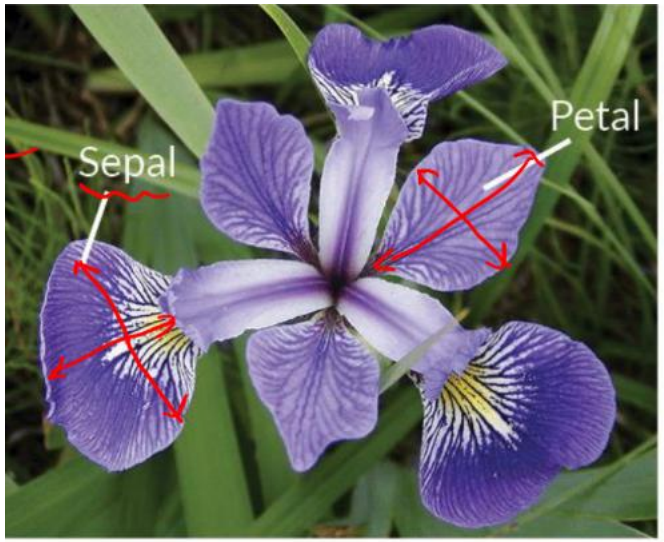
Computer Vision

CVI620

Session 12
02/2025

Iris Data Classification



Iris Versicolor

Iris Setosa

Iris Virginica

Steps to Solve ML Problems

Data and Preprocessing

ML Algorithm

Plot and Evaluate

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score

df = pd.read_csv("iris.data", header=None)
df.columns = ["sepal_length", "sepal_width", "petal_length", "petal_width", "target"]

X = df.drop(columns=["target"])
y = df["target"]

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)


y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Diabetes Classification

Data and Preprocessing

ML Algorithm

Plot and Evaluate

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score


dataset = pd.read_csv("S12_diabetes.csv")

zero_not_accepted = ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]
for column in zero_not_accepted:
    dataset[column] = dataset[column].replace(0, np.nan)
    mean = int(dataset[column].mean(skipna=True))
    dataset[column] = dataset[column].replace(np.nan, mean)

X = dataset.iloc[:, :8]
y = dataset.iloc[:, 8]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)


clf = KNeighborsClassifier(n_neighbors=11)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f'accuracy: {acc}')
```

Data Preprocessing

- Convert to numerical values
- Encoding categorical values
- Replace Null values
- Normalize values
- Splitting data

Normalization

Prevent features with larger ranges from dominating

Base comparison in distance based models

Standard scaler

Min-Max scaler

Standard Scaler

- Standard Scaler transforms features to have zero mean and unit variance using:

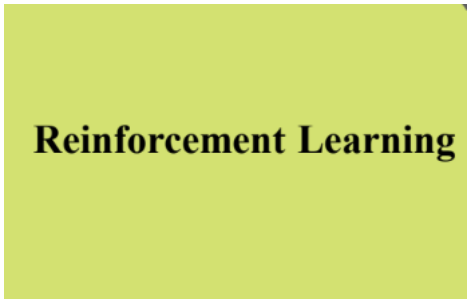
$$z = \frac{x - \mu}{\sigma}$$

μ = Mean
σ = Standard Deviation

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

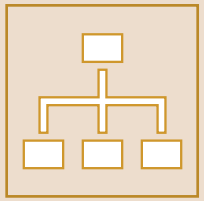
```python
from sklearn.preprocessing import StandardScaler

data = [[0, 0],
        [0, 0],
        [1, 1],
        [1, 1]]

scaler = StandardScaler()
new_data = scaler.fit_transform(data)
print(new_data)
```

ML Algorithm Categorizations



ML Algorithm Categorizations

Supervised Learning Algorithms Categorization



Classification

Regression

Regression

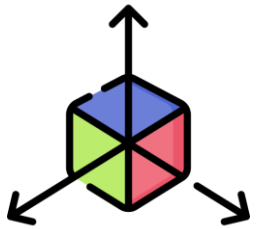
i) Linear

- linear

- nonlinear

i) Multiple

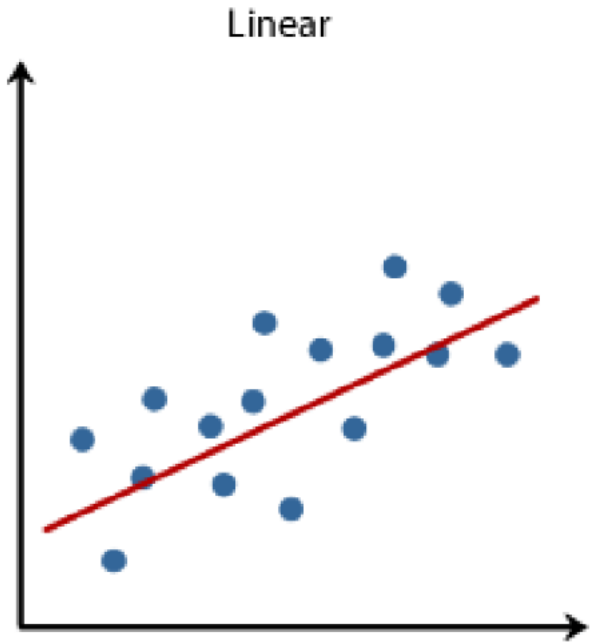
- linear
- nonlinear



Polynomial

Simple Linear Regression

model the relationship between one independent variable (X) and one dependent variable (Y)

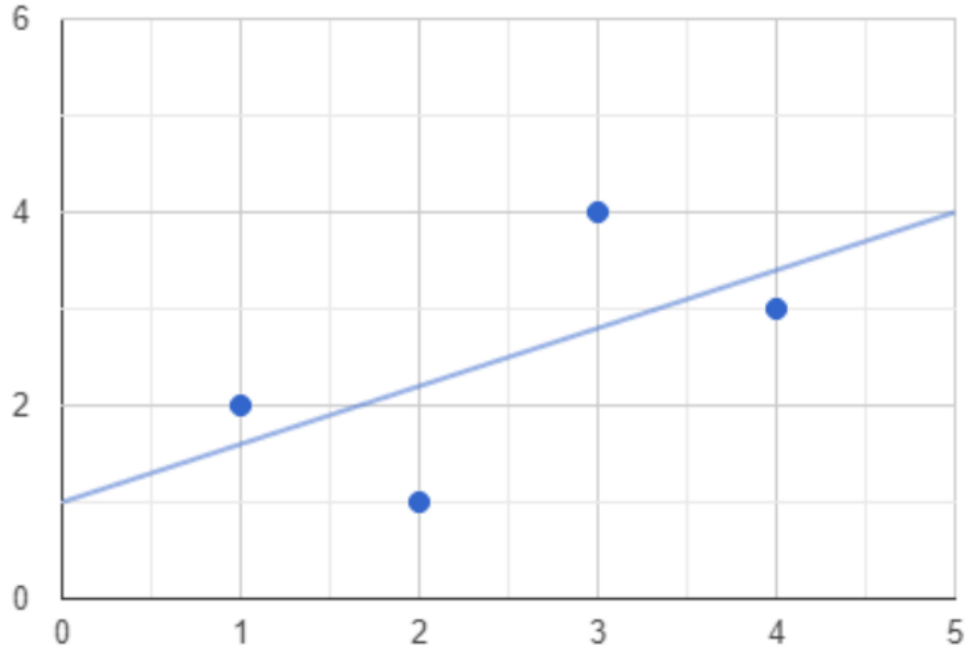


intercept

$$y = a + bx$$

slope

Problem



Our goal:

find a line that minimizes our error

Solution 1

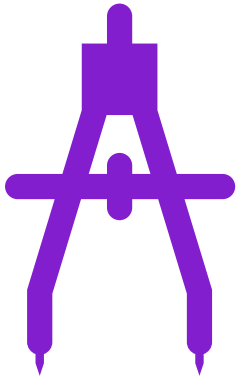
Visualize data and plot points → not scalable

Use Closed-Form Formulas → complex and not exact match

Use optimization algorithms

Formulate

Points: (1,2), (2,1), (3,4), (4,3)



Solution: map to a linear line

Solution

- Points: (1,2), (2,1), (3,4), (4,3)

$$a*1 + b = 2$$
$$a*2 + b = 1$$
$$a*3 + b = 4$$
$$a*4 + b = 3$$

Can’t find the exact **linear** line

Solution

- Points: (1,2), (2,1), (3,4), (4,3)

a*1 + b = 2

a*2 + b = 1

a*3 + b = 4

a*4 + b = 3

Solve with optimization

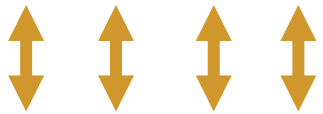
Solution

- Points: (1,2), (2,1), (3,4), (4,3)

$$a*1 + b = 2$$
$$a*2 + b = 1$$
$$a*3 + b = 4$$
$$a*4 + b = 3$$

We want to find a line to minimize the distance between the real value and the lines output

y1, y2, y3, y4



y1′, y2′, y3′, y4′

Solution

- Points: (1,2), (2,1), (3,4), (4,3)

$a*1 + b = 2$

$a*2 + b = 1$

$a*3 + b = 4$

$a*4 + b = 3$

We want to find a line to minimize the distance between the real value and the lines output

y1, y2, y3, y4

y1′, y2′, y3′, y4′

→ Minimize the distance

Solution

- Points: (1,2), (2,1), (3,4), (4,3)

$a*1 + b = 2$

$a*2 + b = 1$

$a*3 + b = 4$

$a*4 + b = 3$

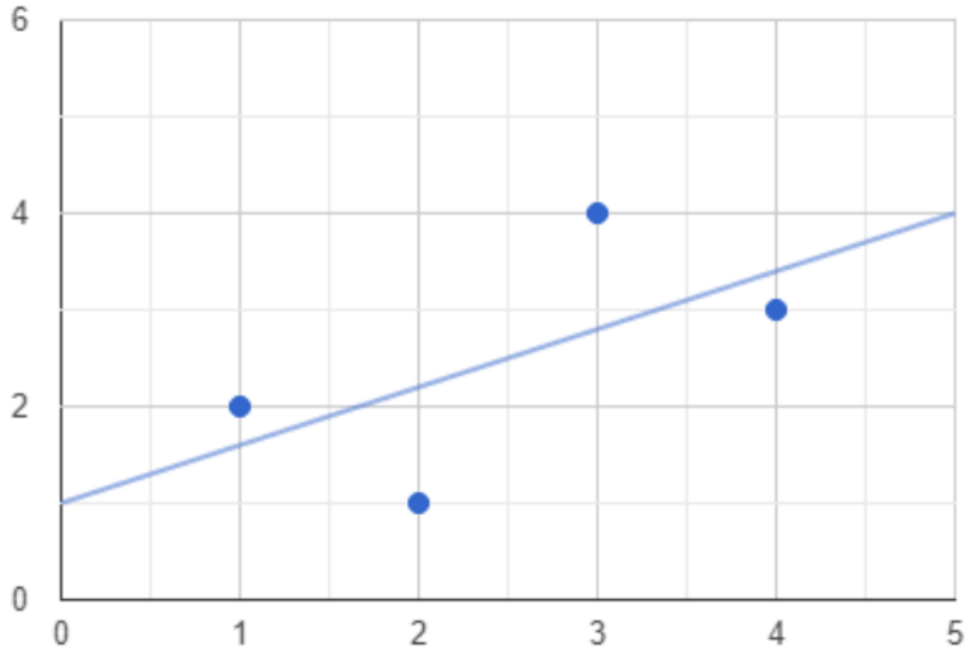
We want to find a line to minimize the distance between the real value and the lines output

$y1,\ y2,\ y3,\ y4$

$y1',\ y2',\ y3',\ y4'$

$$\min \left[(y1-y1')^2 + (y2-y2')^2 + (y3-y3')^2 + (y4-y4')^2 \right]$$

Example

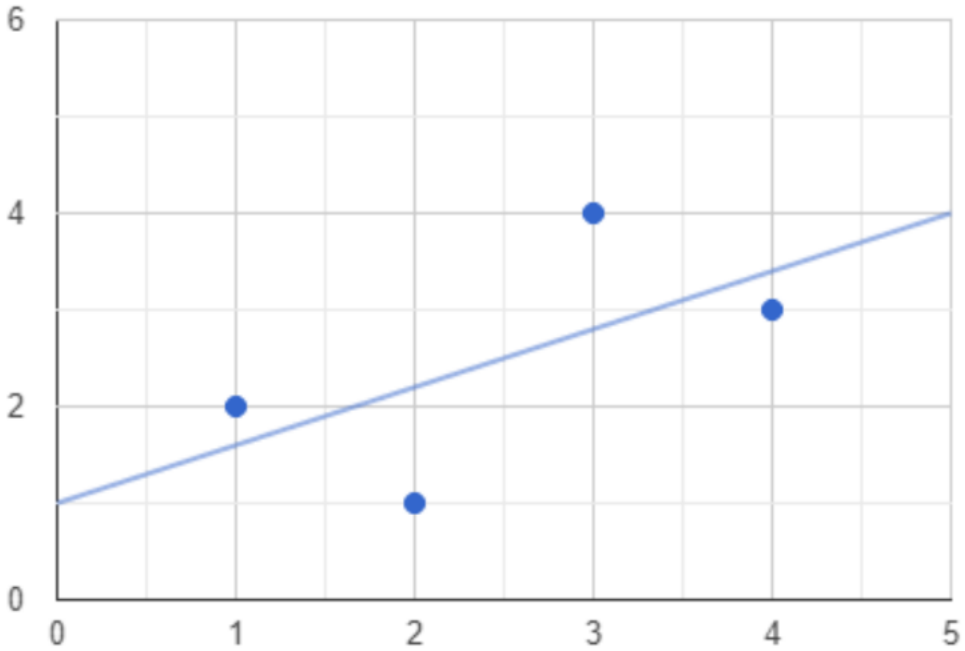


Actual outputs: 2, 1, 4, 3

Line's outputs: -1, 0, 1, 2

We do not know the equation of the line! But we want to find it
$y' = ax + b$

Example



Actual outputs: 2, 1, 4, 3

Line's outputs: -1, 0, 1, 2

$(y1-y1')^2 + (y2-y2')^2 + (y3-y3')^2 + (y4-y4')^2$

$\text{loss/distance} = (2-(a+b))^2 + (1-(2a+b))^2 + (4-(3a+b))^2 + (3-(4a+b))^2$

Example

$$\text{loss/distance} = (2-(a+b))^2 + (1-(2a+b))^2 + (4-(3a+b))^2 + (3-(4a+b))^2$$

$$= 30*a^2 + 20*a*b + 56*a + 4*b^2 - 20*b + 30$$

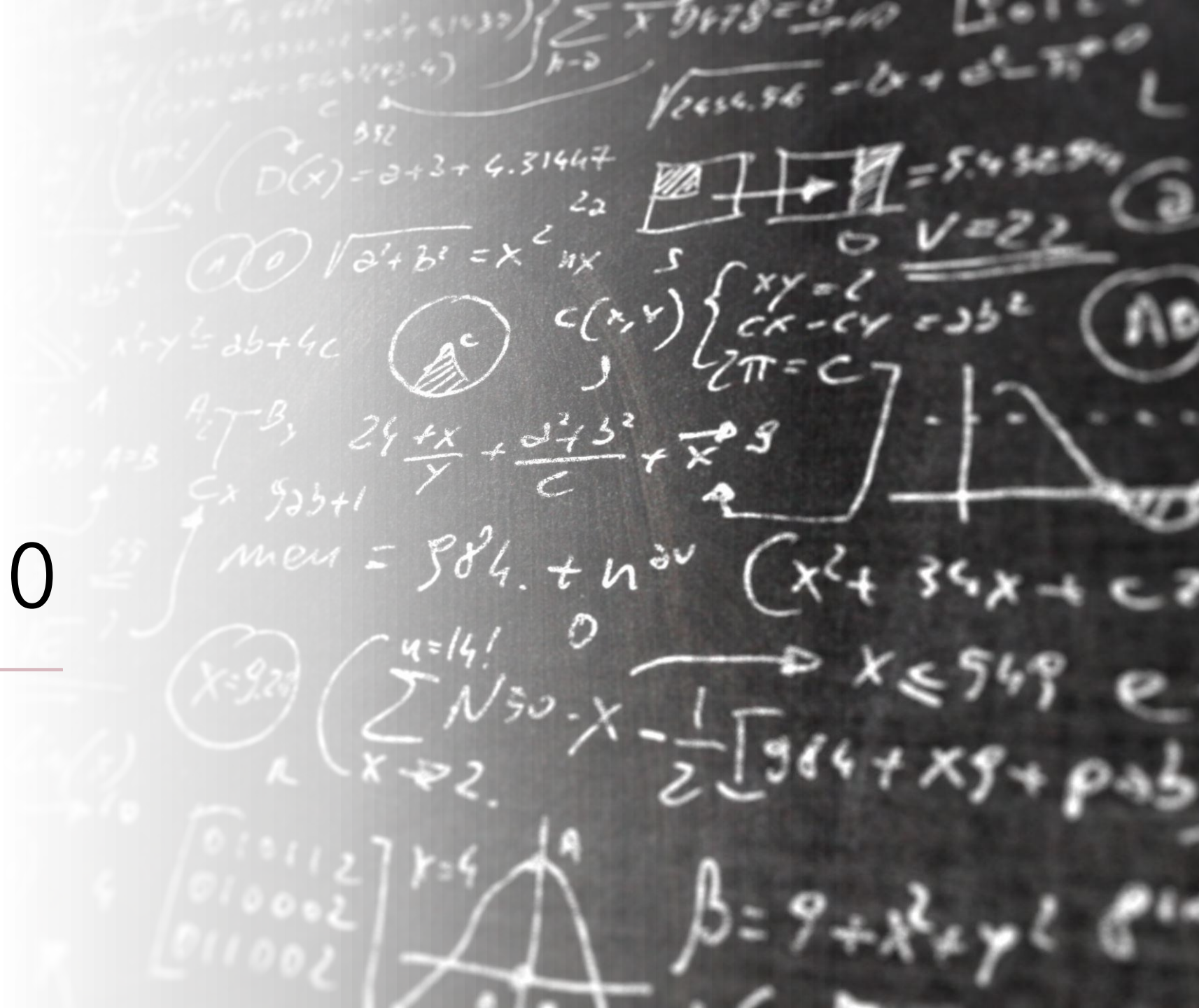
Minimum Loss

- We want to move towards a point with minimum loss!
- It is not the best fit, but it is better than nothing!

Minimum Loss

- Mathematically speaking, when we want the minimum of something what did we do?

Derivative = 0



Equation

- $30*a^2 + 20*a*b + 56*a + 4*b^2 - 20*b + 30$

$df/da = 0$

$df/db = 0$

Solve Equation

- $30*a^2 + 20*a*b + 56*a + 4*b^2 - 20*b + 30$

$df/da = 0$

$df/db = 0$

$60a + 20b + 56 = 0$

$20a + 8b + 20 = 0$

Solve Equation

- $30*a^2 + 20*a*b + 56*a + 4*b^2 - 20*b + 30$

$df/da = 0$

$df/db = 0$

$60a + 20b + 56 = 0$

$20a + 8b + 20 = 0$

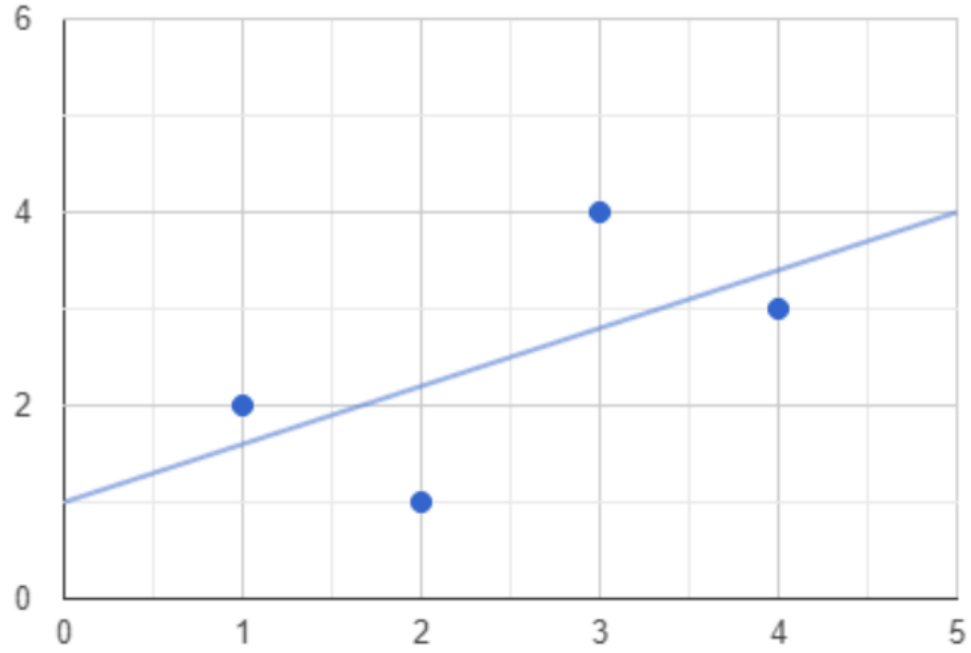
$a = 1$
$b = 0.6$

Final Equation

Now we have a line with minimum error

$$y = x + 0.6$$

Final Output



$y = x + 0.6$

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error


dataset = pd.read_csv("S12_student_scores.csv")

X = dataset.iloc[:, :-1]
y = dataset.iloc[:, 1]

plt.scatter(X, y)
plt.title("Hours vs Marks")
plt.xlabel("Hours")
plt.ylabel("Marks")
plt.show()

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

regressor = LinearRegression()
regressor.fit(X_train, y_train)

print(regressor.coef_)
print(regressor.intercept_)

y_pred = regressor.predict(X_test)
df = pd.DataFrame({'Actual':  y_test, "prediction": y_pred})

print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
print(f"MAE: {mean_squared_error(y_test, y_pred)}")
```