

Three Basic Algorithms

1. Linear Regression
2. KNN
3. K Means

1. Linear Regression Algorithm

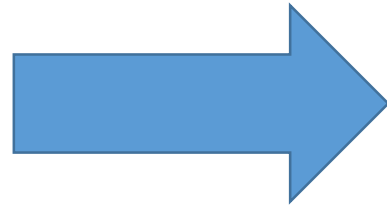
- 1. Collect Data:** Gather data for the dependent and independent variables.
- 2. Visualize Data:** Plot the data to see if there is an apparent linear relationship.
- 3. Fit the Model:** Use the least squares method to find the best-fitting line.
- 4. Evaluate the Model:** Check how well the line fits the data using metrics like Mean Squared Error (MSE).
- 5. Make Predictions:** Use the fitted model to make predictions on new data.

Example1:

- Suppose you **run a social networking** site that charges a monthly subscription **fee of \$25**, and that this is your only source of revenue.
- Each month you collect data and count your number of users and total revenue. You've done this daily over the **course of two years**, recording it all in a spreadsheet. You could express this data as a series of points. Here are the first four:
- $S = \{(x, y) = (1, 25), (10, 250), (100, 2500), (200, 5000)\}$

Equation to represent the relationship of x and y

x	y
1	25
10	250
100	2500
200	5000



$$y=25x$$

Deduce this mentally and identify the following:

- There's a linear pattern.
- The coefficient relating x and y is 25.
- The relationship seems deterministic.

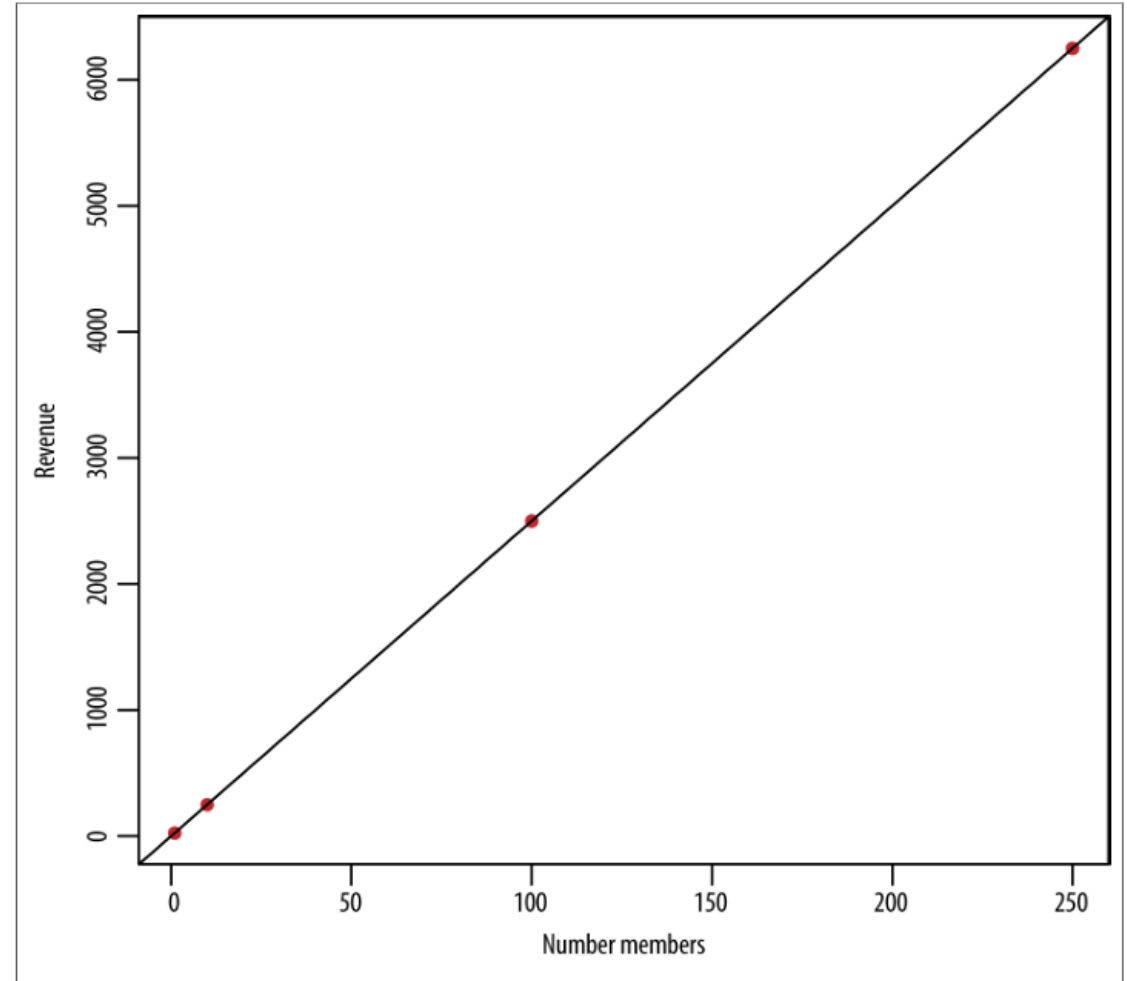


Figure 3-1. An obvious linear pattern

Example2: Sample Data and Plot:

new_friends	time_spent (seconds)
7	276
3	43
4	82
6	136
10	417
9	269

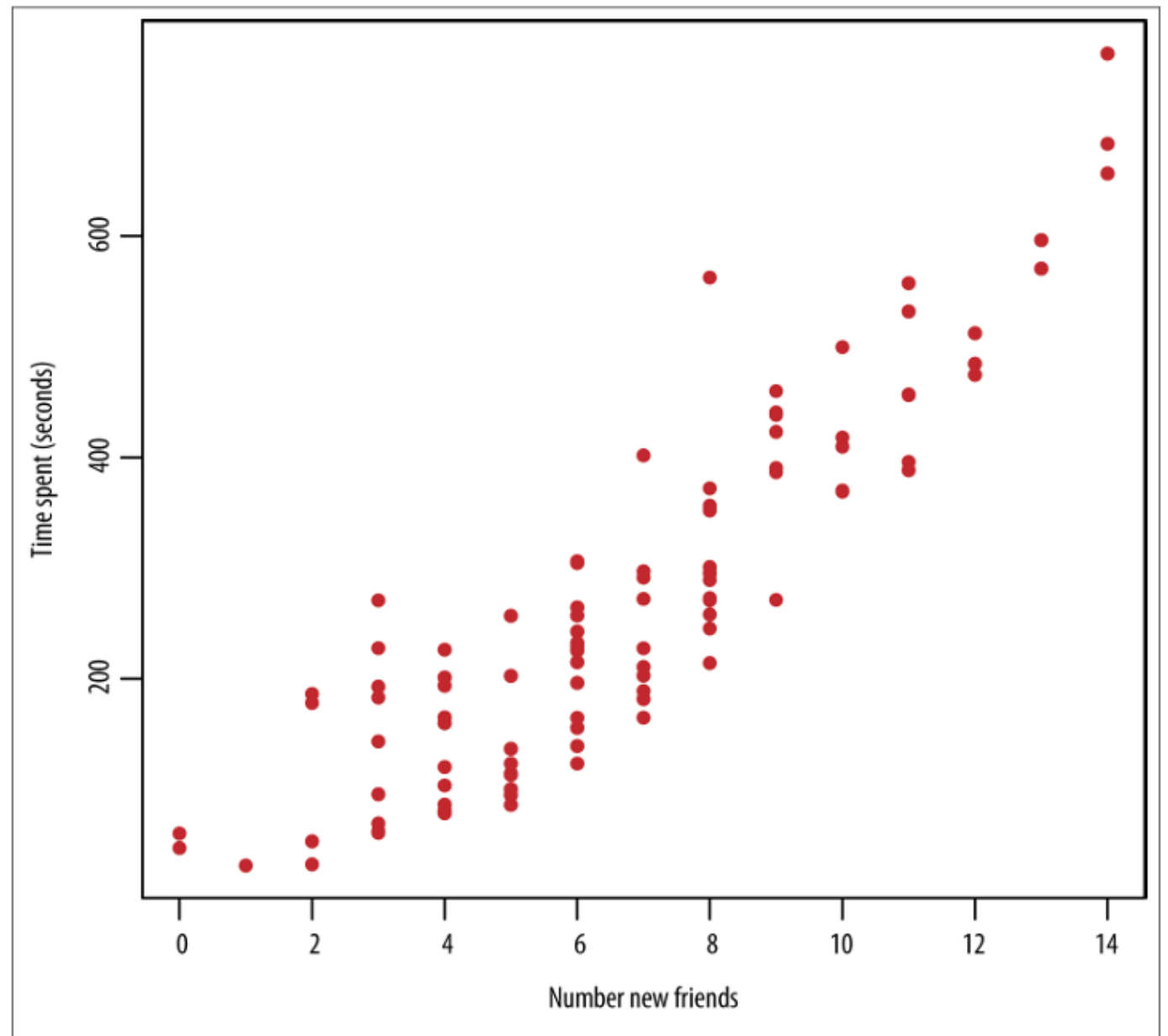


Figure 3-2. Looking kind of linear

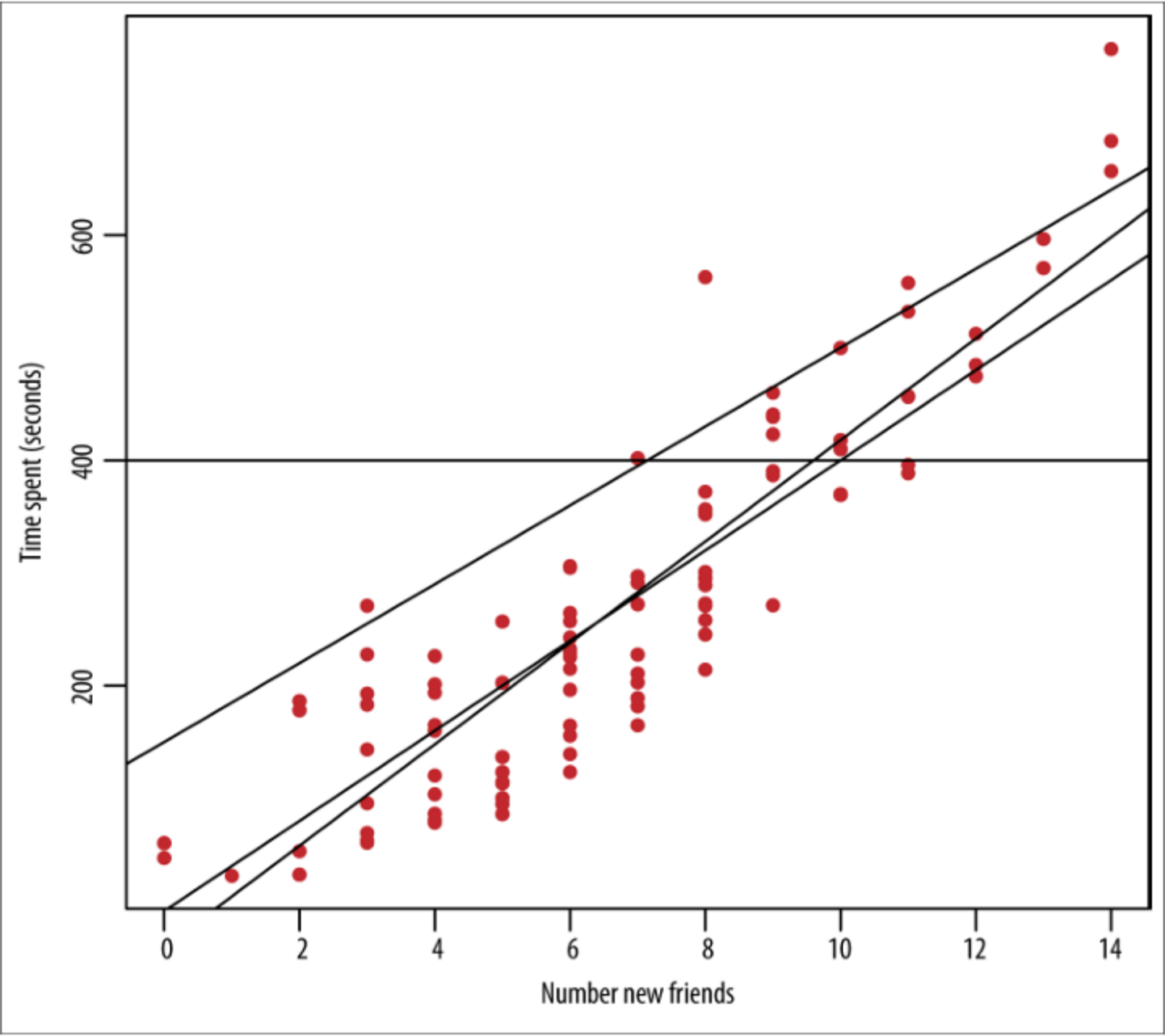


Figure 3-3. Which line is the best fit?

Which Line is the best fit?

- **Assuming a Linear Relationship:**
 - Start with the assumption: $y = \beta_0 + \beta_1 x$,
 - β_0 and β_1 represent the intercept and slope, respectively
- **Estimate the parameters (β_0 and β_1)** using observed data pairs:
 - $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$.

Linear model in matrix notation

- Express the linear model in **matrix notation**: $y = x \cdot \beta$.
- x is the matrix of input variables, and β is the parameter vector.

x	y
2	5
3	7
5	11

$$\begin{pmatrix} 5 \\ 7 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 5 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$$

In this example:

- $y_1=5$, $y_2=7$, and $y_3=11$ are the observed values of the dependent variable.
- $x_1=2$, $x_2=3$, and $x_3=5$ are the values of the independent variable.
- β_0 is the intercept and β_1 is the slope of the linear

Fitting the model

- By applying least squares estimation, Linear regression seeks to find the line that minimizes the **sum of the squares** of the vertical distances between the **approximated or predicted** y_i^{\wedge} s and the **observed** y_i s.
- To find this line, you define the "residual sum of squares" (RSS), denoted as:
- **$RSS(\beta) = \sum_i (y_i - \beta x_i)^2$**
 - where i ranges over the various data points.

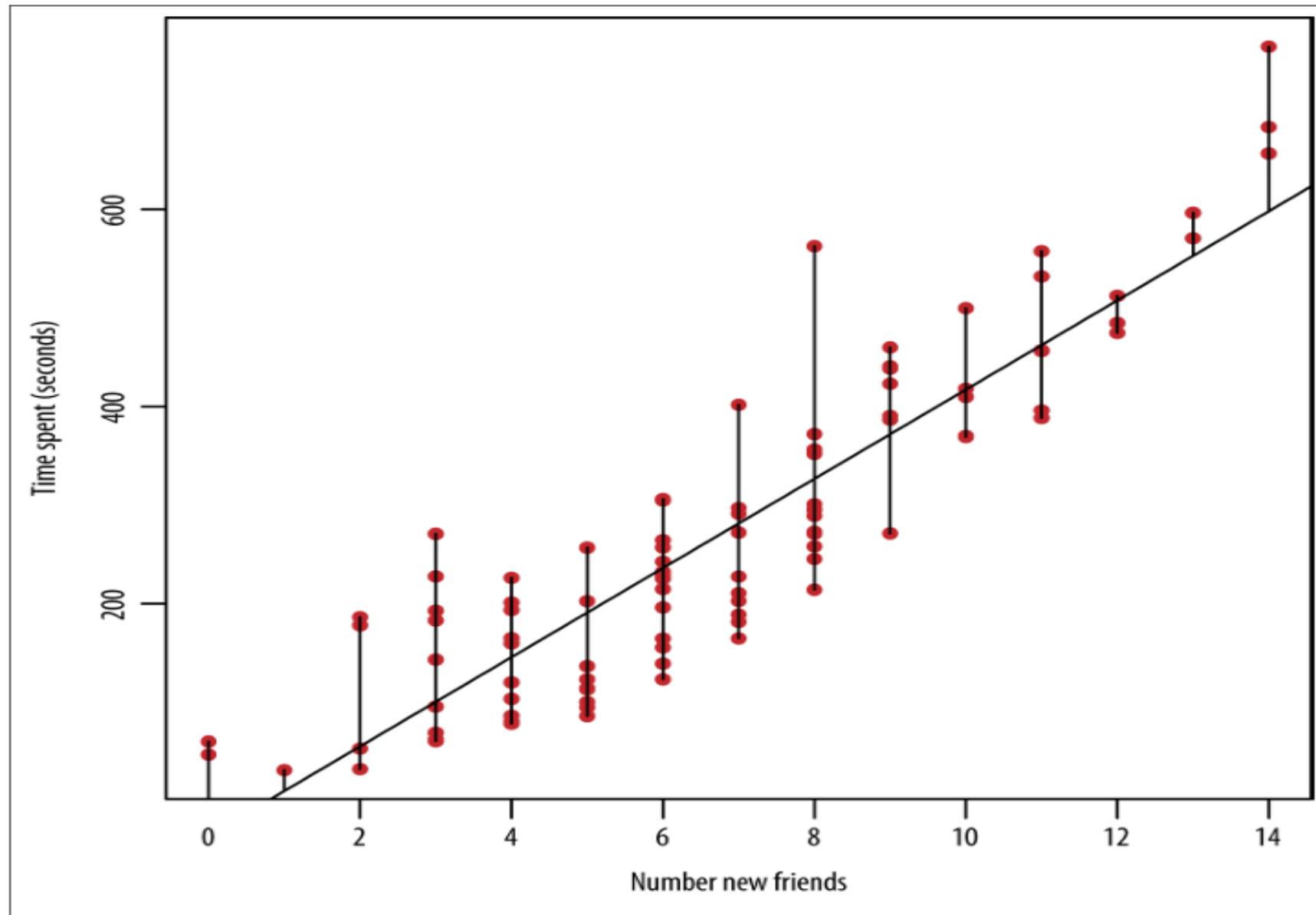


Figure 3-4. The line closest to all the points

Evaluation metrics

- **R-squared (R^2)** : R^2 measures the proportion of the variance in the **dependent variable that is predictable** from the **independent variable(s)**.
- R^2 ranges from **0 to 1**. A higher R^2 value indicates a better fit of the model to the data.

Formula:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

- $\sum_i (y_i - \hat{y}_i)^2$: Sum of squared residuals (errors between observed and predicted values).
- $\sum_i (y_i - \bar{y})^2$: Total sum of squares (variance of the observed values from their mean).

Evaluation metrics

- **p-values:** The p-value tests the null hypothesis that a coefficient (β) is equal to zero (no effect).
- A low p-value (**< 0.05**) indicates that you can reject the **null hypothesis**, meaning the coefficient is significantly different from zero.

```
summary (model)
```

```
Call:
```

```
lm(formula = y ~ x)
```

```
Residuals:
```

```
      Min       1Q   Median       3Q      Max
-121.17  -52.63   -9.72   41.54  356.27
```

```
Coefficients:
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -32.083     16.623   -1.93  0.0565 .
x              45.918      2.141   21.45 <2e-16 ***
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 77.47 on 98 degrees of freedom
```

```
Multiple R-squared:  0.8244,    Adjusted R-squared:  0.8226
```

```
F-statistic:   460 on 1 and 98 DF,  p-value: < 2.2e-16
```



Example R Code for the Sample Data

new_friends	time_spent (seconds)
7	276
3	43
4	82
6	136
10	417
9	269

Example of Code in R

```
# Example dataset  
x <- c(7, 3, 4, 6, 10, 9)  
y <- c(276, 43, 82, 136, 417, 269)
```

```
# Fit the linear model
```

```
model <- lm(y ~ x)
```

```
# Print the model summary
```

```
summary(model)
```

Output:

Call:

```
lm(formula = y ~ x)
```

Residuals:

```
 1      2      3      4      5      6  
47.547 11.507  1.267 -43.213 40.827 -57.933
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-116.227	54.792	-2.121	0.10120
x	49.240	7.868	6.258	0.00332

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 48.18 on 4 degrees of freedom

Multiple R-squared: 0.9073, **Adjusted R-squared:** 0.8842

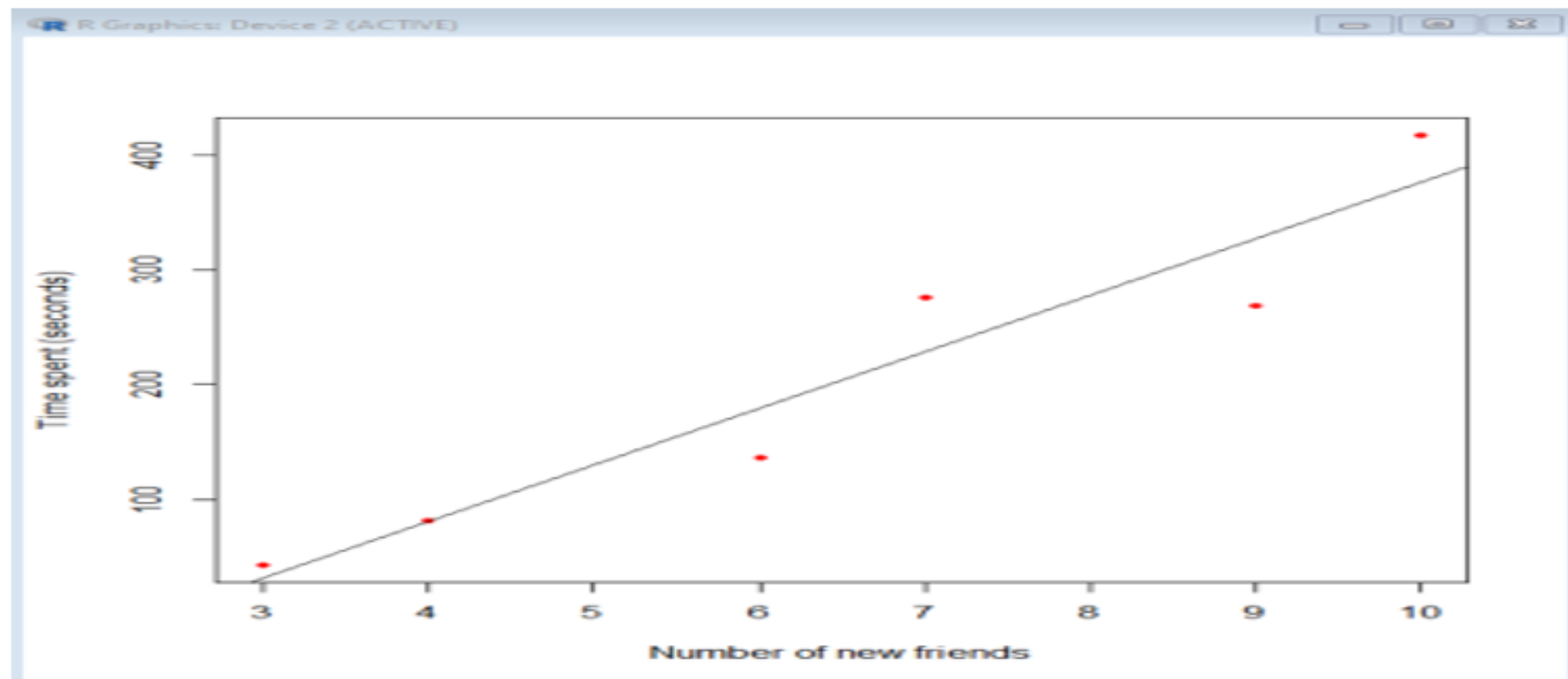
F-statistic: 39.17 on 1 and 4 DF, **p-value:** 0.003325

Plot the data and the fitted line

```
plot(x, y, pch=20, col="red", xlab="Number of new friends", ylab="Time spent  
(seconds)")
```

```
abline(model)
```

Output:



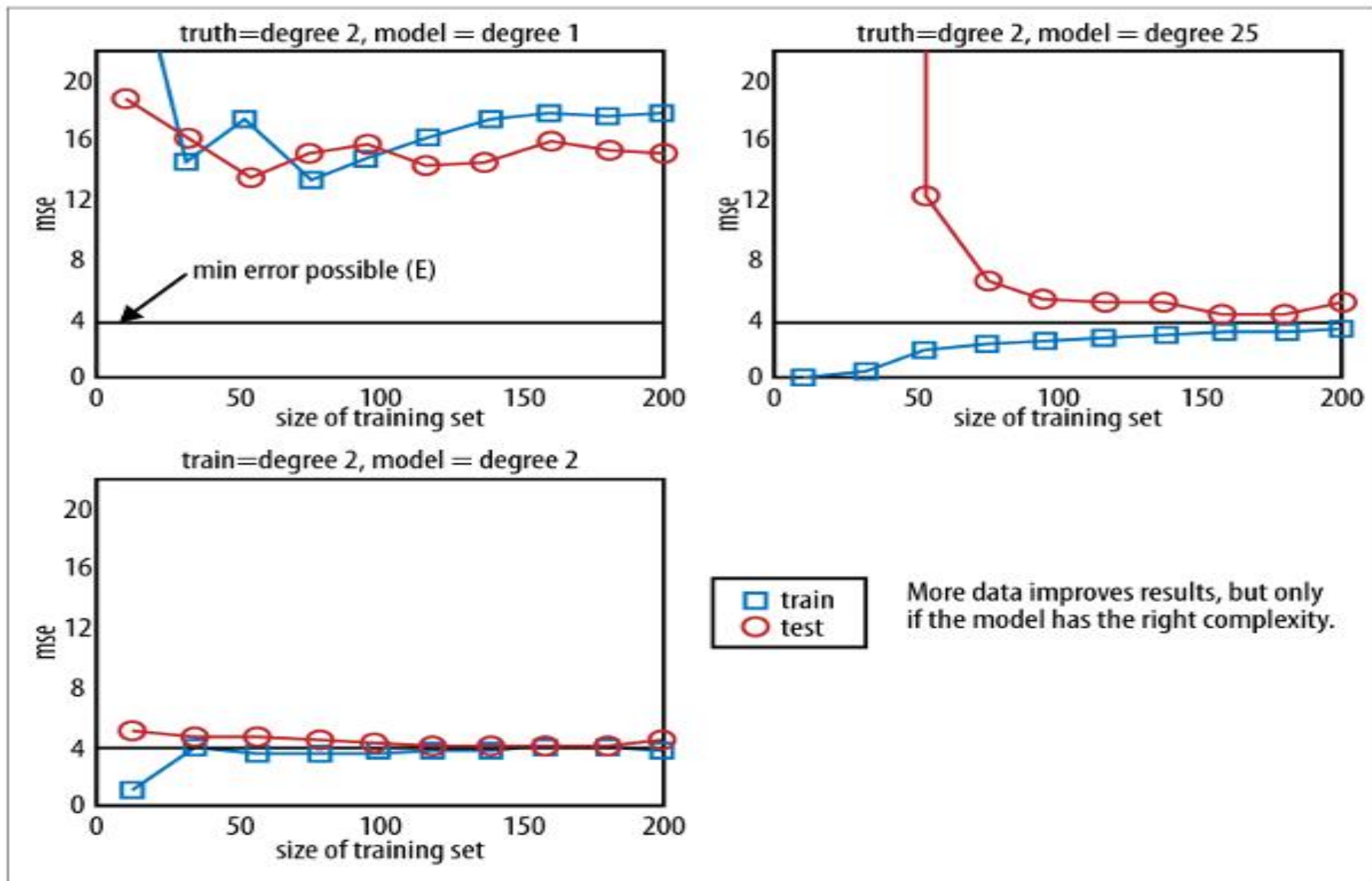


Figure 3-6. Comparing mean squared error in training and test set, taken from a slide of Professor Nando de Freitas; here, the ground truth is known because it came from a dataset with data simulated from a known distribution

2. KNN Algorithm

- The K-Nearest Neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used for both classification and regression tasks.
- It works based on the principle that similar data points (neighbors) are likely to have similar outcomes.

2. KNN Algorithm

- 1. Data Preparation:** Collect and prepare the dataset, ensuring it's in a suitable format for analysis.
- 2. Choosing K:** Select the number of neighbors (K) to consider for making predictions.
- 3. Distance Calculation:** For a given data point that needs to be classified or predicted, calculate the distance between this point and all other points in the dataset. Common distance metrics include Euclidean, Manhattan, and Minkowski distances.
- 4. Identify Neighbors:** Identify the K closest data points (neighbors) to the data point in question.
- 5. Voting/Prediction:**
 - **Classification:** The class with the majority among the K neighbors is assigned to the data point.
 - **Regression:** The average value of the K neighbors is assigned to the data point.

Key Points:

- **Distance Metrics:** Euclidean distance is most common, but other metrics can be used depending on the problem.
- **Choosing K:** The value of K can significantly impact the algorithm's performance. A common approach is to use cross-validation to find the optimal K.
- **Data Scaling:** KNN is sensitive to the scale of the data. It's important to normalize or standardize the data before applying KNN.

Example Data Set:

X1	X2	Label
1	2	A
2	3	A
3	3	B
6	5	B
7	8	B
8	8	A

We want to predict the label of a new point (4, 4) using KNN with $K=3$.

Step-by-Step Example:

We want to predict the label of a new point (4, 4) using KNN with K=3

1. **Choose K:** We choose K=3.

2. **Calculate Distances:** Compute the Euclidean distance from (4, 4) to all points in the dataset.

- Distance to (1, 2): $\sqrt{(4 - 1)^2 + (4 - 2)^2} = \sqrt{9 + 4} = \sqrt{13} \approx 3.61$
- Distance to (2, 3): $\sqrt{(4 - 2)^2 + (4 - 3)^2} = \sqrt{4 + 1} = \sqrt{5} \approx 2.24$
- Distance to (3, 3): $\sqrt{(4 - 3)^2 + (4 - 3)^2} = \sqrt{1 + 1} = \sqrt{2} \approx 1.41$
- Distance to (6, 5): $\sqrt{(4 - 6)^2 + (4 - 5)^2} = \sqrt{4 + 1} = \sqrt{5} \approx 2.24$
- Distance to (7, 8): $\sqrt{(4 - 7)^2 + (4 - 8)^2} = \sqrt{9 + 16} = \sqrt{25} = 5$
- Distance to (8, 8): $\sqrt{(4 - 8)^2 + (4 - 8)^2} = \sqrt{16 + 16} = \sqrt{32} \approx 5.66$

Step-by-Step Example:

3. **Identify Neighbors:** The 3 closest points to (4, 4) are:

- (3, 3) with distance ≈ 1.41 (Label B)
- (2, 3) with distance ≈ 2.24 (Label A)
- (6, 5) with distance ≈ 2.24 (Label B)

4. **Voting/Prediction:**

- Labels of the 3 closest points: B, A, B
- Majority label is B

Thus, the predicted label for the new point (4, 4) is **B**.

Implementation in Python:

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

# Example dataset
X = np.array([[1, 2], [2, 3], [3, 3], [6, 5], [7, 8], [8, 8]])
y = np.array(['A', 'A', 'B', 'B', 'B', 'A'])

# New point to classify
new_point = np.array([[4, 4]])


# Create KNN classifier with K=3
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, y)

# Predict the label for the new point
prediction = knn.predict(new_point)
print(f"The predicted label for point {new_point[0]} is {prediction[0]}")
```


Implementation in Python:

Output:

CSS

 Copy code

```
The predicted label for point [4 4] is B
```

Implementation in R Program

```
# Install and load the required package
install.packages("class")
library(class)

# Example dataset
data <- data.frame(
  X1 = c(1, 2, 3, 6, 7, 8),
  X2 = c(2, 3, 3, 5, 8, 8),
  Label = factor(c('A', 'A', 'B', 'B', 'B', 'A'))
)

# Extract features and labels
features <- data[, 1:2]
labels <- data$Label

# Set the number of neighbors
k <- 3

# Apply KNN
predicted_label <- knn(train = features, test = new_point, cl = labels, k = k)
print(paste("The predicted label for point (4, 4) is", predicted_label))
```

KNN Applications

- **1. Image Classification**
- **2. Recommendation Systems**
- **3. Medical Diagnosis**
- **4. Fraud Detection**
- **5. Customer Segmentation**

