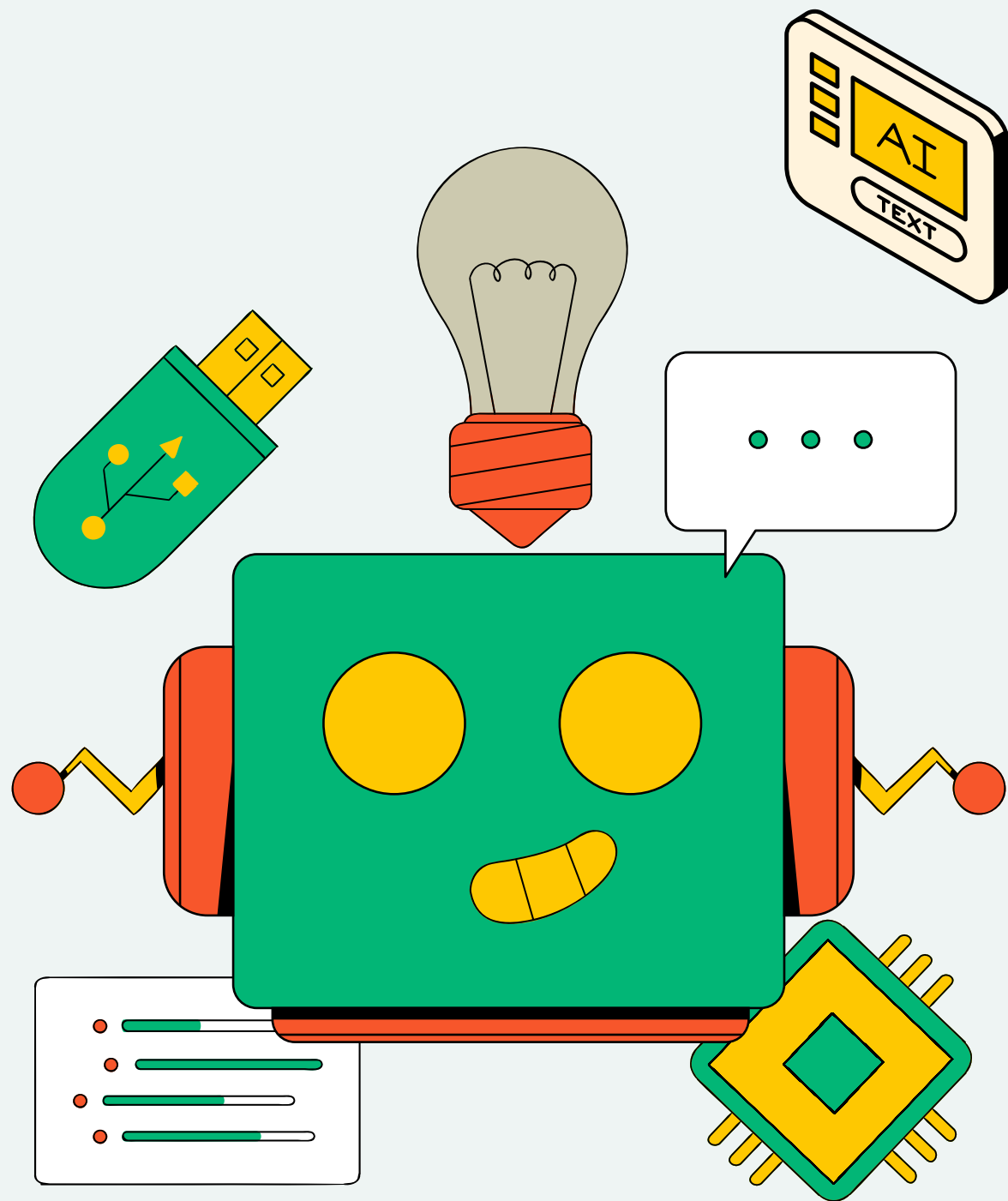




THYNK UNLIMITED
WE LEARN FOR THE FUTURE

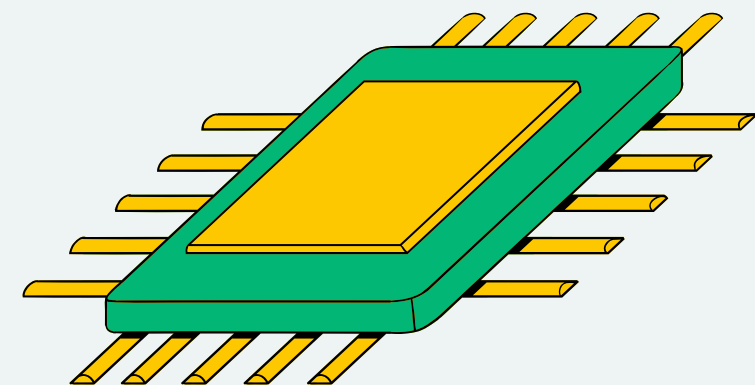


TRAFFIC PREDICTION

USING LOGISTIC REGRESSION

PRESENTED BY:

JIRATCHAYA PANPHINIJ
6310400941





PRESENTATION OUTLINE

- Dataset
- Preprocessing Data
- Model
- Experiment
- Training
- Testing



DATASET

Afghanistan, Kabul, Abdul-haq Crossroad

kaggle

+

Create

Home

Competitions

Datasets

Models

Code

Discussions

Learn

More

Your Work

VIEWED

dataset source

Traffic Prediction D...

Source of the dataset

The dataset

🔍 Search

HASIBULLAH AMAN · UPDATED 4 MONTHS AGO

61

New Notebook

Download (85 kB)

Traffic Prediction Dataset

Real Traffic Prediction Dataset

Data Card

Code (12)

Discussion (4)

Suggestions (0)

About Dataset

Traffic congestion and related problems are a common concern in urban areas. Understanding traffic patterns and analyzing data can provide valuable insights for transportation planning, infrastructure development, and congestion management.

What exactly is this dataset and how was it created?

it is a valuable resource for studying traffic conditions as it contains information collected by a computer vision model. The model detects four classes of vehicles: cars, bikes, buses, and trucks. The dataset is stored in a CSV file and includes additional columns such as time in hours, date, days of the week, and counts for each vehicle type (CarCount, BikeCount, BusCount, TruckCount). The "Total" column represents the total count of all vehicle types detected within a 15-minute duration.

The dataset is updated every 15 minutes, providing a comprehensive view of traffic patterns over the course of one month. Additionally, the dataset includes a column indicating the traffic situation categorized into four classes: 1-Heavy, 2-High, 3-Normal, and 4-Low. This information can help assess the severity of congestion and monitor traffic conditions at different times and days of the week.

Usability ⓘ

8.24

License

Other (specified in description)

Expected update frequency

Monthly

Tags

Business

Data Analytics

Classification

Intermediate

Advanced

A stylized illustration of a person with dark hair, wearing a yellow tank top and green pants, sitting on a yellow laptop. They are holding a green magnifying glass over a document that features a bar chart. A green globe is positioned to the left of the laptop. A speech bubble with three dots is above the person's head. The background is a light blue gradient.

A green circular icon containing three white right-pointing arrows, indicating a 'next' or 'forward' action.

3

DATASET

Time : Time
Day of the week : Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday
CarCount : Number
BikeCount : Number
BusCount : Number
TruckCount : Number
Total : Number
Traffic Situation : low, normal, high, heavy



	Time	Day of the week	CarCount	BikeCount	BusCount	TruckCount	Total	Traffic Situation
0	12:00:00 AM	Tuesday	13	2	2	24	41	normal
1	12:15:00 AM	Tuesday	14	1	1	36	52	normal
2	12:30:00 AM	Tuesday	10	2	2	32	46	normal
3	12:45:00 AM	Tuesday	10	2	2	36	50	normal
4	1:00:00 AM	Tuesday	11	2	1	34	48	normal



PREPROCESSING DATA

	Time	Day of the week	CarCount	BikeCount	BusCount	TruckCount	Total	Traffic Situation
0	12:00:00 AM	Tuesday	13	2	2	24	41	normal
1	12:15:00 AM	Tuesday	14	1	1	36	52	normal
2	12:30:00 AM	Tuesday	10	2	2	32	46	normal
3	12:45:00 AM	Tuesday	10	2	2	36	50	normal
4	1:00:00 AM	Tuesday	11	2	1	34	48	normal

	Time	Day of the week	CarCount	BikeCount	BusCount	TruckCount	Total	Traffic Situation
0	0	2	13	2	2	24	41	0
1	0	2	14	1	1	36	52	0
2	0	2	10	2	2	32	46	0
3	0	2	10	2	2	36	50	0
4	1	2	11	2	1	34	48	0



PREPROCESSING DATA

SPLIT TRAIN AND TEST DATA

```
from sklearn.model_selection import train_test_split  
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Training 80%

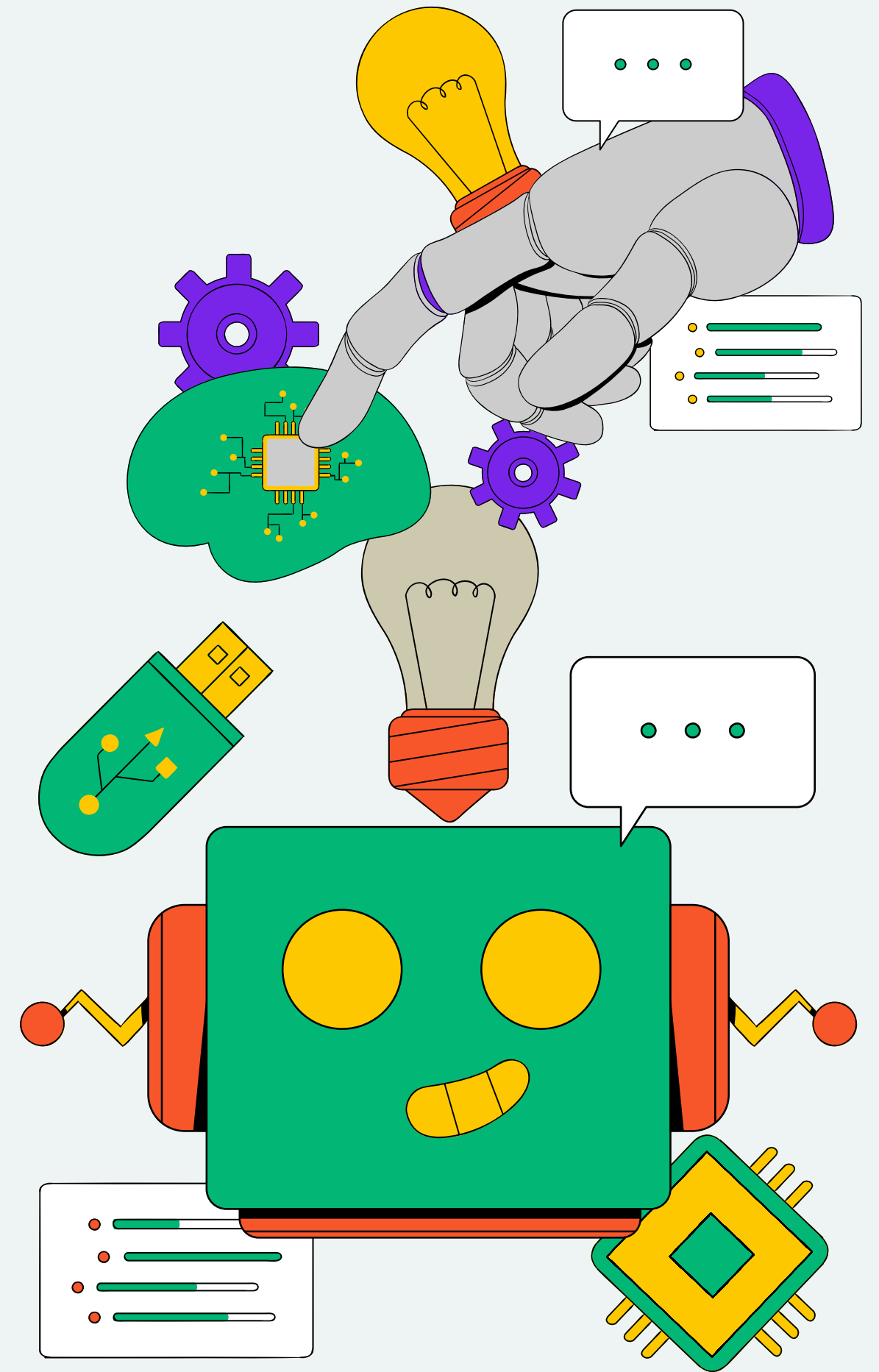
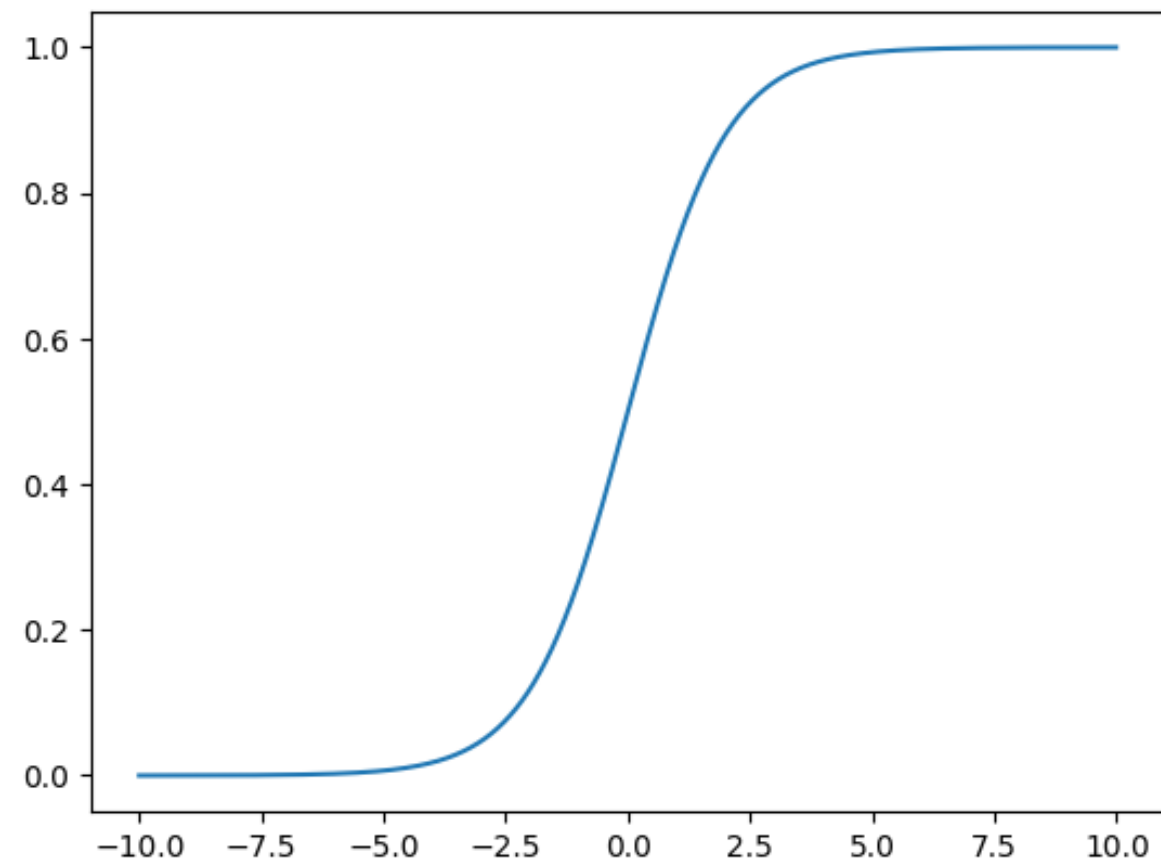
Test 20%



MODEL

SIGMOID FUNCTION

$$P(y|x) = \frac{1}{1 + e^{-(w^T x + b)}}$$



MODEL

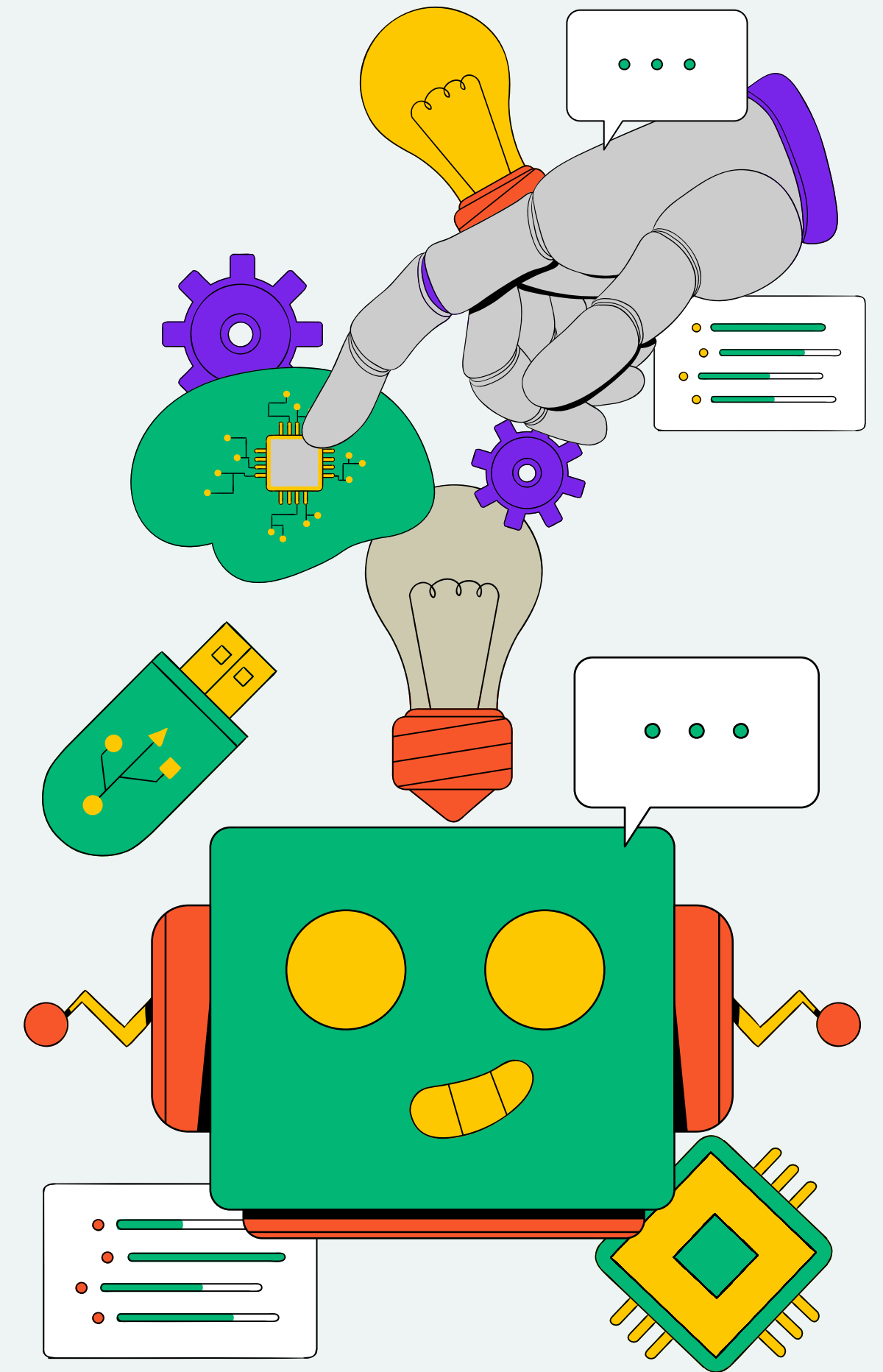
LOSS FUNCTION : BINARY CROSS ENTROPY

$$h_{w,b}(x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

$$J(w, b) = -\frac{1}{n} \sum_{i=1}^n \left[y_i \log(h_{w,b}(x_i)) + (1 - y_i) \log(1 - h_{w,b}(x_i)) \right] + \frac{\lambda}{2} \sum_{j=1}^d |w_j|^2$$

```
def sigmoid(self, z):  
    return 1/(1+np.exp(-z))
```

```
def cross_entropy(self, x, y):  
    eps = 1e-15 # Small constant value  
    z = np.dot(self.w, x.T) + self.b  
    y_pred = self.sigmoid(z)  
    return -(np.dot(y.T, np.log(y_pred + eps)) + np.dot((1-y).T, np.log(1-y_pred + eps)))/x.shape[0] +  
    self.c*np.sum(np.square(self.w))/(2*x.shape[1])
```



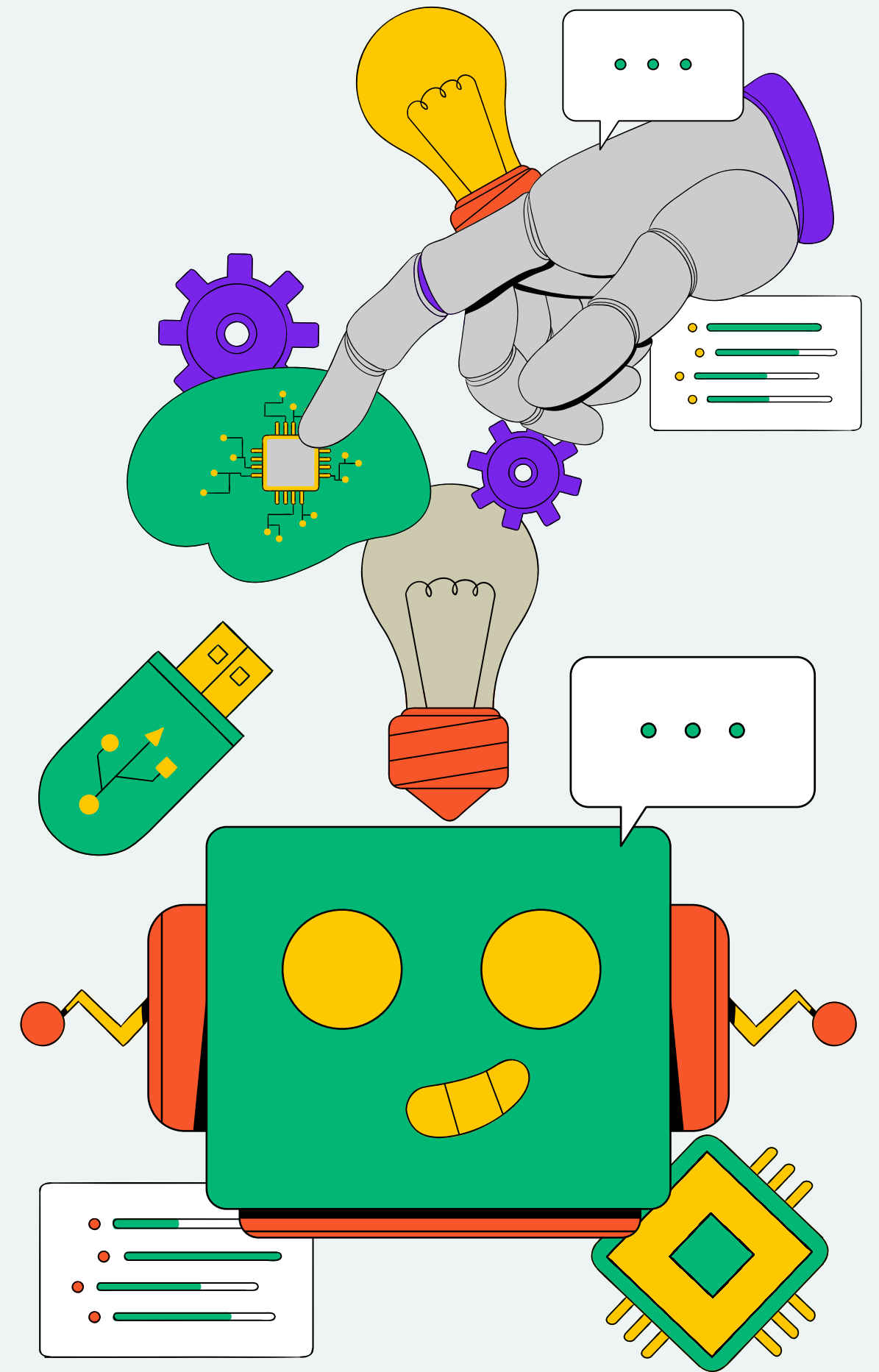
MODEL

GRADIENT DESCENT

$$h_{w,b}(x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

$$\frac{\partial}{\partial w_j} J(w, b) = \frac{1}{n} \sum_{i=1}^n [(h_{w,b}(x_i) - y_i) x_i^j] + \lambda w_j$$

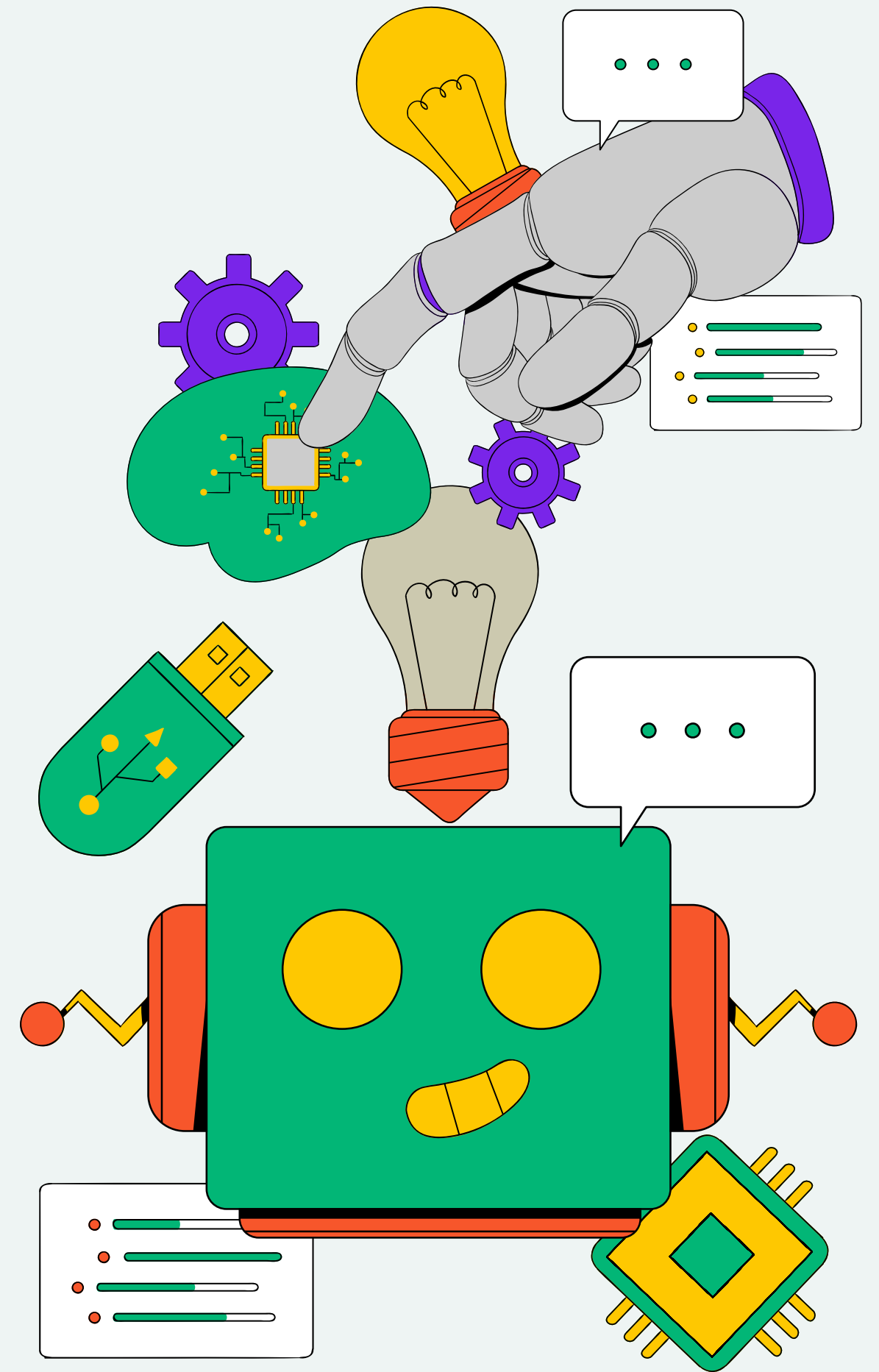
$$\frac{\partial}{\partial b} J(w, b) = \frac{1}{n} \sum_{i=1}^n (h_{w,b}(x_i) - y_i)$$



MODEL

GRADIENT DESCENT

```
for i in range(self.iteration):  
    # Random training data  
    idx = np.random.choice(num_samples, int(batch_size*num_samples))  
    x_batch = x.iloc[idx]  
    y_batch = y[idx]  
  
    # Sigmoid  
    z = np.dot(self.w, x_batch.T) + self.b  
    y_pred = self.sigmoid(z)  
  
    # Calculate gradient  
    gred_w = np.dot(x_batch.T, (y_pred - y_batch))/num_samples  
    gred_b = np.sum(y_pred - y_batch)/num_samples  
  
    # Regularization  
    gred_w = gred_w + self.c*self.w  
  
    # Update parameter  
    self.w = self.w - (self.learning_rete*gred_w)  
    self.b = self.b - (self.learning_rete*gred_b)
```

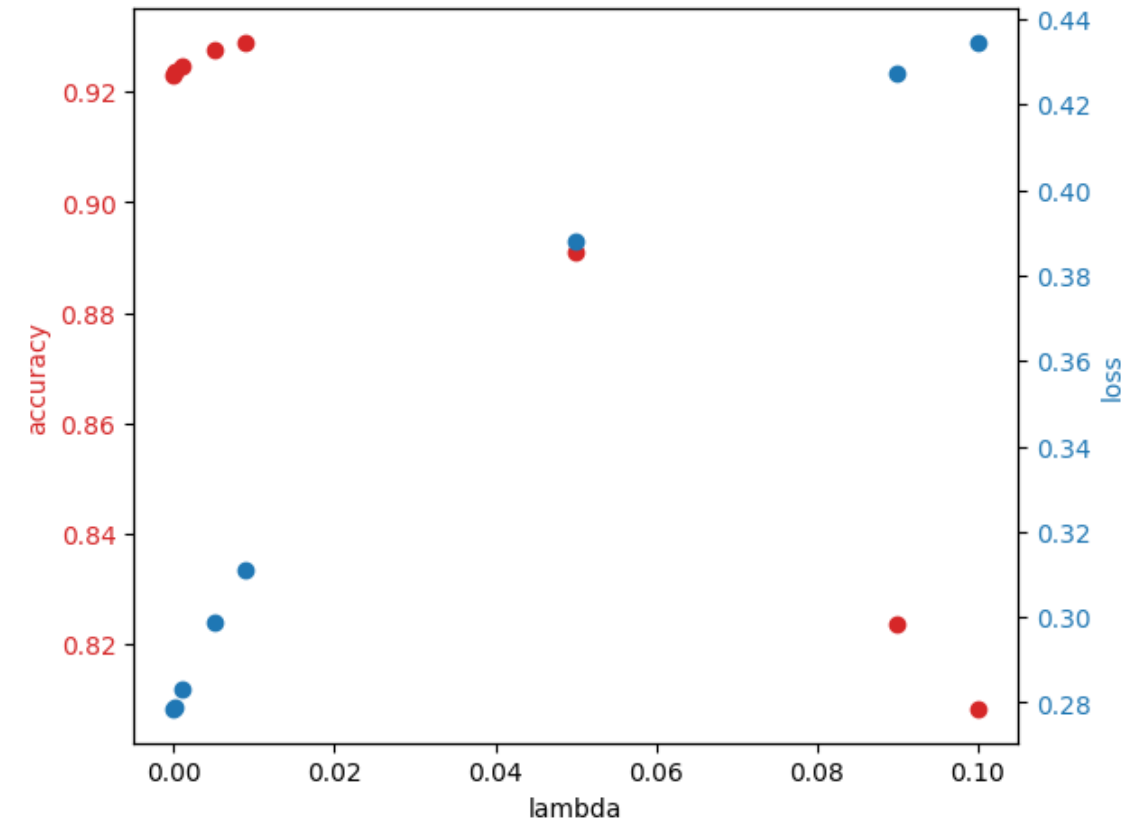


EXPERIMENT

LAMBDA SEARCH

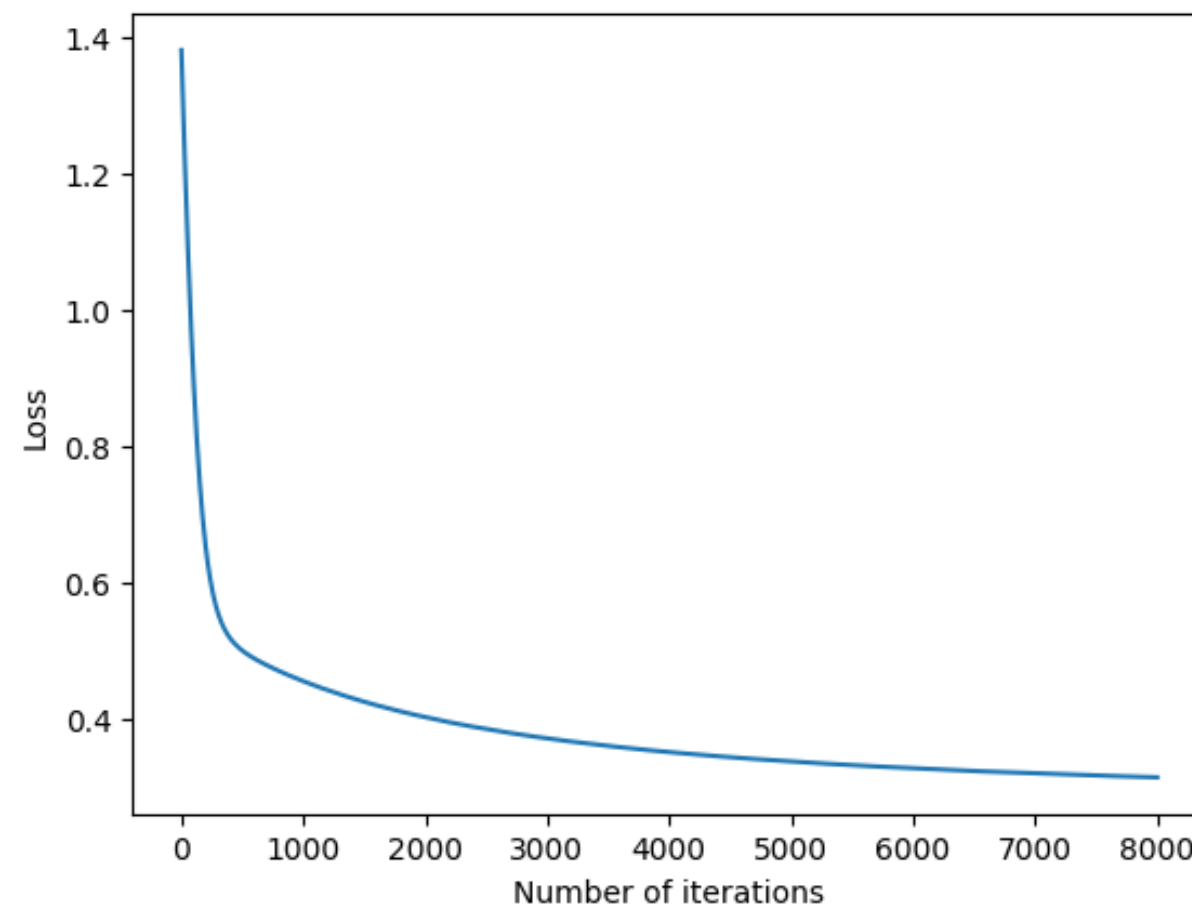
lambda = [0.1, 0.05, 0.09, 0.001, 0.005, 0.009, 0.0001, 0]

	lambda	accuracy	loss
0	0.1000	0.808024	0.434550
1	0.0500	0.890989	0.387831
2	0.0900	0.823356	0.427341
3	0.0010	0.924806	0.283117
4	0.0050	0.927536	0.298757
5	0.0090	0.929007	0.311225
6	0.0001	0.923756	0.278900
7	0.0000	0.923125	0.278410



TRAINING

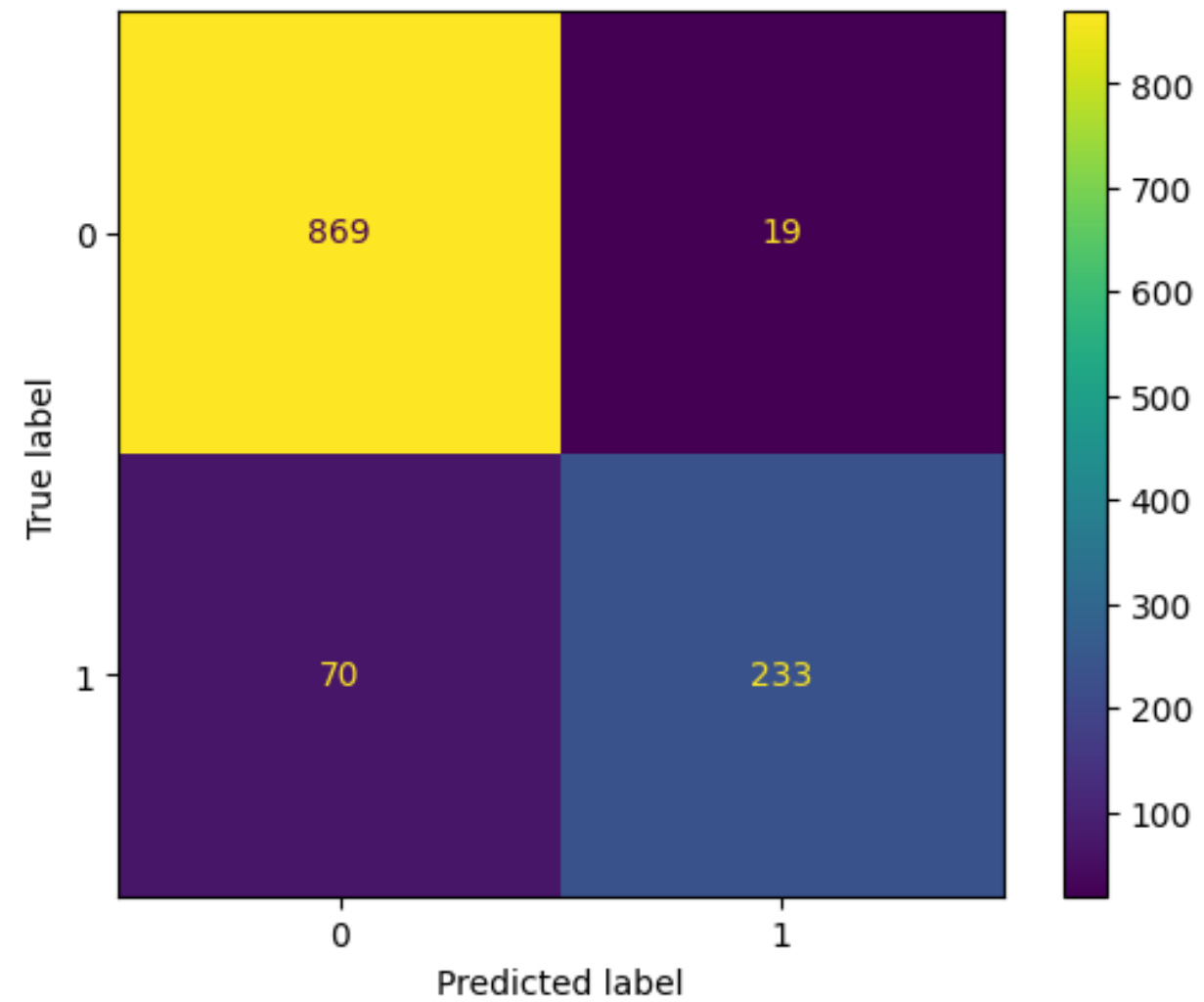
```
clf = LogisticRegression(iteration=8000, learning_rate=0.01, c=0.01, penalty='l2')  
clf.fit(x_train, y_train, batch_size=0.8)
```



accuracy : 0.92
loss : 0.31

TESTING

accuracy : 0.92



THANK YOU

