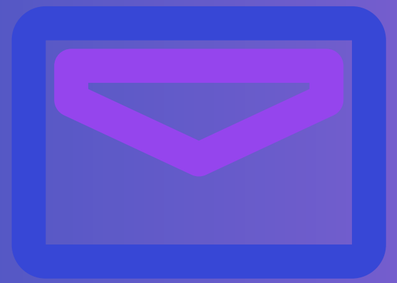
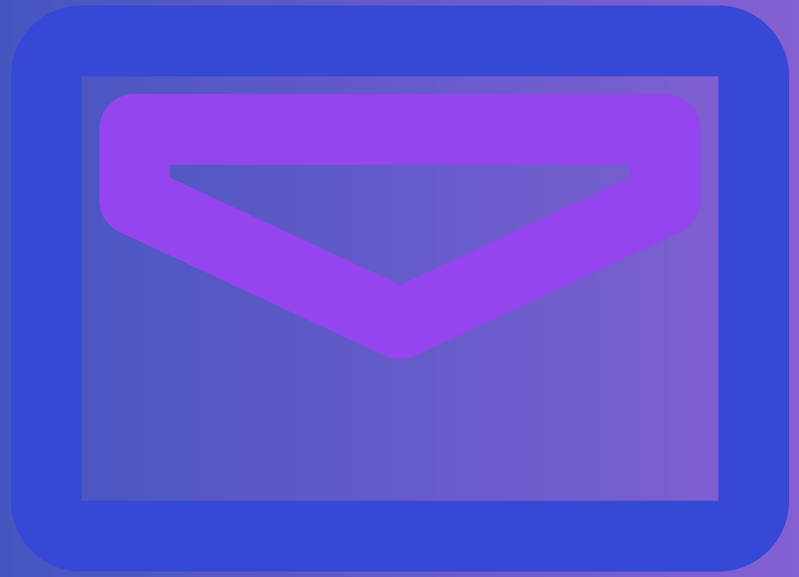
Institute of Technology of Cambodia

Department of Applied Mathematics and Statistics

EMAIL SPAM

GROUP 3





EMAIL!

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

LECTURE BY:

MEMBERS:

CONTENTS:

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

LECTURE BY:

MR.TOUCH SOPHEAK

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

LECTURE BY:

MEMBERS:

CONTENTS:

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

MEMBERS:

HENG SEAKLONG

BO SANE

CHIN VISAL

VITOU RATANAK

DOK DOMINIQUE

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

LECTURE BY:

MEMBERS:

CONTENTS:

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

CONTENTS:

UNDERSTANDING DATASET

DATA PREPROCESSING AND EDA

BUILDING MODEL

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

UNDERSTANDING DATASET

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

UNDERSTANDING DATASET

```
1 # load data
2 email = pd.read_csv("mail_data.csv")
3 email.head()
```

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

DATA PREPROCESSING AND EDA

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

DATA PREPROCESSING

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from tabulate import tabulate
6 from sklearn.feature_extraction.text import TfidfVectorizer
7 from sklearn.linear_model import LogisticRegression
```

```
1 email.shape
```

```
(5572, 2)
```

```
1 email.isnull().sum()
```

```
Category    0
```

```
1 from wordcloud import WordCloud, STOPWORDS
2 comment_words = ''
3 stopwords = set(STOPWORDS)
4 for val in email.Message:
5     # typecaste each val to string
6     val = str(val)
7
8     # split the value
9     tokens = val.split()
10
11    # Converts each token into lowercase
```

```
6 from sklearn.feature_extraction.text import TfidfVectorizer
7 from sklearn.linear_model import LogisticRegression
```

```
email.shape
```

```
(5572, 2)
```

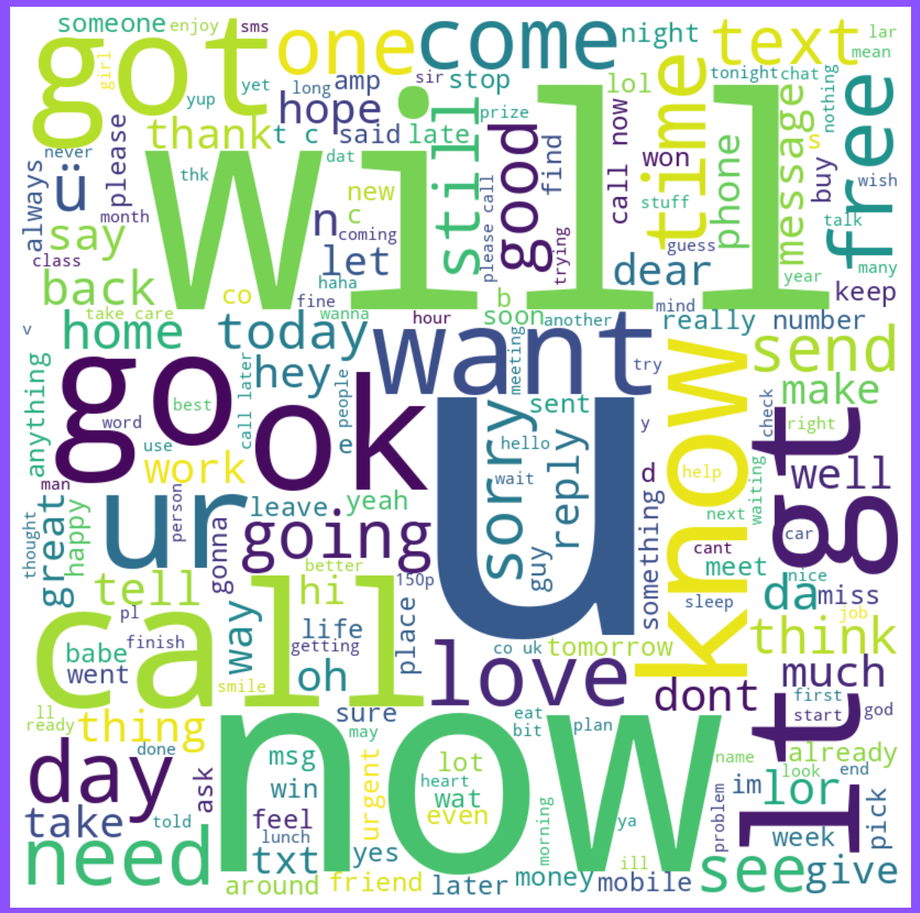
```
email.isnull().sum()
```

```
Category    0
Message     0
dtype: int64
```

```
from wordcloud import WordCloud, STOPWORDS
comment_words = ''
stopwords = set(STOPWORDS)
for val in email.Message:
    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    comment_words += " ".join(tokens)+" "

wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)
```

```
# plot the WordCloud image
```

```
# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```




```
#check dubplicat
```

```
0        False
```

```
1 #check dubplicat
2 email.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
5567    False
5568    False
5569    False
5570    False
5571    False
Length: 5572, dtype: bool
```

```
1 #check
2 print(email[email.duplicated()])
```

```
     Category                                            Message
103       ham  As per your request 'Melle Melle (Oru Minnamin...
154       ham  As per your request 'Melle Melle (Oru Minnamin...
207       ham  As I entered my cabin my PA said, '' Happy B'd...
223       ham                          Sorry, I'll call later
326       ham              No calls..messages..missed calls
...       ...                                             ...
5524     spam  You are awarded a SiPix Digital Camera! call 0...
5535      ham  I know you are thinkin malaria. But relax, chi...
5539      ham                    Just sleeping..and surfing
5553      ham                   Hahaha..use your brain dear
5558      ham                          Sorry, I'll call later

[415 rows x 2 columns]
```

```
1 email.duplicated().sum()
```

```
415
```

```
1 email.drop_duplicates(inplace=True)
2 email.duplicated().sum()
```

```
1 #encoding
2 # Map 'ham' to 0 and 'spam' to 1
3 email['Category'] = email['Category'].map({'ham': 0, 'spam': 1})
4
5 # Split the data into features (X) and labels (y)
6 X = email['Message']
7 y = email['Category']
8
9 # Convert features and labels to numpy arrays
10 X = np.array(X)
11 y = np.array(y)
12 X
```

```
array(['Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...',
       'Ok lar... Joking wif u oni...',
       "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply
08452810075over18's",
       ..., 'Pity, * was in mood for that. So...any other suggestions?',
       "The guy did some bitching but I acted like i'd be interested in buying something else next week and he gave it to us for free",
       'Rofl. Its true to its name'], dtype=object)
```

```
1 y = y.reshape(-1,1)
2 y
```

```
array([[0],
       [0],
       [1],
       ...,
       [0],
       [0],
       [0]])
```

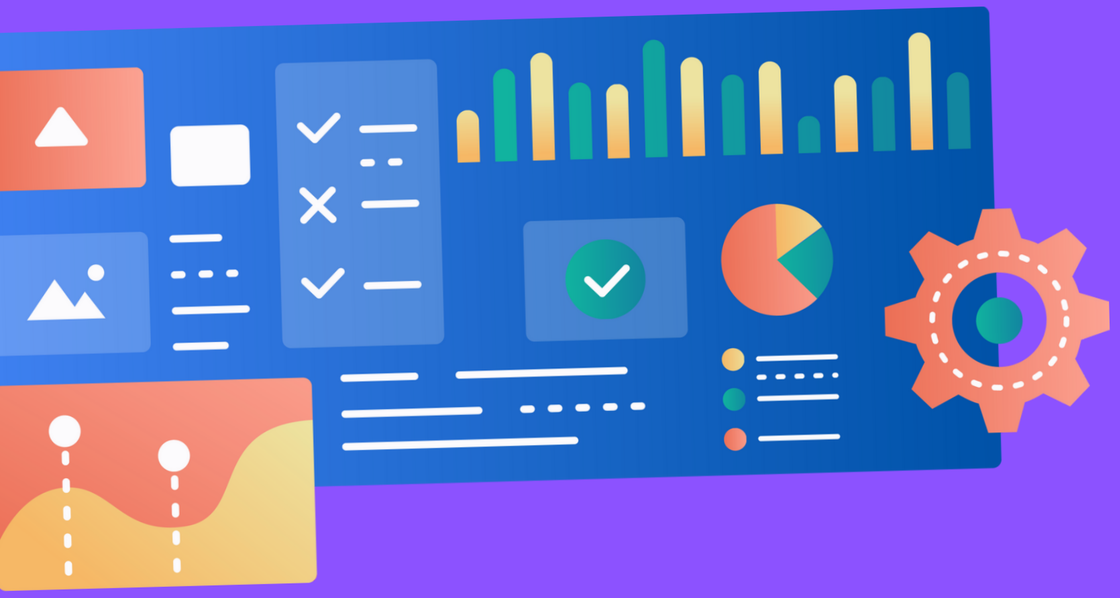
```
1 print(tabulate(email.head(), headers=email.columns, tablefmt="fancy_grid"))
```

	Category	Message
0	0	Go jurong point , crazy .. Available bugis n great world la e buffet ... Cine got amore wat ...
1	0	Ok lar ... Joking wif u oni ...
2	1	Free entry 2 wkly comp win FA Cup final tkts 21st May 2005 . Text FA 87121 receive entry question (std txt rate) T & C 's apply
3	0	U dun say early hor ... U c already say ...
4	0	Nah I n't think goes usf , lives around though

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD



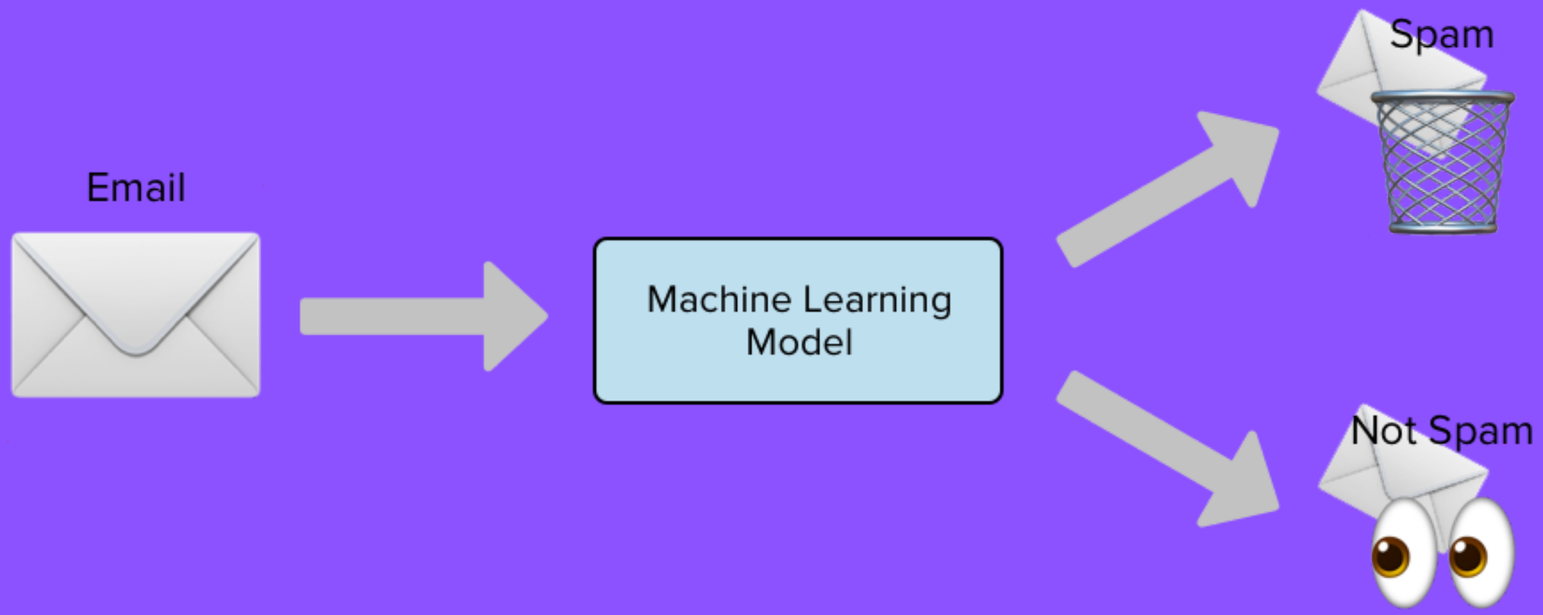
BUILDING MODEL

EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

BUILDING MODEL



BUILDING MODEL

```python
import numpy as np
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# Define the sigmoid function
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Define the cost function
def cost(h, y):
    return (-y * np.log(h) - (1 - y) * np.log(1 - h)).mean()

# Define the gradient descent function
def gradient_descent(X, h, y):
    return np.dot(X.T, (h - y)) / y.shape[0]

# Define the update function
def update_weights(weight, learning_rate, gradient):
    return weight - learning_rate * gradient

```

```python

# Define the prediction function
def predict(X, weight):
    return sigmoid(np.dot(X, weight))

# Define the training function
learning_rate = 1
def train(X, y, weight, learning_rate):
    for i in range(1000):
        h = predict(X, weight)
        gradient = gradient_descent(X, h, y)
        weight = update_weights(weight, learning_rate, gradient)

        if(i % 10 == 0):
          print(f'iterations: {i} cost: {cost(h, y)} \t')
    return weight
```

```
14
15 # Define the gradient descent function
16 def gradient_descent(X, h, y):
17     return np.dot(X.T, (h - y)) / y.shape[0]
18
19 # Define the update function
20 def update_weights(weight, learning_rate, gradient):
21     return weight - learning_rate * gradient
22
```

```
30     for i in range(1000):
31         h = predict(X, weight)
32         gradient = gradient_descent(X, h, y)
33         weight = update_weights(weight, learning_rate, gradient)
34
35         if(i % 10 == 0):
36           print(f'iterations: {i} cost: {cost(h, y)} \t')
37     return weight
```

```
1 import nltk
2 nltk.download('punkt')
3 stop_words = set(stopwords.words('english'))
4 email['Message'] = email['Message'].apply(lambda x: ' '.join([word for word in word_tokenize(x) if word not in stop_words]))
5
```

```
1 vectorizer = TfidfVectorizer()
2 X = vectorizer.fit_transform(email['Message']).toarray()
```

```
    return weight - learning_rate * gradient

```

```
            print(f'cost: {cost(h, y)} \t')
    return weight
```

```
import nltk
nltk.download('punkt')
stop_words = set(stopwords.words('english'))
email['Message'] = email['Message'].apply(lambda x: ' '.join([word for word in word_tokenize(x) if word not in stop_words]))

```

```
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(email['Message']).toarray()
```

```
X
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
y = np.array(email['Category'])
y

```

```
array([0, 0, 1, ..., 0, 0, 0])
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.2, random_state=42)

```

```
weight = np.zeros(train_X.shape[1])
```

EMAIL!

1111 UNREAD

INBOX

ATTACHMENTS

UNREAD

BUILDING MODEL

```
1 weight = train(train_X, train_y, weight, learning_rate)
2
```

```
iterations: 0 cost: 0.6931471805599453
iterations: 10 cost: 0.6799476784484476
iterations: 20 cost: 0.6672763290721779
iterations: 30 cost: 0.655109147740921
iterations: 40 cost: 0.6434228010737892
iterations: 50 cost: 0.6321946832139774
iterations: 60 cost: 0.6214029737056127
iterations: 70 cost: 0.6110266788546717
iterations: 80 cost: 0.6010456585033914
iterations: 90 cost: 0.5914406401610933
iterations: 100 cost: 0.5821932223759007
iterations: 110 cost: 0.5732858691218713
iterations: 120 cost: 0.5647018968323918
iterations: 130 cost: 0.5564254555480103
iterations: 140 cost: 0.5484415054767838
iterations: 150 cost: 0.5407357900963782
iterations: 160 cost: 0.533294806765663
iterations: 170 cost: 0.5261057756634268
iterations: 180 cost: 0.5191566077353799
iterations: 190 cost: 0.5124358722088913
iterations: 200 cost: 0.5059327641280194
iterations: 210 cost: 0.49963707226881743
iterations: 220 cost: 0.49353914771570867
iterations: 230 cost: 0.48762987331273555
iterations: 240 cost: 0.48190063414744444
```

```
iterations: 250 cost: 0.47634328917873914
iterations: 260 cost: 0.4709501440820046
iterations: 270 cost: 0.4657139253539668
iterations: 280 cost: 0.46062775569506026
iterations: 290 cost: 0.4556851306675572
iterations: 300 cost: 0.4508798966125283
iterations: 310 cost: 0.44620622979711555
iterations: 320 cost: 0.44165861675497897
iterations: 330 cost: 0.4372318357765696
iterations: 340 cost: 0.43292093950164834
iterations: 350 cost: 0.4287212385638034
iterations: 360 cost: 0.42462828623531645
iterations: 370 cost: 0.4206378640203059
iterations: 380 cost: 0.4167459681444243
iterations: 390 cost: 0.4129487968903194
iterations: 400 cost: 0.40924273872944167
iterations: 410 cost: 0.405624361202464
iterations: 420 cost: 0.402090400502495
iterations: 430 cost: 0.3986377517173175
iterations: 440 cost: 0.3952634596890193
iterations: 450 cost: 0.391964710451559
iterations: 460 cost: 0.3887388232089747
iterations: 470 cost: 0.38558324281908324
iterations: 480 cost: 0.3824955327496061
iterations: 490 cost: 0.37947336847567437
iterations: 500 cost: 0.3765145312896055
```

```
iterations: 740 cost: 0.32013005335932315
iterations: 750 cost: 0.31825193742482216
iterations: 760 cost: 0.31640310351697554
iterations: 770 cost: 0.3145828103456778
iterations: 780 cost: 0.3127903423340411
iterations: 790 cost: 0.31102500849959114
iterations: 800 cost: 0.30928614139309135
iterations: 810 cost: 0.30757309609162353
iterations: 820 cost: 0.3058852492427687
iterations: 830 cost: 0.3042219981569364
iterations: 840 cost: 0.30258275994507683
iterations: 850 cost: 0.3009669706991882
iterations: 860 cost: 0.2993740847131934
iterations: 870 cost: 0.29780357374191224
iterations: 880 cost: 0.2962549262959996
iterations: 890 cost: 0.29472764697084924
iterations: 900 cost: 0.2932212558075897
iterations: 910 cost: 0.29173528768440976
iterations: 920 cost: 0.2902692917365639
iterations: 930 cost: 0.28882283080350263
iterations: 940 cost: 0.2873954809016707
iterations: 950 cost: 0.28598683072160214
iterations: 960 cost: 0.28459648114802155
iterations: 970 cost: 0.28322404480174107
iterations: 980 cost: 0.28186914560220966
iterations: 990 cost: 0.28053141834964135
```

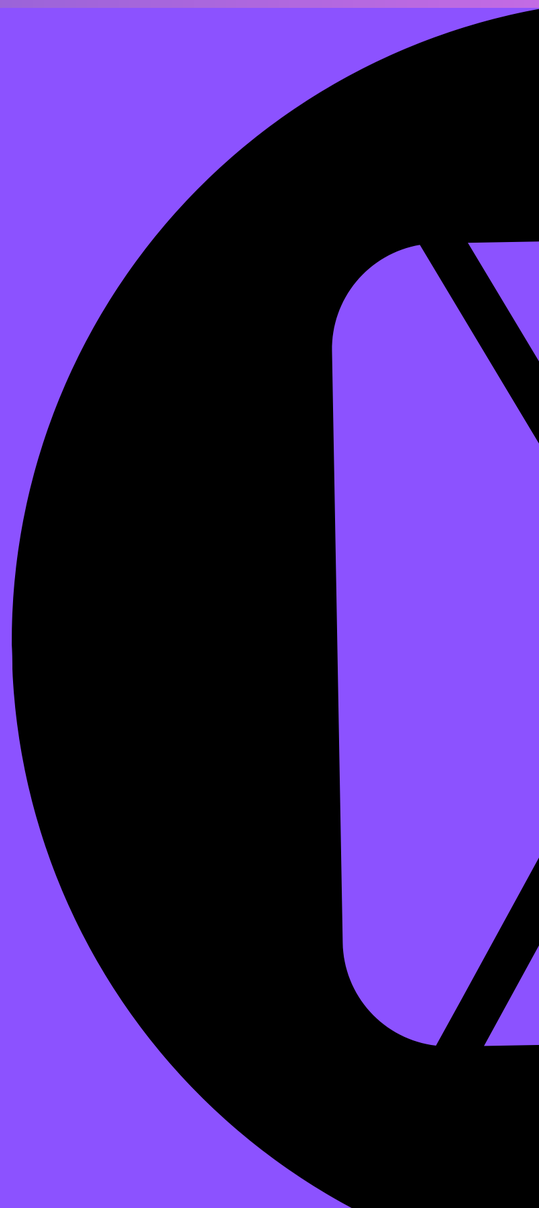
EMAIL!

1111 UNREAD

INBOX ATTACHMENTS UNREAD

```python
predictions = predict(train_X, weight) >= 0.5

accuracy = np.mean(predictions == train_y) * 100
print(f"Accuracy: {accuracy}%")

```

```
Accuracy: 96.92121212121212%
```



EMAIL!

1111 UNREAD

INBOX

ATTACHMENTS

UNREAD

YOUR PARAGRAPH TEXT
YOUR PARAGRAPH TEXT

THANK YOU!

THANK YOU!