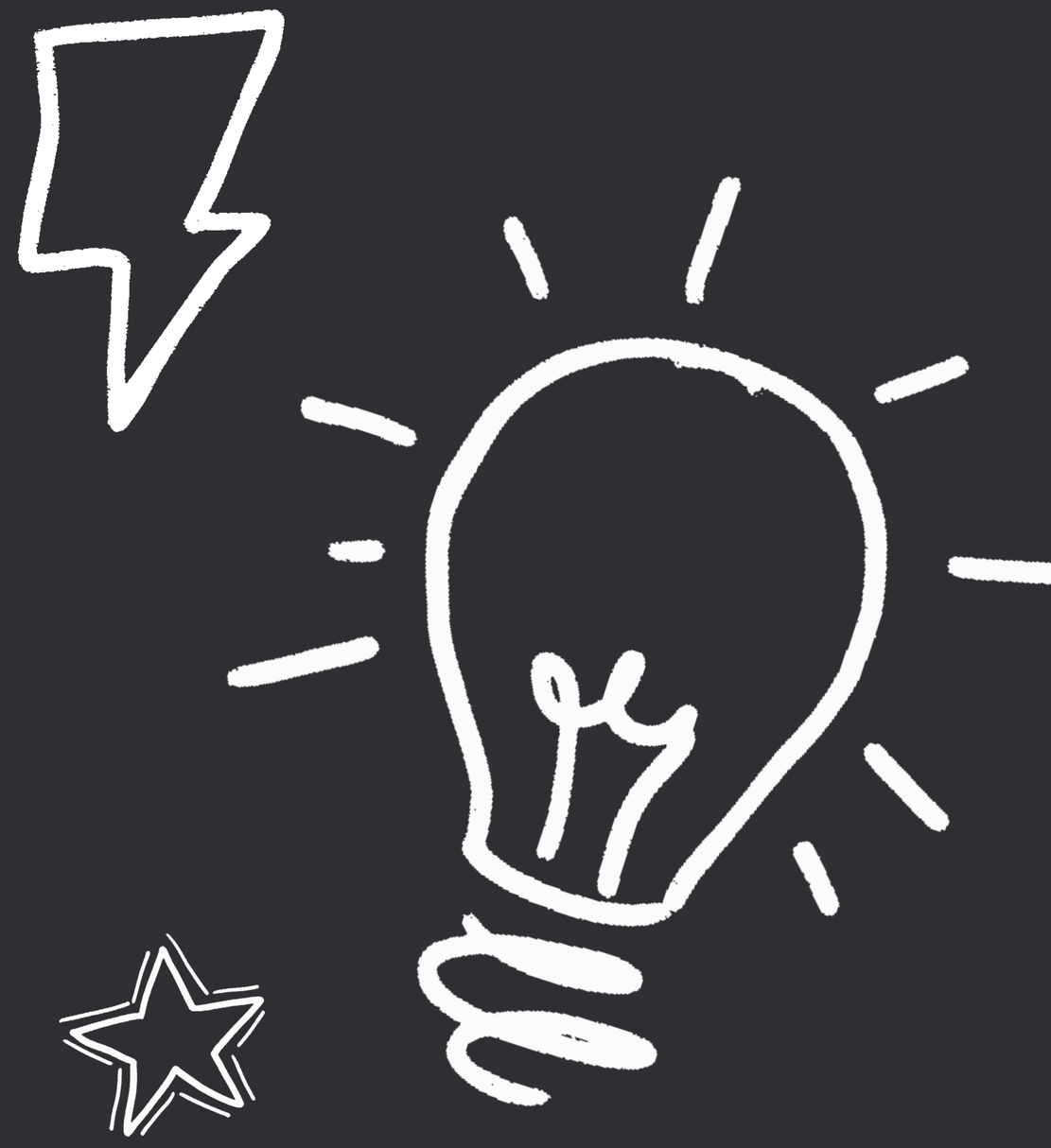
SC1015
Mini Project



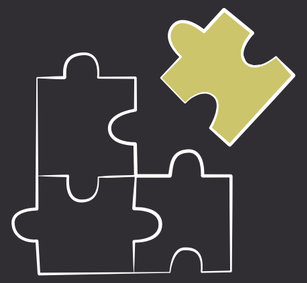
ACDA1 | TEAM 1

Ganesh Rudra Prasadh (U2223543J)

Kow Zi Ting (U2310485B)

Law Yu Chen (U2310175B)

Table of Contents



Problem Definition

Exploratory Data Analysis

Data Preparation

Machine Learning Techniques

Data Driven Insights & Recommendations

INTRODUCTION

Introduction

Have you ever written a cheque?

Introduction

1 in 5

cashless payments are done by cheque in the **United States** (by value)

Source: Federal Bank of Atlanta

Introduction

1 in 3

cashless payments are done by cheque in **Singapore** (by value)

Source: Federal Bank of Atlanta

Heavy reliance on manpower for manual cheque data entry

PROBLEM DEFINITION

Problem Definition

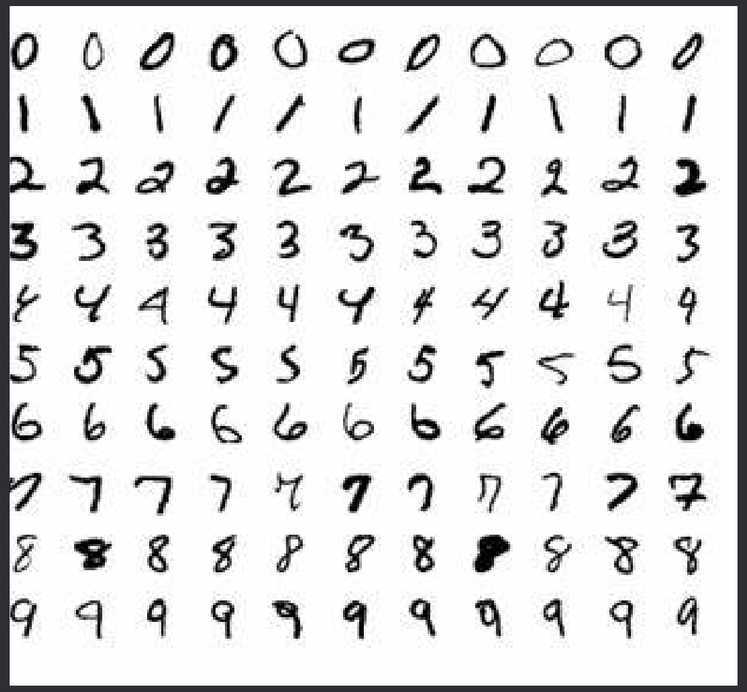
Leverage machine learning to expedite cheque verification and checking process

Problem Definition

Categorical Prediction

Classify handwritten numbers into their correct categories 0-9

Our Dataset



MNIST Dataset - Kaggle

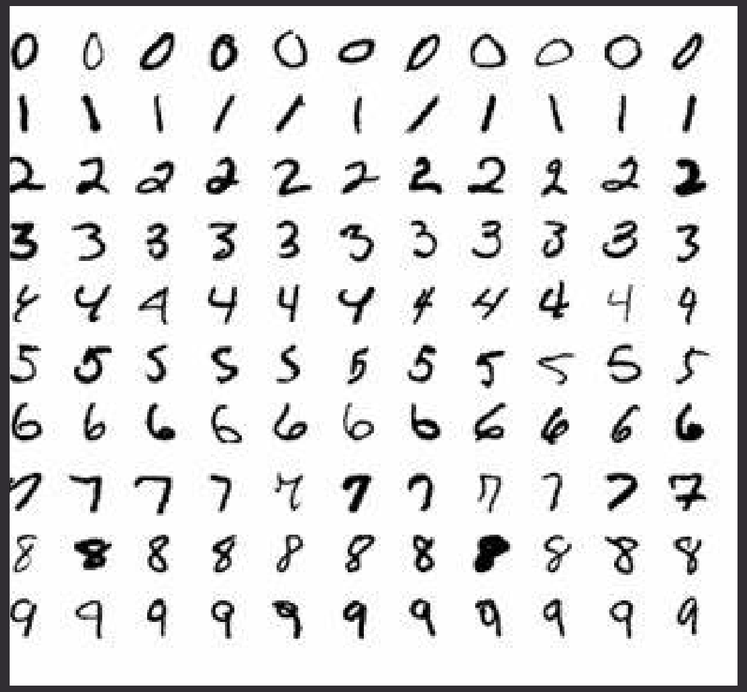
MNIST

Modified National Institute of Standards and Technology

ORIGIN

Based in the United States

Our Dataset



MNIST Dataset - Kaggle

Format

CSV File
contributed by Kaggler
Dariel Dato-on

Size of DS

70,000
unique handwriting data

Content of DS

Number Label

Pixel Values

EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60000 entries, 0 to 59999
Columns: 785 entries, label to 28x28
dtypes: int64(785)
memory usage: 359.3 MB
```

Summary Statistics

Middle Pixels

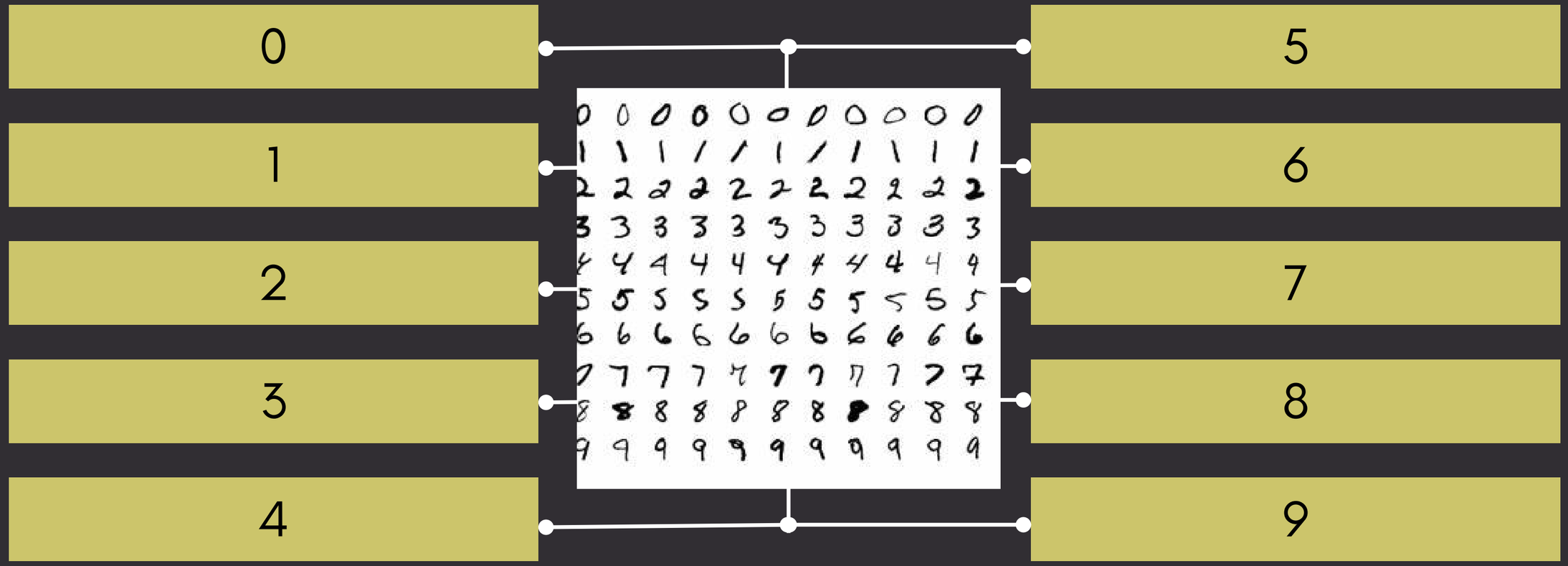
	1x1	1x2
count	60000.0	60000.0
mean	0.0	0.0
std	0.0	0.0
min	0.0	0.0
25%	0.0	0.0
50%	0.0	0.0
75%	0.0	0.0
max	0.0	0.0

Extremity Pixels

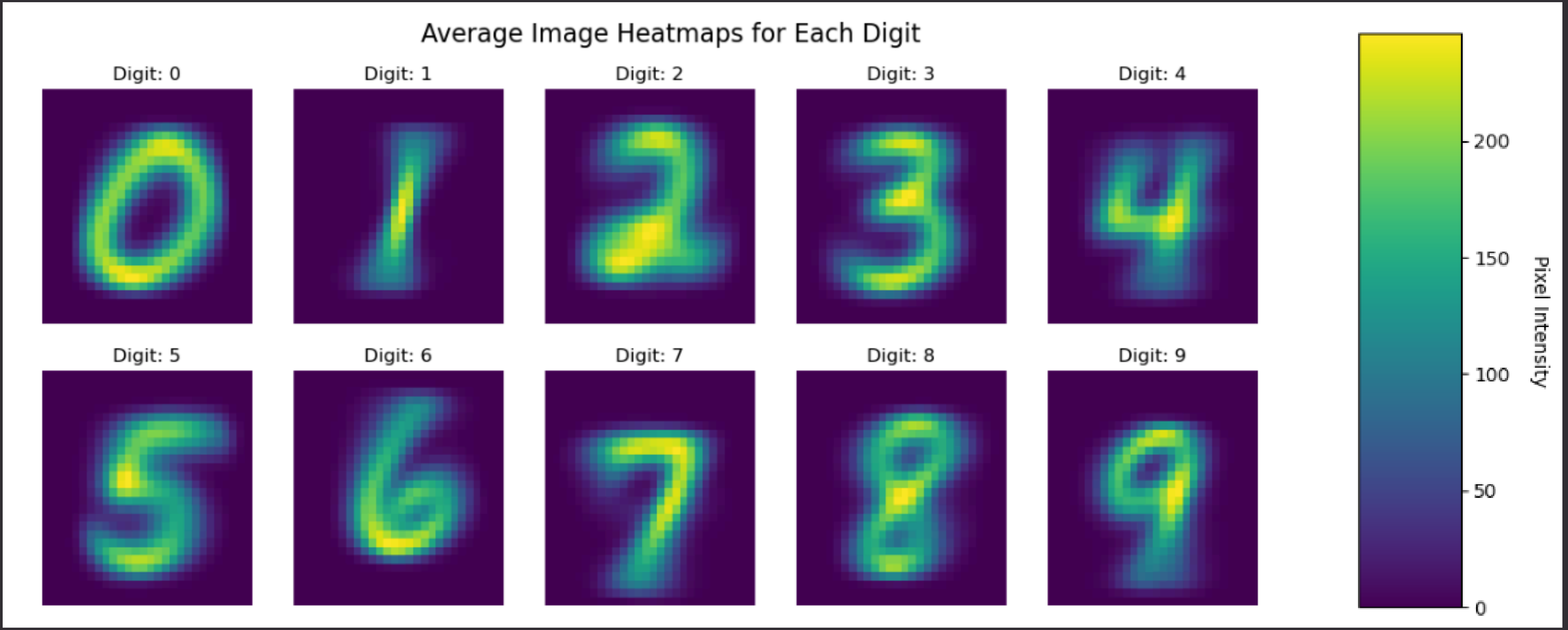
	15x16	15x17
count	60000.000000	60000.000000
mean	139.553600	137.100633
std	109.522758	109.840158
min	0.000000	0.000000
25%	0.000000	0.000000
50%	172.000000	164.000000
75%	253.000000	253.000000
max	255.000000	255.000000

DATA VISUALISATION

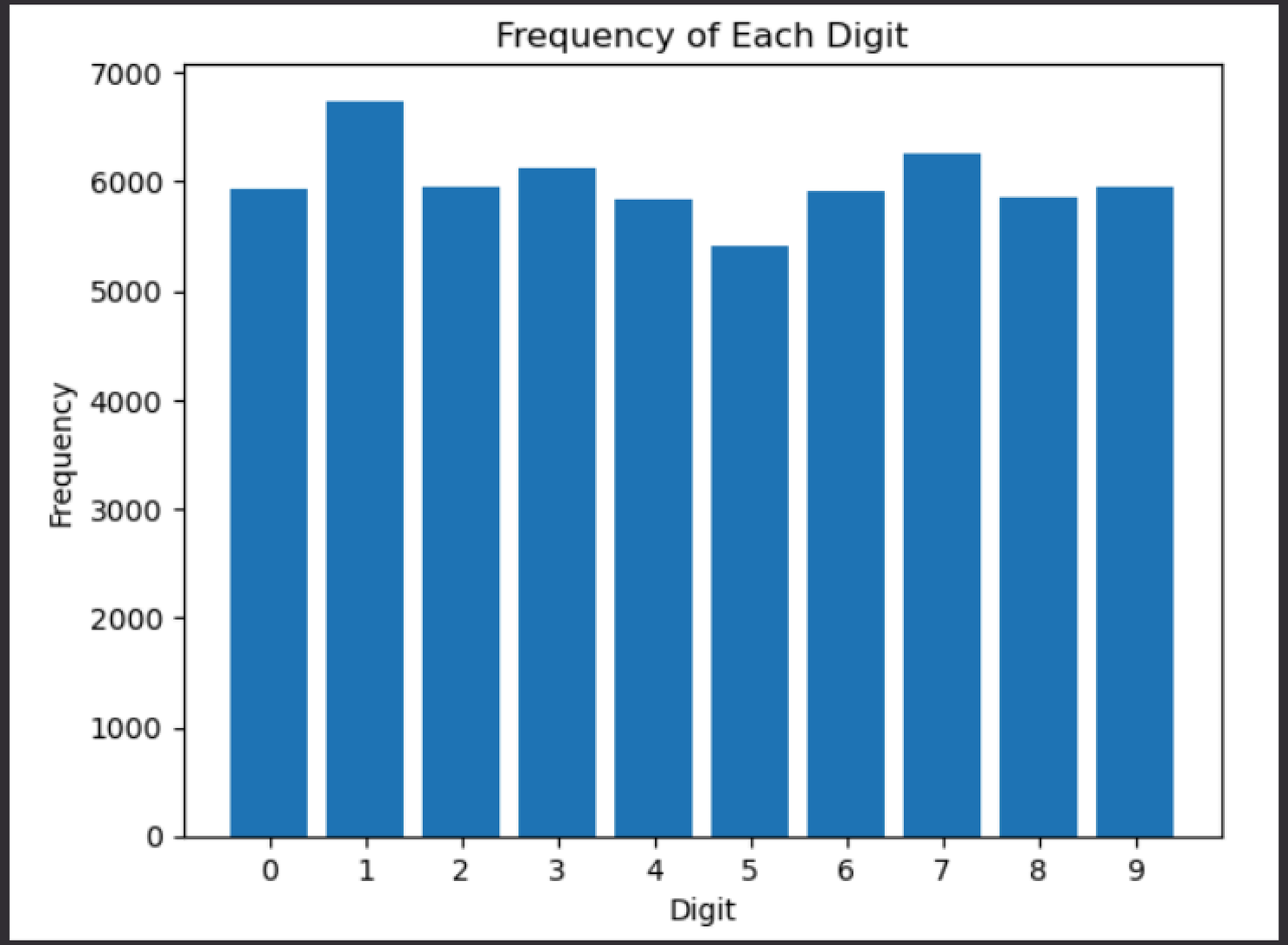
Heatmap



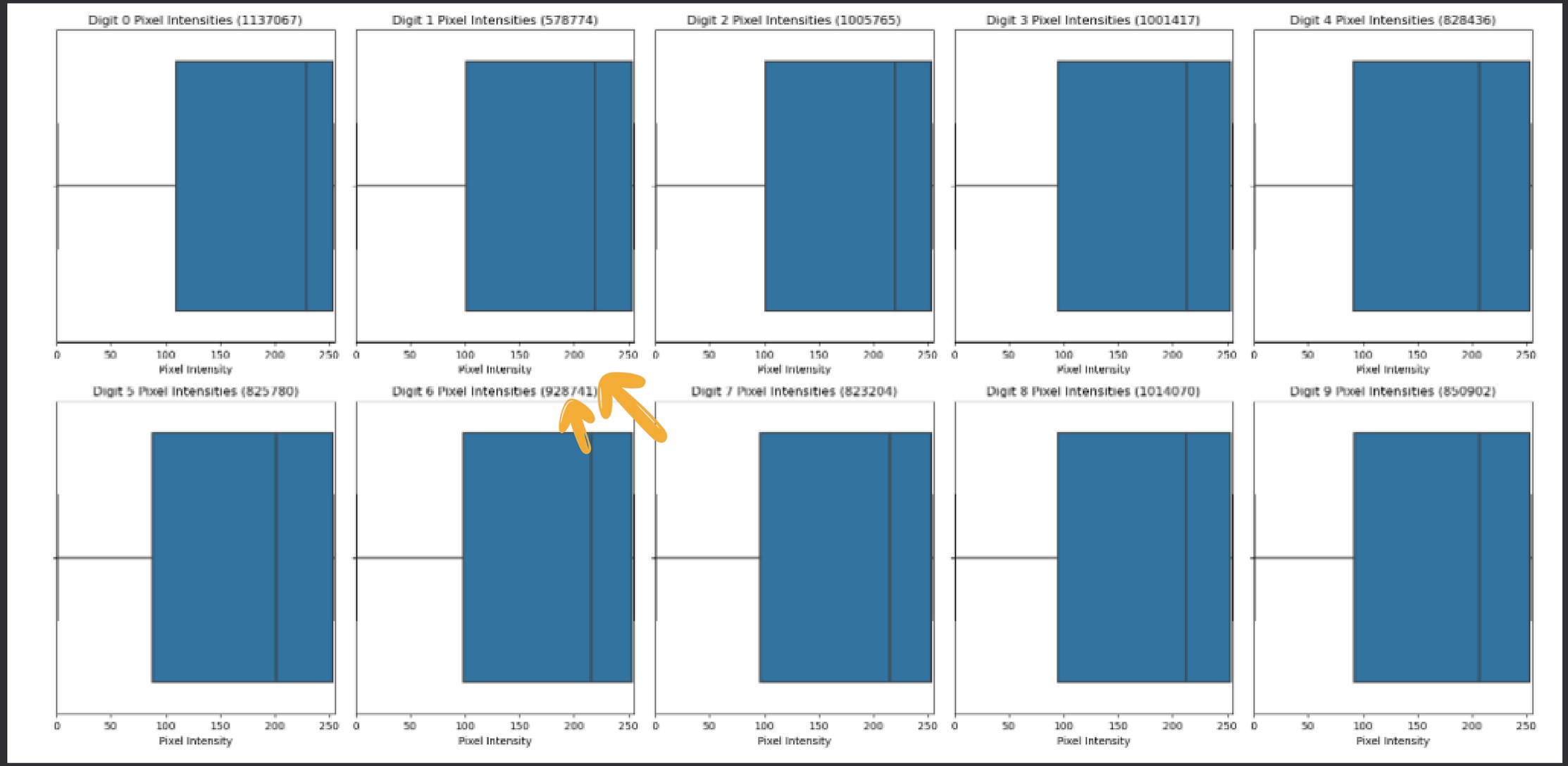
Heatmap


Average Image Heatmaps for Each Digit
Digit: 0
Digit: 1
Digit: 2
Digit: 3
Digit: 4
Digit: 5
Digit: 6
Digit: 7
Digit: 8
Digit: 9
200
150
100
50
0
Pixel Intensity

Bar Chart

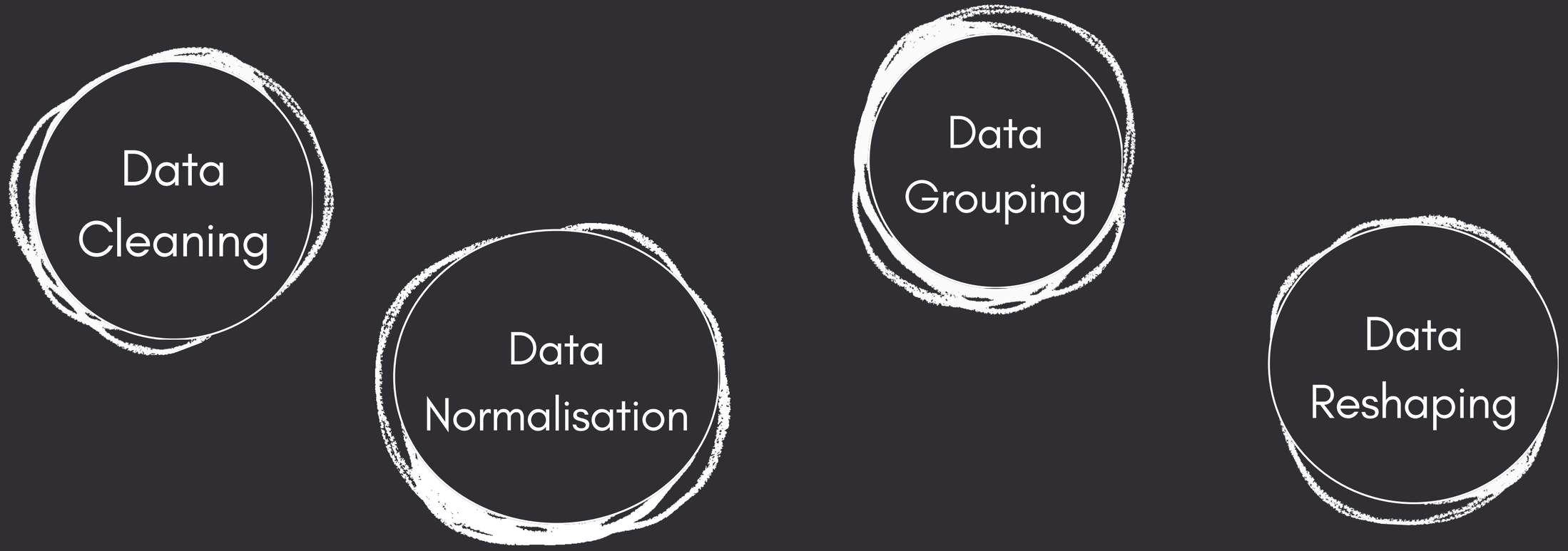

Frequency of Each Digit
Frequency
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
6
7
8
9
Digit

Univariate Box Plots

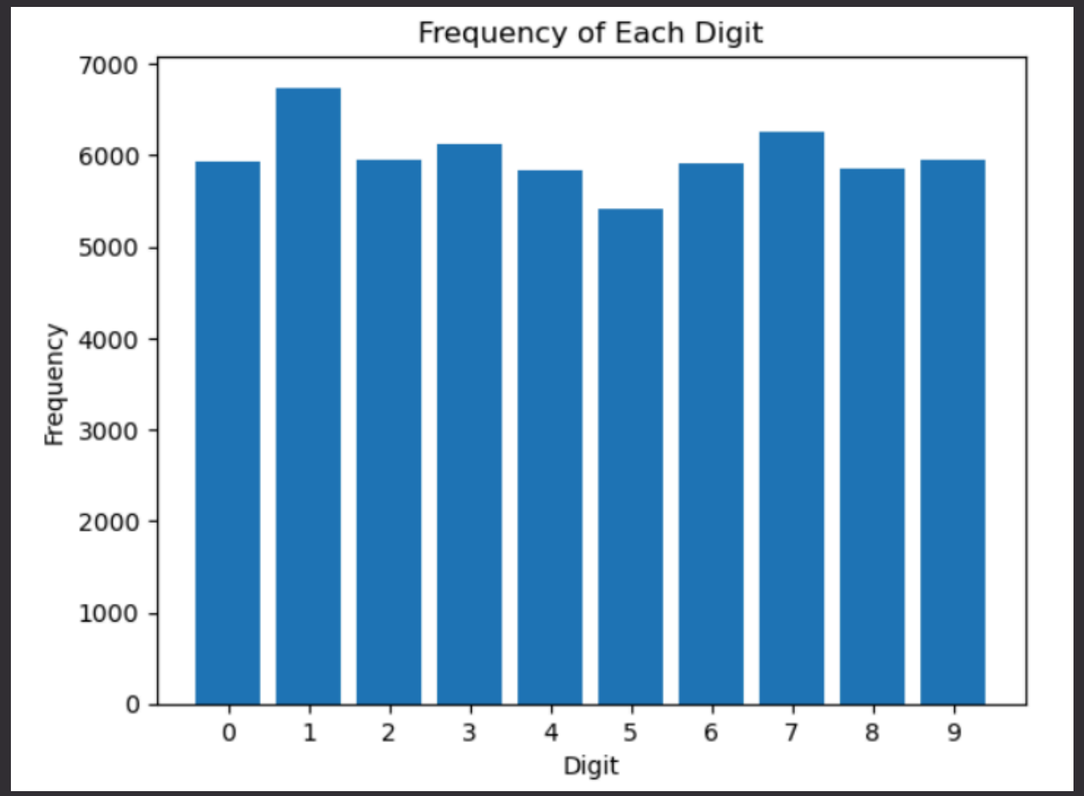


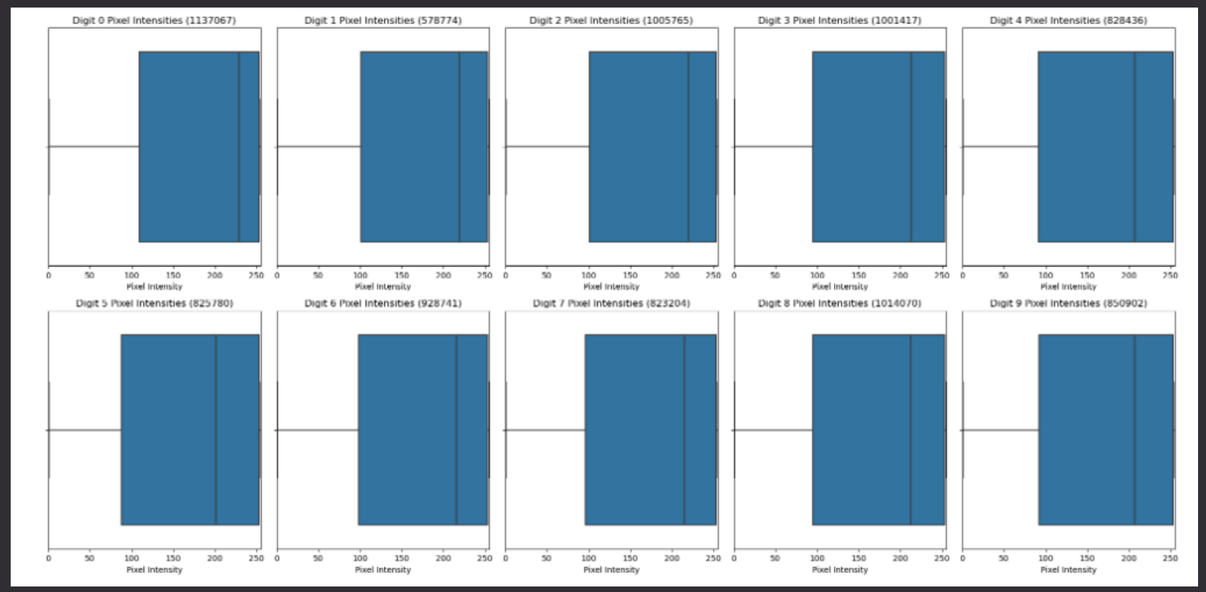
DATA PREPARATION

Data Preparation



Data Cleaning





Frequency of occurrence is generally equal

No outliers

Data Cleaning

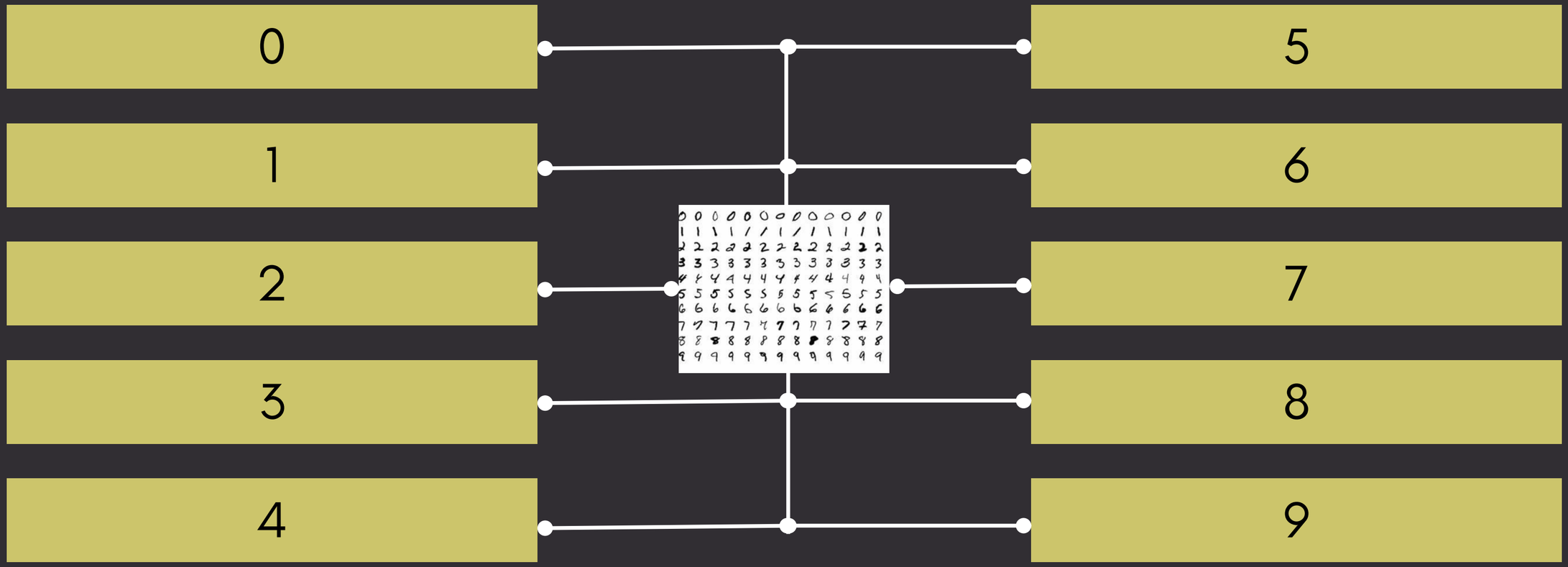
There is no need to clean.

Data Normalisation

```
# Normalizing the RGB codes by dividing it to the max RGB value(255)
X_train = X_train / 255
```

Dividing pixel values by the maximum RGB value (255) so the outputs fall withinm the range of 0 and 1.

Data Grouping



Data Reshaping

```
def one_hot_encoding(y, max):
    result = np.eye(max)[y]
    return result

def one_hot_decoding(y):
    result = np.argmax(y, axis=0).reshape((1,-1))
    return result


max = np.unique(y_train).size
y_train = one_hot_encoding(y_train, max)
```

MACHINE LEARNING

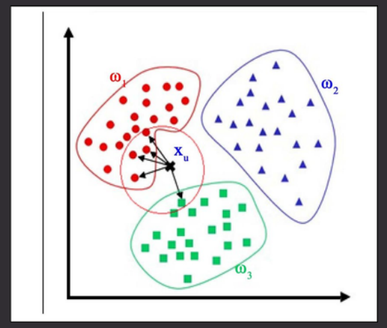
Machine Learning

Data Science Problem Type:

Classification

Decision Tree

Logistic Regression



K-Nearest Neighbours

Convoluted Neural Networks (CNN)

Machine Learning

Decision Tree

Train and Test Sets

mnist_train.csv
mnist_test.csv
(from Kaggle)

Best Depth

18

Overall Accuracy

88.29%

Machine Learning

Decision Tree

Accuracy for Respective Digits

```
Accuracy for digit 0: 0.9408
Accuracy for digit 1: 0.9683
Accuracy for digit 2: 0.8663
Accuracy for digit 3: 0.8624
Accuracy for digit 4: 0.8839
Accuracy for digit 5: 0.8374
Accuracy for digit 6: 0.8810
Accuracy for digit 7: 0.9086
Accuracy for digit 8: 0.7957
Accuracy for digit 9: 0.8672
```

Machine Learning

Logistic Regression

Train and Test Sets

mnist_train.csv
mnist_test.csv
(from Kaggle)

Type

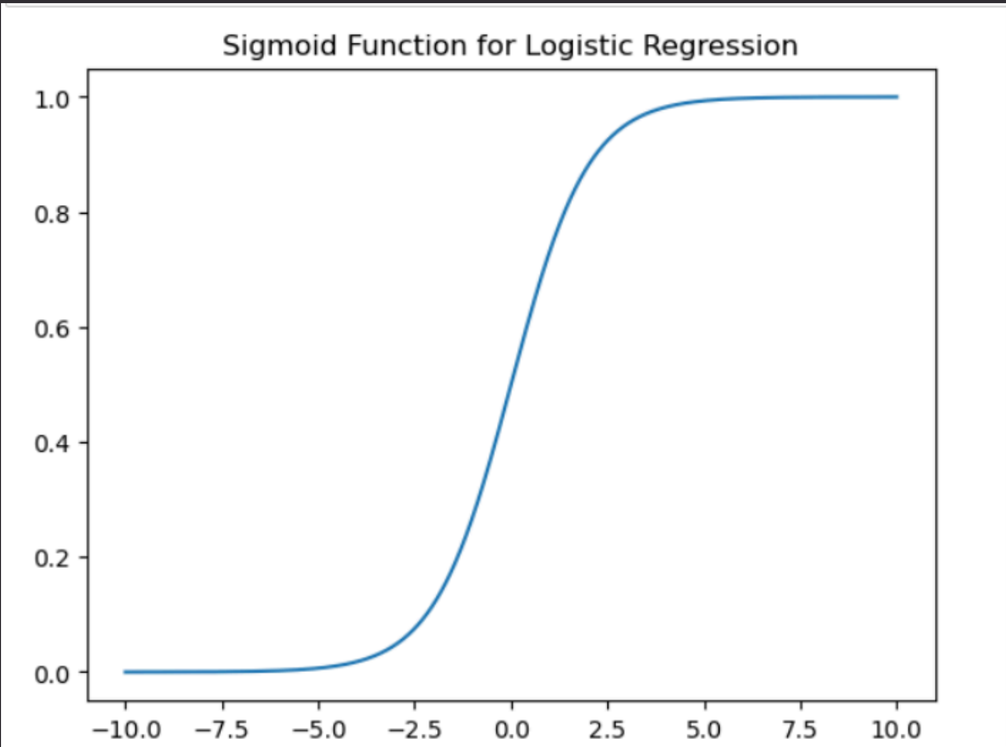
Multinomial
Logistic Regression

Overall Accuracy

92.07%

Logistic Regression

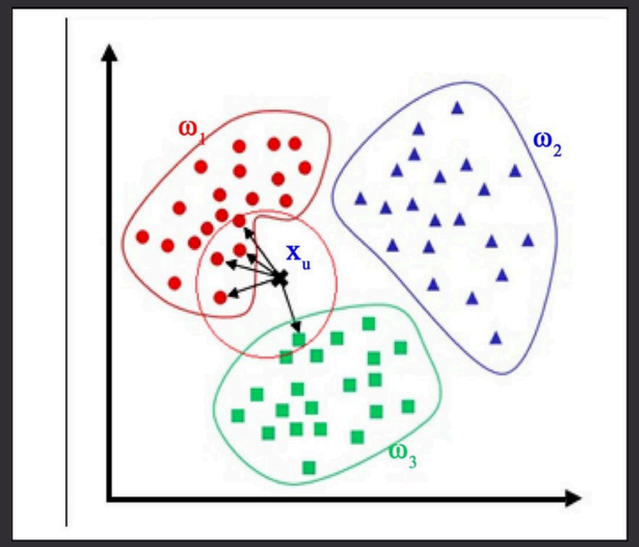
Sigmoid Function



Accuracy for Respective Digits

```
Accuracy for digit 0: 0.9735
Accuracy for digit 1: 0.9789
Accuracy for digit 2: 0.8876
Accuracy for digit 3: 0.9109
Accuracy for digit 4: 0.9257
Accuracy for digit 5: 0.8509
Accuracy for digit 6: 0.9457
Accuracy for digit 7: 0.9193
Accuracy for digit 8: 0.8881
Accuracy for digit 9: 0.9138
```

Machine Learning



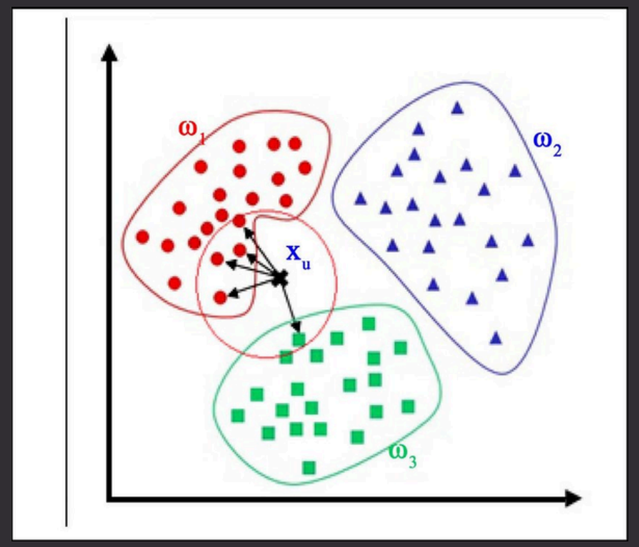
K-Nearest Neighbours

Train and Test Sets — mnist_train.csv, mnist_test.csv (from Kaggle)

No of Neighbours, k — 5

Overall Accuracy — 96.88%

Machine Learning



K-Nearest Neighbours

Accuracy for Respective Digits

```
Accuracy for digit 0: 0.9939
Accuracy for digit 1: 0.9982
Accuracy for digit 2: 0.9603
Accuracy for digit 3: 0.9663
Accuracy for digit 4: 0.9613
Accuracy for digit 5: 0.9664
Accuracy for digit 6: 0.9864
Accuracy for digit 7: 0.9611
Accuracy for digit 8: 0.9374
Accuracy for digit 9: 0.9534
```

Machine Learning

CNN

1. Data Preparation
2. Model Initialization
3. Forward Propagation
4. Loss Computation
5. Backward Propagation
6. Parameter Update
7. Iterative Training

Machine Learning

→ Data Preparation

```
X_train = X_train / 255
```

```
y_train = one_hot_encoding(y_train, max)
```

CNN

Machine Learning

→ Model Initialization

```python
def init_params(nx, nh, ny):
    W1 = np.random.normal(0, 0.3, (nh, nx + 1))
    W2 = np.random.normal(0, 0.3, (ny, nh + 1))
    params = {'W1': W1, 'W2': W2}
    return params
```

```python
# Initialize the network parameters
params = init_params(X.shape[1], hidden, y.shape[1])
```

CNN

Machine Learning

CNN

→ Forward Propagation

```python
def forward(params, X):
    W1 = params['W1']
    W2 = params['W2']

    # add bias column because the matrix will be transposed
    ones_vector = np.ones(X.shape[0])
    A0 = np.insert(X, 0, ones_vector, axis=1)
    Z1 = np.matmul(W1, A0.transpose())
    A1 = tanh(Z1)

    # add bias row because the matrix won't be transposed
    ones_vector = np.ones(A1.shape[1])
    A1 = np.insert(A1, 0, ones_vector, axis=0)
    Z2 =np.matmul(W2, A1)
    A2 = softmax(Z2)

    outputs = {'A0': A0, 'Z1': Z1, 'A1': A1, 'Z2': Z2, 'A2': A2}
    return outputs
```

```python
outputs = forward(params, x_batch)
```

Machine Learning

CNN

→ Loss Computation

```python
def loss_accuracy(y_pred, y):
    loss = -np.mean(np.sum(y * np.log(y_pred), axis=1))
    accuracy = np.mean(np.argmax(y, axis=1) == np.argmax(y_pred, axis=1))
    return loss, accuracy
```

```python
    # Compute loss and accuracy
    loss, accuracy = loss_accuracy(outputs['A2'].T, y_batch)
```

Machine Learning

→ Backward Propagation

CNN

```python
def backward(X, params, outputs, Y):
    X = np.vstack([np.ones(X.shape[0]), X.T])

    d2 = outputs['A2'] - Y.T
    dW2 = np.matmul(d2, outputs['A1'].T)

    d1 = np.matmul(params['W2'].T[1:], d2) * tanh_derivative(outputs['Z1'])
    dW1 = np.matmul(d1, X.T)

    grads = {'dW1': dW1, 'dW2': dW2}
    return grads
```

```python
    # Backward pass
    grads = backward(x_batch, params, outputs, y_batch)
```

Machine Learning

CNN

→ Parameter Update

```python
def sgd(params, grads, eta):
    params['W1'] -= eta * grads['dW1']
    params['W2'] -= eta * grads['dW2']
    return params
```

```python
# Update the parameters
params = sgd(params, grads, learning_rate)
```

Iterative Training

CNN

Train and Test Sets

mnist_train.csv
mnist_test.csv
(from Kaggle)

No of epochs

50

Highest Accuracy

97.92%

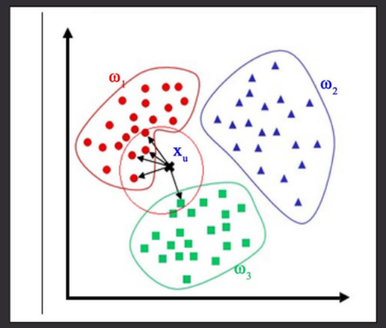
DATA DRIVEN INSIGHTS

and recommendations :)

Comparison of ML Accuracies

Convoluted Neural Networks (CNN)

>



K-Nearest Neighbours

>

Logistic Regression

>

Decision Tree

Comparison of ML Accuracies

KNN

DT

Recommendations

Convoluted Neural Networks

Recommendations

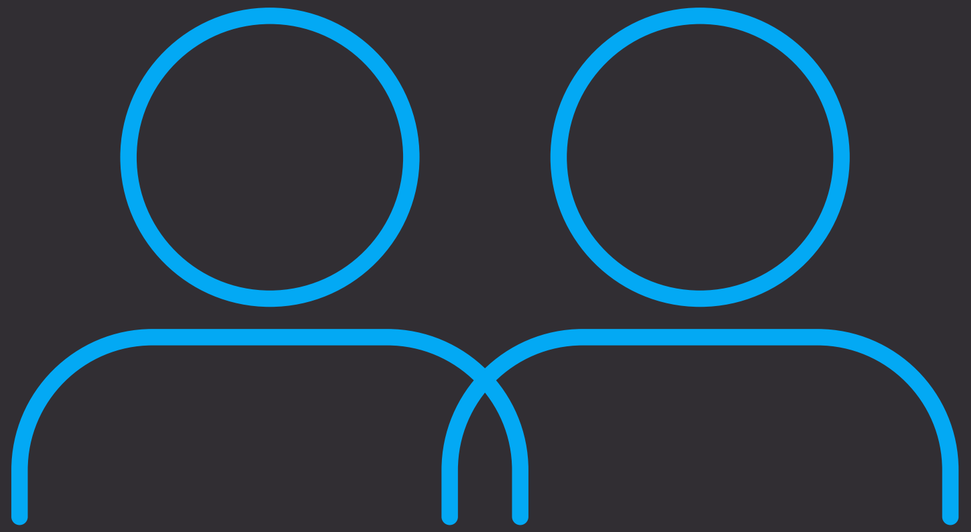
Convoluted Neural Networks

Human Checks

Recommendations

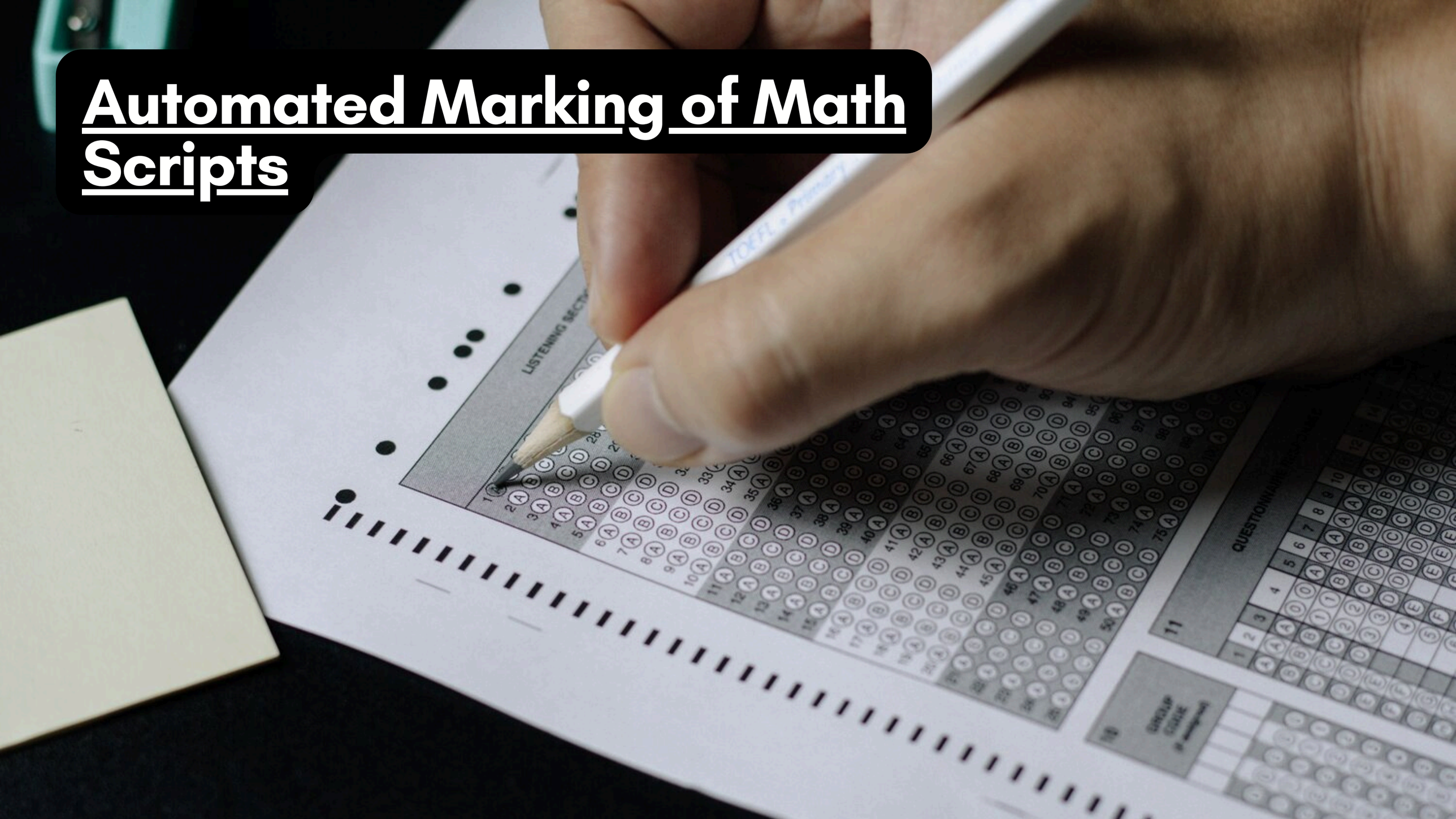
Human Vigilance

Recommendations



Human Vigilance

Automated Marking of Math Scripts



Recognising Price Signs

CONCLUSION

CONCLUSION

SEE YOU NEXT TIME :)