Introduction

This project aims to design and develop a machine learning application which is capable of classifying SMS messages as either malicious messages (**Spam**) or real/legit messages (**Ham**), due to the growth of mobile communication SMS has become one of the primary targets for phishing attacks and unprompted marketing. This project aims to filter these types of messages in order to make user experience secure and solicit.

The project uses **Supervised Machine Learning** in which the model learns to map input data(Features) to output labels(Target Variable) based on the labelled dataset. For the project following concepts will be utilized:

- Natural Language Processing (NLP):
    - It is the field of AI focused on helping computers understand human language.
    - The use of NLP in this project is to clean the data (raw text) and convert it to numerical format for the computer to understand.
- Vectorization:
    - It is the process of converting non numerical data into numerical vectors for processing.
    - For the project TF-IDF (Term Frequency-Inverse Document Frequency) will be used to transform text into numerical vectors.
    - TF-IDF‘s main concept is to assign weight based on their importance.
- Classification Algorithm:
    - They are tools in ML which sorts data into categories or classes.
    - The project uses 3 distinct types of learning algorithms for classification model:
        - Naïve Bayes (Probabilistic)
        - Logistic Regression (Linear)
        - KNN K-Nearest Neighbour (Instance based)

Problem Domain

The **SMS Spam Detection** is a type of binary classification problem. The input is a string of text or sentences (SMS message) and the output is a binary label in our case we assign "0 for Ham" and "1 for Spam". The unstructured structure of the SMS data is a challenge as it contains less characters more often than not messages containing abbreviation, slangs, and misspelling with the addition of lack of metadata.

Dataset Description

Dataset used: **UCI SMS Spam Collection**

The project utilizes the UCI SMS Spam Collection which is a public dataset sourced from **University of California, Irvine's (UCI) Machine Learning Repository**.

Here are some facts that we can get from the dataset: (shown in figure 1)

- Type: .txt (the original dataset is in txt file format)
- No of rows of data: 5572
- Data Type: Unstructured English data (object)
- Class Imbalance: Heavily Imbalanced
    - Ham=4824 (86.591276%) ,
    - Spam=747 (13.408724%)
- Implications: Due to the heavy class imbalance the model may have less accuracy so Precision and Recall metrices will be more important than accuracy during the evaluation of the model.

Societal or Business Relevance

- Cybersecurity: SMS phishing is a major threat in current time as it is used to steal banking details or other private information, this classifier is like the first line of defence against these types of threats.
- Business Integrity: Even now almost all top businesses and companies such as X, Meta etc rely on SMS for one time OTP but if due to constant spam email threats users stop trusting or using SMS it compromises security.
- User friendly: the spam filters automatically filters spam SMS saving time for the user and also decluttering mobile storage.

SMS_Spam_Detection

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
# df = pd.read_csv("sms.csv", encoding='utf-8')--- This doesnot work as in the dataset we have characters such as dollar sign, emoji's which u

df = pd.read_csv("sms.csv", encoding='latin-1')
```

[3]:
```
df.shape
```

[3]:
```
(5571, 2)
```

[4]:
```
df.size
```

[4]:
```
11142
```

[9]:
```
df.head(10)
```

[9]:

	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
0	ham	Ok lar... Joking wif u oni...
1	spam	Free entry in 2 a wkly comp to win FA Cup fina...
2	ham	U dun say so early hor... U c already then say...
3	ham	Nah I don't think he goes to usf, he lives aro...
4	spam	FreeMsg Hey there darling it's been 3 week's n...
5	ham	Even my brother is not like to speak with me. ...
6	ham	As per your request 'Melle Melle (Oru Minnamin...
7	spam	WINNER!! As a valued network customer you have...
8	spam	Had your mobile 11 months or more? U R entitle...
9	ham	I'm gonna be home soon and i don't want to tal...

[11]:
```
df['ham'].size
```

[11]:
```
5571
```

[13]:
```
df['Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...'].dtype
```

[13]:
```
dtype('O')
```

[15]:
```
print(df.dtypes)
```

```
ham                                                                 object
Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...    object
dtype: object
```

[17]:
```
print(df.info)
```

```
<bound method DataFrame.info of        ham  \
0      ham
1     spam
2      ham
3      ham
4     spam
...    ...
5566  spam
5567   ham
5568   ham
5569   ham
5570   ham

     Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
0                         Ok lar... Joking wif u oni...
1     Free entry in 2 a wkly comp to win FA Cup fina...
2     U dun say so early hor... U c already then say...
3     Nah I don't think he goes to usf, he lives aro...
4     FreeMsg Hey there darling it's been 3 week's n...
...                                                 ...
5566  This is the 2nd time we have tried 2 contact u...
5567              Will  b going to esplanade fr home?
5568  Pity, * was in mood for that. So...any other s...
5569  The guy did some bitching but I acted like i'd...
5570                       Rofl. Its true to its name

[5571 rows x 2 columns]>
```

[ ]:
```
```

[20]:
```
print(df['ham'].value_counts())
```

```
ham
ham     4824
spam     747
Name: count, dtype: int64
```

[24]:
```
print(df['ham'].value_counts(normalize=True)*100)
```

```
ham
ham     86.591276
spam    13.408724
Name: proportion, dtype: float64
```

[ ]:
```
```

Solution

The solution developed for this implements a full machine learning pipeline:

- Text Preprocessing
- Feature Extraction
- ML Algorithms

Text Preprocessing: The raw data (text) is noisy so we apply the following techniques:

- Tokenization: Breaks sentences into words.
- Stop Word Removal: removes words like is, the, that, etc.
- Normalization: lowercases the text
- Stemming/Lemmatization: convert word to root form

Feature Extraction: Since the data is in textual form, we need to convert it to numerical form and for that following techniques is to be used:

- TF-IDF Vectorization: converts text into numerical vectors. (This method was chosen because it normalizes count of words which prevents longer messages from having unfair weightage.)

Machine Learning Algorithm: Following machine learning algorithm are to be used:

- Multinomial Naïve Bayes:
    - assumes independence between features
    - Speed and high performance on high dimensionality data.
    - Uses techniques like Laplace smoothing to handle unseen words preventing zero probabilities
- Logistic Regression:
    - Learns by creating linear decision boundary between classes.
    - Uses sigmoid function.
    - Provides probabilities for prediction of class.
- KNN (K Nearest Neighbour):
    - Classifies based on distance similarity.
    - Instance based supervised machine learning algorithm.

Pseudocode

Pseudocode for Naïve Bayes:

```
PROCESS NaiveBayes
  READ Dataset
  SET SpamMessageCount TO 0
  SET HamMessageCount TO 0

  INITIALIZE SpamWordCount AS empty dictionary
  INITIALIZE HamWordCount AS empty dictionary
  INITIALIZE Vocabulary AS empty set

  SET TotalSpamWords TO 0
  SET TotalHamWords TO 0

  FOR EACH Record IN Dataset
    GET Message
    GET Label

    IF Label IS "spam" THEN
      INCREMENT SpamMessageCount
    ELSE
      INCREMENT HamMessageCount
    END IF

    Tokenize Message INTO Words

    FOR EACH Word IN Words
      ADD Word TO Vocabulary
```

```
            IF Label IS "spam" THEN
               INCREMENT SpamWordCount[Word]
               INCREMENT TotalSpamWords
            ELSE
               INCREMENT HamWordCount[Word]
               INCREMENT TotalHamWords
            END IF
         END FOR
      END FOR

      CALCULATE PriorSpam AS SpamMessageCount / TotalMessages
      CALCULATE PriorHam AS HamMessageCount / TotalMessages

      STORE ModelParameters
   END PROCESS
```

Pseudocode for logistic regression:

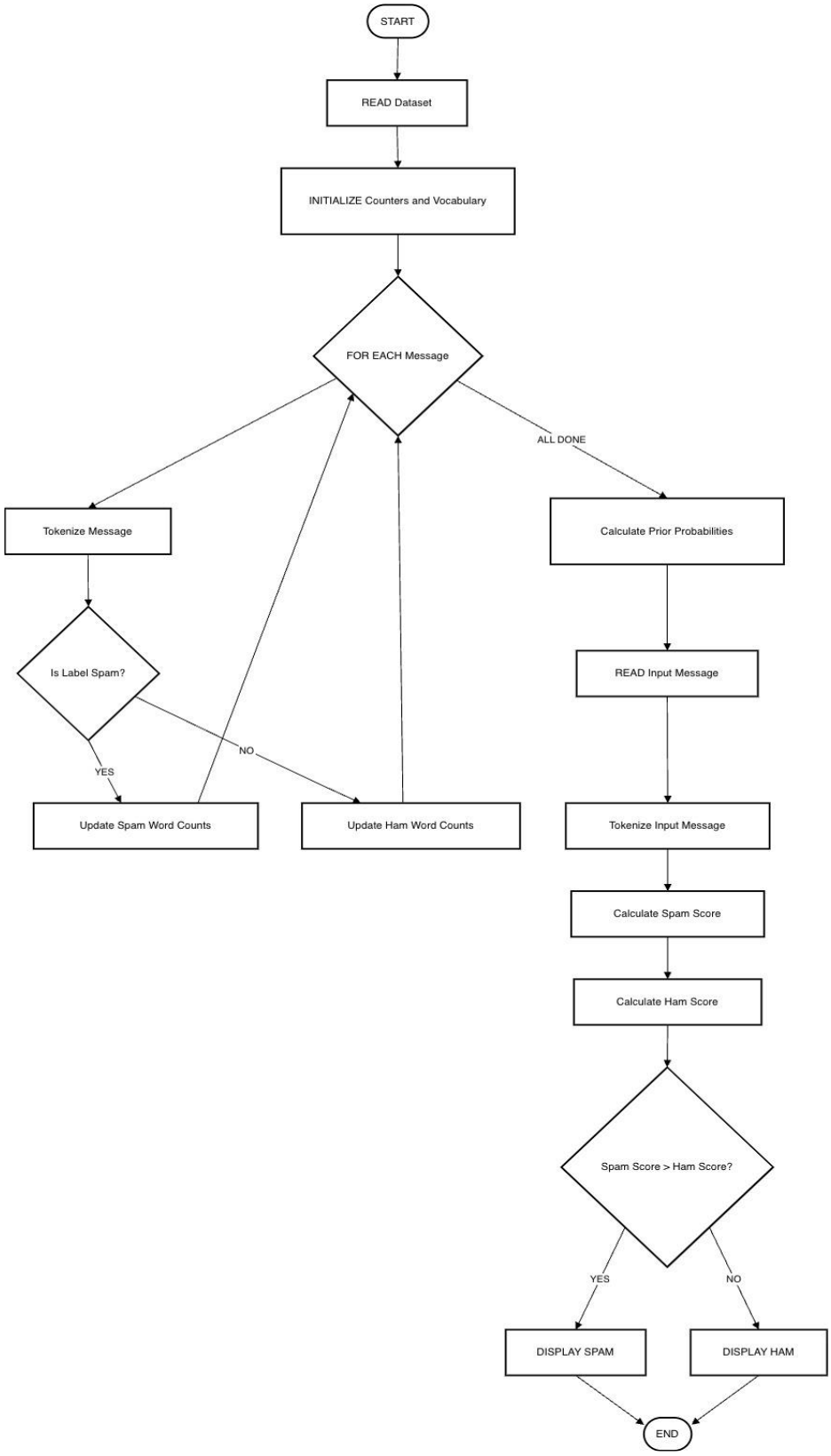
```
PROCESS LogisticRegression
  READ Dataset
  READ LearningRate
  READ NumberOfIterations

  INITIALIZE Weight TO 0
  INITIALIZE Bias TO 0

  SET NumberOfSamples TO size of Dataset

  FOR Iteration FROM 1 TO NumberOfIterations
    SET WeightGradient TO 0
    SET BiasGradient TO 0

    FOR EACH Record IN Dataset
      GET InputValue
      GET ActualLabel

      CALCULATE LinearOutput AS (Weight * InputValue) + Bias
      CALCULATE PredictedProbability AS Sigmoid(LinearOutput)

      CALCULATE Error AS PredictedProbability - ActualLabel

      ADD Error * InputValue TO WeightGradient
      ADD Error TO BiasGradient
    END FOR
```

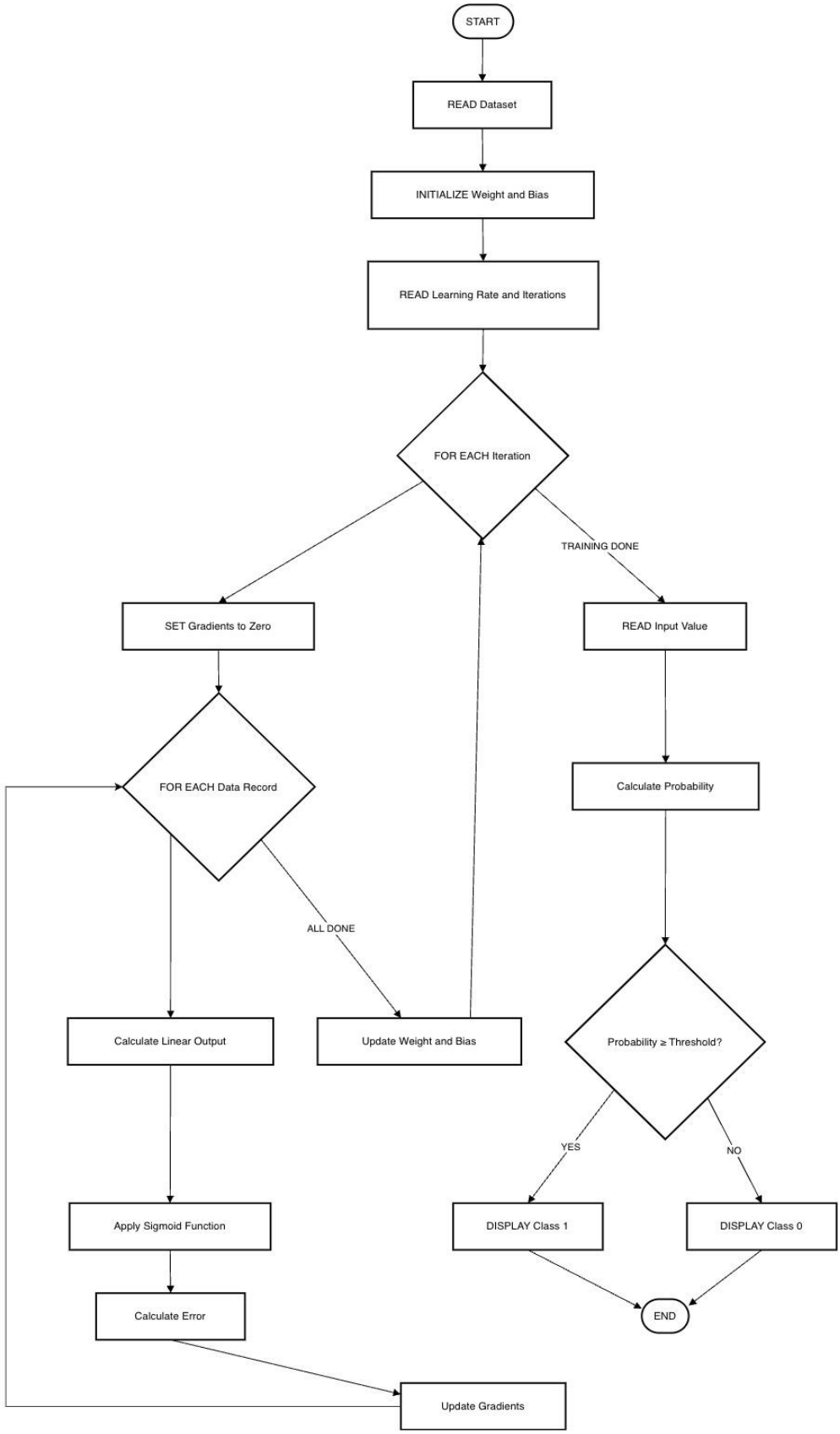
```
        CALCULATE WeightGradient AS WeightGradient / NumberOfSamples
        CALCULATE BiasGradient AS BiasGradient / NumberOfSamples

        UPDATE Weight AS Weight - (LearningRate * WeightGradient)
        UPDATE Bias AS Bias - (LearningRate * BiasGradient)
    END FOR

    STORE Weight
    STORE Bias
END PROCESS
```

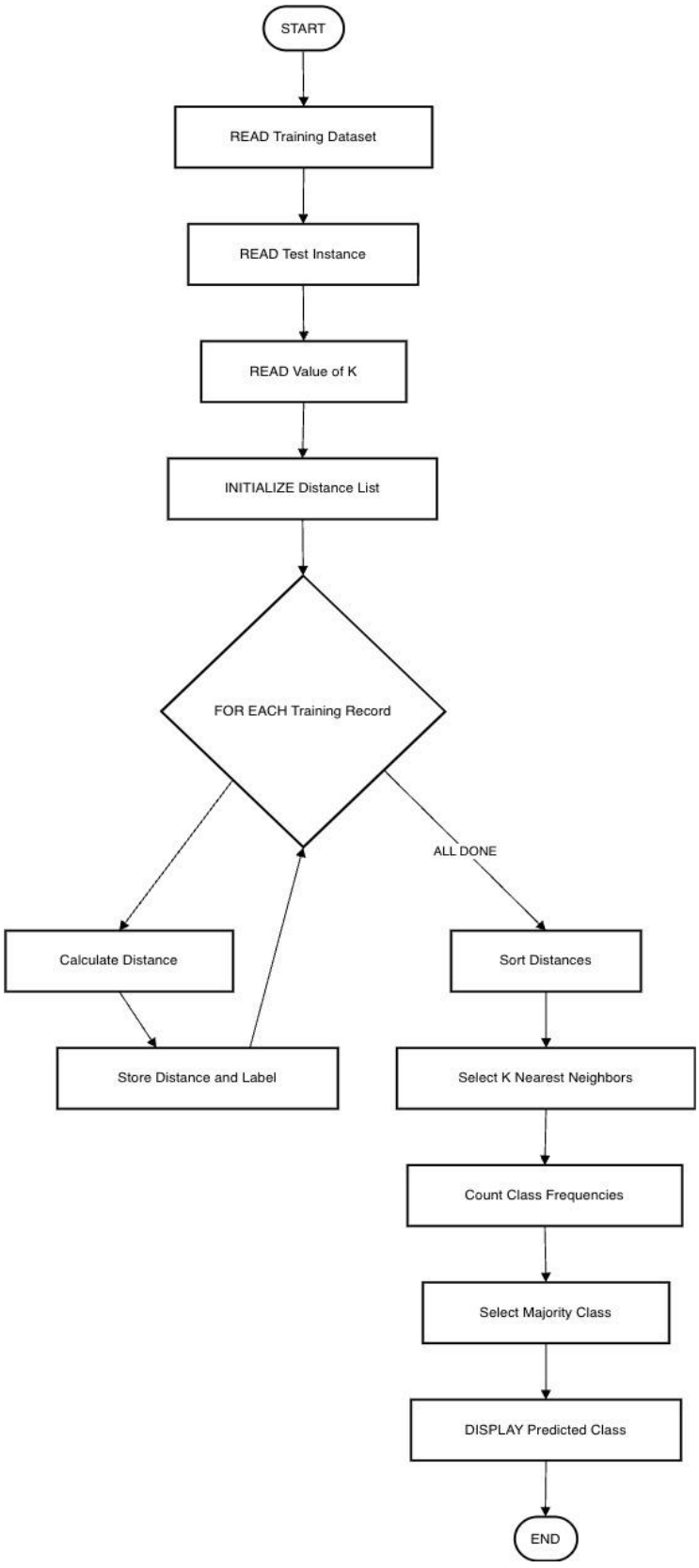
Pseudocode for KNN:

```
PROCESS KNN
  READ TrainingData
  READ TestInstance
  READ K

  INITIALIZE DistanceList AS empty list

  FOR EACH Record IN TrainingData
    GET TrainingFeatures
    GET TrainingLabel

    CALCULATE Distance BETWEEN TestInstance AND TrainingFeatures

    ADD (Distance, TrainingLabel) TO DistanceList
  END FOR

  SORT DistanceList BY Distance IN ASCENDING ORDER

  SELECT First K Records FROM DistanceList AS NearestNeighbors

  INITIALIZE ClassFrequency AS empty dictionary

  FOR EACH Neighbor IN NearestNeighbors
    GET NeighborLabel
    INCREMENT ClassFrequency[NeighborLabel]
  END FOR
  SELECT Class WITH MAXIMUM Frequency FROM ClassFrequency
  DISPLAY SelectedClass
END PROCESS
```

Flowcharts


START
READ Dataset
INITIALIZE Counters and Vocabulary
FOR EACH Message
ALL DONE
Tokenize Message
Is Label Spam?
YES
NO
Update Spam Word Counts
Update Ham Word Counts
Calculate Prior Probabilities
READ Input Message
Tokenize Input Message
Calculate Spam Score
Calculate Ham Score
Spam Score > Ham Score?
YES
NO
DISPLAY SPAM
DISPLAY HAM
END


START
READ Dataset
INITIALIZE Weight and Bias
READ Learning Rate and Iterations
FOR EACH Iteration
TRAINING DONE
SET Gradients to Zero
READ Input Value
FOR EACH Data Record
Calculate Probability
ALL DONE
Calculate Linear Output
Update Weight and Bias
Probability ≥ Threshold?
YES
NO
Apply Sigmoid Function
DISPLAY Class 1
DISPLAY Class 0
Calculate Error
END
Update Gradients


START
READ Training Dataset
READ Test Instance
READ Value of K
INITIALIZE Distance List
FOR EACH Training Record
ALL DONE
Calculate Distance
Store Distance and Label
Sort Distances
Select K Nearest Neighbors
Count Class Frequencies
Select Majority Class
DISPLAY Predicted Class
END