**K - NN**

**CSE 303: Machine Learning**

Submitted by

Name: Guntur Ridhi

Roll No: AP22110011467

Section: CSE M

Lab Date: 16/09/2024

Submission Date: 24/09/2024

**A picture containing text

Description automatically generated**

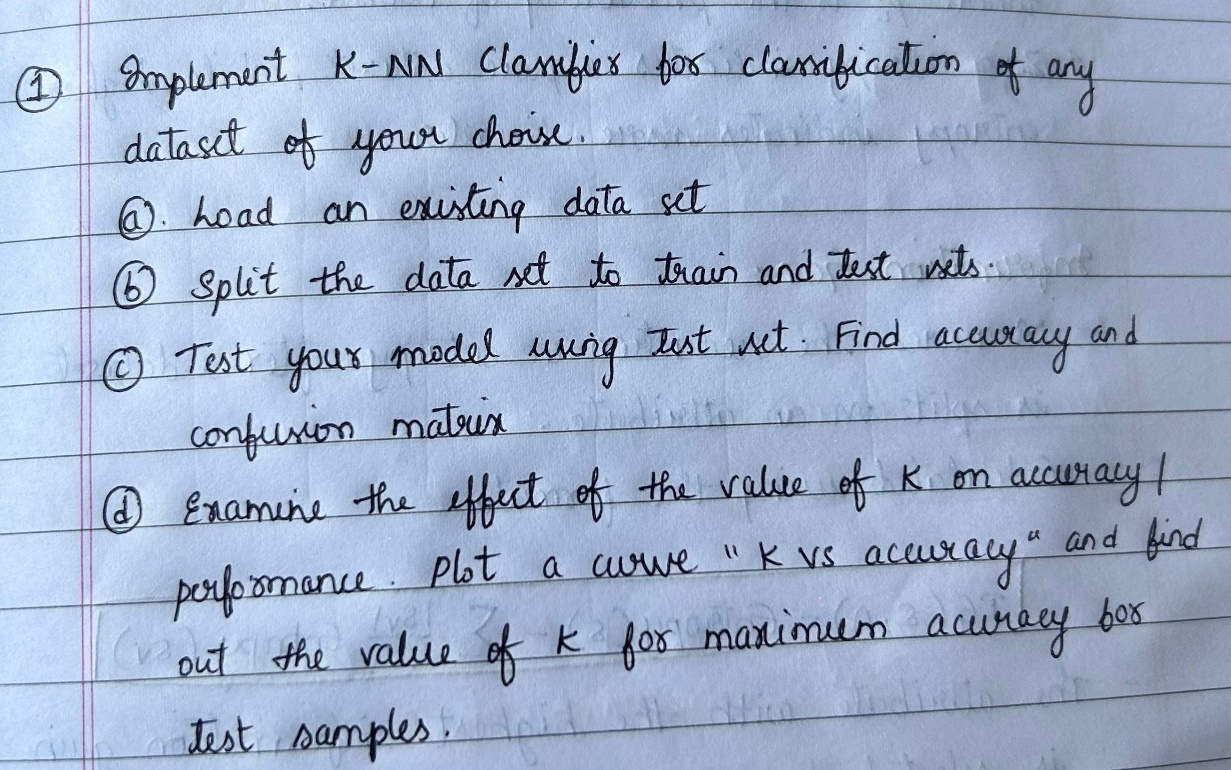
**Department Computer Science and Engineering**

**School of Engineering and Sciences**

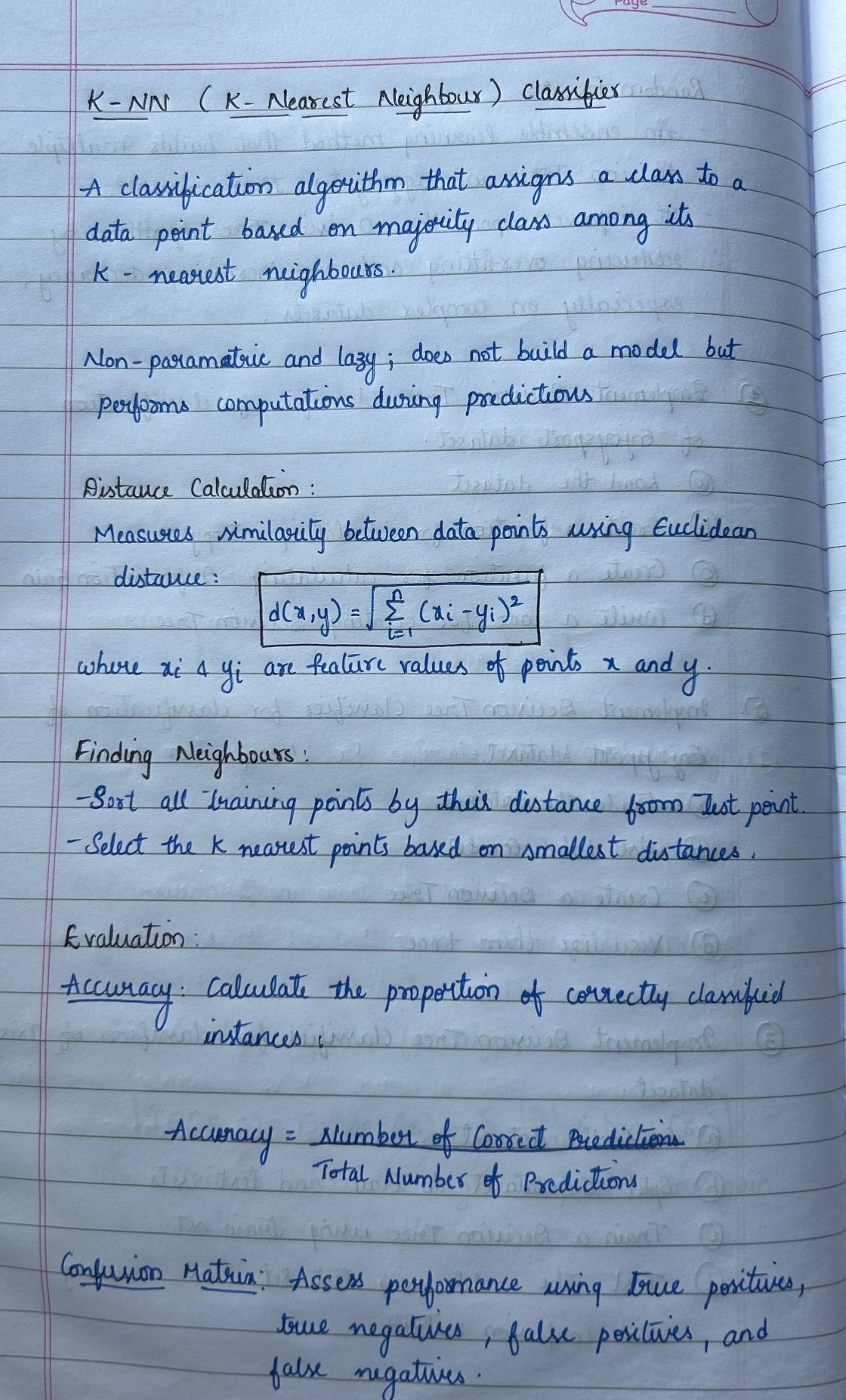
**SRM University–AP**

**Amaravati, Andhra Pradesh – 522 240, India**

1. **Question**



1. **Algorithm Description**



1. **Solution**

**K-NN Classifier Implementation**

|  |
| --- |
| **1. Loading and Preprocessing the Dataset** |
| import numpy as np  import pandas as pd |
| # Load the dataset  car\_data = pd.read\_csv(r"C:\Users\guntu\Downloads\car\_data.csv") |
| **Drop 'User ID' and encode 'Gender' column as 0 (Female) and 1 (Male)** |
| # Split data into features (X) and target (y)  X = car\_data[['Gender', 'Age', 'AnnualSalary']].values  y = car\_data['Purchased'].values |
| **2. Splitting the Data into Train and Test Sets** |
| # Function to split data into train and test sets  def train\_test\_split(X, y, test\_size=0.2):  indices = np.arange(X.shape[0])  np.random.shuffle(indices)  split\_idx = int(X.shape[0] \* (1 - test\_size))  train\_idx, test\_idx = indices[:split\_idx], indices[split\_idx:]  return X[train\_idx], X[test\_idx], y[train\_idx], y[test\_idx]  # Split the dataset into training and testing sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y) |
| **3. Implementing the K-NN Classifier** |
| # Euclidean distance calculation  def euclidean\_distance(x1, x2):  return np.sqrt(np.sum((x1 - x2) \*\* 2)) |
| # K-NN algorithm without using external libraries like Counter  def knn\_classifier(X\_train, y\_train, X\_test, k):  y\_pred = []  for test\_point in X\_test:  # Compute distances from the test point to all training points  distances = [euclidean\_distance(test\_point, x\_train) for x\_train in X\_train]    # Get the indices of the k-nearest neighbors  k\_indices = np.argsort(distances)[:k]    # Get the corresponding labels of the nearest neighbors  k\_nearest\_labels = [y\_train[i] for i in k\_indices]    # Manual majority voting  label\_count = {}  for label in k\_nearest\_labels:  if label in label\_count:  label\_count[label] += 1  else:  label\_count[label] = 1    # Determine the class with the highest count (majority voting)  majority\_class = max(label\_count, key=label\_count.get)  y\_pred.append(majority\_class)    return np.array(y\_pred) |
| **4. Evaluating the Model (Accuracy and Confusion Matrix)** |
| # Function to compute accuracy  def compute\_accuracy(y\_true, y\_pred):  return np.sum(y\_true == y\_pred) / len(y\_true)  # Function to compute confusion matrix  def confusion\_matrix(y\_true, y\_pred):  tp = np.sum((y\_true == 1) & (y\_pred == 1))  tn = np.sum((y\_true == 0) & (y\_pred == 0))  fp = np.sum((y\_true == 0) & (y\_pred == 1))  fn = np.sum((y\_true == 1) & (y\_pred == 0))  return np.array([[tn, fp], [fn, tp]]) |
| # Test the K-NN Classifier with k=5  k = 5  y\_pred = knn\_classifier(X\_train, y\_train, X\_test, k)  # Compute accuracy  accuracy = compute\_accuracy(y\_test, y\_pred)  print(f"Accuracy: {accuracy}") |
| Accuracy: 0.835 |
| # Compute confusion matrix  conf\_matrix = confusion\_matrix(y\_test, y\_pred)  print(f"Confusion Matrix:\n{conf\_matrix}") |
| Confusion Matrix:  [[107 14]  [ 19 60]] |
| **5. Examining the Effect of K on Accuracy** |
| import matplotlib.pyplot as plt  # Test the K-NN Classifier for different values of k and store the accuracies  k\_values = range(1, 20)  accuracies = []  for k in k\_values:  y\_pred = knn\_classifier(X\_train, y\_train, X\_test, k)  accuracy = compute\_accuracy(y\_test, y\_pred)  accuracies.append(accuracy)  # Plot k vs accuracy  plt.plot(k\_values, accuracies, marker='o')  plt.xlabel('k (Number of Neighbors)')  plt.ylabel('Accuracy')  plt.title('K vs Accuracy')  plt.show() |
|  |
| # Find the best k for maximum accuracy  best\_k = k\_values[np.argmax(accuracies)]  print(f"Best value of k: {best\_k}") |

1. **Code Repository:**

GitHub Link: <https://github.com/Ridhi-215/K-NN>