K-Nearest Neighbor (K-NN)

Team Members: Caleb Dulay, Daniel Gebran, Henry Delgado

Contributions

Caleb Dulay

- Researched K-NN
- Worked on Presentation slides
- Contributed to reports
- Wrote functions for recall, F1, and precision
- Helped analyze complexities

Henry Delgado

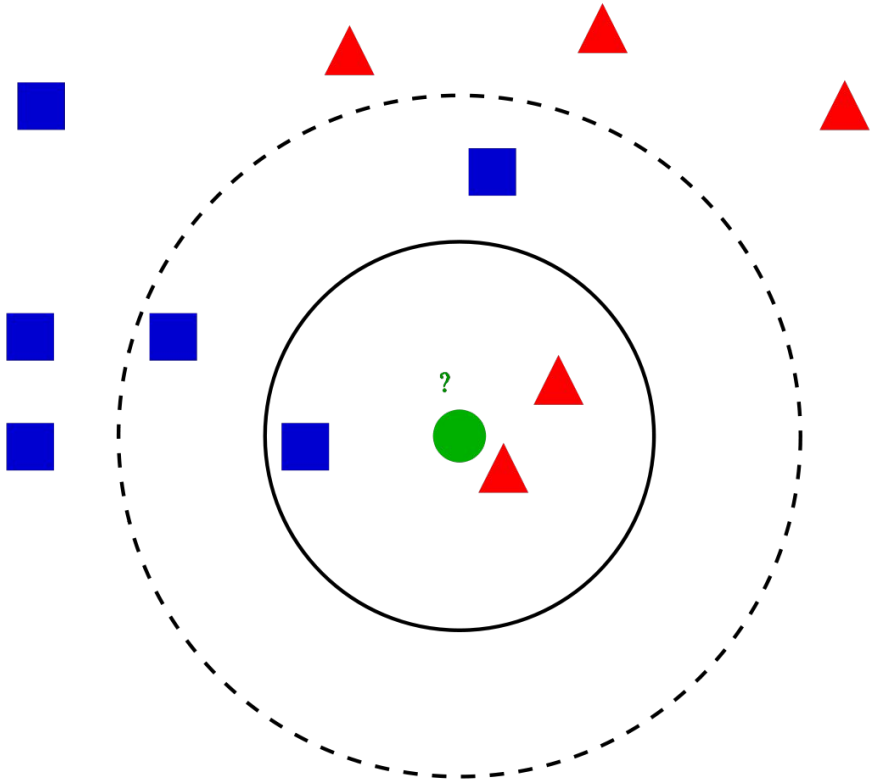
- Found/Plotted dataset we used
- Contributed to reports
- Contributed to Presentation slides
- Research

Daniel Gebran

- Researched K-NN and weighted K-NN
- Designed Algorithm
- Wrote the C++ code
- Found ways to optimize K-NN
- Contributed to reports
- Analyzed performance results
- Analyzed the complexities

K-NN Background

- KNN is one of the fundamental algorithms in machine learning. Machine learning models use a set of input values to predict output values. KNN is one of the more simpler forms of machine learning algorithms mostly used for classification. It classifies the data point on how its neighbor is classified. KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems.



Hypothesis/Proposed Method

We want to implement basic K-NN using a unique dataset, then by using weighted K-NN we hope to improve the prediction **accuracy**, **precision**, **recall** and the **F1 score.**

Advantages of K-NN

- Adapts Easily –it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
- Few Hyperparameters – The only parameters required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

K-NN Example

Height(cm)	Weight(kg)	Class
167	51	Underweight
182	62	Normal
176	69	Normal
173	64	Normal
172	65	Normal
174	56	Underweight
169	58	Normal
173	57	Normal
170	55	Normal
170	57	?

Euclidean Distance

- In this case, X being the height and Y being the weight and X2 will be 170 and Y2 will be 57 since this is what we are trying to find the class for.

Distance Formula

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$d_1 = \sqrt{(170 - 167)^2 + (57 - 51)^2}$$

$$= \sqrt{(3)^2 + (6)^2}$$

$$= \sqrt{9 + 36}$$

$$= \sqrt{45} \approx 6.7$$

Height(cm)	Weight(kg)	Class	Distance	Rank/Order/(K)
167	51	Underweight	6.7	5
182	62	Normal	13	8
176	69	Normal	13.4	9
173	64	Normal	7.6	6
172	65	Normal	8.2	7
174	56	Underweight	4.1	4
169	58	Normal	1.4	1
173	57	Normal	3	3
170	55	Normal	2	2
170	57	?	X	X

Outcomes

- If K = 1, class = normal
- If K = 2, class = normal
- If K = 3, class = normal
- If K = 4, class = normal
- If K = 5, class = normal

And so on and so forth…

Height(cm)	Weight(kg)	Class	Distance	Rank/Order/(K)
167	51	Underweight	6.7	5
182	62	Normal	13	8
176	69	Normal	13.4	9
173	64	Normal	7.6	6
172	65	Normal	8.2	7
174	56	Underweight	4.1	4
169	58	Normal	1.4	1
173	57	Normal	3	3
170	55	Normal	2	2
170	57	?	?	

K-NN Algorithm

```
knnClassifier(trainData[], testPoint)

distances ← List of pairs
for i ← 1 to length(trainData) - 1
     distance ← eucledianDistance(testPoint, trainData[i])
     distances.append((distance, i))

     sort(distances)

     classCount ← 0
     for i ← 1 to k - 1
          do neighborIndex ← distances[i].second
          classCount ← classCount + trainData[neighborIndex].classLabel

     prediction ← (classCount > k/2) ? 1 : 0
     return prediction
```

Weighted K-NN Algorithm

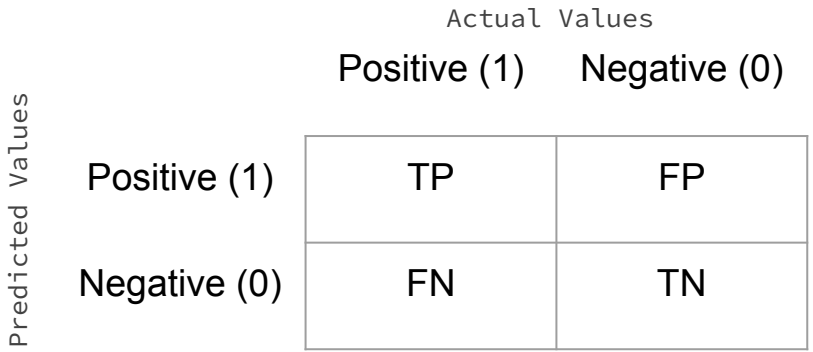
```
weightedKNN(trainData[], testPoint)

...
Same as knnClassifier() until counting the classLabel matches
...

for i ← 1 to k - 1
    do neighborIndex ← distances[i].second
    if trainData[neighborIndex].classLabel ← 0 then
        freq1 ← freq1 + (1 / distances[i].first)
    else if trainData[neighborIndex].classLabel ← 1 then
        freq2 ← freq2 + (1 / distances[i].first)

    prediction ← (freq1 > freq2) ? 0 : 1
    return prediction
```

KNN Evaluation Metrics

- In addition to the time and space complexities, it is crucial to evaluate the performance of a machine learning model using the following metrics:
  - **Accuracy**
  - **Precision**
  - **Recall**
  - **F1 Score**

Confusion Matrix

- In order to obtain those metrics, we first need to build a **confusion matrix**, which groups the classification results into four categories:
  - True Positive (TP): Number of samples *correctly* predicted as “positive.”
  - True Negative (TN): Number of samples *wrongly* predicted as “positive.”
  - False Positive (FP): Number of samples *correctly* predicted as “negative.”
  - False Negative (FN): Number of samples *wrongly* predicted as “negative.”

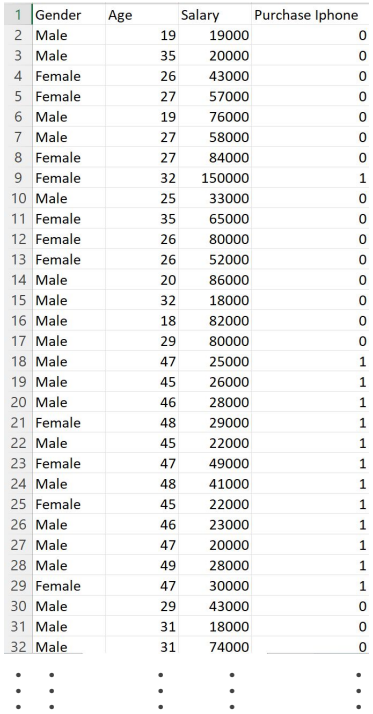


Performance Metrics

After getting a confusion matrix, we can now compute the performance metrics:

- **Accuracy**: the number of correct predictions divided by the total number of predictions
  or (TN + TP) / (TN + TP + FN + FP)
- **Precision**: TP / (TP + FP)
- **Recall**: TP / (TP + FN)
- **F1 Score**: (2 * Precision * Recall) / (Precision + Recall)

Example: Testing KNN on an iPhone Purchase dataset



	Gender	Age	Salary	Purchase Iphone
2	Male	19	19000	0
3	Male	35	20000	0
4	Female	26	43000	0
5	Female	27	57000	0
6	Male	19	76000	0
7	Male	27	58000	0
8	Female	27	84000	0
9	Female	32	150000	1
10	Male	25	33000	0
11	Female	35	65000	0
12	Female	26	80000	0
13	Female	26	52000	0
14	Male	20	86000	0
15	Male	32	18000	0
16	Male	18	82000	0
17	Male	29	80000	0
18	Male	47	25000	1
19	Male	45	26000	1
20	Male	46	28000	1
21	Female	48	29000	1
22	Male	45	22000	1
23	Female	47	49000	1
24	Male	48	41000	1
25	Female	45	22000	1
26	Male	46	23000	1
27	Male	47	20000	1
28	Male	49	28000	1
29	Female	47	30000	1
30	Male	29	43000	0
31	Male	31	18000	0
32	Male	31	74000	0

400

1. Read CSV file
2. Split dataset into training set (80%) and testing set (20%)
3. Make predictions on testing set
4. Compare predicted and actual data by evaluating accuracy, precision, recall, and F1 score
5. Repeat Step 3 using Weighted KNN then evaluate performance metrics

Performance Results

1st trial:

```
Regular KNN Results:

Confusion Matrix:
119 15
16  20
Accuracy: 0.75
Precision: 0.571429
Recall: 0.555556
F1 Score: 0.56338
```

```
Weighted KNN Results:

Confusion Matrix:
120 14
11  25
Accuracy: 0.825
Precision: 0.641026
Recall: 0.694444
F1 Score: 0.666667
```

2nd trial:

```
Regular KNN Results:

Confusion Matrix:
128 55
8  18
Accuracy: 0.8375
Precision: 0.246575
Recall: 0.692308
F1 Score: 0.363636
```

```
Weighted KNN Results:

Confusion Matrix:
124 59
5  21
Accuracy: 0.825
Precision: 0.2625
Recall: 0.807692
F1 Score: 0.396226
```

Performance Results (cont.)

3rd trial:

```
Regular KNN Results:

Confusion Matrix:
125 65
9  19
Accuracy: 0.8125
Precision: 0.22619
Recall: 0.678571
F1 Score: 0.339286
```

```
Confusion Matrix:
123 67
6  22
Accuracy: 0.825
Precision: 0.247191
Recall: 0.785714
F1 Score: 0.376068
```

Analysis of Time/Space Complexity

- Time and Space Complexity: O(n*m)
  where n is the dataset size and m is the number of features.

- The time complexity of weighted KNN vs basic KNN is generally the same, as the additional computations for weighing the contributions of neighbors is usually a constant factor and does not significantly impact the overall complexity.

Conclusion

- **Objective**
  - Improve the prediction accuracy, precision, recall and the F1 score using Weighted KNN.
- **Method**
  - Weighted KNN
- **Experimentation**
  - Record accuracy,recall, precision, and F1 score and compared it between basic and weighted KNN.
  - Compared space/time complexities of basic and weighted KNN
- **Was objective/optimization successful?**
  - Yes, our objective was successful as we were able to use weighted KNN to optimize accuracy, precision, F1 score, and recall.

References

1. K-Nearest Neighbor(KNN) Algorithm - GeeksforGeeks
2. K-Nearest Neighbor Classifiers | STAT 508 (psu.edu)
3. Weighted K-NN - GeeksforGeeks
4. Train Test Split: What it Means and How to Use It | Built In
5. K-Nearest Neighbors (KNN) Classification with scikit-learn | Datacamp
6. Classification Evaluation Metrics: Accuracy, Precision, Recall, and F1 Visually Explained | Cohere Blog
7. KNN Algorithm: When? Why? How?. KNN: K Nearest Neighbour is one of the… | by Aditya Kumar | Towards Data Science