Weather Forecast Classification using k-NN and Centroid-Based Classification

Hans Sebastian, Michael Liem, Edison Widjaja, Vincent Liem

May 10, 2025

1 Introduction

Classification is one of the most common use-cases for artificial intelligence. From identifying cancerous tissue to spam e-mail detection, the growing need for the ability to reliably detect and discern one item from another is becoming increasingly important in the modern age. Other uses of classification includes predicting upcoming events based on patterns exhibited in prior data. Examples of this include predicting the state of weather in a certain regions, as well as predicting user preferences based on their activity logs. Due to its proven potential for information gathering, there is an increasing need for an efficient, reliable classification method. As a result, numerous algorithms have been developed specifically to tackle this issue. One such algorithm is the k-NN (k-Nearest Neighbors) algorithm first discovered by Evelyn Fix and Joseph Hodges in 1951[1]. It is categorized in statistics as a non-parametric[2] supervised machine learning[3] algorithm which categorizes new inputs based on its distance from other neighboring data points. The assign classes to new data based on a majority vote, where for any 2 class C_0 and C_1, the class that appears most frequently is selected as the class of the new input. Another popular algorithm for classifying new data is the nearest centroid classifier algorithm which derives the classes of new data based on their measured distance to class centroids of previous input data instead of individual data points[4]. In this case, the class of the nearest centroid is chosen as the input data's class. Both methods are used to classify numerical data points into different their respective groups.

The aim of this study is to determined the effectiveness of a k-NN model in classifying and predicting the weather from a given dataset. The paper will utilize a publicly available weather dataset to measure the effectiveness of each classifier. The dataset includes five attributes: temperature, humidity, wind speed, cloud coverage, and atmospheric pressure, with the target variable being rainfall occurrence. This dataset is chosen due to its potential impact in determining the weather of any given day, which could help in areas such as agriculture and meteorology. The ability to predict future weather conditions will be beneficial for disaster mitigation and growing season analysis, futrher increasing the importance of said dataset. The classification is performed using a model which implements both the k-NN and nearest centroid algorithm. The performance of the model will be measured by 3 metrics, accuracy, precision and recall. Both the model and performance measurement implementation will be provided as a complement to this paper.

2 Methodology and Implementation

This section details the methodology and implementation of the model's required functionalities. The dataset used to train the model is supplied in the form of a CSV (Comma-Separated Values) file. This paper defines the requirements of the model to include:

- The ability to process training data into a usable format for classification purposes.
- The ability to accurately and reliably predict new input data from existing training data in an efficient manner.

2.1 Dataset and Preprocessing

To improve the reliability and robustness of the model, the dataset is normalized by dividing each attribute of the data with its Euclidian norm to limit the values between 0 and 1. Additionally, k-fold cross-validation / rotation estimation is applied to the data to reduce potential dataset bias and prevent model overfitting[5, 6]. It is done by splitting the original dataset into $k-1$ parts, and grouping the subsets into 2 datasets T and E where the former represents the dataset used for training, and the latter is used for evaluating the model's performance[7]. Each k subset is used exactly once for evaluation to ensure that all parts of the data are equally represented when training the model. As both models rely on a concrete definition of distance, the implementation of the model is equipped with a conversion algorithm which enables the model to process qualitative data.

2.2 Classification Algorithms

The model is implemented using Python, primarily utilizing NumPy and Pandas libraries. The classification process integrates:

- K-Nearest Neighbors (KNN)
- Centroid-Based Classification

Each classifier employs different distance metrics to determine class labels for test data. 3 primary distance metrics are used for classification: cosine similarity, Euclidean distance, and Mahalanobis distance. Cosine similarity is the measure of similarity between 2 vectors/data points with respect to their attributes. For example, given 2 n-dimensional points $\mathbf{A}$ and $\mathbf{B}$ containing numerical data, It measures the angle difference between the points using the formula

$$d_C(\mathbf{A},\mathbf{B}) = \cos\theta = \frac{\mathbf{A}\cdot\mathbf{B}}{||\mathbf{A}||||\mathbf{B}||}$$

Where $||\mathbf{A}||$ and $||\mathbf{B}||$ denotes the Euclidian norm of $\mathbf{A}$ and $\mathbf{B}$ respectively[8]. The Euclidian distance of A and B is defined as line segment between them as denoted by the formula

$$d_M(\mathbf{A},\mathbf{B}) = \sqrt{\sum_{n}^{i=0}(a_i - b_i)^2}$$

where a_i and b_i denotes the i-th attribute of both data points[9]. Lastly, the Mahalanobis distance between two points is defined as the Euclidean distance normalized by the covariance structure of a distribution of points belonging to a set. It does this by transforming the axes such that it is no longer perpendicular to each other, accounting for correlation between attributes of the data points[10, 11].

$$d_M(\mathbf{A},\mathbf{B}) = \sqrt{(\mathbf{A}-\mathbf{B})^{\mathrm{T}}C^{-1}(\mathbf{A}-\mathbf{B})}$$

These distance metrics are implemented within the model as an alternative way of classifying new data in order to measure which metric will generate the best evaluated output.

2.3 Performance Evaluation

The classification performance is assessed using accuracy, precision, and recall[12]. Accuracy can be defined as the ratio between the correct results and all results. The mathematical formula for accuracy is written as

$$\text{Accuracy} = \frac{\mathrm{T_P} + \mathrm{T_F}}{\mathrm{T_P} + \mathrm{T_N} + \mathrm{F_P} + \mathrm{F_P}}$$

where $\mathrm{T_P}$ and $\mathrm{T_P}$ are true positive and negative predictions, and $\mathrm{F_P}$ and $\mathrm{F_P}$ are false positive and negative predictions. Precision can be defined as the ratio of true positives to all positive predictions. It is given by the formula

$$\text{Precision} = \frac{\mathrm{T_P}}{\mathrm{T_P} + \mathrm{F_P}}$$

Lastly, recall is described as the model's ability to correctly identify the positive class for any given dataset. Given the dataset is binary, the positive class can be arbitrarily chosen from 1 of the 2 categories. The governing formula for recall is written as

$$\text{Recall} = \frac{\mathrm{T_P}}{\mathrm{T_P} + \mathrm{F_N}}$$

The result from all three measurements are combined to select the best overall model for a given value of k. The combination process include taking the average of the sum of all measurements and choosing the model with the greatest average. The model evaluation is conducted through k-fold cross-validation, ensuring comprehensive testing. To reduce bias and randomness, the same subsets are used during every model's training phase. In addition, the covariance matrix required to measure the Mahalanobis Distance will be calculated using the formula

$$C = \mathrm{cov}(D_{\mathrm{train}})$$

where D_{train} represents the training dataset as a $n \times m$ matrix.

Multiple models are constructed, each with a different value of k as a way to minimize potential biases in training. A total of 5 $(3, 5, 7, 9, 11)$ values of k were used during evaluation to determine the most effective value of k. Only odd values of k are considered to minimize conflicts when the number of nearest neighbor's classes are equal. Each model will be tested using all metrics listed during section as discussed in Section 2.2. A combination of both k and distance metrics will be used as the evaluation target. Additionally, each fold of the data will be used to train a new model in order to find the most suitable train-test subset. The best models in each metric will be saved to be used for later classification. Results are shown as a series of tables with their corresponding values as seen in the Appendix section.

3 Results and Discussion

Model evaluation shows that the accuracy metric of each model using each distance metric varies slightly, although it is worthy to note that the model's accuracy using Euclidian and Mahalanobis distance decreases as k increases, with the latter experiencing a larger decrease in all metrics. It is theorized that the covariance matrix used to calculate the mahalanobis distance is not appropriate for the given dataset, which could explain the measured results. Furthermore, it is observed that the measurement for cosine similarity and Euclidian distance is identical, as it can be proven that using cosine similarity to normalized data on will produce the same ordering as Euclidian distance[13]. This is due to the fact that as the length of each data point converges to 1, the Euclidian distance decreases and the cosine similarity increases.

Further analysis of the results show that models using Euclidean distance and cosine similarity achieve identical performance metrics. This is due to data normalization before model fitting, which ensures consistent distance calculations. However, the centroid-based classification method exhibits the lowest performance in comparison. Although it could be argued that the centroid-based classification ranks the highest in terms of computation time compared to the other models, its lack of accuracy causes a large hindrance in terms of the model's usability.

The implementation details for the model and metric evaluation can be found in the `model.py` and `metrics.py` modules, respectively. Future work can explore additional normalization techniques, parameter tuning, and alternative distance metrics to enhance classification accuracy.

References

[1] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *International Statistical Review / Revue Internationale de Statistique*, vol. 57, p. 238, 12 1989.

[2] M. Grant, "How nonparametric statistics work," 10 2022.

[3] G. Cloud, "What is supervised learning?."

[4] T. Bellotti, I. Nouretdinov, M. Yang, and A. Gammerman, "Chapter 6 - feature selection," in *Conformal Prediction for Reliable Machine Learning* (V. N. Balasubramanian, S.-S. Ho, and V. Vovk, eds.), pp. 115–130, Boston: Morgan Kaufmann, 2014.

[5] J. Brownlee, "A gentle introduction to k-fold cross-validation," 10 2023.

[6] N. Arya, "Why use k-fold cross validation?," 07 2022.

[7] D. Berrar, "Cross-validation," *Encyclopedia of Bioinformatics and Computational Biology*, vol. 1, pp. 542–545, 2019.

[8] J. Han, M. Kamber, and J. Pei, "2 - getting to know your data," in *Data Mining (Third Edition)* (J. Han, M. Kamber, and J. Pei, eds.), The Morgan Kaufmann Series in Data Management Systems, pp. 39–82, Boston: Morgan Kaufmann, third edition ed., 2012.

[9] J. Tabak, *Geometry: The Language of Space and Form*. Facts on File math library, Facts On File, Incorporated, 2014.

[10] Stephanie, "Mahalanobis distance: Simple definition, examples," 11 2017.

[11] G. Mclachlan, "Mahalanobis distance," *Resonance*, vol. 4, pp. 20–26, 06 1999.

[12] Google, "Classification: Accuracy, recall, precision, and related metrics," 2024.

[13] L. (https://stats.stackexchange.com/users/7733/lucas), "Is cosine similarity identical to l2-normalized euclidean distance?." Cross Validated. URL:https://stats.stackexchange.com/q/146279 (version: 2015-04-14).

Appendix

		euclidian	mahalanobis	cosine	centroid
Accuracy	0	0.926441	0.932406	0.926441	0.868787
	1	0.926441	0.918489	0.926441	0.831014
	2	0.936382	0.924453	0.936382	0.886680
	3	0.934394	0.920477	0.934394	0.844930
	4	0.912525	0.910537	0.912525	0.854871
Macro Precision	0	0.927007	0.932992	0.927007	0.869051
	1	0.927340	0.919442	0.927340	0.832282
	2	0.933698	0.922291	0.933698	0.883619
	3	0.934796	0.921237	0.934796	0.845509
	4	0.913386	0.911417	0.913386	0.856062
Macro Recall	0	0.931759	0.938188	0.931759	0.869788
	1	0.933218	0.926044	0.933218	0.842463
	2	0.943737	0.929020	0.943737	0.894070
	3	0.935623	0.925190	0.935623	0.847322
	4	0.924915	0.923469	0.924915	0.876391

Table 1: Training evaluation for $k = 3$

		euclidian	mahalanobis	cosine	centroid
Accuracy	0	0.922465	0.912525	0.922465	0.868787
	1	0.916501	0.896620	0.916501	0.831014
	2	0.930417	0.926441	0.930417	0.886680
	3	0.934394	0.912525	0.934394	0.844930
	4	0.912525	0.900596	0.912525	0.854871
Macro Precision	0	0.923149	0.913346	0.923149	0.869051
	1	0.917536	0.897818	0.917536	0.832282
	2	0.927266	0.923645	0.927266	0.883619
	3	0.935128	0.913559	0.935128	0.845509
	4	0.913386	0.901575	0.913386	0.856062
Macro Recall	0	0.930293	0.923758	0.930293	0.869788
	1	0.925462	0.908330	0.925462	0.842463
	2	0.940152	0.934014	0.940152	0.894070
	3	0.938876	0.921409	0.938876	0.847322
	4	0.924915	0.916388	0.924915	0.876391

Table 2: Training evaluation for $k = 5$

		euclidian	mahalanobis	cosine	centroid
Accuracy	0	0.914513	0.906561	0.914513	0.868787
	1	0.910537	0.900596	0.910537	0.831014
	2	0.924453	0.924453	0.924453	0.886680
	3	0.938370	0.894632	0.938370	0.844930
	4	0.898608	0.886680	0.898608	0.854871
Macro Precision	0	0.915315	0.907480	0.915315	0.869051
	1	0.911654	0.901795	0.911654	0.832282
	2	0.921016	0.921380	0.921016	0.883619
	3	0.939160	0.896078	0.939160	0.845509
	4	0.899606	0.887795	0.899606	0.856062
Macro Recall	0	0.925239	0.920608	0.925239	0.869788
	1	0.920933	0.912412	0.920933	0.842463
	2	0.935676	0.933466	0.935676	0.894070
	3	0.943636	0.911960	0.943636	0.847322
	4	0.915000	0.906863	0.915000	0.876391

Table 3: Training evaluation for $k = 7$

		euclidian	mahalanobis	cosine	centroid
Accuracy	0	0.920477	0.898608	0.920477	0.868787
	1	0.904573	0.882704	0.904573	0.831014
	2	0.920477	0.914513	0.920477	0.886680
	3	0.924453	0.886680	0.924453	0.844930
	4	0.892644	0.874751	0.892644	0.854871
Macro Precision	0	0.921220	0.899567	0.921220	0.869051
	1	0.905827	0.884148	0.905827	0.832282
	2	0.916667	0.911145	0.916667	0.883619
	3	0.925435	0.888069	0.925435	0.845509
	4	0.893701	0.875984	0.893701	0.856062
Macro Recall	0	0.929741	0.913668	0.929741	0.869788
	1	0.917676	0.899551	0.917676	0.842463
	2	0.933993	0.924808	0.933993	0.894070
	3	0.932576	0.902346	0.932576	0.847322
	4	0.910891	0.899038	0.910891	0.876391

Table 4: Training evaluation for $k = 9$

		euclidian	mahalanobis	cosine	centroid
Accuracy	0	0.922465	0.886680	0.922465	0.868787
	1	0.902584	0.884692	0.902584	0.831014
	2	0.914513	0.916501	0.914513	0.886680
	3	0.920477	0.862823	0.920477	0.844930
	4	0.886680	0.870775	0.886680	0.854871
Macro Precision	0	0.923189	0.887756	0.923189	0.869051
	1	0.903811	0.886108	0.903811	0.832282
	2	0.910417	0.912682	0.910417	0.883619
	3	0.921513	0.864651	0.921513	0.845509
	4	0.887795	0.872047	0.887795	0.856062
Macro Recall	0	0.931262	0.905382	0.931262	0.869788
	1	0.915033	0.900944	0.915033	0.842463
	2	0.929739	0.929843	0.929739	0.894070
	3	0.929514	0.889404	0.929514	0.847322
	4	0.906863	0.896497	0.906863	0.876391

Table 5: Training evaluation for $k = 11$