Contents lists available at https://citedness.com/index.php/jdsi

Data Science Insights

Journal Page is available to https://citedness.com/index.php/jdsi

Research article

Evaluating and Deploying Predictive Models for Weather Classification

Jolin Arfina Lie

Department of Informatics Engineering, Institut Bisnis dan Teknologi Pelita Indonesia, Pekanbaru, Riau, Indonesia
email: jolin.arfina@student.pelitaindonesia.ac.id

ARTICLE INFO

Article history:
Received
Revised July 14, 2024
Accepted July 17, 2024
Available online 01 August, 2024

Keywords:
Analysis
Algorithm
Classification
Prediction
Weather

Please cite this article in IEEE style as:
J. A. Lie, "Evaluating and Deploying Predictive Models for Weather Classification", Data Science Insights, vol. 2, no. 2.

ABSTRAK

Weather is the condition of the atmosphere in a specific location over a relatively short period of time, described through various parameters such as temperature, air pressure, wind speed, humidity, and other atmospheric phenomena. It differs from climate, which refers to the average atmospheric conditions over a large area and a long time period—studied under the field of climatology. Weather can vary from hot to cold, wet to dry, and windy to calm. It is influenced by dynamic changes in the Earth's atmosphere, including warming and cooling processes.In recent years, weather changes have become more frequent and unpredictable, significantly affecting daily human activities. Therefore, an intelligent system capable of detecting and predicting weather conditions is increasingly needed. This study aims to apply classification algorithms to predict weather conditions based on relevant meteorological parameters. The algorithms used include k-Nearest Neighbor, Random Forest, Naïve Bayes, Decision Tree, and Deep Learning.Given the irregularity and complexity of weather patterns, manual prediction becomes unreliable. Although it is impossible to predict the weather with absolute certainty, computational methods can provide reasonably accurate estimations. Based on the evaluation results, the Random Forest algorithm demonstrated the highest accuracy among the tested models. Furthermore, the final model was successfully deployed using Python, enabling real-time predictions on incoming weather data.

Correspondence:
Jolin Arfina Lie, Department of Informatics Engineering, Institut Bisnis dan Teknologi Pelita Indonesia, Pekanbaru, Riau, Indonesia

1. Introduction

Weather is the condition of the air in a place over a relatively short period of time, expressed by the values of several parameters such as temperature, air pressure, wind speed, air humidity, and various other atmospheric phenomena [1]. Weather is different from climate. Climate is the average temperature over a large area over a long period of time. The science that studies climate is called climatology. Weather can be hot or cold, wet or dry, windy or calm. Weather is caused by changes in the atmosphere around the Earth, either warming or cooling.

Recently, weather changes have been frequent and have had a significant impact on daily activities. Rapid weather changes can hinder human activities. Weather changes that are difficult to predict make it difficult for people to determine alternatives and anticipate weather changes when traveling [2]. Therefore, a system is needed that can detect weather conditions.

This research aims to apply and compare several classification algorithms to predict weather conditions. Given the irregularity of weather patterns, manual prediction is extremely difficult. While we cannot predict the weather with certainty, we can still make estimates.

2. Literature Review

Classification is a technique for forming models from unclassified data, to be used to classify new data [3]. Data classification is the process of finding a model or function that explains and differentiates data classes and their concepts [4]. This classification is a supervised learning method that attempts to find relationships between input attributes and target attributes. The purpose of this classification is to increase the reliability of the results obtained from the data [5].

2.1 k-Nearest Neighbors

The Nearest Neighbor algorithm is an algorithm that classifies data based on the proximity (distance) of data to other data. Classification is the process of assessing data objects to assign them to a specific class from among a number of available classes [6]. This method is non-parametric, meaning it makes no assumptions about the underlying data distribution. In other words, there is no fixed number of parameters or parameter estimates in the model, regardless of the data size. k-NN classifies new data based on the majority class of the k-Nearest Neighbors in the feature space as shown in Figure 1.

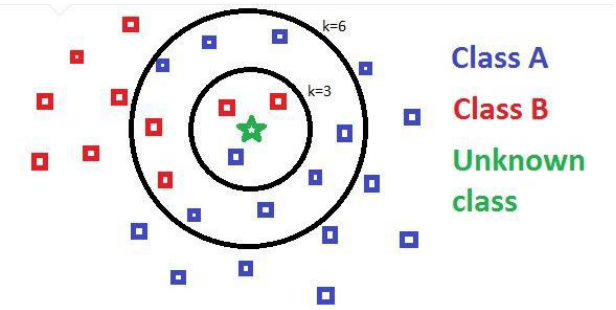

Figure 1. k-Nerest Neighbor (k-NN) Illustration

2.2 Decision Trees and Random Forests

Decision treeis a machine learning algorithm used to perform classification and regression by breaking down a dataset into smaller subsets based on certain features.This algorithm creates a model in the form of a tree structure, where each internal node represents a feature or attribute, each branch represents a decision rule, and each leaf represents an outcome or class label [7]. Meanwhile, Random Forest is a combination of several decision trees. By combining several decision trees, random forest can produce a more accurate classification model.

2.3 Naïve Bayes

Naïve Bayes is a method that divides problems into classes based on characteristics of similarities and differences by using statistics that can predict the probability of a class [8]. NBC is a classification algorithm that can be used to predict the probability of class membership by applying Bayes' theorem [9].

2.4 Deep Learning

*Deep learning*is a representation learning method that allows computational models composed of many processing layers to learn data representations with many levels of abstraction [10]. Deep learning models mimic the way the human brain works to solve classification problems. Deep learning processes data by automatically extracting complex features.

3. Research Methodology

Data processing (Figure 2) is the process of manipulating, organizing, and storing data in order to draw conclusions. The data processing process involves several stages: data collection, data processing, data analysis, and data presentation.

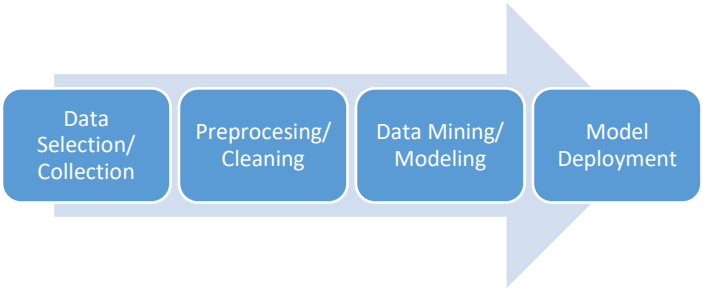

Figure 2. Data Science Process

3.1 *Data Selection/ Collection*

From the operational data set before starting the Knowledge Discovery in Database (KDD) process. The data that has been selected for use in the data mining process is then stored in a separate file from the operational database [11]. At this stage, data collection was carried out in the form of a dataset regarding past weather through the Kaggle website, with the title Weather Type Classification [12].

3.2 *Pre-processing/ Cleaning*

Data Cleaning is a stage carried out to handle data if missing values occur, removing columns that are not really used or imbalances in the data[13]. In this process, we remove data in the weather dataset that has no value (null) and duplicate data [14].

3.3 Data Mining/ Modelling

Data Mining is a process of data extraction or data filtering by utilizing data sets that are quite large in size through a series of processes to obtain valuable information from the data [15]. Model development was carried out by testing five different algorithms and comparing them to find the most suitable one. For the weather dataset, several classification algorithms will be compared to compare their accuracy. The methods used for this comparison are Cross Validation and Hold Out.

3.4. Model Deployment

After the development and training process of the weather classification model is completed, the next important stage is model deployment. This stage aims to implement the classification model into a system that can be used in a real-world context. The deployment process enables the trained model to receive actual weather data and produce weather classifications automatically, quickly, and accurately

4. Results and Discussion

4.1. Data Selection/ Collection Results

Based on the obtained dataset, there are several attributes such as Temperature, Humidity, Wind Speed, and others (Figure 3). The weather dataset includes several Weather Types (weather classes): Rainy, Cloudy, Sunny, and Snowy, which will be used as labels. The dataset contains 13,200 rows of data.

Temperature	Humidity	Wind Speed	Precipitation (%)	Cloud Cover	Atmospheric Pressure	UV Index	Season	Visibility (km)	Location	Weather Type
14.0	73	9.5	82.0	partly cloudy	1010.82	2	Winter	3.5	inland	Rainy
39.0	96	8.5	71.0	partly cloudy	1011.43	7	Spring	10.0	inland	Cloudy
30.0	64	7.0	16.0	clear	1018.72	5	Spring	5.5	mountain	Sunny
38.0	83	1.5	82.0	clear	1026.25	7	Spring	1.0	coastal	Sunny
27.0	74	17.0	66.0	overcast	990.67	1	Winter	2.5	mountain	Rainy
32.0	55	3.5	26.0	overcast	1010.03	2	Summer	5.0	inland	Cloudy
-2.0	97	8.0	86.0	overcast	990.87	1	Winter	4.0	inland	Snowy
3.0	85	6.0	96.0	partly cloudy	984.46	1	Winter	3.5	inland	Snowy
3.0	83	6.0	66.0	overcast	999.44	0	Winter	1.0	mountain	Snowy
28.0	74	8.5	107.0	clear	1012.13	8	Winter	7.5	coastal	Sunny
35.0	45	6.0	86.0	partly cloudy	879.88	2	Spring	1.0	mountain	Cloudy
38.0	43	2.0	16.0	clear	1029.16	11	Autumn	7.5	inland	Sunny
12.0	59	10.5	25.0	partly cloudy	1016.08	3	Autumn	5.5	mountain	Cloudy
-10.0	87	15.0	67.0	overcast	986.19	0	Winter	1.5	inland	Snowy
24.0	21	3.5	8.0	clear	1018.88	8	Winter	5.5	coastal	Sunny
10.0	50	6.5	46.0	partly cloudy	1000.44	2	Summer	8.5	mountain	Cloudy
30.0	27	7.0	13.0	partly cloudy	1016.38	5	Spring	7.5	inland	Sunny
33.0	51	0.5	27.0	overcast	1009.18	3	Autumn	5.5	coastal	Cloudy
43.0	46	0.5	15.0	clear	1025.8	9	Spring	6.0	mountain	Sunny
13.0	102	12.0	72.0	clear	1012.25	4	Summer	8.0	inland	Sunny

Figure 3. Sample Weather Dataset

4.2. Preprocessing or Cleaning Results

In the data cleaning process for the weather dataset, the examples filter (Figure 3) was used to remove missing values. After checking, no missing values were found in the processed data (Figure 4).

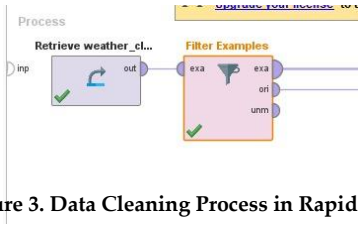

Figure 3. Data Cleaning Process in Rapid Miner

Row No.	Temperature	Humidity	Wind Speed	Precipitatio...	Cloud Cover	Atmospheri...	UV Index	Season
1	14	73	9.500	82	partly cloudy	1010.820	2	Winter
2	39	96	8.500	71	partly cloudy	1011.430	7	Spring
3	30	64	7	16	clear	1018.720	5	Spring
4	38	83	1.500	82	clear	1026.250	7	Spring
5	27	74	17	66	overcast	990.670	1	Winter
6	32	55	3.500	26	overcast	1010.030	2	Summer
7	-2	97	8	86	overcast	990.870	1	Winter
8	3	85	6	96	partly cloudy	984.460	1	Winter
9	3	83	6	66	overcast	999.440	0	Winter
10	28	74	8.500	107	clear	1012.130	8	Winter
11	35	45	6	86	partly cloudy	879.880	2	Spring
12	38	43	2	16	clear	1029.160	11	Autumn
13	12	59	10.500	25	partly cloudy	1016.080	3	Autumn
14	-10	87	15	67	overcast	986.190	0	Winter

Figure 4. Filter examples

4.3. Data Mining/ Modeling

In the model building process, it is essential to evaluate model performance objectively to ensure its ability to generalize well to unseen data. Therefore, this study employs two commonly used validation approaches in machine learning: the Hold-Out method and Cross-Validation. By using both validation methods, the study allows for a more comprehensive comparison of model performance, and provides a stronger basis for selecting the best algorithm to be deployed in the final stage.

4.3.1. Hold Out Method

Model development was carried out by testing five different algorithms and comparing them to find the most suitable one. The rapidminer structure, as shown in Figure 5, begins by reading the dataset and then dividing it into 70% test data and 30% training data.

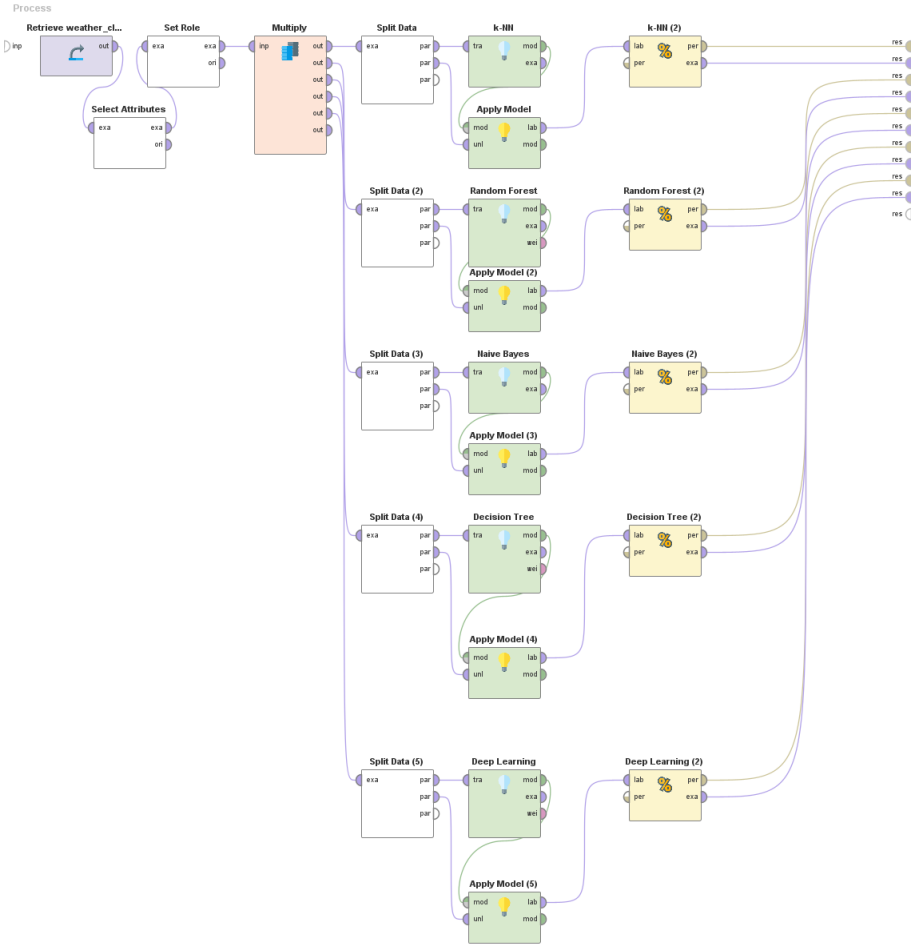

Figure 5. Hold Out Model

Naïve Bayes

accuracy: 87.60%

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	846	125	29	7	84.01%
pred. Cloudy	36	808	49	27	87.83%
pred. Sunny	46	52	916	41	86.82%
pred. Snowy	50	15	14	899	91.92%
class recall	86.50%	80.80%	90.87%	92.30%	

Figure 6. Results of the Naïve Bayes Algorithm

The results of the naïve Bayes algorithm can be seen in Figure 6. The algorithm has an accuracy level of 87.60%.

k-NN (k-nearest neighbors)

accuracy: 88.84%

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	872	74	49	19	86.00%
pred. Cloudy	60	844	49	20	86.74%
pred. Sunny	27	66	887	18	88.88%
pred. Snowy	18	20	22	915	93.85%
class recall	89.25%	84.06%	88.08%	94.14%	

Figure 7. Results of the K-Nearest Neighbors Algorithm

The results of the k-NN algorithm can be seen in Figure 7. The algorithm has an accuracy level of 88.84%.

Decision Tree

accuracy: 90.08%

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	952	85	60	40	83.73%
pred. Cloudy	41	846	51	28	87.58%
pred. Sunny	17	21	877	19	93.90%
pred. Snowy	9	11	11	892	96.64%
class recall	93.42%	87.85%	87.79%	91.11%	

Figure 8. Decision Tree Algorithm Results

The results of the decision tree algorithm can be seen in Figure 8. The algorithm has an accuracy level of 90.08%.

Random Forest

accuracy: 90.35%

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	880	72	23	13	89.07%
pred. Cloudy	39	865	45	32	88.18%
pred. Sunny	27	27	903	28	91.68%
pred. Snowy	28	26	22	930	92.45%
class recall	90.35%	87.37%	90.94%	92.72%	

Figure 9. Random Forest Algorithm Results

The results of the random forest algorithm can be seen in Figure 9. The algorithm has an accuracy level of 90.35%.

Deep Learning

accuracy: 90.93%

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	922	46	13	7	93.32%
pred. Cloudy	61	885	50	39	85.51%
pred. Sunny	34	43	875	45	87.76%
pred. Snowy	10	5	6	919	97.77%
class recall	89.78%	90.40%	92.69%	90.99%	

Figure 10. Deep Learning Algorithm Results

The results of the deep learning algorithm can be seen in Figure 10. The algorithm has an accuracy rate of 90.93%. This accuracy rate is the highest compared to other algorithms.

4.3.2. Cross Validation Method

The rapidminer structure, as shown in Figure 11, begins by reading the dataset and then dividing it into 70% test data and 30% training data. Afterward, cross-validation is performed, which includes classification.

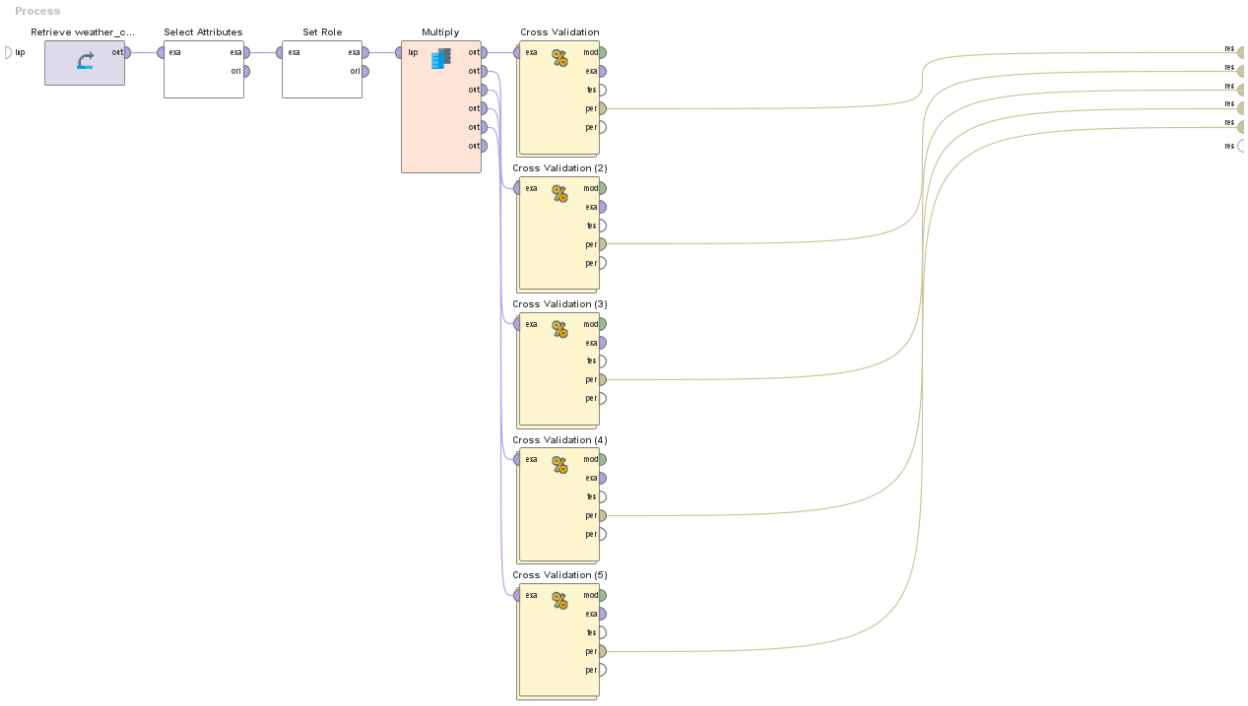

Figure 11. Cross Validation Model

Naïve Bayes

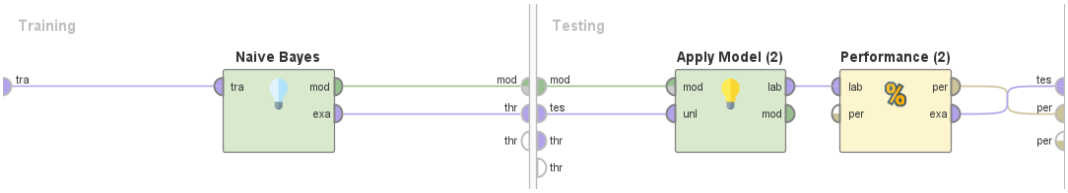

Figure 12. Naïve Bayes Algorithm Using Rapidminer

In the training process, the naïve Bayes algorithm process will be run (Figure 12), and for the testing section, the Apply Model and Performance processes will be used to produce the results of the process and its accuracy.

accuracy: 87.14% +/- 1.34% (micro average: 87.14%)

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	2818	422	93	29	83.82%
pred. Cloudy	156	2639	146	74	87.53%
pred. Sunny	152	198	3023	175	85.20%
pred. Snowy	174	41	38	3022	92.27%
class recall	85.39%	79.97%	91.61%	91.58%	

Figure 13. Results of the Naïve Bayes Algorithm

The results of the naive Bayes algorithm can be seen in Figure 13. The algorithm has an accuracy rate of 87.14%. This accuracy is supported by relatively high precision and recall values.

k-Nearest Neighbors (k-NN)

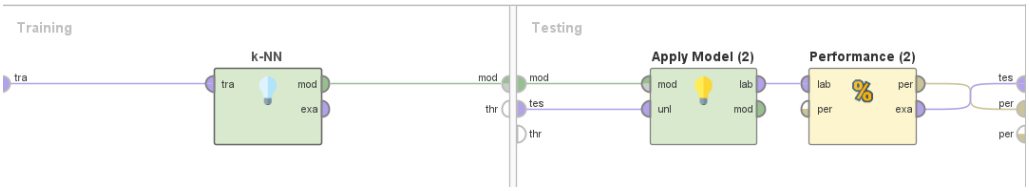

Figure 14. K-Nearest Neighbors Algorithm using Rapidminer

In the training process, the k-NN algorithm process will be run (Figure 14), and for the testing section, the Apply Model and Performance processes will be used to produce the results of the process and its accuracy.

accuracy: 88.80% +/- 0.41% (micro average: 88.80%)

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	2949	282	150	66	85.55%
pred. Cloudy	170	2776	140	80	87.68%
pred. Sunny	100	158	2928	85	89.51%
pred. Snowy	81	84	82	3069	92.55%
class recall	89.36%	84.12%	88.73%	93.00%	

Figure 15. Results of the K-Nearest Neighbors Algorithm

The results of the k-NN algorithm can be seen in Figure 15. The algorithm has an accuracy rate of 88.80%. This accuracy rate is higher than testing on data using the Naive Bayes algorithm.

Decision Tree

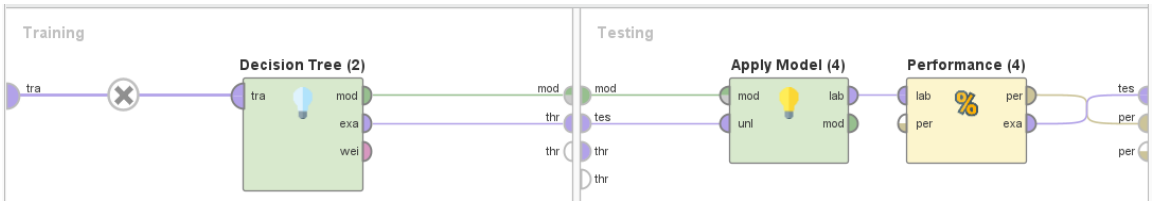

Figure 16. Decision Tree Algorithm Using Rapidminer

In the training process, the Decision Tree algorithm process will be run (Figure 16), and for the testing section, the Apply Model and Performance processes will be used to produce the process results and their accuracy.

accuracy: 90.35% +/- 0.68% (micro average: 90.35%)

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	3002	215	124	67	88.09%
pred. Cloudy	183	2957	186	131	85.54%
pred. Sunny	50	66	2915	50	94.61%
pred. Snowy	65	62	75	3052	93.79%
class recall	90.97%	89.61%	88.33%	92.48%	

Figure 17. Decision Tree Algorithm Results

The results of the decision tree algorithm can be seen in Figure 17. The algorithm has an accuracy level of 90.35%.

Random Forest

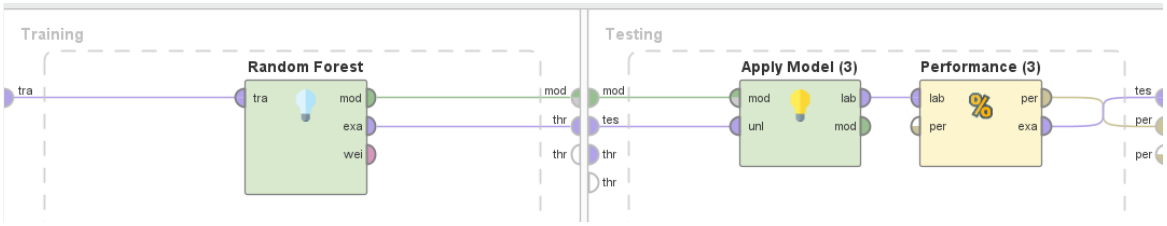

Figure 18. Random Forest Algorithm Using Rapidminer

In the training process, the Random Forest algorithm process will be run (Figure 18), and for the testing section, the Apply Model and Performance processes will be used to produce the results of the process and its accuracy.

accuracy: 90.95% +/- 0.76% (micro average: 90.95%)

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	3022	220	92	61	89.01%
pred. Cloudy	156	2924	140	116	87.65%
pred. Sunny	85	107	3025	88	91.53%
pred. Snowy	37	49	43	3035	95.92%
class recall	91.58%	88.61%	91.67%	91.97%	

Figure 19. Random Forest Algorithm Results

The results of the random forest algorithm can be seen in Figure 19. The algorithm has an accuracy rate of 90.95%. This accuracy rate is the highest compared to other algorithms.

Deep Learning

Figure 20. Deep Learning Algorithm Using Rapidminer

In the training process, the Random Forest algorithm process will be run Figure 20, and for the testing section, the Apply Model and Performance processes will be used to produce the results of the process and its accuracy.

accuracy: 90.74% +/- 0.91% (micro average: 90.74%)

	true Rainy	true Cloudy	true Sunny	true Snowy	class precision
pred. Rainy	3015	212	91	71	88.96%
pred. Cloudy	150	2940	181	102	87.16%
pred. Sunny	68	82	2962	66	93.20%
pred. Snowy	67	66	66	3061	93.90%
class recall	91.36%	89.09%	89.76%	92.76%	

Figure 21. Deep Learning Algorithm Results

The results of the deep learning algorithm can be seen in Figure 21. The algorithm has an accuracy level of 90.74%.

4.3.3. Comparison

The evaluation phase involves observing the prediction results and comparing several classification algorithms (Table 1), namely Naïve Bayes, K-NN, Decision Tree, Random Forest, and Deep Learning. This process allows us to determine which algorithm has the best accuracy in predicting weather.

Model	Hold Out	Cross Validation
k-NN	88.84%	88.80%
Random Forest	90.35%	90.95%
Naïve Bayes	87.60%	87.14%
Decision Tree	90.08%	90.35%
Deep Learning	90.93%	90.74%

Table 1. Comparison of classification model accuracy

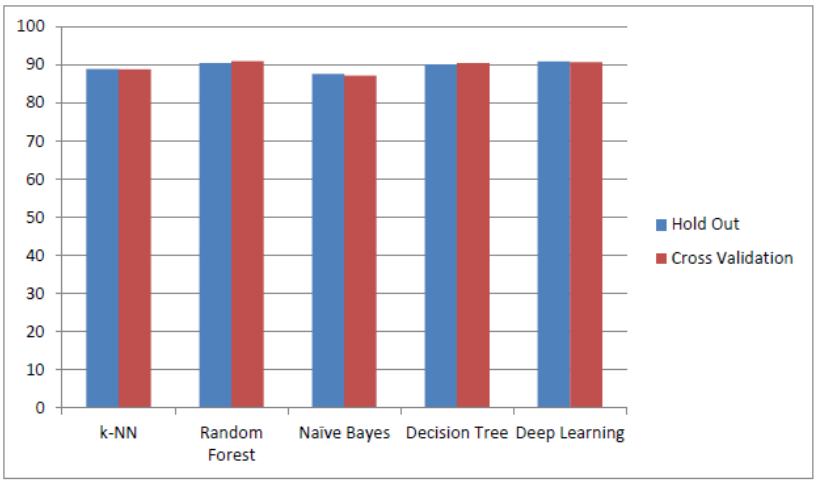

Figure 22. Comparison of classification algorithm model accuracy

Judging from the accuracy comparison in Table 1 and Figure 22, we can conclude that the most suitable classification algorithms for weather prediction are Random Forest and Deep Learning. If using the cross-validation method, Random Forest is the most suitable, while for Hold Out, Deep Learning is the most suitable.

4. Deployment Model

Deployment model is the process of taking a trained machine learning model and converting it into an application for use by users or other systems in the real world. The application will be based on the Random Forest algorithm, using the Python programming language.

4.1 Model Training

Before coding the application, the first step is to train the model. Model training is the process by which a computer learns from data to recognize patterns and make predictions or decisions based on them. Model training will use the following code.

4.1.1 Import Library

The program begins by importing the necessary libraries (Figure 23). Pandas is used for data manipulation, while train_test_split from sklearn.model_selection is used to divide the dataset into training and test data. LabelEncoder from sklearn.preprocessing converts categorical labels into numbers. Next, RandomForestClassifier from sklearn.ensemble is used to train the machine learning algorithm. Finally, joblib is used to save the trained model so it can be reused without retraining.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import joblib
```

Figure 23. Import Library in coding

4.1.2 Load Data and Encode Labels

The program reads a file named "weather_classification_data.xlsx" using the read_excel function from the Pandas library and stores it in the df variable (Figure 24). This data will be used as a source of information for training the weather classification model.

```
# Load data
df = pd.read_excel("weather_classification_data.xlsx")
```

Figure 24. Coding load data

The program uses LabelEncoder to convert category values such as "Cloud Cover," "Season," "Location," and "Weather Type" into numeric representations. Each column is processed in a loop, and its encoder is stored in the label_encoders dictionary for reuse during decoding or new predictions (Figure 25).

```
# Label encode
label_encoders = {}
categorical_cols = ['Cloud Cover', 'Season', 'Location', 'Weather Type']
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
```

Figure 25. Coding encode data

5.1.3 Split Data and Train Model

This step will separate the data into two main parts: features (X) and labels (y). Features contain all columns except "Weather Type," while labels contain the "Weather Type" column to be predicted. The data is then split into training and testing data in an 80:20 ratio using train_test_split, ensuring that the data is tested on the portion of the data that the model has not previously trained on.

Next, a classification model was created using the Random Forest algorithm with parameters n_estimators=200 (the number of trees in the forest) and max_depth=10 (the maximum tree depth) to control model complexity. This model was then trained using the training data (X_train, y_train) to recognize patterns in the given weather data (Figure 26).

```
# Pisahkan fitur dan label
X = df.drop("Weather Type", axis=1)
y = df["Weather Type"]

# Split data 80% data latih dan 20% data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = RandomForestClassifier(
    n_estimators=200,
    max_depth=10,
    random_state=42
)
model.fit(X_train, y_train)
```

Figure 26. Coding split data and train model

5.1.4 Prediksi and Save

The model evaluation process continues using the test data. The previously trained model is used to make predictions on the test data (X_test), and the results are stored in the y_pred variable. Next, accuracy is calculated using the accuracy_score function, which compares the predicted results with the original labels (y_test). The accuracy value is then displayed in a two-digit decimal format for ease of reading.

After the evaluation is complete, the model is saved to a file named "weather_model.pkl" using joblib.dump, so it can be reused without retraining. In addition to the model, all LabelEncoder objects used to transform the category values are also saved to the "label_encoders.pkl" file. Finally, the program prints a message stating that the model training and saving were successful (Figure 27).

```
# Prediksi pada data uji
y_pred = model.predict(X_test)

# Hitung dan tampilkan akurasi
accuracy = accuracy_score(y_test, y_pred)
print(f"Akurasi model: {accuracy:.2f}")

# Save model & encoders
joblib.dump(model, "weather_model.pkl")
joblib.dump(label_encoders, "label_encoders.pkl")

print("Model trained and saved successfully.")
```

Figure 27. Coding data prediction and saving model

Finally, after the entire code has been successfully executed without error, the output will appear as in the image above which displays "Model accuracy: 0.91" as the result of the model performance evaluation, as well as the message "Model trained and saved successfully." which indicates that the model training and saving process has been completed successfully (Figure 28).

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\S4_DataScienceII\Test1> & C:/Users/user/AppData/Local/Programs/Python/Python313/python.exe c:/S4_DataScienceII/Test1/train_model.py
Akurasi model: 0.91
Model trained and saved successfully.
```

Figure 28. Output from coding train_model.py

5.2 Application

Once the training process is complete and the model is successfully saved, the next step is to create an application code that allows users to make weather predictions directly using the model that has been created.

5.2.1 Import Library

The program imports the tkinter module as tk, which is used to create the graphical interface (Figure 29). Additionally, ttk and messagebox from tkinter are imported to provide more modern GUI elements and notification functionality to the user. The joblib module is also imported to load previously saved models and encoders for use in the prediction process.

```
import tkinter as tk
from tkinter import ttk, messagebox
import joblib
```

Figure 29. Import library for application coding

5.2.2 Load model and encoder

Initially, the prediction model (weather_model.pkl) and the LabelEncoder object (label_encoders.pkl) are loaded using joblib.load (Figure 30). Next, the features list contains the names of the features the user will input, such as temperature, humidity, wind speed, and categorical data such as season and location. To distinguish between input types, categorical fields are explicitly defined in categorical_fields. The predict_weather() function is defined to handle the prediction process when the user presses the prediction button in the application. This function takes input from the user through the GUI entries component and then processes each input value. If the input falls into a category like "Season" or "Location," the value is converted to numeric form using the LabelEncoder. Otherwise, the value is converted to a float. All these input values are then stored in the input_data list and used as input to the model. After that, we can add coding to the application display according to our taste.

```python
# Load model dan encoder
model = joblib.load("weather_model.pkl")
label_encoders = joblib.load("label_encoders.pkl")

features = [
    "Temperature", "Humidity", "Wind Speed", "Precipitation (%)", "Cloud Cover",
    "Atmospheric Pressure", "UV Index", "Season", "Visibility (km)", "Location"
]

categorical_fields = ["Cloud Cover", "Season", "Location"]

def predict_weather():
    try:
        input_data = []
        for feat in features:
            val = entries[feat].get()
            if feat in categorical_fields:
                val = label_encoders[feat].transform([val])[0]
            else:
                val = float(val)
            input_data.append(val)

        pred = model.predict([input_data])[0]
        result = label_encoders["Weather Type"].inverse_transform([pred])[0]
        messagebox.showinfo("🌤 Hasil Prediksi Cuaca", f"🌧 Cuaca diprediksi: {result.upper()}")
    except Exception as e:
        messagebox.showerror("Terjadi Kesalahan", str(e))
```

Figure 30. Coding load model and encoder

5.3 Implementation of Coding

Based on the code above, after running, the program will produce an interface display like the following which allows users to enter weather data and view the prediction results directly (Figure 31).

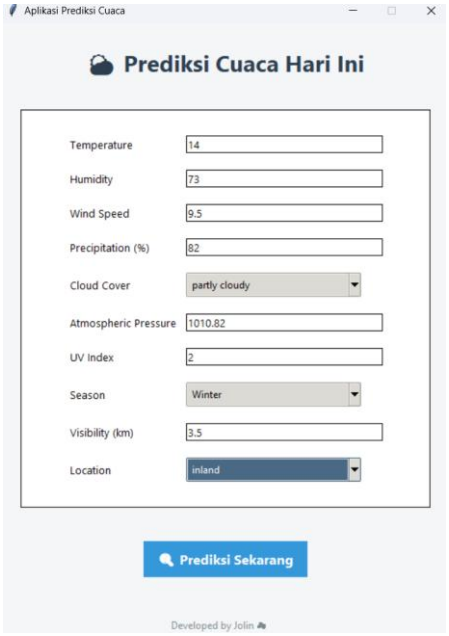

Figure 31. Display of the weather prediction program after data input

In this display, users can enter numbers corresponding to the weather conditions they wish to predict. Once all data is entered and the program is run, the system will process the input and display the weather prediction results. An example can be seen in the following illustration. After users enter weather data into the app, the system automatically processes it and displays the forecast. Based on the input provided, the forecast indicates that the expected weather conditions are "rainy" (Figure 32)

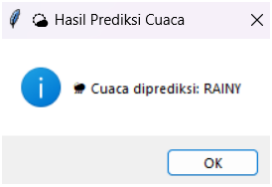

Figure 32. Weather Prediction Results

6. Conclusions

Classification algorithms have their own advantages and disadvantages in weather prediction. Deep Learning excels for large datasets with complex patterns but requires high resources. Random Forest provides accurate results and is resistant to overfitting, suitable for medium-sized datasets with many features, although it is slower for inference. Decision Trees are easy to understand and fast to train, but are prone to overfitting without pruning. k-NN is effective for small datasets with local patterns but less efficient on large datasets. Naive Bayes is fast and simple but often inaccurate due to the feature independence assumption, which rarely holds for weather data. The choice of algorithm depends on the dataset size, pattern complexity, and resource availability. While weather cannot be measured, it can be predicted.

References

[1] J. Brawijaya, F. Djohan, and K. M. Suryaningrum, "Aplikasi Pendeteksi Dan Analisa Cuaca Menggunakan Metode K-Nearest Neighbor Berbasis Android," *JSTIE (Jurnal Sarj. Tek. Inform.*, vol. 8, no. 2, p. 43, 2020, doi: 10.12928/jstie.v8i1.14623.

[2] A. K. Budiningtyas, "Analisis Kesulitan Siswa Dalam Menemukan Gagasan Pokok Pada Tema Cuaca Subtema Pengaruh Cuaca Bagi Kehidupan Manusia Pada Siswa Kelas Iii Sekolah Dasar," *INOPENDAS J. Ilm. Kependidikan*, vol. 5, no. 2, pp. 75–81, 2022, doi: 10.24176/jino.v5i2.7707.

[3] D. Sartika and D. I. Sensuse, "Perbandingan Algoritma Klasifikasi Naive Bayes, Nearest Neighbour, dan Decision Tree pada Studi Kasus Pengambilan Keputusan Pemilihan Pola Pakaian," *Jatisi*, vol. 1, no. 2, pp. 151–161, 2017, [Online]. Available: https://jurnal.mdp.ac.id/index.php/jatisi/article/view/78

[4] N. B. Putri and A. W. Wijayanto, "Analisis Komparasi Algoritma Klasifikasi Data Mining Dalam Klasifikasi Website Phishing," *Komputika J. Sist. Komput.*, vol. 11, no. 1, pp. 59–66, 2022, doi: 10.34010/komputika.v11i1.4350.

[5] S. Adiningsi, "Mengenal Metode Algoritma Klasifikasi dan Penerapannya pada Data Mining," Kompasiana. [Online]. Available: https://www.kompasiana.com/sriadiningsi7834/63500b0d08a8b535984c06d3/mengenal-metode-algoritma-klasifikasi-dan-penerapannya-pada-data-mining#google_vignette

[6] Adminlp2m, "Algoritma k-Nearest Neighbors (k-NN) – Pengertian dan Penerapan," LP2M. [Online]. Available: https://lp2m.uma.ac.id/2023/02/16/algoritma-k-nearest-neighbors-k-NN-pengertian-dan-penerapan/

[7] M. Rohmah, "Apa itu Random Forest? Pengertian, Cara Kerja & Contohnya," dibimbing.id. [Online]. Available: https://dibimbing.id/blog/detail/apa-itu-random-forest-pengertian-cara-kerja-contohnya

[8] S. Lestari, A. Akmaludin, and M. Badrul, "Implementasi Klasifikasi Naive Bayes Untuk Prediksi Kelayakan Pemberian Pinjaman Pada Koperasi Anugerah Bintang Cemerlang," *PROSISKO J. Pengemb. Ris. dan Obs. Sist. Komput.*, vol. 7, no. 1, pp. 8–16, 2020, doi: 10.30656/prosisko.v7i1.2129.

[9] I. M. B. Adnyana, "Implementasi Naïve Bayes Untuk Memprediksi Waktu Tunggu Alumni Dalam Memperoleh Pekerjaan," *Semin. Nas. Teknol. Komput. Sains*, vol. 1, no. 1, pp. 131–134, 2020, [Online]. Available: http://prosiding.seminar-id.com/index.php/sainteks/article/view/418

[10] M. R. D. Septian, A. A. A. Paliwang, M. Cahyanti, and E. R. Swedia, "Penyakit Tanaman Apel Dari Citra Daun Dengan Convolutional Neural Network," *Sebatik*, vol. 24, no. 2, pp. 207–212, 2020, doi: 10.46984/sebatik.v24i2.1060.

[11] A. Wibowo and I. R. Mahartika, "Data Mining Klasterisasi dengan Algoritme K-Means untuk Pengelompokkan Provinsi Berdasarkan Konsumsi Bahan Bakar Minyak Nasional." [Online]. Available: https://seminar.iaii.or.id/index.php/SISFOTEK/article/view/108

[12] N. Narayan, "Weather Type Classification," 2024, [Online]. Available: https://www.kaggle.com/datasets/nikhil7280/weather-type-classification

[13] W. Handoko and M. Iqbal, "Prediksi Peminatan Program Studi Pada Penerimaan Mahasiswa Baru Stmik Royal Menggunakan Naïve Bayes," *J. Sci. Soc. Res.*, vol. 4, no. 2, p. 231, 2021, doi: 10.54314/jssr.v4i2.661.

[14] F. Harits Muzaki, W. Joko Pranoto, and M. Kalimantan Timur, "Analisis Regresi Linear Dalam Data Mining Untuk Prediksi Sijil Off Di Ksop Kelas I Samarinda," *J. Ilmu Tek.*, vol. 1, no. 2, pp. 261–266, 2024, [Online]. Available: https://doi.org/10.62017/tektonik

[15] S. Asyuti and A. A. Setyawan, "Data Mining Dalam Penggunaan Presensi Karyawan Denga Cluster Means," *J. Ilm. Sains Teknol. Dan Inf.*, vol. 1, no. 1, pp. 01–10, 2023, [Online]. Available: https://jurnal.alimspublishing.co.id/index.php/JITI/article/download/6/6