



ProbWeatherPredict

Overview

Develop a weather prediction model using machine learning with Python, specifically leveraging the Gaussian Naive Bayes algorithm to predict weather conditions based on historical data.

Theory

In the weather prediction model using the Gaussian Naive Bayes algorithm, probability is used to make predictions about future weather conditions based on historical data.

Here's a detailed explanation of how probability is utilized:

Bayes' Theorem

The Gaussian Naive Bayes algorithm is based on Bayes' Theorem, which provides a way to update the probability estimate for a hypothesis as more evidence or information becomes available. Bayes' Theorem is expressed as:

What is Bayes Theorem?

- Bayes' theorem relates the conditional and unconditional probabilities of events A and B, where B has a non-zero probability:

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

- Each term in Bayes' theorem has a conventional name:
 - $P(A)$ is the prior probability or unconditional probability of A.
 - It is "prior" in the sense that it does not take into account any information about B.
 - $P(A|B)$ is the conditional probability of A, given B.
 - $P(B|A)$ is the conditional probability of B given A.
 - $P(B)$ is the prior or marginal probability of B

Gaussian Naive Bayes Algorithm

The Gaussian Naive Bayes algorithm applies Bayes' Theorem with the assumption that the features (e.g., temperature, humidity, pressure) are normally distributed and independent of each other given the class (weather condition).

What Is Gaussian Naive Bayes Classifier?

Gaussian Naive Bayes is a machine learning classification technique based on a probabilistic approach that assumes each class follows a normal distribution. It assumes each parameter has an independent capacity of predicting the output variable. It is able to predict the probability of a dependent variable to be classified in each group.

What Is Gaussian Distribution?

Gaussian distribution is also called [normal distribution](#). Normal distribution is a statistical model that describes the distributions of continuous random variables in nature and is defined by its bell-shaped curve. The two most important features of the normal distribution are the [mean](#) (μ) and [standard deviation](#) (σ). The mean is the average value of a distribution, and the standard deviation is the “width” of the distribution around the mean.

A variable (X) that is normally distributed is distributed continuously (continuous variable) from $-\infty < X < \infty$, and the total area under the model curve is 1.


Steps in the Gaussian Naive Bayes Algorithm:

1. **Calculate Prior Probabilities ($P(H)P(H)P(H)$):**
 - Determine the prior probability for each weather condition based on historical data. For example, the probability of a sunny day, rainy day, etc.
2. **Calculate Likelihood ($P(E | H)P(E | H)P(E | H)$):**
 - For each feature (e.g., temperature, humidity), calculate the likelihood using the Gaussian (normal) distribution:
3. **Calculate Posterior Probabilities ($P(H | E)P(H | E)P(H | E)$):**
 - Use Bayes' Theorem to calculate the posterior probability for each weather condition given the new evidence (current weather feature values)
4. **Predict the Weather Condition:**
 - Choose the weather condition with the highest posterior probability as the predicted outcome.

Example

Suppose we want to predict whether tomorrow will be sunny or rainy based on today's temperature and humidity. The steps would be:

1. **Calculate Priors:**
 - $P(\text{Sunny})=0.6$ $P(\text{Sunny}) = 0.6$ $P(\text{Sunny})=0.6$
 - $P(\text{Rainy})=0.4$ $P(\text{Rainy}) = 0.4$ $P(\text{Rainy})=0.4$
2. **Calculate Likelihoods:**
 - Given today's temperature and humidity, calculate the likelihood for sunny and rainy conditions using the Gaussian distribution.
3. **Calculate Posteriors:**
 - Combine the priors and likelihoods to get the posterior probabilities for sunny and rainy conditions.
4. **Make Prediction:**
 - Predict the condition with the higher posterior probability.



This probabilistic approach allows the model to quantify uncertainty and make predictions based on the most probable weather conditions given the observed data.

Key Components

1. **Data Collection:** Gather historical weather data, including parameters such as temperature, humidity, pressure, wind speed, and precipitation.
2. **Data Preprocessing:** Clean and preprocess the data to handle missing values, normalize features, and split the data into training and testing sets.
3. **Feature Selection:** Identify relevant features that significantly impact weather conditions.
4. **Model Development:** Implement the Gaussian Naive Bayes algorithm to build the prediction model.
5. **Model Training:** Train the model using the training dataset, optimizing the parameters to improve prediction accuracy.
6. **Model Evaluation:** Evaluate the model's performance on the testing dataset using metrics such as accuracy, precision and recall.
7. **Prediction:** Use the trained model to predict future weather conditions based on new input data.

Specifications

A reliable and efficient weather prediction model that can forecast weather conditions with reasonable accuracy, aiding in decision-making for various applications such as agriculture, transportation, and event planning.

DATASET

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Outlook	Temp	Humidity	Windy	Play																		
2	Rainy	Hot	High	f	no																		
3	Rainy	Hot	High	t	no																		
4	Overcast	Hot	High	f	yes																		
5	Sunny	Mild	High	f	yes																		
6	Sunny	Cool	Normal	f	yes																		
7	Sunny	Cool	Normal	t	no																		
8	Overcast	Cool	Normal	t	yes																		
9	Rainy	Mild	High	f	no																		
10	Rainy	Cool	Normal	f	yes																		
11	Sunny	Mild	Normal	f	yes																		
12	Rainy	Mild	Normal	t	yes																		
13	Overcast	Mild	High	t	yes																		
14	Overcast	Hot	Normal	f	yes																		
15	Sunny	Mild	High	t	no																		
16																							
17																							
18																							
19																							
20																							
21																							
22																							
23																							
24																							
25																							
26																							
27																							
28																							
29																							
30																							
31																							

Summary and Explanation of the Naive Bayes Weather Prediction Model

In this project, we use a Naive Bayes classifier to predict whether to play tennis based on weather conditions such as outlook, temperature, humidity, and whether it is windy. The steps are detailed below:

Step-by-Step Explanation

Libraries Used:

- **Pandas**: used data manipulation and analysis

- **Scikit learn**: Machine learning library that provides simple and efficient tools for data mining and data analysis
- **Job lib**: used for saving trained model
- **Flask**: lightweight and flexible web framework for Python

Step 1: Installing Dependencies

We first installed the necessary libraries: **pandas** for data manipulation and **sklearn** (scikit-learn) for machine learning.

```
pip install pandas
pip install sklearn
```

Step 2: Importing Libraries

We import the required modules: **pandas** for handling data, **LabelEncoder** for encoding categorical variables, and **GaussianNB** for the Naive Bayes classifier.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
```

Step 3: Reading the CSV

We load the dataset into a DataFrame using pandas. The dataset contains weather conditions and the decision to play or not.

```
df = pd.read_csv("new_dataset.csv")
df
```

Step 4: Encoding Strings into Numbers

Since the Naive Bayes classifier in scikit-learn requires numerical input, we use `LabelEncoder` to convert categorical variables into numerical values.

```
Numerics = LabelEncoder()
```

Step 5: Dropping the Target Column

We separate the features (input variables) from the target (output variable).

```
inputs = df.drop('Play', axis='columns')
target = df['Play']
```

Step 6: Creating a New DataFrame with Encoded Values

We encode each categorical feature into numerical values and add these encoded values to the DataFrame.

```
inputs['outlook_n'] = Numerics.fit_transform(inputs['Outlook'])
inputs['Temp_n'] = Numerics.fit_transform(inputs['Temp'])
inputs['Hum_n'] = Numerics.fit_transform(inputs['Humidity'])
inputs['win_n'] = Numerics.fit_transform(inputs['Windy'])
inputs
```

Step 7: Dropping the Original String Columns

We remove the original categorical columns, leaving only the numerical representations.

```
inputs_n = inputs.drop(['Outlook', 'Temp', 'Humidity', 'Windy'],
axis='columns')
inputs_n
```

Step 8: Applying Gaussian Naive Bayes

We initialize the Naive Bayes classifier and fit it to our encoded feature set and target.

```
classifier = GaussianNB()  
classifier.fit(inputs_n, target)
```

Step 9: Prediction

We use the trained model to make a prediction. For example,

`classifier.predict([[0,0,0,1]])` might correspond to a specific weather condition (encoded values).

```
classifier.predict([[0,0,0,1]])
```

How the Model Predicts

1. **Data Preparation**: The weather data is first prepared by encoding categorical variables into numerical values. This step is crucial because machine learning algorithms in scikit-learn generally do not handle categorical data directly.
2. **Training**: The Naive Bayes classifier is trained using the historical data. During this training, the classifier calculates the probabilities of different weather conditions leading to a decision to play tennis. It learns the likelihood of playing tennis given certain conditions.
3. **Prediction**: For new data, the classifier calculates the posterior probabilities for each class (play or not play) based on the input features using Bayes' theorem. It then predicts the class with the highest probability.

Example Prediction

If the input is `[[0,0,0,1]]`, this represents a specific set of weather conditions (e.g., outlook, temperature, humidity, windy). The classifier uses the probabilities

learned during training to predict whether tennis will be played under these conditions.

- **Outlook:** Encoded as 0 (e.g., Sunny)
- **Temperature:** Encoded as 0 (e.g., Hot)
- **Humidity:** Encoded as 0 (e.g., High)
- **Windy:** Encoded as 1 (e.g., True)

The model will compute the likelihood of playing tennis based on these conditions and predict the outcome.

MODEL DEPICTION

(FAVORABLE CONDITION)

The screenshot displays the ProbWeatherPredict web application. The header includes the title "ProbWeatherPredict" and navigation links for "Home" and "About". The main content area is divided into two sections. On the left, there is a form with four dropdown menus labeled "Outlook", "Temp", "Humidity", and "Windy", each with a "Choose..." placeholder and a downward arrow. Below these is a yellow "Submit" button. On the right, there is a large thumbs-up icon with a blue cloud and a yellow sun in the background. Below the icon, the text "Favorable weather for playing" is displayed. The footer contains the copyright notice "© 2024 My Website. All rights reserved." and a link to "WeatherPlay on GitHub".

(UNFAVORABLE CONDITION)

ProbWeatherPredict [Home](#) [About](#)


Outlook Choose... ▾

Temp Choose... ▾

Humidity Choose... ▾

Windy Choose... ▾

Submit



Not Suitable for Playing

© 2024 My Website. All rights reserved.
WeatherPlay on GitHub

(ABOUT THE MODEL)

ProbWeatherPredict [Home](#) [About](#)

About

In this project, we use a machine learning algorithm called Gaussian Naive Bayes to predict whether it's suitable to play a game based on weather conditions like outlook, temperature, humidity, and wind. The model is trained on historical weather data, where each feature (like "Sunny" or "Rainy" for outlook) is converted into numbers, and it learns the patterns associated with playing or not playing. Gaussian Naive Bayes assumes that these features follow a normal distribution and calculates probabilities for each possible outcome based on these features. When given new weather conditions, the model uses what it learned to predict whether it's a good day to play, by determining which outcome (play or not play) is most likely based on the current conditions.

Developers

Machine Learning Model: **Ojasvi Raj 23BA110164**
Full-Stack Development: **Priyam Jain 23BA111158**

© 2024 My Website. All rights reserved.
WeatherPlay on GitHub