# CS 165A – Artificial Intelligence, Spring 2024
# Machine Problem 1

Responsible TA: Xuan Luo, xuan_luo@ucsb.edu

Due: May 16, 2024 11:59pm

## 1 Notes

■ Make sure to read the "Policy on Academic Integrity" in the course syllabus.
■ Any updates or corrections will be posted on Canvas.
■ You must work individually on this assignment.
■ Each student must submit their report and code electronically.
■ If you have any questions about MP1, please post them on the Canvas MP1 Q&A.
■ Plagiarism Warning: We will use software to check for plagiarism. You are expected to complete the project independently without using any code from others.

## 2 Problem Definition

This project involves developing a weather forecasting algorithm using the Naïve Bayes method. Your task is to predict the weather conditions for the next time step based on the historical data of the previous 28 time steps. This data includes variables such as weather condition, temperature, wind speed, wind direction, etc. You are expected to design and implement the Naïve Bayes algorithm from scratch, using these historical data points to train your model.

It is important to note that the use of pre-existing Naïve Bayes libraries, such as those found in Scikit-learn, **is not permitted**. You must code the Naïve Bayes algorithm yourself. You need to use Python3 for this programming assignment. You can use any built-in module, as well as numpy, pandas or scipy. Any machine learning libraries, e.g. Scikit-learn, are not allowed.

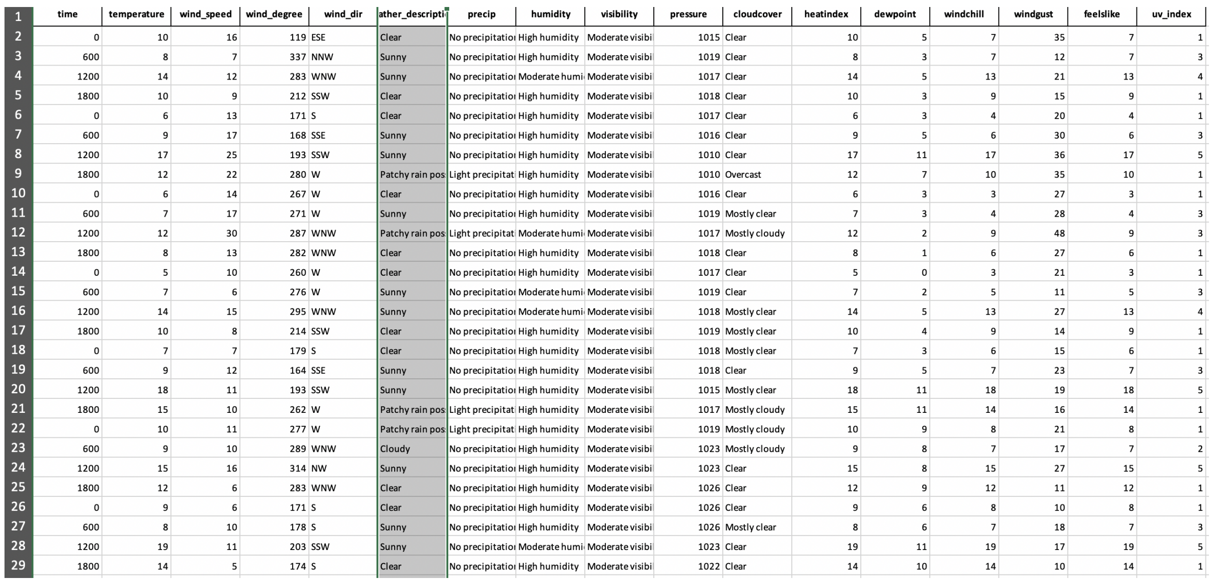
## 3 Data Format

Figure 1 displays an example of the dataset format, which includes 28 data points. The 'time' column indicates when the data was recorded (e.g., 0 for 00:00, 600 for 06:00, 1800 for 18:00). Therefore, the 28 data points will correspond to the weather history of previous 7 days. Features such as temperature, wind speed, wind degree, pressure, heat index, dew point, wind chill, wind gust, feels like, and UV index are represented as integer values. Categorical features include wind direction, weather descriptions, precipitation, humidity, visibility, and cloud cover.

In our test, we will provide you with 28 consecutive data points, and you are tasked with predicting the 'weather descriptions' for the subsequent, or 29th, time step. For example, if Figure 1 represents data for 28 time points, you will use this historical data to forecast the weather conditions for the next point in the sequence. In Figure 1, the 29th data point actually means the weather at 00:00 am on the 8th day. You can use any features from the 28 historical data points as your feature variables ('X') to predict the target variable ('y'), which in this case is the weather condition of the 29th time step.

## 4 Program Input and Output

Your program is required to handle two file paths as command-line arguments. The first path should lead to a training dataset named `./training.xlsx`, and the second should point to a folder containing 1000 test samples, designated as `./tests`. This folder includes files named `test1.xlsx, test2.xlsx,`



| 1 | time | temperature | wind_speed | wind_degree | wind_dir | ather_descripti | precip | humidity | visibility | pressure | cloudcover | heatindex | dewpoint | windchill | windgust | feelslike | uv_index |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0 | 10 | 16 | 119 | ESE | Clear | No precipitatio | High humidity | Moderate visibi | 1015 | Clear | 10 | 5 | 7 | 35 | 7 | 1 |
| 3 | 600 | 8 | 7 | 337 | NNW | Sunny | No precipitatio | High humidity | Moderate visibi | 1019 | Clear | 8 | 3 | 7 | 12 | 7 | 3 |
| 4 | 1200 | 14 | 12 | 283 | WNW | Sunny | No precipitatio | Moderate humi | Moderate visibi | 1017 | Clear | 14 | 5 | 13 | 21 | 13 | 4 |
| 5 | 1800 | 10 | 9 | 212 | SSW | Clear | No precipitatio | High humidity | Moderate visibi | 1018 | Clear | 10 | 3 | 9 | 15 | 9 | 1 |
| 6 | 0 | 6 | 13 | 171 | S | Clear | No precipitatio | High humidity | Moderate visibi | 1017 | Clear | 6 | 3 | 4 | 20 | 4 | 1 |
| 7 | 600 | 9 | 17 | 168 | SSE | Sunny | No precipitatio | High humidity | Moderate visibi | 1016 | Clear | 9 | 5 | 6 | 30 | 6 | 3 |
| 8 | 1200 | 17 | 25 | 193 | SSW | Sunny | No precipitatio | High humidity | Moderate visibi | 1010 | Clear | 17 | 11 | 17 | 36 | 17 | 5 |
| 9 | 1800 | 12 | 22 | 280 | W | Patchy rain pos | Light precipitat | High humidity | Moderate visibi | 1010 | Overcast | 12 | 7 | 10 | 35 | 10 | 1 |
| 10 | 0 | 6 | 14 | 267 | W | Clear | No precipitatio | High humidity | Moderate visibi | 1016 | Clear | 6 | 3 | 3 | 27 | 3 | 1 |
| 11 | 600 | 7 | 17 | 271 | W | Sunny | No precipitatio | High humidity | Moderate visibi | 1019 | Mostly clear | 7 | 3 | 4 | 28 | 4 | 3 |
| 12 | 1200 | 12 | 30 | 287 | WNW | Patchy rain pos | Light precipitat | Moderate humi | Moderate visibi | 1017 | Mostly cloudy | 12 | 2 | 9 | 48 | 9 | 3 |
| 13 | 1800 | 8 | 13 | 282 | WNW | Clear | No precipitatio | High humidity | Moderate visibi | 1018 | Clear | 8 | 1 | 6 | 27 | 6 | 1 |
| 14 | 0 | 5 | 10 | 260 | W | Clear | No precipitatio | High humidity | Moderate visibi | 1017 | Clear | 5 | 0 | 3 | 21 | 3 | 1 |
| 15 | 600 | 7 | 6 | 276 | W | Sunny | No precipitatio | Moderate humi | Moderate visibi | 1019 | Clear | 7 | 2 | 5 | 11 | 5 | 3 |
| 16 | 1200 | 14 | 15 | 295 | WNW | Sunny | No precipitatio | Moderate humi | Moderate visibi | 1018 | Mostly clear | 14 | 5 | 13 | 27 | 13 | 4 |
| 17 | 1800 | 10 | 8 | 214 | SSW | Clear | No precipitatio | High humidity | Moderate visibi | 1019 | Mostly clear | 10 | 4 | 9 | 14 | 9 | 1 |
| 18 | 0 | 7 | 7 | 179 | S | Clear | No precipitatio | High humidity | Moderate visibi | 1018 | Mostly clear | 7 | 3 | 6 | 15 | 6 | 1 |
| 19 | 600 | 9 | 12 | 164 | SSE | Sunny | No precipitatio | High humidity | Moderate visibi | 1018 | Clear | 9 | 5 | 7 | 23 | 7 | 3 |
| 20 | 1200 | 18 | 11 | 193 | SSW | Sunny | No precipitatio | High humidity | Moderate visibi | 1015 | Mostly clear | 18 | 11 | 18 | 19 | 18 | 5 |
| 21 | 1800 | 15 | 10 | 262 | W | Patchy rain pos | Light precipitat | High humidity | Moderate visibi | 1017 | Mostly cloudy | 15 | 11 | 14 | 16 | 14 | 1 |
| 22 | 0 | 10 | 11 | 277 | W | Patchy rain pos | Light precipitat | High humidity | Moderate visibi | 1019 | Mostly cloudy | 10 | 9 | 8 | 21 | 8 | 1 |
| 23 | 600 | 9 | 10 | 289 | WNW | Cloudy | No precipitatio | High humidity | Moderate visibi | 1023 | Mostly cloudy | 9 | 8 | 7 | 17 | 7 | 2 |
| 24 | 1200 | 15 | 16 | 314 | NW | Sunny | No precipitatio | High humidity | Moderate visibi | 1023 | Clear | 15 | 8 | 15 | 27 | 15 | 5 |
| 25 | 1800 | 12 | 6 | 283 | WNW | Clear | No precipitatio | High humidity | Moderate visibi | 1026 | Clear | 12 | 9 | 12 | 11 | 12 | 1 |
| 26 | 0 | 9 | 6 | 171 | S | Clear | No precipitatio | High humidity | Moderate visibi | 1026 | Clear | 9 | 6 | 8 | 10 | 8 | 1 |
| 27 | 600 | 8 | 10 | 178 | S | Sunny | No precipitatio | High humidity | Moderate visibi | 1026 | Mostly clear | 8 | 6 | 7 | 18 | 7 | 3 |
| 28 | 1200 | 19 | 11 | 203 | SSW | Sunny | No precipitatio | Moderate humi | Moderate visibi | 1023 | Clear | 19 | 11 | 19 | 17 | 19 | 5 |
| 29 | 1800 | 14 | 5 | 174 | S | Clear | No precipitatio | High humidity | Moderate visibi | 1022 | Clear | 14 | 10 | 14 | 10 | 14 | 1 |

Figure 1: Example of 28 data points.

`..., test1000.xlsx`, each file consisting of 28 rows of sequential historic weather data (refer to Figure 1 for an example). The tasks for your program are to: (1) train the Naive Bayes algorithm using `training.xlsx`, and (2) test the algorithm on all files in `./tests` and output the results to the terminal. For each test, the output should be a string indicating the "weather_descriptions" for the subsequent data point. The accuracy of your program will be evaluated based on how closely the output aligns with the actual weather conditions. You can verify your accuracy using the `ground_truth.json` file, which contains the correct answers for the 1000 test samples provided.

**Example Input and Output** Below is an example of how to execute your program and the expected output format. The output will be 1000 rows contains the predicted weathers for `test1.xlsx, test2.xlsx, ..., test1000.xlsx`. All output should be directed in the standard output, DO NOT write the results in a file.

**Example Input:**
```
python3 NaiveBayesClassifier.py training.xlsx tests
```

**Example Output:**
```
Cloudy
Sunny
...
(1000 rows in total)
```

During grading we will use the same training dataset but a different testing dataset (which is called the private testing dataset). So don't use the public testing dataset to train your classifier hoping to achieve greater accuracy. This is called overfitting and it would make your classifier very accurate on this specific dataset only, but not on unseen one. Also, there will be a time limit of 10 minutes and your program must finish execution within this time. If it runs for more than 10 minutes, the grader will terminate your program and your accuracy score will be 0.

## 5 Hints for implementation

To streamline the problem initially, consider using only the features from the most recent time step (the 28th data point) to predict the weather conditions for the upcoming 29th time step. This approach simplifies the task and helps you achieve a basic result quickly.

Once you have established this baseline, you can experiment with different configurations of features. Begin by adding features from earlier time steps or selecting different combinations of features to see how they impact the performance of your model. This methodical approach allows you to identify the most effective features for enhancing your forecast accuracy.

## 6 Submission Guidelines

Please follow these instructions to ensure your submissions are processed correctly by the autograder and eligible for manual grading if necessary.

### 6.1 Gradescope Submission

We will create two distinct assignments on Gradescope for this module:

**MP1:** Submit only your PDF report to this assignment.

**MP1_code:** Submit all code-related documents here, including:

- A shell script named `NaiveBayesClassifier.sh`
- Your source code, e.g., `NaiveBayesClassifier.py`
- Any additional scripts required for execution

**Important: Do not submit any data files**. Also, do not place your scripts or code inside a directory or zip file. The autograder should be able to run your code and provide accuracy results within a few minutes. While there is no limit to the number of submissions, only the last submission will be considered for grading.

### 6.2 Execution and Environment

- **Executable File:** Ensure your program's executable is named `NaiveBayesClassifier`.
- **Wrapper Script:** Include a shell script named `NaiveBayesClassifier.sh` that executes your Python code. Example usage:

  ```
  ./NaiveBayesClassifier.sh training.xlsx tests
  ```

  This script should allow command line arguments to be passed directly to your Python script.
- **Python Version:** Clearly state the version of Python used in your project within your report. Ensure all code is compatible with this version. We will use python 3 on gradescope.

**Note:** If the Gradescope autograder does not support a package you require, please make a post on Canvas. We aim to accommodate all necessary tools to ensure your code runs smoothly.

## 7 Grading Criteria

### 7.1 Grading Breakdown

The final grade for this project will be determined based on the following criteria:

- **Complete Report:** 20%
- **Execution Specifications:** 10%
- **Correct Program Specifications:** 10%
  - Reads file names from command-line arguments
  - Produces correct standard output results
  - No segfaults or unhandled exceptions
- **Test Accuracy and Runtime:** 60%

### 7.2 Leaderboard and Bonuses

- The top scorer in classifier testing accuracy will receive a 50% bonus on their project grade.
- The top three scorers will each receive a 25% bonus.
- A real-time leaderboard will be available on Gradescope to track these rankings.

**Note:** We have a limit of 100 submissions on Gradescope.

### 7.3 Accuracy and Time

The majority of the grade (60%) is based on the testing accuracy and runtime:

- **Testing Accuracy:** Assessed on a private dataset.
- **Runtime:** Your code must complete execution within 10 minutes; otherwise, it will be terminated, and the accuracy score will be zero.

#### Score Calculation

- A naive implementation that achieves less than 0.37 accuracy will receive a 0% score for the accuracy component.
- Scores for accuracy levels:
  - $0.30 \leq \text{Accuracy} < 0.60$: Scores will be linearly interpolated between 60% and 90%.
  - $0.60 \leq \text{Accuracy} < 0.70$: Scores will be linearly interpolated between 90% and 100%.
  - $0.70 \leq \text{Accuracy} \leq 1.00$: Scores will be linearly interpolated between 100% and 120%.
- A correct implementation of Naive Bayes should achieve 0.60 accuracy.

This scoring is designed to reward improvements beyond the basic requirement and encourage innovative solutions that enhance the model's accuracy.

## 8 Report Guidelines

The project report should adhere to the following specifications:

- **Length:** The report must be between 1 and 3 pages, with no exceptions exceeding this limit.
- **Font Size:** The text must be at least 11pt in size.
- **Content Requirements:** Your report should comprehensively cover the following sections:
  1. **Architecture:** Provide a concise explanation of your code's architecture, including a description of classes and their basic functionalities.
  2. **Preprocessing:** Describe how you represent an instance within the dataset.
  3. **Model Building:** Explain the process of training your Naive Bayes Classifier.
  4. **Results:** Discuss your results on the provided datasets, including accuracy and running time.
  5. **Challenges:** Detail the challenges you encountered during the project and how you addressed them.
- **Submission Format:** Submit only one PDF report containing all the sections listed above. Do not include additional README files.
- **Identification:** Your report must include your name, email, and perm number at the top of the first page.

Please ensure all sections are clearly defined and thoroughly addressed to meet the project's evaluation criteria.