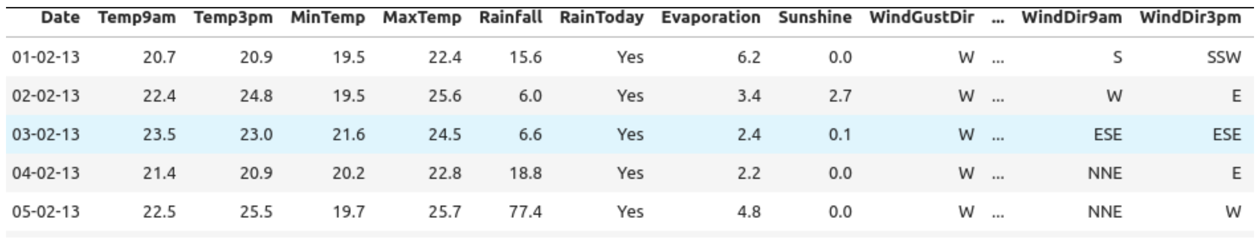
Data Analysis and Data Mining Assignment 2

Data cleaning and analysis

Author: Bolyki Balázs
Neptun code: N5IF3V
Date: 2023.12.08.

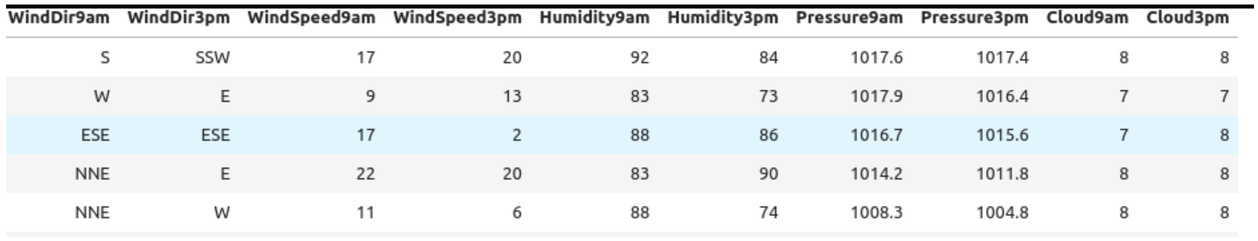
Dataset

The chosen dataset for the task is the Bangladesh weather history available at kaggle: https://www.kaggle.com/datasets/apurboshahidshawon/weatherdatabangladesh/
The data contains weather records from 2013.02.01. up to 2022.01.15. The records are available in csv format, where every row represents one day.
In the following screenshot there is an excerpt from the dataset. Most columns are in numeric format. Column names mostly speak for themselves, though there are special cases I discuss in the data cleaning step.



Date	Temp9am	Temp3pm	MinTemp	MaxTemp	Rainfall	RainToday	Evaporation	Sunshine	WindGustDir	...	WindDir9am	WindDir3pm
01-02-13	20.7	20.9	19.5	22.4	15.6	Yes	6.2	0.0	W	...	S	SSW
02-02-13	22.4	24.8	19.5	25.6	6.0	Yes	3.4	2.7	W	...	W	E
03-02-13	23.5	23.0	21.6	24.5	6.6	Yes	2.4	0.1	W	...	ESE	ESE
04-02-13	21.4	20.9	20.2	22.8	18.8	Yes	2.2	0.0	W	...	NNE	E
05-02-13	22.5	25.5	19.7	25.7	77.4	Yes	4.8	0.0	W	...	NNE	W

The dataset altogether contains 3271 rows and 21 columns.



WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm
S	SSW	17	20	92	84	1017.6	1017.4	8	8
W	E	9	13	83	73	1017.9	1016.4	7	7
ESE	ESE	17	2	88	86	1016.7	1015.6	7	8
NNE	E	22	20	83	90	1014.2	1011.8	8	8
NNE	W	11	6	88	74	1008.3	1004.8	8	8

The assignment was written in Jupyter Notebook.

Data cleaning

The dataset was particularly clean in its base form, so most cleaning steps aimed to convert string data to numerical values. The dataset was loaded into a pandas dataframe, then I started with the conversion and checking steps.

Handling dates

Date field is not of particular importance in the analysis. I have written a small code segment to check whether every day is covered from start day to end day. It simply checks if the difference in days between the start and end time of the dates equals the number of rows minus one.
Since the dataset is ordered by dates and the start date is known, the dropping of the date columns results in no information loss (provided the start date is preserved). Thus the date is simply dropped.

RainToday

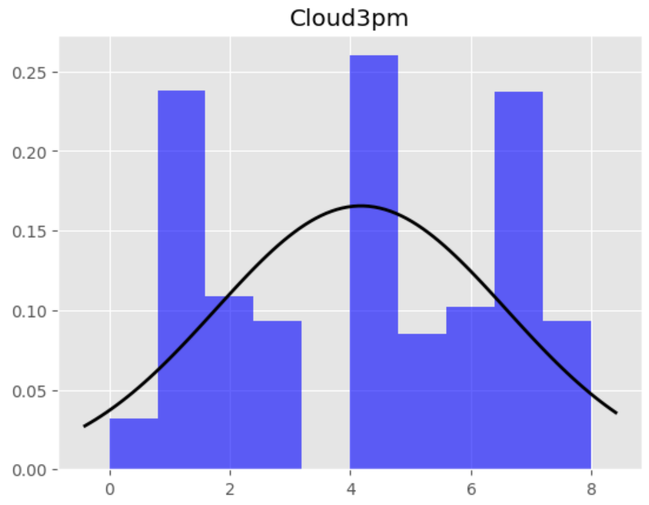
The *RainToday* field contains Yes/No values, but is somewhat special. According to the website it should be interpreted as "Is today rainy". Upon closer observation though, the records do not reflect how much it rained that day. The RainFall column tells us how many millimeters of rainfall was observed, and it does not unambiguously correlate to the RainToday column. This means the column cannot simply be dropped.
Instead of dropping the column, I created a new one: IsTodayRainy. The new column's values were loaded with ones and zeros according to RainToday. Only then was RainToday dropped.

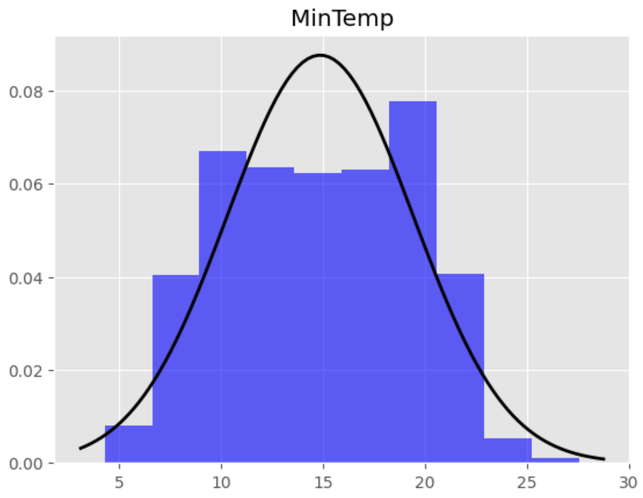
Wind directions

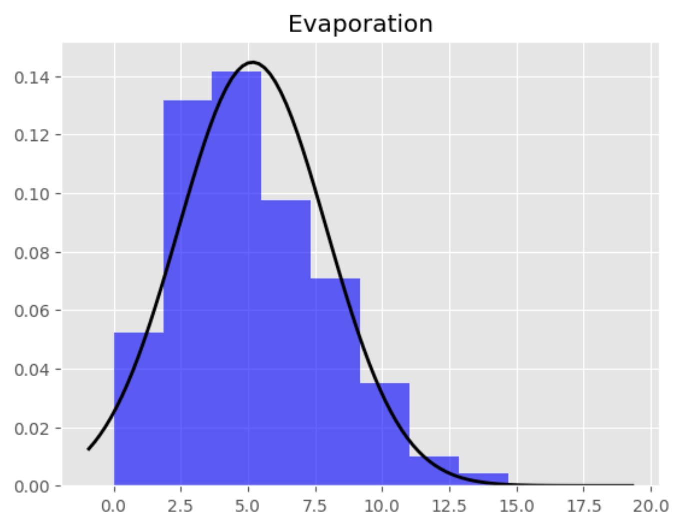
Wind directions were the most complicated to convert. There are three different columns: WindGustDir, WindDir9am, WindDir3pm. They are all in string format, where for example the SSE string could be interpreted as South South-East. That is the maximum precision the direction has.
I separated each direction column into 4 other columns. Based on WindGustDir I created WindGustDir_north, WindGustDir_south, WindGustDir_west and WindGustDir_east. I did the same for each column. These new columns are numeric, and they all contain a value in the [0;1] interval.
The values are calculated according to the wind direction. If the wind blows south, the WindGustDir_south field is set to 1. If it blows South South-East, WindGustDir_south is set to 0.66 WindGustDir_east to 0.33. This is achieved by simple counting and normalizing logic: the different letters are counted separately, then each category is divided by the number of letters.
Once the new columns are loaded, the old ones are dropped.

Initial analysis

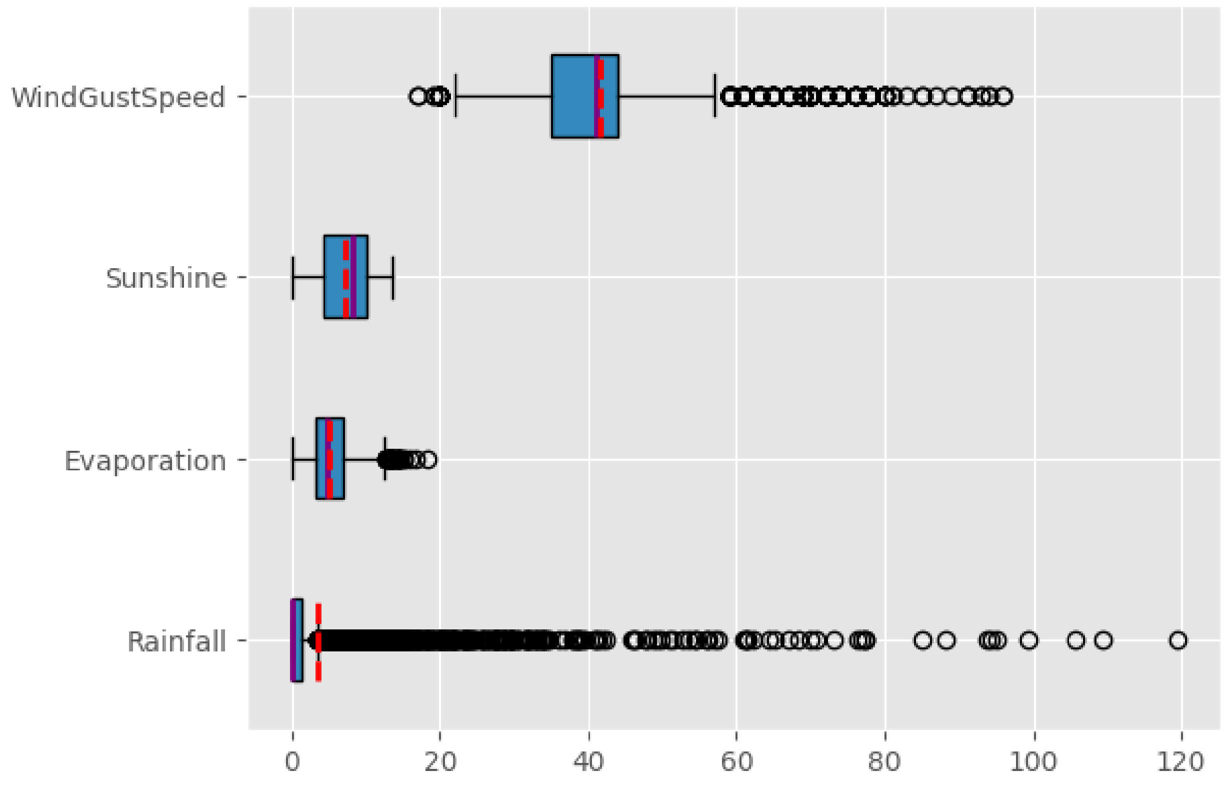
I have tried fitting bell curves on most of the values (fitting on the wind directions is obviously pointless).
Bell curve fits some columns well. For example, Evaporation bell curve fits nicely, MinTemp (Minimum Temperature) moderately, cloudiness terribly.





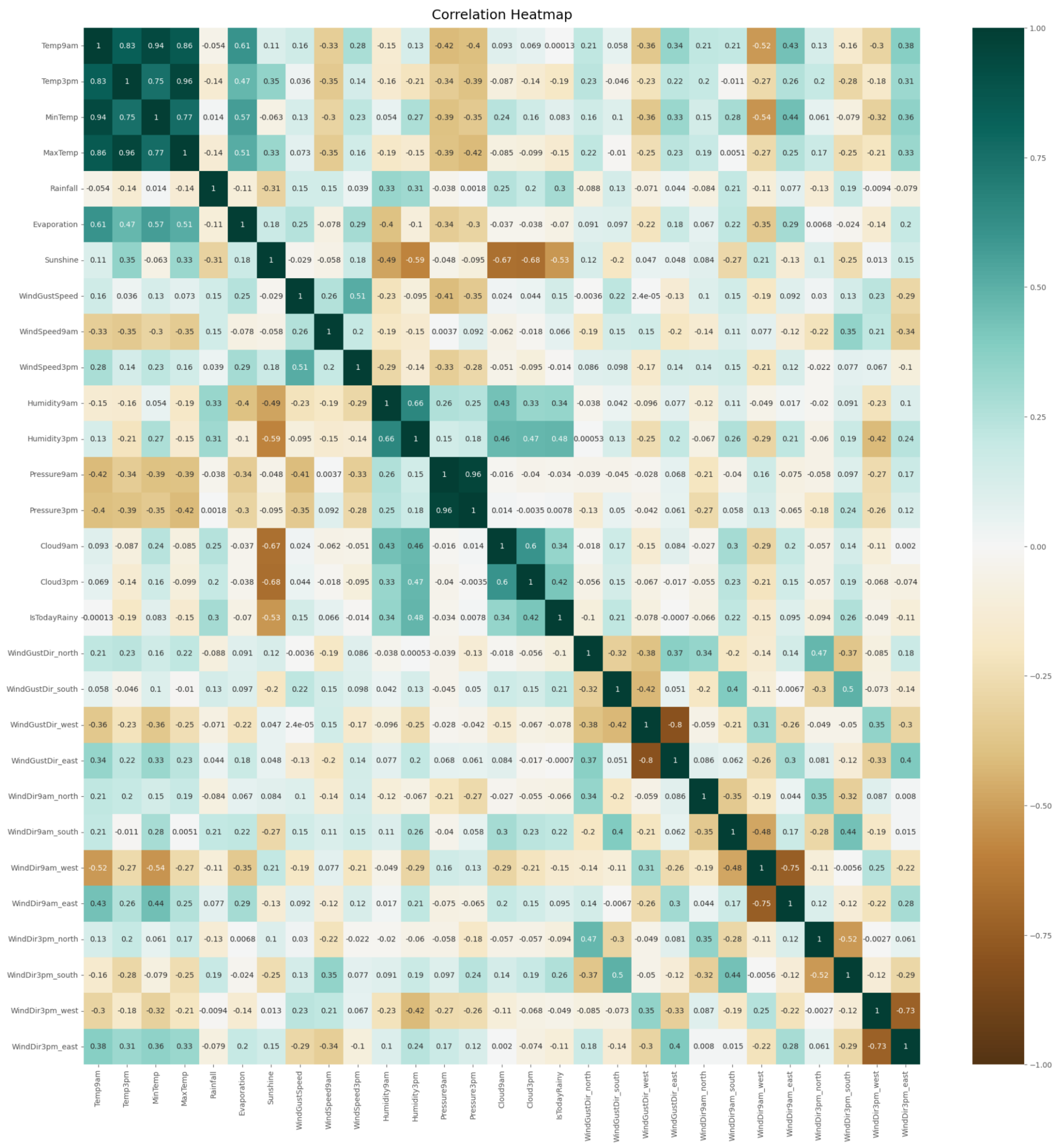


I also analyzed minimum and maximum values whether they make sense in the context. The dataset was relatively clean. For example there are outliers in the temperatures, but they are not at all unfeasible: maximum temperature is 45.8, which is well within possibilities.
Outlier analysis is also hard on some columns, rainfall for example: on most days, there is no rain at all, so everything is outside the main interval. WindGustSpeed also has some extreme values, while sunshine is very consistent.



Covariance

I have created a correlation heatmap from the dataset (see notebook for better resolution).



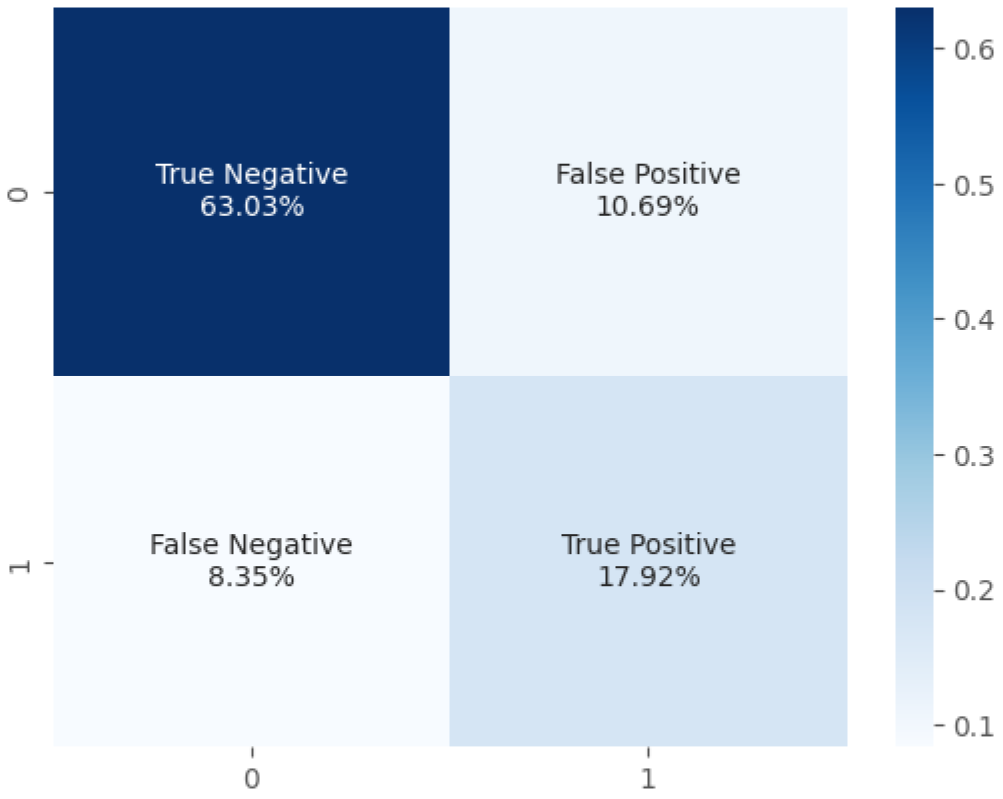
The covariance has shown correlation information between columns:

- There are positive correlations: temperatures at 9am and 3pm are very much related.
- There are inverse correlations: between cloud coverage and sunshine.

- There are also no correlating columns: air pressure and wind directions show almost no connection at all.

Classification

For classification I used the naive Bayes classifier from the *sklearn* package. The target of the classification was to determine whether a given day was rainy; the classifier attempts to predict the value of the IsTodayRainy field.
The resulting confusion matrix is shown below.



Some relevant metrics are the following:

Accuracy	0.81
Precision	0.626
Recall	0.682
F1 score	0.653