Final project Introduction to Data Science

Lâm Ngọc Phương Anh - 18127039
Nguyễn Công Anh Khoa - 18127261

Topic: WEATHER FORECASTING

sources: google image

Collect data

- We use selenium to crawl data from wunderground website.

sources: google image

- This is the sample of data that we collect. Each row of data that describes the condition of the weather is collected every 30 minutes.

Daily Observations

Time	Temperature	Dew Point	Humidity	Wind	Wind Speed	Wind Gust	Pressure	Precip.	Condition
12:00 AM	79 °F	75 °F	89 %	WSW	6 mph	0 mph	29.79 in	0.0 in	Mostly Cloudy
12:30 AM	79 °F	75 °F	89 %	W	5 mph	0 mph	29.76 in	0.0 in	Mostly Cloudy
1:00 AM	77 °F	75 °F	94 %	CALM	0 mph	0 mph	29.76 in	0.0 in	Mostly Cloudy

- From each row, we will convert it to a dictionary and then using a list to store these dictionaries.

- Using “pandas” package to create DataFrame from a list of dictionaries.
- Store data into an excel file.

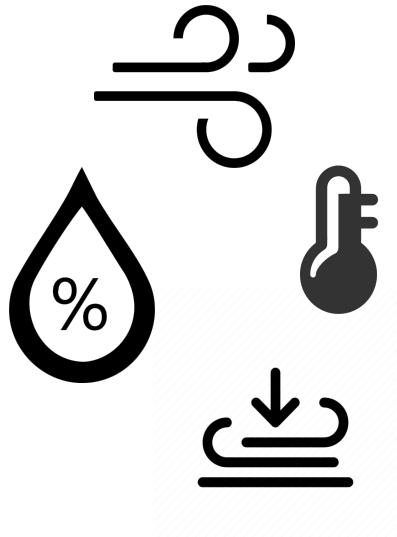
sources: google image

Explore data

- Check if data has duplicate rows, missing values.
- Describe the meaning of each column and check their data types.

Meaning of each column

Question



Predict whether it will rain or not in the following 30 minutes based on data about weather conditions of that time.

sources: google image

Preprocess Data

- Check each row of the time and date columns to be continuous, eliminating the non-continuous rows since they don't have the corresponding condition column data. Then shift the values.

	date	time	temperature	dew_point	humidity	wind	wind_speed	wind_gust	pressure	precip.	condition
0	2021-07-01	12:00 AM	81	79	94	WSW	6	0	29.76	0.0	Partly Cloudy
1	2021-07-01	12:30 AM	81	79	94	WSW	7	0	29.76	0.0	Partly Cloudy
2	2021-07-01	1:00 AM	82	79	89	SW	6	0	29.76	0.0	Fair
3	2021-07-01	1:30 AM	81	79	94	SW	6	0	29.76	0.0	Fair
4	2021-07-01	2:00 AM	81	79	94	SSW	7	0	29.73	0.0	Fair

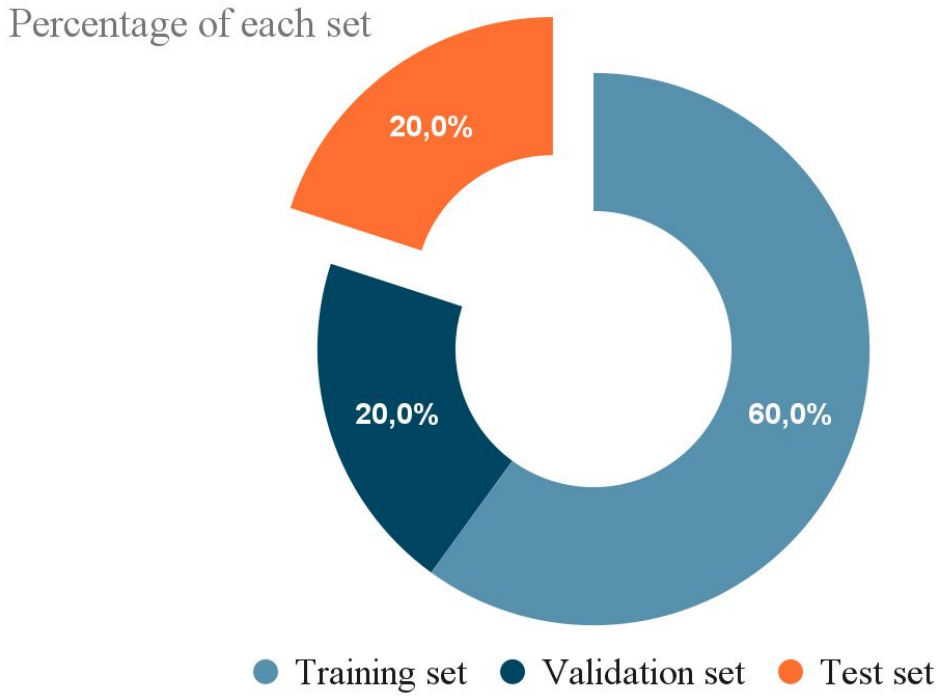
Some example rows of data before applying preprocess

	date	time	temperature	dew_point	humidity	wind	wind_speed	wind_gust	pressure	precip.	condition
0	2021-07-01	12:00 AM	81	79	94	WSW	6	0	29.76	0.0	Partly Cloudy
1	2021-07-01	12:30 AM	81	79	94	WSW	7	0	29.76	0.0	Fair
2	2021-07-01	1:00 AM	82	79	89	SW	6	0	29.76	0.0	Fair
3	2021-07-01	1:30 AM	81	79	94	SW	6	0	29.76	0.0	Fair
4	2021-07-01	2:00 AM	81	79	94	SSW	7	0	29.73	0.0	Fair

Some example rows of data after applying preprocess

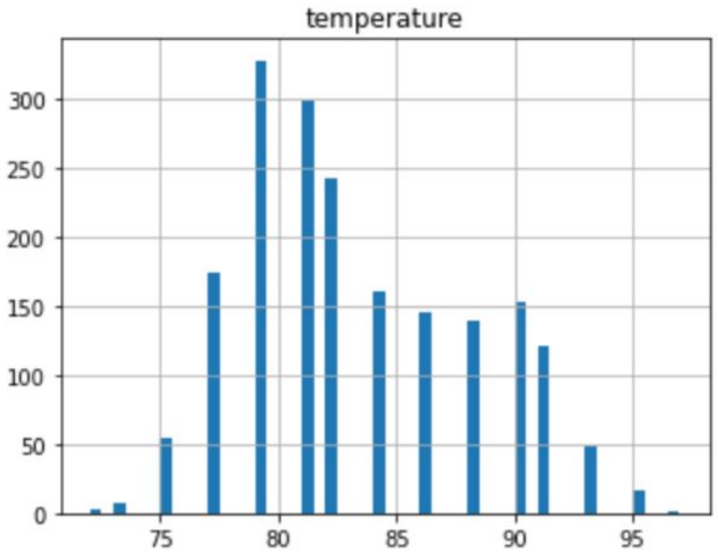
Split Data

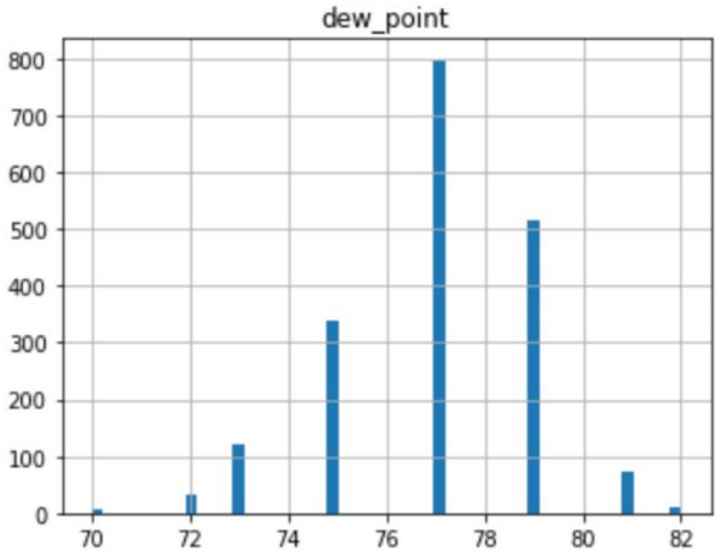
- Before we continue to preprocess our data, we will split it into 3 sets: train, validation and test (save it into 3 separate csv files).

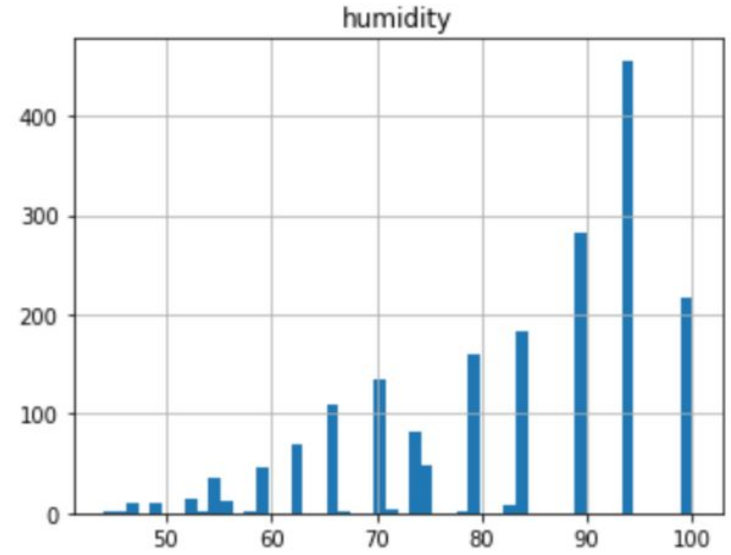


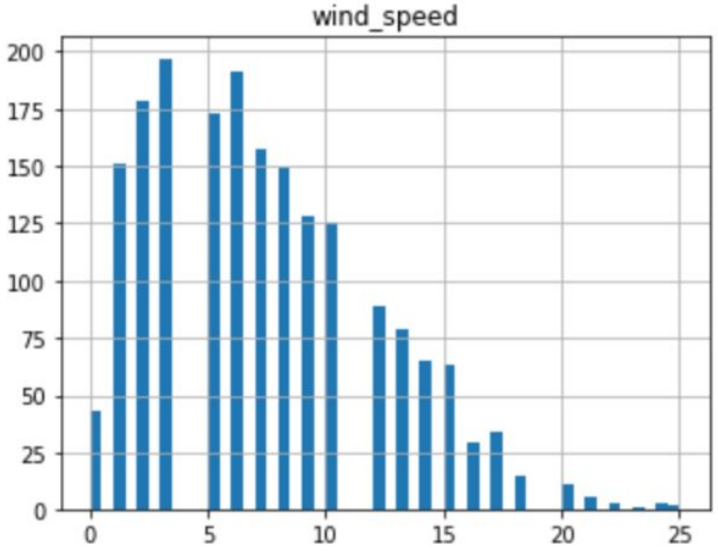
Explore Training Data

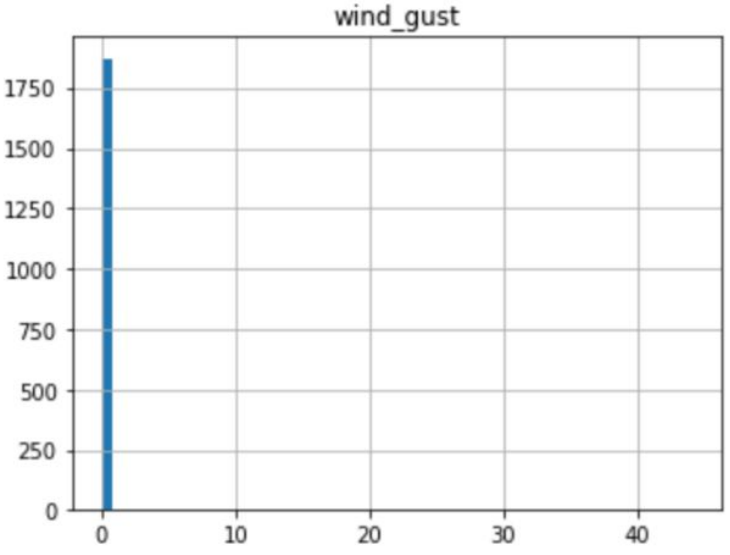
- We want to know the distribution of each data column, correlation of 2 columns, illustrates a histogram for numeric columns and a bar chart for categorical columns.

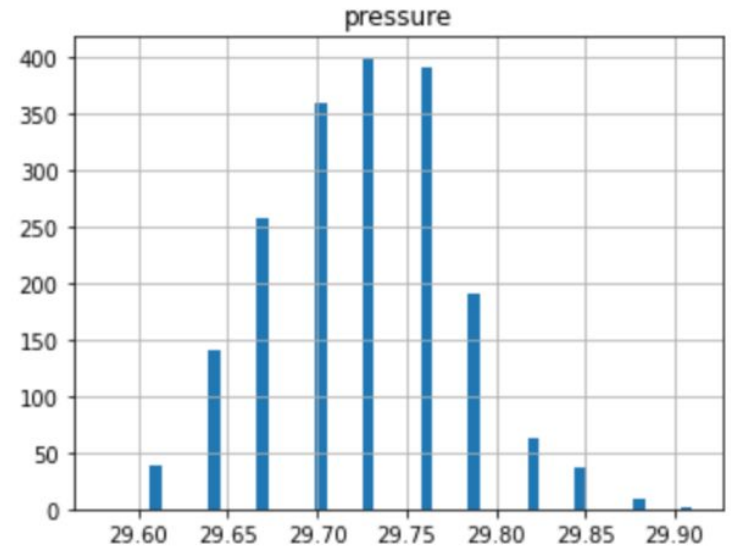




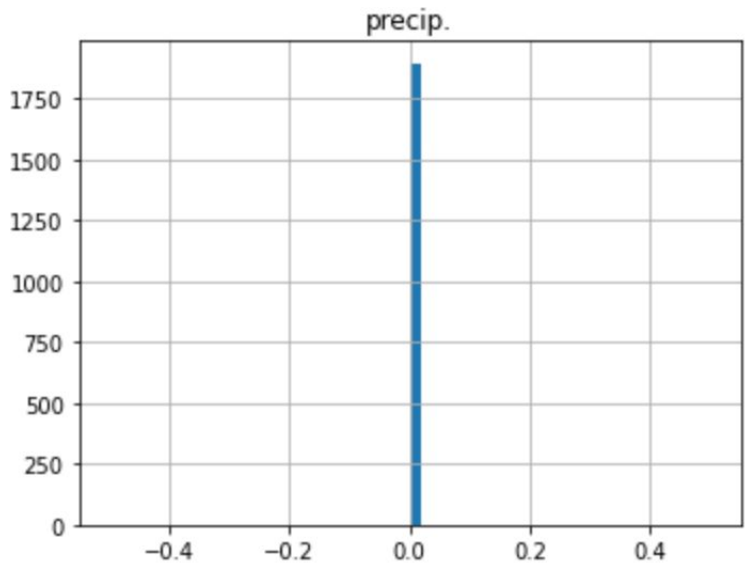






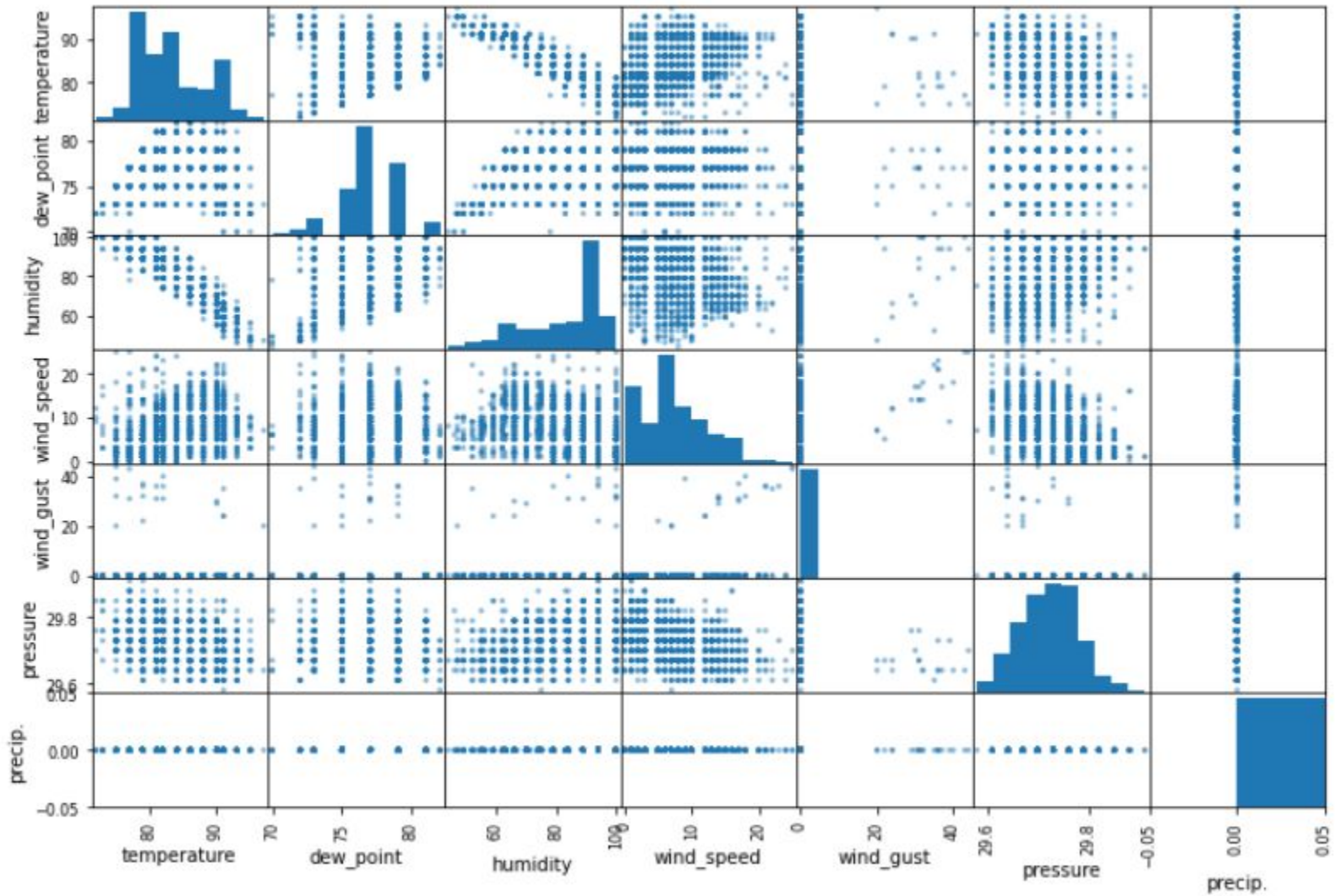


Numeric columns

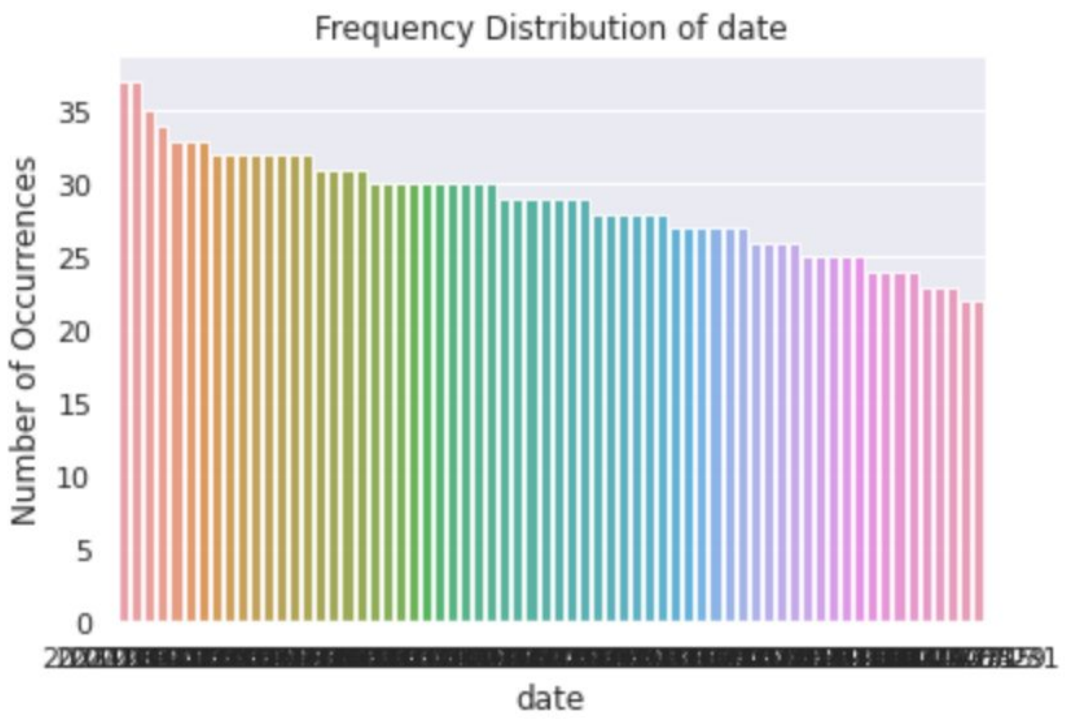


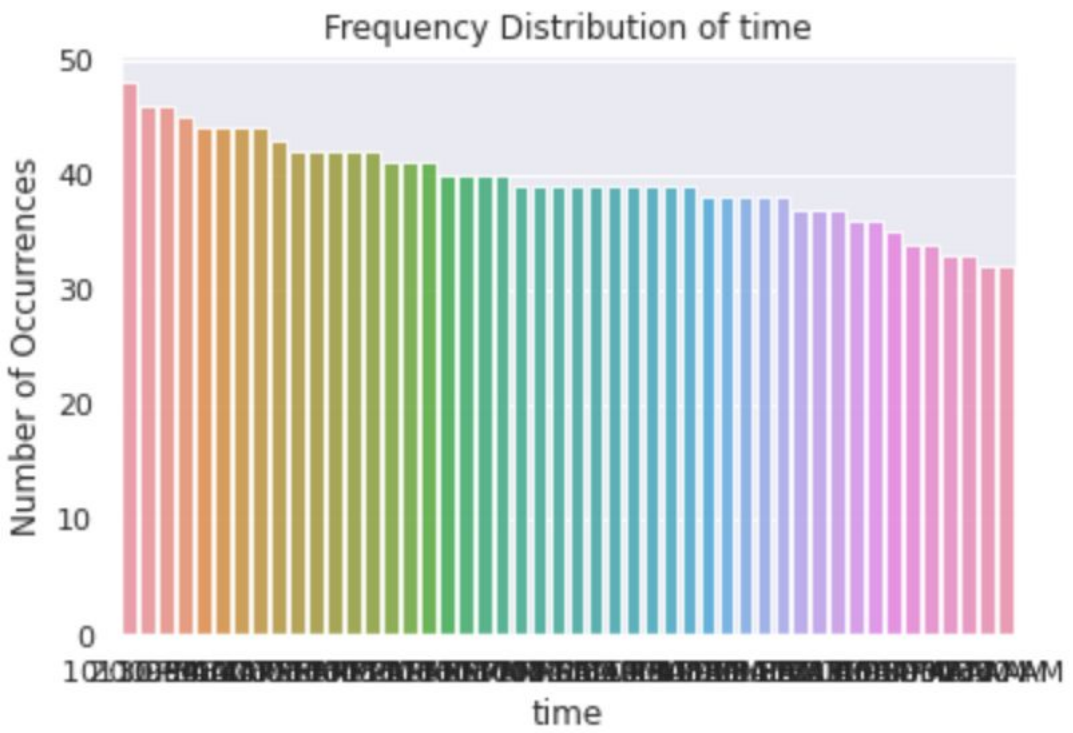
We have few observations:

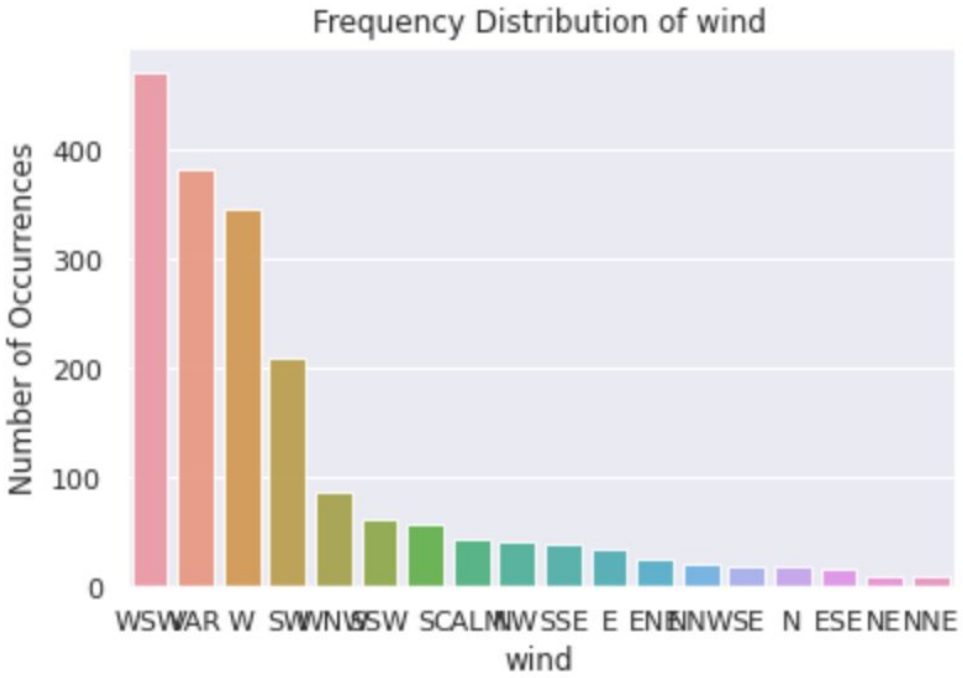
- These attributes have very different scales.
- precip. and wind_gust columns is dominated by **0**, so this attribute does not supply any information.



Scatter matrix of numeric columns







Categorical columns

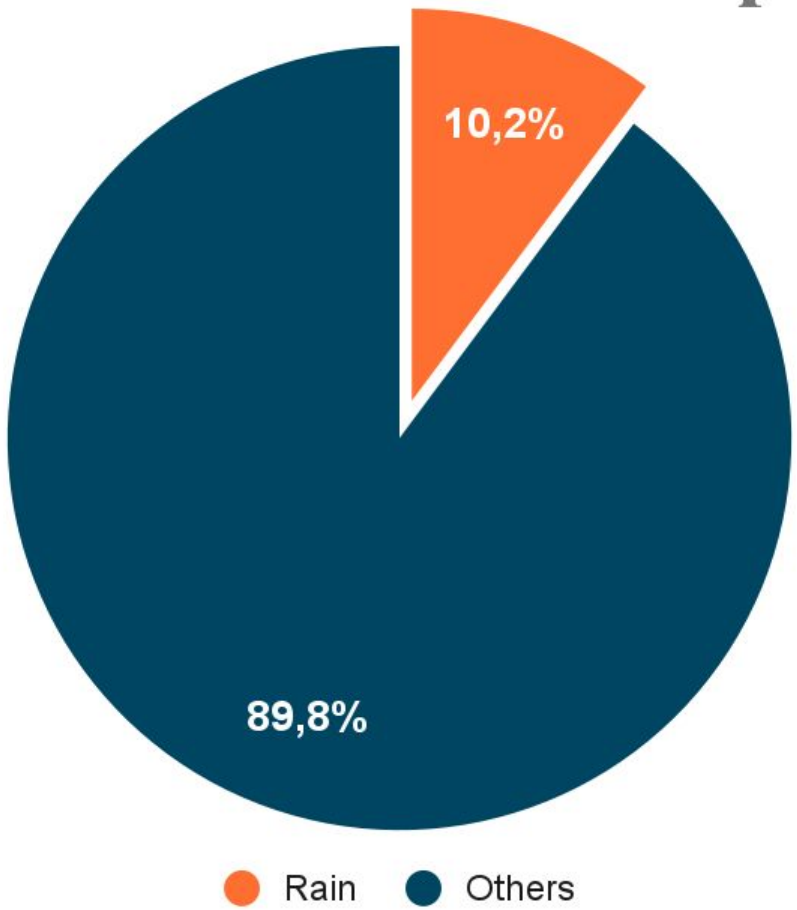
We have some observations:

- Distribution of date and time is uniform.
- wind has some values which frequency are low.
- There are different weather labels that show it's raining.

Preprocess Data

- Convert label values into “Rain” and “Others”.
- Remove column(s): precip., wind_gust, date.
- Split column time into time and day_night to decrease the number of distinct values.
- wind column: some values in wind appear few, so we use a hyperparameter to keep the topmost frequent values.

Percentage of each value after preprocess



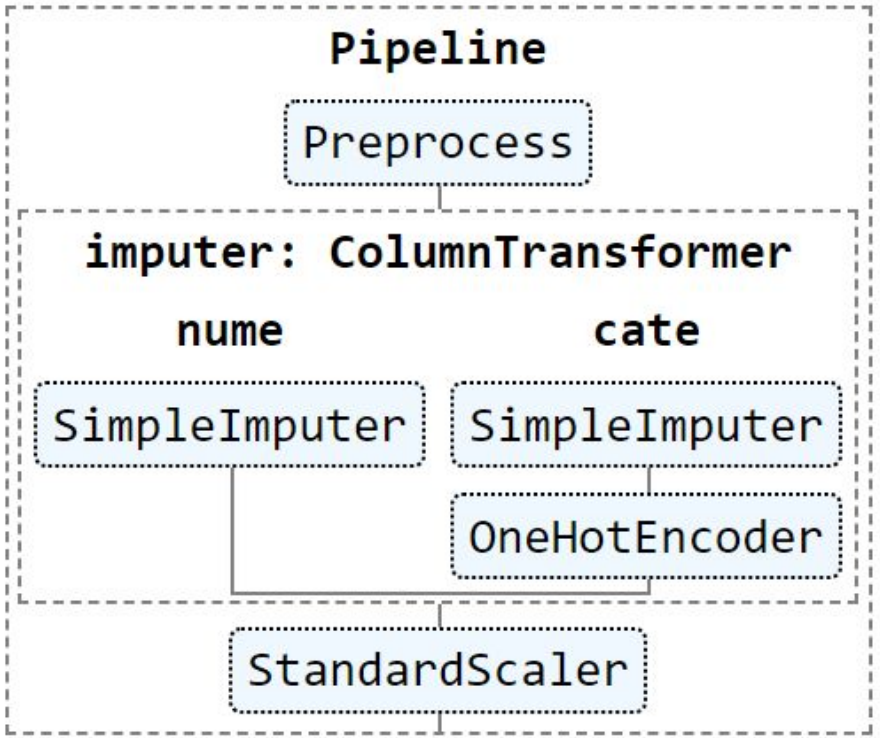
	time	temperature	dew_point	humidity	wind	wind_speed	pressure	day_night
0	8:00	82	75	79	VAR	2	29.79	AM
1	4:00	79	79	100	E	3	29.73	AM
2	4:30	79	75	89	W	7	29.67	PM
3	5:30	79	77	94	VAR	1	29.73	AM
4	2:30	79	75	89	VAR	2	29.73	AM

Some example rows of data after applying preprocess step

Preprocess Training data

We create a pipeline to make it convenience to preprocess data.

- Preprocess: remove columns, split time, keep top frequent values of wind.
- Numerical columns: fill in missing values by mean use Simple Imputer.
- Categorical columns: fill in missing values by top frequency, then apply One-hot encoder
- StandardScaler: normalize data



Models

- Traditional Machine Learning:
  + Linear model: Logistic model, Linear SVM
  + Non-linear model: kNN, Nonlinear SVM
  + Tree: Decision tree, Random Forest
- Deep learning: Neural network

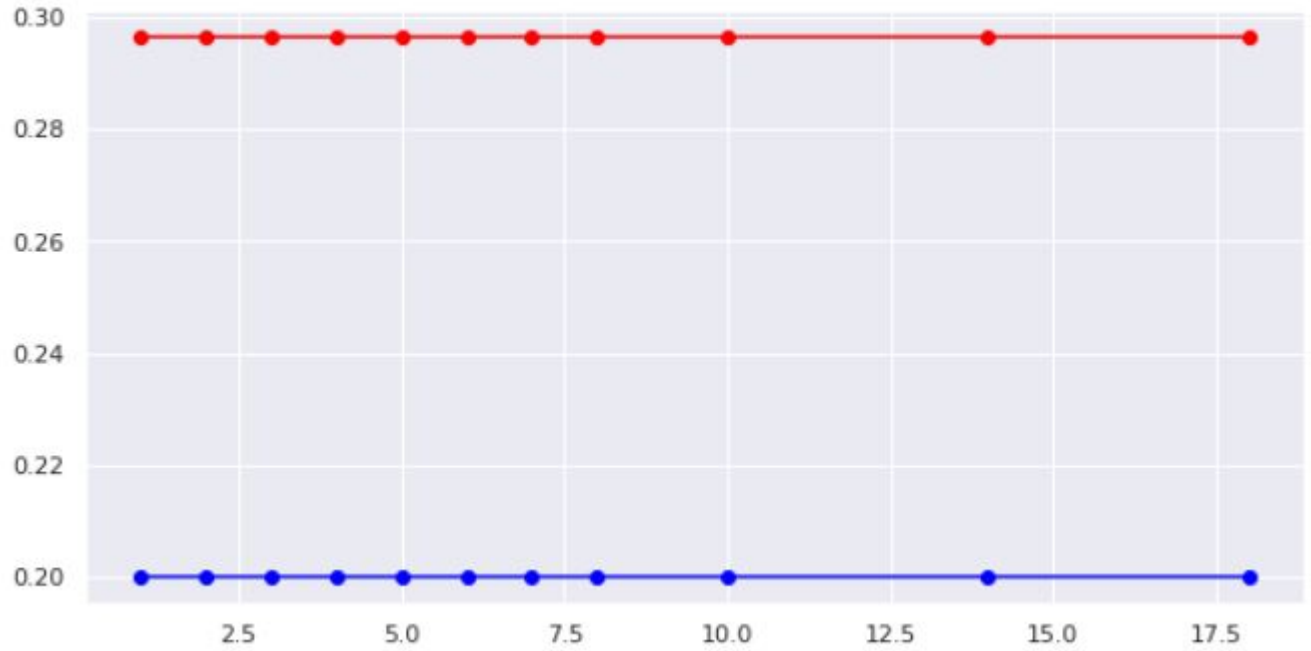
Because we focus on forecast raining weather, and in our data, raining condition is the minority. Therefore, we use the f1 score of the raining class to be our metric.

Linear Models

Logistic model

We create a pipeline for logistic model with max_iter as a parameter is 500.

```
Best hyperparameter: 1
Best validation score: 0.2
```

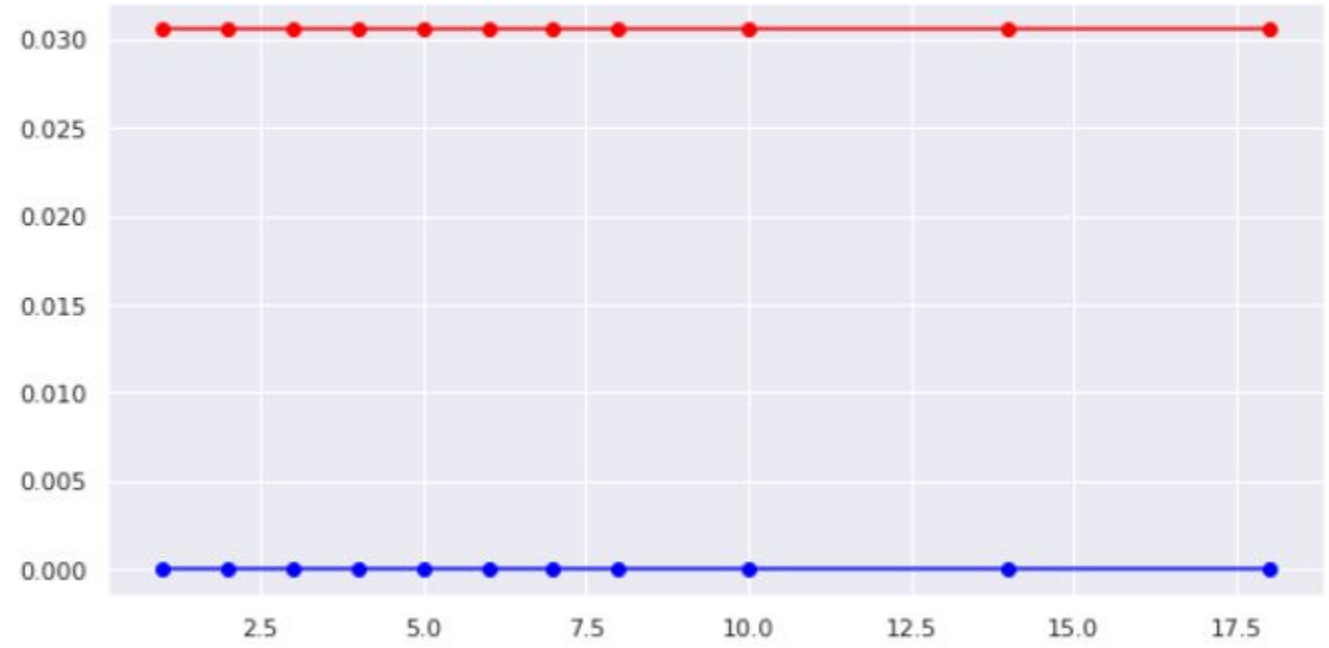


- **Comments:**
  + The model is underfitting.
  + Feature wind does not affect the model.

Linear SVM

We create a pipeline for linear SVM with kernel is “linear” as a parameter.

```
Best hyperparameter: 1
Best validation score: 0.0
```



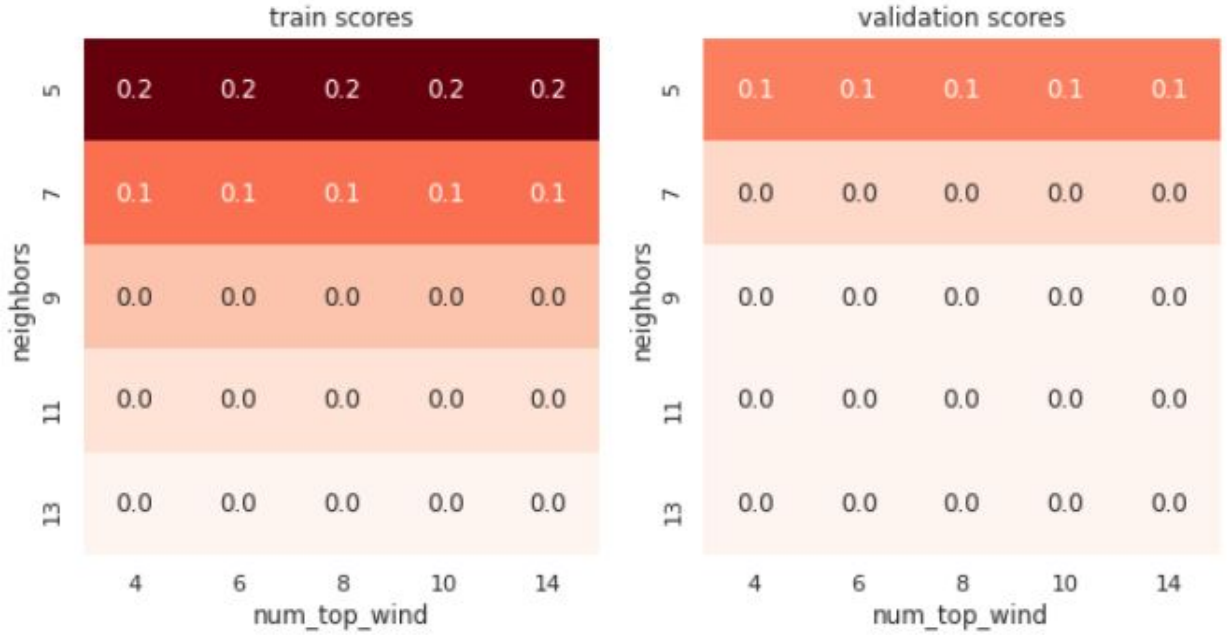
- **Comments:**
  + The model is underfitting.
  + Feature wind does not affect the model.
  + It is even worse than logistic regression.

Non-Linear Models

kNN

kNN model has 2 hyperparameters, the number of neighbors and the number of top wind. We create a pipeline for kNN model.

```
Best num_top_wind: 4
Best neighbors: 5
Best validation score: 0.07894736842105263
```



- **Comments:**

  + The model is underfitting.

  + Feature wind does not affect the model.

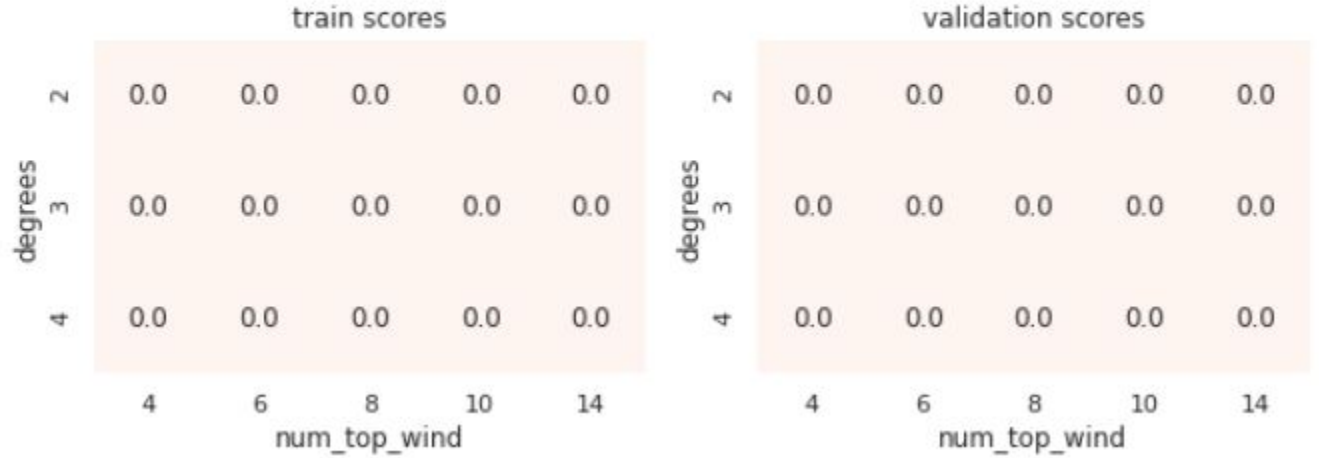
  + It is even worse than linear models.

  + As low as hyperparameter neighbors, as good as model is.

Non-Linear SVM

We create a pipeline for non-linear SVM with the kernel is “Poly”, the degrees of poly as hyperparameters.

```
Best num_top_wind: 4
Best degrees: 2
Best validation score: 0.0
```



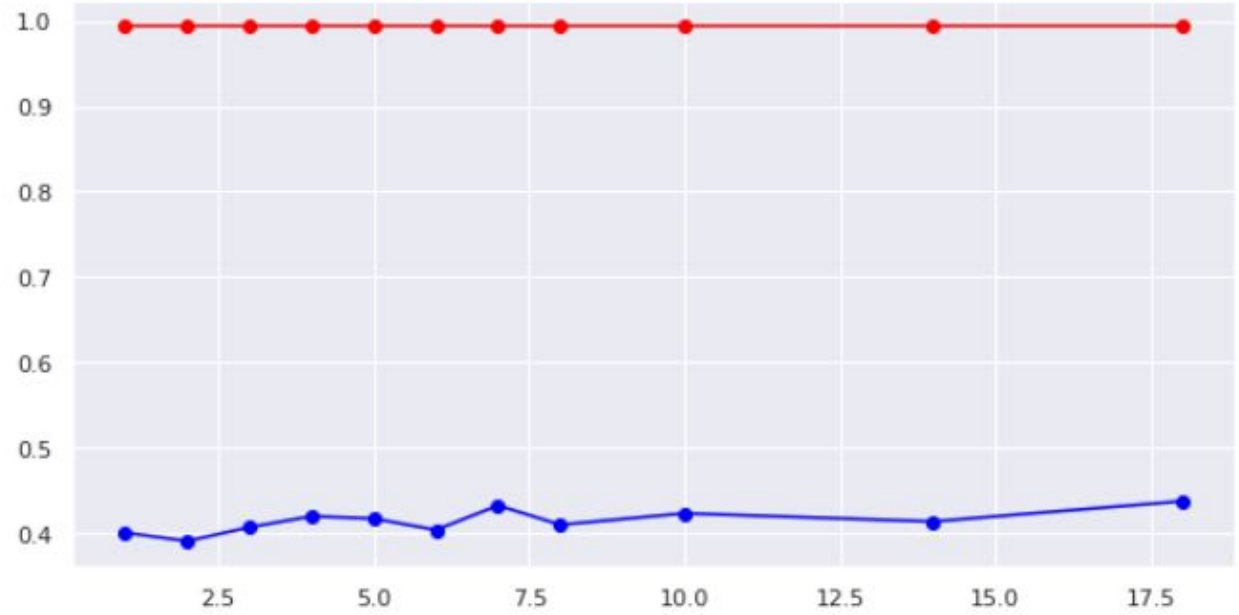
- **Comments:**
  + The model is underfitting.
  + Feature wind does not affect the model.
  + It is the worst.

Tree

Decision Tree

We create a pipeline for Decision Tree model.

```
Best num_top_wind: 18
Best validation score: 0.43697478991596633
```



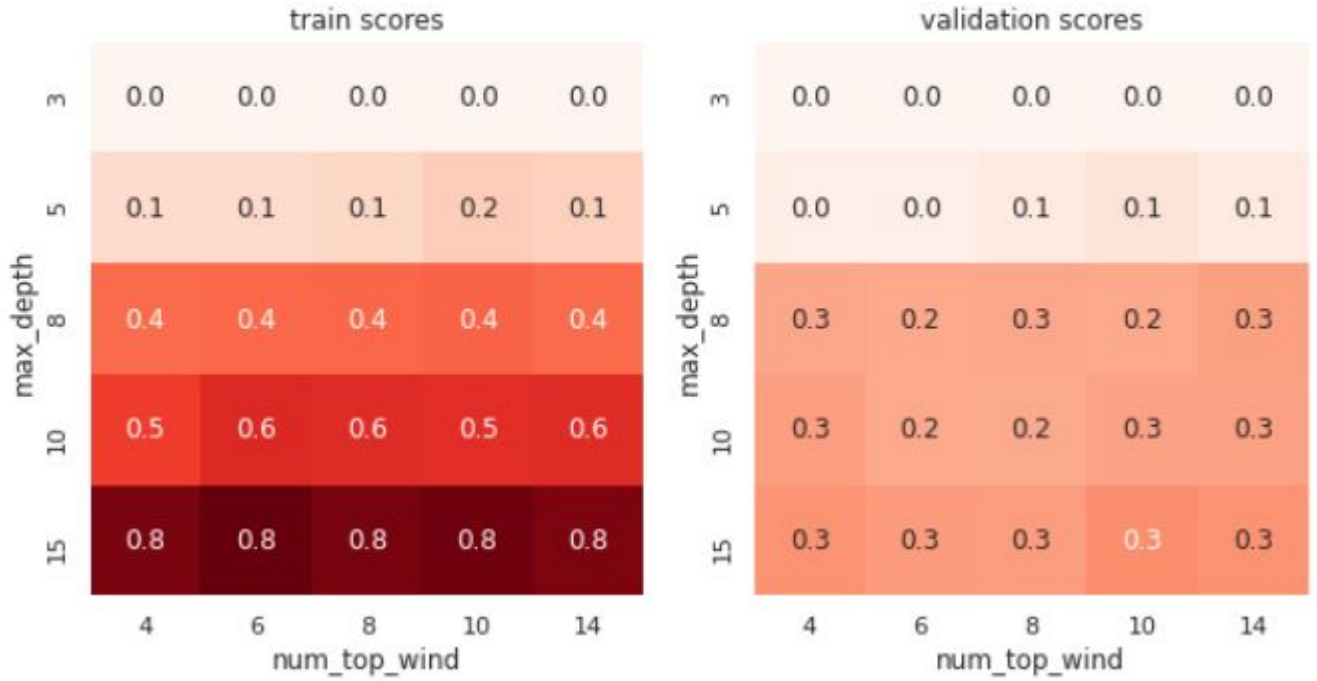
- **Comments:**

  + The model is very overfitting, the score is high on training set but low in on validation.

  + It has the best performance until now.

Random forest

```
Best num_top_wind: 10
Best max_depth: 15
Best validation score: 0.32183908045977005
```



We create a pipeline for random forest, with the max_depth as a parameter, we use it to prevent overfitting..

- **Comment:**

  + The model is overfitting, the error is high on training set but low in on validation

  + It has the best performance.

  + As large max_depth is, as overfitting model is, and as high the f1 score validation is.

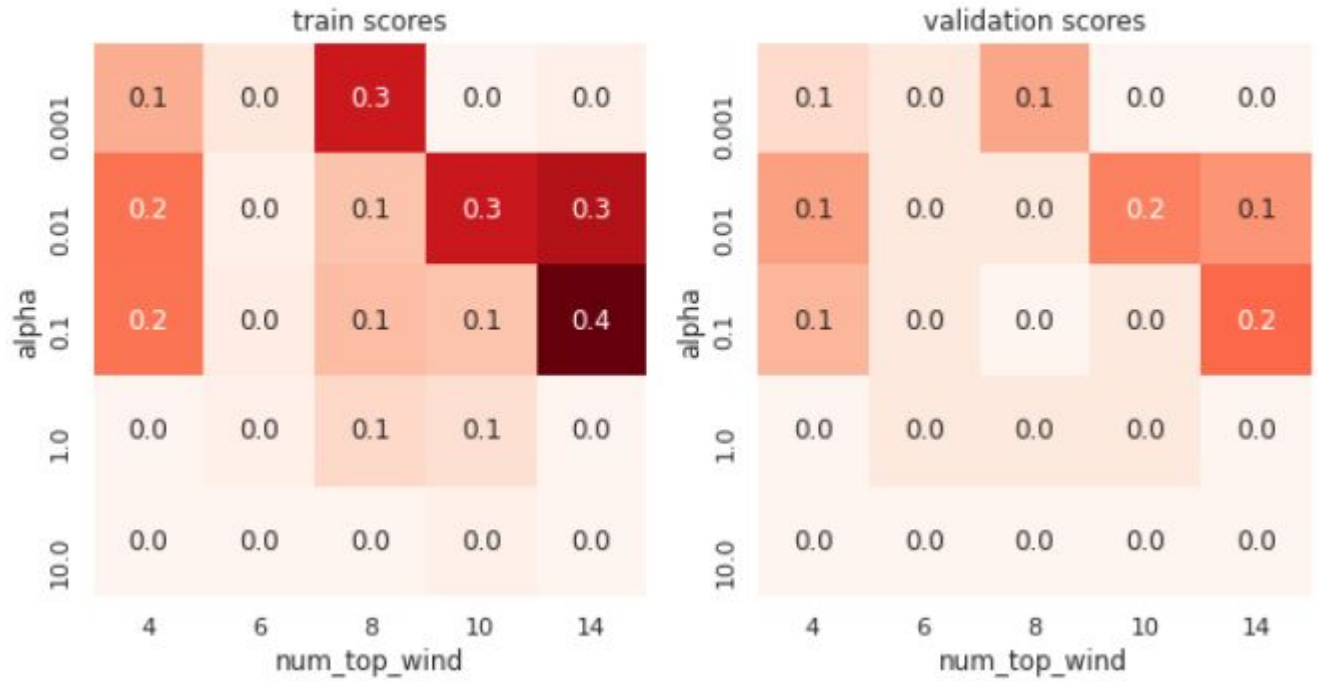
  + Feature wind seems does not affect.

Deep Learning

Multi Layer Perceptron

We create a pipeline for MLP model, with hidden_layer_sizes is 100 units as a parameter.

```
Best num_top_wind: 14
Best alpha: 0.1
Best validation score: 0.19999999999999998
```



- **Comments:**

  + The model is underfitting.

  + To improve, we can add more hidden layers and increase the number of iterations.

TEST SET

After trying many algorithms, the one that gives the best score on validation is the **Decision Tree** algorithm, therefore, we choose this model to evaluate in the test set.

We train it again on both the training and validation sets with the best hyperparameters

	precision	recall	f1-score	support
Others	0.93	0.91	0.92	568
Rain	0.32	0.37	0.34	63
accuracy			0.86	631
macro avg	0.62	0.64	0.63	631
weighted avg	0.87	0.86	0.86	631

Test set result

Observation

- Simple models such as linear models have better results than complex models, like SVM, kNN.
- As more overfitting models (decision tree, random forest), as better result is.

- **Explain**

  + Data is skew => the complex models (SVM, kNN) try to fit the dominated class than the minority ones => The results are low.

  + The more overfitting models (decision tree, random forest) try to fit all data => try to fit 'Rain' class => their results are quite better.

- **Conclusion**

  + The main reason that makes our algorithms is too weak, is the unbalance data and the data is not enough. We have some ideas to fix it:

    - Collect more data
    - Oversample data
    - Use more proper algorithms

Reflection

- Difficulties
- What have you learned?
- What would you do if you have more time?

Difficulties

Lâm Ngọc Phương Anh	**Nguyễn Công Anh Khoa**
- download driver for Selenium	- visualize data
- get the suitable dataset	- choose which columns to keep
	- improve model

What I have learned

Lâm Ngọc Phương Anh	Nguyễn Công Anh Khoa
- use Selenium, API properly to collect data in specific circumstances	- explore data use visualization tools
- visualize, apply statistics into data	- evaluate data and implement sklearn algorithm
- clean and filter data	
- choose suitable model in specific circumstances	

ANH KHOA

Before this course, I always focus on mathematics and the theory of machine learning and forget to hone programming skills. Now I realize that the gap between theory and reality is huge, and I rethink that the main purpose of data science is to apply to real world. After this project, as well as this course, I know that I need to balance between the theory and programming techniques. So that, I can achieve competencies for data science.

PHUONG ANH

Data science is a fancy field. Through this subject, I have a more deep insight about the values that data science applications and products bring to the world. Furthermore, I also learn lots of new skills like crawling and analyzing data then applying Machine Learning model to solve real-world problems.

What would you do if you have more time?

- Obviously our data is poor, so the simple way to improve is collecting more data.

- Weather is time series data, for example, if you know that after a storm, its probability is sunning in a few next days. To handle sequence data, RNNs network now is the best method.

- Weather is changing. For example, this year is hotter than previous years. Therefore, we want to build a system that can learn online for updating the knowledge of model.

Thanks for your listening and watching

References

Pandas, DataFrame

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

Pandas, Series

https://pandas.pydata.org/docs/reference/api/pandas.Series.html

Format of file from HW3 - Introduction to Data Science