# AML4Design - Tutorial 3

TU Delft

# Task 1 - Smell Pittsburgh

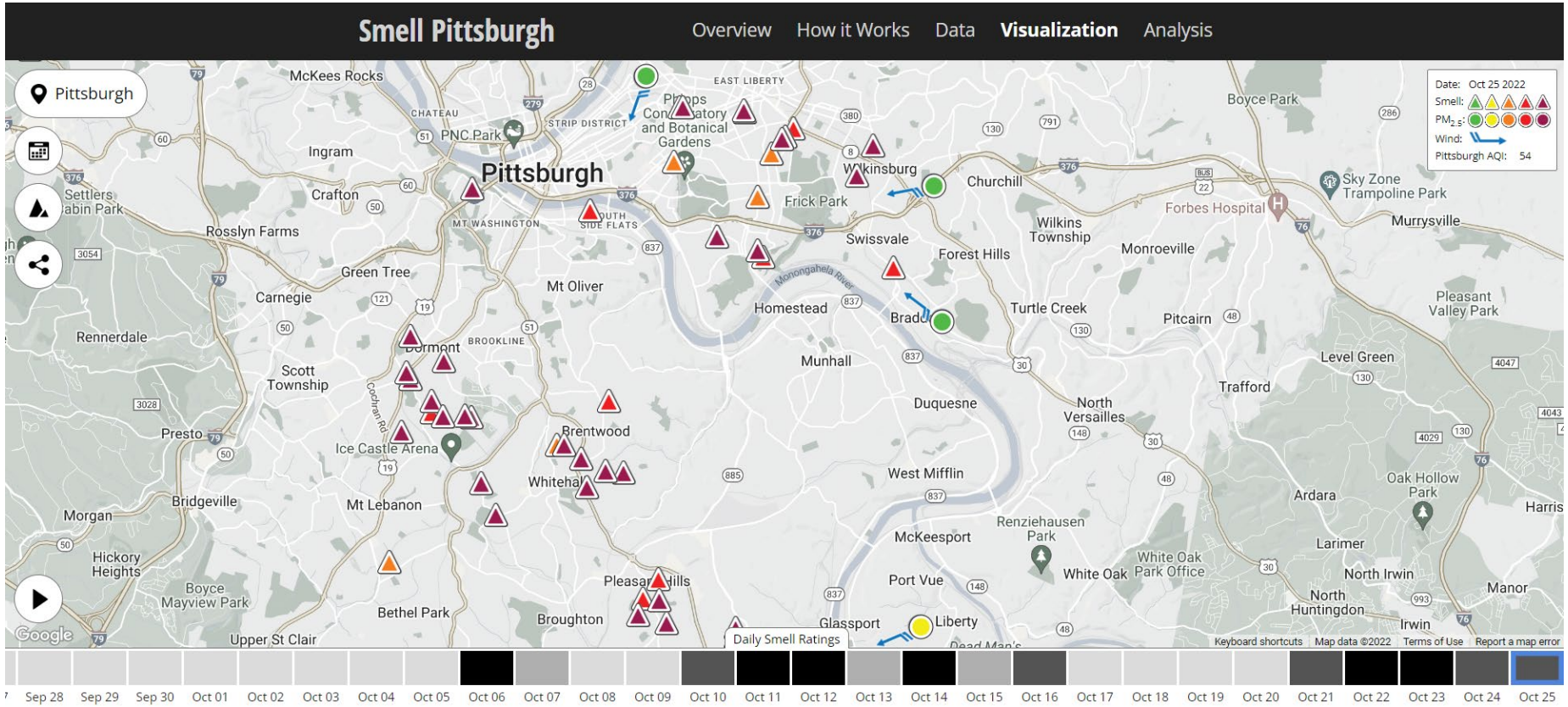
Please download and read paper “Yen-Chia Hsu, Jennifer Cross, Paul Dille, Michael Tasota, Beatrice Dias, Randy Sargent, Ting-Hao (Kenneth) Huang, and Illah Nourbakhsh. 2020. Smell Pittsburgh: Engaging Community Citizen Science for Air Quality. ACM Transactions on Interactive Intelligent Systems. 10, 4, Article 32”

- https://aml4design.github.io/tutorials/structured-data-module/preparation/#task-1

TU Delft

# Task 2 – Interacting with Data Visualization

- https://smellpgh.org/visualization




TU Delft

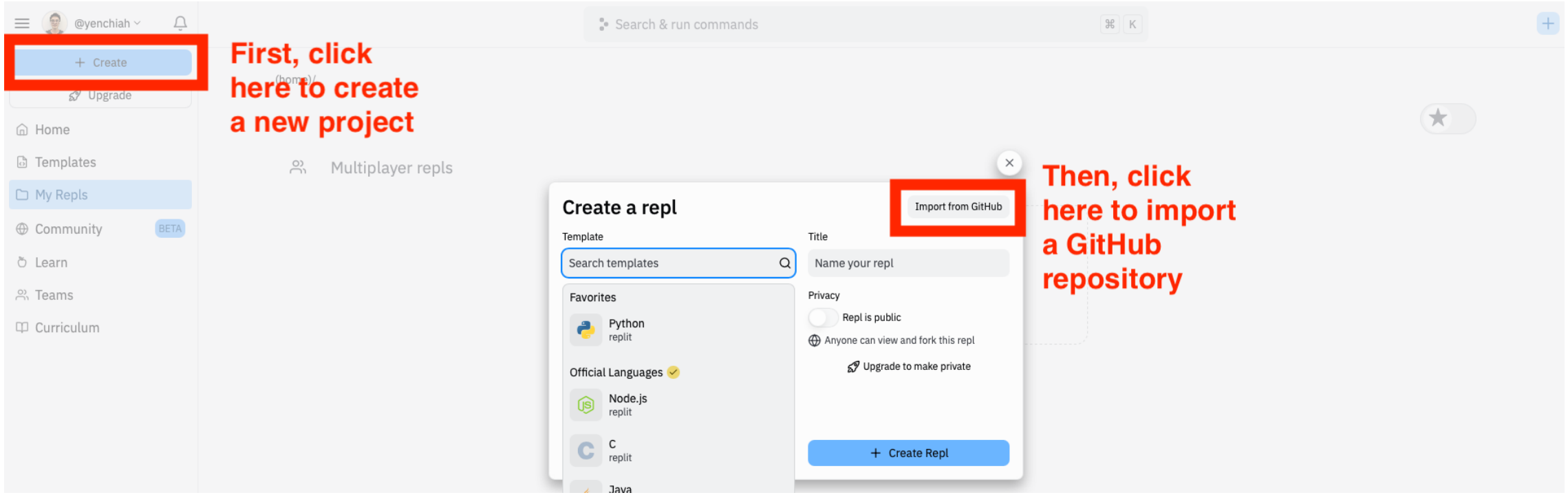
# Task 3 – Interpreting Data

- Data analysis page: https://smellpgh.org/analysis
  - Are there any characteristics about the distribution of smell reports over **time** and **geographical regions**?
  - What are the common **descriptions** of bad **odors** that people reported?
  - What are the possible **predictors** (e.g., chemical compounds, weather data) of bad smell in the Pittsburgh region?

TU Delft

# Preparation – Importing Project to Replit

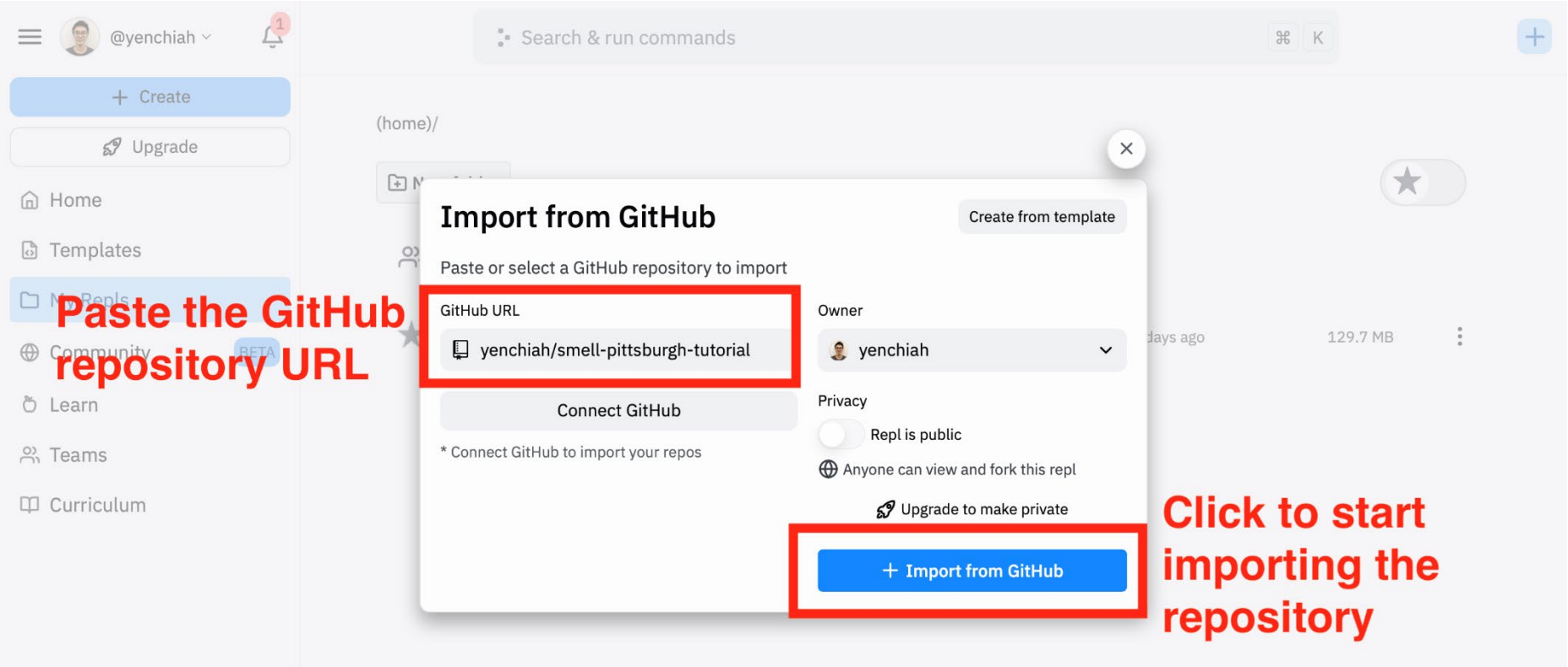
- Go to Replit: https://replit.com/




TU Delft

# Preparation – Importing Project to Replit

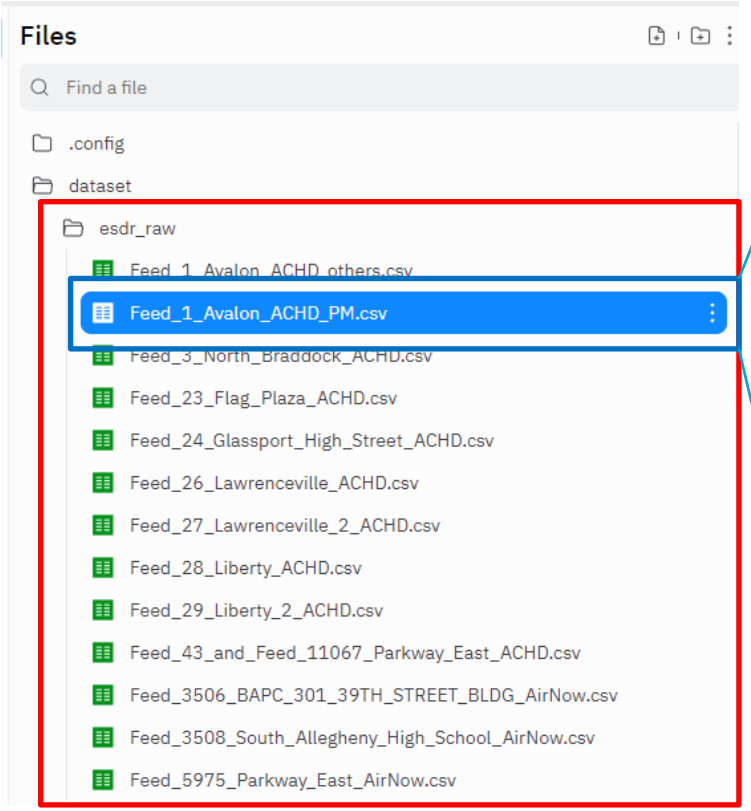
- Link to the GitHub repository: https://github.com/aml4design/smell-pittsburgh-tutorial

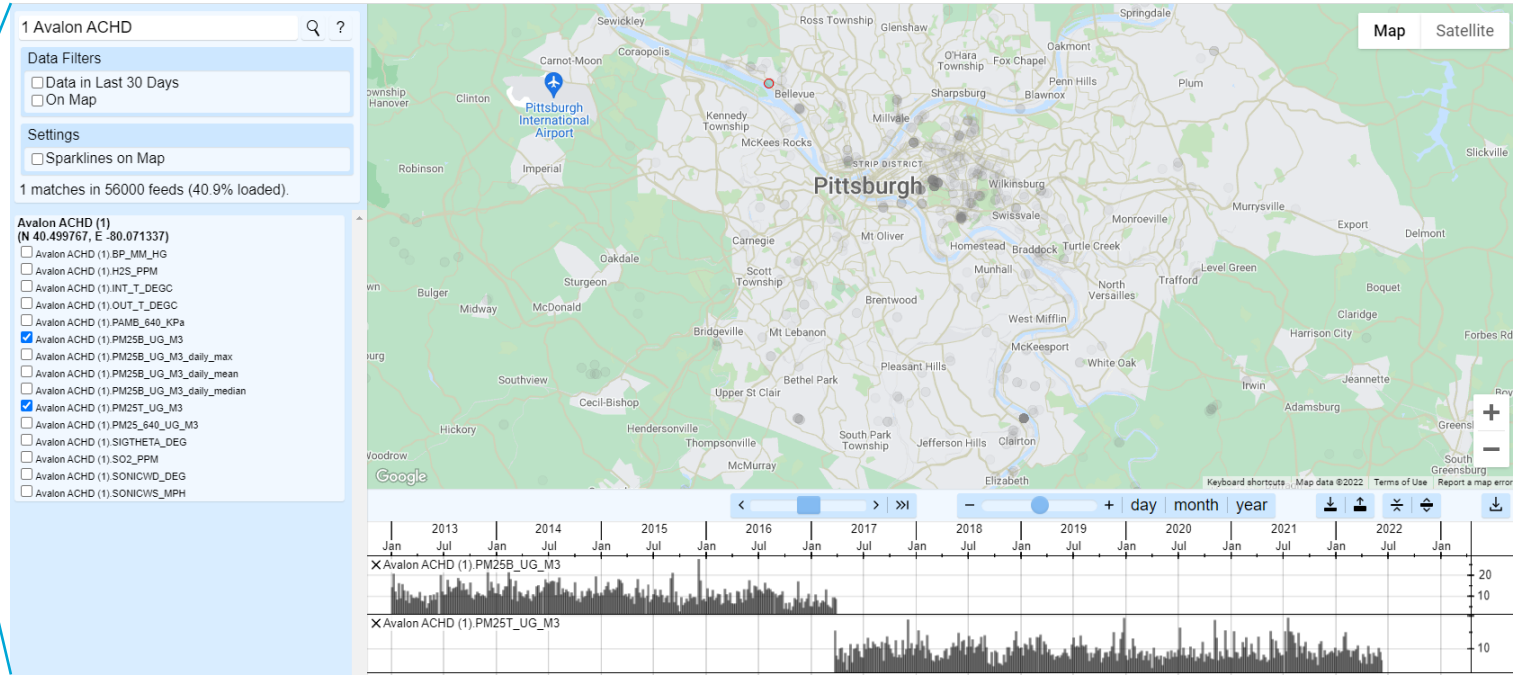



TU Delft

# Preparation – Explanation (Data)

- Data (historical)




TU Delft

# Preparation – Explanation (Data)

- PM25_UG_M3: fine particulate matter (PM2.5) in micrograms per cubic meter
- PM25T_UG_M3: same as PM25_UG_M3
- PM25B_UG_M3: same as PM25_UG_M3

dataset/esdr_raw/Feed_1_Avalon_ACHD_PM.csv

```
EpochTime,3.feed_1.PM25B_UG_M3..3.feed_1.PM25T_UG_M3
1477891800,6
1477895400,4
1477899000,3
1477902600,2
1477906200,3
1477909800,3
1477913400,1
1477917000,3
1477920600,4
1477924200,3
1477927800,1
1477931400,0
1477935000,4
1477938600,6
1477942200,4
1477945800,4
1477949400,2
1477953000,0
1477956600,-2
1477960200,6
1477963800,20
1477967400,16
1477971000,8
1477974600,6
```

TU Delft

# Preparation – Explanation (Smell Event)

- What is the definition of smell events:
  - We define smell events through two parameters: smell_thr and smell_predict_hrs

smell_thr:

Parameter "smell_thr" is the **threshold** to define a smell event. If the sum of smell ratings is larger than this threshold, the model will think that there will be a smell event.

Example:

40: this may mean that there are 10 people, and each of them reported smell rating 4.

TU Delft

# Preparation – Explanation (Smell Event)

- What is the definition of smell events:
  - We define smell events through two parameters: smell_thr and smell_predict_hrs

smell_predict_hrs:

Parameter "smell_predict_hrs" is **the number of future hours** to predict smell events.

Example:

8: means to predict the smell event in the future 8 hours.

If it is 12:00 now, the model predicts if smell events will happen between 12:00 and 20:00.

TU Delft

# Preparation – Explanation (Smell Event)

- Let’s combine both smell_thr and smell_predict_hrs

Example: smell_thr = 40 and smell_predict_hrs = 8

We want to know if the sum of the smell ratings will be greater than 40 in the following eight hours or not.

- Keep in mind that the prediction is Boolean (**yes or no**).

TU Delft

# Preparation – Explanation (Predicted Result)

- True positives
  - There is a smell event in the real world, and the model correctly predicts that there is a smell event.
- False positives
  - There is no smell event in the real world, but the model falsely predicts that there is a smell event.
- True negatives
  - There is no smell event in the real world, and the model correctly predicts that there is no smell event.
- False negatives
  - There is a smell event in the real world, but the model falsely predicts that there is no smell event.

| | | Actual class | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Predicted class | Positive | **TP:** True Positive | **FP:** False Positive (Type I Error) | **Precision:** TP / (TP + FP) |
| | Negative | **FN:** False Negative (Type II Error) | **TN:** True Negative | **Negative Predictive Value:** TN / (TN+FN) |
| | | **Recall or Sensitivity:** TP / (TP + FN) | **Specificity:** TN / (TN + FP) | **Accuracy:** TP + TN / (TP + TN + FP + FN) |

TU Delft

# Preparation – Explanation (Predicted Result)

- True positives
  - There is a smell event in the real world, and the model correctly predicts that there is a smell event.
- False positives
  - There is no smell event in the real world, but the model falsely predicts that there is a smell event.
- True negatives
  - There is no smell event in the real world, and the model correctly predicts that there is no smell event.
- False negatives
  - There is a smell event in the real world, but the model falsely predicts that there is no smell event.

| | | Actual class | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Predicted class | Positive | **TP:** True Positive | **FP:** False Positive (Type I Error) | **Precision:** TP / (TP + FP) |
| | Negative | **FN:** False Negative (Type II Error) | **TN:** True Negative | **Negative Predictive Value:** TN / (TN+FN) |
| | | **Recall or Sensitivity:** TP / (TP + FN) | **Specificity:** TN / (TN + FP) | **Accuracy:** TP + TN / (TP + TN + FP + FN) |

TU Delft

# Preparation – Explanation (Predicted Result)

- True positives
  - There is a smell event in the real world, and the model correctly predicts that there is a smell event.
- False positives
  - There is no smell event in the real world, but the model falsely predicts that there is a smell event.
- True negatives
  - There is no smell event in the real world, and the model correctly predicts that there is no smell event.
- False negatives
  - There is a smell event in the real world, but the model falsely predicts that there is no smell event.

| | | Actual class | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Predicted class | Positive | **TP:** True Positive | **FP:** False Positive (Type I Error) | **Precision:** TP / (TP + FP) |
| | Negative | **FN:** False Negative (Type II Error) | **TN:** True Negative | **Negative Predictive Value:** TN / (TN+FN) |
| | | **Recall or Sensitivity:** TP / (TP + FN) | **Specificity:** TN / (TN + FP) | **Accuracy:** TP + TN / (TP + TN + FP + FN) |

TU Delft

# Preparation – Explanation (Predicted Result)

- True positives
  - There is a smell event in the real world, and the model correctly predicts that there is a smell event.
- False positives
  - There is no smell event in the real world, but the model falsely predicts that there is a smell event.
- True negatives
  - There is no smell event in the real world, and the model correctly predicts that there is no smell event.
- False negatives
  - There is a smell event in the real world, but the model falsely predicts that there is no smell event.

| | | Actual class | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Predicted class | Positive | **TP:** True Positive | **FP:** False Positive (Type I Error) | **Precision:** TP / (TP + FP) |
| | Negative | **FN:** False Negative (Type II Error) | **TN:** True Negative | **Negative Predictive Value:** TN / (TN+FN) |
| | | **Recall or Sensitivity:** TP / (TP + FN) | **Specificity:** TN / (TN + FP) | **Accuracy:** TP + TN / (TP + TN + FP + FN) |

TU Delft

# Preparation – Explanation (Predicted Result)

**Accuracy** is the number of correct predictions divided by the total number of data points.

It is a good metric if the data distribution is not skewed (i.e., the number of data records that have a bad smell and do not have a bad smell is roughly equal).

| | | Actual class | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Predicted class | Positive | **TP:** True Positive | **FP:** False Positive (Type I Error) | **Precision:** TP / (TP + FP) |
| | Negative | **FN:** False Negative (Type II Error) | **TN:** True Negative | **Negative Predictive Value:** TN / (TN + FN) |
| | | **Recall or Sensitivity:** TP / (TP + FN) | **Specificity:** TN / (TN + FP) | **Accuracy:** TP + TN / (TP + TN + FP + FN) |

TU Delft

# Preparation – Explanation (Predicted Result)

**Precision** means how precise the prediction is.

High precision means that if the model predicts “yes” for smell events, it is highly likely that the prediction is correct. We want high precision because we want the model to be as precise as possible when it says there will be smell events.

| | | Actual class | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Predicted class | Positive | **TP:** True Positive | **FP:** False Positive (Type I Error) | **Precision:** TP / (TP + FP) |
| | Negative | **FN:** False Negative (Type II Error) | **TN:** True Negative | **Negative Predictive Value:** TN / (TN+FN) |
| | | **Recall or Sensitivity:** TP / (TP + FN) | **Specificity:** TN / (TN + FP) | **Accuracy:** TP + TN / (TP + TN + FP + FN) |

TU Delft

# Preparation – Explanation (Predicted Result)

**Recall** means the ability of the model to catch events.

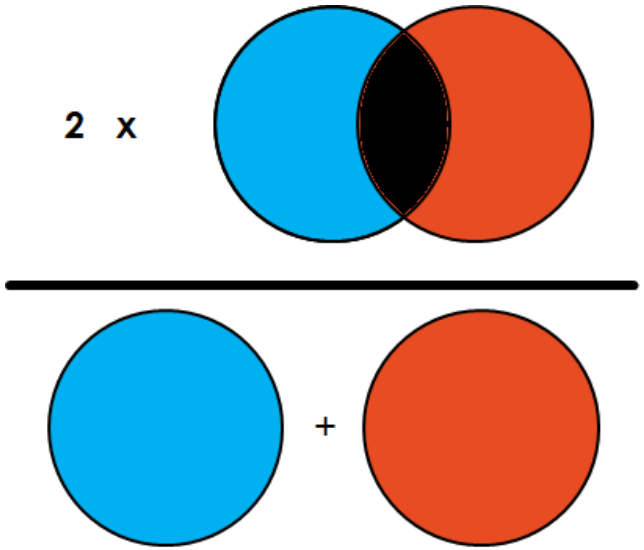
High recall means that the model has a low chance to miss the events that happen in the real world. We want high recall because we want the model to catch all smell events without missing them.

| | | Actual class | | |
|---|---|---|---|---|
| | | Positive | Negative | |
| Predicted class | Positive | **TP:** True Positive | **FP:** False Positive (Type I Error) | **Precision:** TP / (TP + FP) |
| | Negative | **FN:** False Negative (Type II Error) | **TN:** True Negative | **Negative Predictive Value:** TN / (TN+FN) |
| | | **Recall or Sensitivity:** TP / (TP + FN) | **Specificity:** TN / (TN + FP) | **Accuracy:** TP + TN / (TP + TN + FP + FN) |

TU Delft

# Preparation – Explanation (Predicted Result)

$$F1\ Score = 2 \times \frac{recall \times precision}{recall + precision}$$




**F1 Score** is the weighted average of Precision and Recall.

Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution.

TU Delft

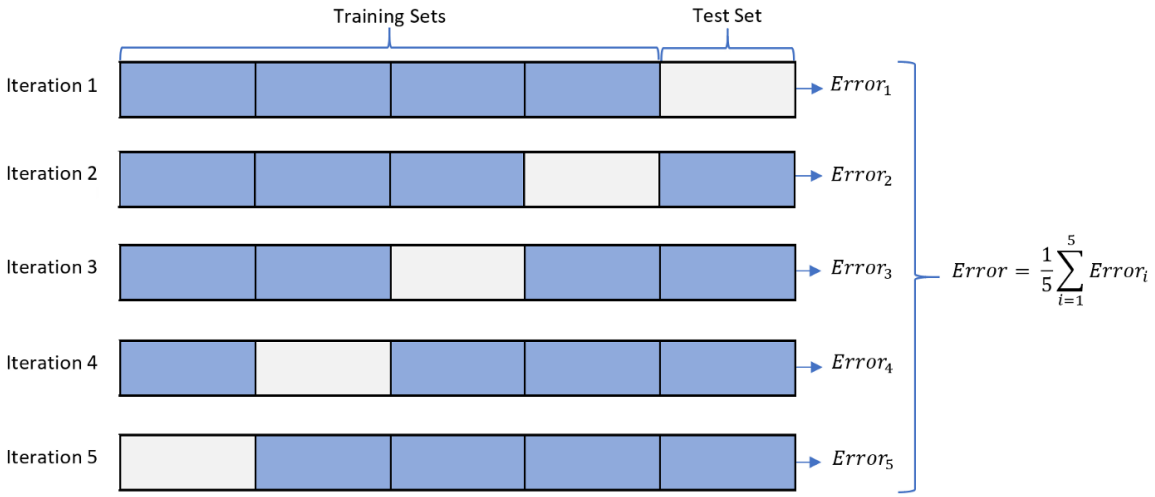
# Preparation – Explaination (K-fold Cross-Validation)

**Cross-Validation** is a technique used to test a model's ability to predict unseen data, data not used to train the model.

- **K-fold Cross-Validation**

**Steps:**

1. Split training data into K equal parts
2. Fit the model on k-1 parts and calculate test error using the fitted model on the kth part
3. Repeat k times, using each data subset as the test set once. (usually k= 5~20)




TU Delft

# Preparation – Explanation (Model)

- Let’s try to predict results through a machine learning model based on sensor data.

Example:

model = DummyClassifier(strategy="constant", constant=0)

Where “constant”: always predicts a constant label that is provided by the user. This is useful for metrics that evaluate a non-majority class.

This shows the result of the model, which is the dummy classifier that always **predicts “no” for the smell events**.

TU Delft

# Preparation – Interpreting results

Use model DummyClassifier(constant=0, strategy='constant')

Perform cross-validation, please wait...

==========

average f1-score: 0.0

average precision: 0.0

average recall: 0.0

average accuracy: 0.92

number of true positives: 0

number of false positives: 0

number of true negatives: 14914

number of false negatives: 1382

==========

What the number tells us?

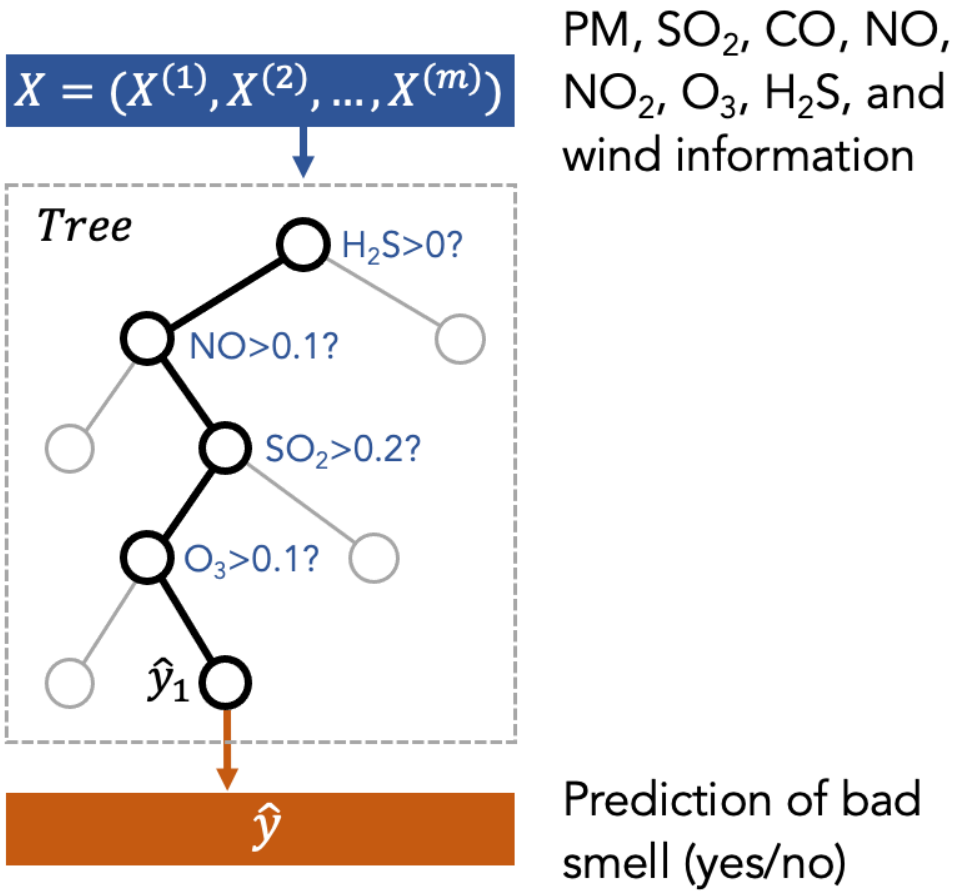
TU Delft

# Task 4 – Changing Models

```
model = DummyClassifier(strategy="constant", constant=0)
```

Replace the line above with the following code:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```




TU Delft

# Task 4 – Changing Models (Result)

Use model DecisionTreeClassifier()

Perform cross-validation, please wait...

==========

average f1-score: 0.13

average precision: 0.19

average recall: 0.16

average accuracy: 0.86

number of true positives: 325

number of false positives: 1295

number of true negatives: 13619

number of false negatives: 1057

==========

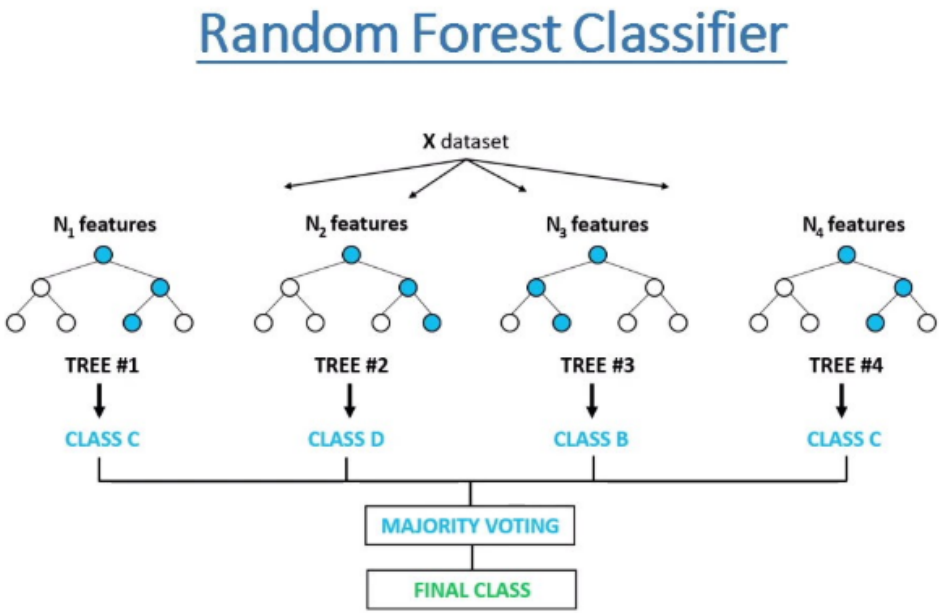
TU Delft

# Task 5 – Improving Model

```
model = DecisionTreeClassifier()
```

Replace the line above with the following code:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
```




TU Delft

# Task 5 – Improving Model (Result)

Use model RandomForestClassifier()

Perform cross-validation, please wait...

==========

average f1-score: 0.11

average precision: 0.19

average recall: 0.1

average accuracy: 0.9

number of true positives: 212

number of false positives: 529

number of true negatives: 14385

number of false negatives: 1170

==========

TU Delft

# Task 5 – Change Parameters

- Firstly, do we really believe that we are using a **good set of features**? Is it sufficient to only use the H2S (hydrogen sulfide) variable? Is it sufficient to only include the data from the current time and the previous hour?
- Secondly, the machine learning pipeline uses 14 days of data in the past to predict smell events in the future 7 days. Do we believe that 14 days are **sufficient for training a good model**?

TU Delft

# Task 5 – Enrich Features (sensor data)

Add wanted sensor value to dataframe:

```
wanted_cols = ["DateTime",
"3.feed_28.H2S_PPM", "3.feed_28.SO2_PPM"]
```

Use model RandomForestClassifier()

Perform cross-validation, please wait...

==========

average f1-score: 0.13

average precision: 0.22

average recall: 0.11

average accuracy: 0.9

number of true positives: 218

number of false positives: 422

number of true negatives: 14492

number of false negatives: 1164

==========

TU Delft

# Task 5 – Enrich Features (historical data)

Change look back hours:

look_back_hrs = 2

Use model RandomForestClassifier()

Perform cross-validation, please wait...

==========

average f1-score: 0.12

average precision: 0.2

average recall: 0.12

average accuracy: 0.89

number of true positives: 221

number of false positives: 583

number of true negatives: 14332

number of false negatives: 1160

==========

TU Delft

# Task 5 – Other Parameters

There are many other parameters you could play with.

For example,

- The threshold that we used to define a bad smell event is 40, which is the sum of smell ratings that people reported within the time range that we want to predict (in our case, it is 8 hours). Do we believe that **40 is a good number to determine the presence of a bad smell**?

TU Delft

# Task 5 – Feature Importance

This part prints feature importance, which indicates the **influence of each feature** on the model prediction result.

Here we use the Random Forest model.

Computer feature importance using RandomForestClassifier(random_state=0)

==========

Display feature importance based on f1-score

| | feature_importance | feature_name |
|---|---|---|
| 0 | 0.63188 | 3.feed_28.H2S_PPM |
| 1 | 0.59183 | HourOfDay |
| 2 | 0.52891 | Day |
| 3 | 0.50336 | 3.feed_28.H2S_PPM_1h |
| 4 | 0.47280 | DayOfWeek |

Column names below:

['feature_importance', 'feature_name']

==========

TU Delft

Thank you!

TU Delft