Center for Building Infrastructure and Public Space (CBIPS)

# Water Sampling Data Science Team

## Team

Haonan Yang

Yi Liang

Siqi Fan

Felipe Paniagua

COLUMBIA | CBIPS
Center for Buildings, Infrastructure and Public Space

Mentor: Charles Shen
Feniosky Peña-Mora
Fredric M. Bell

# Overview of Our Research Project

## *01* Background

- Water Sampling
- Water Monitoring

## *02* Machine Learning Based

- How can ML help?
- Research & Case Study

## *03* Achievement

- Conversation with DEP
- Our Models

## *04* Summary

- Conclusions
- Future Work

# *01* Background:

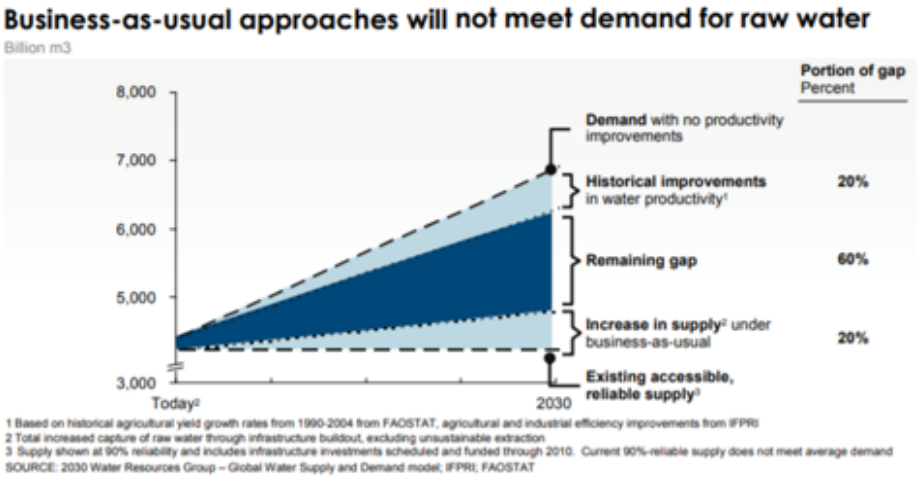
Why Is Water
Sampling & Monitoring
So IMPORTANT?

# Background

The linkage between water and the economy is so compelling that decisions about water are rarely deferred. Decisions that are uninformed almost always have unintended consequences, with impacts on the environment, health, and society.

- **Water challenge is tied to population growth, emerging developing countries.**
- **The water sector is severely under-funded, with especially serious financing gaps in developing countries.**

## Supply - Demand Issue




(Source:https://www.mckinsey.com/~/media/mckinsey/dotcom/client_service/sustainability/pdfs/charting%20our%20water%20future/charting_our_water_future_full_report_.ashx)

# Drinking Water Sampling


- DEP continuously sample, and conduct analyses for numerous water quality parameters, including **microbiological, chemical, and physical measures**, throughout the watershed and as the water enters the distribution system.
- DEP also regularly tests water quality at nearly **1,000** water quality sampling **stations** throughout NYC.
- In 2019, DEP performed approximately 456,500 analyses on 36,300 samples from the distribution system, **meeting all state and federal monitoring requirements.** DEP also performed approximately 262,500 analyses on 15,000 samples from the reservoirs and their watersheds.



(Source:https://www1.nyc.gov/assets/dep/downloads/pdf/water/drinking-water/drinking-water-supply-quality-report/2019-drinking-water-supply-quality-report.pdf)

# Drinking Water Monitoring


- DEP monitors the water in the **distribution system, upstate reservoirs and feeder streams, and wells** that are sources for New York City's drinking water supply.

- A growing network of **robotic monitoring** stations provided another 2 million water quality measurements last year, allowing DEP to optimize its operation of the reservoirs, support watershed protection efforts, and study water quality trends.



(Source:https://www1.nyc.gov/site/dep/water/groundwater-supply-system.page)

# Issues with Drinking Water @ NYC

ASCE New York Infrastructure Grade
(2015 the most recent)

Drinking Water: C




(Source:https://www.infrastructurereportcard.org/state-item/new-york/
Picture:https://www.smartcitiesdive.com/news/how-ai-and-data-turn-city-water-management-from-an-art-to-a-science/559424/)

# 02 Machine Learning Based

- How can Machine learning help?

# How can machine learning help?

Machine learning can automate, simplify and improve many aspects of water monitoring including:

- Detect and correct equipment malfunctions
- Detect environmental anomalies
- Predict the effects of policy decisions
- Improve modeling and analysis
- Automate and control allocation and distribution



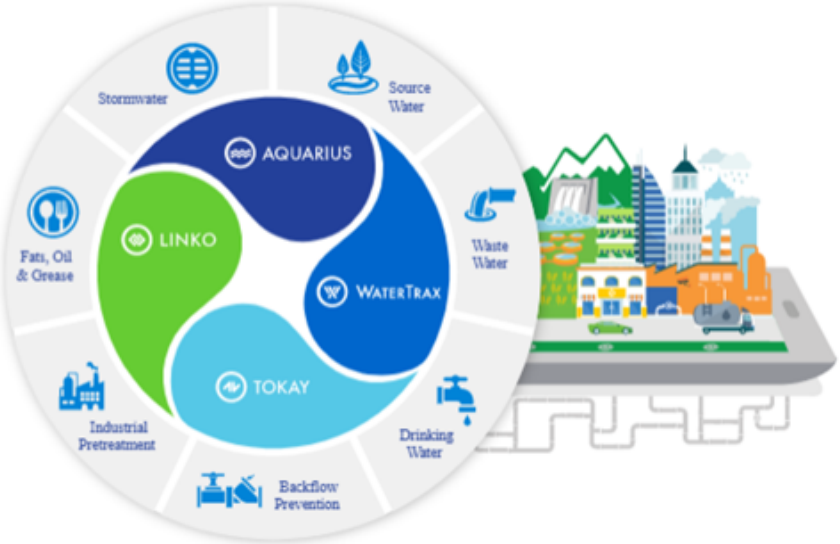
(Source:Aquatic Informatics Workshop-Machine learning for water monitoring, hydrology and sustainability Kevin Swersky: http://www.cs.toronto.edu/~kswersky/wp-content/uploads/WorkshopPresentation.pdf)

# 02 Machine Learning Based

- Research & Case Study

# 1. Aquatic Informatics: software solutions provider

AQUATIC Informatics




## One Water One Platform:

Water Data Management for the entire water cycle based on four software.

- AQUARIUS: Analytics Software.
- WaterTrax: Water Quality Compliance Software.
- Linko: Regulatory Compliance Software.
- Tokay: Backflow Prevention Software.

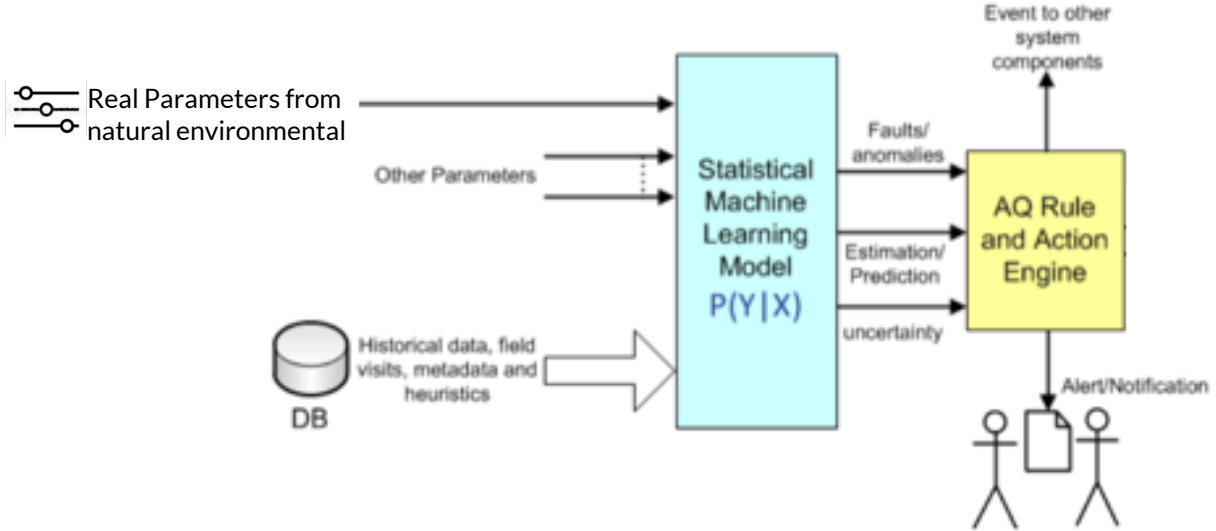
- Automate workflows
- Correct errors
- Meet regulations
- Predict problems
- Enforce compliance
- Alert stakeholders



(Source: https://aquaticinformatics.com/)

# 1. Aquatic Informatics: software solutions provider

AQUATIC Informatics




Reasons why we choose:

- ➔ Clear to see their Data Data Processing Pipeline.
- ➔ Provide us knowledge that how ML algorithms applied in applications in an established software development company.

(Source: https://aquaticinformatics.com/products/aquarius/aquarius-forecast/)

# 2. ML In Predicting Water Quality

pH values in drinking water samples

↓

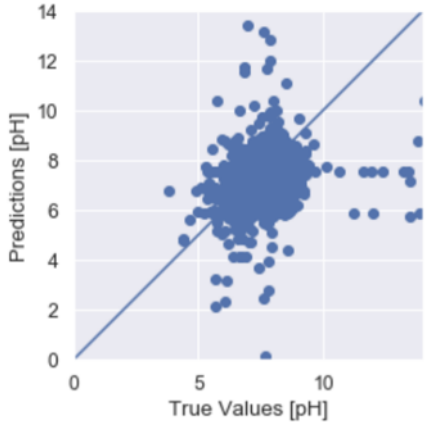
16,114 measurements, 261 missing data

↓

Machine Learning Model

↓

Successful predictions of pH values



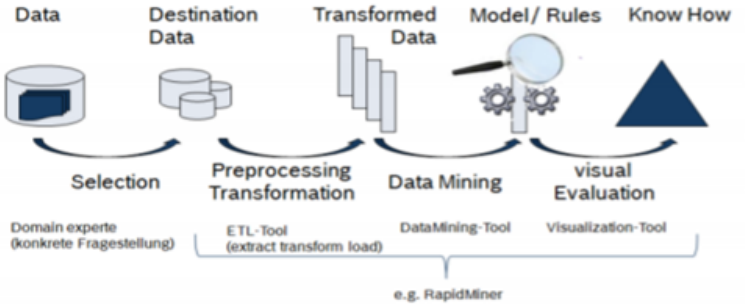

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Location II | Years | Measurement Date | Measurement Time | Casing Volume | Dissolved Oxygen | Flow (in gpm) | Oxidation pH | | Specific Conductance | Temperature | Turbidity |
| 2 | 03-B-10 | 2006 | Jun | 11:00 | | 34.2 | | 175.1 | 5.78 | 0.286 | 13.64 | 46 |
| 3 | 03-B-10 | 2006 | Aug | 10:15 | | | | 270.6 | 6.29 | 366 | 15.2 | 153.9 |
| 4 | 03-B-10 | 2006 | Dec | 10:53 | | 2.7 | | 455.7 | 5.98 | 280 | 13.5 | 44.8 |
| 5 | 03-B-10 | 2007 | Mar | 10:51 | | 0.63 | | 346 | 6.12 | 1493 | 12.4 | 15.8 |
| 6 | 03-B-10 | 2007 | Jul | 11:05 | | 0.81 | | 302 | 6.16 | 577 | 14.6 | 41.6 |
| 7 | 03-B-10 | 2007 | Sep | 8:15 | | 3.41 | | 253 | 5.72 | 424 | 14.7 | 77.1 |
| 8 | 03-B-10 | 2007 | Dec | 15:10 | | 2.42 | 0.184 | 390 | 5.75 | 2.14 | 12.8 | 8.03 |
| 9 | 03-B-10 | 2008 | Mar | 10:40 | | 1 | 0.18 | 221 | 5.75 | 1623 | 12.3 | 33.2 |
| 10 | 03-B-10 | 2008 | Jun | 10:35 | | 0.81 | 0.16 | 215 | 6.11 | 1210 | 14 | 70.8 |
| 11 | 03-B-10 | 2008 | Sep | 11:55 | | 0.1 | 0.125 | 14 | 6.2 | 726 | 13.9 | 78 |
| 12 | 03-B-10 | 2008 | Dec | 13:05 | | 5.15 | 0.13 | 278.4 | 5.98 | 384 | 12.85 | 65.4 |
| 13 | 03-B-10 | 2009 | Mar | 12:30 | | 2.28 | 0.12 | 275 | 5.99 | 623 | 13.21 | 50.4 |
| 14 | 03-B-10 | 2009 | Jun | 16:00 | | 1.27 | 0.13 | 196.8 | 5.96 | 492 | 13.09 | 1000 |
| 15 | 03-B-10 | 2009 | Sep | 11:00 | | 3.68 | 0.69 | 166.4 | 6.1 | 268 | 13.78 | |
| 16 | 03-B-13 | 2006 | Jun | 9:50 | | 30.8 | | 151.1 | 5.77 | 260 | 14.29 | 73.9 |
| 17 | 03-B-13 | 2006 | Aug | 11:50 | | 1.2 | | 225.4 | 6.22 | 328 | 15.8 | 122 |
| 18 | 03-B-13 | 2006 | Dec | 12:37 | | 1 | | 448.1 | 5.91 | 356 | 13.3 | 27.2 |
| 19 | 03-B-13 | 2007 | Mar | 13:00 | | 1.09 | | 275 | 6.05 | 1428 | 11.9 | 10.5 |

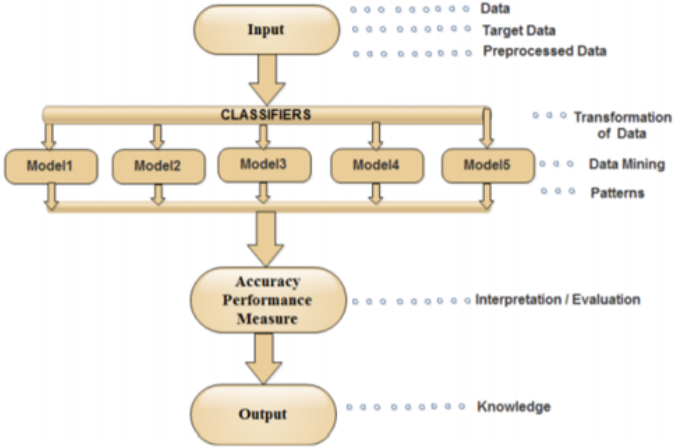
(Source: https://locustec.com/applications/industry/water-utilities/)

# 3. Pattern Extraction In Water Quality Prediction

**Scholar Article** ➔ Aims to implore data mining technique for pattern extraction and model prediction of water quality in water reservoir using different parameters and water quality index.

## Methodology




Jefferson Lerios - Technological Institute of the Philippines
Mia Villarica - Laguna State Polytechnic University

International Journal of Mechanical Engineering and Robotics Research -Nov. 2019

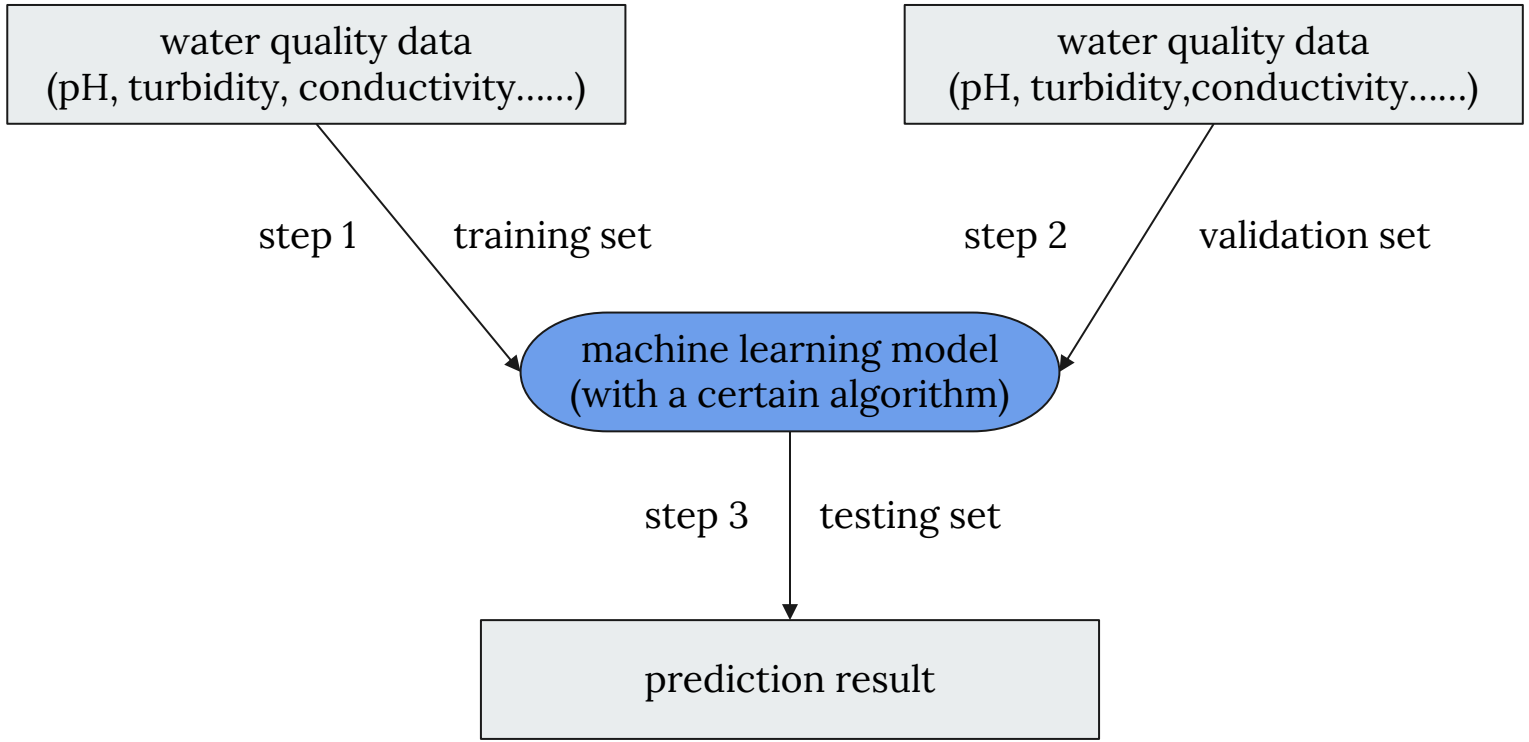


(Source: http://www.ijmerr.com/uploadfile/2019/0709/20190709114518923.pdf)

# *03* Achievement

- What we are aiming to achieve? By how?
  - Our Conversation with DEP
  - Model Explanation

# What we are aiming to do?



# Data Source

## GECCO 2019 Industrial Challenge: Monitoring of drinking-water quality

Thüringer Fernwasserversorgung
Mehr als reines Wasser

32K testing data
100K training data

7 variables

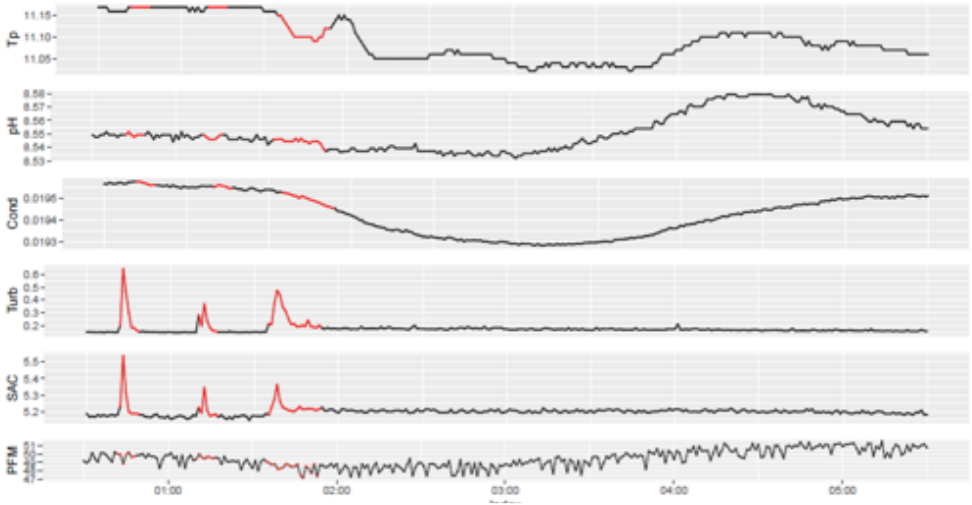
| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | Time | Tp | pH | Cond | Turb | SAC | PFM | Event |
| 2 | 0 | 2017-07-0 | 6.94 | 8.60774 | 0.020953 | 0.125931 | 3.58683 | 43.7559 | FALSE |
| 3 | 1 | 2017-07-0 | 6.93 | 8.60589 | 0.020965 | 0.127219 | 3.59025 | 43.4366 | FALSE |
| 4 | 2 | 2017-07-0 | 6.94 | 8.6022 | 0.020968 | 0.126482 | 3.58318 | 43.5994 | FALSE |
| 5 | 3 | 2017-07-0 | 6.94 | 8.6022 | 0.020971 | 0.126184 | 3.58769 | 43.3704 | FALSE |
| 6 | 4 | 2017-07-0 | 6.94 | 8.60405 | 0.020973 | 0.127908 | 3.58287 | 43.1656 | FALSE |
| 7 | 5 | 2017-07-0 | 6.94 | 8.60405 | 0.020980 | 0.126111 | 3.5878 | 43.4366 | FALSE |
| 8 | 6 | 2017-07-0 | 6.95 | 8.60405 | 0.020984 | 0.126327 | 3.58538 | 43.319 | FALSE |
| 9 | 7 | 2017-07-0 | 6.95 | 8.6022 | 0.020984 | 0.125598 | 3.59004 | 43.1656 | FALSE |
| 10 | 8 | 2017-07-0 | 6.95 | 8.6022 | 0.020998 | 0.126622 | 3.59189 | 42.5627 | FALSE |
| 11 | 9 | 2017-07-0 | 6.94 | 8.6022 | 0.020991 | 0.126646 | 3.58851 | 43.2824 | FALSE |
| 12 | 10 | 2017-07-0 | 6.95 | 8.60405 | 0.021001 | 0.12672 | 3.59556 | 43.5401 | FALSE |
| 13 | 11 | 2017-07-0 | 6.95 | 8.6022 | 0.021014 | 0.126704 | 3.59434 | 42.6052 | FALSE |
| 14 | 12 | 2017-07-0 | 6.95 | 8.60405 | 0.021016 | 0.1259 | 3.60614 | 43.3998 | FALSE |
| 15 | 13 | 2017-07-0 | 6.95 | 8.60589 | 0.021024 | 0.126167 | 3.59635 | 42.4146 | FALSE |
| 16 | 14 | 2017-07-0 | 6.96 | 8.60589 | 0.021030 | 0.127636 | 3.585 | 42.2745 | FALSE |
| 17 | 15 | 2017-07-0 | 6.96 | 8.60405 | 0.021026 | 0.124759 | 3.58992 | 43.8985 | FALSE |
| 18 | 16 | 2017-07-0 | 6.96 | 8.60589 | 0.021030 | 0.126349 | 3.58556 | 42.2745 | FALSE |
| 19 | 17 | 2017-07-0 | 6.96 | 8.60589 | 0.021028 | 0.12642 | 3.57833 | 43.7185 | FALSE |
| 20 | 18 | 2017-07-0 | 6.96 | 8.6022 | 0.021026 | 0.126454 | 3.59874 | 42.591 | FALSE |
| 21 | 19 | 2017-07-0 | 6.96 | 8.6022 | 0.021038 | 0.126788 | 3.58915 | 43.4661 | FALSE |
| 22 | 20 | 2017-07-0 | 6.96 | 8.60589 | 0.021042 | 0.126777 | 3.59976 | 43.3337 | FALSE |



(Source: https://gecco-2019.sigevo.org/index.html/Competitions)

# Water Quality Indicators

| Column name | Description |
|---|---|
| Time | Time of measurement, given in following format: yyyy-mm-dd HH:MM:SS |
| Tp | The temperature of the water, given in °C. |
| pH | pH value of the water |
| Cond | Electric conductivity of the water, given in S/m |
| Turb | Turbidity of the water, given in FNU |
| SAC | Spectral absorption coefficient, given in 1/m |
| PFM | Pulse-Frequency-Modulation, given in Hz |




# Our Conversation with DEP

Conductivity: is important is because it can tell how much dissolved substances, chemicals, and minerals are present in the water. Higher amounts of these impurities will lead to a higher conductivity.

Predict **Conductivity**



(Source: New York City 2018 Drinking Water Supply and Quality Report, GECCO 2019 Industrial Challenge)

# *03* Achievement

- Models Explanation
  - Ridge Regression

COLUMBIA | CBIPS
Center for Buildings, Infrastructure and Public Space

# Methodology

## Ridge Regression:

The objective function (also called the cost) to be minimized is the RSS (Residual Sum of Squares) plus the sum of square of the magnitude of weights. This can be depicted mathematically as:

$$Cost\ (W) = RSS(W) + \lambda * (sum\ of\ squares\ of\ weights)$$

$$= \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2 + \lambda \sum_{j=0}^{M} w_j^2$$

is the sum of the squares of residuals (deviations predicted from actual empirical values of data). It is a measure of the discrepancy between the data and an estimation model.

The term with lambda is often called 'Penalty' since it increases RSS. We iterate certain values onto the lambda and evaluate the model.



(Source:https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db)

# Python Coding -Ridge Regression


```python
df1 = pd.read_excel("C:/Users/user/Desktop/ML/watersample-test.xlsx").drop(["Ev
df1 = df1.dropna()
df1 = df1[["Tp","pH","Turb","SAC","PFM","Cond"]]
test_x = df1.loc[:,"Tp":"PFM"]
test_x = scaler.fit_transform(test_x)
test_y = df1.loc[:,"Cond"]


def model(size):
    X = data.loc[:size,"Tp":"PFM"]
    Y = data.loc[:size,"Cond"]
    X_train,X_test,y_train,y_test = train_test_split(X,Y, test_size = 0.2)
    #Use Ridge regression
    #Using cross validation to decide the best alpha for ridge
    alpha_ridge =  [0.1, 1, 10, 100, 1000,2000,4000,8000, 10000, 15000]
    model_ridge = RidgeCV(alphas = alpha_ridge).fit(X_train,y_train)


    #Compute test error
    pred_ridge = model_ridge.predict(test_x)
    test_error = mean_absolute_error(test_y,pred_ridge)
    #Compute validation error
    val_ridge = model_ridge.predict(X_test)
    val_error = mean_absolute_error(y_test,val_ridge)
    #Compute train error
    train_ridge = model_ridge.predict(X_train)
    train_error = mean_absolute_error(y_train,train_ridge)

    return test_error,val_error,train_error

#Computer the val_error and train_error with the interval of 5000
gap = 200
index = 0
test_error = []
val_error = []
train_error = []
```



(PictureSource:https://devrant.com/search?term=coding+meme)

# Results

Output →

```
The mean absolute error for the entire data set is(test_error)  0.0003074708634444559
The mean absolute error for the entire data set is(train_error)  0.00026323328485385234
The mean absolute error for the entire data set is (val_error)  0.00026521310090718396
The prediction are: [0.02135556 0.02134536 0.02131694 ... 0.02086572 0.02083075 0.02085077]

In [7]: print("The prediction are:",result[:20])
The prediction are: [0.02135556 0.02134536 0.02131694 0.02132844 0.02132125 0.02133675
 0.02130564 0.02131144 0.02132444 0.0213268  0.0213325  0.02131131
 0.02130093 0.0212625  0.02126737 0.02128587 0.02130741 0.02130025
 0.02129438 0.02128749]
```

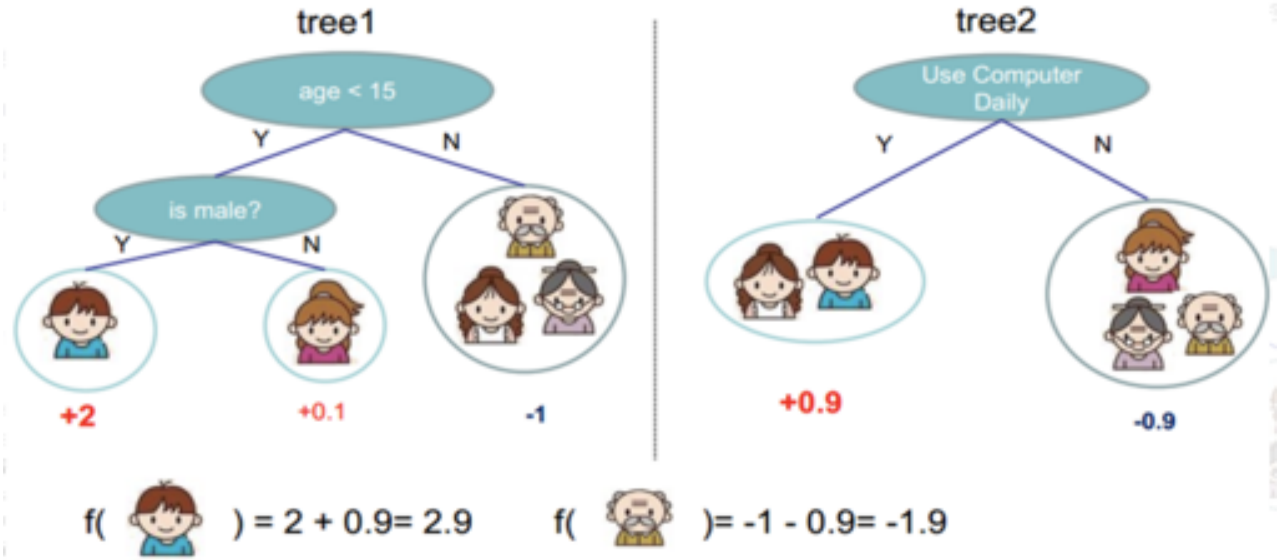
# *03* Achievement

- Models Explanation
  - Ridge Regression
  - Boosting Algorithm

# Methodology

## Boosting algorithm

Boosting algorithm combines the outputs from weak learner and creates a strong learner which eventually improves the prediction power of the model.






(Source:https://blog.csdn.net/a1b2c3d4123456/article/details/52849091 )

# Python Coding -Boosting Algorithm

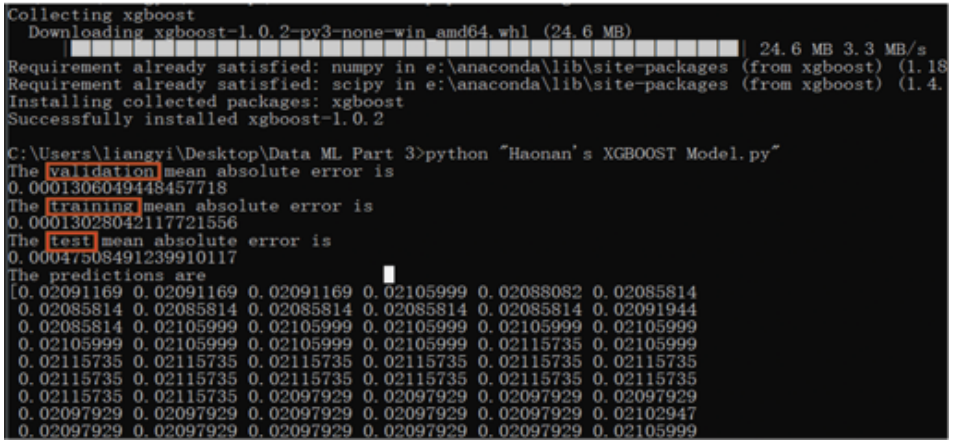
## Coding Process

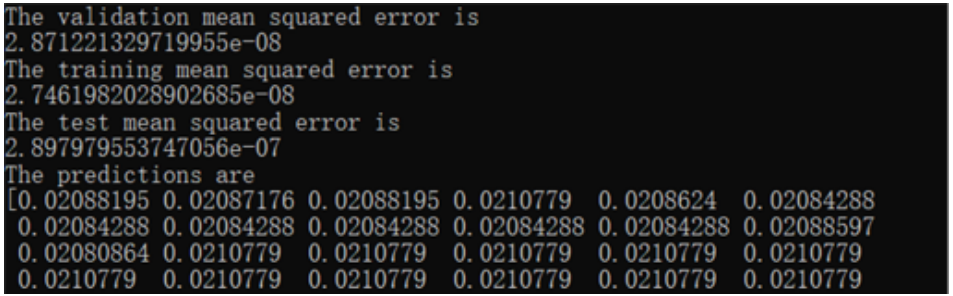
```python
def xgboost():
    df_train = pd.read_csv(fp_train)
    df_test = pd.read_csv(fp_test)
    #watersample_data = shuffle(watersample_data)
    y = df_train.Cond[:10000]
    y_test = df_test.Cond[:10000]
    #print(y)
    watersample_features = ['pH','Tp','Turb','SAC','PFM']
    X = df_train[watersample_features][:10000]
    X_test = df_test[watersample_features][:10000]
    #print(X)
    #watersample_data = watersample_data.dropna()
    train_X, val_X, train_y, val_y = train_test_split(X, y, test_size = 0.2)
    model = xgb.XGBRegressor(max_depth=5)
    #watersample_model.fit(X, y)
    model.fit(train_X, train_y)
    val_predictions = model.predict(val_X)
    test_predictions = model.predict(X_test)
    predicted_watersample_Cond = model.predict(X)

    print("The validation mean absolute error is")
    print(mean_absolute_error(val_y, val_predictions))

    print("The training mean absolute error is")
    print(mean_absolute_error(y, predicted_watersample_Cond))

    print("The test mean absolute error is")
    print(mean_absolute_error(y_test, test_predictions))
    #print("Making predictions for the following 5 watersamples:")
    print("The predictions are")
    print(model.predict(X[:300]))


if __name__ == '__main__':
    xgboost()
```

Split the data into two parts: training and validation

# Results

Output →



# 03 Achievement

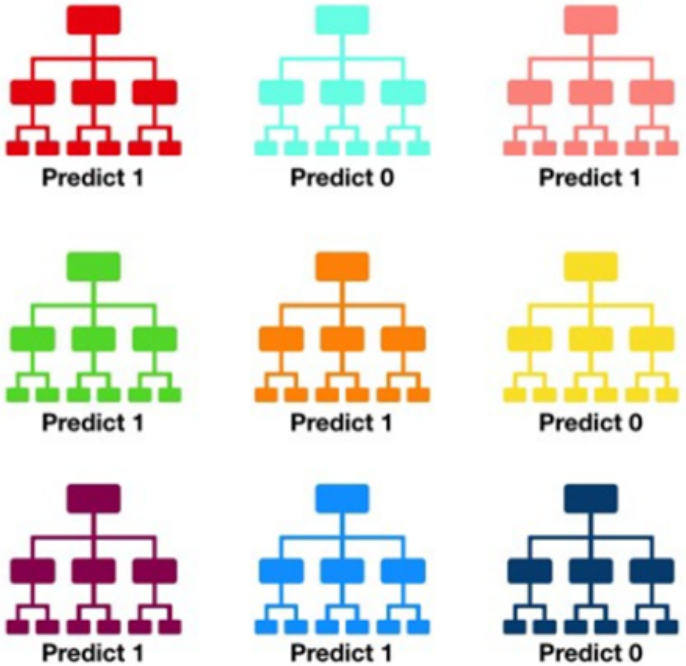
- Models Explanation
  - Ridge Regression
  - Decision Tree
  - Random Forest

# Methodology

## Random Forest Algorithm

**Decision Tree algorithm** belongs to the family of supervised learning algorithms. the decision tree algorithm can be used for solving **regression and classification problems**.

**Random Forest** consists of a **large number of individual decision trees**. Each individual tree in the random forest spits out a class prediction and the class with the **most votes becomes its model's prediction.**






(Source:https://towardsdatascience.com/understanding-random-forest-58381e0602d2)

# Python Coding - Random Forest Regressor

## Coding Process

```python
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
#from sklearn.utils import shuffle
fp_train = r'C:\Users\liangyi\Desktop\Data ML Part 3\watersample-train.csv'
fp_test = r'C:\Users\liangyi\Desktop\Data ML Part 3\watersample-test.csv'


def Forest_model():
    df_train = pd.read_csv(fp_train)
    df_test = pd.read_csv(fp_test)
    #watersample_data = shuffle(watersample_data)
    y = df_train.Cond[:10000]
    y_test = df_test.Cond[:10000]
    #print(y)
    watersample_features = ['pH','Tp','Turb','SAC','PFM']
    X = df_train[watersample_features][:10000]
    X_test = df_test[watersample_features][:10000]
    #print(X)
    #watersample_data = watersample_data.dropna()
    train_X, val_X, train_y, val_y = train_test_split(X, y, test_size = 0.2)
    model = RandomForestRegressor(max_depth=5)
    #watersample_model.fit(X, y)
    model.fit(train_X, train_y)
    val_predictions = model.predict(val_X)
    test_predictions = model.predict(X_test)
    predicted_watersample_Cond = model.predict(X)

    print("The validation mean squared error is")
    print(mean_squared_error(val_y, val_predictions))

    print("The training mean squared error is")
    print(mean_squared_error(y, predicted_watersample_Cond))

    print("The test mean squared error is")
    print(mean_squared_error(y_test, test_predictions))
    #print("Making predictions for the following 5 watersamples:")
```

# Results

Output

```
The validation mean absolute error is
0.00017848401165124973
The training mean absolute error is
0.0001804644757287498
The test mean absolute error is
0.00042701659617813703
The predictions are
[0.02106792 0.02106792 0.02106792 0.02106792 0.02106792 0.02106792
 0.02106792 0.02106792 0.02106792 0.02106792 0.02106792 0.02106792
 0.02106792 0.02106792 0.02106792 0.02106792 0.02106792 0.02106792
 0.02106792 0.02106792 0.02106792 0.02106792 0.02106792 0.02106792
```

```
The validation mean squared error is
4.436288942087643e-08
The training mean squared error is
4.618563612324684e-08
The test mean squared error is
2.4465875705867854e-07
The predictions are
[0.02106778 0.02106778 0.02106778 0.02106778 0.02106778 0.02106778
 0.02106778 0.02106778 0.02106778 0.02106778 0.02106778 0.02106778
 0.02106778 0.02106778 0.02106778 0.02106778 0.02106778 0.02106778
 0.02106778 0.02106778 0.02106778 0.02106778 0.02106778 0.02106778
```

# Models Comparison（MAE）

**Mean Absolute Error** is the average over the test sample of the absolute differences between prediction and actual observation.

| | Ridge Regression | Boosting | Random Forest |
|---|---|---|---|
| MAE (Training) | 0.000263233 | 0.000130280 | 0.000180485 |
| MAE (Validation) | 0.000265221 | 0.000130604 | 0.000182120 |
| MAE (Testing) | 0.000307470 | 0.000475084 | 0.000425865 |

MAE (Training) : Ridge Regression > Random Forest > Boosting.
MAE (Validation): Ridge Regression > Random Forest > Boosting.
MAE (Testing): Boosting> Random Forest> Ridge Regression.

# Models Comparison（MSE）

**Mean Squared Error** is the average squared difference between the estimated values and the actual value.

| | Ridge Regression | Boosting | Random Forest |
|---|---|---|---|
| MSE (Training) | 1.0580*e^-7 | 2.7461*e^-8 | 4.6185*e^-8 |
| MSE (Validation) | 9.9343*e^-8 | 2.8712*e^-8 | 4.4362*e^-8 |
| MSE (Testing) | 1.5453*e^-7 | 2.8979*e^-7 | 2.4465*e^-7 |

MSE (Training) : Ridge Regression > Random Forest > Boosting.
MSE (Validation): Ridge Regression > Random Forest > Boosting.
MSE (Testing): Boosting > Random Forest> Ridge Regression.

# 04 Summary

- Conclusion
- Improvement
- Current Trend
- Future Opportunity

# Conclusion & Improvement

## Conclusion

- After comparing with other data sets eg: NYC open data, we choose the industrial challenge data because the overall number of the variables and the completeness of the dataset was appropriate.
- Given three models, in term of the testing error, ridge regression performs better than other two models.

## Improvement

- More feature analysis and cross validation will be needed to further strengthen our conclusions and complete comparison among the different algorithms.
- On the one hand, we shall see whether the model can generalize to other data. On the other hand, we could apply other algorithms.
- Classification problems, which differs from regressions, could be our next focus.

# Current Trend & Future Opportunity of ML in Water Mgmt

## Current Trend

- Monitor water quality data.
- Anticipate water supply flow.
- Optimize management of sewage, treatment plants, and desalination plants.

## Future Opportunity

- The environmental time series data can contain outliers, anomalies and gaps. Our models need to be flexible, fast and general.
- Be able to develop good models for anomaly and fault detection for advanced study.
- Apply ML regression technique to predict even harder-to-measure metrics which would require appropriate datasets being available.
- Apply Deep Learning and more advanced AI technologies.



(Source:https://link.springer.com/article/10.1007/s00216-018-1015-9)

# Resources

- Ridge Regression for Better Usage
  https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db
  https://www.analyticsvidhya.com/blog/2016/01/ridge-lasso-regression-python-complete-tutorial/
- Python Model Tuning Methods Using Cross Validation and Grid Search
  https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db
  http://karlrosaen.com/ml/learning-log/2016-06-20/
  https://towardsdatascience.com/machine-learning-algorithms-part-14-cross-validation-and-ridge-regression-example-in-python-2d9e0c4de875
- Statistical Learning Theory
  https://en.wikipedia.org/wiki/Statistical_learning_theory
- NY Codes for Water Quality
  https://govt.westlaw.com/nycrr/Document/I4ed90412cd1711dda432a117e6e0f345?viewType=FullText&originationContext=documenttoc&transitionType=CategoryPageItem&contextData=(sc.Default)
- Water Quality Standards
  http://mrccc.org.au/wp-content/uploads/2013/10/Water-Quality-Salinity-Standards.pdf
- How machine learning can help?
  http://www.cs.toronto.edu/~kswersky/wp-content/uploads/WorkshopPresentation.pdf
- Data Source:
  https://www.th-koeln.de/mam/downloads/deutsch/hochschule/fakultaeten/informatik_und_ingenieurwissenschaften/rulesgeccoic2019.pdf
- Drinking Water:
  https://www.smartcitiesdive.com/news/how-ai-and-data-turn-city-water-management-from-an-art-to-a-science/559424/
  https://www.infrastructurereportcard.org/state-item/new-york/

# Thank You!

Grateful if any comment comes up!

You can find us at:
hy2636@columbia.edu
yl4322@columbia.edu
sf2970@columbia.edu