# Flood prediction model

## Problem:

To build a prediction model with a specified lead time to floods at a particular site.

## Literature:

Literature review paper:
https://arxiv.org/ftp/arxiv/papers/1908/1908.02781.pdf
Main points:

- The major ML algorithms applied to flood prediction include ANNs, neuro-fuzzy, adaptive neuro-fuzzy inference systems (ANFIS), support vector machines (SVM), wavelet neural networks (WNN), and multilayer perceptron (MLP)
- It was observed that the characteristics of the ML methods used varied significantly according to the period of prediction. The paper classifies lead time greater than a week as a long-term prediction.
- For both short-term or long-term rainfall–runoff modeling, overall, the accuracy, precision, and performance of most **decomposed ML** algorithms (e.g., WNN) were reported as better than those which were trained using un-decomposed time series.
- Longer lead time accuracy can be improved with an **autoregressive** model (eg RNN).
- **Ensembles** perform better than single model.
- In ensembles, however, it is noted that human decision as the input variable provided superior performance than models without.
- There is still space for improvements in ANN **architectures**.

Genetic algo:
https://www.researchgate.net/publication/273160666_Genetic_algorithm_and_fuzzy_neural_networks_combined_with_the_hydrological_modeling_system_for_forecasting_watershed_runoff_discharge
Limitations of the paper:

- Comparison is done only between a modelling software named the hydrological engineering center hydrological modeling system (HEC-HMS), and 2 hybrid systems which are HEC-HMS-GANN and HEC-HMS-ANFIS.
- The two hybrid systems augments the original HEC-HMS model with a genetic algorithm neural network (GANN) and an adaptive neuro-fuzzy inference system approach (ANFIS) respectively.
- Drawback of genetic algo: slow convergence speed to optima when a large network is involved / evolved.

Advantages of genetic algo:

- General learning algorithm capable of semi/self-supervised learning
- Can learn to evolve network architectures from scratch (eg NEAT)

An example of genetic neural network algorithm: NeuroEvolution of Augmenting Topologies (NEAT) implemented in Python:
https://github.com/drallensmith/neat-python

## Project milestone:

1. Set up relevant ML framework
2. Collect data
3. Initial tests
4. Survey model architectures based on findings
5. Iterate on findings (hyperparams tuning, ensemble, etc)

## Resources:

1. https://www.hec.usace.army.mil/software/hec-hms/
2. https://github.com/esowc/ml_flood
3. https://github.com/wimagguc/tf-weather
4. https://www.ijcaonline.org/archives/volume143/number11/zaytar-2016-ijca-910497.pdf
5. http://publicinfobanjir.water.gov.my/View/OnlineFloodInfo/PublicRainFall.aspx?scode=WLH
6. http://publicinfobanjir.water.gov.my/View/OnlineFloodInfo/PublicWaterLevel.aspx?scode=WLH
7. https://www.ecmwf.int/en/forecasts/datasets/browse-reanalysis-datasets
8. Variables: https://nbviewer.jupyter.org/github/esowc/ml_flood/blob/master/notebooks/1_data_download_analysis_visualization/1.03_data_overview.ipynb
9. Meteo regression Linear Regression Predictor and Predictand Linear Regression

## Possible challenges:

- Non-stationary nature of time series data
- Noisy / low signal-to-noise ratio
- Trend shift causing problems for backtesting

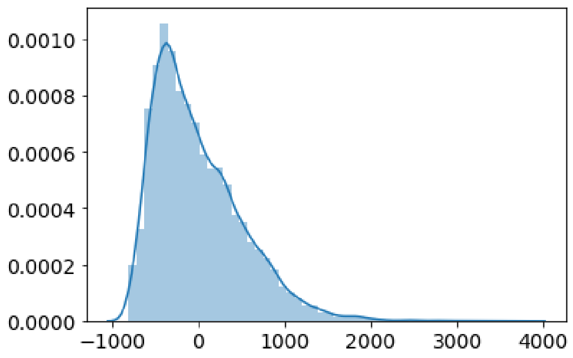
## Encountered issues:

- Data scraping from Malaysian Met / Info Banjir website:
    - Met only provides 24-hour rainfall data (ranges from 0-2mm)
    - Info Banjir only provides 7-day rainfall data (ranges from 0-61mm)
- Problems with generating synthetic data:
    - Unknown prior distribution of rainfall

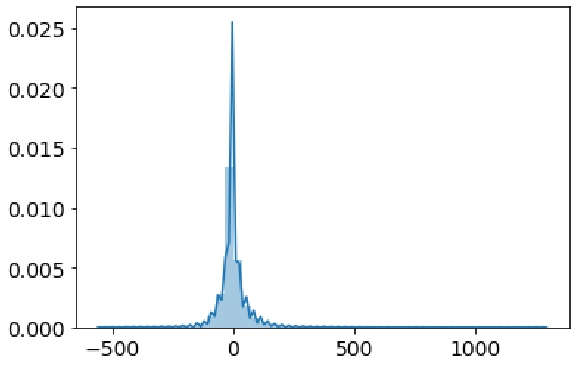
## Completed / In-progress:

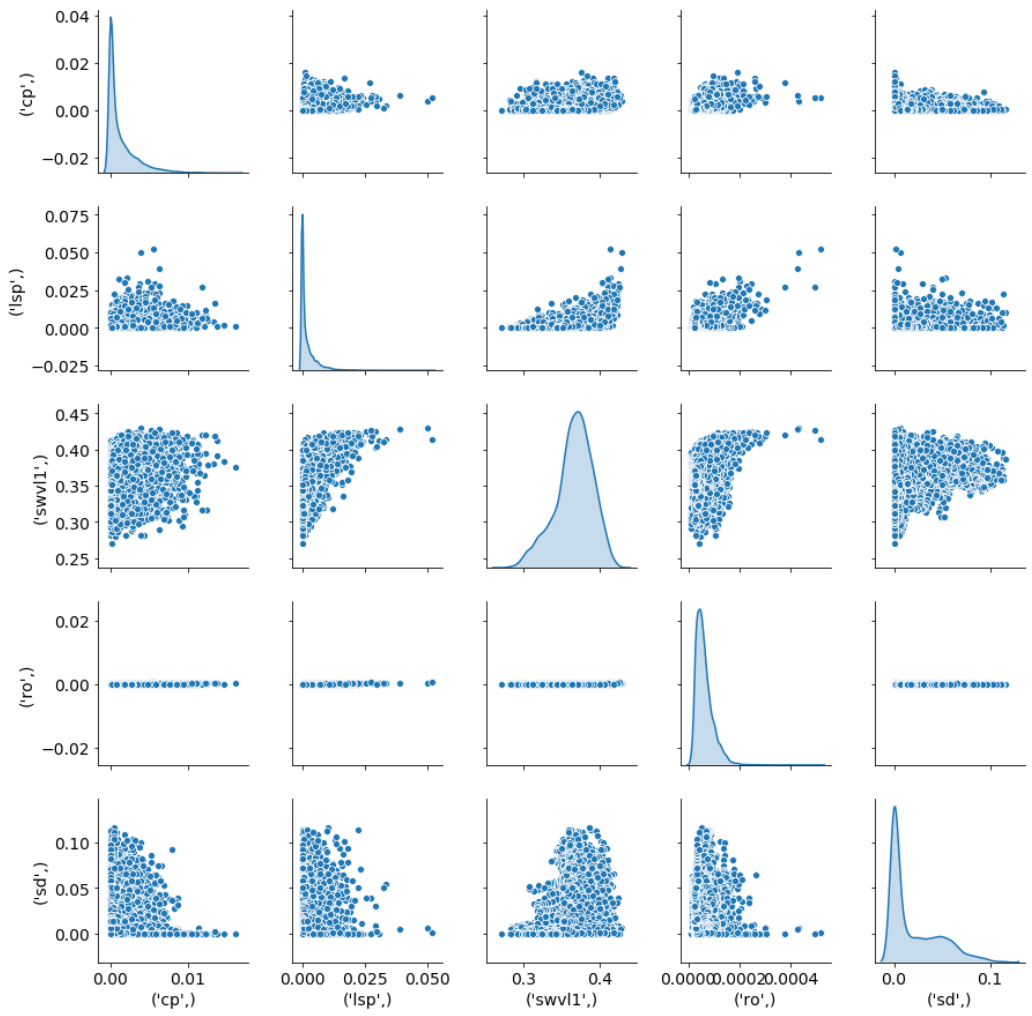
- Data parsing and splitting into train, validation and test sets - COMPLETED
- Data inspection using pandas and seaborn that shows the joint distribution of pairs of columns from the training set- COMPLETED

**distribution of discharge**



**distribution of change in discharge**





- Core NN training code in TensorFlow 2.0 - IN PROGRESS

# Operator’s outfit detection

## Problem:

To detect whether an operator is wearing a mask, gloves, head cover and proper shirt.

## Proposed solution:

It can be done by using detection framework such as Faster-RCNN or Mask-RCNN. A straightforward way of doing this is by treating it as a classification problem, for example (wearing masks vs not wearing masks, wearing head cover vs not wearing head cover, etc).

We can use transfer learning to initialise the selected model and then fine tuned it on our own dataset. Then we can only evaluate our own model on our dataset.

There are two existing deep learning detection frameworks, tensorflow object detection api and detectron 2. We recommend using detectron 2 because it is better maintained, and probably easier to set up and kick start for prototyping.

## Possible challenges:

- Need to build a dataset that is catered to our requirements. It must consist images of operators in different outfits. They will then have to be annotated before we can start our training.
- GPU intensive training process

## References:

**https://blogs.bing.com/search-quality-insights/2017-06/beyond-text-queries-searching-with-bing-visual-search**

**https://www.kaggle.com/c/imaterialist-fashion-2019-FGVC6/overview**

**https://suited.fit/assessflow**

## Project milestone:

1. Set up object detection framework (tensorflow object detection api / detectron 2)
2. Collect images from target domain
3. Annotate collected images (do we need our own annotation tool?)
4. Find suitable fashion dataset for pre-training
5. Run fine tuning
6. Initial tests
7. Survey model architectures based on findings
8. Iterate on findings (hyperparams tuning, ensemble, etc)

## Completed / In-progress:

- TensorFlow Object Detection API and PyTorch Detectron2 built into Docker images.
- Downloading and initial exploration of Fashion dataset (in-progress)