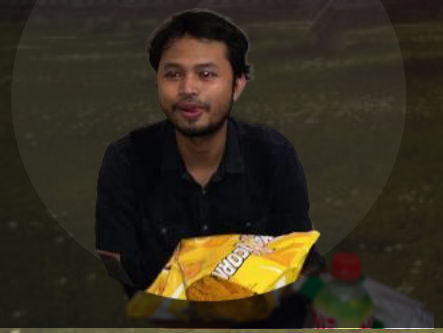
Team Alexandra presents
level 1 pitch



Atwin Paramudya
BNI

Wayan Rezaldi
BNI

Kevin Octavian
Astra

Dataset details

The dataset comprises of a whole year of weather data in two forms:

- Weather data: Interval 1 min (22896000 rows), including the target feature
- Sky camera data: Image data with 10 mins interval and available only during day time

We were asked to predict the total cloud cover percentage (TCC%) (available in weather data) for the 4 upcoming 30-min intervals. The dataset is mostly clean with no NaNs, the only problems, besides that the very huge dataset, are that TCC% consists of negative values: -1 for night time TCC% values and -7999 presumably for the times the sensor fails to read the data.

During the development for stage 1, we utilize only the **weather data**.

Dataset details

The dataset comprises of a whole year of weather data in two forms:

1. **Weather data**: 1 min interval (22.9M rows), including the target feature
2. **Sky camera data**: Image data with 10 mins interval and available only during daytime

Data characteristics

- No null values
- TCC% consists of negative values: -1 for night time
- Presumably for the times the measurement failed the TCC% value is -7999

The preprocessing

- Resample the dataset by averaging some number of data points. For the final models, the number is decided to be 2, where the accuracy is best (compared to some bigger numbers—less data) but with training time not exceeding a day work
- Remove nighttime records **(TCC% = -1)** from the training dataset since the test set has negligible amount of nighttime data
- Replacing outlier records **(TCC% = -7999)** with the average of the previous and next non-negative values

Our solution solely depends on the dataset provided by the committee.

Experiment results

- Model details

Our final models are trained from scratch. We didn’t use any pre-trained model nor done any transfer learning in building the final model. We didn’t even do hyperparameter tuning.

Experiment results

- Performance details

To optimize the performance, we did do:

- Replace the outliers (-7999 TCC% values) with the average of the closest non-negative values during preprocessing.
- Trying out 5 traditional ML models (linear regression, decision tree, SVR, Naïve Bayes, XGBoost) to find the most fitting model; SVR turned out to have the highest validation accuracy.
- We tried to make the dataset time-series-like by engineering new features from the past by shifting these features up from some number of previous data points.

...

Experiment results

- Performance details
  - Ensemble the models:
    - **Naïve**: After a quick look at the dataset, we picked up that over 120 mins horizon the TCC% value does not change much if at all. The first thing we did was try a naïve approach by taking the last non-negative value as our prediction for the 4 upcoming time intervals. We acquired an even better result than we anticipated (after doing all the aforementioned preprocessing as well), a solid 88.87167—easily into the top 15 even after 3 weeks into the cournament.
    - **Non-naïve**: We figured that if we have any better-than-random prediction, we could ensemble it with the already decent-performing naïve model and improve it, ever so slightly. We tourned out to be right.

      Our best non-naïve model on its own achieved accuracy of over 87, and when ensembled (with 1:5 ratio to the naïve solution) reached our best record with an accuracy of 89.2.

    This ensemble with the non-naïve solution is essential because with only the naïve prediction, we would rank around 30 or 40-ish in public leaderboard (the competition is fierce around the lower end of 89 mark) and wouldn't stand a chance of getting through.

Additional information

- Hardware

We did not use accelerators. We however utilize Google Colab for most of the training process and our own desktop for parallel work while the model is being trained on Colab.

- Software

Python 3.9 programming language and its libraries (pandas, numpy, scikit-learn)

Jupyter Notebook and Google Colab

- Performance numbers

Time spent training the models on Colab that gets us the final submitted score was around 2-3 hours.

- License

There is no license of any form applied to our solution.