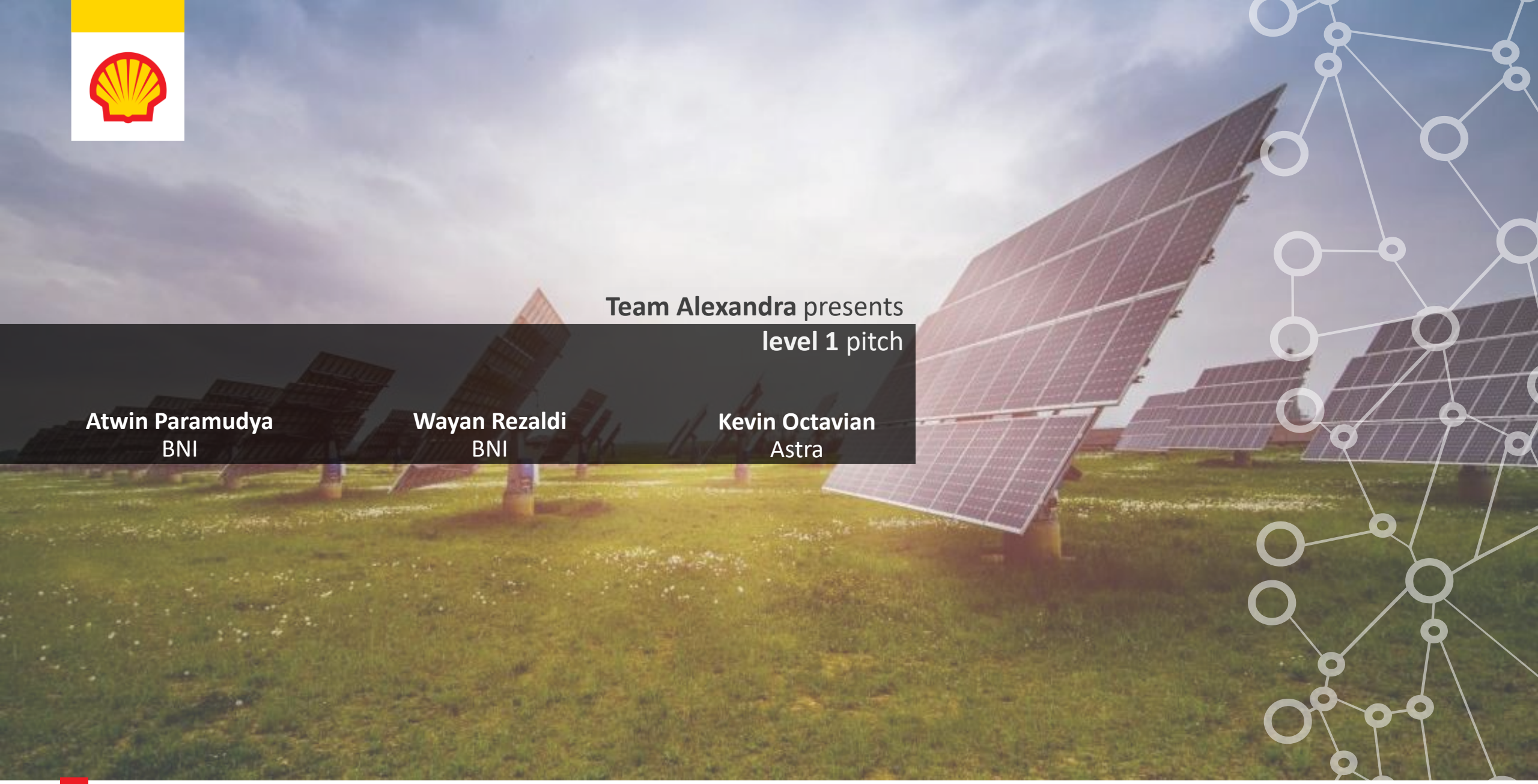

Team Alexandra presents

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.

Dataset details

Hence, the preprocessing we did was:

- 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.
- After reviewing the test set, we find that there is very little -1 TCC% values on them so in order to further shrink the dataset, we decide to remove the data points with -1 TCC% values from the dataset.
- Replacing the -7999 values with the average of the previous and next non-negative values to remove what we deem as outliers.

Our team solely uses the dataset provided by the committee and it has no license that we know of.

Existing/pre-existing work

Our team did not encounter any other work similar work and thus did the work without any reference in mind.

Experiment results

- Model details

We only leveraged the weather data—without using the image data at all—therefore we didn’t have the chance to use any pre-trained model nor done any transfer learning.

Experiment results

- Performance details

What we did do to improve the models:

- Replacing the outliers (-7999 TCC% values) with the average of the closest non-negative values during preprocessing.
- Evaluating 5 traditional ML models (linreg, deviion tree, svr, .., xhb) to find the fitting model; turns out to be SVR.
- Even though our team utilizes conventional ML models, 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.

Hardware, software, training, license

- Hardware used for stage 1

We did not use accelerators and only utilize google colab for most of the training process and our own desktop for parallel work while training.

- Software packages used for stage 1

Python, scikit-learn, jupyter notebook and Google Colab.

- Performance numbers

Total time spent on training the models that get us the best score was around 2-3 hours.

- License

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