COMPFEST

DKI Jakarta **Floods**

Region Clusterization and Forecasting

Ad Astra Explorers

Our team

Sonia Julianti

linkedin.com/in/sonia-julianti-65bb69227
SoniaJulianti (github.com)

Andi Eka N

linkedin.com/in/andi-eka-nugraha
AnCodingML (AnCodingML) (github.com)

Hanum Fazah A

linkedin.com/in/hanumfazah
hanumfzh (github.com)

Source Dataset:

Kumpulan Dataset | Open Data Jakarta

(for more details, open the Dataset Preparation section of the document)

Source Code:

AnCodingML/Jakarta-Flood-Forecasting-and-Clustering (github.com)

Dashboard:

https://bit.ly/AdAstraExplorers

Analysis and Modeling Methodology

1. Problem Understanding
2. Dataset Preparation
3. Data Understanding & Exploratory Data Analysis
4. Clustering
    1. Data Preparation
    2. Feature Enginering
    3. Modeling & Hyperparameter Tuning
    4. Model Evaluation
5. Forecasting
    1. Data Preparation
    2. Feature Enginering
    3. Modeling & Hyperparameter Tuning
    4. Model Evaluation
6. Model Testing
7. Recomendation

Problem Understanding

Current Issue

Floods are an annual problem in the DKI Jakarta region, and the government is required to anticipate and mitigate floods in order to reduce casualties.

As a Data Science team, our task is to perform clustering of flood risk areas based on flood levels and damages caused by floods. We also need to conduct forecasting to assess the potential for floods in the next five years using Machine Learning models.

Subsequently, we will provide suggestions and recommendations based on the findings of the data analysis and ML modeling.

Expetation

Goals

Generating recommendations to mitigate economic losses and flood levels, segmenting flood-prone areas for infrastructure recommendations, and forecasting the future conditions of the regions to prepare for flood incidents.

Objectives

- The flood area mapping is created using several parameters, including **floodwater level, duration of inundation, and the number of casualties** (If additional parameters are required, the respective datasets need to be obtained beforehand).
- **The flood mapping is categorized into 3-5 different levels**, and each level will receive different treatments from the government, which will be further analyzed once the results are obtained.
- **Forecasting can be employed to reduce the number of casualties** (deaths, missing persons, injuries).
- **Forecasting can also be conducted for floodwater levels and climate** (datasets need to be obtained).

Variable Metric

- Flood-affected areas
- Floodwater level conditions in each area
- Damages caused by floods in each area

Dataset Preparation

Required Features

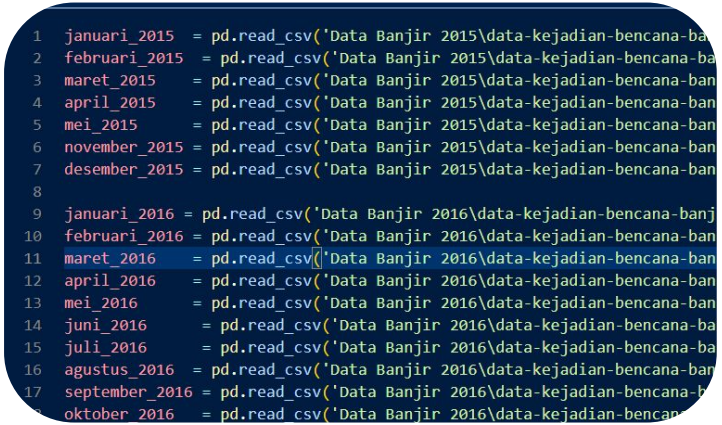
1. Water levels in each area, including administrative cities, sub-districts, and neighborhood units (RW).
1. Losses incurred in each area.
2. Vulnerability of each area to floods.
3. Proximity of each area to the nearest river.
4. Floodwater levels and the duration of flooding.

Dataset Source

- https://data.jakarta.go.id/dataset/rekap-banjir-tahun-2016
- https://data.jakarta.go.id/dataset/data-kejadian-bencana-banjir-tahun-2017-di-provinsi-dki-jakarta
- https://data.jakarta.go.id/dataset/data-kejadian-bencana-banjir-di-provinsi-dki-jakarta-tahun-2018
- https://data.jakarta.go.id/dataset/data-kejadian-bencana-banjir-di-provinsi-dki-jakarta-tahun-2019
- https://data.jakarta.go.id/dataset/data-kejadian-bencana-banjir-di-provinsi-dki-jakarta-tahun-2020
- https://data.jakarta.go.id/dataset/daerahrawanbanjirdkijakarta
- https://data.jakarta.go.id/dataset/data-wilayah-potensi-banjir-akibat-luapan-sungai
- https://data.jakarta.go.id/dataset/daerah-rawan-banjir-provinsi-dki-jakarta

Dataset Agregation

Aggregation is performed on **the historical flood dataset in DKI Jakarta for the past 5 years**, from 2016 to 2020. The dataset includes flood conditions per year, and each year has monthly datasets. Aggregation is done to **combine the months within each year**.



```
januari_2015  = pd.read_csv('Data Banjir 2015\data-kejadian-bencana-ba
februari_2015  = pd.read_csv('Data Banjir 2015\data-kejadian-bencana-ba
maret_2015    = pd.read_csv('Data Banjir 2015\data-kejadian-bencana-ban
april_2015    = pd.read_csv('Data Banjir 2015\data-kejadian-bencana-ban
mei_2015      = pd.read_csv('Data Banjir 2015\data-kejadian-bencana-ban
november_2015 = pd.read_csv('Data Banjir 2015\data-kejadian-bencana-ban
desember_2015 = pd.read_csv('Data Banjir 2015\data-kejadian-bencana-ban

januari_2016 = pd.read_csv('Data Banjir 2016\data-kejadian-bencana-banj
februari_2016 = pd.read_csv('Data Banjir 2016\data-kejadian-bencana-ban
maret_2016    = pd.read_csv('Data Banjir 2016\data-kejadian-bencana-ban
april_2016    = pd.read_csv('Data Banjir 2016\data-kejadian-bencana-ban
mei_2016      = pd.read_csv('Data Banjir 2016\data-kejadian-bencana-ban
juni_2016      = pd.read_csv('Data Banjir 2016\data-kejadian-bencana-ba
juli_2016      = pd.read_csv('Data Banjir 2016\data-kejadian-bencana-ba
agustus_2016  = pd.read_csv('Data Banjir 2016\data-kejadian-bencana-ban
september_2016 = pd.read_csv('Data Banjir 2016\data-kejadian-bencana-b
oktober_2016   = pd.read_csv('Data Banjir 2016\data-kejadian-bencan
```

```
januari_2015['bulan'] = 'Januari'
februari_2015['bulan'] = 'Februari'
maret_2015['bulan'] = 'Maret'
april_2015['bulan'] = 'April'
mei_2015['bulan'] = 'Mei'
november_2015['bulan'] = 'November'
desember_2015['bulan'] = 'Desember'
year_2015 = pd.concat([januari_2015, februari_2015,
                       maret_2015, april_2015, mei_2015,
                       november_2015, desember_2015])
year_2015['tahun'] = '2015'

januari_2016 ['bulan'] = 'Januari'
februari_2016 ['bulan'] = 'Februari'
maret_2016 ['bulan'] = 'Maret'
april_2016 ['bulan'] = 'April'
```

Before merging the monthly datasets, it is necessary to **add a column indicating the month** when the flood occurred. After that, **add a feature for the year** when the flood occurred according to the current dataset.

For more details open jupyter notebook here

Dataset Cleaning

- The "tanggal_kejadian" feature: In one data row, this feature contains multiple data points that need to be split so that only one data is displayed per row. This is done to facilitate data reading and improve comprehension.

- The "rw" feature: Similar to the previous feature, in one data row, this feature also contains multiple data points. Therefore, it is necessary to split the data so that only one data is displayed per row, making it easier to understand.

- The "ketinggian_air" feature: This feature also requires cleaning to remove ambiguous values. The maximum height value is taken.

To see the specific keywords that were changed, please refer to the source code.

Before :

```
1  year_2017['tanggal_kejadian'].unique()
✓ 0.0s
```

```
array(['10 - 11\n(2 Hari)', '10\n(1 Hari)', '11\n(1 Hari)', 0.0, 1.0, 2.0,
       nan, '23 - 24', '28', '3', '4', '1 - 3', '1 - 8', '2 - 3', '1 - 4',
       '2 - 3, 8', '1 - 2, 8', '1 - 2', '1 - 3, 17 - 18', '8', '18 - 19',
       '23', '1', '18', '8 - 9', '2 - 3, 28', '1 - 2, 8 - 9, 18', '7 - 8',
       '1 - 3, 7 - 8, 18, 23', '2', '11 - 12\n(2 Hari)', '27\n(1 Hari)',
```

After :

```
17 year_2017['tanggal_kejadian'] = year_2017['tanggal_kejadian'].str.replace('tgl.', '')
18 year_2017['tanggal_kejadian'] = year_2017['tanggal_kejadian'].str.replace(r'\(|\)', '')
19 year_2017['tanggal_kejadian'] = year_2017['tanggal_kejadian'].str.replace('1-3', '1,2,3')
20 year_2017['tanggal_kejadian'] = year_2017['tanggal_kejadian'].str.replace('3-5', '3,4,5')
21 year_2017['tanggal_kejadian'] = year_2017['tanggal_kejadian'].str.replace('10-12', '10,11,12')
22 year_2017['tanggal_kejadian'] = year_2017['tanggal_kejadian'].str.replace('11-13', '11,12,13')
23 year_2017['tanggal_kejadian'] = year_2017['tanggal_kejadian'].str.replace('-', ',')
24 year_2017['tanggal_kejadian'].unique()
```

```
array(['10,11', '10', '11', nan, '23,24', '28', '3', '4', '1,2,3', '1,8',
       '2,3', '1,4', '2,3,8', '1,2,8', '1,2', '1,2,3,17,18', '8', '18,19',
       '23', '1', '18', '8,9', '2,3,28', '1,2,8,9,18', '7,8',
       '1,2,3,7,8,18,23', '2', '11,12', '27', '3,4,5,11,12', '11,12,26',
       '10,11,12,26,27', '4,5,11,12', '11,14,22,23', '14,20', '14',
```

Concat to be one dataset:

```
1  dataset = pd.concat([year_2015, year_2016, year_2017, year_2018, year_2019, year_2020])
```

Split column:

```
1  dataset['rw'] = dataset['rw'].str.split(',')
2  dataset = dataset.explode('rw')
3  dataset = dataset.drop('jumlah_terdampak_rw', axis=1)
4
5  dataset['tanggal_kejadian'] = dataset['tanggal_kejadian'].str.split(',')
6  dataset = dataset.explode('tanggal_kejadian')
```

For more details open jupyter notebook here

Dataset Cleaning

Before :

```
0   kota_administrasi            11454 non-null  object
1   kecamatan                    11454 non-null  object
2   kelurahan                    11454 non-null  object
3   rw                           11454 non-null  object
4   jumlah_terdampak_rt          11454 non-null  float64
5   jumlah_terdampak_kk          11454 non-null  object
6   jumlah_terdampak_jiwa        11454 non-null  object
7   ketinggian_air               11454 non-null  int64
8   tanggal_kejadian             11454 non-null  object
9   lama_genangan                11454 non-null  float64
10  jumlah_meninggal             11454 non-null  float64
11  jumlah_hilang                11454 non-null  float64
12  jumlah_luka_berat            11454 non-null  float64
13  jumlah_luka_ringan           11454 non-null  float64
14  jumlah_pengungsi_tertinggi   11454 non-null  float64
15  jumlah_tempat_pengungsian    11454 non-null  float64
16  nilai_kerugian               11454 non-null  float64
17  bulan                        11454 non-null  object
18  tahun                        11454 non-null  object
```

Drop unnecessary data :

```
dataset = dataset.drop(dataset[dataset['tanggal_kejadian']==''].index)
dataset = dataset.drop(dataset[dataset['rw']==''].index)
dataset = dataset.drop(dataset[dataset['jumlah_terdampak_kk']==' '].index)
```

Handling type data :

```
dataset['tanggal_kejadian'] = dataset['tanggal_kejadian'].astype('int')
dataset['rw'] = dataset['rw'].astype('int')
dataset['tahun'] = dataset['tahun'].astype('int')
dataset['jumlah_terdampak_jiwa'] = dataset['jumlah_terdampak_jiwa'].astype('float')
dataset['jumlah_terdampak_kk'] = dataset['jumlah_terdampak_kk'].astype('float')
```

After :

```
0   kota_administrasi            7227 non-null   object
1   kecamatan                    7227 non-null   object
2   kelurahan                    7227 non-null   object
3   rw                           7227 non-null   int32
4   jumlah_terdampak_rt          7227 non-null   float64
5   jumlah_terdampak_kk          7227 non-null   int32
6   jumlah_terdampak_jiwa        7227 non-null   int32
7   ketinggian_air               7227 non-null   int64
8   tanggal_kejadian             7227 non-null   int32
9   lama_genangan                7227 non-null   float64
10  jumlah_meninggal             7227 non-null   float64
11  jumlah_hilang                7227 non-null   float64
12  jumlah_luka_berat            7227 non-null   float64
13  jumlah_luka_ringan           7227 non-null   float64
14  jumlah_pengungsi_tertinggi   7227 non-null   float64
15  jumlah_tempat_pengungsian    7227 non-null   float64
16  nilai_kerugian               7227 non-null   float64
17  bulan                        7227 non-null   object
18  tahun                        7227 non-null   int32
```

Drop undefined data in the 'tanggal_kejadian' (occurrence date), 'rw', and 'jumlah_terdampak_kk' (affected household count) features.

Then, convert the data types of the 'tanggal_kejadian' (occurrence date), 'rw', 'tahun' (year), 'jumlah_terdampak_jiwa' (affected population count), and 'jumlah_terdampak_kk' (affected household count) features to integer.

For more details open jupyter notebook here

Aggregation Dataset

Import dataset & drop unnecessary column :

```
sungai_meluap = pd.read_csv('potensi-wilayah-banjir-akibat-aliran-sungai-2013.csv')
0.3s

drop = ['kota','kecamatan']
sungai_meluap =  sungai_meluap.drop(drop, axis=1)
sungai_meluap['kelurahan'] = sungai_meluap['kelurahan'].str.title()
```

To add the name of the river in each region, do Import the dataset then remove the 'kota' and ‘kecamatan’ features

Left join on ‘kelurahan’ and then fill in the missing data :

```
dataset_2 = dataset.merge(sungai_meluap, on='kelurahan', how='left')
0.0s

dataset_2['sungai'] = dataset_2['sungai'].fillna('tdk ada sungai rawan meluap')
```

Then join left to the previous dataset and fill in the blank value in the feature to be 'tdk ada sungai rawan meluap'

For more details open jupyter notebook here

Aggregation Dataset

Import dataset & make new features :

```
rawan_banjir_2020 = pd.read_csv('daerah-rawan-banjir-provinsi-dki-jakarta(2020).csv')
rawan_banjir_2020['rawan_banjir_2020'] = '1'
drop = ['kota','kecamatan']
rawan_banjir_2020 = rawan_banjir_2020.drop(drop, axis=1)
rawan_banjir_2020['kelurahan'] = rawan_banjir_2020['kelurahan'].str.title()
rawan_banjir_2020
```

```
rawan_banjir_2014 = pd.read_csv('daerah-rawan-banjir-dki-jakarta(2014).csv')
rawan_banjir_2014['rawan_banjir_2014'] = '1'
drop = ['Wilayah', 'Kecamatan']
rawan_banjir_2014 = rawan_banjir_2014.drop(drop, axis=1)
rawan_banjir_2014
```

To add flood-prone areas in 2014 and 2020, import both datasets and then in each dataset create a new feature named rawan_banjir_(year)' then fill in the feature data with 'True' to confirm all areas in the dataset are prone areas flood. Then drop the 'Kota' and 'Kecamatan' features because they are not needed.

Left join dataset on ‘kelurahan’:

```
rawan_banjir = rawan_banjir_2020.merge(rawan_banjir_2014, on=['kelurahan'], how='left')
dataset_2 = dataset_2.merge(rawan_banjir, on=['kelurahan'], how='left')
```

Join into the previous dataset so that there is a combination of information.

Features on dataset

0	kota_administrasi	35737 non-null	object
1	kecamatan	35737 non-null	object
2	kelurahan	35737 non-null	object
3	rw_x	35737 non-null	int32
4	jumlah_terdampak_rt	35737 non-null	float64
5	jumlah_terdampak_kk	35737 non-null	int32
6	jumlah_terdampak_jiwa	35737 non-null	int32
7	ketinggian_air	35737 non-null	int64
8	tanggal_kejadian	35737 non-null	int32
9	lama_genangan	35737 non-null	float64
10	jumlah_meninggal	35737 non-null	float64
11	jumlah_hilang	35737 non-null	float64
12	jumlah_luka_berat	35737 non-null	float64
13	jumlah_luka_ringan	35737 non-null	float64
14	jumlah_pengungsi_tertinggi	35737 non-null	float64
15	jumlah_tempat_pengungsian	35737 non-null	float64
16	nilai_kerugian	35737 non-null	float64
17	bulan	35737 non-null	object
18	tahun	35737 non-null	int32
19	sungai	35737 non-null	object
20	rawan_banjir_2020	33533 non-null	object
21	rw_y	32840 non-null	float64
22	rawan_banjir_2014	32840 non-null	object

For more details open jupyter notebook here

Dataset Final

New array dataset :

```
x = ['kota_administrasi', 'kecamatan', 'kelurahan', 'rw',
       'jumlah_terdampak_rt', 'jumlah_terdampak_kk', 'jumlah_terdampak_jiwa',
       'ketinggian_air', 'tanggal_kejadian', 'bulan', 'tahun','lama_genangan',
       'jumlah_meninggal', 'jumlah_hilang', 'jumlah_luka_berat',
       'jumlah_luka_ringan', 'jumlah_pengungsi_tertinggi',
       'jumlah_tempat_pengungsian', 'nilai_kerugian',
       'sungai', 'rawan_banjir_2020', 'rawan_banjir_2014']
dataset_2 = dataset_2.reindex(columns=x)
```

Drop ‘rw_y’(redundant) & rename ‘rw_x’ :

```
dataset_2 = dataset_2.drop('rw_y', axis=1)
dataset_2 = dataset_2.rename(columns={'rw_x': 'rw'})
```

Adjust the column names then rearrange the features in the dataset to make it easier to read the dataset

Save Dataset :

```
dataset_2.to_csv('dataset_banjir_DKI_Jakarta.csv')
```

Save new datasets for analysis and modeling by other data teams.

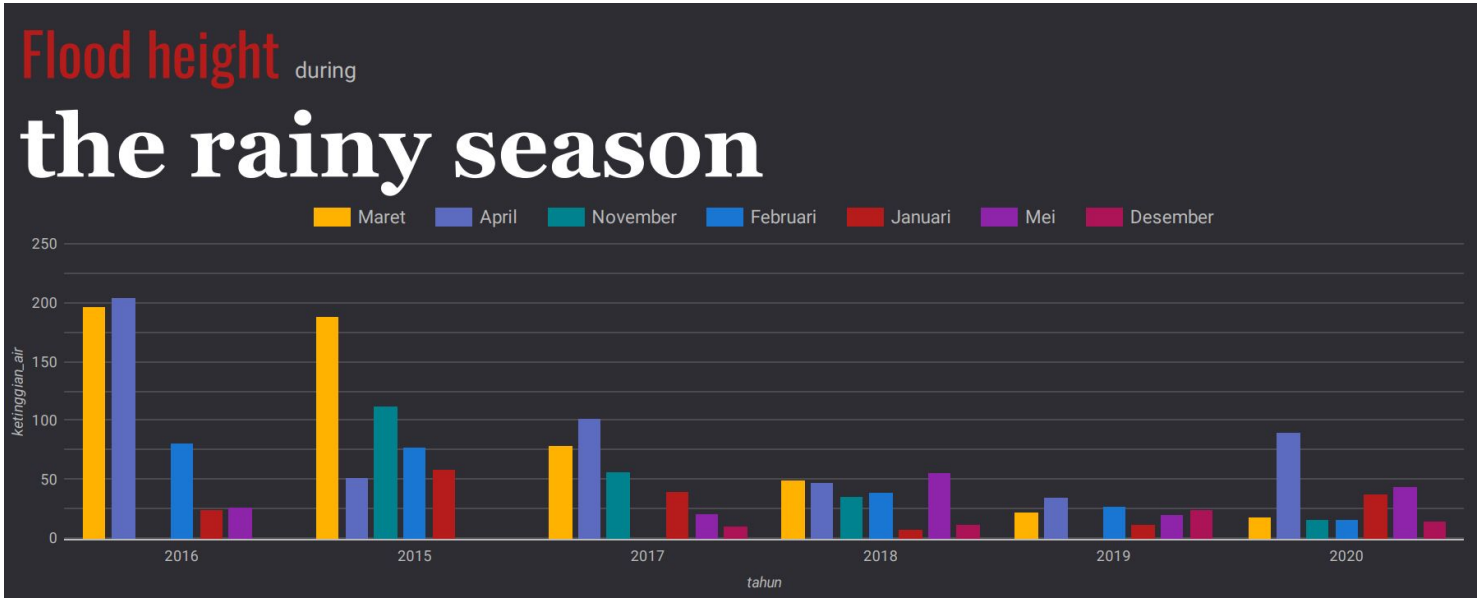
For more details open jupyter notebook here

Data Understanding & Exploratory Data Analysis

Data Understanding

Link Dashboard

https://bit.ly/AdAstraExplorers

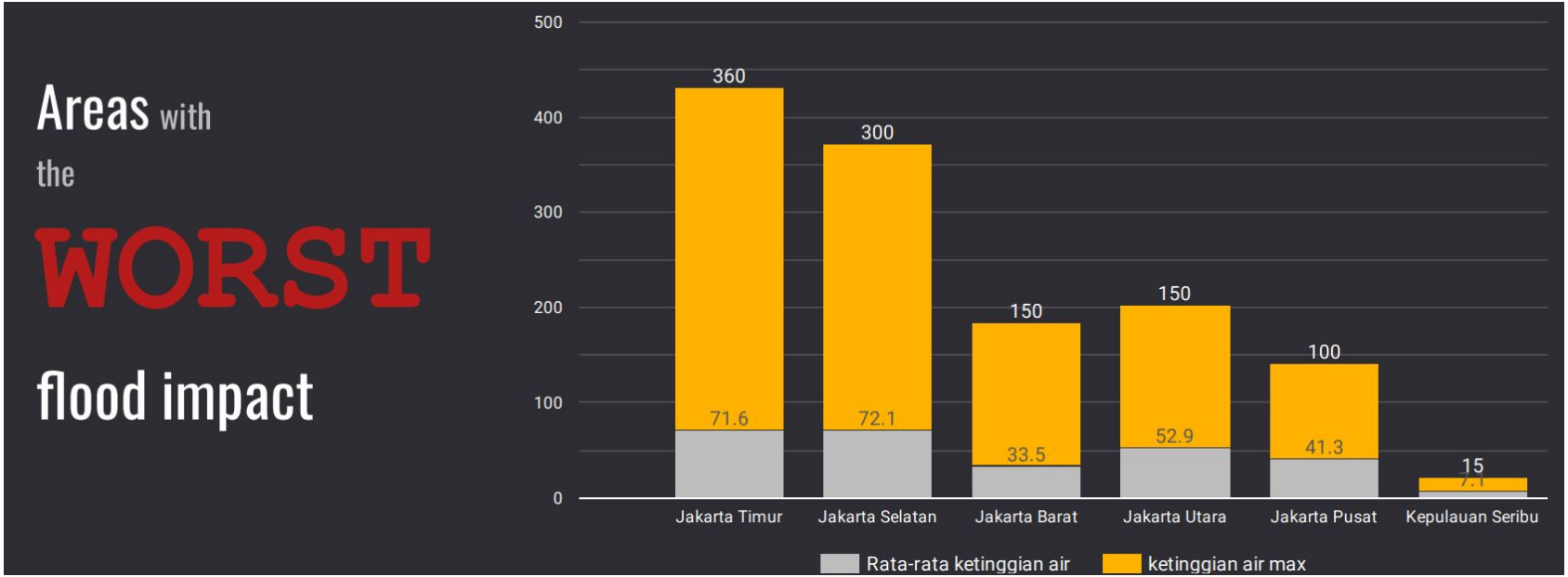


The highest average flood heights occur in the rainy season from 2016-2020, dominated in March and April

Data Understanding

Link Dashboard

https://bit.ly/AdAstraExplorers

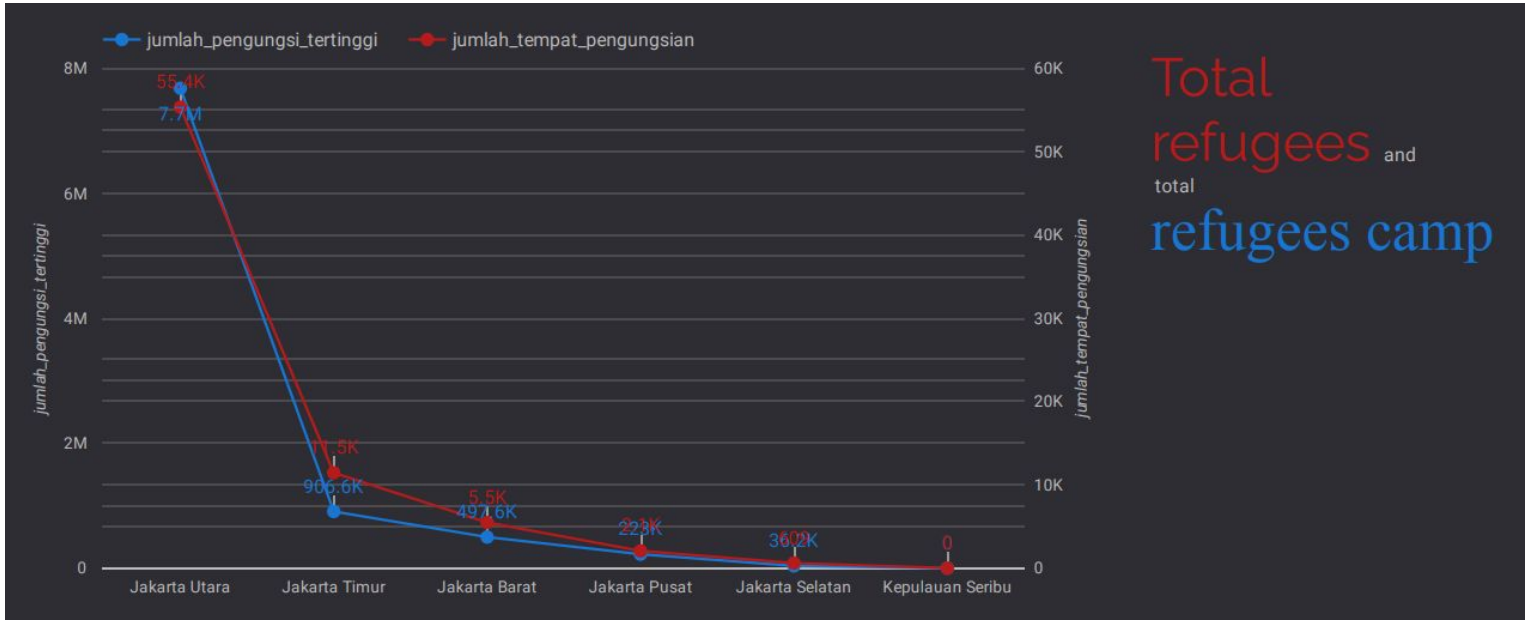


East Jakarta is the area with the highest flood level, while South Jakarta is the area with the highest average flood height

Data Understanding

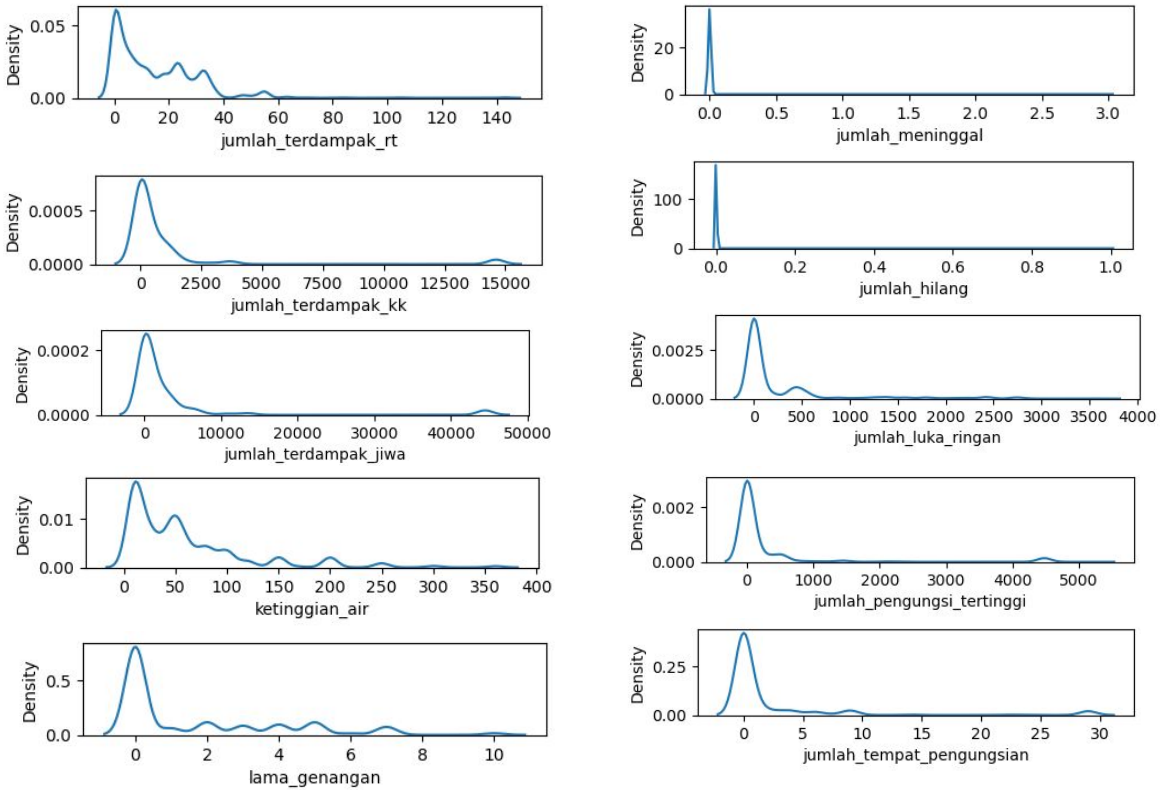
Link Dashboard

https://bit.ly/AdAstraExplorers



The worst comparison between refugees and refugee camps is occupied by North Jakarta with a comparison of 134 people per refugee camp, followed by Central Jakarta with a total of 107 refugees per refugee camp.

Exploratory Data Analysis



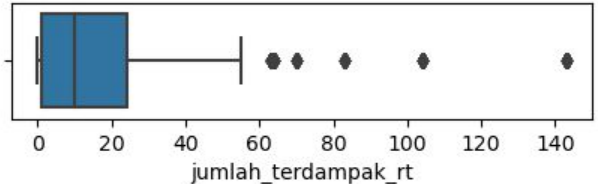
The numerical features displayed on the graph on the left have a **"left skew"** characteristic, indicating that the tail of the distribution is to the left of the central value, and lower values have a higher frequency than higher values.

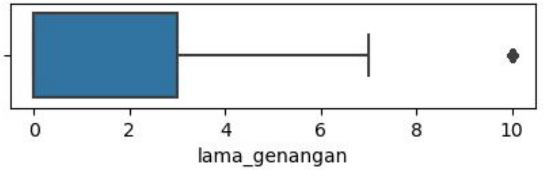
This indicates that there is **accumulation of data at a lower value** and cases **with high-value data are rare**. This data type has the **potential to have many outlier values.**

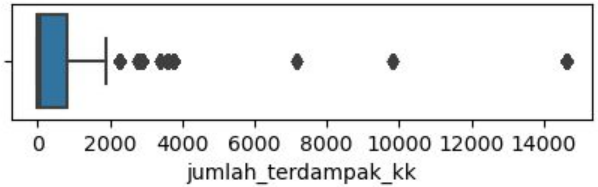
For more details open jupyter notebook here

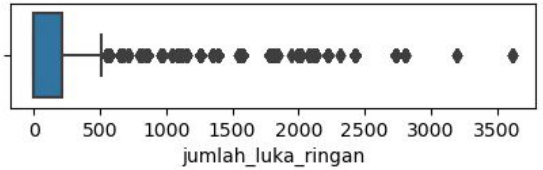
Exploratory Data Analysis

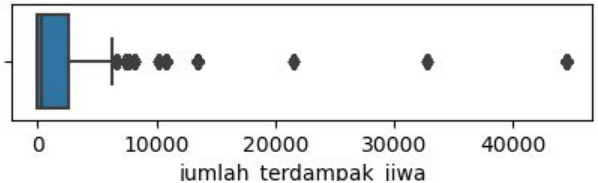
New array dataset :

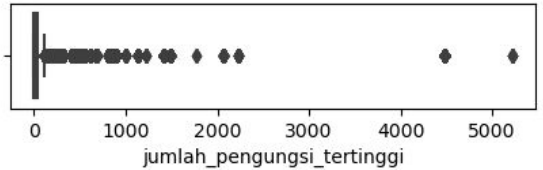


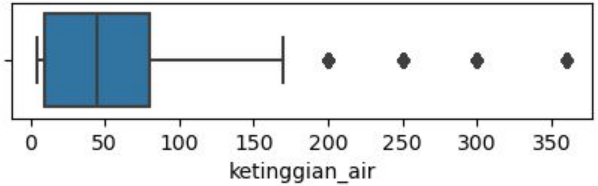


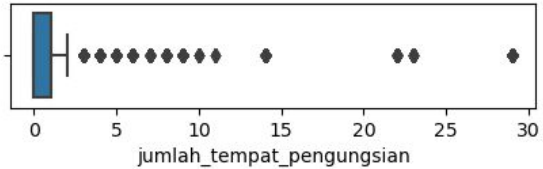










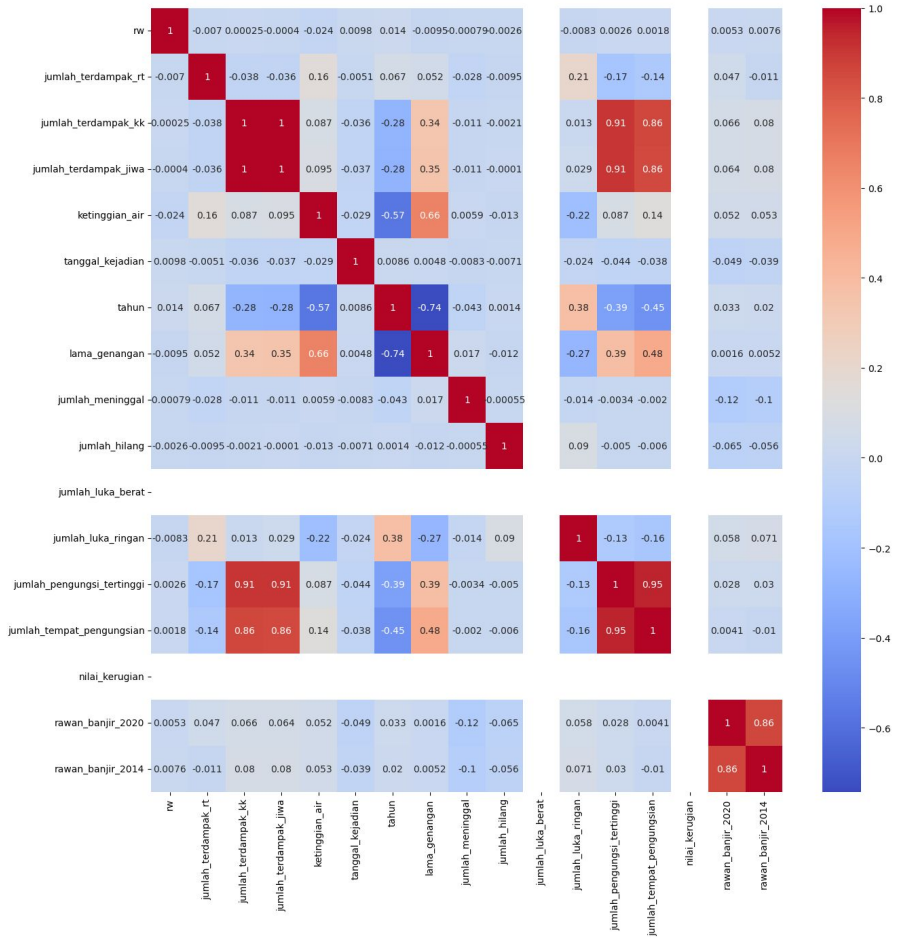


The graph beside is a boxplot graph of the numerical features of the dataset, it can be seen that there are **many outlier values that occur**, this should be **considered to drop the outlier values or keep using them to maintain data interpretation.**

For more details open jupyter notebook here

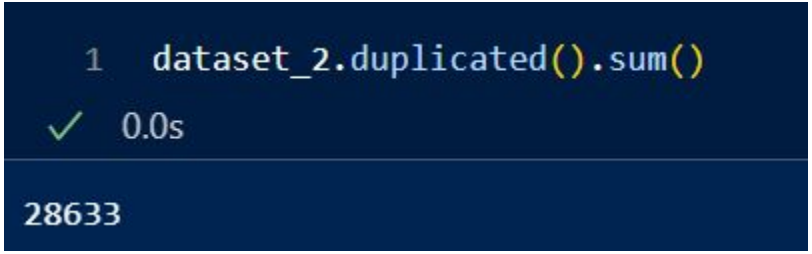
Exploratory Data Analysis

Based on the correlation chart, there are **several features that are redundant**, because they have a correlation score above 0.7. However, based on an analysis of understanding the data, **it is advisable to continue to use the dataset so as not to reduce the interpretation of the conditions of each flood-affected area** considering that the conditions in each region are different.



For more details open jupyter notebook here

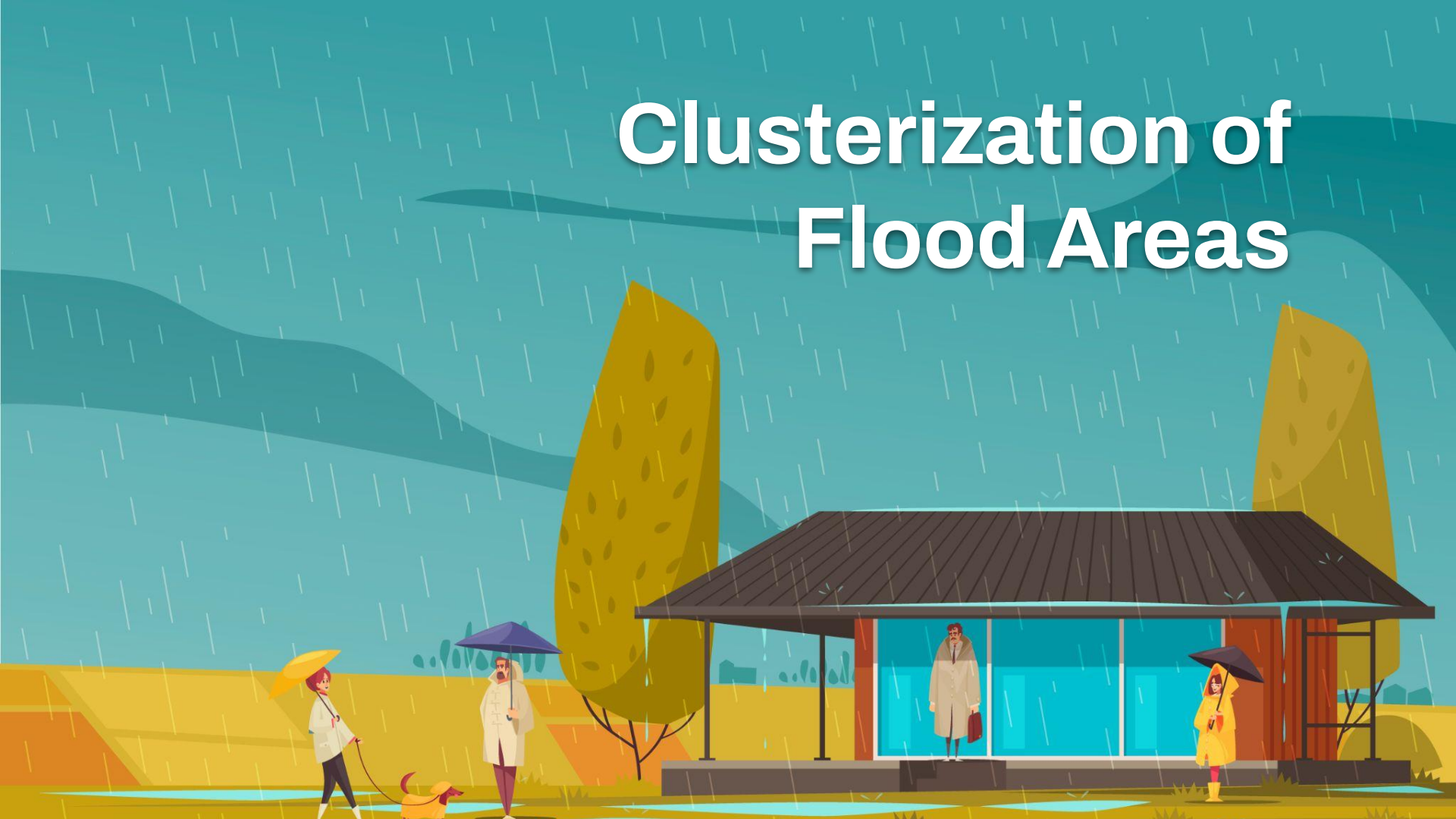
Exploratory Data Analysis



The dataset has 28633 duplicate rows, it is considered to drop duplicate data to avoid redundancy and impact on model quality.

For more details open jupyter notebook here

Clusterization of Flood Areas



Clustering : Data Preparation

```
clustering = dataset_2.drop_duplicates()

cat = dataset_2.select_dtypes(include='object')
```

Drop duplicated data to **prevent redundancy and reduce model performance**, then group data into categories to make encoding easier.

Performs encoder labels to **convert categorical variables into numerical representations**. This is useful in data processing and statistical modeling.

```
label_encoder = LabelEncoder()
for col in cat:
    clustering[col] = label_encoder.fit_transform(clustering[col])
    label_names = label_encoder.classes_  # Mendapatkan nama label yang diubah
    print(f"Nama label yang diubah pada kolom {col}:")
    print(label_names)
    print()
```

```
month_labeling = {'Januari':1, 'Februari':2, 'Maret':3, 'April':4, 'Mei':5, 'Juni':6, 'Juli':7,
                  'Agustus':8, 'September':9, 'Oktober':10, 'November':11, 'Desember':12}

clustering['bulan'] = clustering['bulan'].replace(month_labeling)
```

Labeling the months to clarify time series sequences numerically for data processing and modeling

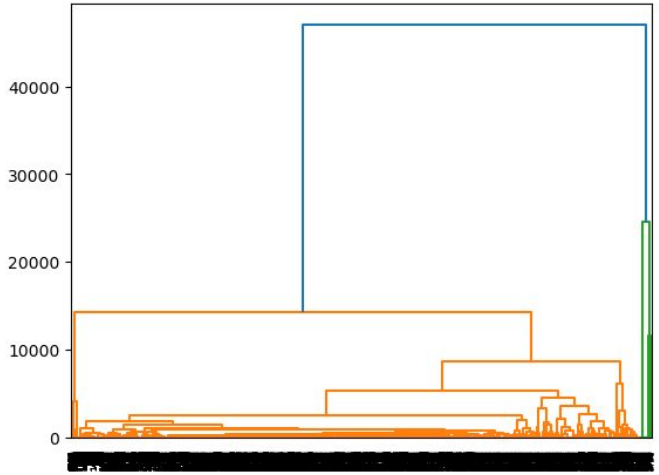
For more details open jupyter notebook here

Clustering : Feature Enginering

```
pcs = PCA(n_components=5).fit_transform(clustering)
pcs = pd.DataFrame(pcs, columns=['PC1', 'PC2', 'PC3', 'PC4', 'PC5'])
pcs
```

Reducing the number of features to simplify and lighten **modeling algorithms using PCA and avoid the "curse of dimensionality"**. Even though PCA is sensitive to **different scales for each feature and sensitive to outliers**, with the consideration of wanting to add weight to the impact of floods and flood heights, it was decided **not to transform the data.**

Dendrogram :




```
from scipy.cluster.hierarchy import dendrogram, linkage

linked = linkage(pcs, 'complete')

dendrogram(linked)
plt.show()
```

The dendrogram is a tree structure that visualizes the cluster hierarchies that are formed. **Dendrograms enable better interpretation of the relationships between clusters and objects in the data.**

Based on the dendrogram, we decided to use **three clustering numbers** because they are considered to have **a relationship between clusters that are quite far apart and easy to interpret.**

For more details open jupyter notebook here

Clustering : Modeling & Model Evaluation

The selected model is a model that **has resistance to outlier values** and the evaluation metric used is the **Silhouette Score**

Hyperparameter Tuning Clusterization Models	Silhouette Score
Mean Shift	0,90
Gaussian Mixture	0,79
DBSCAN	0,87
Agglomerative Clustering	**0,92**

If the Silhouette Score is close to 1, it indicates that the cluster is properly configured, with objects in clusters that are close together and clusters that are well separated. Based on the Silhouette evaluation metric, the result that has the best value is the result of **Hyperparameter Tuning Agglomerative Clustering**.

For more details open jupyter notebook here

Future Flood Forecasting

Forecasting

DATA PREPARATION

```
[ ]  # Load the dataset
     data = pd.read_csv('ejaksel.csv')
```

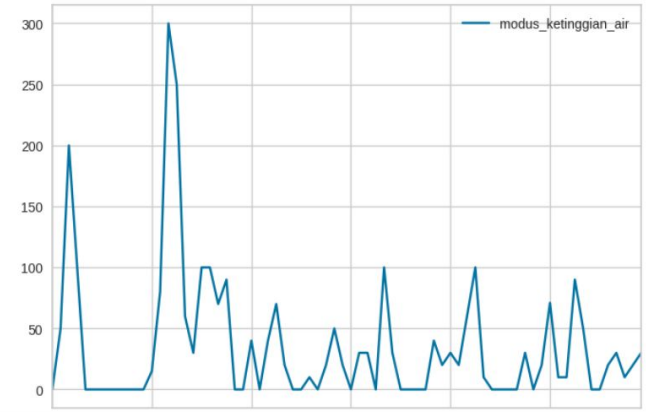
1 to 72 of 72 entries Filter

date	modus_ketinggian_air
2015-01-01	0
2015-02-01	50
2015-03-01	200
2015-04-01	100
2015-05-01	0
2015-06-01	0
2015-07-01	0
2015-08-01	0
2015-09-01	0
2015-10-01	0
2015-11-01	0
2015-12-01	0
2016-01-01	15
2016-02-01	80
2016-03-01	300
2016-04-01	250
2016-05-01	60
2016-06-01	30
2016-07-01	100
2016-08-01	100
2016-09-01	70
2016-10-01	90
2016-11-01	0
2016-12-01	0
2017-01-01	40

We take mode data based on the time of the flood event for each administrative city of Jakarta

```
data.plot()
```

```
<Axes: xlabel='date'>
```



Plotting the time series of flood events from 2015-2016

For more details open jupyter notebook here

Forecasting

FEATURE ENGINEERING

- Changing data type of the date column

```
data['date'] = pd.to_datetime(data['date'])
```

- Index adjustment

```
data.set_index('date', drop=True, inplace=True)
```

- Data duplicate cleaning

```
data = data.loc[~data.index.duplicated(keep='first')]
```

For more details open jupyter notebook here

Modeling & Hyperparameter Tuning For Forecasting

```
# import pycaret time series and init setup
from pycaret.time_series import *
s = setup(data, fh=3, session_id=123)
```

This part of the code sets up the time series forecasting experiment using the PyCaret library, including dataset preparation and configuration of the forecasting horizon.

```
# import TSForecastingExperiment and init the class
from pycaret.time_series import TSForecastingExperiment
exp = TSForecastingExperiment()
```

This part of the code imports and initializes the TSForecastingExperiment class from the pycaret.time_series module, enabling the use of time series forecasting experiment features.

For more details open jupyter notebook here

```
# check the type of exp
type(exp)

# init setup on exp
exp.setup(data, fh=3, session_id=123)

# check statistical tests on original data
check_stats()

# compare baseline models
best = compare_models()

# compare models using OOP
exp.compare_models()
```

The code snippet consists of function calls and variable assignments that perform operations related to setting up a time series forecasting experiment, checking statistical tests, and comparing models using the pycaret.time_series library in Python.

```
# train a dt model with default params
dt = create_model('rf_cds_dt')

# tune hyperparameters of dt
tuned_dt = tune_model(dt)

# define tuning grid
dt_grid = {'regressor__max_depth': [None, 2, 4, 6, 8, 10, 12]}

# tune model with custom grid and metric = MAE
tuned_dt = tune_model(dt, custom_grid=dt_grid, optimize='MAE')

# see tuned_dt params
tuned_dt

# to access the tuner object you can set return_tuner = True
tuned_dt, tuner = tune_model(dt, return_tuner=True)

# model object
tuned_dt

# tuner object
tuner
```

This code performs model training and hyperparameter tuning for a decision tree model using the pycaret library in Python..

Tuning Model

Administrative City Areas	Base Model	Tuning Model
North Jakarta	ThetaForecaster	RandomForestRegressor
West Jakarta	Croston	RandomForestRegressor
Central Jakarta	NaiveForecaster	ExtraTreesRegressor
South Jakarta	BaseCdsDtForecaster	ExtraTreesRegressor
East Jakarta	RandomForestRegressor	RandomForestRegressor

For more details open jupyter notebook here

Model Evaluation

	MAE	RMSE	MASE
RandomForestRegressor	4.8718	5.6333	0.6030
ThetaForecaster	3.6825	4.5863	0.4521
NaiveForecaster	9.0000	13.7988	2.6550
ExtraTreesRegressor	11.4922	15.8997	1.3609
Croston	13.3512	17.8775	0.8286
BaseCdsDtForecaster	23.6970	29.7603	0.6551

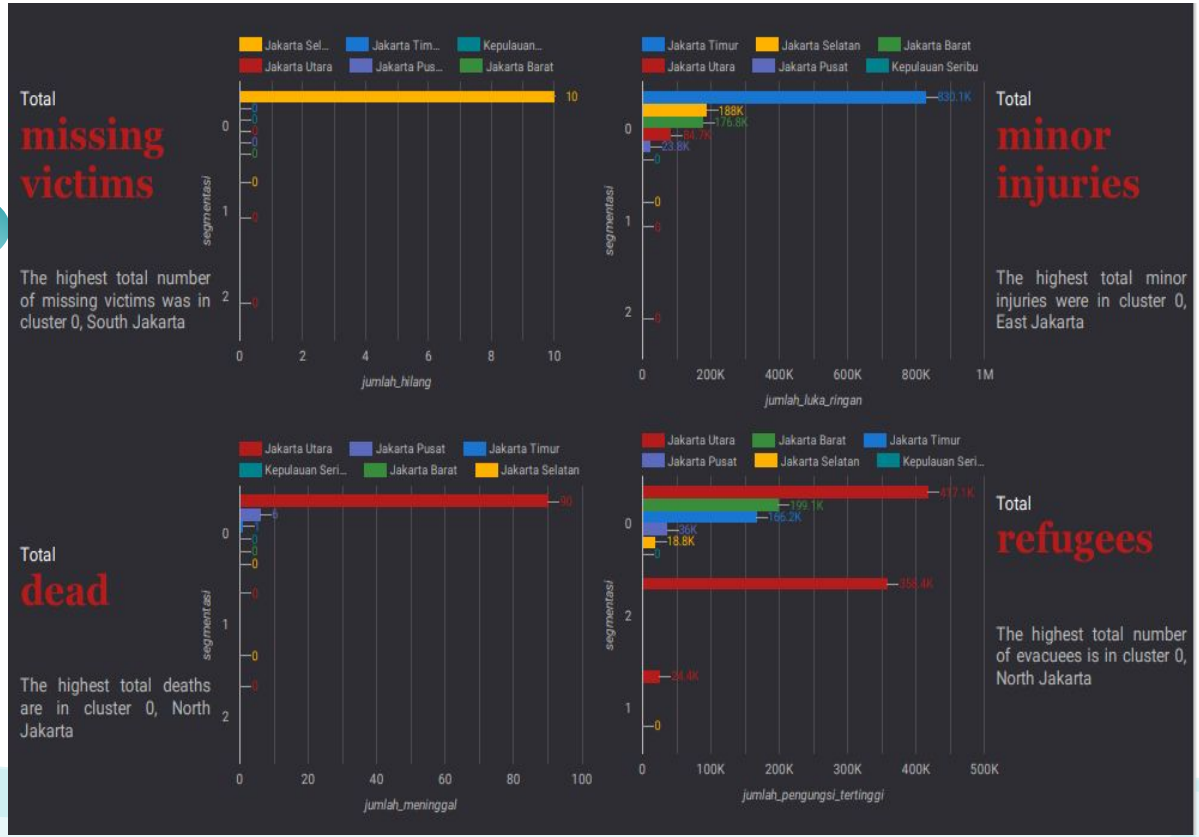
For more details open jupyter notebook here

Testing the Clustering Model and Forecasting Model

Model Testing

Link Dashboard

https://bit.ly/AdAstraExplorers



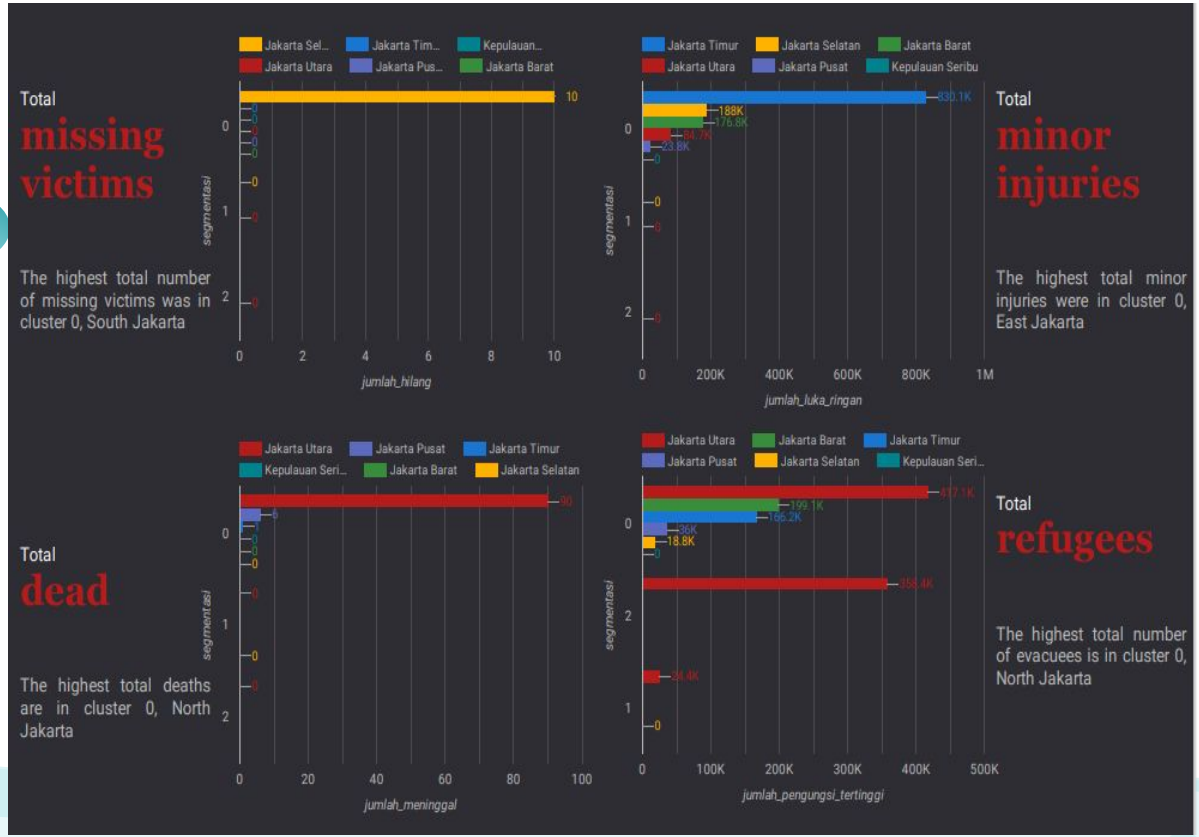
Segmentasi

The highest total missing victims, minor injuries, deaths, and refugees were in cluster 0 with a total of 10 people missing in South Jakarta, 830,000 minor injuries in East Jakarta, 90 deaths in North Jakarta and a total of 417,000 refugees in North Jakarta.

Model Testing

Link Dashboard

https://bit.ly/AdAstraExplorers



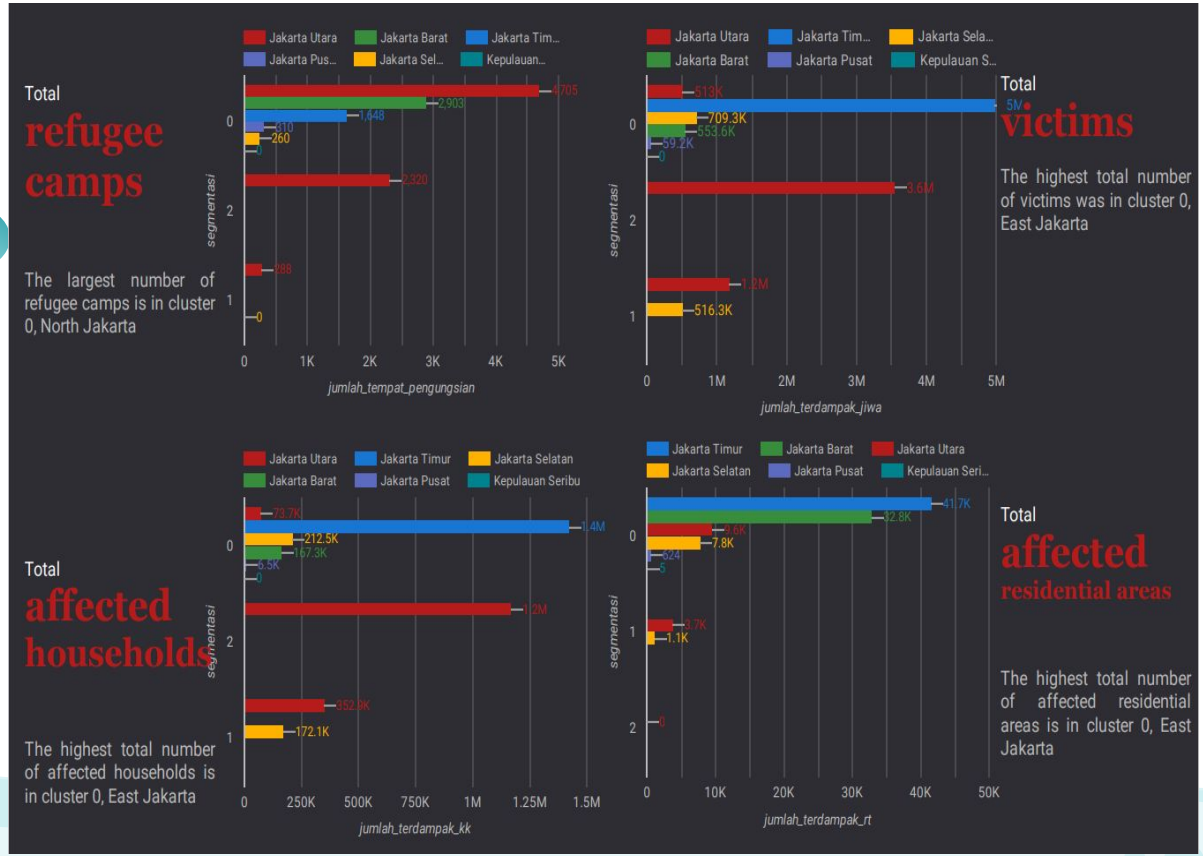
Segmentasi

The highest total missing victims, minor injuries, deaths, and refugees were in cluster 0 with a total of 10 people missing in South Jakarta, 830,000 minor injuries in East Jakarta, 90 deaths in North Jakarta and a total of 417,000 refugees in North Jakarta.

Model Testing

Link Dashboard

https://bit.ly/AdAstraExplorers



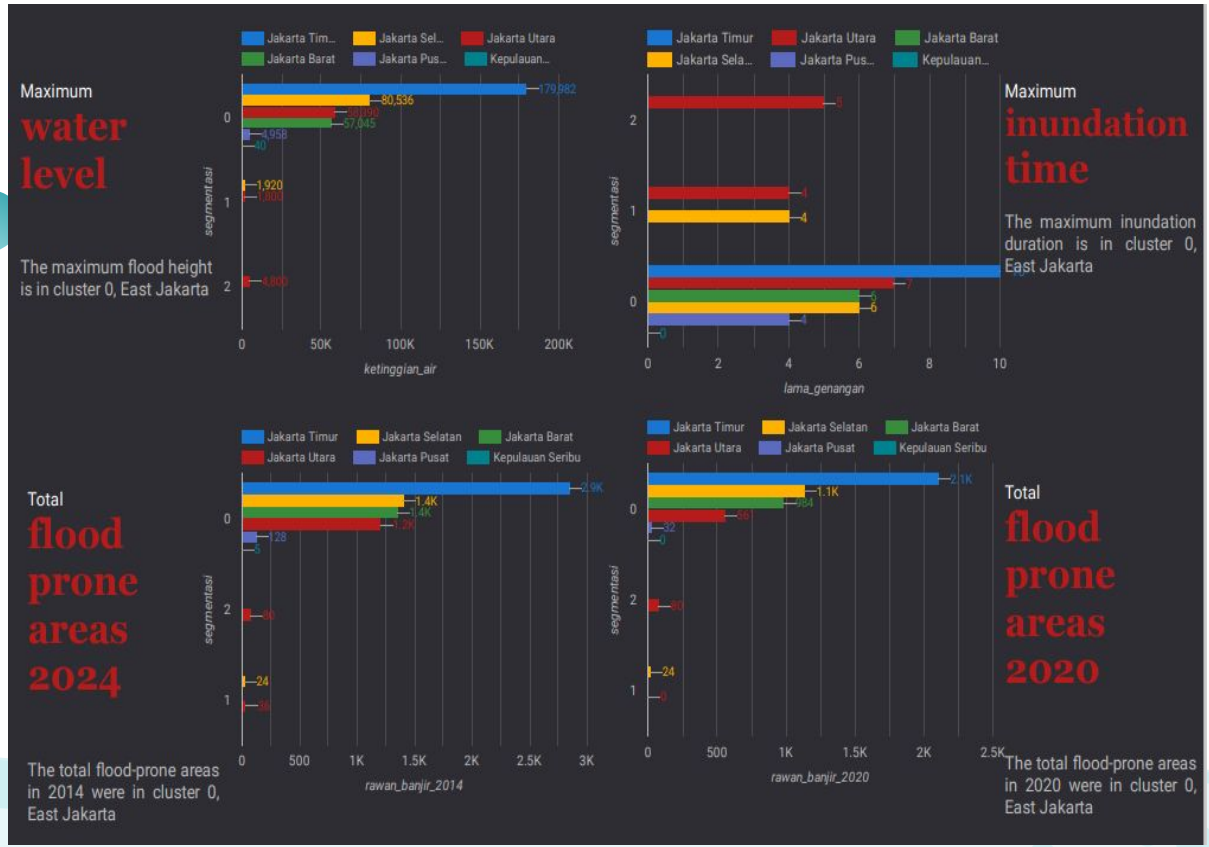
Segmentasi

The largest number of evacuees, victims, affected households, and affected households were in cluster 0 with a total of 4,000 evacuees in North Jakarta, 5 million victims in East Jakarta, 1.4 million affected households in East Jakarta and 41 thousand affected residential areas in East Jakarta

Model Testing

Link Dashboard

https://bit.ly/AdAstraExplorers



Segmentasi

Maximum water level, length of inundation, total flood-prone areas in 2024, and total flood-prone areas in 2020 are in cluster 0 with a maximum water level of 1.8 m in East Jakarta, 10 days of inundation in East Jakarta, 2.9 thousand flood-prone areas in 2024 in East Jakarta and 2.1 thousand flood-prone areas in 2020 in East Jakarta

Summary Segmentasi

Link Dashboard

https://bit.ly/AdAstraExplorers

Clusters 0
Has the condition of the highest water level and the longest inundation compared to other clusters and is a flood-prone area but the affected community is not too large compared to clusters 2 and 1. This indicates that **the area of cluster 0 is not as large as clusters 1 and 2. It has a high number of victims because alarming flood levels.**

Clusters 1
Including areas that are not flooded too often, but still have the potential for flooding. Cluster 1 is an area that has flood conditions not as worrying as cluster 0 and cluster 2.

Clusters 2
It has a fairly high water level and is a flood-prone area and has a high population density in each area. **This needs to be considered to anticipate the evacuation of residents during a flood because the area is very dense.**

Model Testing

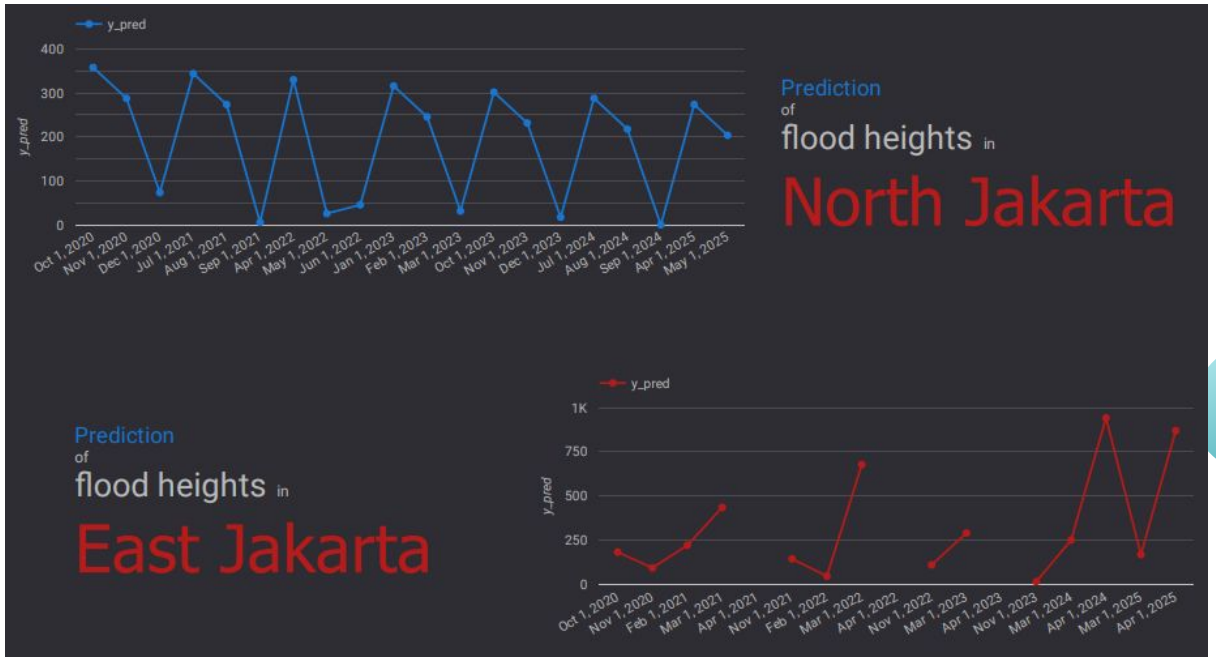
Link Dashboard

https://bit.ly/AdAstraExplorers

Forecasting

In North Jakarta there are several times with a flood height of around 300 cm, namely on 1 October 2020, 1 July 2021, 1 April 2022, 1 January 2023, 1 October 2023, 1 July 2024 and 1 April 2025

For East Jakarta there are several dates with flood heights above 400 cm, namely March 1 2021, March 1 2022, April 1 2024, April 1 2025



Model Testing

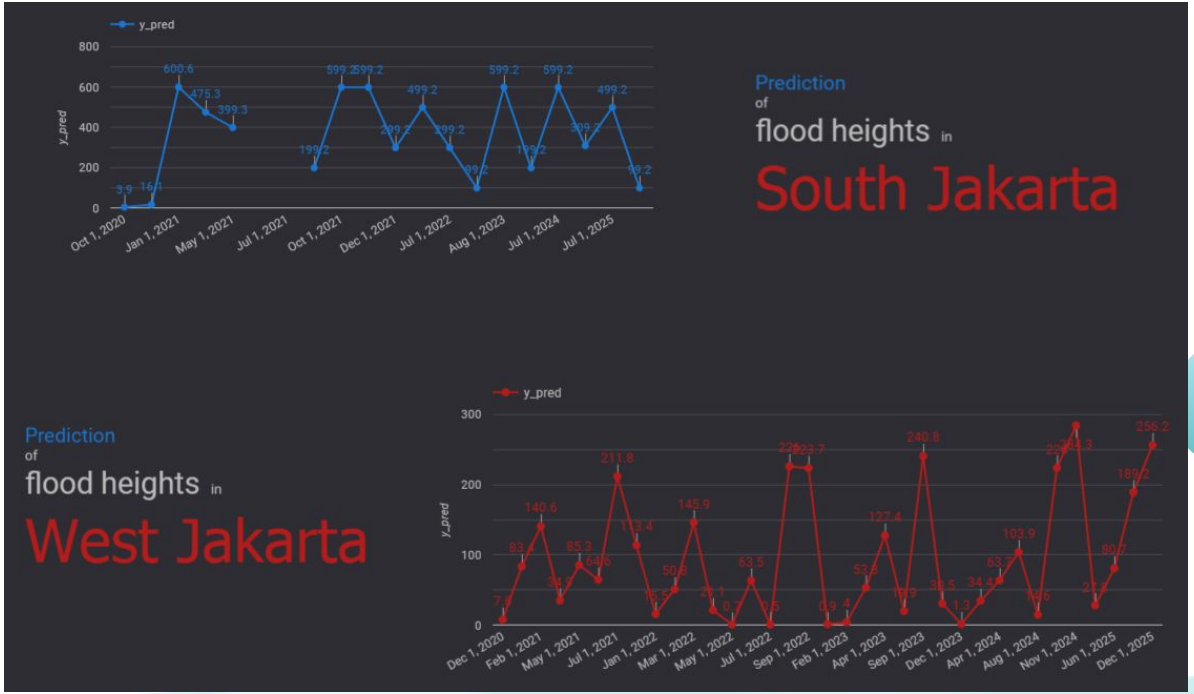
Link Dashboard

https://bit.ly/AdAstraExplorers

Forecasting

In South Jakarta, there are several times when floods are overcome by 500 cm, including January 1 2021, October 1 2021, November 1 2021, August 1 2023 and July 1 2024

Whereas for West Jakarta the flood height reached above 200 cm, including July 1 2021, August 1 2022, November 1 2022, September 1 2023, October 1 2024, November 1 2024, December 1 2025



RECOMMENDATION

Based on Analysis and Modeling :

- The government pays more attention to the spatial conditions of the city, especially the enlargement/widening of drainage during the **rainy season in clusters 0 & 2**, and pays attention to the regional characteristics of each cluster both **for flood prevention and victim evacuation strategies** in the event of a flood.
- The government pays attention to the conditions of the refugee camps against the number of refugees to maintain the conduciveness of the refugee camps.
- Based on data analysis and forecasting, all parties **are preparing for March to May and August to November** which are predicted to be the peak flood heights.
- Conducting river **normalization during the transition from the dry season to the rainy** season to anticipate flooding from August to November.
- Completing flood disaster data every time to make it easier to take insights and make decisions based on data analysis.

General :

- Collaborating with stakeholders for flood prevention for each region.
- The use of AI for real-time and accurate monitoring of each river and embankment to provide flood early warning.
- Flood insurance as compensation for individuals or community groups to ease the government's burden.


Thank you
"Floods remind us of human vulnerability to climate change and the need for action to protect the environment."