III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. **Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.**

- I am a student pursuing an undergraduate in Electrical Engineering at the Indian Institute of Technology, BHU, Varanasi, India. My interests lie in Data Science and Machine Learning.

2. **What motivated you to compete in this challenge?**

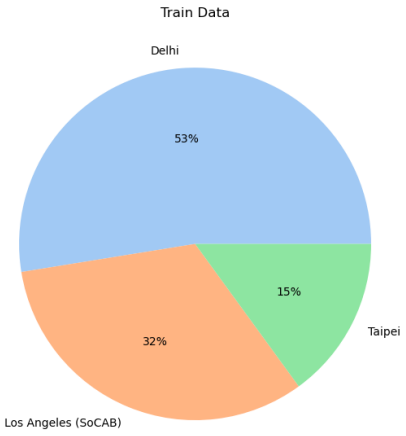
- For the past few months, I have been participating in competitions hosted on DrivenData. The last competition I participated in was related to cloud segmentation, this is when I developed an interest in working with remote sensing data. And these types of competitions are trying to solve certain problems around our environment and nature. So, contributing to this noble cause is one of the motivations.

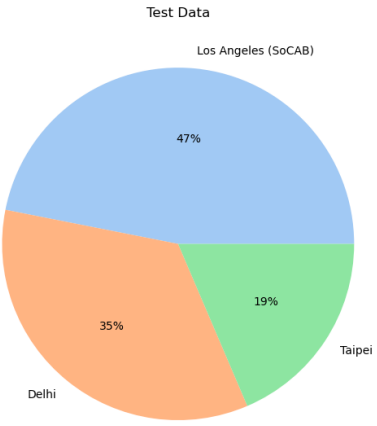
3. **High level summary of your approach: what did you do and why?**

- Firstly, raw data is processed, and then we impute the data using grid wise mean imputation method. After that, we generate temporal difference features and load metadata, and then train two pipelines. Checkout summary in the README.md file for more details.

4. **Do you have any useful charts, graphs, or visualizations from the process?**

- Following pie charts show amount of data from each location.





5. **Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.**

- Following code shows grid wise mean imputation. Here train_df, test_df contains features from all data products. This imputation worked better than just mean imputation.

```
feat_columns = [col for col in train_df.columns if col != 'grid_id']
for grid_id in train_metadata['grid_id'].unique():
    for col in feat_columns:
        indices = train_df[train_df['grid_id'] == grid_id].index
        mean_val = train_df.loc[indices, col].mean()
        train_df.loc[indices, col] = train_df.loc[indices, col].fillna(mean_val)

        indices = test_df[test_df['grid_id'] == grid_id].index
        test_df.loc[indices, col] = test_df.loc[indices, col].fillna(mean_val)
```

- Using temporal difference features also helped.

```
# Let's say today and yesterday contains features for predicting today's
# yesterday's concentration respectively.
# NOTE: today and yesterday must belong to same grid cell

for col in feat_columns:
    today[col + '_temporal_diff'] = today[col] - yesterday[col]
```

- Using `mean_value` feature helped bump the leaderboard score.

```
# Grid wise mean value is calculated in following way
# train_metadata: train_labels.csv file
# test_metadata: submission_format.csv file
train_df['mean_value'] = train_metadata['grid_id'].apply(
    lambda x: train_metadata[train_metadata['grid_id'] == x]['value'].mean()
)
test_df['mean_value'] = test_metadata['grid_id'].apply(
    lambda x: train_metadata[train_metadata['grid_id'] == x]['value'].mean()
)
```

6. **Please provide the machine specs and time you used to run your model.**
   - CPU (model): Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz
   - GPU (model or N/A): N/A
   - Memory (GB): 8GB
   - OS: Windows 11 Home Single Language 64-bit
   - Train duration: 26 mins (without including data downloading and pre-processing)
   - Inference duration: Around 30 mins to predict one day's concentrations for all the grids in three cities (Including data downloading and pre-processing)
   - Time taken to process MAIAC data for train and test instances: Around 17 hours
   - Time taken to process MISR data for train and test instances: Around 2 hours
   - Time taken to process GFS data for train and test instances: Around 10 hours
   - Time taken to process NASADEM data for train and test instances: Around 1 hour

7. **Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?**
- Make sure to use correct datetime format while running predict.py file.

8. **Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?**
- No

9. **How did you evaluate performance of the model other than the provided metric, if at all?**
- I used both R^2 and RMSE to evaluate performance of the models.

10. **What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?**
- Tried grid wise temporal imputation but didn't work as expected.
- Tried to train convolutional neural networks location wise but didn't work as good as current approach.
- Implemented features selection but it didn't make it into final workflow.

11. **If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?**
- I would try TabNet, which is an interpretable canonical deep tabular deep learning architecture.
- I would try to include features selection process into pipeline.
- I would also include more ancillary data products.