Model Documentation and Write-up

1. Who are you (mini-bio) and what do you do professionally?

   I am Ammar Ali, a second year master student at ITMO University, Russia. I am studying at programming and computer technologies faculty. My precise specialty is Business Information systems. As well, I am a computer vision engineer working on ITMO University in three different projects: (Drive Safely) a project to monitor the driver behavior in car cabin including (behavior detection and head pose estimation). A recommendation system for enhancing the meditation process and the last project about extracting information from 2D Solidworks drawings to automate 2D to 3D conversion in the future.

2. What motivated you to compete in this challenge?

   I started working with competitions on Kaggle 3 months ago and I liked it. When I found this competition on Driven Data. I was also interested so I decided to participate. Time series is somehow a new field for me, I really wanted to learn, and I did. I learned a lot from this competition. Thanks for you and For Driven Data for this amazing competition hope to see more soon.

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

   I started reading the baseline solution and how to submit a solution at first to completely understand the problem and what I am dealing with. After that I started the enhancement of the baseline solution according to the recommendations were given in the baseline and my expertise. The major changes I have made are:

- Change the imputation of the missing values to the most frequent strategy.
- Change the testing and Validation Data from the tail of the data to the head (because it will not be time connected in this case and it will give more realistic loss values)
- Build a new model architecture according to my expertise and experiments
- Take more features from the data.
- Use a dummy model to manipulate the starting point (initialized weights) of my model.

   More details about how I built the solution and why I chose this architecture and more will be attached on another file “Magnet Solution Details”.

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

   Implementation Details:

- Model Architecture

```
# define our model
model_config = {"n_epochs": 100, "n_neurons": 192*2, "dropout": 0.0, "stateful": False}

input1 = Input(shape=( data_config["timesteps"], len(XCOLS)), name='x1')
lstm1= Bidirectional(LSTM(
        model_config["n_neurons"],
        # usually set to (`batch_size`, `sequence_length`, `n_features`)
        # setting the batch size to None allows for variable length batches
        #activation = 'gelu',
        #recurrent_activation = 'swish',
        stateful=model_config["stateful"],
        dropout=model_config["dropout"],return_sequences=True
    ))(input1)
gru1= Bidirectional(GRU(
         model_config["n_neurons"] *3,
        # usually set to (`batch_size`, `sequence_length`, `n_features`)
        # setting the batch size to None allows for variable length batches

        stateful=model_config["stateful"],
        dropout=0.0,return_sequences=True
    ))(lstm1)

gaverage = Flatten() (gru1)
dense1 = Dense(96)(gaverage)
dense1 = Dense(128)(dense1)
dense1 = Dense(64)(dense1)
dense = Dense(2)(dense1)


model = keras.models.Model(inputs=input1, outputs=dense)
model.compile(
    loss='mean_squared_error',
    optimizer=tf.keras.optimizers.Adam(0.0001),
)
```

As you can see My best model consists of a Bidirectional LSTM layer connected to a Bidirectional GRU then 3 dense layers: the combination of Bi_LSTM and Bi_GRU is a well known one. And it almost always will show better performance than just a single LSTM layer. The number of neurons specified according to the following criteria.

I am using 29 features as an input with 128 timestamps . the first power of two that is bigger than 29 is 32 so I multiplied this number by two and sum it to 128 will give 192 which is the base number of neurons of my LSTM and GRU. GRU number of neurons then multiplied by 3 because it showed better performance according to the experiments and then the number of neurons

for LSTM and GRU also multiplied by 2 on My final model. First Dense layer has 192/2 neurons then 128 and 64 to prevent the overfitting. Dropout was zero for the best performance.

- Missing Values Imputation

```
if imp == None:
  #estimator =  BayesianRidge()
  imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
  imp.fit(feature_df[cols])

#feature_df = feature_df.interpolate(method='spline',order = 5)
temp = imp.transform(feature_df[cols])
```

Different imputation techniques were tried and a simple imputer with most-frequent strategy showed the best response.

- More features

```
SOLAR_WIND_FEATURES = [
    "bt",
    "temperature",
    "bx_gse",
    "by_gse",
    "bz_gse",
    "phi_gse",
    "theta_gse",
    "bx_gsm",
    "by_gsm",
    "bz_gsm",
    "phi_gsm",
    "theta_gsm",
    "speed",
    "density",
]
```

Using more features even if they were highly correlated showed a better performance. I believe that we should provide the NN all features we have and let it decide what to use and what to ignore as long as we still on the time limit and other restrictions.

5. Please provide the machine specs and time you used to run your model.

I trained on Google Colab pro all on CPU but the specification will be related to the time of usage. The back OS is Linux. Training duration is about 243s for an epoch and it needs 27 epochs to converge. Which means about two hours for training. Inference submission took about 5 hours to finish testing on your devices.

6. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

   I tried many operations some of them enhanced my validation but did not reflect on the leader board. For example, using a combination of standard scaler and power transformer enhanced my validation and testing but it had worse results on the leaderboard.

   Simple augmentations also didn't work. Scaling using MinMax scaler, Robust Scaler and Normalizer also didn't work. I thought that I should normalize the features between 0 and 1 since I am using an LSTM with tanh activations but unfortunately it didn't.

   Imputations (iterative imputer with Bayesian Ridge estimation and most-frequent initial strategy) also thought that it should work better but it didn't as well for other imputation techniques.

7. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

   No I didn't all my work is on a Colab notebook and you can regenerate the results by just running it.

8. How did you evaluate performance of the model other than the provided metric, if at all?

   Using MSE loss function as it was on the baseline I wanted to try other metrics but the limitation of submissions number was not helpful for that.

9. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

   Just make sure that the data are accessible by the notebook for me I uploaded the data to drive and the notebook will have access from there. To run on your devices you just have to make sure that the data are accessible and the libraries versions are identical with Colab libraries versions.

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

    Well, The same for baseline you will see them when you run the notebook.

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?

    Yes for Enhancement:

- Design a GANs model to generate new data (it will be better to generate data that is similar to the data, which has huge dst absolute values because it was considered as outliers). Then train the model on the generated data. After that use, the weights as a starting point for the training (transfer learning).

- Data augmentation Also an important factor to enhance the model and minimize the loss.
- Use of satellite_positions data not as features. However, to build a model for missing values or to build the GANs model or a multi head model.
- If it showed that somehow Gaussian distribution is related to the problem then power transformer should be used (I have a strong feeling that it should be used).