README

Deliverables

A git repository at a provider of your choice that includes:

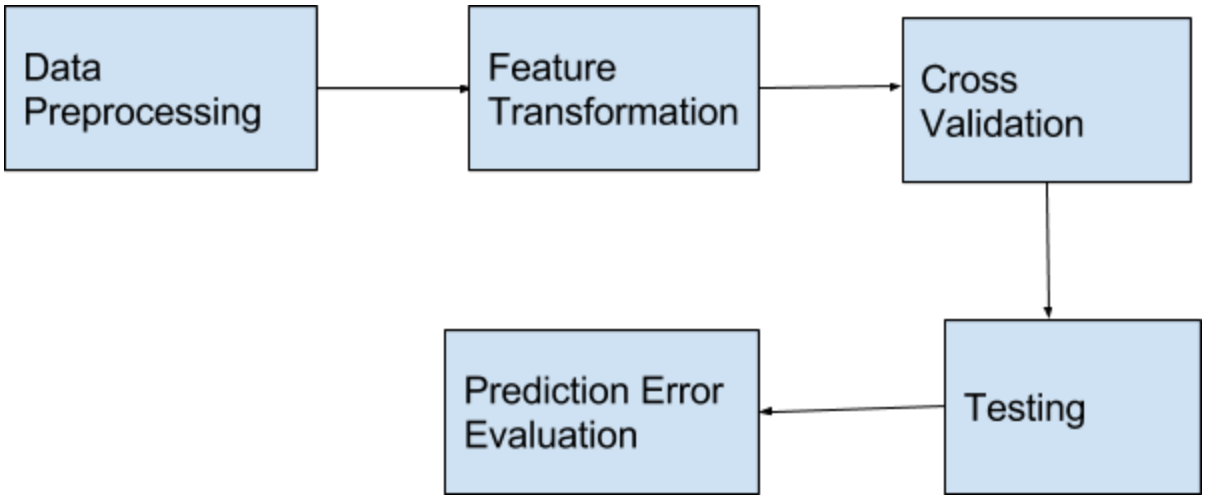
- Source code for data tranformation and any feature engineering you did.
- Readme explaining your approach, what you did, what you tried, why those choices.
- Short overview of the results.

1) Theoretical considerations for model/parameter selection
2) Explanations why these types of feature engineering

The problem provided is a regression problem since the the remaining useful life of an aircraft engine is continuous, accuracy on any set would not be a meaningful parameter as it has its classical relevance in a classification problem.

Working Pipeline:

The following is the working pipeline adopted to solve the given problem:



MODEL VALIDATION: Assessing the Goodness of the model

We know that since the validation_split parameter of model_fit of Keras does not involve bootstrapping or random sampling, only this cannot be used as a basis to decide about the goodness of a model.
One can use R-squared as its high value shows high correlation of the features to the output variable. Unfortunately, since the coefficient of determination is inherently based on a linear

model, R-squared is not robust to outliers and when the model complexity is high, it R-squared does tend to 1 since it is nondecreasing with respect to increasing model complexity.

- Thus, one should definitely adhere to some sort of cross-validation error such as cross-validation RMSE or cross-validation MAE or MAPE.

  Watch out for overestimating the RUL since it seems to be lower on the testing data.

The training and testing data are given in roughly 60-40 proportions.

For more realistic and accurate estimate on the test engines, the test split is randomly split into 2 equal parts – cross validation and test set.

Why RNN and LSTM?

recurrent neural networks (RNN), a class of neural networks

that exploit the sequential nature of their input. Such inputs could be text, speech, time series, and

anything else where the occurrence of an element in the sequence is dependent on the elements that

appeared before it.

One thing to realize is that an LSTM is a drop-in replacement for a SimpleRNN cell, the only

difference is that LSTMs are resistant to the vanishing gradient problem. You can replace an RNN

cell in a network with an LSTM without worrying about any side effects. You should generally see

better results along with longer training times.

Other possible models:

Poisson Process or Gaussian Processes based regressor could also be used for regression.

Furthermore, Gaussian processes can be used in the Bayesian optimization of the LSTM network parameters.

Why these parameters?

-- Window Size -- the higher the window size, the

-- Training parameters:

Dropout -- the probability can be

Optimizer,

Activation Function -- RELU vs Linear vs Sigmoid. It is clear that we cannot have softmax, since we have only 1 output cell. Sigmoid is not chosen because sigmoid is used ful classification. RELU is not chosen because it gives only 0 output for negative feature values. Since we have performed minimax normalization and most features values are nonnegative (between 0 and 1), thus relu should perform similarily to linear activation function, but RELU would be a special case of the linear activation function: namely the one which passes the origin, thus we choose linear.

Number of Hidden layers

Activation Function

BAYESIAN OPTIMIZATION PART

https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

'*the optimal size of the hidden layer is usually between the size of the input and size of the output layers*'.

Bayesian optimization of the LSTM neural network was experimented using hyperopt's fmin, but this function execution turned out to be too slow for practical applications. As an example, just to test one random probability value of dropout:

```
Params testing:  {'dropout1': 0.2939556619766569}
Epoch 1/5
 - 281s - loss: 12709.6188
Epoch 2/5
 - 205s - loss: 7008.2583
```

```
In general it is noted that if a Mongo database can be used to execute the
computation in parallel, a search space can easily be defined in a fashion
as
```

```
space = {   'units1': hp.choice('units1', [32,64]),

            'units2': hp.choice('units2', [13,26]),


            'dropout1': hp.uniform('dropout1', .25,.75),
```

```
            'dropout2': hp.uniform('dropout2',  .25,.75),


            'batch_size' : hp.choice('batch_size', [32,64,128]),


            'epochs' :  5,

            'optimizer':
hp.choice('optimizer',['adadelta','adam','rmsprop']),

            'activation': hp.choice('activation',['linear','relu'])

        }
```

And by calling the fmin function on this search space should iterate through that space to find the best combination of hyperparameters

2), HYPERAS OPTIM was utilized. However, since hyperas optim uses fmin and fmin turned out to be extremely slow, this approach was neglected.

3) Bayesian Optimization can also be accomplished using Gaussian Processes

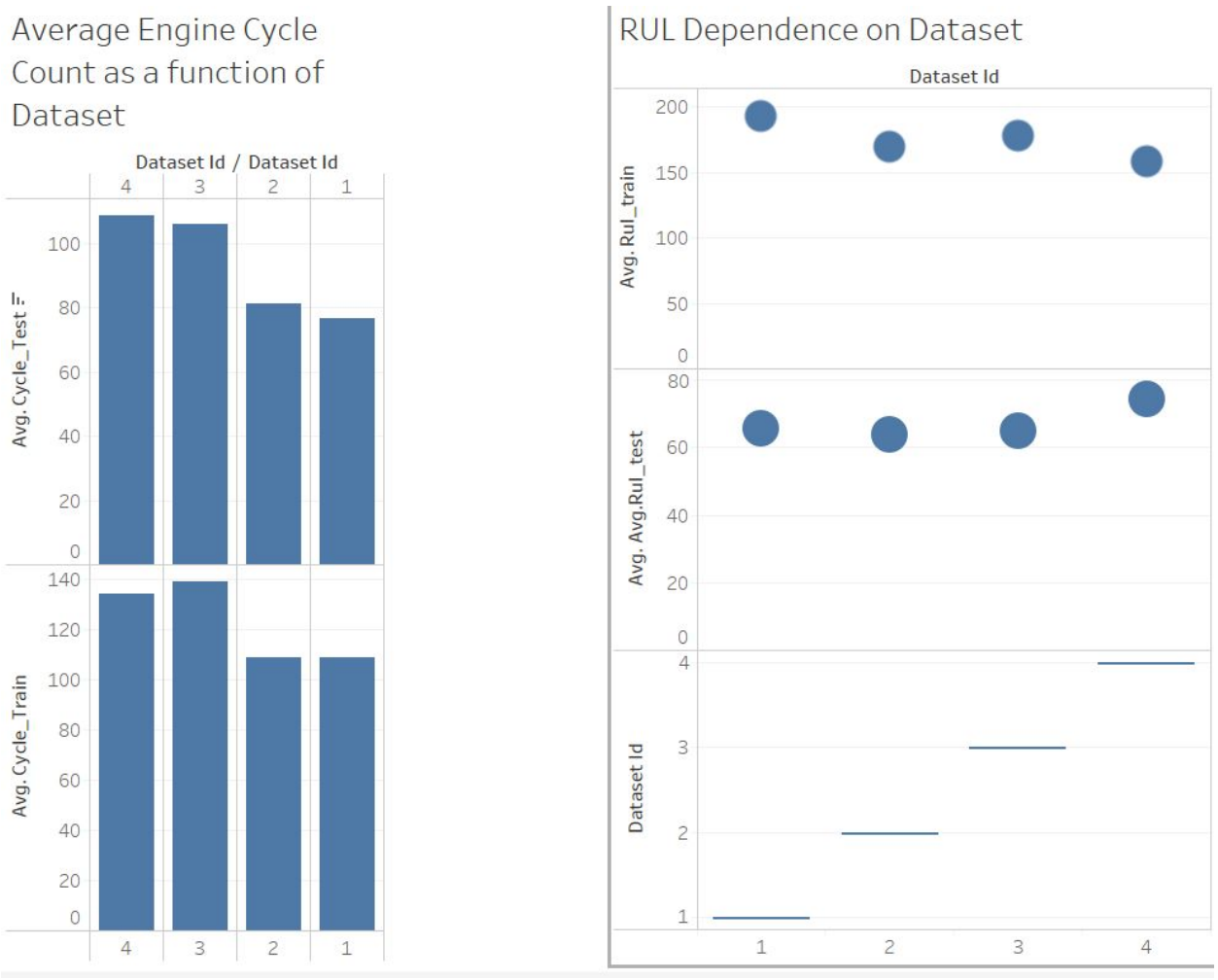
Since the first two obvious attempts at Bayesian optimization were too slow, manual optimization was opted for.

```
def base_minimizer(model, data, functions, algo, max_evals, trials,
                   rseed=1337, full_model_string=None, notebook_name=None,
                   verbose=True, stack=3):
    if full_model_string is not None:
        model_str = full_model_string
    else:
        model_str = get_hyperopt_model_string(model, data, functions, notebook_name, verbose, stack)
    temp_file = './temp_model.py'
    write_temp_files(model_str, temp_file)

    if 'temp_model' in sys.modules:
        del sys.modules["temp_model"]

    try:
        from temp_model import keras_fmin_fnct, get_space
    except:
        print("Unexpected error: {}".format(sys.exc_info()[0]))
        raise
    try:
        os.remove(temp_file)
        os.remove(temp_file + 'c')
    except OSError:
        pass

    try:
        # for backward compatibility.
        return (
            fmin(keras_fmin_fnct,
                 space=get_space(),
                 algo=algo,
```



1. 1. LEFT: Avg engine max cycle count. The cycle means are higher on the training sett, and higher in datasets 4 and 3. RIGHT: RUL mean is higher in the training set. This is because of full training runs. No major difference between datasets.