Interpretação do plano de manutenção e proposta de novo plano utilizando a solução de predição desenvolvida

Anteriormente, para garantir a manutenção preventiva de todos os assets, utilizava-se o valor de ciclo de falha mínimo conhecido do sistema (128). Ou seja, numa abordagem conservativa e totalmente preventiva, porém não econômica, realizava-se a manutenção de todos os assets a cada 128 ciclos. A perda econômica para esse cenário pode ser calculada como o número total de assets (100) vezes o erro absoluto médio (78) [slide "Our system"]; portanto, totalizando 7.800 ciclos desperdiçados.

Ao invés de utilizar o valor de ciclo de falha mínimo, é possível utilizar um valor maior, em função da média (206) e do desvio-padrão (46) conhecidos do sistema [slide "Our system"]. Porém, nesse caso, tolera-se parcialmente a manutenção corretiva dos assets. Por exemplo, considerando-se a distribuição Gaussiana, realizar a manutenção a cada 160 ciclos ao invés de 128 (média subtraída de um desvio-padrão), corresponderia a 16% de manutenção corretiva, com contrapartida da menor perda por manutenção precoce (< 7.800).

Conhecer antecipadamente o momento da falha de cada asset em função das respostas do sistema diminui tanto a perda econômica devido à manutenção precoce quanto a porcentagem de manutenção corretiva tolerada.

Por exemplo, assumindo que os erros obtidos pela solução proposta, que foram simulados com os dados de validação, se aplicam também aos dados de teste: 66 (erro máximo) e 18 (erro médio) [slide "Model analysis"], o cenário de manutenção totalmente preventiva seria o seguinte [SUZ-139],

1. a partir do valor de ciclo de falha mínimo conhecido do sistema (128) subtraído do valor de erro máximo (66), iniciaria-se a predição de falha do asset;

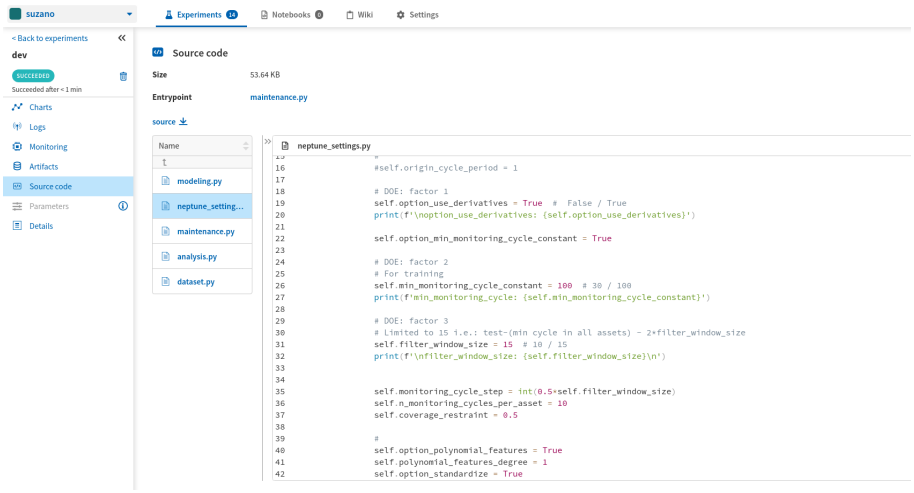
2. caso a predição da vida útil restante (rul) seja igual ou menor do que 66 (valor limite de decisão de manutenção), a manutenção deve ser realizada. Caso contrário, a predição deve se repetir no próximo ciclo de operação.

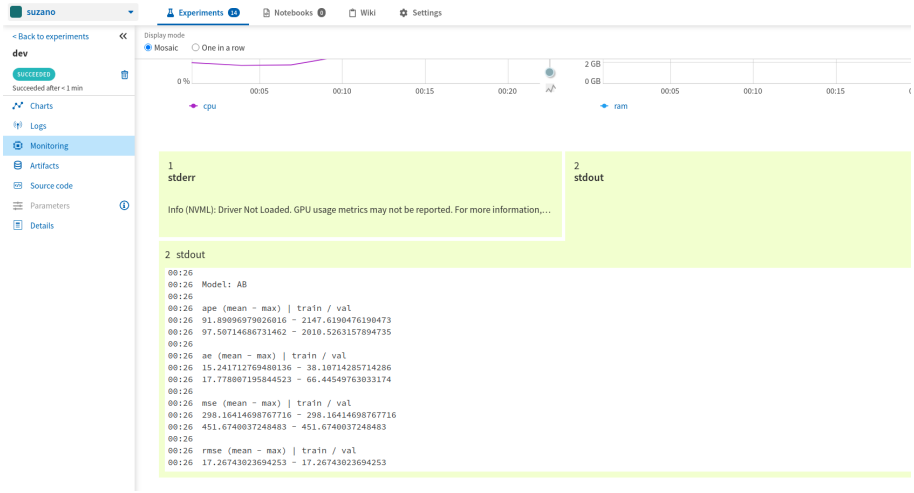
Nesse cenário, a perda econômica devido à imprecisão na predição da 'rul' também pode ser calculada. Ela corresponde ao número total de assets (100) vezes o erro absoluto médio (18); portanto, totalizando 1.800 ciclos desperdiçados. Ou seja, 77% de economia com relação ao plano de manutenção inicial.

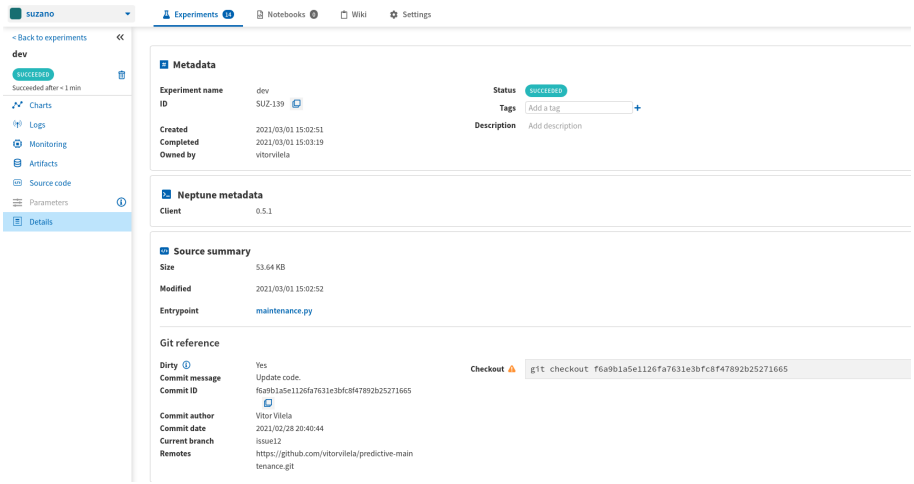
Para assumir parcialmente a manutenção corretiva, é possível substituir o valor de erro máximo 66 por uma função do erro médio e do desvio-padrão (18 e aprox. 16, respectivamente) [SUZ-139 @SUZ-182]. Por exemplo, considerando a distribuição uma Gaussiana, basear-se no valor de erro 34 ao invés de 66 (média adicionada de um desvio-padrão), corresponderia a 16% de manutenção corretiva, com contrapartida da menor perda por manutenção precoce (< 1.800). Outra possibilidade é considerar dois desvios-padrão, baseando-se no valor de erro 50 ao invés de 66, o que corresponderia a 3% de manutenção corretiva.

Uma abordagem adicional que pode ser aplicada tem semelhança com a estatística Bayesiana. 1. A distribuição dos ciclos de falha atuais do sistema é conhecida. 2. Essa distribuição deve ser transformada para outra que represente os assets com ciclos realizados maior do que alguns valores limites definidos. 3. Iniciando-se um novo lote de assets, com características similares, a cada manutenção (guiada por um valor limite de decisão de manutenção dinâmico) ou nova falha do asset (considerando a parcela de manutenção corretiva tolerada), a distribuição dos assets com ciclos realizados acima dos valores limites definidos deve ser atualizada. 3. A diferença dessas duas distribuições fornecerá uma função densidade de probabilidade indicativa de quais ciclos de falha têm maior probabilidade de ocorrer. Isso indicará o novo valor limite de decisão de manutenção que deve ser utilizado na próxima predição. Esse processo diminuirá ainda mais os prejuízos por manutenção precoce e a quantidade de manutenção corretiva.

[SUZ-139]


suzano
Experiments 14
Notebooks 0
Wiki
Settings
< Back to experiments
dev
SUCCEEDED
Succeeded after < 1 min
Charts
Logs
Monitoring
Artifacts
Source code
Parameters
Details
Source code
Size 53.64 KB
Entrypoint maintenance.py
source
Name
modeling.py
neptune_setting...
maintenance.py
analysis.py
dataset.py
neptune_settings.py
#self.origin_cycle_period = 1
# DOE: factor 1
self.option_use_derivatives = True # False / True
print(f'\noption_use_derivatives: {self.option_use_derivatives}')
self.option_min_monitoring_cycle_constant = True
# DOE: factor 2
# For training
self.min_monitoring_cycle_constant = 100 # 30 / 100
print(f'min_monitoring_cycle: {self.min_monitoring_cycle_constant}')
# DOE: factor 3
# Limited to 15 i.e.: test-(min cycle in all assets) - 2*filter_window_size
self.filter_window_size = 15 # 10 / 15
print(f'\nfilter_window_size: {self.filter_window_size}\n')
self.monitoring_cycle_step = int(0.5*self.filter_window_size)
self.n_monitoring_cycles_per_asset = 10
self.coverage_restraint = 0.5
#
self.option_polynomial_features = True
self.polynomial_features_degree = 1
self.option_standardize = True


suzano
Experiments 14
Notebooks 0
Wiki
Settings
< Back to experiments
dev
SUCCEEDED
Succeeded after < 1 min
Charts
Logs
Monitoring
Artifacts
Source code
Parameters
Details
Display mode
Mosaic
One in a row
0 %
2 GB
0 GB
00:05
00:10
00:15
00:20
cpu
ram
1
stderr
Info (NVML): Driver Not Loaded. GPU usage metrics may not be reported. For more information,...
2
stdout
2 stdout
00:26
00:26 Model: AB
00:26
00:26 ape (mean - max) | train / val
00:26 91.89096979026016 - 2147.6190476190473
00:26 97.50714686731462 - 2010.5263157894735
00:26
00:26 ae (mean - max) | train / val
00:26 15.241712769480136 - 38.10714285714286
00:26 17.778807195844523 - 66.44549763033174
00:26
00:26 mse (mean - max) | train / val
00:26 298.16414698767716 - 298.16414698767716
00:26 451.6740037248483 - 451.6740037248483
00:26
00:26 rmse (mean - max) | train / val
00:26 17.26743023694253 - 17.26743023694253


suzano
Experiments 14
Notebooks 0
Wiki
Settings
< Back to experiments
dev
SUCCEEDED
Succeeded after < 1 min
Charts
Logs
Monitoring
Artifacts
Source code
Parameters
Details
Metadata
Experiment name dev
ID SUZ-139
Created 2021/03/01 15:02:51
Completed 2021/03/01 15:03:19
Owned by vitorvilela
Status SUCCEEDED
Tags Add a tag
Description Add description
Neptune metadata
Client 0.5.1
Source summary
Size 53.64 KB
Modified 2021/03/01 15:02:52
Entrypoint maintenance.py
Git reference
Dirty Yes
Commit message Update code.
Commit ID f6a9b1a5e1126fa7631e3bfc8f47892b25271665
Commit author Vitor Vilela
Commit date 2021/02/28 20:40:44
Current branch issue12
Remotes https://github.com/vitorvilela/predictive-maintenance.git
Checkout
git checkout f6a9b1a5e1126fa7631e3bfc8f47892b25271665