



KICS Jupyter Notebook for Predictive Maintenance

**Author(s): George Makridis (gmakridis@unipi.gr),
George Fatouros (gfatouros@unipi.gr)**

Introduction

Η προγνωστική και η διαχείριση της υγείας συστημάτων (PHM) είναι ένας επιστημονικός κλάδος που χρησιμοποιεί αισθητήρες για να αξιολογήσει την υγεία των συστημάτων, να διαγνώσει ανώμαλη συμπεριφορά και να προβλέπει την υπολειπόμενη χρήσιμη απόδοση (RUL) κατά τη διάρκεια ζωής του περιουσιακού στοιχείου. Οι μεθοδολογίες PHM βασίζονται σε πολλά βασικά στοιχεία στα οποία οι αισθητήρες παρέχουν τη δυνατότητα παρακολούθησης των προδρόμων αστοχιών και των περιβαλλοντικών συνθηκών φόρτισης (π.χ. τάσεις). Στο PHM, υπάρχουν τρεις κύριοι άξονες: παρατήρηση, ανάλυση και δράση. Σε αυτό το πλαίσιο, η προγνωστική συντήρηση (PdM) συνδέεται άμεσα με τον άξονα παρατήρησης του PHM χρησιμοποιώντας έξυπνες μεθόδους για την πρόβλεψη αστοχιών συστημάτων. Το PdM που τροφοδοτείται από δεδομένα ιστορικού και σε πραγματικό χρόνο μπορεί να προβλέψει τάσεις, μοτίβα συμπεριφοράς και συσχετίσεις αξιοποιώντας στατιστικά μοντέλα ή μοντέλα μηχανικής μάθησης που ενισχύουν το χρόνο διακοπής λειτουργίας της μηχανής, το κόστος, τον έλεγχο και την ποιότητα της παραγωγής.

Business Motivation

Δεδομένου ότι ο αντίκτυπος της συντήρησης αντιπροσωπεύει συνολικά το 15 έως 60% του συνολικού κόστους λειτουργίας όλων των βιομηχανιών παραγωγής, είναι σημαντικό για τις βιομηχανίες να εξαλείψουν τις απρογραμμάτιστες διακοπές λειτουργίας του εξοπλισμού που προκαλούνται από αστοχία εξοπλισμού ή συστήματος με στόχο την αύξηση της χρήσης εργατικού δυναμικού και της παραγωγικής ικανότητας. Σύμφωνα με την εταιρεία συμβούλων διαχείρισης McKinsey, η προγνωστική συντήρηση μπορεί να αποφέρει σημαντική εξοικονόμηση πόρων αυξάνοντας τη διαθεσιμότητα της γραμμής παραγωγής κατά 5 έως 15% και μειώνοντας το κόστος συντήρησης κατά 18 έως 25%. Επιπλέον, το PdM είναι η πιο οικονομική στρατηγική συντήρησης σε σύγκριση με τις στρατηγικές προληπτικής και αντιδραστικής συντήρησης. Για παράδειγμα, σύμφωνα με το Υπουργείο Ενέργειας των ΗΠΑ, στη βιομηχανία πετρελαίου και φυσικού αερίου η προγνωστική συντήρηση βοηθά στην εξοικονόμηση μεταξύ 8% και 12% έναντι της προληπτικής συντήρησης. Βοηθά επίσης τις εταιρείες να εξοικονομήσουν πάνω από 30% έως 40% σε σχέση με την αντιδραστική συντήρηση

Scientific Background

Στα προγνωστικά, η υποβάθμιση ενός εξαρτήματος ή συστήματος είναι συνήθως μια μη γραμμική συνάρτηση πολλών παραμέτρων όπως το περιβάλλον λειτουργίας και το φόρτο εργασίας. Κανονικά, ένα σύστημα ή ένα εξάρτημα βρίσκεται σε καλή κατάσταση λειτουργίας σε πρώιμο στάδιο χρήσης. Η απόδοση του συστήματος αρχίζει να υποβαθμίζεται μετά τη λειτουργία του για κάποιο χρονικό διάστημα. Η διαδικασία αποδόμησης

επιταχύνεται με την πάροδο του χρόνου έως ότου συμβεί πλήρης διάσπαση. Οι καμπύλες υποβάθμισης συνήθως αντιπροσωπεύουν μεταβάσεις από μια περίπου σταθερή τιμή σε μια γραμμικά φθίνουσα καμπύλη προς το τέλος της ζωής. Το γόνατο στην καμπύλη αντανακλά το σημείο που αρχίζει η υποβάθμιση.

Η πιο σημαντική προσέγγιση για την εκτίμηση του RUL βασίζεται σε μεθόδους που βασίζονται σε δεδομένα, οι οποίες βασίζονται αποκλειστικά σε διαθέσιμα παλαιότερα παρατηρούμενα δεδομένα και στατιστικά μοντέλα. Μεταξύ των μεθόδων που βασίζονται σε δεδομένα, τα βαθιά νευρωνικά δίκτυα εφαρμόζονται ευρέως για προβλήματα ταξινόμησης και πρόβλεψης ακολουθιών. Οι C. Zhang et al. χρησιμοποίησε ένα πολυεπίπεδο νευρωνικό δίκτυο για την εκτίμηση RUL σε ένα πραγματικό σύνολο δεδομένων με πολλαπλές μετρήσεις αισθητήρων. Τα δίκτυα LSTM είναι ένα ειδικό είδος επαναλαμβανόμενου νευρωνικού δικτύου (RNN) που μπορεί να διατηρήσει μακροπρόθεσμες μνήμες. Λόγω της ικανότητάς τους να μαθαίνουν μακροπρόθεσμες εξαρτήσεις, τα δίκτυα LSTM έχουν αποδειχθεί ευρέως ότι είναι χρήσιμα για την εκμάθηση ακολουθιών με μακροπρόθεσμα μοτίβα. Πολλαπλές ερευνητικές ομάδες έχουν χρησιμοποιήσει το LSTM για την εκτίμηση RUL πρότειναν δίκτυα LSTM ενσωματωμένα με πλήρως συνδεδεμένα νευρωνικά δίκτυα (NN) και τα αποτελέσματά τους δείχνουν ότι το μοντέλο τους υπερέχει άλλων μεθόδων βαθιάς μάθησης στην εκτίμηση RUL σε πολλαπλά σύνολα δεδομένων.

Για τις απότομες βλάβες, η λειτουργία υποβάθμισης ξεκινά πολύ κοντά στην αστοχία. Η εκτίμηση RUL είναι πολύ δύσκολη όταν έχουμε απότομες αποτυχίες. Στο βήμα εκμάθησης, η μεγάλη ποσότητα κανονικών δεδομένων μπορεί να κρύψει το μοτίβο υποβάθμισης στο σύνολο δεδομένων. Στο βήμα υλοποίησης, η πραγματοποίηση εκτίμησης RUL ενώ το σύστημα είναι ακόμα σε υγιή τρόπο λειτουργίας και οι αισθητήρες δεν δείχνουν σημάδια υποβάθμισης μπορεί να οδηγήσει σε ανούσιες εκτιμήσεις και να βλάψει την αξιοπιστία του μοντέλου εκτιμητή RUL. Σε αυτή την εργασία, θα ακολουθήσουμε τους Huang, Wei, et al. προσέγγιση σχεδιασμού μιας μονάδας έγκαιρης ανίχνευσης σφαλμάτων για την ανίχνευση υποβάθμισης του συστήματος στα αρχικά στάδια. Όταν ανιχνεύονται οι τρόποι υποβάθμισης, εφαρμόζουμε μια δομή δικτύων LSTM που βασίζεται στους Zheng et al. για την εκτίμηση του συστήματος RUL.

Data Sources

Common data sources for predictive maintenance problems are :

- **Failure history:** The failure history of a machine or component within the machine.
- **Maintenance history:** The repair history of a machine, e.g. error codes, previous maintenance activities or component replacements.
- **Machine conditions and usage:** The operating conditions of a machine e.g. data collected from sensors.
- **Machine features:** The features of a machine, e.g. engine size, make and model, location.
- **Operator features:** The features of the operator, e.g. gender, past experience. The data for this example comes from 4 different sources which are real-time telemetry data collected from machines, error messages, historical maintenance records that include failures and machine information such as type and age.

Explanatory Data Analysis

Import libraries/frameworks

In [36]:

```

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn import preprocessing
import missingno as msno
import seaborn as sns
import scipy.stats as st
import matplotlib.pyplot as plt
colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
pd.set_option('display.max_columns', None)

```

Loading the data

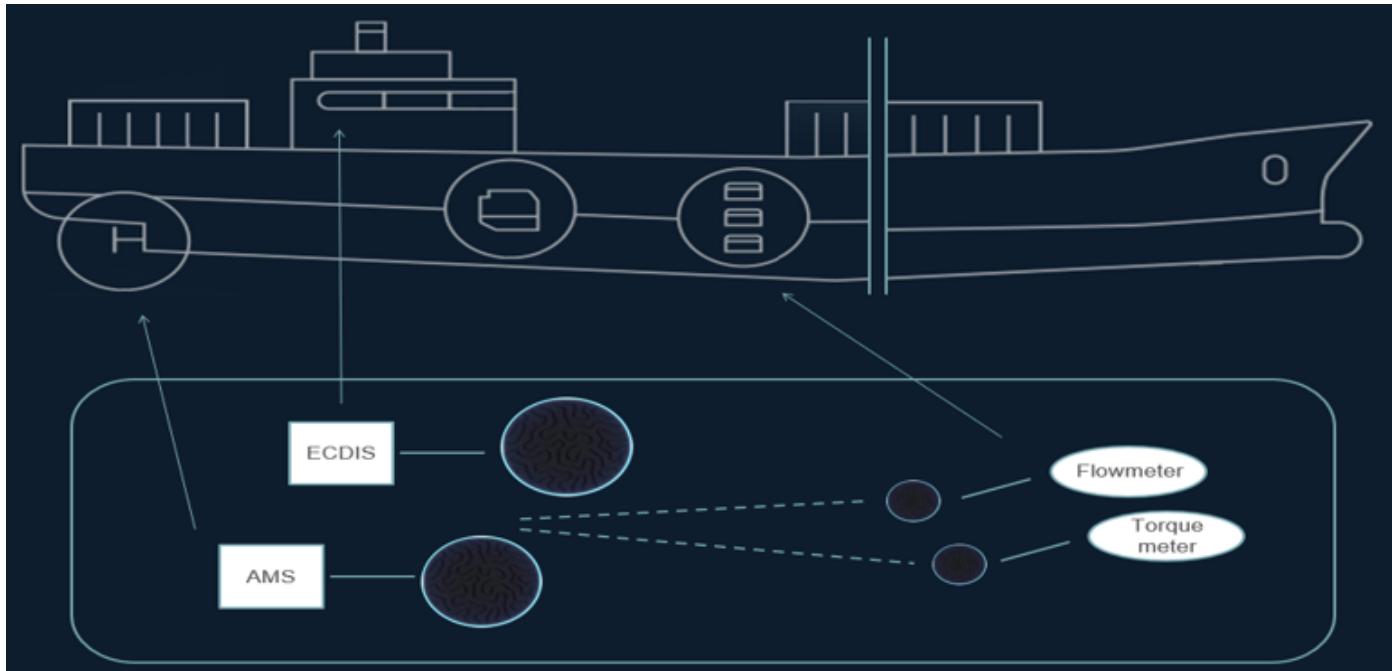
Στόχος μας είναι να χρησιμοποιήσουμε τη Μηχανική Εκμάθηση και τις στατιστικές προσεγγίσεις για τον έγκαιρο εντοπισμό μη φυσιολογικών λειτουργιών που σχετίζονται με ελαττώματα σταυρωτά ρουλεμάν.

Τα δεδομένα που χρησιμοποιούνται στην εκπαίδευση μοντέλων έχουν ανακτηθεί μέσω της βάσης δεδομένων μιας ναυτιλιακής εταιρείας που ενσωματώνει τις ροές δεδομένων από όλους τους αισθητήρες που έχουν αναπτυχθεί στα σκάφη. Αποτελούνται από πολλαπλές ροές δεδομένων από τους κύριους αισθητήρες κινητήρα που συνοδεύονται από κάποια γενική ροή δεδομένων σκάφους. Κάθε ροή αντιπροσωπεύει τα αρχεία που λαμβάνονται διαχρονικά από έναν συγκεκριμένο αισθητήρα στο πλοίο. Δειγματοληψία όλων αυτών βασίζεται σε χρονική περίοδο 1 λεπτού ή 10 λεπτών. Αυτά θα αναφέρονται επίσης ως "χαρακτηριστικά" στο υπόλοιπο σημειωματάριο. Αυτά τα δεδομένα συνοψίζονται στον Πίνακα I.

TABLE I: Brief Description of the utilized Dataset

VESSEL_DATA table structure	MAIN_ENGINE_DATA table structure
Consumed power (kW),	Air Cooler Cooling Water Inlet Pressure (Pa)
Wind-speed (kn),	Cylinder #1-10 Scavenge Air Fire Detection Temperature (°C),
GPS speed (kn),	Cooling Fresh Water Inlet Pressure (Pa),
Speed through water – longitudinal (kn),	Control Air Pressure (Pa),
Speed through water – transverse (kn),	Exhaust Valve Spring Air Inlet Pressure (Pa)
rotations per minute of the main shaft,	Fuel Oil Flow Rate (lt),
Wind angle (0-359.99 degrees),	Fuel Oil Inlet Pressure (Pa),
Total Twenty-foot Equivalent Unit (TEU) (# of containers),	Fuel Oil Inlet Temperature (°C),
Total Forty-foot Equivalent Unit (FEU) (# of containers),	Jacket Cooling Fresh Water Inlet Temperature Low (°C)
Low-sulfur fuel oil consumption (metric tones),	Cylinder #1-10 Exhaust Gas Out Temperature (°C),
High-sulfur fuel oil consumption (metric tones),	Cylinder #1-10 Jacket Cooling Fresh Water Outlet Temperature (°C),
Vessel draft at port-side (left-side looking to the fore) (m),	Cylinder #1-10 Piston Cooling Outlet Temperature (°C),
Vessel draft at starboard-side (right-side looking to the fore) (m),	Turbo-Charger #1-3 Exhaust Gas Inlet Temperature (°C)
Vessel draft at stern (m),	Turbo-Charger #1-3 Exhaust Gas Outlet Temperature (°C),
Vessel draft at fore (m),	Turbo-Charger #1-3 Lube Oil Inlet Pressure (Pa),
Speed through water – calculated by stw_trans and stw_lon (kn),	Turbo-Charger #1-3 Lube Oil Outlet Pressure (Pa),
Total number of containers,	Turbo-Charger #1-3 RPMs,
Vessel draft at mid-line (m),	Scavenge Air Inlet Pressure (Pa),
The trim of the vessel, calculated by draft_aft and draft_fore,	Scavenge Air Receiver Temperature (°C),
The latitude of the vessel's position,	Starting Air Pressure (Pa),
The longitude of the vessel's position,	Thrust Pad Temperature (°C),
	Main Lube Oil Inlet Pressure (Pa),
	Main Lube Oil Inlet Temperature (°C)
	Fuel Oil Temperature (°C)
	Torque of the main shaft (N/m),
	Turbo-Charger #1-3 Air Cooler Cooling Water Outlet Temperature (°C),

Ιστορικά δεδομένα από XX παρόμοια πλοία ενσωματώθηκαν και δεδομένα για την πρόβλεψη ανακτήθηκαν μέσω ενός συστήματος DAQ (απόκτηση δεδομένων) (βλ. Εικ. 1).



Θα πρέπει να αναφερθεί ότι τα δεδομένα είχαν ήδη κατηγοριοποιηθεί (επισημανθεί) με προσαρμοσμένη διαδικασία που θα περιγραφεί παρακάτω.

Vessel Data

Η πρώτη πηγή δεδομένων είναι τα δεδομένα χρονοσειράς σκάφους που αποτελούνται από γενικές μετρήσεις σκαφών (δηλαδή άνεμος, ταχύτητα, φορτίο, κατανάλωση κ.λπ.) που συλλέγονται από ένα πλοίο κάθε ένα λεπτό. Παρακάτω, εμφανίζουμε τις πρώτες 10 εγγραφές στο σύνολο δεδομένων. Παρέχεται επίσης μια περίληψη ολόκληρου του συνόλου δεδομένων.

In [37]:

```
vds_5 = pd.read_csv('input/5_vds.csv', index_col='datetime', parse_dates=['datetime'])
vds_5.head()
```

Out[37]:

	vessel_code	power	apparent_wind_speed	speed_overground	stw_long	stw_trans
datetime						
2012-11-30 18:20:00	5	NaN		NaN	16.7	NaN
2012-11-30 18:21:00	5	NaN		NaN	16.9	NaN
2012-11-30 18:22:00	5	NaN		NaN	17.0	NaN
2012-11-30 18:23:00	5	NaN		NaN	16.5	NaN
2012-11-30 18:24:00	5	NaN		NaN	15.5	NaN

Main Engine Data

Εν συνεχεία εχουμε τα δεδομετα της κύριας μηχανής που αποτελούνται από μετρήσεις θερμοκρασίας και πιέσεων σε διάφορα σημεία της κύριας μηχανής σκαφών που συλλέγονται από ένα πλοίο κάθε δεκα με ένα λεπτό. Παρακάτω, εμφανίζουμε τις πρώτες 10 εγγραφές στο σύνολο δεδομένων. Παρέχεται επίσης μια περίληψη ολόκληρου του συνόλου δεδομένων.

In [38]:

```
mes_5 = pd.read_csv('input/5_mes.csv', index_col='datetime', parse_dates=['datetime'])
mes_5.head()
```

Out[38]:

vessel_code	airCoolerCWinLETPress	scavAirFireDetTempNo1	scavAirFireDetTempNo2
-------------	-----------------------	-----------------------	-----------------------

vessel_code	airCoolerCWinLETPress	scavAirFireDetTempNo1	scavAirFireDetTempNo2	
2012-12-01 22:31:00	5	NaN	NaN	NaN
2012-12-01 22:41:00	5	NaN	NaN	NaN
2012-12-01 22:51:00	5	NaN	NaN	NaN
2012-12-01 23:01:00	5	NaN	NaN	NaN
2012-12-01 23:11:00	5	NaN	NaN	NaN

Περιγραφή Δεδομένων

Η περιγραφή των χαρακτηριστικών των δεδομένων μπορεί να γίνει σε διάφορα στάδια με τη χρήση της βιβλιοθήκης "pandas"

In [39]:

```
print(mes_5.info())
print(vds_5.info())

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2999485 entries, 2012-12-01 22:31:00 to 2018-10-05 08:0
3:00
Columns: 101 entries, vessel_code to foID
dtypes: float64(100), int64(1)
memory usage: 2.3 GB
None
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3013258 entries, 2012-11-30 18:20:00 to 2018-10-05 08:0
3:00
Data columns (total 39 columns):
 #   Column           Dtype  
 --- 
 0   vessel_code      int64  
 1   power            float64
 2   apparent_wind_speed float64
 3   speed_overground float64
 4   stw_long          float64
 5   stw_trans         float64
 6   rpm               float64
 7   apparent_wind_angle float64
 8   total_teus        float64
 9   total_feus         float64
 10  cons_ifo_static_counter float64
 11  port_mid_draft    float64
 12  stbd_mid_draft    float64
 13  draft_aft          float64
 14  draft_fore         float64
 15  stw               float64
 16  equivalent_teus    float64
 17  mid_draft          float64
 18  trim              float64
 19  latitude           float64
 20  longitude          float64
 21  cons_ifo_static1_counter float64
 22  wind_speed          float64
 23  wind_ucomp          float64
 24  wind_vcomp          float64
 25  wind_wave_dir       float64
 26  wind_wave_height    float64
 27  wind_wave_mean_period float64
 28  current_ucomp       float64
 29  current_vcomp       float64
 30  relative_humidity   float64
 31  primary_wave_dir    float64
 32  primary_wave_length  float64
 33  primary_wave_mean_period float64
 34  swell_wave_dir      float64
 35  swell_wave_height   float64
 36  swell_wave_mean_period float64
 37  air_temperature      float64
 38  comb_wind_swell_wave_height float64
dtypes: float64(38), int64(1)
memory usage: 919.6 MB
None
```

Missing Values

Στην συνέχεια θα εξετάσουμε πόσες από τις μεταβλητές μας έχουν κενές τιμές, και θα απεικονίσουμε το σύνολο των δεδομένων για να γίνει κατανοητό.

In [40]:

```
mes_5.isna().sum(), vds_5.isna().sum()
```

Out[40]:

(vessel_code	0
airCoolerCWInLETPress	1451476
scavAirFireDetTempNo1	1454124
scavAirFireDetTempNo2	1454123
scavAirFireDetTempNo3	1454123
scavAirFireDetTempNo4	1454154
scavAirFireDetTempNo5	1454154
scavAirFireDetTempNo6	1454154
scavAirFireDetTempNo7	1454154
scavAirFireDetTempNo8	1454154
scavAirFireDetTempNo9	1454148
scavAirFireDetTempNo10	1454148
scavAirFireDetTempNo11	2999485
scavAirFireDetTempNo12	2999485
coolerCWInTemp	2999485
cfWInPress	1114425
controlAirPress	1114412
cvlLoTemp	2999485

Visualizing Missing Values

αρχικά για τα main engine data και στη συνέχεια για τα vessel data

In [41]:

```
msno.matrix(mes_5)
```

Out[41]:

```
<AxesSubplot:>
```

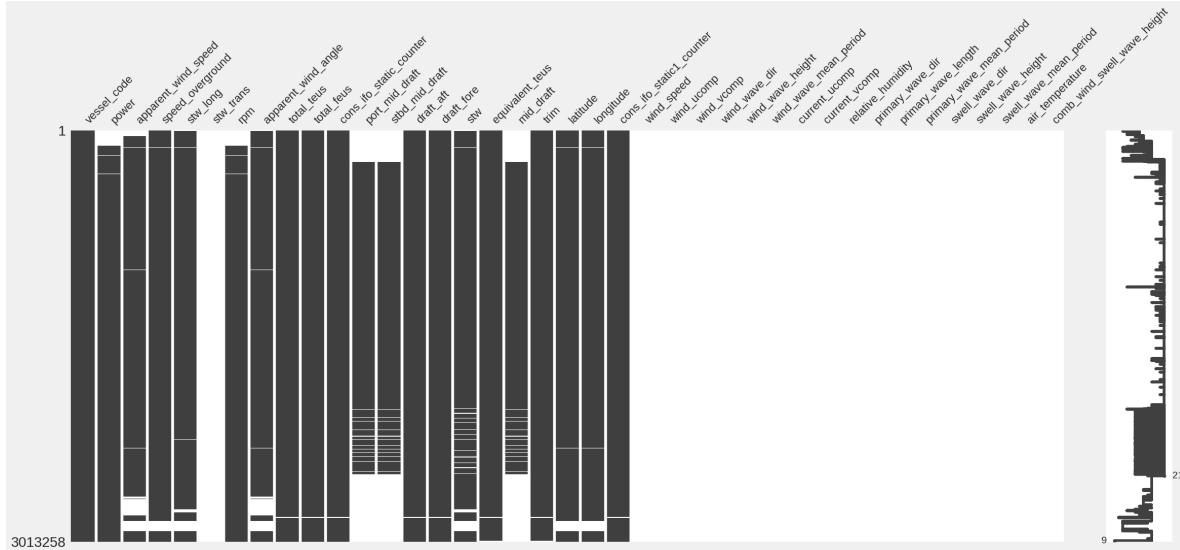


In [42]:

```
msno.matrix(vds_5)
```

Out[42]:

<AxesSubplot:>



Οπως παρατηρούμε πολλες απο τα δεδομένα μας είναι κενά, δηλαδή ολόκληρη η στήλη που περιέχει δεδομένα από ένα αισθητήρα είναι κενά. Σε αυτή την περίπτωση αυτή διώχνουμε τις στήλες αυτές.

In [43]:

```
mes_5.dropna(axis=1, how='all', inplace=True)
```

In [44]:

```
vds_5.dropna(axis=1, how='all', inplace=True)
```

Feature Engineering

Το πρώτο βήμα στις εφαρμογές προγνωστικής συντήρησης είναι η μηχανική χαρακτηριστικών που απαιτεί τη συγκέντρωση των διαφορετικών πηγών δεδομένων για τη δημιουργία χαρακτηριστικών που περιγράφουν καλύτερα την κατάσταση υγείας ενός μηχανήματος σε μια δεδομένη χρονική στιγμή. Στις επόμενες ενότητες, χρησιμοποιούνται διάφορες μέθοδοι μηχανικής χαρακτηριστικών για τη δημιουργία χαρακτηριστικών με βάση τις ιδιότητες κάθε πηγής δεδομένων.

Lag Features

Τα δεδομένα αισθητήρα σχεδόν πάντα συνοδεύονται από χρονικές σημάνσεις, γεγονός που το καθιστά κατάλληλο για τον υπολογισμό των χαρακτηριστικών που υστερούν. Μια συνηθισμένη μέθοδος είναι να επιλέγετε ένα μέγεθος παραθύρου για τα χαρακτηριστικά υστέρησης που θα δημιουργηθούν και να υπολογίσετε τα κυλιόμενα αθροιστικά μέτρα όπως ο μέσος όρος, η τυπική απόκλιση, το ελάχιστο, το μέγιστο, κ.λπ. για να αντιπροσωπεύσετε το βραχυπρόθεσμο ιστορικό της τηλεμετρίας στο παράθυρο υστέρησης.

Στη συνέχεια, επαναδειγματολειπτούμε τα δεδομένα σε επίπεδο ημέρας, υπολογίζοντας την μέση τιμή κάθε μέρας.

In [45]:

```
mes_5 = mes_5.resample('D').mean()  
vds_5 = vds_5.resample('D').mean()
```

In [46]:

```
mes_5.shape
```

Out[46]:

```
(2135, 79)
```

Οι τιμές που λείπουν καταλογίστηκαν εφαρμόζοντας πρώτα το "Forward Fill" και μετά το "Backward Fill". Στα περισσότερα από τα ρεύματα υπήρχαν κάποιες ακραίες τιμές που μπορεί να αντιστοιχούν στο ελάττωμα κάθε αισθητήρα κάποιου άλλου μετασχηματισμού κατά τη διαδικασία αποθήκευσης. Αν και τα διανύσματα χαρακτηριστικών σαρώνονται για στοιχεία με τιμές πέραν των 3 SD (τυπικές αποκλίσεις), τότε περικόψαμε αυτά τα διανύσματα. Υποθέτοντας μια κανονική κατανομή, το 99,7% των κανονικών δεδομένων θα πρέπει να είναι εντός του μέσου όρου + 3SD. Επομένως, αυτή η διατύπωση φίλτραρει τα περισσότερα μη φυσιολογικά σημεία δεδομένων από το σύνολο δεδομένων, χωρίς να επηρεάζει τη συντριπτική πλειοψηφία των κανονικών σημείων. Προκειμένου να εξομαλυνθεί περαιτέρω η χρονοσειρά, χρησιμοποιήσαμε ένα κυλιόμενο μέσο για τη μείωση ή την εξάλειψη της βραχυπρόθεσμης αστάθειας στα δεδομένα.

fill

In [47]:

```
mes_5 = mes_5.fillna(method='ffill')
```

In [48]:

```
mes_5 = mes_5.fillna(method='bfill')
```

In [49]:

```
vds_5 = vds_5.fillna(method='ffill')
```

In [50]:

```
vds_5 = vds_5.fillna(method='bfill')
```

clip

In [51]:

```
useful_cols = [x for x in mes_5.columns if x not in ['datetime', 'vessel_code']]  
  
for column in useful_cols:  
    data = mes_5[column]  
    UPPERBOUND, LOWERBOUND = data.quantile([0.001, 0.999])  
    data.clip(lower=LOWERBOUND, upper=UPPERBOUND, inplace=True)  
    mes_5[column] = data
```

In [52]:

```
useful_cols = [x for x in vds_5.columns if x not in ['datetime', 'vessel_code']]  
  
for column in useful_cols:  
    data = vds_5[column]  
    UPPERBOUND, LOWERBOUND = data.quantile([0.001, 0.999])  
    data.clip(lower=LOWERBOUND, upper=UPPERBOUND, inplace=True)  
    vds_5[column] = data
```

Errors

The second major data source is the error logs. These are **non-breaking errors thrown while the machine is still operational and do not constitute as failures**. The **error date and times** are rounded to the closest hour since the telemetry data is collected at an hourly rate.

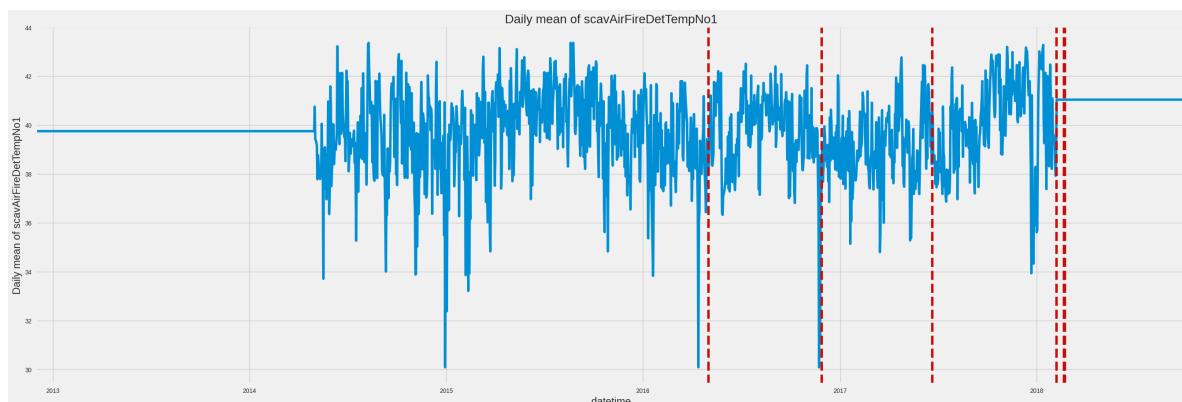
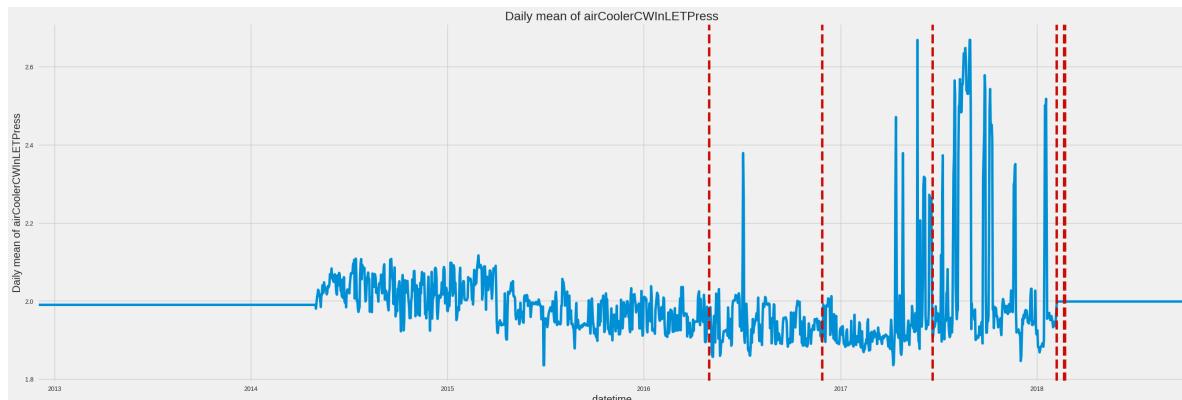
In [53]:

```
failure_dates = ["2016-05-02", '2016-11-28', '2017-06-21', '2018-02-07', '2018-02-2
```

Ας επεικονίσουμε κάποιες χρονοσειρές

In [54]:

```
for column in mes_5.columns[1:3]:
    data = mes_5[column]
    ax = data.plot(figsize=(30,10))
    ax.set_ylabel("Daily mean of " + str(column))
    ax.set_title("Daily mean of " + str(column))
    xposition = failure_dates
    for xc in xposition:
        plt.axvline(x=xc, color='r', linestyle='--')
    plt.show()
```



In []:

In [55]:

```
import seaborn as sns

def heatMap(df):
    corr = df.corr()
    fig, ax = plt.subplots(figsize=(50, 50))
    colormap = sns.diverging_palette(220, 10, as_cmap=True)
    sns.heatmap(corr, cmap=colormap, annot=True, fmt=".2f")
    plt.show()
```

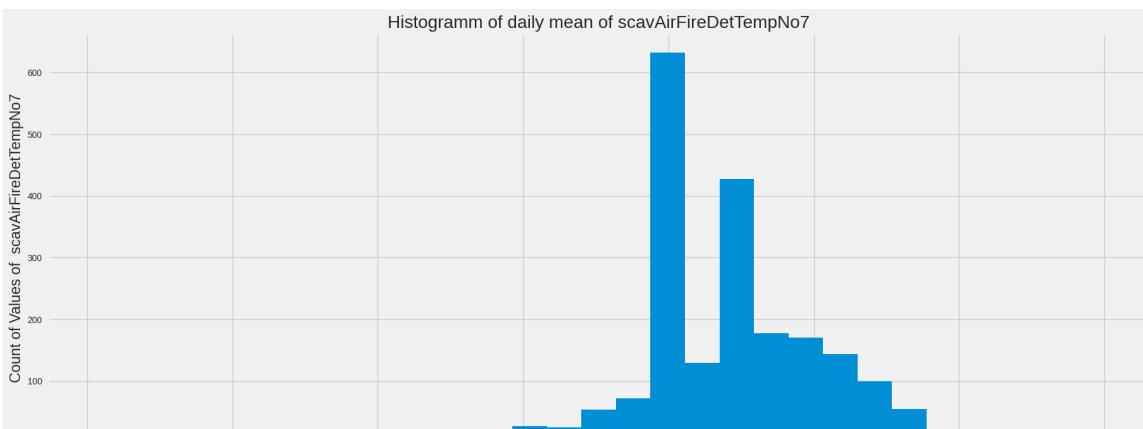
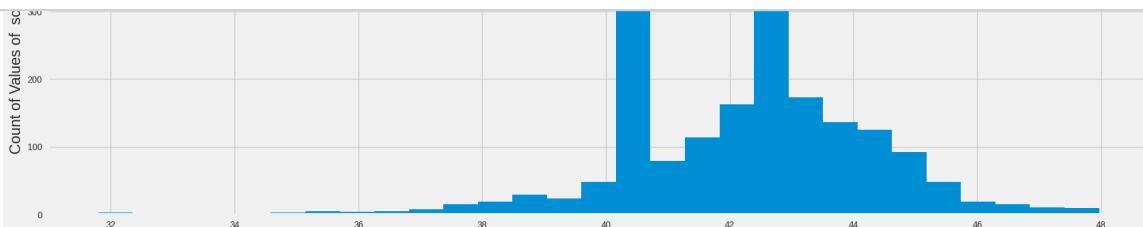
In [56]:

heatMap(df = mes_5[mes_5.columns[1:-1]])



In [57]:

```
for column in mes_5.columns[1:-1]:  
  
    data = mes_5[column]  
  
    if data.max()-data.min() < 30:  
        num = 30  
    elif data.max()-data.min() >10000:  
        num = (data.max()-data.min())/500  
    elif data.max() - data.min() > 1000:  
        num = (data.max() - data.min()) / 50  
    elif data.max()-data.min() >100:  
        num = (data.max()-data.min())/5  
    else:  
        num = data.max()-data.min()  
  
    bins = np.linspace(data.min(), data.max(), num)  
  
    ax = data.plot.hist(figsize=(20,8) , bins=bins)  
  
    ax.set_ylabel("Count of Values of " + str(column))  
    ax.set_title("Histogramm of daily mean of " + str(column))  
    plt.show()
```



Pandas Profiling

In [58]:

```
import pandas_profiling  
pandas_profiling.ProfileReport(vds_5)
```

Summarize dataset: 396/396 [00:35<00:00, 8.05it/s,

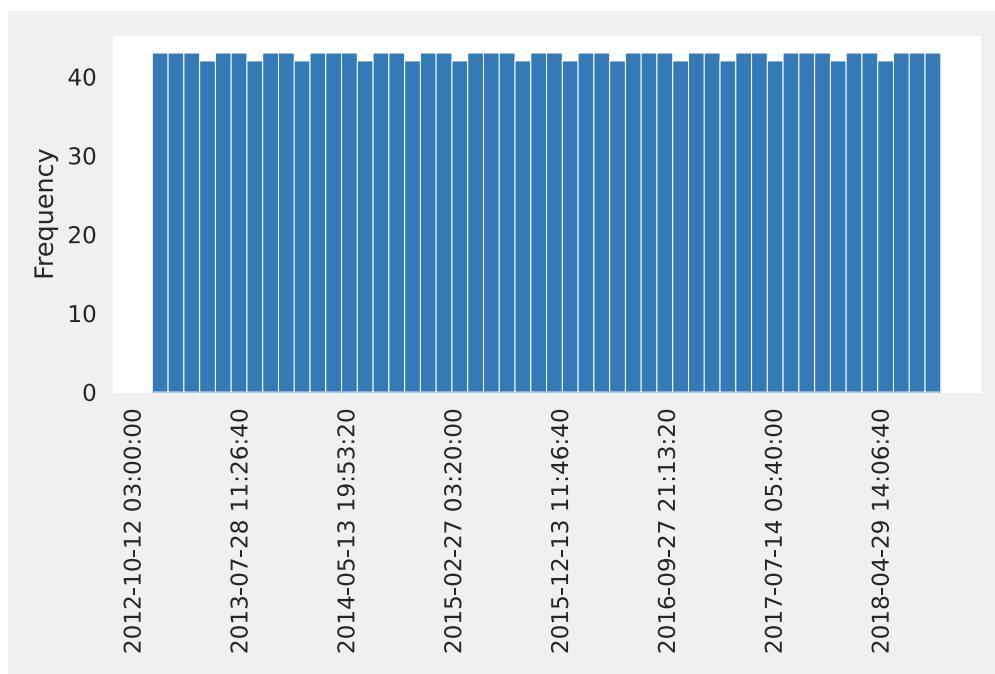
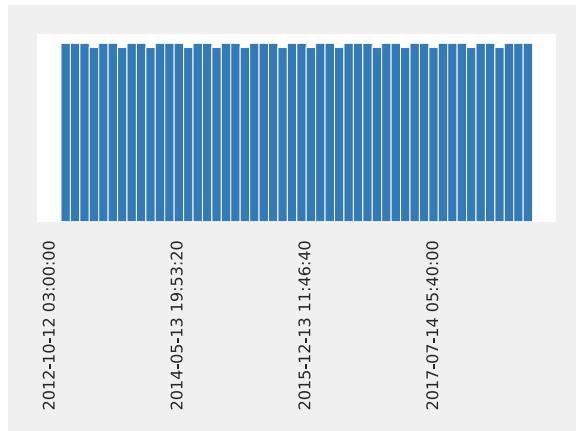
100% Completed]

Generate report structure: 1/1 [00:05<00:00,

100% 5.27s/it]

Render HTML: 100% 1/1 [00:06<00:00, 6.23s/it]

Missing (%)	0.0%
Memory size	16.8 KiB
Minimum	2012-11-30 00:00:00
Maximum	2018-10-05 00:00:00



Histogram with fixed size bins (bins=50)

Out[58]:

Stationarity Check

In [59]:

```
from pandas import Series
from statsmodels.tsa.stattools import adfuller
for col in mes_5.columns.values:
    X = mes_5[col].values
    result = adfuller(X)
    print('-----'+str(col)+'-----')
    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))

1%: -3.433
5%: -2.863
10%: -2.568
-----scavAirFireDetTempNo1-----
ADF Statistic: -6.694516
p-value: 0.000000
Critical Values:
    1%: -3.433
    5%: -2.863
    10%: -2.567
-----scavAirFireDetTempNo2-----
ADF Statistic: -5.775869
p-value: 0.000001
Critical Values:
    1%: -3.433
    5%: -2.863
    10%: -2.567
-----scavAirFireDetTempNo3-----
ADF Statistic: -7.352123
p-value: 0.000000
```

Modelling

Στα προγνωστικά, η υποβάθμιση ενός εξαρτήματος ή συστήματος είναι συνήθως μια μη γραμμική συνάρτηση πολλών παραμέτρων. Η διαδικασία αποικοδόμησης επιταχύνεται με την πάροδο του χρόνου έως ότου συμβεί πλήρης διάσπαση.

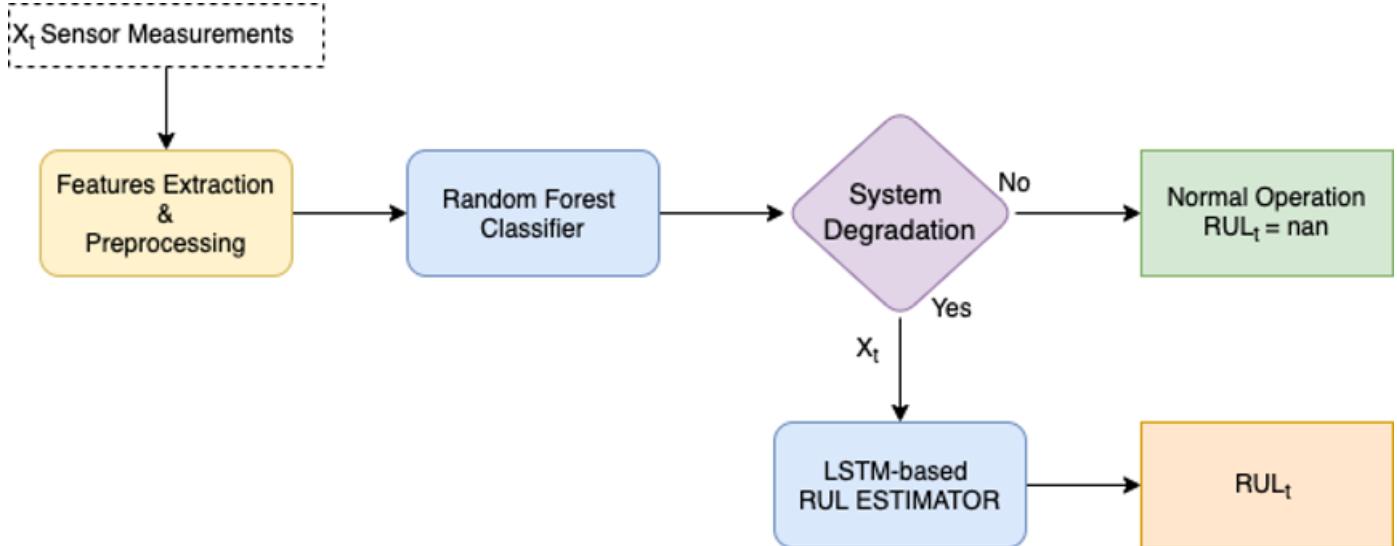
Η πιο εξέχουσα προσέγγιση για την εκτίμηση του RUL βασίζεται σε μεθόδους που βασίζονται σε δεδομένα, οι οποίες βασίζονται καθαρά σε διαθέσιμα δεδομένα που παρατηρήθηκαν στο παρελθόν και σε στατιστικά μοντέλα. Μεταξύ των μεθόδων που βασίζονται σε δεδομένα, τα βαθιά νευρωνικά δίκτυα εφαρμόζονται ευρέως για προβλήματα ταξινόμησης και πρόβλεψης ακολουθιών. Οι C. Zhang et al. χρησιμοποίησε ένα πολυεπίπεδο νευρωνικό δίκτυο για την εκτίμηση RUL σε ένα πραγματικό σύνολο δεδομένων με πολλαπλές μετρήσεις αισθητήρων.

Για τις απότομες βλάβες, η λειτουργία υποβάθμισης ξεκινά πολύ κοντά στην αστοχία. Η εκτίμηση RUL είναι πολύ δύσκολη όταν έχουμε απότομες αστοχίες. Στο βήμα εκμάθησης, η μεγάλη ποσότητα κανονικών δεδομένων μπορεί να κρύψει το μοτίβο υποβάθμισης στο σύνολο δεδομένων. Στο βήμα υλοποίησης, η πραγματοποίηση εκτίμησης RUL ενώ το σύστημα είναι ακόμα σε υγιή τρόπο λειτουργίας και οι αισθητήρες δεν δείχνουν σημάδια υποβάθμισης μπορεί να οδηγήσει σε ανούσιες εκτιμήσεις και να βλάψει την αξιοπιστία του μοντέλου εκτιμητή RUL. Σε αυτή την εργασία, θα ακολουθήσουμε τους Huang, Wei, et al. προσέγγιση σχεδιασμού μιας μονάδας έγκαιρης ανίχνευσης σφαλμάτων για την ανίχνευση υποβάθμίσεων του συστήματος στα αρχικά στάδια. Όταν ανιχνεύονται οι τρόποι υποβάθμισης, εφαρμόζουμε μια δομή δικτύων LSTM που

Βασίζεται στους Zheng et al. για την εκτίμηση του συστήματος RUL. Σε σχέση με την προτεινόμενη λύση και την ολότητά της, θα πρέπει να σημειωθεί ότι υπάρχουν δύο επικρατούσες προσεγγίσεις που χρησιμοποιούν μηχανική μάθηση για την εκτίμηση της τρέχουσας κατάστασης του μηχανικού εξοπλισμού:

η πρώτη αναφέρεται στην πρόβλεψη του χρόνου μεταξύ των βλαβών (RUL),

ενώ η δεύτερη αναφέρεται στην κατηγοριοποιησης κατάστασης(Anomaly Detection).



Μετά τα βήματα της μηχανικής χαρακτηριστικών και της επισήμανσης, αυτό το σημειωματάριο μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός μοντέλου πρόβλεψης. Παρακάτω, περιγράφουμε τη διαδικασία μοντελοποίησης και παρέχουμε ένα παράδειγμα μοντέλου Python.

Labeling

In [60]:

```

from datetime import timedelta

def custom_labeling(train=None, vessel_id=None):
    WINDOW = 30

    train['label'] = 0
    defect_days_df = pd.read_csv('input/Bearing Damages.csv')
    xposition = defect_days_df['defect_date'][defect_days_df['vessel_id'] == vessel_id]
    print(xposition)
    for i in xposition:
        train['label'].loc[train.index == i] = 1

    idx = train.index[train['label'] == 1]
    print(idx)
    for j in idx:
        train['label'].loc[j - timedelta(minutes=WINDOW): j] = 1

    return train
  
```

In [61]:

```
ID=5
vds_5 = custom_labeling(train=vds_5, vessel_id=ID)
# mes_5 = custom_labeling(train=mes_5, vessel_id=ID)

3    2018-02-21
4    2016-11-28
5    2018-04-22
6    2017-06-21
7    2018-02-21
8    2018-04-22
9    2018-02-21
10   2018-02-07
11   2016-05-02
12   2018-04-21
13   2018-04-21
14   2018-04-21
15   2018-04-21
Name: defect_date, dtype: object
DatetimeIndex(['2016-05-02', '2016-11-28', '2017-06-21', '2018-02-07',
               '2018-02-21', '2018-04-21', '2018-04-22'],
              dtype='datetime64[ns]', name='datetime', freq=None)
```

In []:

In []:

Classification

In [62]:

```
imbalance_ratio = vds_5['label'].sum()/len(vds_5)
print('Imbalance Ratio :' + str(imbalance_ratio))
```

Imbalance Ratio :0.0032771535580524347

In [63]:

```
# imbalance_ratio = mes_5['label'].sum()/len(mes_5)
# print('Imbalance Ratio :' + str(imbalance_ratio))
```

In [64]:

```
df_final = mes_5.join(vds_5, on=None, how='left', lsuffix='_mes', rsuffix='_vds', s
```

In []:

Training, Validation and Testing

Ένα κρίσιμο σύνολο δεδομένων σε αυτό το παράδειγμα είναι οι εγγραφές ελαττωμάτων που περιέχουν πληροφορίες για τις εγγραφές αντικατάστασης εξαρτημάτων. Οι πιο σχετικές πληροφορίες θα ήταν ο υπολογισμός του χρόνου που έχει περάσει από την τελευταία φορά που αντικαταστάθηκε ένα εξάρτημα, καθώς αυτό θα αναμενόταν να συσχετιστεί καλύτερα με τις αστοχίες εξαρτήματος, καθώς όσο περισσότερο χρησιμοποιείται ένα εξάρτημα, τόσο μεγαλύτερη θα πρέπει να αναμένεται η υποβάθμιση. Στη συνέχεια, οι ημέρες από την τελευταία αντικατάσταση εξαρτήματος υπολογίζονται για κάθε τύπο εξαρτήματος ως χαρακτηριστικά από τα δεδομένα συντήρησης. Όταν εργάζεστε με δεδομένα με χρονική σήμανση, όπως σε αυτό το παράδειγμα, η κατάτμηση εγγραφών σε σύνολα εκπαίδευσης, επικύρωσης και δοκιμής θα πρέπει να εκτελείται προσεκτικά για να αποφευχθεί η υπερεκτίμηση της απόδοσης των μοντέλων. Στην προγνωστική συντήρηση, τα χαρακτηριστικά δημιουργούνται συνήθως χρησιμοποιώντας υστερούντα συγκεντρωτικά στοιχεία: οι εγγραφές στο ίδιο χρονικό παράθυρο πιθανότατα θα έχουν πανομοιότυπες ετικέτες και παρόμοιες τιμές χαρακτηριστικών. Αυτοί οι συσχετισμοί μπορούν να δώσουν σε ένα μοντέλο ένα «αθέμιτο πλεονέκτημα» όταν προβλέπει ένα ρεκόρ σε δοκιμής που μοιράζεται το χρονικό του παράθυρο με ένα ρεκόρ σε εκπαίδευσης. Διαχωρίζουμε τις εγγραφές σε σετ εκπαίδευσης, επικύρωσης και δοκιμών σε μεγάλα κομμάτια, για να ελαχιστοποιήσουμε τον αριθμό των χρονικών διαστημάτων που μοιράζονται μεταξύ τους.

Τα προγνωστικά μοντέλα δεν έχουν προηγμένη γνώση των μελλοντικών χρονολογικών τάσεων: στην πράξη, τέτοιες κινήσεις είναι πιθανό να υπάρχουν και να επηρεάζουν αρνητικά την απόδοση του μοντέλου. Για να λάβετε μια ακριβή αξιολόγηση της απόδοσης ενός προγνωστικού μοντέλου, συνιστούμε εκπαίδευση σε παλαιότερες εγγραφές και επικύρωση/δοκιμή χρησιμοποιώντας νεότερες εγγραφές.

Και για τους δύο αυτούς λόγους, μια εξαρτώμενη από το χρόνο στρατηγική διαχωρισμού αρχείων είναι μια εξαιρετική επιλογή για μοντέλα πρόβλεψης συντήρησης. Ο διαχωρισμός πραγματοποιείται επιλέγοντας ένα χρονικό σημείο με βάση το επιθυμητό μέγεθος των συνόλων εκπαίδευσης και δοκιμής: όλες οι εγγραφές πριν από το χρονικό σημείο χρησιμοποιούνται για την εκπαίδευση του μοντέλου και όλες οι υπόλοιπες εγγραφές χρησιμοποιούνται για δοκιμή. (Εάν είναι επιθυμητό, η γραμμή χρόνου θα μπορούσε να διαιρεθεί περαιτέρω για να δημιουργηθούν σύνολα επικύρωσης για την επιλογή παραμέτρων.) Για να αποτρέψουμε τυχόν εγγραφές στο σύνολο εκπαίδευσης να μοιράζονται χρονικά παράθυρα με τις εγγραφές στο σύνολο δοκιμής, αφαιρούμε τυχόν εγγραφές στο όριο -- σε αυτό περίπτωση, αγνοώντας δεδομένα αξίας 24 ωρών πριν από το χρονικό σημείο.

In [70]:

df_final

Out[70]:

mpNo7	scavAirFireDetTempNo8	...	draft_aft	draft_fore	stw	equivalent_teus	mid_draft	-
833333	39.566667	...	11.8000	11.700	20.661240	3680.000000	12.8257	-
833333	39.566667	...	11.8000	11.700	20.661240	3680.000000	12.8257	-
833333	39.566667	...	11.8000	11.700	20.661240	3680.000000	12.8257	-
833333	39.566667	...	11.8000	11.700	20.661240	3680.000000	12.8257	-
833333	39.566667	...	10.8000	10.780	20.661240	2967.000000	12.8257	-
...
316960	41.491814	...	13.9500	14.000	9.383299	3866.000000	11.2872	-
316960	41.491814	...	14.0875	14.125	5.077021	3866.000000	11.2872	-
316960	41.491814	...	14.5000	14.500	19.173250	8475.681667	11.2872	
316960	41.491814	...	14.5000	14.500	7.525299	8475.681667	11.2872	
316960	41.491814	...	14.5000	14.500	0.170414	8475.681667	11.2872	

Pycaret

In [65]:

```
from pycaret.classification import *
import pandas as pd
import time
import math
import seaborn as sns
import pandas as pd
import numpy as np
import scipy as sci
import plotly.offline as py
import plotly.graph_objs as go
py.init_notebook_mode(connected=True)

from imblearn.over_sampling import SMOTE

# from catboost import Pool

# from catboost import CatBoostClassifier
from sklearn.metrics import roc_auc_score, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.manifold import TSNE
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score

from matplotlib import pyplot as plt
plt.style.use('fivethirtyeight')
```

In [66]:

```
exp = setup(df_final,
target="label",
session_id=123,
numeric_imputation='mean',
normalize = True,
transformation = True,
ignore_low_variance = True,
remove_multicollinearity = True,
multicollinearity_threshold = 0.95,
log_experiment = True, experiment_name = 'predictive_maintenance'
)
```

	Description	Value
0	session_id	123
1	Target	label
2	Target Type	Binary
3	Label Encoded	None
4	Original Data	(2135, 101)
5	Missing Values	False
6	Numeric Features	100
7	Categorical Features	0
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(1494, 51)
12	Transformed Test Set	(641, 51)
13	Shuffle Train-Test	True
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	True
20	Experiment Name	'predictive_maintenance'
21	USI	988a
22	Imputation Type	simple
23	Iterative Imputation Iteration	None
24	Numeric Imputer	mean
25	Iterative Imputation Numeric Model	None
26	Categorical Imputer	constant
27	Iterative Imputation Categorical Model	None
28	Unknown Categoricals Handling	least_frequent

	Description	Value
29	Normalize	True
30	Normalize Method	zscore
31	Transformation	True
32	Transformation Method	yeo-johnson
33	PCA	False
34	PCA Method	None
35	PCA Components	None
36	Ignore Low Variance	True
37	Combine Rare Levels	False
38	Rare Level Threshold	None
39	Numeric Binning	False
40	Remove Outliers	False
41	Outliers Threshold	None
42	Remove Multicollinearity	True
43	Multicollinearity Threshold	0.950000
44	Remove Perfect Collinearity	True
45	Clustering	False
46	Clustering Iteration	None
47	Polynomial Features	False
48	Polynomial Degree	None
49	Trigonometry Features	False
50	Polynomial Threshold	None
51	Group Features	False
52	Feature Selection	False
53	Feature Selection Method	classic
54	Features Selection Threshold	None
55	Feature Interaction	False
56	Feature Ratio	False
57	Interaction Threshold	None
58	Fix Imbalance	False
59	Fix Imbalance Method	SMOTE

In [67]:

```
top3 = compare_models(n_select = 3)
/home/researcher/anaconda3/envs/py38/lib/python3.8/site-packages/sklearn/metrics/_classification.py:846: RuntimeWarning: invalid value encountered in double_scalars
    mcc = cov_ytyp / np.sqrt(cov_ytyt * cov_ypyp)
/home/researcher/anaconda3/envs/py38/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1221: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 due to no true samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/home/researcher/anaconda3/envs/py38/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1221: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(average, modifier, msg_start, len(result))
/home/researcher/anaconda3/envs/py38/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1464: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 due to no true nor predicted samples. Use `zero_division` parameter to control this behavior.
    _warn_prf(
/home/researcher/anaconda3/envs/py38/lib/python3.8/site-packages/sk
```

In [68]:

```
LABEL = 'label'
```

Όπως βλέπουμε λόγω ότι είναι μη ισορροπημένο το σύνολο των δεδομένων όλοι οι αλγόριθμοι δεν μπορούν να διακρίνουν τις περιόδους όπου είναι σε βλάβη το πλοίο. Το συγκεκριμένο Framework χρησιμοποιεί τη μέθοδο SMOTE για να αντιμετωπίσει την ανισορροπία των δεδομένων χωρίς επιτυχία.

In [72]:

```
train, test = train_test_split(df_final,stratify=df_final[LABEL], test_size=0.3, shuffle=True)
train, val = train_test_split(train,stratify=train[LABEL], test_size=0.3, shuffle=True)
```

In [74]:

```
cols_without_label = [x for x in df_final.columns if x not in ['label']]
```

In [75]:

```
y_train = train[LABEL]
train = train[cols_without_label]
y_val = val[LABEL]
val = val[cols_without_label]
y_test = test[LABEL]
test = test[cols_without_label]
```

In [89]:

```
from catboost import Pool
import shap

from catboost import CatBoostClassifier
```

In [100]:

```
model = CatBoostClassifier(iterations=2000, scale_pos_weight=1000000, learning_rate
                           use_best_model=True)
model.fit(train, y_train, logging_level='Verbose',
          plot=True, eval_set=(val, y_val))

fname = 'catboost'
model.save_model(fname, format="cbm")
```

32: learn: 0.9996069 test: 0.9997408 best: 0.9997527 (25)
total: 1.22s remaining: 1m 12s
33: learn: 0.9996069 test: 0.9997408 best: 0.9997527 (25)
total: 1.25s remaining: 1m 12s
34: learn: 0.9996069 test: 0.9997408 best: 0.9997527 (25)
total: 1.29s remaining: 1m 12s
35: learn: 0.9996069 test: 0.9997408 best: 0.9997527 (25)
total: 1.33s remaining: 1m 12s
36: learn: 0.9996069 test: 0.9997408 best: 0.9997527 (25)
total: 1.36s remaining: 1m 12s
37: learn: 0.9996069 test: 0.9997408 best: 0.9997527 (25)
total: 1.4s remaining: 1m 12s
38: learn: 0.9996056 test: 0.9997398 best: 0.9997527 (25)
total: 1.43s remaining: 1m 12s
39: learn: 0.9996056 test: 0.9997398 best: 0.9997527 (25)
total: 1.47s remaining: 1m 12s
40: learn: 0.9996056 test: 0.9997398 best: 0.9997527 (25)
total: 1.51s remaining: 1m 12s
41: learn: 0.9996043 test: 0.9997398 best: 0.9997527 (25)
total: 1.55s remaining: 1m 12s

In [101]:

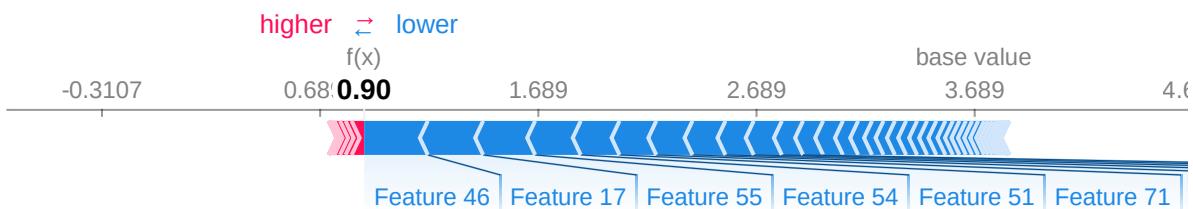
```
shap_values = model.get_feature_importance(Pool(val, label=y_val),
                                            type="ShapValue")
expected_value = shap_values[0, -1]
shap_values = shap_values[:, :-1]

# print(shap_values)

shap.initjs()
shap.force_plot(expected_value, shap_values[0])
```

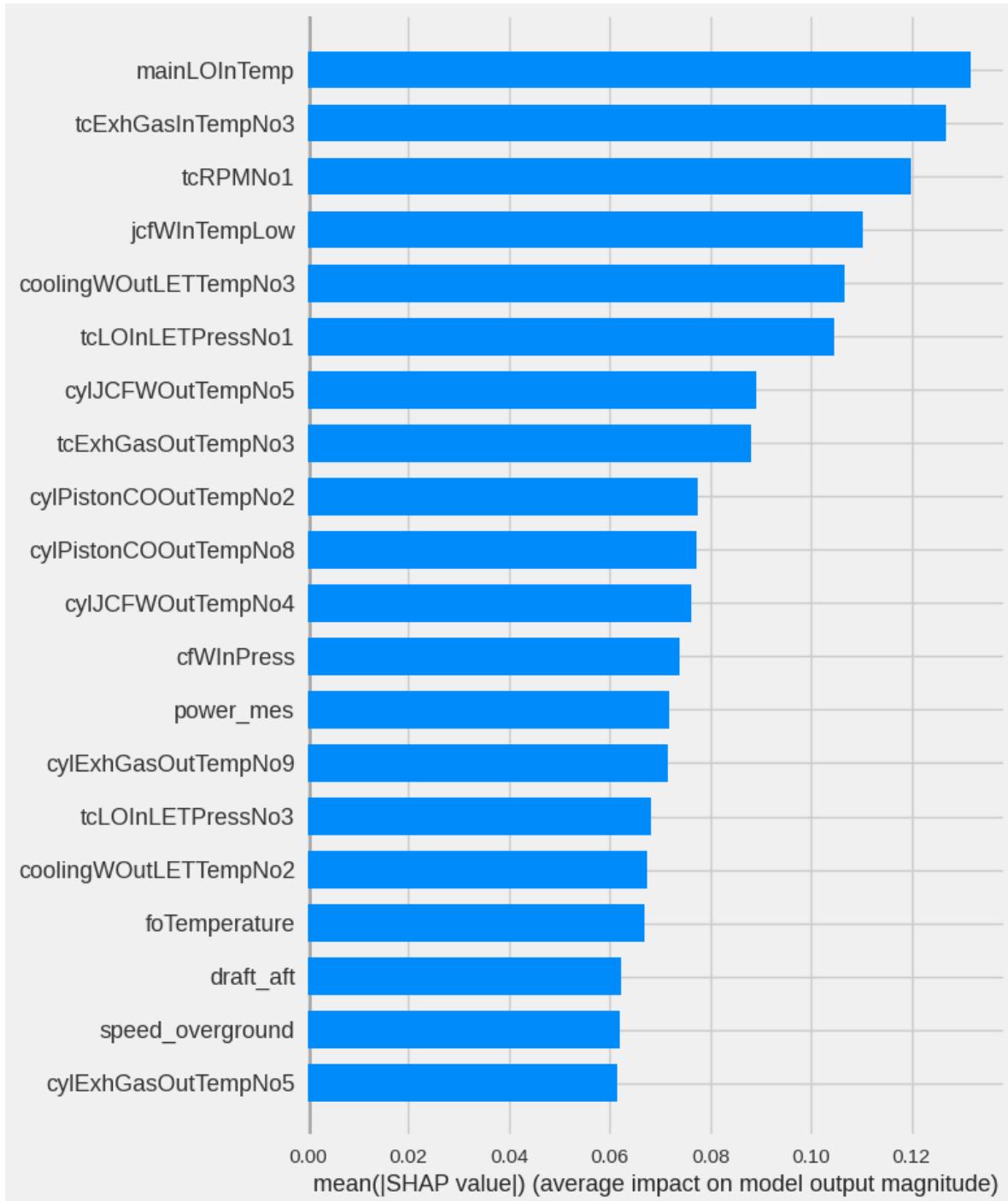


Out[101]:



In [102]:

```
# Then we convert the transposed shap values into a list, where each element will be  
shap.summary_plot(shap_values, features=train, class_names=y_train.unique(), plot_type="bar")
```



In [103]:

```
from sklearn.metrics import confusion_matrix  
y_preds = model.predict(test)  
cm_df = pd.DataFrame(confusion_matrix(y_test, y_preds))
```

In [104]:

```
cm_df
```

Out[104]:

	0	1
0	90	549
1	0	2

PERMUTATION ENTROPY

In [105]:

```

import itertools
import numpy as np
import pandas as pd
from scipy.spatial.distance import euclidean

def s_entropy(freq_list):
    ''' This function computes the shannon entropy of a given frequency distribution
    USAGE: shannon_entropy(freq_list)
    ARGS: freq_list = Numeric vector representing the frequency distribution
    OUTPUT: A numeric value representing shannon's entropy'''
    freq_list = [element for element in freq_list if element != 0]
    sh_entropy = 0.0
    for freq in freq_list:
        sh_entropy += freq * np.log(freq)
    sh_entropy = -sh_entropy
    return(sh_entropy)

def ordinal_patterns(ts, embdim, embdelay):
    ''' This function computes the ordinal patterns of a time series for a given embedding dimension
    USAGE: ordinal_patterns(ts, embdim, embdelay)
    ARGS: ts = Numeric vector representing the time series, embdim = embedding dimension
    OUTPUT: A numeric vector representing frequencies of ordinal patterns'''
    time_series = ts
    possible_permutations = list(itertools.permutations(range(embdim)))
    lst = list()
    for i in range(len(time_series) - embdelay * (embdim - 1)):
        sorted_index_array = list(np.argsort(time_series[i:(embdim+i)]))
        lst.append(sorted_index_array)
    lst = np.array(lst)
    element, freq = np.unique(lst, return_counts = True, axis = 0)
    freq = list(freq)
    if len(freq) != len(possible_permutations):
        for i in range(len(possible_permutations)-len(freq)):
            freq.append(0)
    return(freq)
else:
    return(freq)

def weighted_ordinal_patterns(ts, embdim, embdelay):
    time_series = ts
    possible_permutations = list(itertools.permutations(range(embdim)))
    temp_list = list()
    wop = list()
    for i in range(len(time_series) - embdelay * (embdim - 1)):
        Xi = time_series[i:(embdim+i)]
        Xn = time_series[(i+embdim-1): (i+embdim+embdim-1)]
        Xi_mean = np.mean(Xi)
        Xi_var = (Xi-Xi_mean)**2
        weight = np.mean(Xi_var)
        sorted_index_array = list(np.argsort(Xi))
        temp_list.append([''.join(map(str, sorted_index_array)), weight])
    result = pd.DataFrame(temp_list, columns=['pattern', 'weights'])
    freqlst = dict(result['pattern'].value_counts())
    for pat in (result['pattern'].unique()):
        wop.append(np.sum(result.loc[result['pattern']==pat, 'weights'].values))
    return(wop)

```

In [109]:

```
def p_entropy(ts, embdim=3, embdelay=1):
    ordinal_pat = weighted_ordinal_patterns(ts, embdim, embdelay)
    max_entropy = np.log(len(ordinal_pat))
    p = np.divide(np.array(ordinal_pat), float(sum(ordinal_pat)))
    return(s_entropy(p)/max_entropy)
```

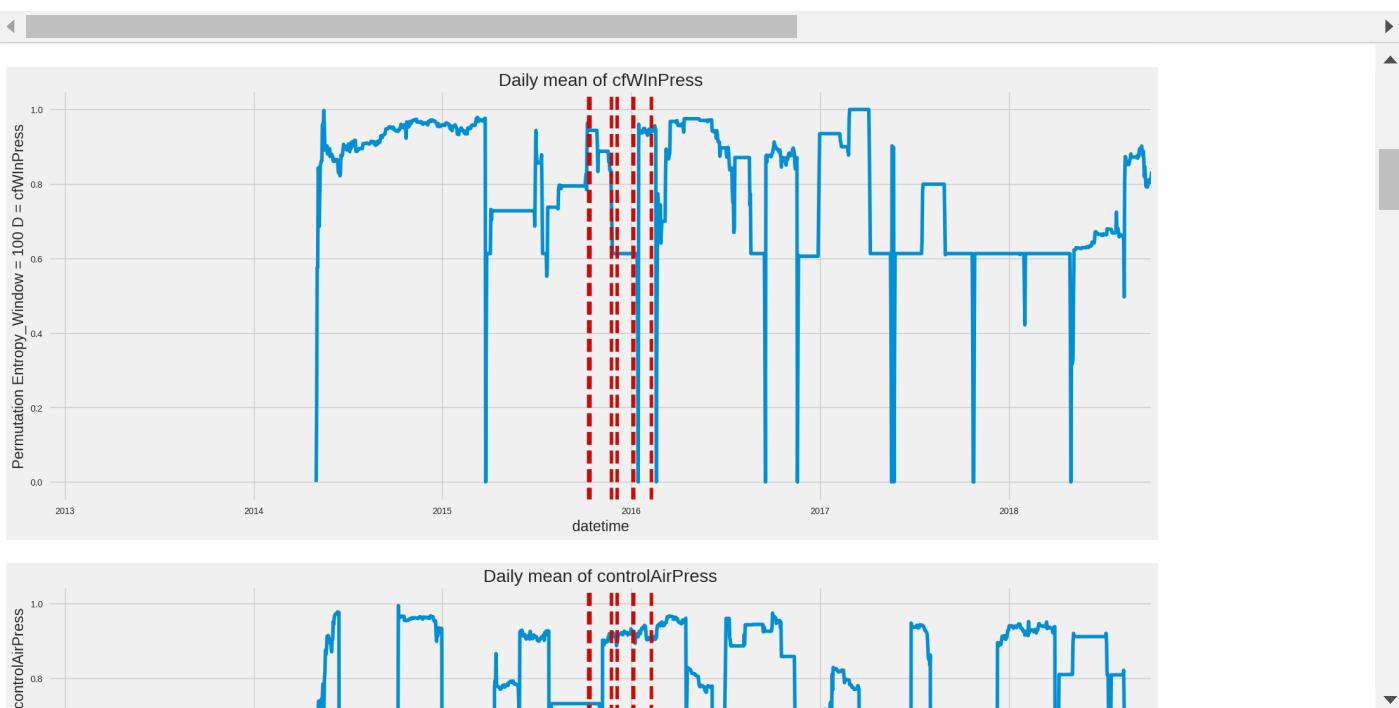
In [110]:

```
import matplotlib.pyplot as plt
import numpy as np

D=3
Window = 100
a = ['vessel_code', '2', '3', '4', '5', '6', '7', '8', '9', '10']
# a = ['vessel_code']

for column in mes_5.columns:
    if not any(x in column for x in a):

        rolling_pe = mes_5[column].rolling(Window).apply(p_entropy)
        data = rolling_pe
        ax = data.plot(figsize=(20,8))
        ax.set_ylabel("Permutation Entropy_Window = " + str(Window) + " D = " + str(D))
        ax.set_title("Daily mean of " + str(column))
#        xposition = [u'2016-05-02 00:00:00', u'2016-11-28 00:00:00', u'2017-06-21 00:
#                    u'2018-02-21 00:00:00', u'2018-02-21 00:00:00',
#                    u'2018-02-22 00:00:00']
#        xposition = [u'2015-10-11 00:00:00', u'2015-10-14 00:00:00', u'2015-11-24
#                     u'2016-01-06 00:00:00', u'2016-02-09 00:00:00']
        for xc in xposition:
            plt.axvline(x=xc, color='r', linestyle='--')
        plt.show()
```



In []:

RUL Prediction

RUL Labeling - Days Since Last Defect

Ένα κρίσιμο σύνολο δεδομένων σε αυτό το παράδειγμα είναι οι εγγραφές ελαττωμάτων που περιέχουν τις πληροφορίες των εγγραφών αντικατάστασης εξαρτημάτων. Οι πιο σχετικές πληροφορίες θα ήταν οι υπολογισμός του χρόνου που έχει περάσει από την τελευταία φορά που αντικαταστάθηκε ένα εξάρτημα, καθώς αυτό θα αναμενόταν να συσχετιστεί καλύτερα με τις αστοχίες εξαρτημάτων, καθώς όσο περισσότερο χρησιμοποιείται ένα εξάρτημα, τόσο μεγαλύτερη θα πρέπει να αναμένεται η υποβάθμιση. Στη συνέχεια, οι ημέρες από την τελευταία αντικατάσταση εξαρτήματος υπολογίζονται για κάθε τύπο εξαρτήματος ως χαρακτηριστικά από τα δεδομένα συντήρησης.

In [111]:

```
def remaining_useful_life(df):
    return len(df) - 1
```

In [116]:

```
df_final['label_rul']=0
count=0
for i in range(len(df_final)):
    if df_final['label'].iloc[i]==1:
        df_final['label_rul'].iloc[i]=0
        count=0
    else:
        df_final['label_rul'].iloc[i]=count
        count+=1
```

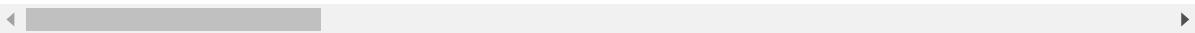
In [117]:

df_final

Out[117]:

	vessel_code_mes	airCoolerCWInLETPress	scavAirFireDetTempNo1	scavAirFireDetTempNo2
datetime				
2012-12-01	5.0	1.990000	39.750000	40.93
2012-12-02	5.0	1.990000	39.750000	40.93
2012-12-03	5.0	1.990000	39.750000	40.93
2012-12-04	5.0	1.990000	39.750000	40.93
2012-12-05	5.0	1.990000	39.750000	40.93
...
2018-10-01	5.0	1.998535	41.040162	41.66
2018-10-02	5.0	1.998535	41.040162	41.66
2018-10-03	5.0	1.998535	41.040162	41.66
2018-10-04	5.0	1.998535	41.040162	41.66
2018-10-05	5.0	1.998535	41.040162	41.66

2135 rows × 102 columns



DeepAR - A probabilistic Deep RNN

DeepAR, μια προσέγγιση DL που εφαρμόζει ένα μοντέλο που βασίζεται σε RNN, κοντά σε αυτό που περιγράφεται στο

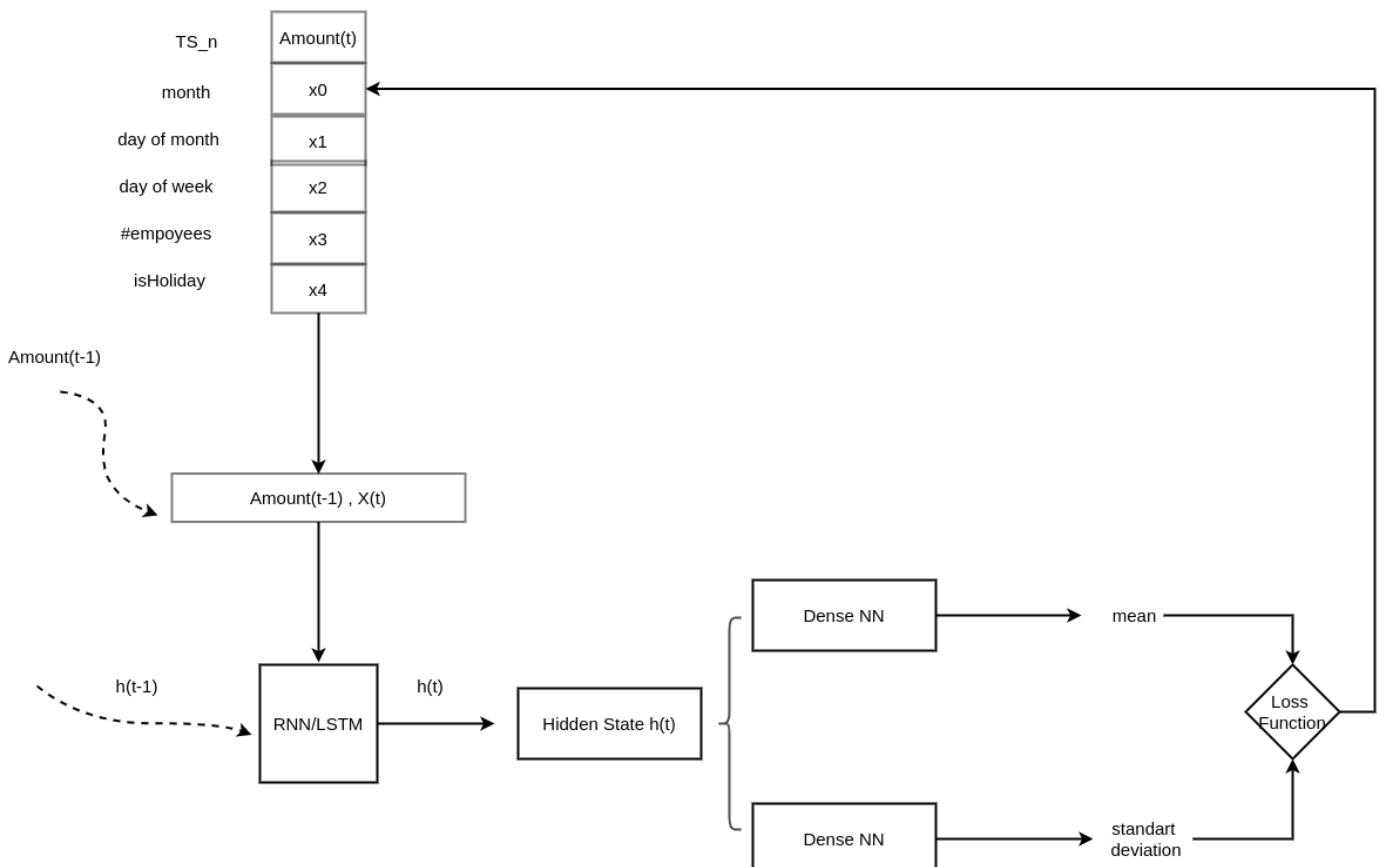
Salinas, D., Flunkert, V., Gasthaus, J., Januschowski, T.: DeepAR: Probabilistic forecasting with autoregressive recurrent networks. International Journal of Forecasting (2019)

,επιλέχθηκε ως η καταλληλότερη, προερχόμενη από τον ανοιχτό κώδικα του GluonTS toolkit <https://ts.gluon.ai/> (<https://ts.gluon.ai/>) .

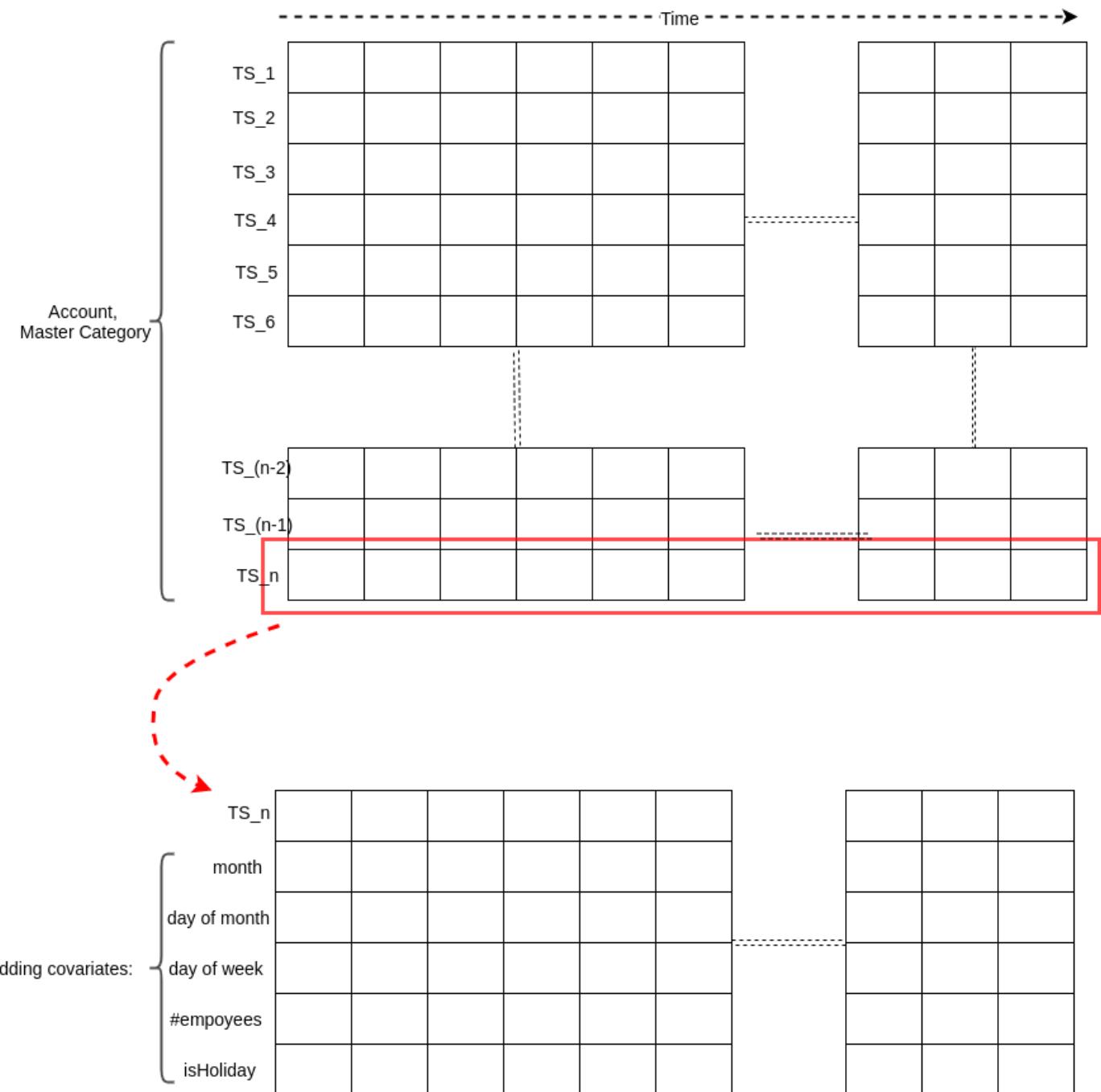
Πιο αναλυτικά, το επιλεγμένο μοντέλο εφαρμόζει μια μεθοδολογία για την παραγωγή ακριβών πιθανοτικών προβλέψεων, βασισμένη στην εκπαίδευση ενός μοντέλου αυτοπαλινδρομικού επαναλαμβανόμενου νευρωνικού δικτύου σε πολλές σχετικές χρονοσειρές.

Τα RNN έχουν την έννοια της «μνήμης» που τους βοηθά να αποθηκεύουν τις καταστάσεις ή τις πληροφορίες προηγούμενων εισόδων για να δημιουργήσουν την επόμενη έξοδο της ακολουθίας. Το RNN που προβλέπει τον μέσο όρο και τη διακύμανση των υποκείμενων χρονοσειρών, συνδυάζεται με μια προσομοίωση Monte Carlo που αποδίδει αποτελέσματα που αντιπροσωπεύονται ως κατανομή. Η κύρια αρχιτεκτονική του DeepAR απεικονίζεται στην παρακάτω εικόνα όπου τα κύρια βήματα για την πρόβλεψη $z(t+1)$ μπορούν να συνοψιστούν σε

- πρώτα μια γκαουσιανή κατανομή δημιουργήθηκε με χρήση των $\mu(t)$ και $\sigma(t)$,
- από αυτήν την κατανομή αντλούνται η δείγματα, η διάμεσος αυτών των δειγμάτων ορίζεται σε $z'(t)$.
- Το $z'(t)$ μαζί με τις τρέχουσες γνωστές συμμεταβλητές $x(t+1)$ και την προηγούμενη κρυφή κατάσταση $h(t)$ τροφοδοτούνται στο εκπαίδευμένο μοντέλο LSTM που βγάζει $h(t+1)$.
- Τα $\mu(t+1)$ και $\sigma(t+1)$ λαμβάνονται χρησιμοποιώντας $h(t+1)$, χρησιμοποιώντας τα νέα $\mu(t+1)$ και $\sigma(t+1)$ δημιουργείται μια γκαουσιανή κατανομή από την οποία η δείγματα επιλεγμένο



Moreover, the chosen model learns seasonal behavior patterns from the given covariates that strengthens its forecasting capabilities. As expected, while configuring the model to our forecasting needs, various long-established challenges regarding time-series forecasting arose.



Μια καλή περιγραφή του μοντελου και του τρόπου λειτυργίας του βρίσκεται στον ακόλουθο σύνδεσμο <https://kshavg.medium.com/deepar-probabilistic-forecasting-with-autoregressive-recurrent-networks-fa65ddd1f5> (<https://kshavg.medium.com/deepar-probabilistic-forecasting-with-autoregressive-recurrent-networks-fa65ddd1f5>)

In [134]:

```
df_final['month'] = df_final.index.month
df_final['day_of_month'] = df_final.index.day
df_final['day_of_week'] = df_final.index.dayofweek
```

In [136]:

```
df_final.drop(['label'], axis=1, inplace=True)
```

In [137]:

```
FREQ="1D"

start_date = df_final.index[0]
end_date = df_final.index[-1]

print(start_date)
print(end_date)
index = pd.date_range(start=start_date, end=end_date, freq=FREQ)
```

```
2012-12-01 00:00:00
2018-10-05 00:00:00
```

In [138]:

```
dataframes = []
filenames = []
```

In [139]:

```
dataframes.append(df_final)
```

In [140]:

```
N=len(dataframes)
T = len(dataframes[0])
starts = []
print('Number of Timeseries :'+ str(N))
print('Number of Samples in each ts :'+str(T))
print('Freq : '+ FREQ)
```

```
Number of Timeseries :1
Number of Samples in each ts :2135
Freq : 1D
```

In [141]:

```
custom_dataset = dataframes
```

In [142]:

```
PREDICTION_LENGTH = 12
CONTEXT_LENGTH = 30

custom_ds_metadata = {'num_series': N,
                     'prediction_length': PREDICTION_LENGTH,
                     'context_length': CONTEXT_LENGTH,
                     'freq': FREQ
                    }
```

In [152]:

```
from gluonts.dataset import common
from gluonts.model import deepar
from gluonts.mx import Trainer
# from gluonts.distribution.piecewise_linear import PiecewiseLinearOutput
from gluonts.evaluation import make_evaluation_predictions

# TODO add all the dynamic features + is_holiday etc + number of employees
train_ds = common.ListDataset([{'target': custom_dataset[i].label_rul[:]-custom_ds_m

    for i in range(N)],
    freq=custom_ds_metadata['freq'])

# test dataset: use the whole dataset, add "target" and "start" fields
test_ds = common.ListDataset([{'target': custom_dataset[i].label_rul[:], 'start': c

    for i in range(N)],
    freq=custom_ds_metadata['freq'])
```

In [153]:

```
estimator_dare = deepar.DeepAREstimator(
    prediction_length=custom_ds_metadata['prediction_length'],
    context_length=custom_ds_metadata['context_length'],
    freq=custom_ds_metadata['freq'],
    trainer=Trainer(epochs=5,
                    ctx="cpu",
                    num_batches_per_epoch=50,
                    ),
    use_feat_dynamic_real=True,
    dropout_rate=0.5
)
```

In [154]:

```
predictor_dare = estimator_dare.train(train_ds)
```

```
100%|██████████| 50/50 [00:04<00:00, 12.20it/s, epoch=1/  
5, avg_epoch_loss=5.47]  
100%|██████████| 50/50 [00:04<00:00, 12.36it/s, epoch=2/  
5, avg_epoch_loss=4.59]  
100%|██████████| 50/50 [00:04<00:00, 12.32it/s, epoch=3/  
5, avg_epoch_loss=4.39]  
100%|██████████| 50/50 [00:04<00:00, 11.91it/s, epoch=4/  
5, avg_epoch_loss=4.18]  
100%|██████████| 50/50 [00:03<00:00, 12.74it/s, epoch=5/  
5, avg_epoch_loss=4.06]
```

In [155]:

```
def make_forecasts(predictor, test_data, n_sampl):  
    """Takes a list of predictors, gluonTS test data and number of samples  
    and returns forecasts for each of them"""  
    forecasts = []  
    tss = []  
    forecast_it, ts_it = make_evaluation_predictions(  
        dataset=test_ds,  
        predictor=predictor,  
        num_samples=n_sampl  
    )  
    forecasts = list(forecast_it)  
    tss = list(ts_it)  
    return forecasts, tss
```

In [156]:

```
forecasts, tss = make_forecasts(predictor_dare, test_ds, 100)
```

In [162]:

forecasts

Out[162]:

```
[gluonts.model.forecast.SampleForecast(info=None, item_id=None, samples=numpy.array([[169.51109313964844, 161.31166076660156, 168.26075744628906, 172.3036346435547, 179.79859924316406, 190.7381591796875, 131.4390106201172, 180.16659545898438, 182.67869567871094, 156.91452026367188, 178.16091918945312, 175.68112182617188], [164.07479858398438, 168.38870239257812, 166.0625457763672, 164.91363525390625, 160.2041015625, 164.54847717285156, 174.82415771484375, 185.95999145507812, 177.38381958007812, 178.53607177734375, 156.49806213378906, 177.39791870117188], [169.8887176513672, 156.74356079101562, 167.40086364746094, 166.43162536621094, 179.35757446289062, 176.2193145751953, 160.46958923339844, 169.34500122070312, 176.74688720703125, 153.3632354736328, 178.583984375, 168.25762939453125], [160.895263671875, 176.40269470214844, 153.61465454101562, 164.0616912841797, 164.48045349121094, 162.95323181152344, 172.34756469726562, 173.96119689941406, 172.0843963623047, 177.84364318847656, 169.21490478515625, 185.34231567382812], [160.8893585205078, 170.5131378173828, 168.0563507080078, 174.5875244140625, 184.63694763183594, 150.4141387939453, 164.89125061035156. 169.4771728515625. 167.99522399902344. 170.149200439453
```

In [178]:

```
def plot_prob_forecasts(ts_entry, forecast_entry):
    plot_length = 60
    prediction_intervals = (50.0, 90.0)
    legend = ["observations", "median prediction"] + [
        f"{k}% prediction interval" for k in prediction_intervals
    ][:-1]

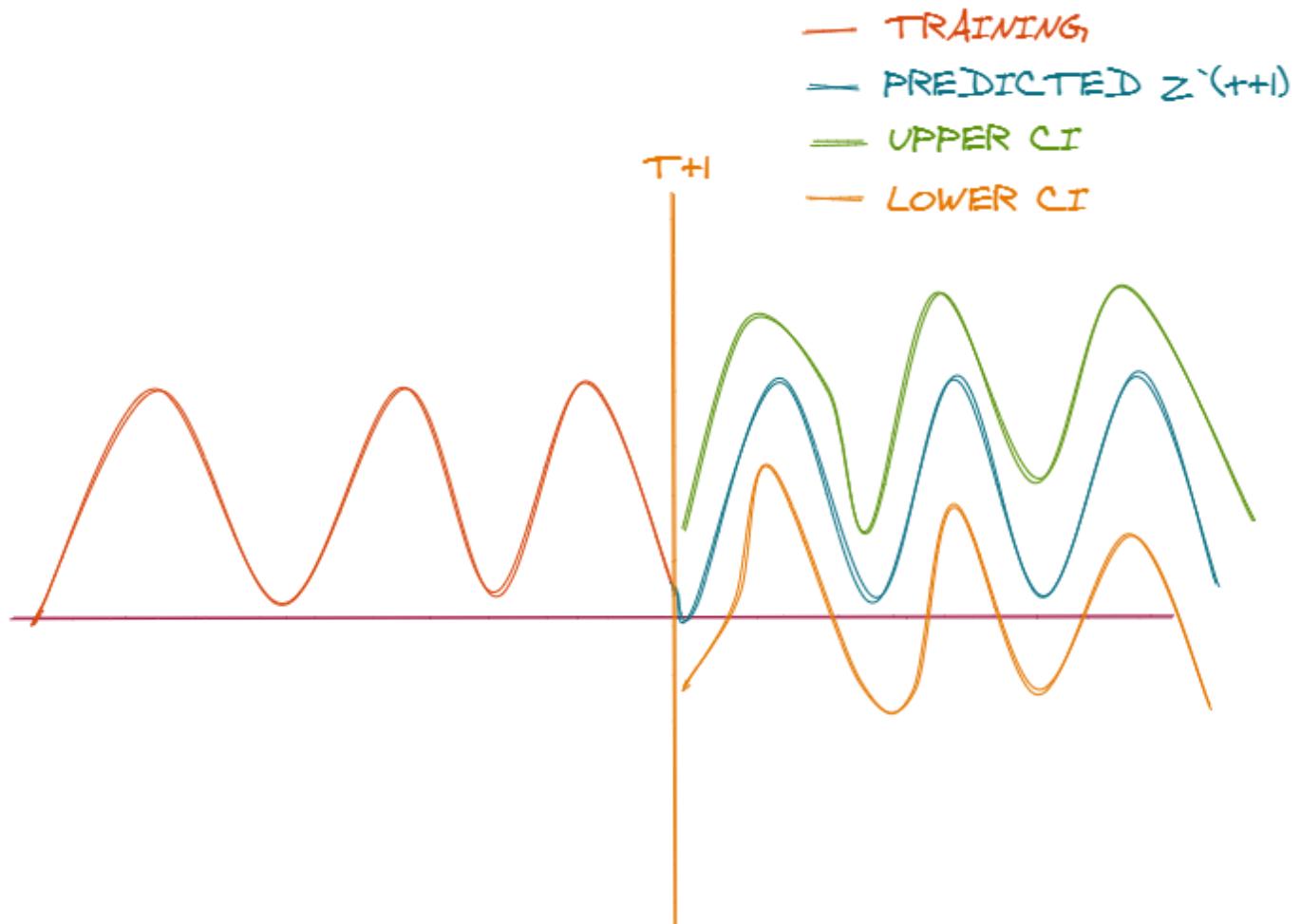
    fig, ax = plt.subplots(1, 1, figsize=(10, 7))
    ts_entry[-plot_length:].plot(ax=ax) # plot the time series
    forecast_entry.plot(prediction_intervals=prediction_intervals, color="g")
    plt.grid(which="both")
    plt.legend(legend, loc="upper left")
    plt.show()
```

Μια βασική ιδέα στο GluonTS είναι ότι δεν παράγουμε απλές τιμές ως προβλέψεις, αλλά στην πραγματικότητα προβλέπουμε διανομές.

Ένας διαισθητικός τρόπος για να το δούμε αυτό είναι να φανταστούμε την πρόβλεψη μιας χρονοσειράς 100 φορές, η οποία επιστρέφει 100 διαφορετικά δείγματα χρονοσειρών, τα οποία σχηματίζουν μια κατανομή γύρω τους. Εκτός από το ότι μπορούμε να εκπέμψουμε απευθείας αυτές τις κατανομές και μετά να αντλήσουμε δείγματα από αυτές.

Οι διανομές παρέχουν το όφελος ότι παρέχουν μια σειρά πιθανών τιμών. Φανταστείτε ότι είστε ένα εστιατόριο που αναρωτιέστε πόσα υλικά να αγοράσετε. Εάν αγοράζουμε πολύ λίγα, δεν θα εξυπηρετήσουμε τη ζήτηση των πελατών, αλλά η αγορά πολλών θα παράγει απόβλητα. Έτσι, όταν προβλέπουμε τη ζήτηση, είναι πολύτιμο εάν ένα μοντέλο μπορεί να μας πει ότι υπάρχει πιθανώς ζήτηση για ας πούμε 50 πιάτα, αλλά απίθανο περισσότερα από 60.

- For a 95% confidence interval set $p = 2$
- Upper Confidence Interval = $z'(t+1) + 2 * ci(t+1)$
- Lower Confidence Interval = $z'(t+1) — 2 * ci(t-1)$



Made with Excalidraw

In [179]:

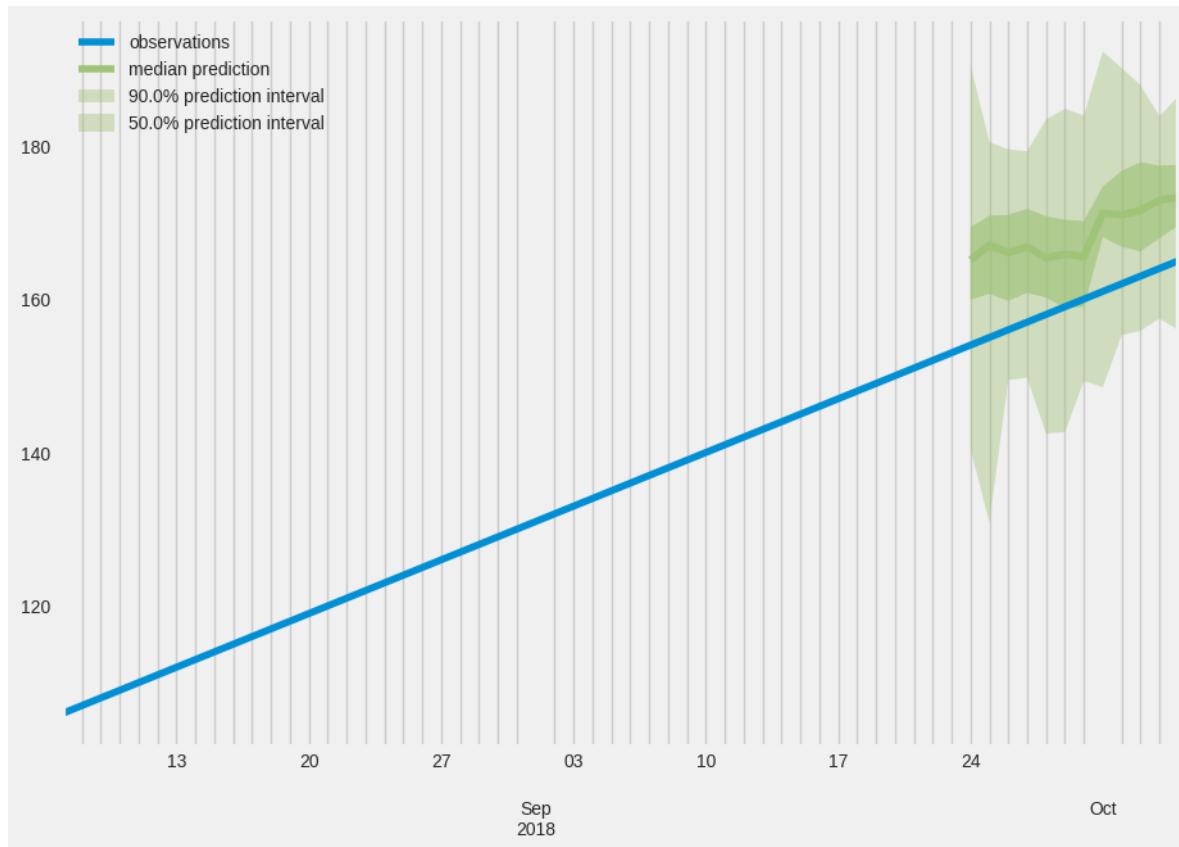
```
forecasts = list(forecasts)
tss = list(tss)
```

In [180]:

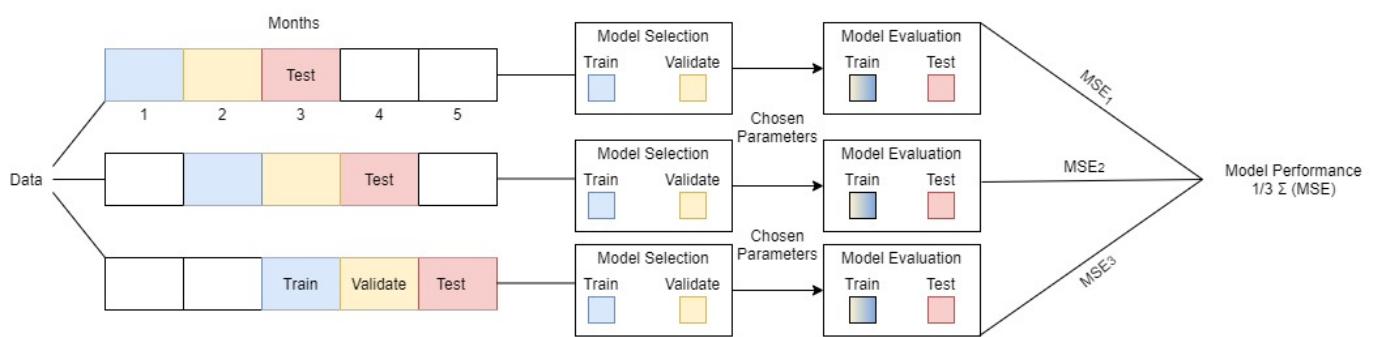
```
ts_entry = tss[0]
forecast_entry = forecasts[0]
```

In [181]:

```
plot_prob_forecasts(ts_entry, forecast_entry)
```



Όσον αφορά το σχήμα αξιολόγησης του μοντέλου που παρουσιάζεται, δεδομένου ότι οι μέθοδοι διασταυρούμενης επικύρωσης αντικατοπτρίζουν μια παγίδα στα σενάρια πρόβλεψης χρονοσειρών, καθώς μπορεί να οδηγήσουν σε σημαντική επικάλυψη μεταξύ αμαξοστοιχίας και δεδομένων δοκιμής, η βέλτιστη προσέγγιση είναι η προσομοίωση μοντέλων σε "βόλτα προς τα εμπρός". », επανεκπαιδεύοντας περιοδικά το μοντέλο ώστε να ενσωματώνει συγκεκριμένα κομμάτια δεδομένων συναλλαγών που είναι διαθέσιμα εκείνη τη στιγμή.



In []: