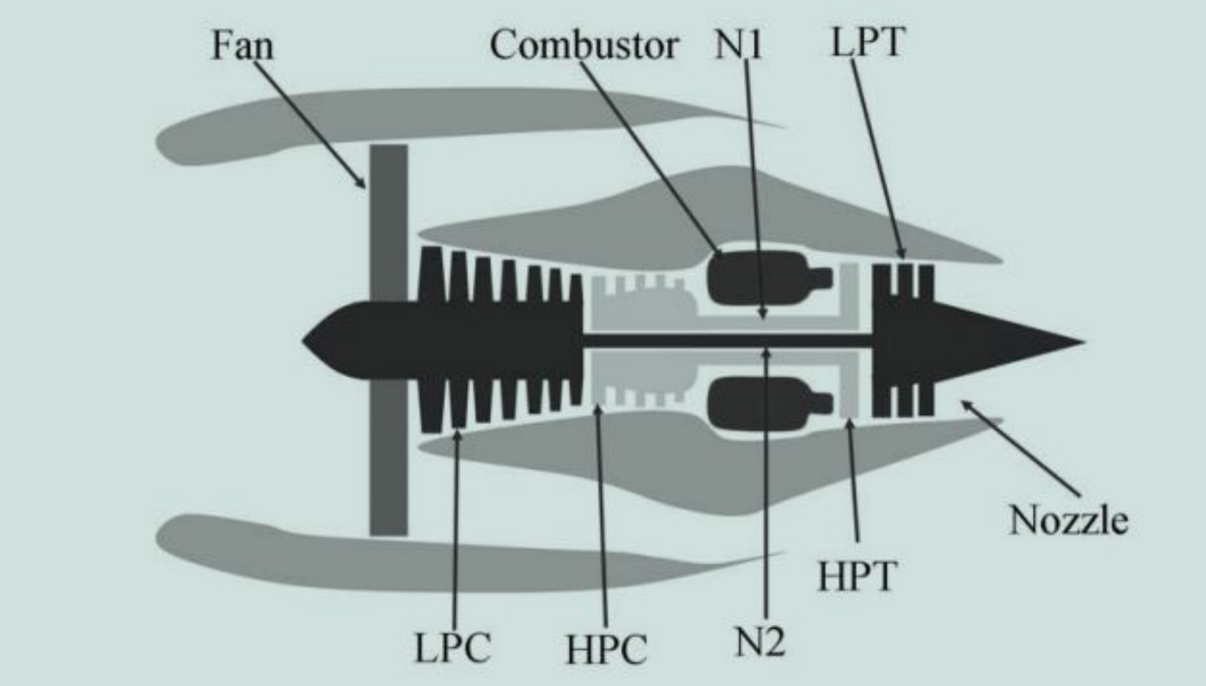


NASA TURBOFAN JET ENGINE FAILURE PREDICTION MODELING

AMII Machine Learning Technician I - Capstone Project
Team 3
December 11, 2020

Team

Uchenna Mgbaja
from Enugu, Nigeria
College Instructor at Reeves College, Edmonton.
M.Sc. Petroleum Engineering

Ekin Atinc
from Istanbul, Turkey
Avid reader, hiker, footballer
Amateur chess player
Work in investment management

Alejandro Coy
from Bogota, Colombia
Amateur Triathlete, M.Sc Chemical Engineer, future Data Scientist(I hope:)!)
Work in Oil & Gas as Process Engineer at Rangeland Engineering.

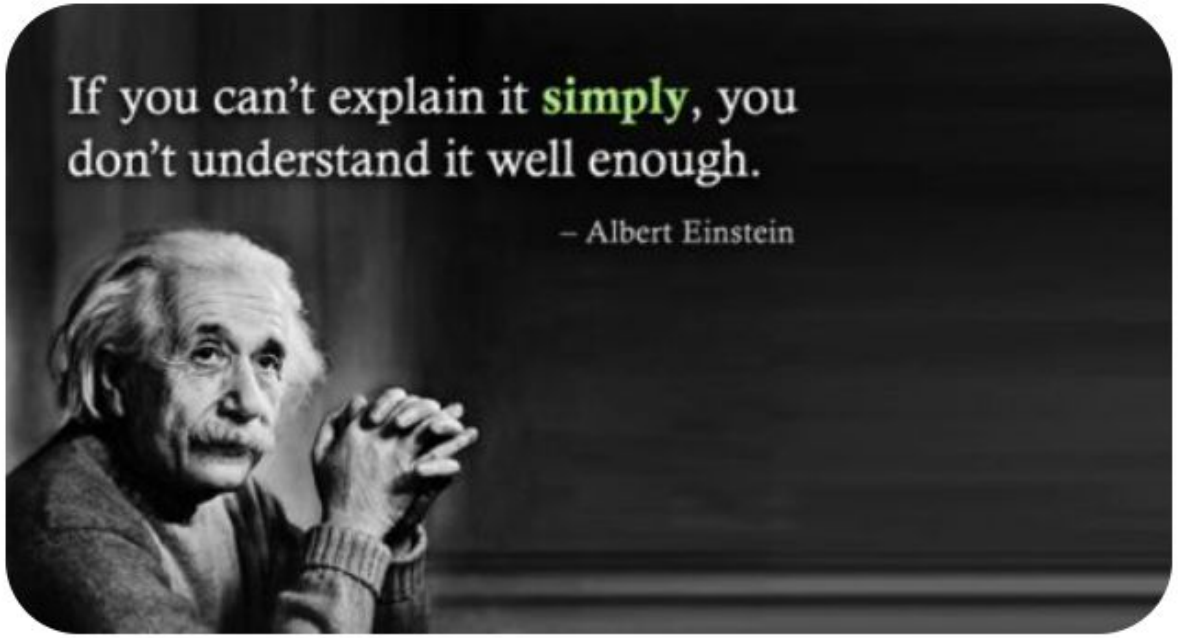
Raihan Khan
from Dhaka, Bangladesh
FC Barcelona fan
Hiker/backpacker/footballer
Amateur photographer
Work in oil & gas to pay bills
MBA Finance, BCom &
now AI/ML enthusiast

Agenda

- Problem & framing
- Exploratory data analysis
- Data cleaning, preprocessing, feature engineering
- Model development, tuning, optimization, evaluation
- Deployment and value generation

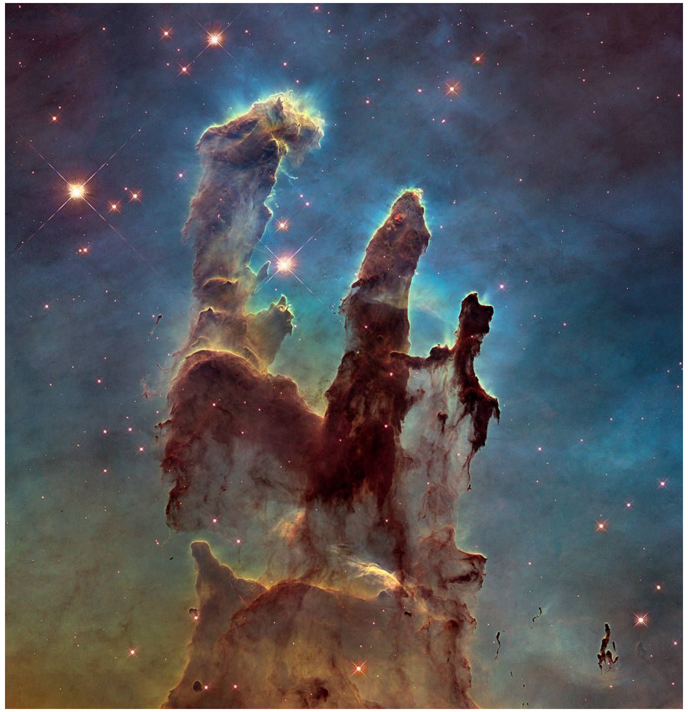
https://hbr.org/2018/10/3-common-mistakes-that-can-derail-your-teams-predictive-analytics-efforts

Problem & framing



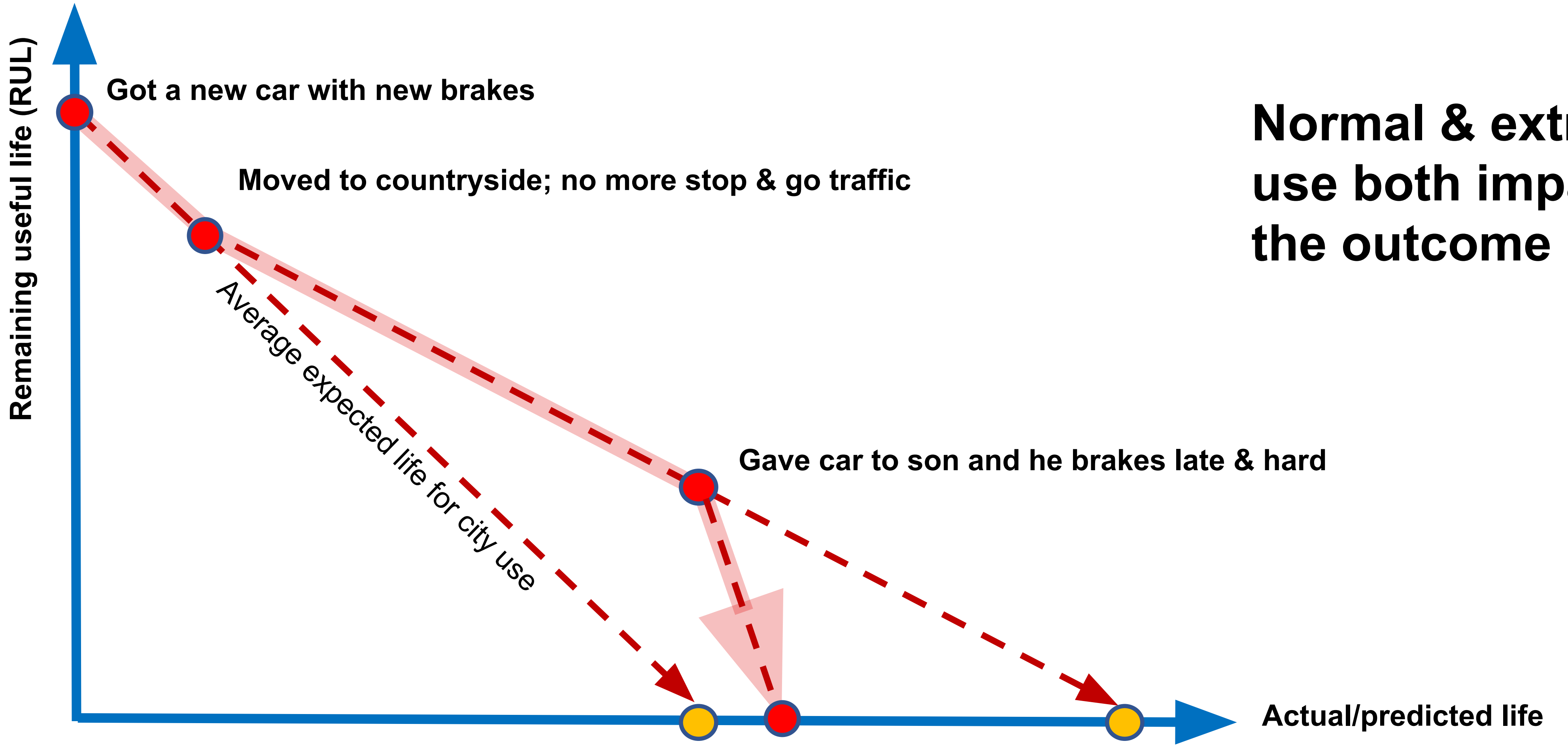
Business problem

- NASA wants a model that can accurately predict remaining useful life (RUL) of a class of turbofan jet engines
  - Have data on 3 settings and 20 sensors for 709 engines to run to failure under six different conditions
- A successful model could reduce maintenance costs
  - Switching to condition-based maintenance, or,
  - Extending current planned maintenance intervals
- NASA prefers to lean towards early failure prediction
  - From which we glean that consequence of failure is high



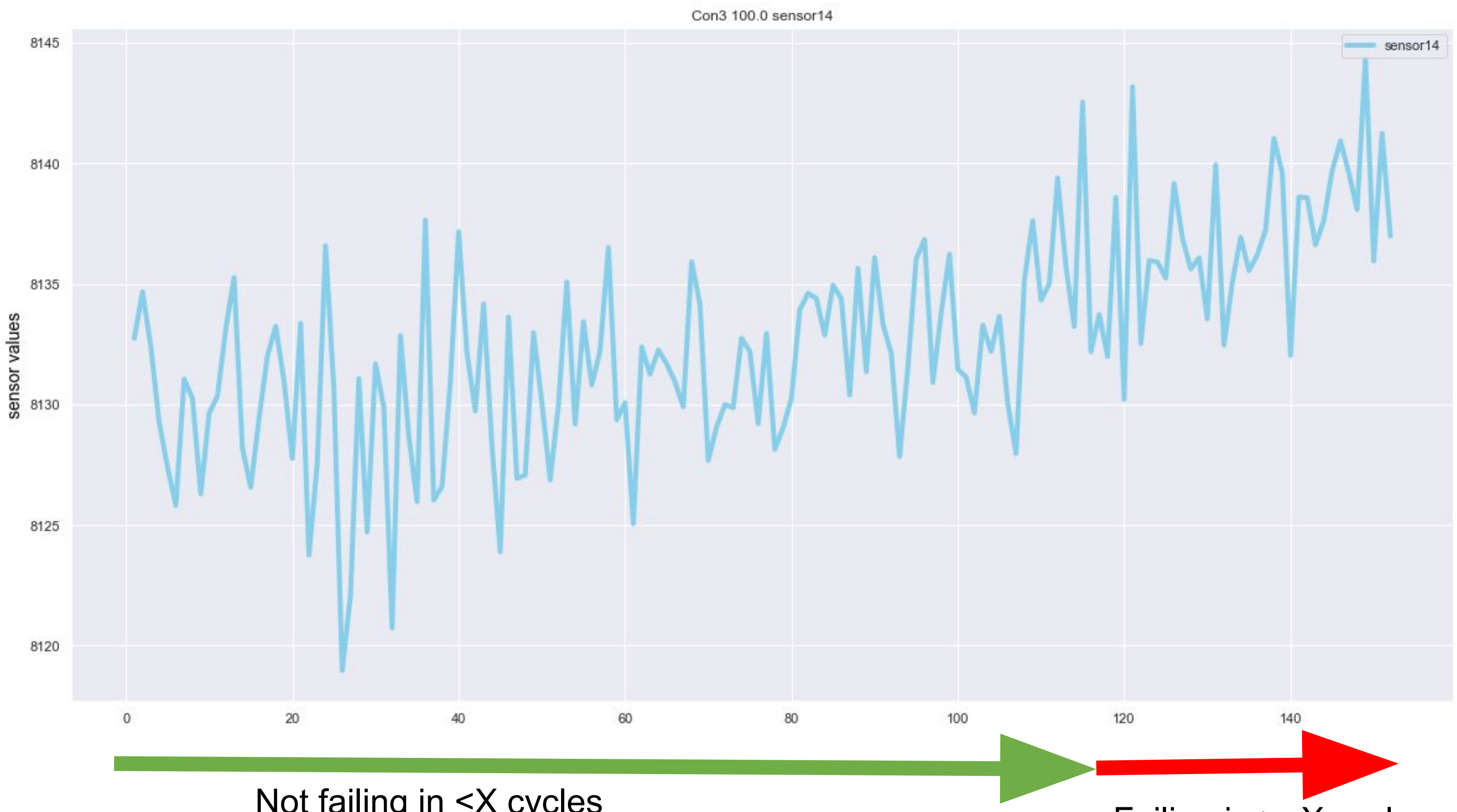
Credits: NASA, ESA and the Hubble Heritage Team (STScI/AURA)

Failure prediction is about modeling degradation



Normal & extreme use both impact the outcome

Framing the hypothesis



Not failing in <X cycles

Failing in >=X cycles

Approach: Classification

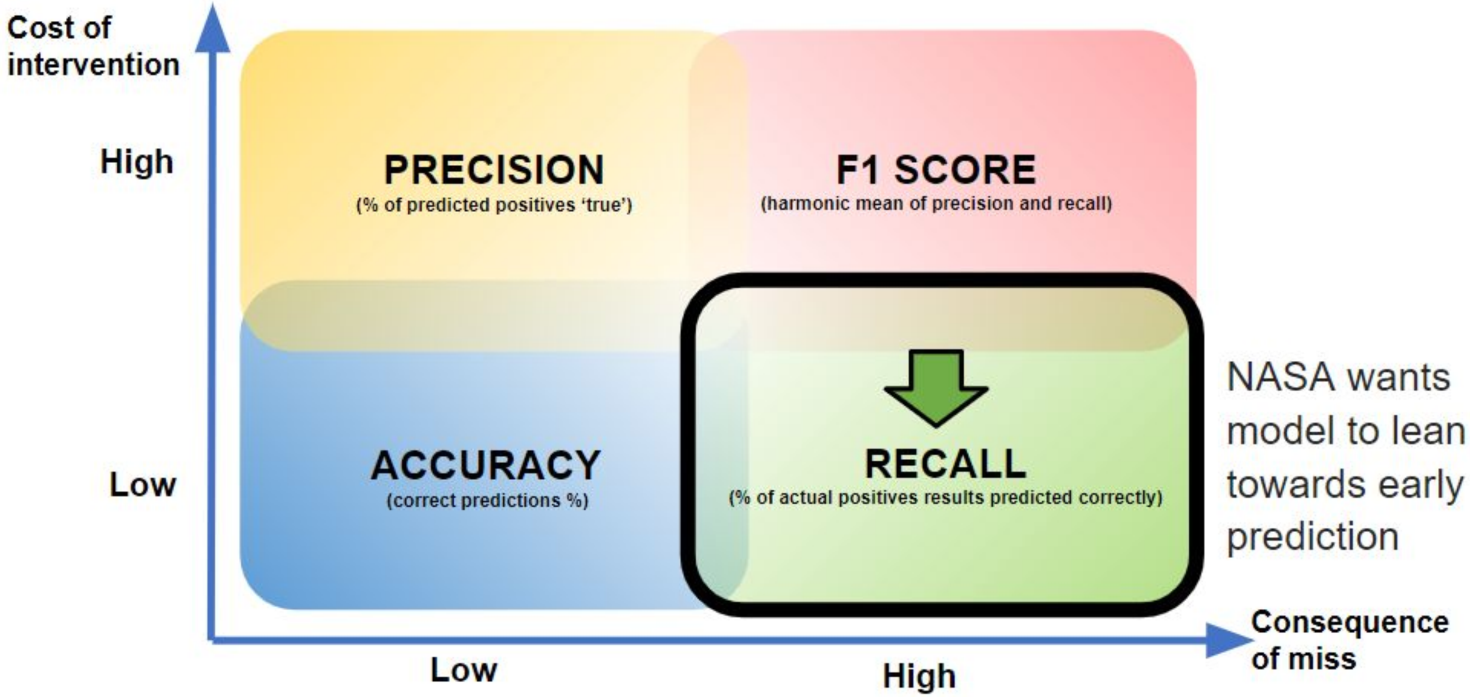
Q: Will an engine fail in next X cycles?

A: Yes, or No

Action: if Yes, trigger maintenance

Value: Prevent

Performance criteria



NASA wants model to lean towards early prediction

Lack of upfront customer alignment on what a predictive model will deliver is the lead reason why many ML efforts fail

Customer agreement

Current maintenance interval is 5 cycles.

If we can predict failures within 25 cycles to a RECALL of 98% NASA will stretch intervals to 10 cycles saving $25M per year.

If a RECALL of 99.9% is achieved NASA might consider a pure condition-based approach and eliminate program maintenance. NASA saves $100M, and we get a bonus.

https://towardsdatascience.com/what-is-the-main-reason-most-ml-projects-fail-515d409a161f

Limitations

- Not real failure data (from simulator)*
- Simulating a business condition & customer agreement
- Not all ML algorithms tried (NN, XGBoost, SVM, etc)

** High consequence of failures lead to many industries never allowing equipment to fail. With very few failures, it becomes incredibly difficult to build reliable predictive models.*

https://www.mckinsey.com/business-functions/operations/our-insights/predictive-maintenance-the-wrong-solution-to-the-right-problem-in-chemicals#

Data analysis and preparation

Description of data

- Learning data is provided as 4 separate outputs ranging from training set 1 to 4 where each dataset correspond to measurement of engine performance under various 6 conditions.
- Learning data is generated from a simulated environment with 14 inputs that measure the health of the engine and consists of numerical data types comprised of floats and integers only.
- Engine's health is calculated by a "Health Index" which measures how far the engine is operating from various pre-defined operational limits.
- Operational limits are specified on the engine's 5 rotating components of , Fan, Low Pressure Compressor (LPC), High Pressure Compressor (HPC), High Pressure Turbine (HPT), and Low-Pressure Turbine (LPT). These parameters change as a function of conditions such as Exhaust Gas Temperature (EGT), Throttle resolver angle (TRA) , and Altitude
- Features that help measuring the stress levels in the 5 main components are T2 (temp at fan inlet), T24 (temp at LPC outlet), T30 (temp at HPC outlet) , T50 (temp at LPT outlet) and related variables to these components
- Identified few extreme values as any sensor value above and beyond 2 std.dev. but didn't remove them

Description of data

Schematic View of the Inputs & Outputs

Inputs

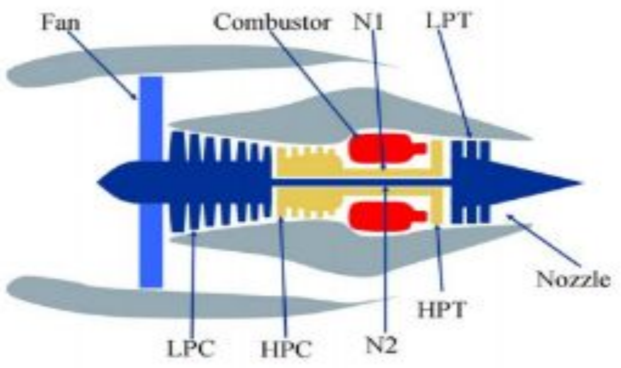
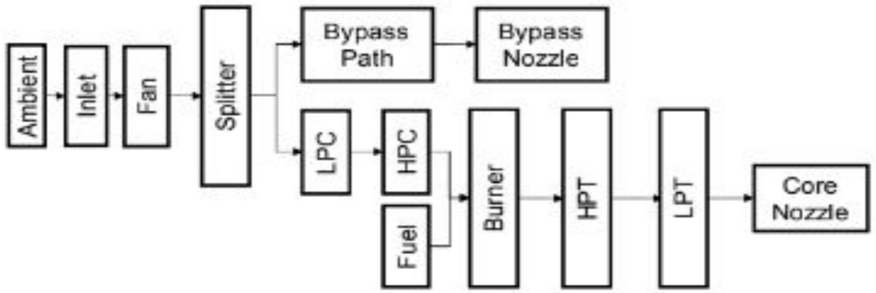


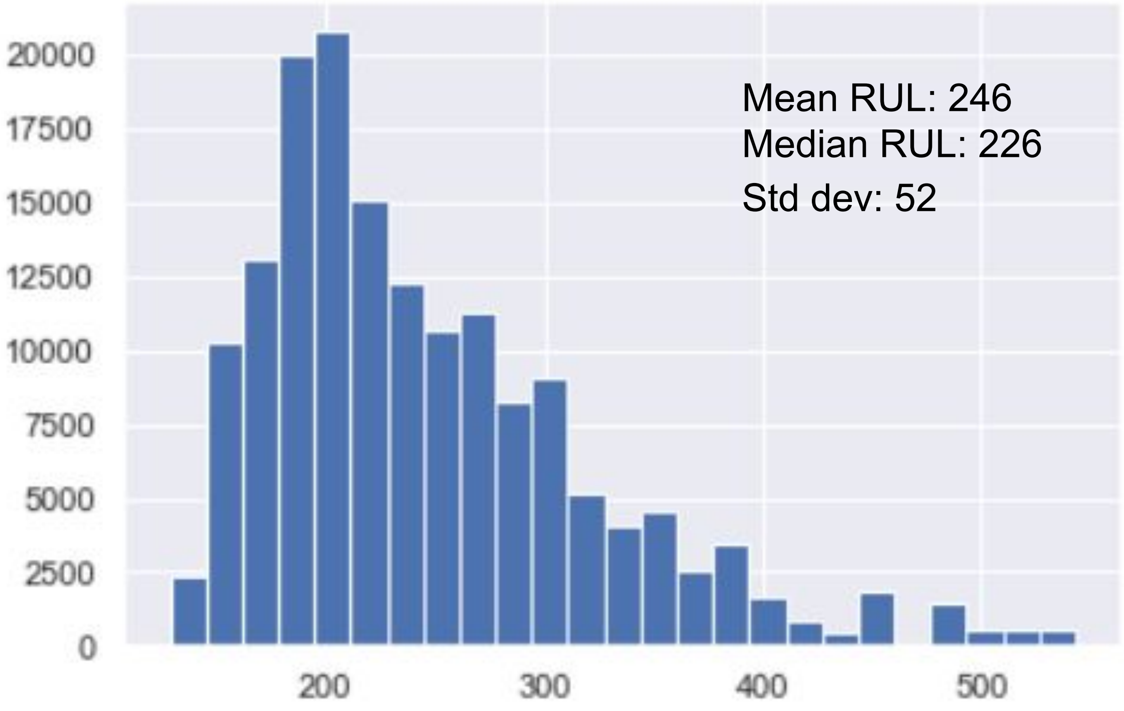
Figure 1. Simplified diagram of engine simulated in C-MAPSS [11].



Feature Set

Symbol	*Description*	*Units*
Parameters available to participants as sensor data		
T2	Total temperature at fan inlet	°R
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outlet	°R
T50	Total temperature at LPT outlet	°R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio (P50/P2)	--
Ps30	Static pressure at HPC outlet	psia
phi	Ratio of fuel flow to Ps30	pps/psi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	--
farB	Burner fuel-air ratio	--
htBleed	Bleed Enthalpy	--
Nf_dmd	Demanded fan speed	rpm
PCNfR_dmd	Demanded corrected fan speed	rpm
W31	HPT coolant bleed	lbm/s
W32	LPT coolant bleed	lbm/s

Distribution of max life cycles



The distribution is Gaussian … but with wide range of values for maximum life achieved by engines

Data preprocessing

- **Step 1:** Combine all four of the provided training datasets to create a 160,359 by 31 sample size. For each training set appended, add a new column called "condition"
  - We believe that the QuAM would be more robust on a dataset with engine information under various conditions

	Unit number	Time in cycles	Altitude	Mach number	TRA	T2	T4	T30	T50	P2	...	farB	htBleed	Nf_dmd	PCNfR_dmd	W31	W32	condition
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	0.03	392	2388	100.0	39.06	23.4190	1
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	0.03	392	2388	100.0	39.00	23.4236	1
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	0.03	390	2388	100.0	38.95	23.3442	1
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	0.03	392	2388	100.0	38.88	23.3739	1
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	0.03	393	2388	100.0	38.90	23.4044	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
61244	249	251	9.9998	0.2500	100.0	489.05	605.33	1516.36	1315.28	10.52	...	0.03	372	2319	100.0	29.11	17.5234	4
61245	249	252	0.0028	0.0015	100.0	518.67	643.42	1598.92	1426.77	14.62	...	0.03	396	2388	100.0	39.38	23.7151	4
61246	249	253	0.0029	0.0000	100.0	518.67	643.68	1607.72	1430.56	14.62	...	0.03	395	2388	100.0	39.78	23.8270	4
61247	249	254	35.0046	0.8400	100.0	449.44	555.77	1381.29	1148.18	5.48	...	0.02	337	2223	100.0	15.26	9.0774	4
61248	249	255	42.0030	0.8400	100.0	445.00	549.85	1369.75	1147.45	3.91	...	0.02	333	2212	100.0	10.66	6.4341	4

160359 rows × 30 columns

- **Step 2:** Derive the target "RUL" by
  1. Find the maximum time cycle of each unit
  2. Append to the dataframe by merging on unit number
  3. RUL = Max Cycle – Time in Cycles

	Unit number	Time in cycles	Altitude	Mach number	TRA	T2	T4	T30	T50	P2	...	BPR	farB	htBleed	Nf_dmd	PCNfR_dmd	W31	W32	condition	Max cycles	target
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	8.4195	0.03	392	2388	100.0	39.06	23.4190	1	192	191
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	8.4318	0.03	392	2388	100.0	39.00	23.4236	1	192	190
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	8.4178	0.03	390	2388	100.0	38.95	23.3442	1	192	189
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	8.3682	0.03	392	2388	100.0	38.88	23.3739	1	192	188
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	8.4294	0.03	393	2388	100.0	38.90	23.4044	1	192	187
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
61244	249	251	9.9998	0.2500	100.0	489.05	605.33	1516.36	1315.28	10.52	...	8.4541	0.03	372	2319	100.0	29.11	17.5234	4	255	4
61245	249	252	0.0028	0.0015	100.0	518.67	643.42	1598.92	1426.77	14.62	...	8.2221	0.03	396	2388	100.0	39.38	23.7151	4	255	3
61246	249	253	0.0029	0.0000	100.0	518.67	643.68	1607.72	1430.56	14.62	...	8.2525	0.03	395	2388	100.0	39.78	23.8270	4	255	2
61247	249	254	35.0046	0.8400	100.0	449.44	555.77	1381.29	1148.18	5.48	...	9.0515	0.02	337	2223	100.0	15.26	9.0774	4	255	1
61248	249	255	42.0030	0.8400	100.0	445.00	549.85	1369.75	1147.45	3.91	...	9.1207	0.02	333	2212	100.0	10.66	6.4341	4	255	0

160359 rows × 29 columns

Data preprocessing

- **Step 3:** Discretize the target variable to [0,1]
  - IF target value < 25 then 1
  - Else 0

	Unit number	Time in cycles	Altitude	Mach number	TRA	T2	T4	T30	T50	P2	...	farB	htBleed	Nf_dmd	PCNfR_dmd	W31	W32	condition	Max cycles	target	label_target
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	0.03	392	2388	100.0	39.06	23.4190	1	192	191	0
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	0.03	392	2388	100.0	39.00	23.4236	1	192	190	0
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	0.03	390	2388	100.0	38.95	23.3442	1	192	189	0
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	0.03	392	2388	100.0	38.88	23.3739	1	192	188	0
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	0.03	393	2388	100.0	38.90	23.4044	1	192	187	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
61244	249	251	9.9998	0.2500	100.0	489.05	605.33	1516.36	1315.28	10.52	...	0.03	372	2319	100.0	29.11	17.5234	4	255	4	1
61245	249	252	0.0028	0.0015	100.0	518.67	643.42	1598.92	1426.77	14.62	...	0.03	396	2388	100.0	39.38	23.7151	4	255	3	1
61246	249	253	0.0029	0.0000	100.0	518.67	643.68	1607.72	1430.56	14.62	...	0.03	395	2388	100.0	39.78	23.8270	4	255	2	1
61247	249	254	35.0046	0.8400	100.0	449.44	555.77	1381.29	1148.18	5.48	...	0.02	337	2223	100.0	15.26	9.0774	4	255	1	1
61248	249	255	42.0030	0.8400	100.0	445.00	549.85	1369.75	1147.45	3.91	...	0.02	333	2212	100.0	10.66	6.4341	4	255	0	1

160359 rows × 30 columns

- **Step 4:** Drop the added features that aren't in the original output except for the label_target
  - Max cycles
  - Condition
  - Unit number
  - Target

	Time in cycles	Altitude	Mach number	TRA	T2	T4	T30	T50	P2	P15	...	NRf	NRc	BPR	farB	htBleed	Nf_dmd	PCNfR_dmd	W31	W32	label_target
0	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	...	2388.02	8138.62	8.4195	0.03	392	2388	100.0	39.06	23.4190	0
1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	...	2388.07	8131.49	8.4318	0.03	392	2388	100.0	39.00	23.4236	0
2	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	...	2388.03	8133.23	8.4178	0.03	390	2388	100.0	38.95	23.3442	0
3	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	...	2388.08	8133.83	8.3682	0.03	392	2388	100.0	38.88	23.3739	0
4	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	...	2388.04	8133.80	8.4294	0.03	393	2388	100.0	38.90	23.4044	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
61244	251	9.9998	0.2500	100.0	489.05	605.33	1516.36	1315.28	10.52	15.46	...	2388.73	8185.69	8.4541	0.03	372	2319	100.0	29.11	17.5234	1
61245	252	0.0028	0.0015	100.0	518.67	643.42	1598.92	1426.77	14.62	21.57	...	2388.46	8185.47	8.2221	0.03	396	2388	100.0	39.38	23.7151	1
61246	253	0.0029	0.0000	100.0	518.67	643.68	1607.72	1430.56	14.62	21.57	...	2388.48	8193.94	8.2525	0.03	395	2388	100.0	39.78	23.8270	1
61247	254	35.0046	0.8400	100.0	449.44	555.77	1381.29	1148.18	5.48	7.96	...	2388.83	8125.64	9.0515	0.02	337	2223	100.0	15.26	9.0774	1
61248	255	42.0030	0.8400	100.0	445.00	549.85	1369.75	1147.45	3.91	5.69	...	2388.66	8144.33	9.1207	0.02	333	2212	100.0	10.66	6.4341	1

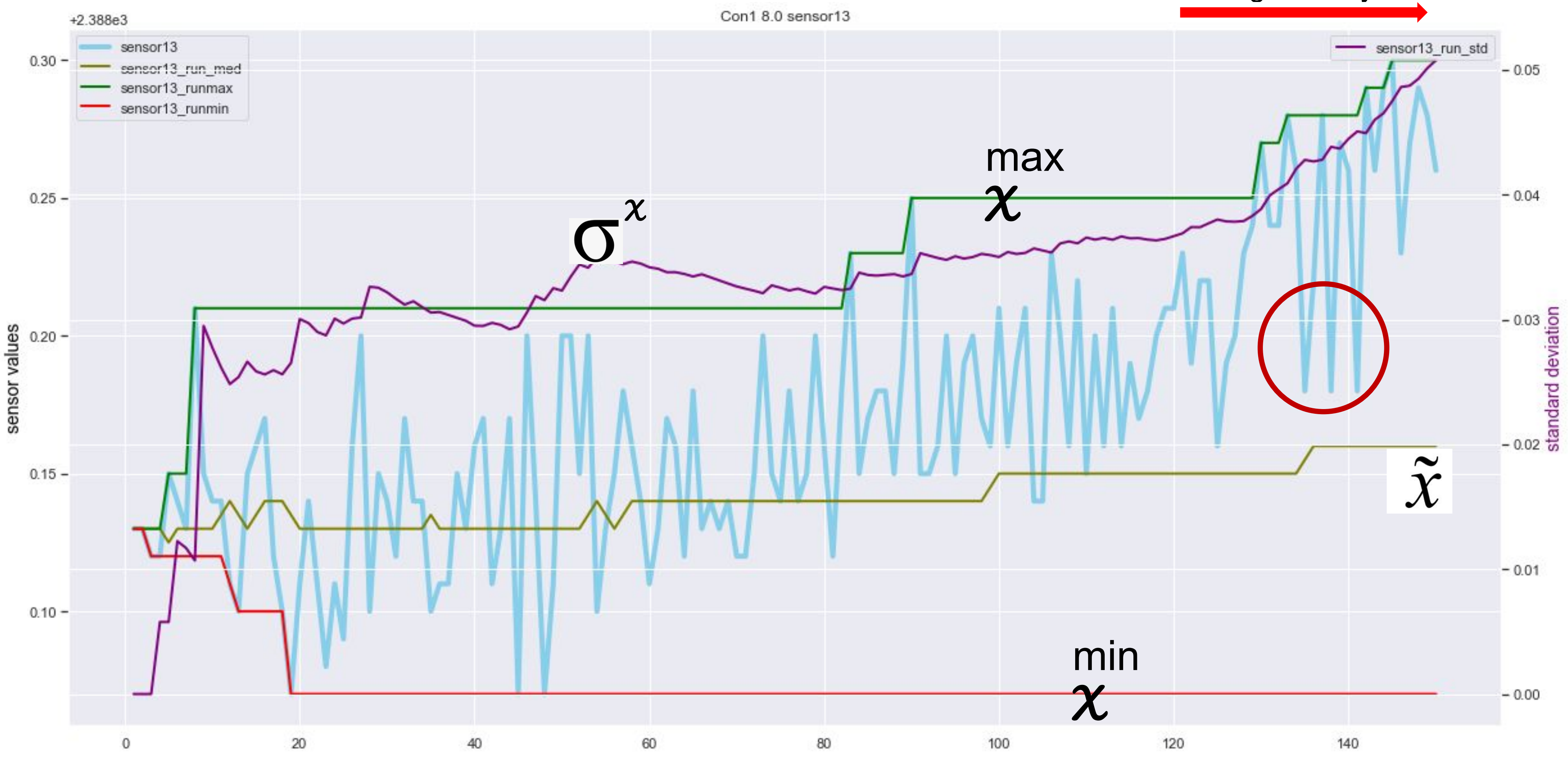
160359 rows × 26 columns

Data preprocessing

- **Step 5:** Partition the dataset into train_test_split. Parameters for split are:
  - X is feature set except for the target
  - y is the label_target column
  - Test size = 0.33
  - Stratify = y
    - splits the dataset in a way that the proportion of values in the sample produced will be the same as the proportion of values provided to parameter stratify.
    - For our dataset, there are 11% of ones and 89% of zeroes
  - Partitioned dataset has the following shapes:
    - X train shape(107440, 25)
    - X test shape(52919, 25)
    - y train shape(107440,)
    - y test shape(52919,)

Mitigating sensor noise

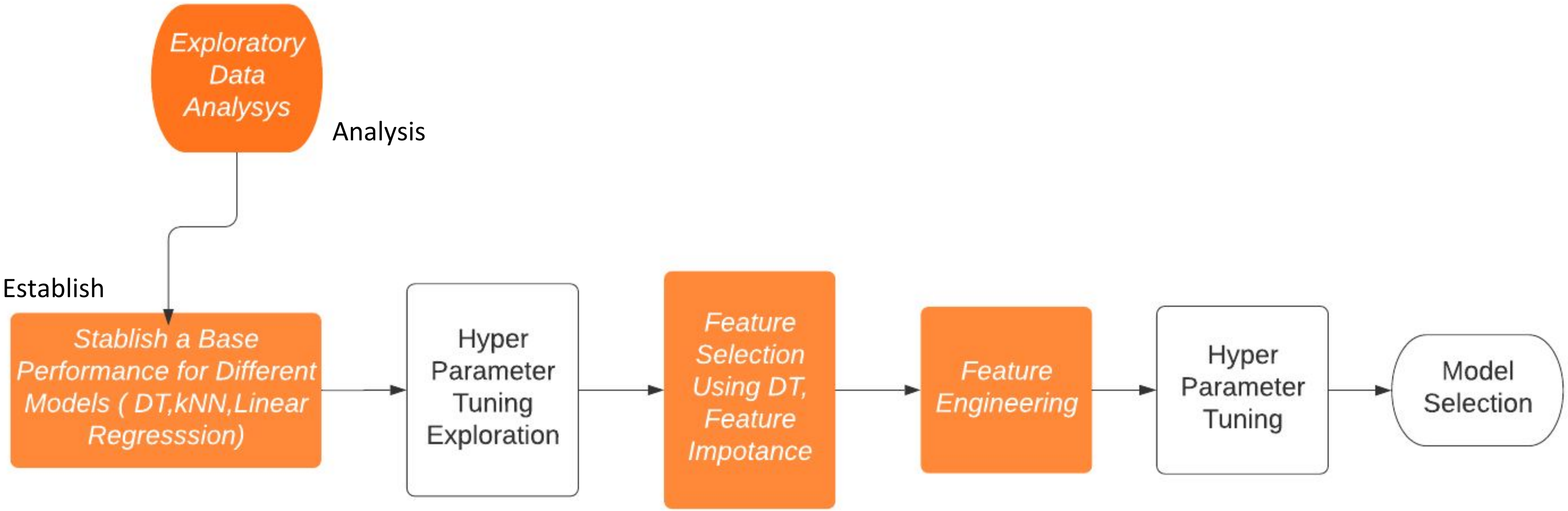
Failing in 25 cycles



Problem | **Data** | Modeling | Evaluation | Deployment & value

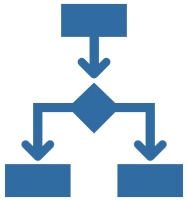
QuAM development

QuAM development



Design and development

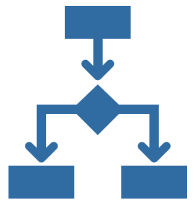
Decision Tree



- **Pros:**
  - Categorical Target Variable
  - Interest in Significance of Features
  - Quick Benchmark
- **Cons:**
  - Not suitable for Complex , Novel Problems
- **Hyperparameters of interest**
  - N_estimators (Width of the Model)
  - Max_depth (Depth of the Model)
  - min_samples_leaf criterion(most significant feature)
  - splitter(best, random)
  - min_sample_split

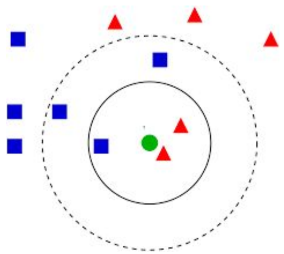
Design and development

Decision Tree



- **Pros:**
  - Categorical Target Variable
  - Interest in Significance of Features
  - Quick Benchmark
- **Cons:**
  - Not suitable for Complex , Novel Problems
- **Hyperparameters of interest**
  - N_estimators (Width of the Model)
  - Max_depth (Depth of the Model)
  - min_samples_leaf criterion(most significant feature)
  - splitter(best, random)
  - min_sample_split

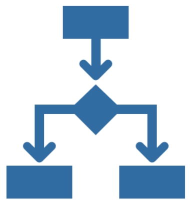
kNN



- **Pros:**
  - Versatile: Categorical /Continuous Target Variable
  - Intuitive, Easy to implement
  - Relatively High Accuracy
- **Cons:**
  - Computationally expensive — because the algorithm stores all of the training data
  - High memory requirement
  - Prediction stage might be slow (with big N)
  - Sensitive to irrelevant features and the scale of the data
- **Hyperparameters of interest**
  - N_neighbours
  - Weights(‘uniform’,’distance’)

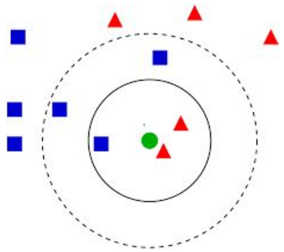
Design and development

Decision Tree



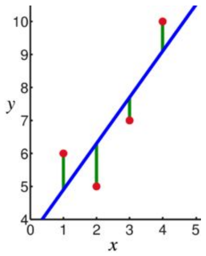
- **Pros:**
  - Categorical Target Variable
  - Interest in Significance of Features
  - Quick Benchmark
- **Cons:**
  - Not suitable for Complex , Novel Problems
- **Hyperparameters of interest**
  - N_estimators (Width of the Model)
  - Max_depth (Depth of the Model)
  - min_samples_leaf criterion(most significant feature)
  - splitter(best, random)
  - min_sample_split

kNN



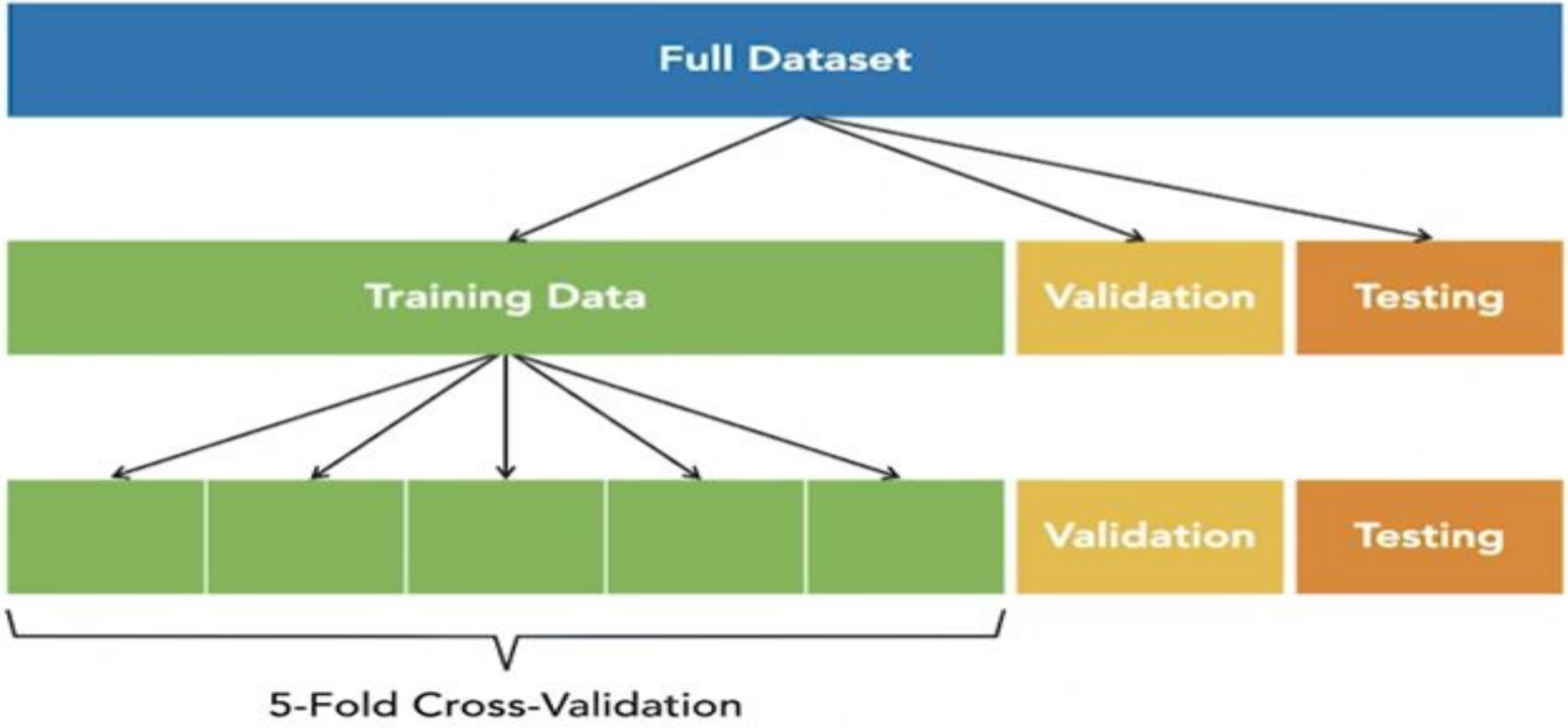
- **Pros:**
  - Versatile: Categorical /Continuous Target Variable
  - Intuitive, Easy to implement
  - Relatively High Accuracy
- **Cons:**
  - Computationally expensive — because the algorithm stores all of the training data
  - High memory requirement
  - Prediction stage might be slow (with big N)
  - Sensitive to irrelevant features and the scale of the data
- **Hyperparameters of interest**
  - N_neighbours
  - Weights(‘uniform’,’distance’)

Linear Regression

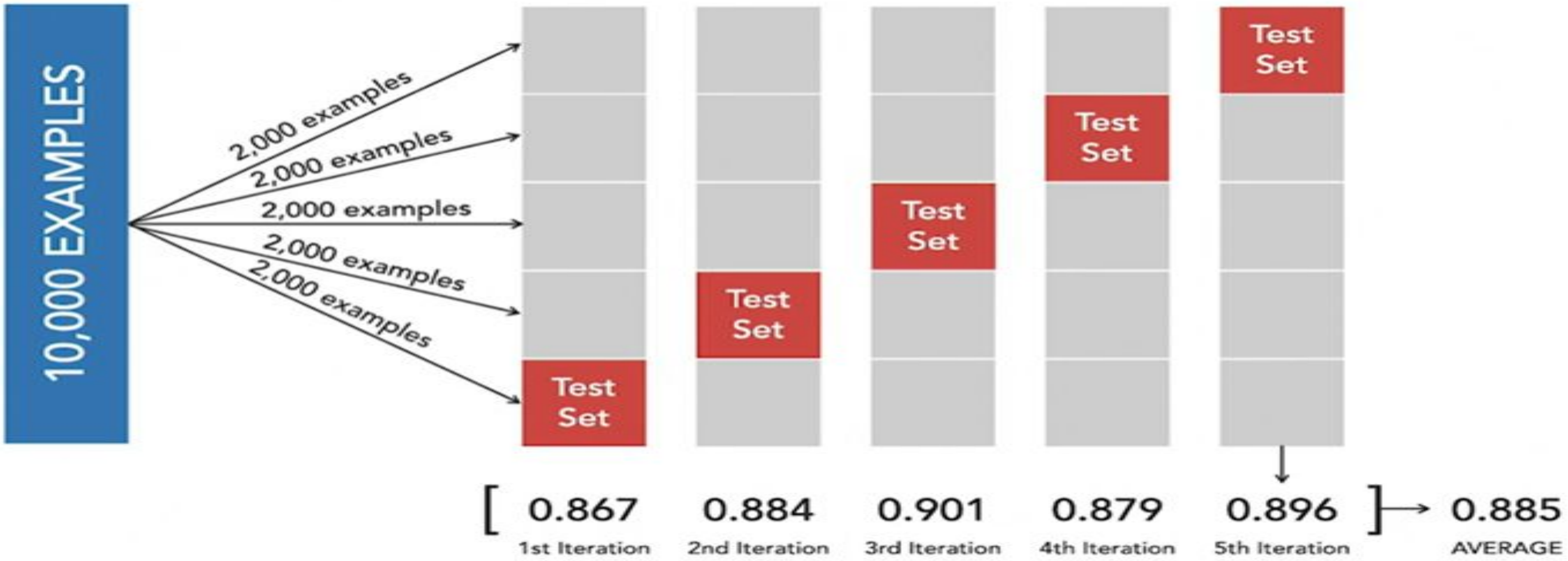


- **Pros:**
  - Versatile: Categorical /Continuous Target Variable
  - Easy to implement & interpret
- **Cons:**
  - assumes a linear relationship between dependent and independent variables.
  - outliers can have huge effects on the regression and boundaries are linear in this technique.
- **Hyperparameters of interest**
  - (Fit_intercept, normalize)

Hyperparameter tuning



Hyperparameters tuning



DT hyperparameters tuning

```
#Fitting the Model and Evaluating
dtree = tree.DecisionTreeClassifier()
parameters = {

    'min_samples_leaf': [1,5,10,15,20],
    # to test 1,5,10,15,20 leaves
    'max_depth': [2,4,8,16,32,None],
    'min_samples_split': [5,10,15,20],
    'criterion': ['gini','entropy'],
    'splitter': ['best','random']
    # to test various depths including No limitiation on the depth i.e. None

}
#using GridSearchCV to loop through predefined hyperparameters and fit your estimator (model) on your training set
cv = GridSearchCV(dtree,parameters, cv = 5)
#cv = 5 meaning it will run 5-fold validation for each hyperparameter combination
cv.fit(tr_features,tr_labels.values.ravel())
# we use ravel for the labels to convert it to an array, since the label is usually just one column and the algorithm expects an array
print_results(cv)
```

```
0.968 (+/-0.003) for {'criterion': 'entropy', 'max_depth': 32, 'min_samples_leaf': 10, 'min_samples_split': 15, 'splitter': 'random'}
0.975 (+/-0.003) for {'criterion': 'entropy', 'max_depth': 32, 'min_samples_leaf': 10, 'min_samples_split': 20, 'splitter': 'best'}
0.968 (+/-0.003) for {'criterion': 'entropy', 'max_depth': 32, 'min_samples_leaf': 10, 'min_samples_split': 20, 'splitter': 'random'}
0.973 (+/-0.003) for {'criterion': 'entropy', 'max_depth': 32, 'min_samples_leaf': 15, 'min_samples_split': 5, 'splitter': 'best'}
0.964 (+/-0.003) for {'criterion': 'entropy', 'max_depth': 32, 'min_samples_leaf': 15, 'min_samples_split': 5, 'splitter': 'random'}
```

Metrics Comparison - Decision Trees

	Decision Tree base model	
	Training Set	Validation Set
Accuracy	0.972617275	0.958880553
Precision	0.941620422	0.840508977
Recall	0.967268418	0.79270097
F1 Score	0.954272116	0.815905245

	Decision Tree with Tuning	
	Training Set	Validation Set
Accuracy	0.999311175	0.985235558
Precision	0.999589634	0.98443895
Recall	0.999032496	0.986057436
F1 Score	0.999310987	0.985247528

Evaluation and selection

- Which model to deploy? By Occam's razor principle, the simpler model, Decision Tree, because it generalizes well enough for all conditions
- We've been able to predict engine failure within 25 cycles within the NASA's expectations

  Recall >98% and F1_Score 99%

- Current model should allow NASA to move maintenance interval from 5 to 10 cycles
- Model would be evaluated and re-trained as necessary
- So how can NASA deploy this?

Q & A

Detecting Overfitting (High Variance)	Detecting Underfitting (High Bias)	Strategies used to combat Overfitting & Underfitting
If "Accuracy" (measured against the training set) is very good and "Validation Accuracy" (measured against a validation set) is not as good, then the model is overfitting.	If Accuracy and Validation Accuracy are similar but are both poor, then the model may be underfitting.	Overfitting - *reduced the complexity* of the model by reducing the number of trainable parameters – We used a correlation Matrix to identify features with high level of correlation (ex. P2 and T2 sensors had 100% correlation)
Apply Occam's Razor test - If two models have comparable performance, pick the simpler one.		5-fold Cross Validation- Cross-validation allows you to tune hyperparameters with only your original training set. This allows you to keep your test set as a truly unseen dataset for selecting your final model.
		Regularization (Ideal hyperparameters for each model – ex. For DT model – Prune the trees using max depth
		SMOTE Oversampling Method was applied. Others - Ensemble model- ex Random Forest as an ensemble for DT model Ensembles are ML methods for combining predictions from multiple separate models.