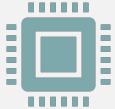
Fault and Malfunction Detection ML Modelling for Onshore Wind Turbine

By

- **Merveille Sonkin**
- **Moe Moe Aye**
- **Rana Adel**
- **Serap Demirhan**

Contents

Introduction



Our focus is on the wind turbine at Kelmarsh wind farm in the UK. Through the utilization of ***data science techniques*** and ***SCADA data***, we aim to improve ***fault*** and ***malfunction*** detection, ensuring enhanced operational ***reliability***.

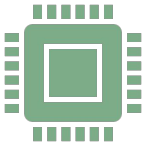
We invite you to join us on this journey towards a ***more sustainable energy future***.

Contents

Objectives

By delving into the dataset, our aim is not only to facilitate ***informed decision-making*** for turbine maintenance teams but also to deepen our understanding of the ***diverse factors*** influencing turbine ***performance and reliability***.

Identify and classify various fault and malfunction patterns



Develop a robust machine learning model for early detection and diagnosis of turbine issues

Optimize turbine performance to ensure maximum energy output

Enhance operational reliability and minimize downtime

Contents

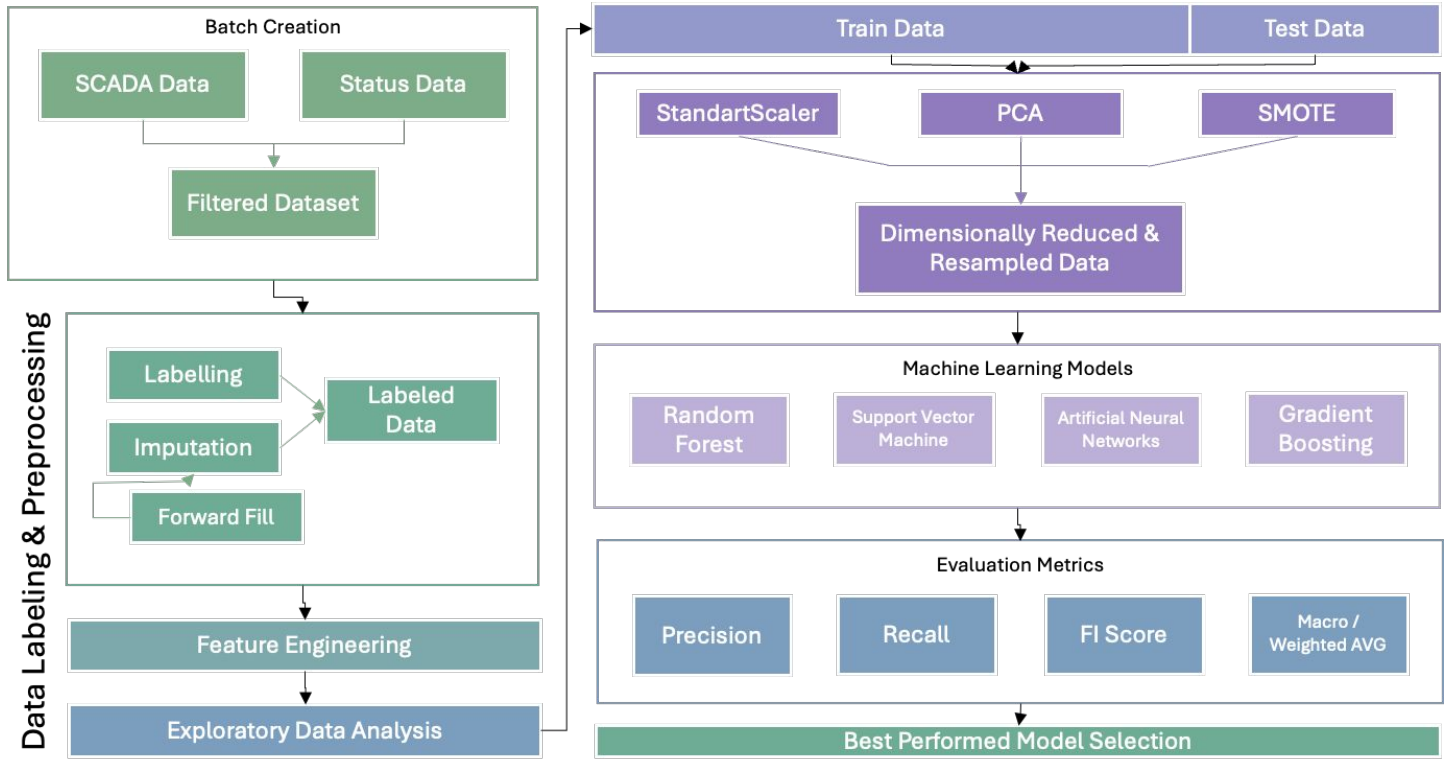
Dataset Overview

Wind Turbine Type	Senvion MM92
Rated Power	2 MW
Rotor diameter	92 m
Hub height	72 m
Cut-in wind speed	3 m/s
Rated wind speed	12.5 m/s
Cut-out wind speed	24.9 m/s

- For this project, only one turbine was selected, adhering to the standard practice ***of testing models*** on a single turbine before ***scaling up***.
- To ensure ***balanced data***, the selection process prioritized identifying the turbine with the highest frequency of ***faults*** and ***malfunctions***.
- Subsequently, the data was grouped by year from ***2021 to the end of 2022***.
- It contains 2 data:
  - 10-minute ***SCADA*** data providing **technical measurements**
  - Minutely ***status/events*** data
  - It has in total ***105120*** observations and ***313*** features.

Contents

Methodology



Contents

Question for Deeper Insights

01

What are the ***main fault*** and ***malfunction patterns*** observed in the turbine's operational data?

02

How do these patterns relate to changes in ***wind speed, rotor speed***, and other ***operational parameters***?

03

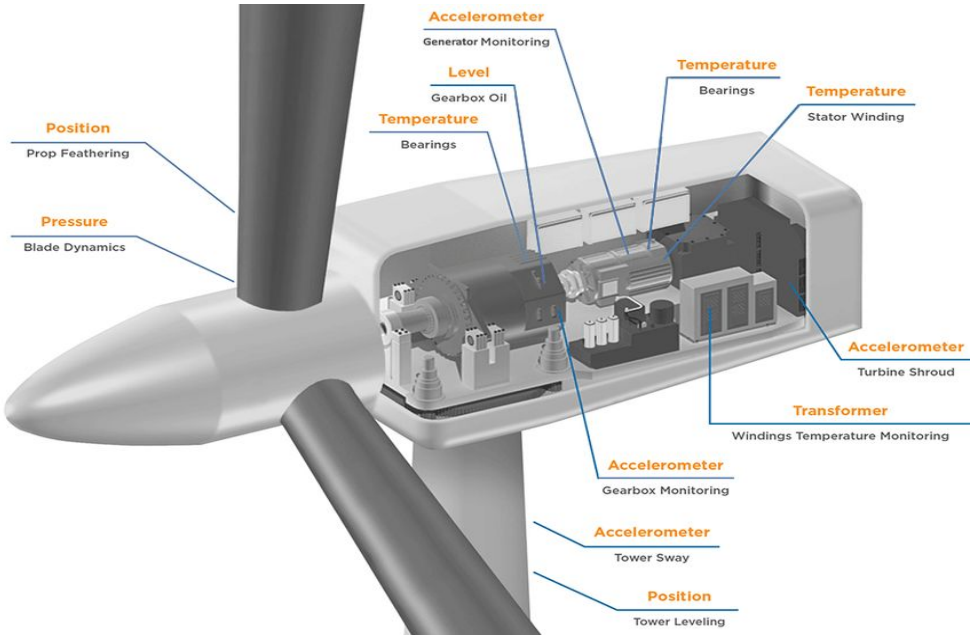
Can we detect **recurring patterns** or **anomalies** in the turbine's performance data that suggest underlying issues?

04

What is the best and most efficient way to **predict and anticipate** these anomalies moving forward?

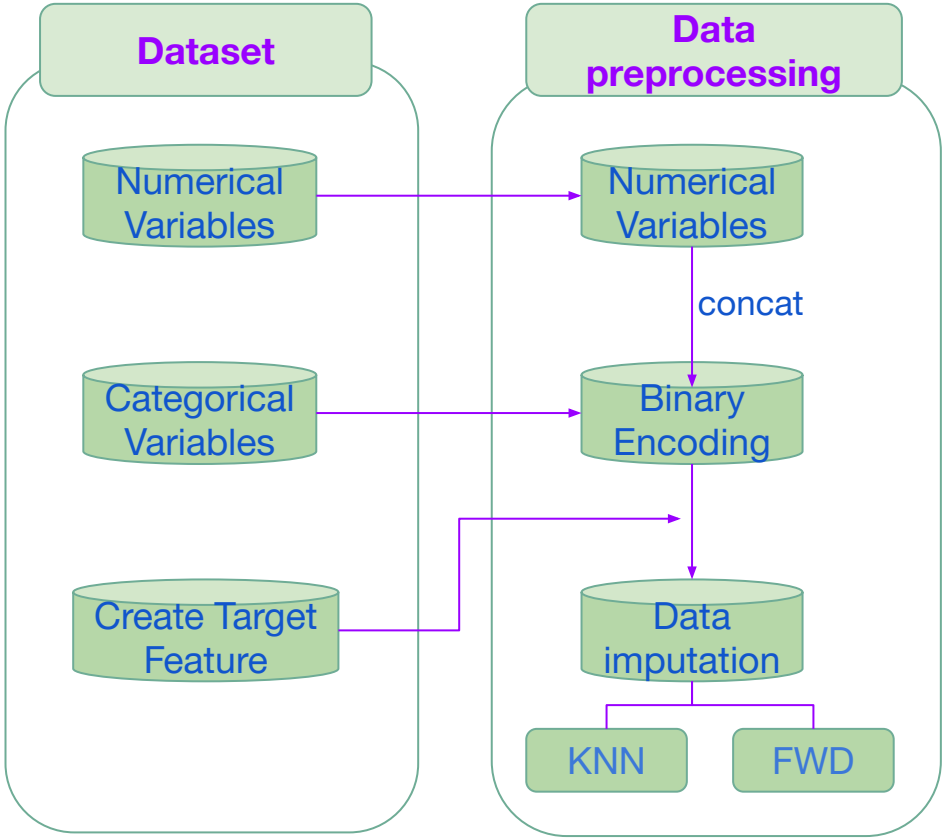
Contents

An impression of sensors locations on a typical onshore wind turbine

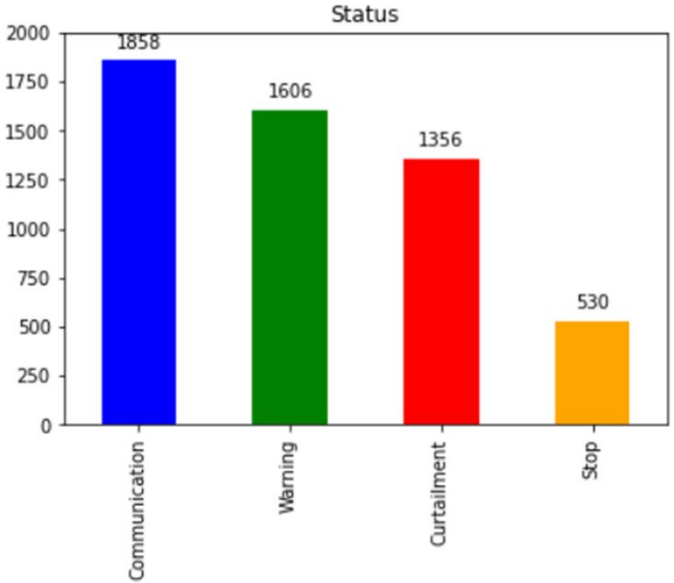


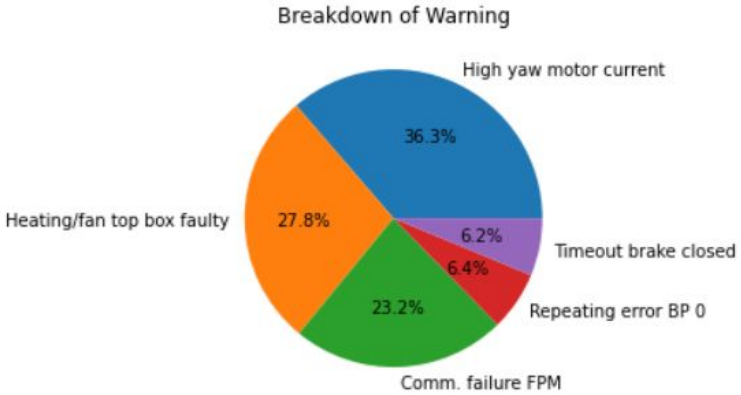
Source: TE Connectivity brochure

Data preprocessing

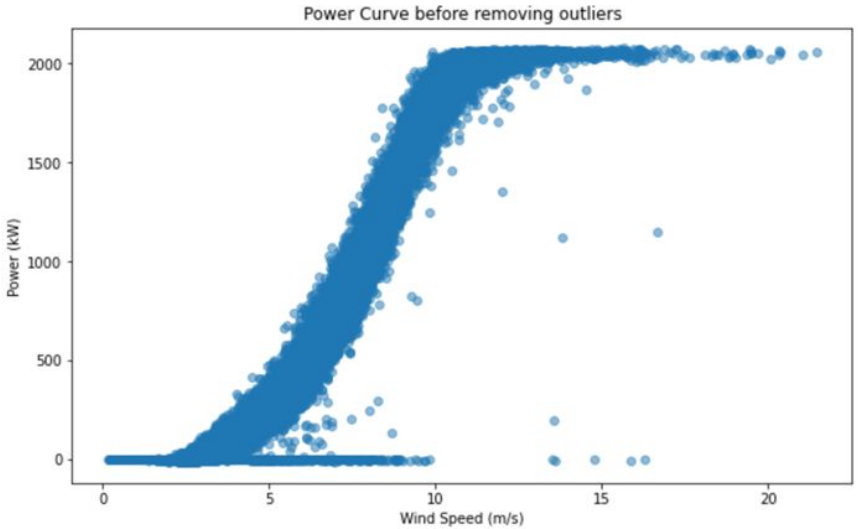


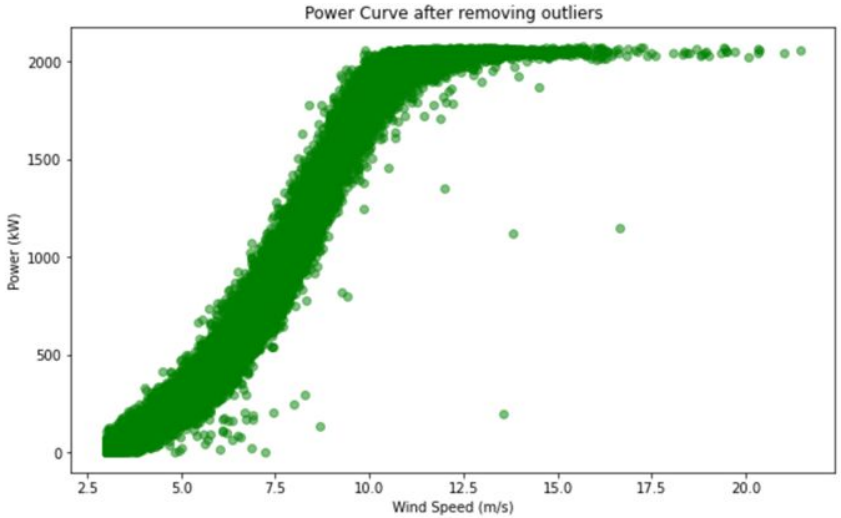
What are anomalies?



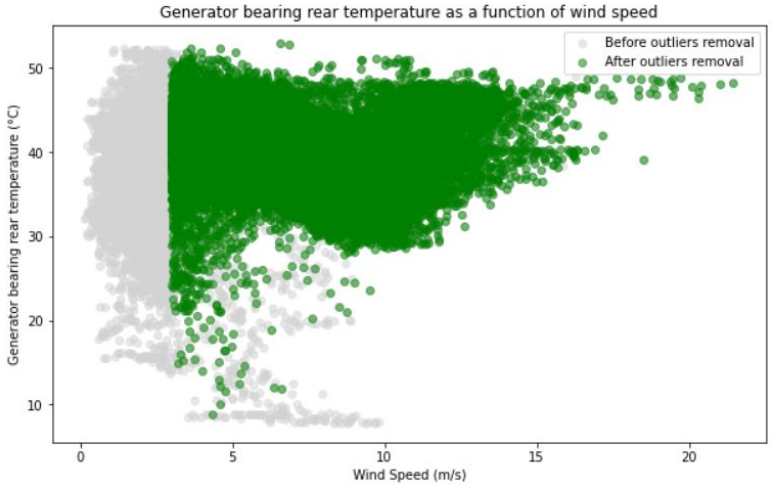


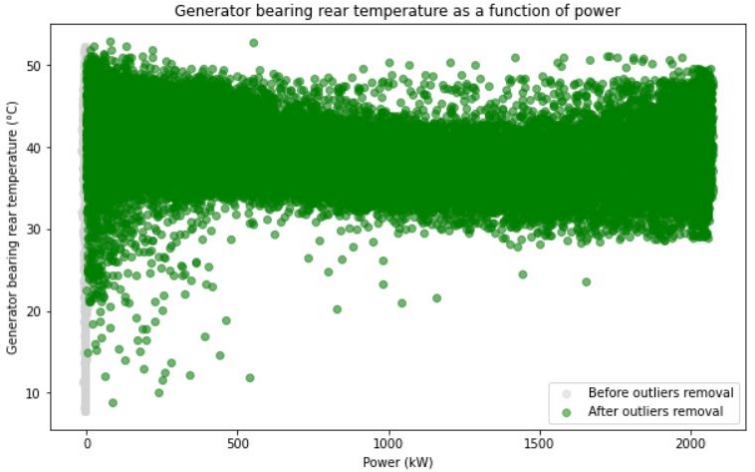
Power curve under healthy and faulty conditions



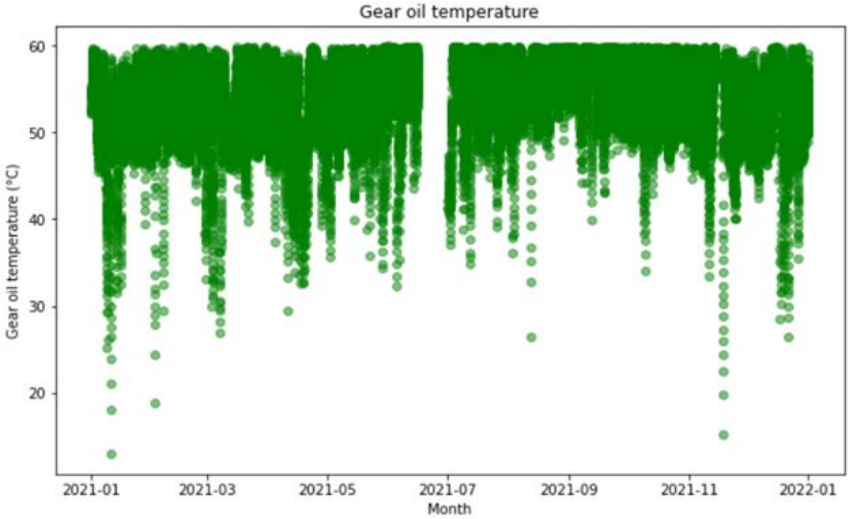


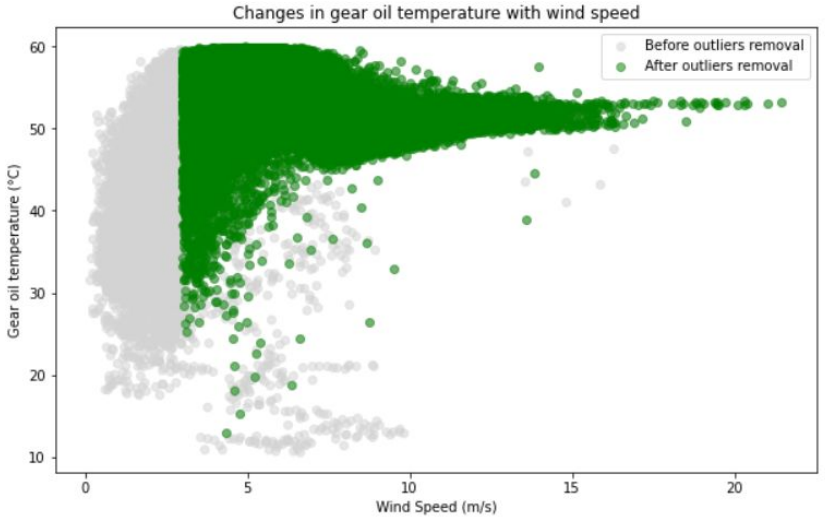
Relationship between components temperature and operating conditions



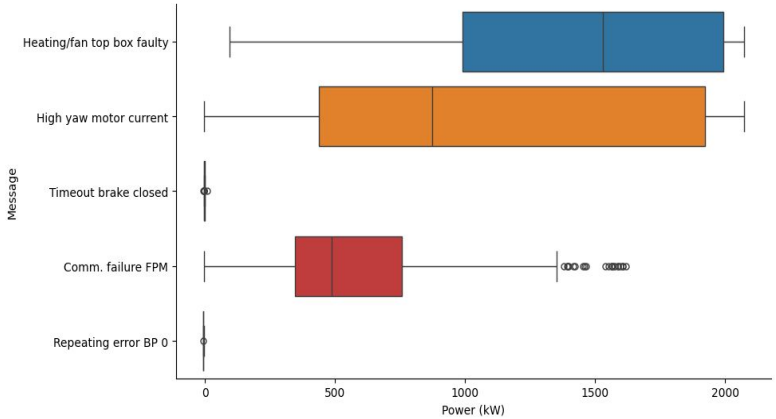


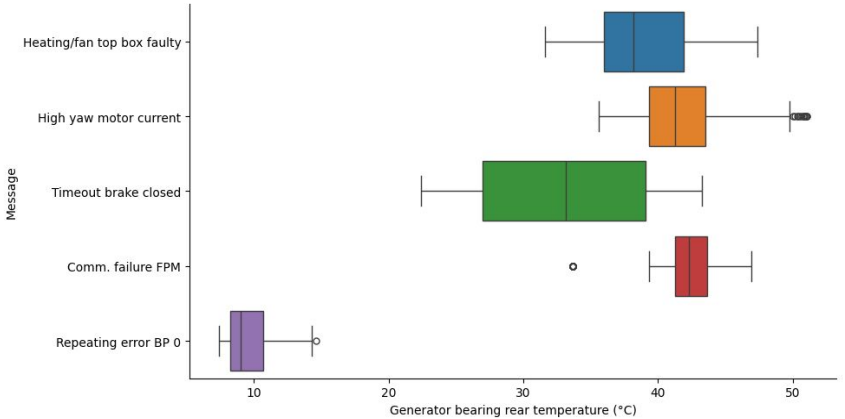
Relationship between components temperature and operating condition





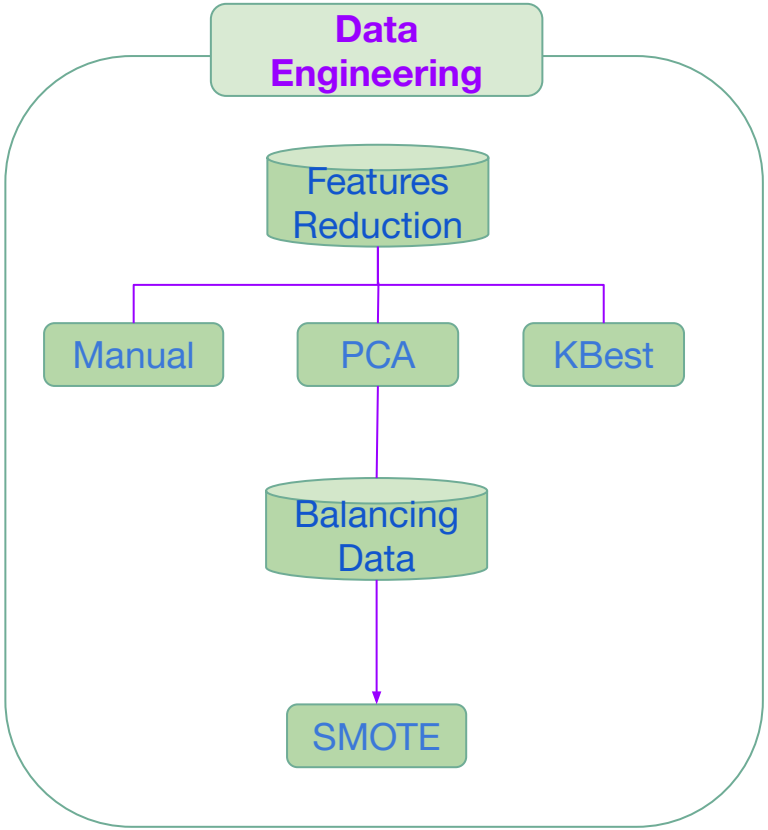
Variation in power and generator bearing rear temperature grouped by anomaly types





Contents

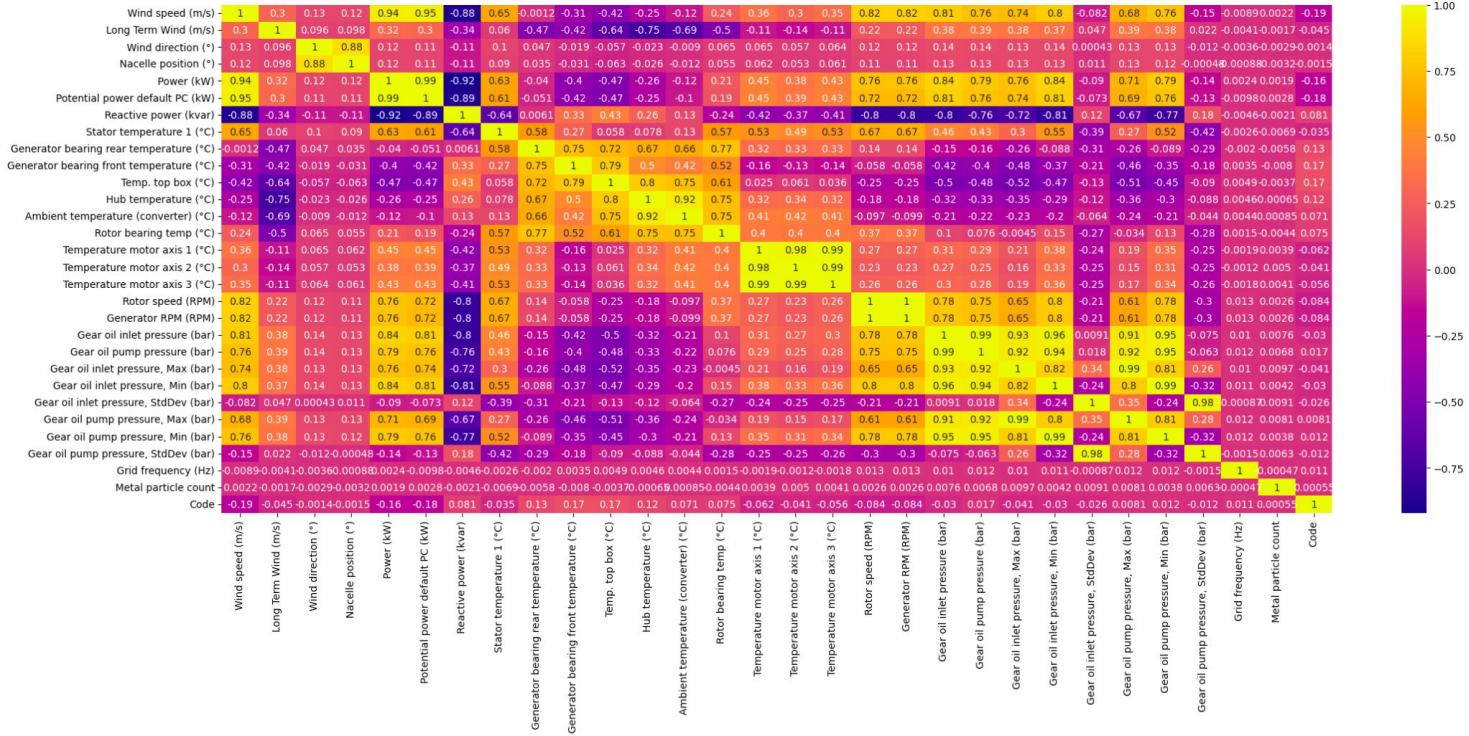
Data engineering



Contents

Feature Selection: Without PCA

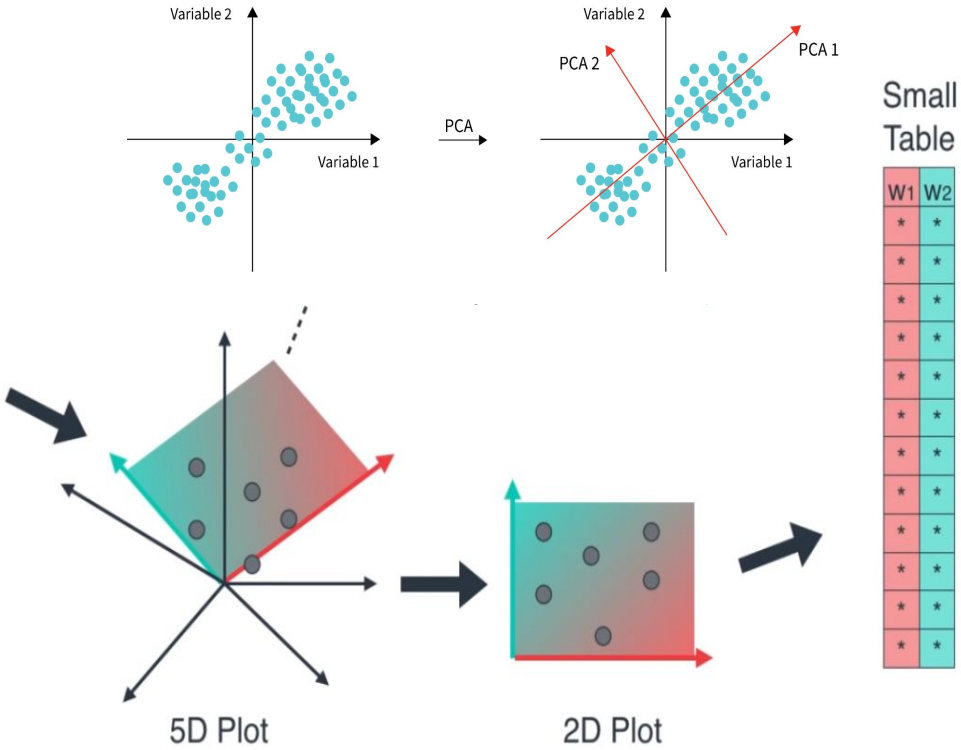
Manually select the features based on expertise and make sure to use the correlation matrix



Feature Selection: With Principal Component Analysis (PCA)

Large Table

X1	X2	X3	X4	X5
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*

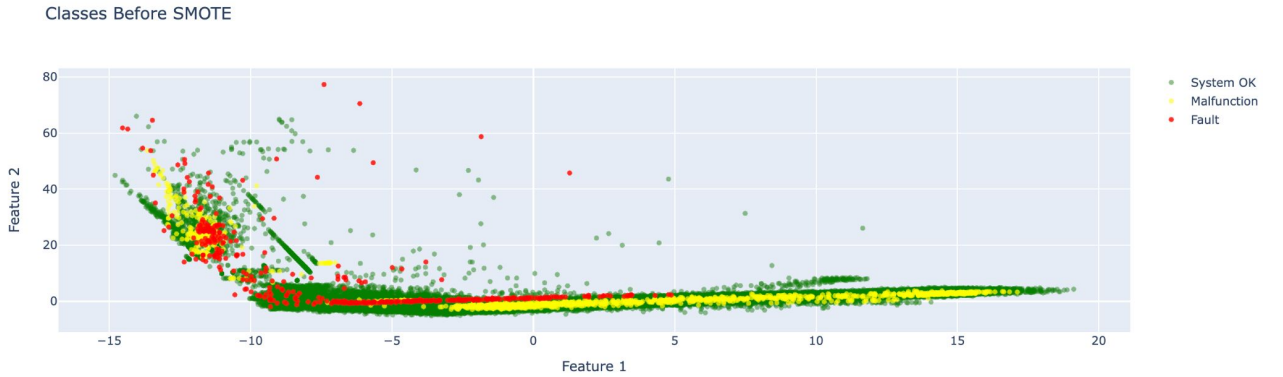


Small Table

W1	W2
*	*
*	*
*	*
*	*
*	*
*	*
*	*
*	*
*	*
*	*
*	*

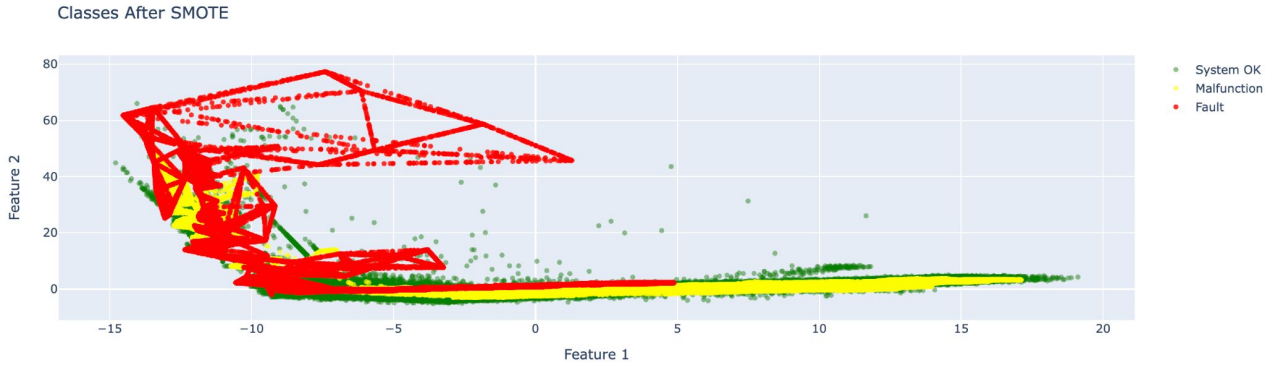
```python
# Choose the number of components you want to keep
pca = PCA(n_components=34)

# Fit PCA on the training set
pca.fit(X_train)

# Transform both the training and testing sets
# using the fitted PCA models
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

Resampling: Synthetic Minority Oversampling Technique




```
y_train.value_counts()
[37] ✓ 0.0s
... System OK      82403
    Malfunction     1275
    Failure          418
    Name: Fault, dtype: int64
```

SMOTE



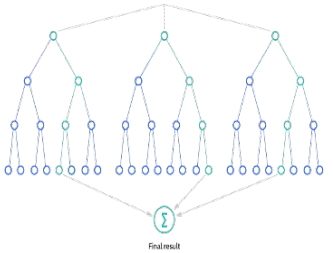

```
y_train_resampled.value_counts()
[36] ✓ 0.0s
... System OK      82403
    Malfunction    82403
    Failure        82403
    Name: Fault, dtype: int64
```

Contents

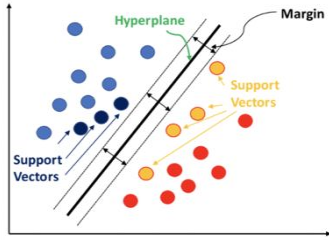
Selected Supervised ML Models

We have ***an imbalanced multiclass (Fault, Malfunction, System OK)*** problem in our dataset.

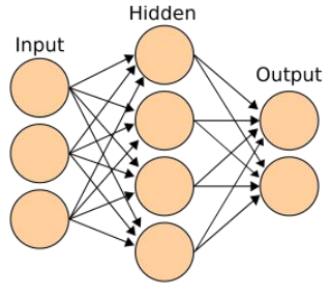
Therefore, we chose algorithms specifically designed to tackle this type of issue.



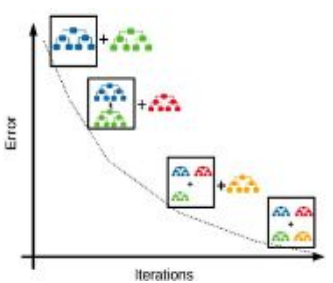
Random Forest



Support Vector Machine



Artificial Neural Networks

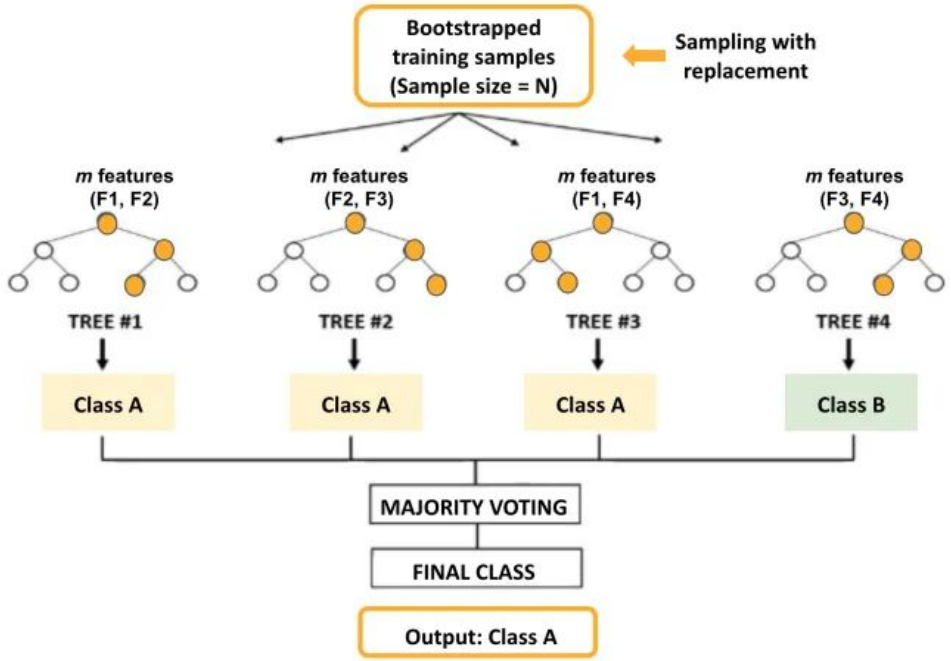


Gradient Boosting

Random Forest Classifier

Training Data
(Sample size, N= 6, No. of features, F = 4)

F1	F2	F3	F4	Y
2.1	0	400	-9	A
3.0	1	890	-42	B
2.2	1	929	0	B
4.0	0	324	-23	A
3.5	1	333	-15	A
6.0	0	215	-9	A

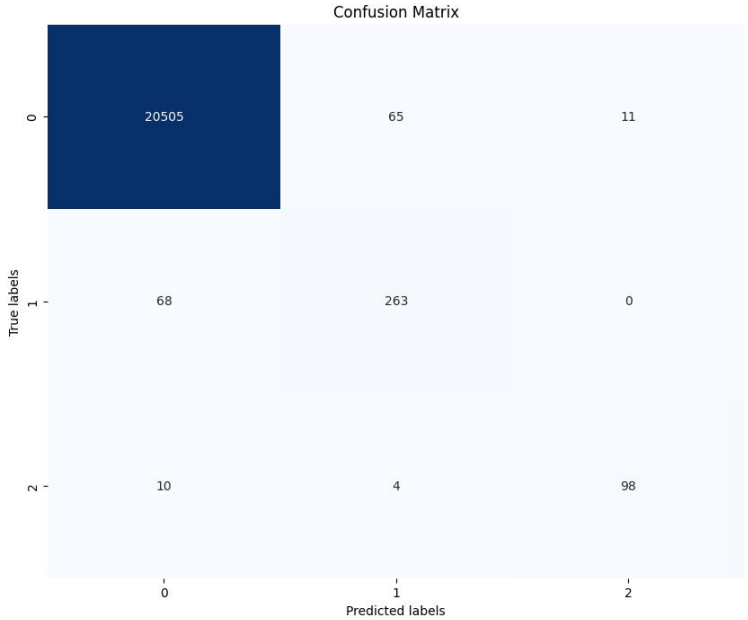


Key parameters of Random Forest Model are: (a) Number of trees , (b) Maximum depth of the trees (c) Size of the random subset of features
In this example, No. of trees = 4, Depth = 2, and Feature subset size, m = 2 (no. of features/2)

Resource: https://aiml.com/what-is-random-forest-2/

RF Model: Fault and Malfunction Prediction

Actual failures (true positives) are crucial to prevent downtime and costly repairs. Therefore, recall takes precedence over precision

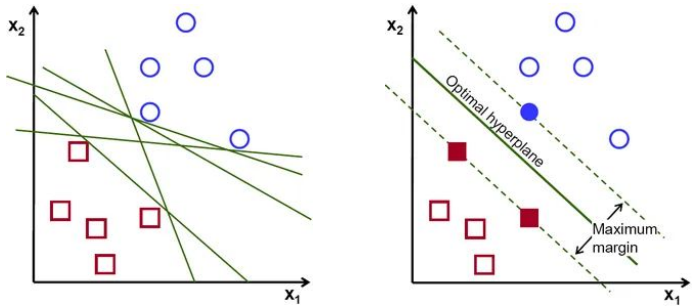


With FFill, PCA, SMOTE

	precision	recall	f1-score	support
Fault	0.90	**0.88**	0.89	112
Malfunction	0.79	**0.79**	0.79	331
System OK	1.00	**1.00**	1.00	20581
accuracy			0.99	21024
macro avg	0.90	0.89	0.89	21024
weighted avg	0.99	0.99	0.99	21024

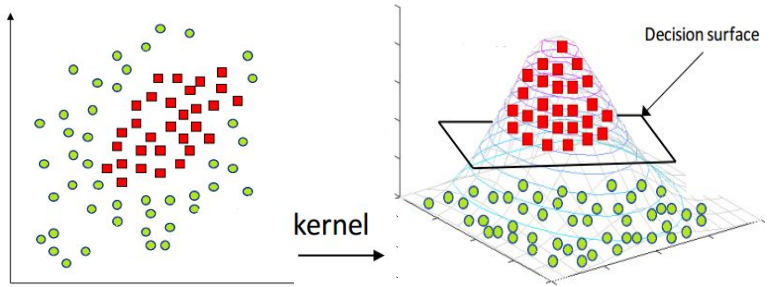
Support Vector Machine: Kernel Trick

Linear Approach




```python
# Parameters for SVM with Linear Kernel
svm_linear = SVC(
    kernel='linear',
    C=1.0,
    decision_function_shape='ovr',
    random_state=42
)
```

Non-linear Approach



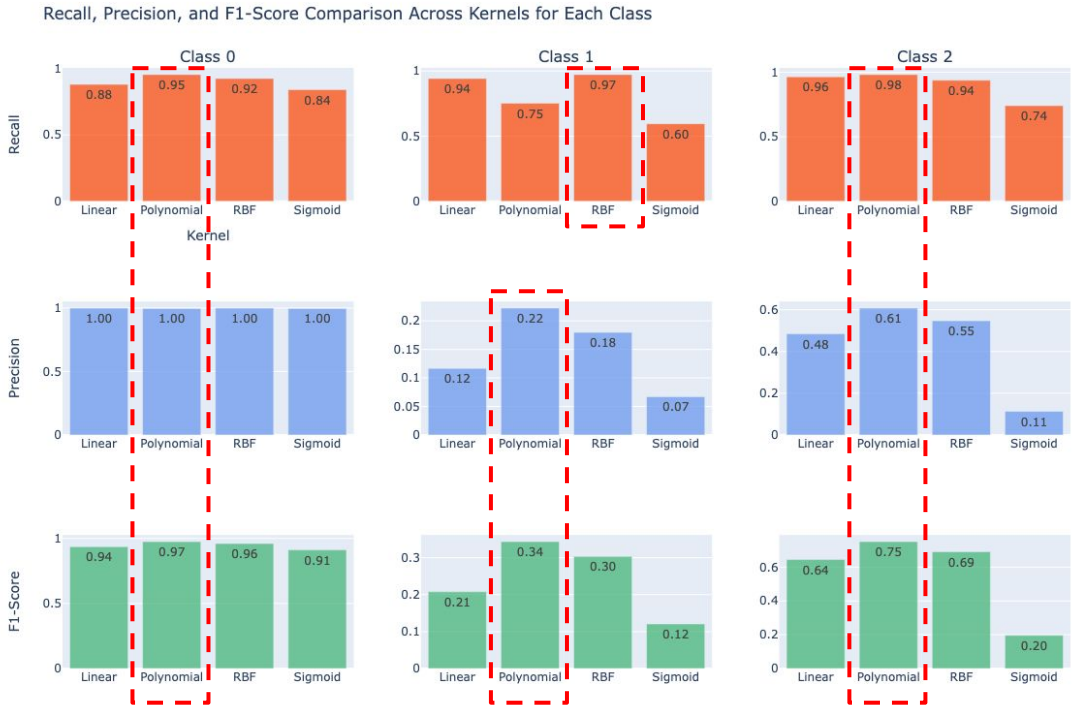

```python
# Parameters for SVM with Polynomial Kernel
svm_poly = SVC(
    kernel='poly',
    C=1.0,
    gamma='scale',
    degree=3,
    coef0=0.0,
    decision_function_shape='ovr',
    random_state=42
)
```

```python
# Parameters for SVM with RBF Kernel
svm_rbf = SVC(
    kernel='rbf',
    C=1.0,
    gamma='scale',
    decision_function_shape='ovr',
    random_state=42
)
```

```python
# Parameters for SVM with Sigmoid Kernel
svm_sigmoid = SVC(
    kernel='sigmoid',
    C=1.0,
    gamma='scale',
    coef0=0.0,
    decision_function_shape='ovr',
    random_state=42
)
```

Resource link for images: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

Support Vector Machine: Kernel's comparison and Insights



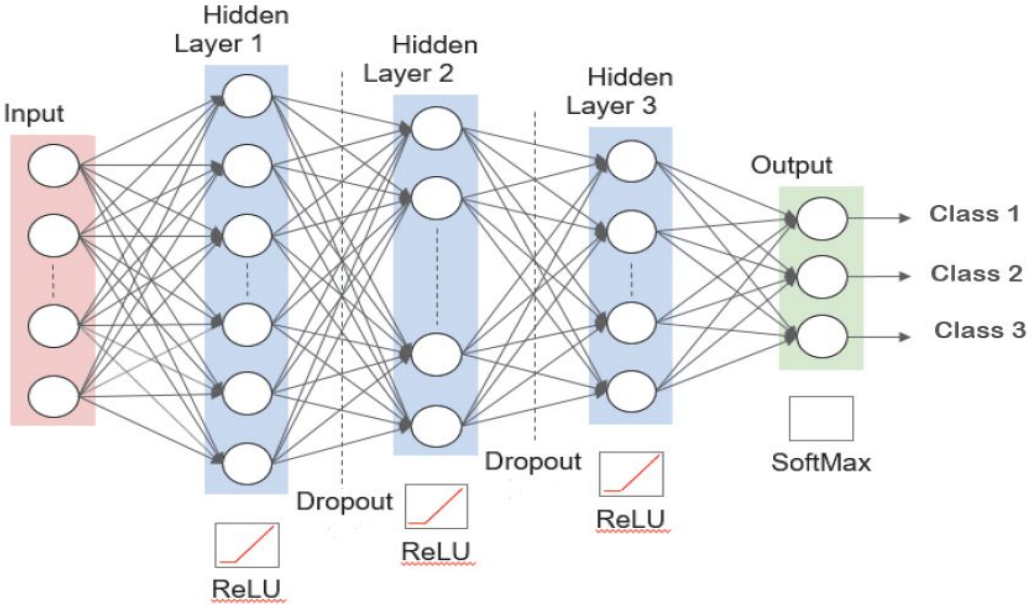
Poly Kernel Shines ✨

- Poly bridges the gap between capturing complex data relationships and ***prioritizing the less frequent classes***.
- This translates to a balanced F1-score, avoiding an ***overload of false alarms*** while effectively identifying ***anomalies***.

Please remember that:

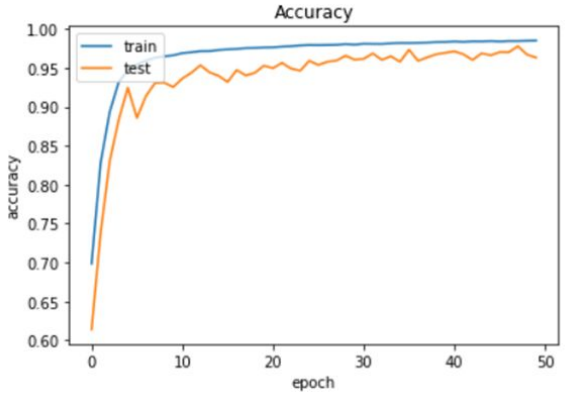
- Further ***hyperparameter tuning*** for each kernel might improve performance.

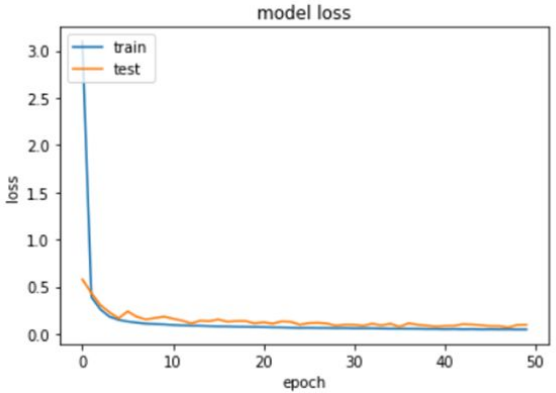
Artificial Neural Networks (ANN)



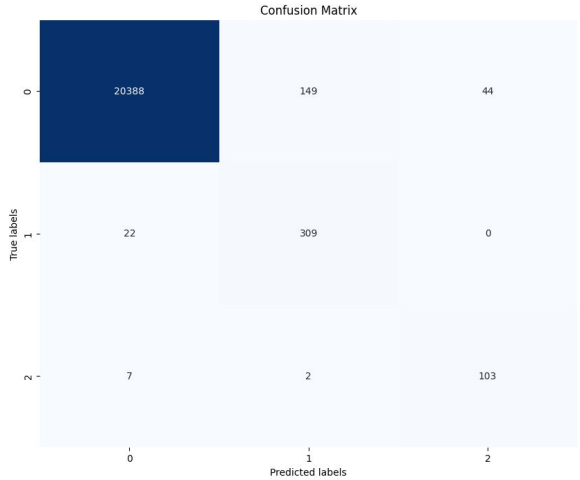

```python
# Optimize , Compile And Train The Model
opt =keras.optimizers.Adam(learning_rate=0.0001)

# Compile the model
ann_model.compile(
    optimizer=opt,
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# Train the model
history = ann_model.fit(
    X_train_resampled,
    y_train_resampled,
    epochs=50,
    batch_size=64,
    validation_data=(X_test, y_test),
    verbose=1
)
```

Artificial Neural Networks (ANN)



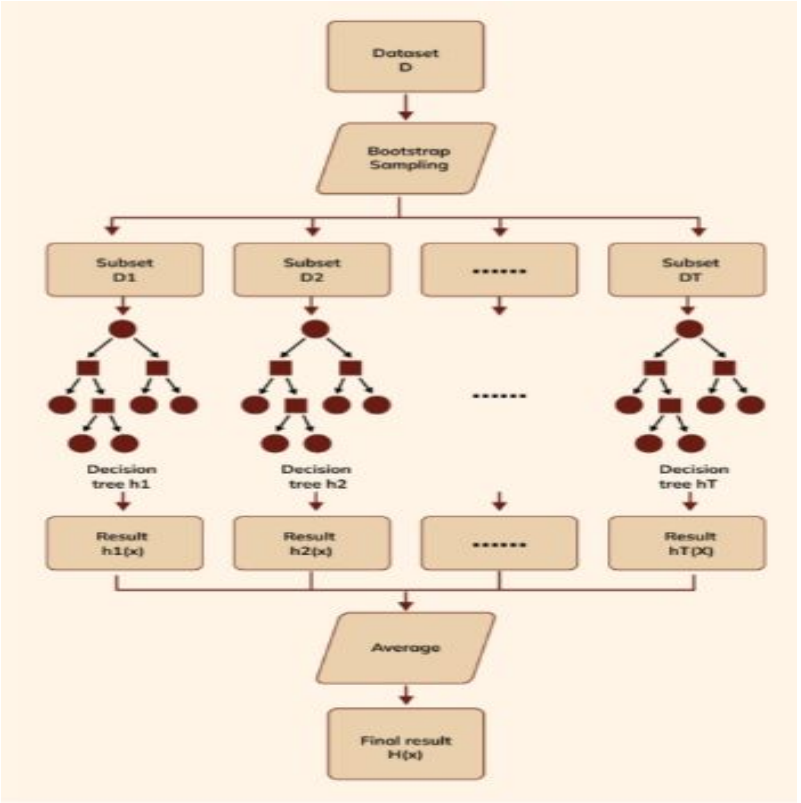


	precision	recall	f1-score	support
Fault	0.70	**0.92**	0.80	112
Malfunction	0.67	**0.93**	0.78	331
System OK	1.00	**0.99**	0.99	20581
accuracy			0.99	21024
macro avg	0.79	0.95	0.86	21024
weighted avg	0.99	0.99	0.99	21024



Gradient Boosting Classifier

- **n_estimators:** The number of boosting rounds (iterations).
- **max_depth**: The maximum depth of each tree.
- **learning_rate**: The learning rate, controlling step size during optimization.
- **Max_features**:
- **Subsample ratio:**



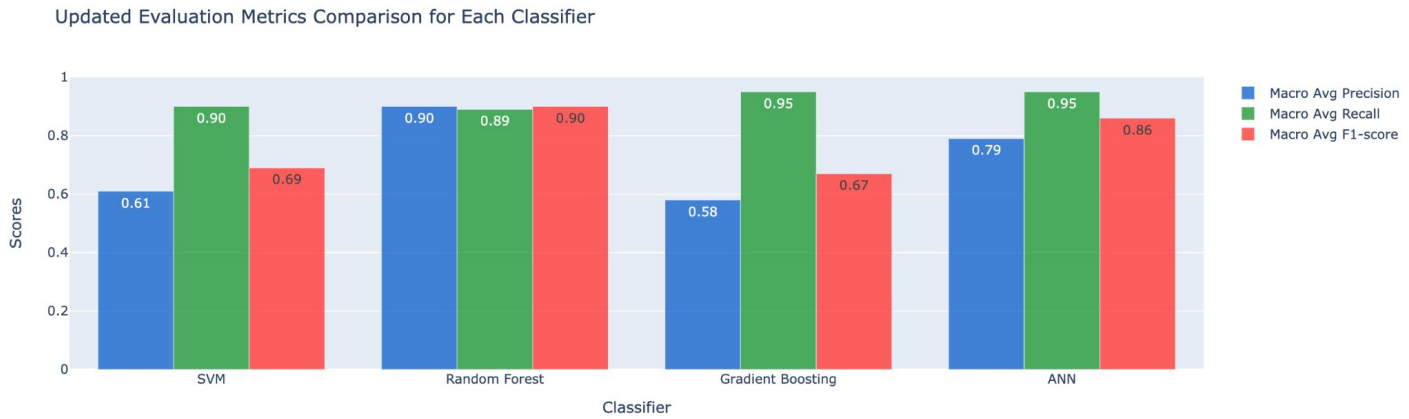
Gradient Boosting Classifier

```
# Define the GradientBoostingClassifier
gbc = GradientBoostingClassifier(n_estimators=300,
                                 learning_rate=0.05,
                                 random_state=100,
                                 max_features=5)
```

	precision	recall	f1-score	support
Fault	0.53	**0.97**	0.69	112
Malfunction	0.21	**0.94**	0.35	331
System OK	1.00	**0.94**	0.97	20581
accuracy			0.94	21024
macro avg	0.58	0.95	0.67	21024
weighted avg	0.98	0.94	0.96	21024

Contents

Conclusion



- Thorough EDA is important in understanding the data and identifying relevant features contributing to our problem.
- In wind turbine fault detection (rare events), ***Random Forest excels***. It prioritizes malfunctions/faults with high recall ***(catches most)*** and good precision ***(avoids false alarms)***, making it the ***best choice*** for this imbalanced multi classification task.
- However, it is important to note none of the models achieve optimal results, some improvements need to be made to ***fine-tune the model parameters*** and ***feature engineering*** to better handle ***imbalanced data*** and ***prioritize*** malfunction/fault classification.


Thanks!! :)

You can find us on:

Merveille Sonkin:
Contact email: mervysonkin@gmail.com
LinkedIn Profile: linkedin.com/in/merveille-sonkin-1568ba121

Moe Moe Aye:
Contact email: moe.moe.aye010@gmail.com
Linkedin Profile: https://www.linkedin.com/in/moe-moe-aye-909381/

Rana Adel:
Contact email: rrradel87@gmail.com
Linkedin Profile: https://www.linkedin.com/in/rana-adel-794337a7/

Serap Demirhan:
Contact email: demirhannserap@gmail.com
Linkedin Profile: https://www.linkedin.com/in/serapdemirhan/