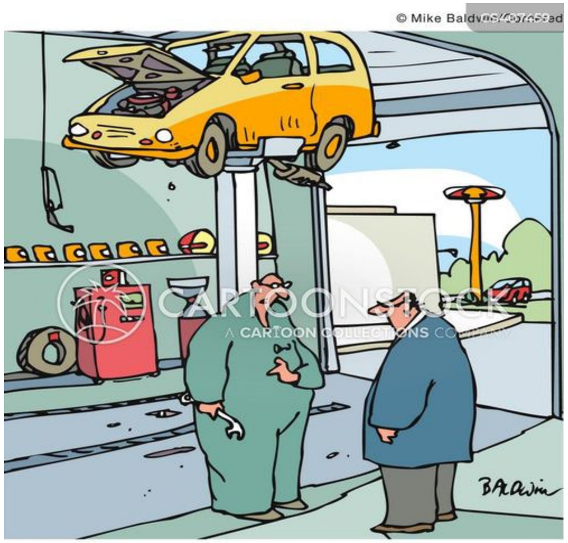
Predictive Maintenance Solves:

Damaged Equipment

Costly Repairs



"It's broke. I could fix it, but then you'd be broke."

.. Tells it like it is

- **Fewer breakdowns**
- **Fewer costly repairs**
- **Peace of mind**

BUSINESS CASE

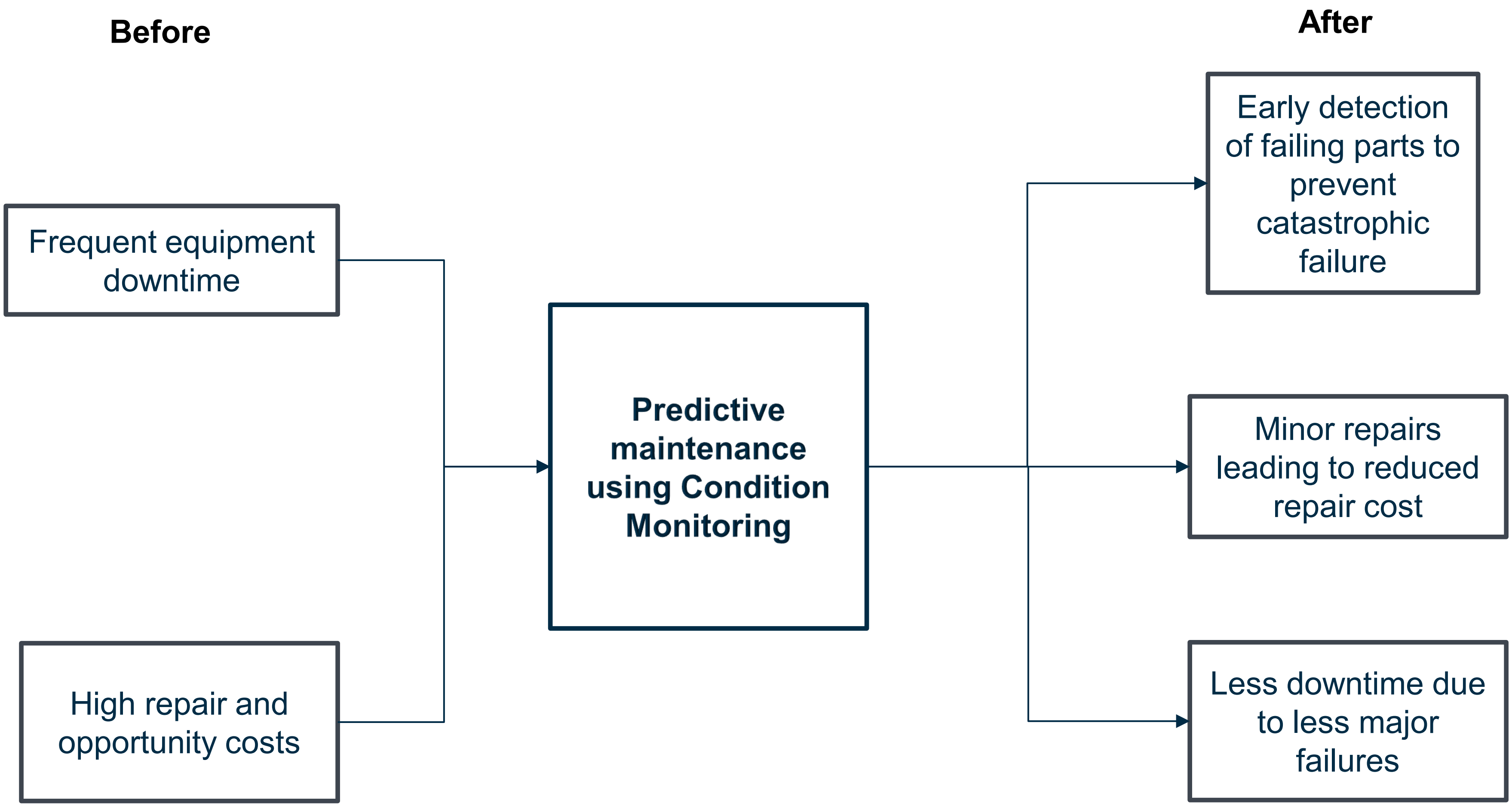
Significance of Predictive Maintenance: Predictive maintenance enhances machinery reliability, maximizing productivity while minimizing operational disruptions significantly.

Deloitte Findings Impact: According to Deloitte, implementing this maintenance strategy reduces breakdowns by 70%, enhancing efficiency immensely.

Cost Efficiency Gains: Predictive maintenance lowers costs by 25% and downtime by 35%, resulting in substantial long-term savings for industries.

Solution?

- Predictive Maintenance using ML based Condition Monitoring



GOAL OF THIS STUDY

- Determine if vibration data can be used to predict worsening or damaged motor arrangement using machine learning
- Success Criteria: more than 70% value of a custom model metric

What we did

- Explored the trends in the data to understand it better
- Prepared and modeled the data for predicting unbalance levels
- Compared the performance of different kinds of models (RF, KNN, XGB and LR)
- Determined the best model/model strategy

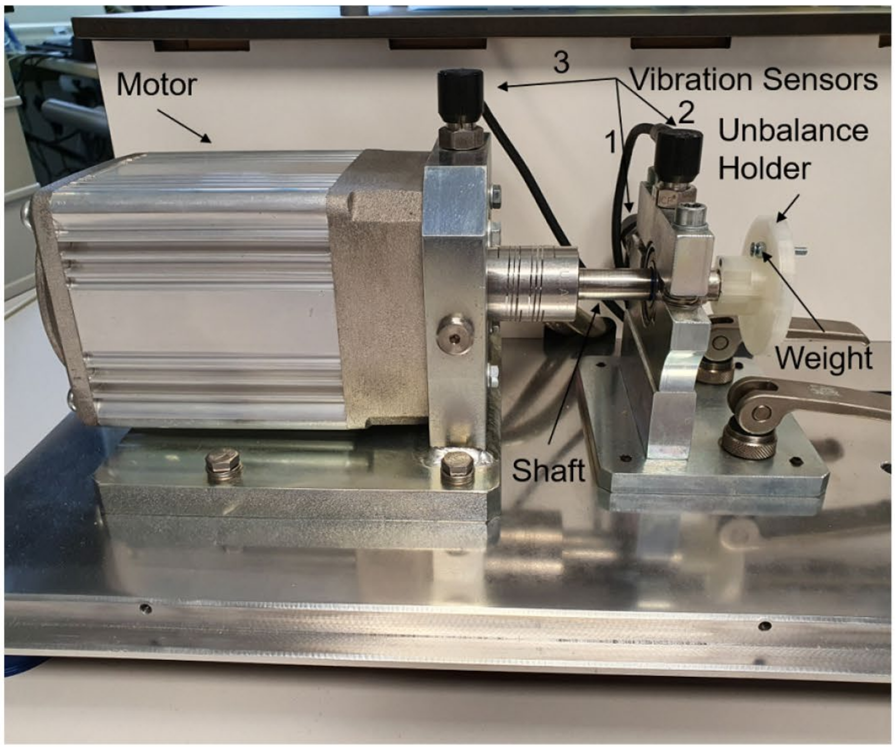
What we found

- XGB, LR and KNN gave significantly better score than RF but XGB was better at predicting middle levels of unbalance
- An XGB model supplemented by KNN for lower levels is the most appropriate modeling strategy

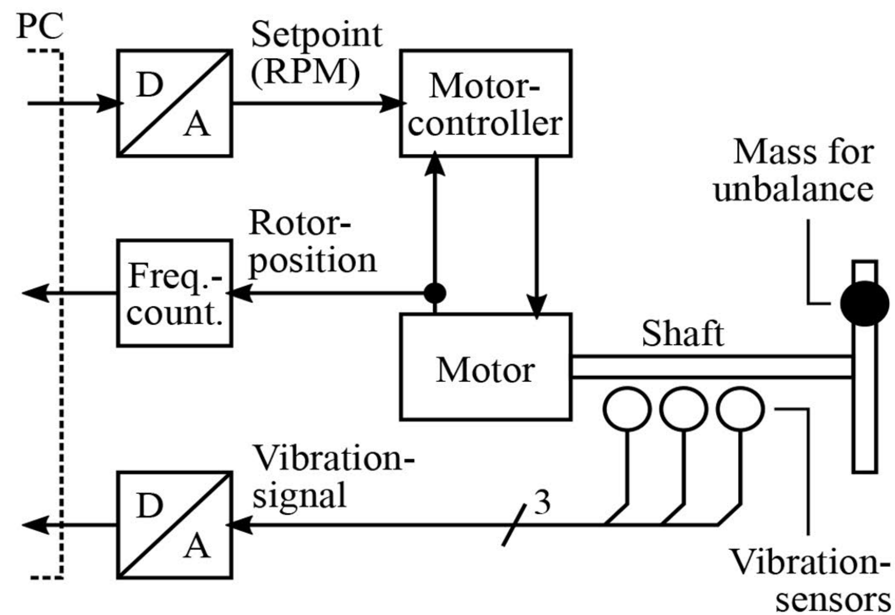
What was explored?

Vibration data for a shaft rotated by a motor with different levels of unbalance was analyzed for modeling.

Ref: https://www.kaggle.com/datasets/jishnukoliyadan/vibration-analysis-on-rotating-shaft



Measurement setup



Block diagram of the measurement setup

Experiment Setup

MORE ABOUT THE DATA..

- Datasets for 4 different unbalance strengths were recorded as well as one dataset with the unbalance holder without additional weight (i.e. without unbalance).
- The rotation speed was varied between approx. 630 and 2330 RPM in the development datasets and between approx. 1060 and 1900 RPM in the evaluation datasets.

Parameters:

1. V_in : The input voltage to the motor controller V_in (in V)
2. Measured_RPM : The rotation speed of the motor (in RPM; computed from speed measurements using the DT9837)
3. Vibration_1 : The signal from the first vibration sensor
4. Vibration_2 : The signal from the second vibration sensor
5. Vibration_3 : The signal from the third vibration sensor

MORE ABOUT THE DATA..

- To enable a comparable division into a **development dataset** and an **evaluation dataset**, separate measurements were taken for each unbalance strength, respectively.
- This separation can be recognized in the names of the csv-files, which are of the form "**1D.csv**":
  The **digit** describes the unbalance strength ("**0**" = **no unbalance**, "**4**" = **strong unbalance**), and
  the **letter** describes the **intended use** of the dataset ("**D**" = **development or training**, "**E**" = **evaluation**)

Unbalance levels

ID	Radius [mm]	Mass [g]
0D/ 0E	-	-
1D/ 1E	14 ± 0.1	3.281 ± 0.003
2D/ 2E	18.5 ± 0.1	3.281 ± 0.003
3D/ 3E	23 ± 0.1	3.281 ± 0.003
4D/ 4E	23 ± 0.1	6.614 ± 0.007

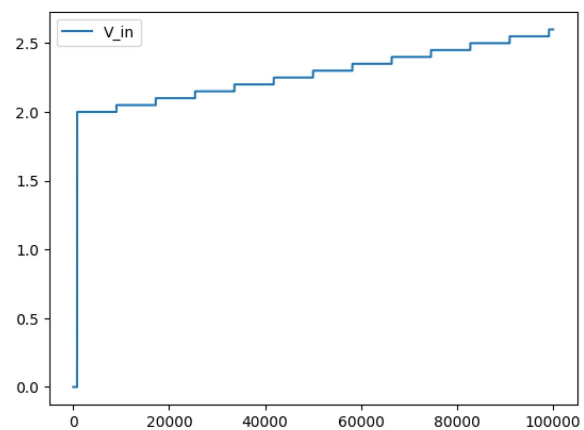
EDA

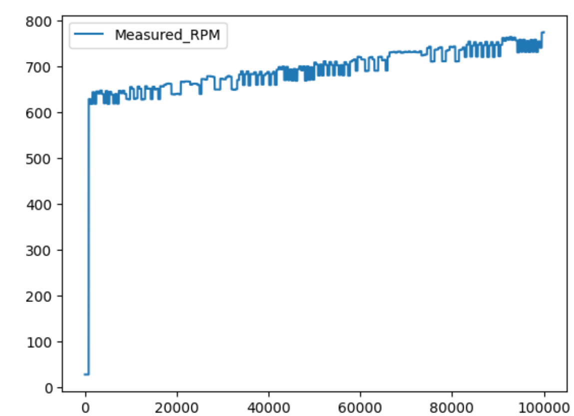
- PySpark was used to read and summarize the development ('D') data
- Initial data had 13 million rows

DESCRIPTION:

summary	V_in	Measured_RPM	Vibration_1	Vibration_2	Vibration_3	load
count	13201553	13201553	13201553	13201553	13201553	13201553
mean	5.995143904660508	-35698.335014651355	0.001651138868151...	0.002609488510923...	0.004045518033946954	1.9993595450474653
stddev	2.327930769161795	2986857.101551288	0.05552156238148596	0.0845743674615158	0.057237432687889736	1.4142643274997209
min	0.0	-2.4E8	-0.12001276	-0.21576047	-0.040333271	0
max	10.0	4091.723	7.8491378	8.7981558	7.8299785	4

- Analysis of line plots for the parameters showed the experiment was run by controlling motor voltage to vary RPM between specific ranges for different levels of unbalance





EDA

- The experiment seems to be designed around setting the motor voltage to specific levels and measure corresponding vibrations for different RPM and unbalance loadings. Therefore, we can possibly group by voltage (mean). However, doing so would also calculate the mean of load values across the voltages, which is not intended. Hence, we can include load in the grouping along with voltage.

- Even for RPM>0, the expected operating range of the equipment seems to be > 600 rpm. Hence, we can filter for data with RPM > 600.

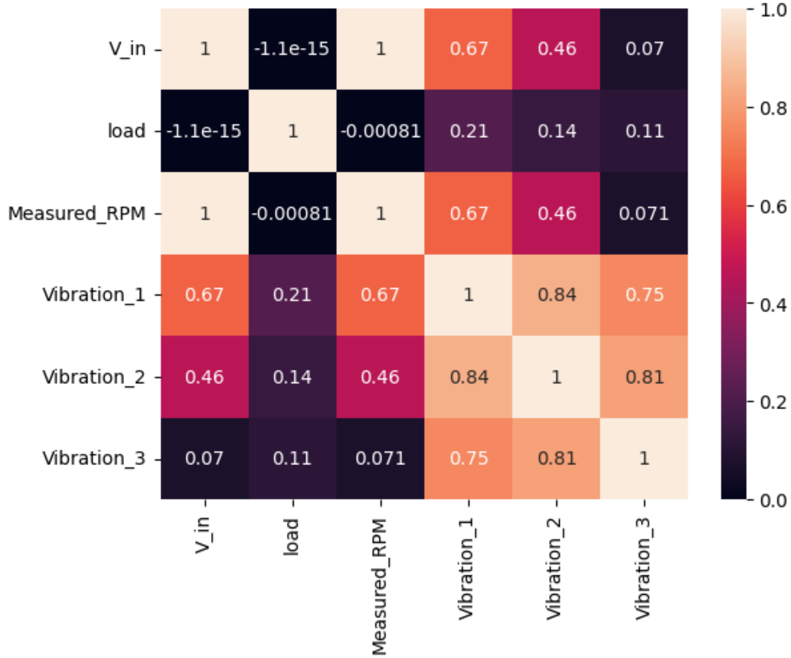
Summary of filtered and grouped data:

summary	V_in	load	avg(V_in)	avg(Measured_RPM)	avg(Vibration_1)	avg(Vibration_2)	avg(Vibration_3)	avg(load)
count	805	805	805	805	805	805	805	805
mean	5.999999999999997	2.0	5.999999999999992	1480.2197952421488	0.006343393632915632	0.008040079157754607	0.004384854192235918	2.0
stddev	2.325234701682919	1.4150927751172553	2.3252347016829047	492.42333510183096	0.005665749415115298	0.00944732536112008	0.004680429372654114	1.4150927751172553
min	2.0	0	2.0	636.9171586791892	5.101755232146109E-4	5.7834928147528E-4	0.002457040609266361	0.0
max	10.0	4	10.0	2332.1845971923567	0.06426996650102809	0.11080615566330182	0.07979620657750469	4.0

- Repeated columns that should be removed
- Much smaller dataset that is easy to work with
- Additional EDA showed that filtering to above 1300 RPM is more appropriate

EDA

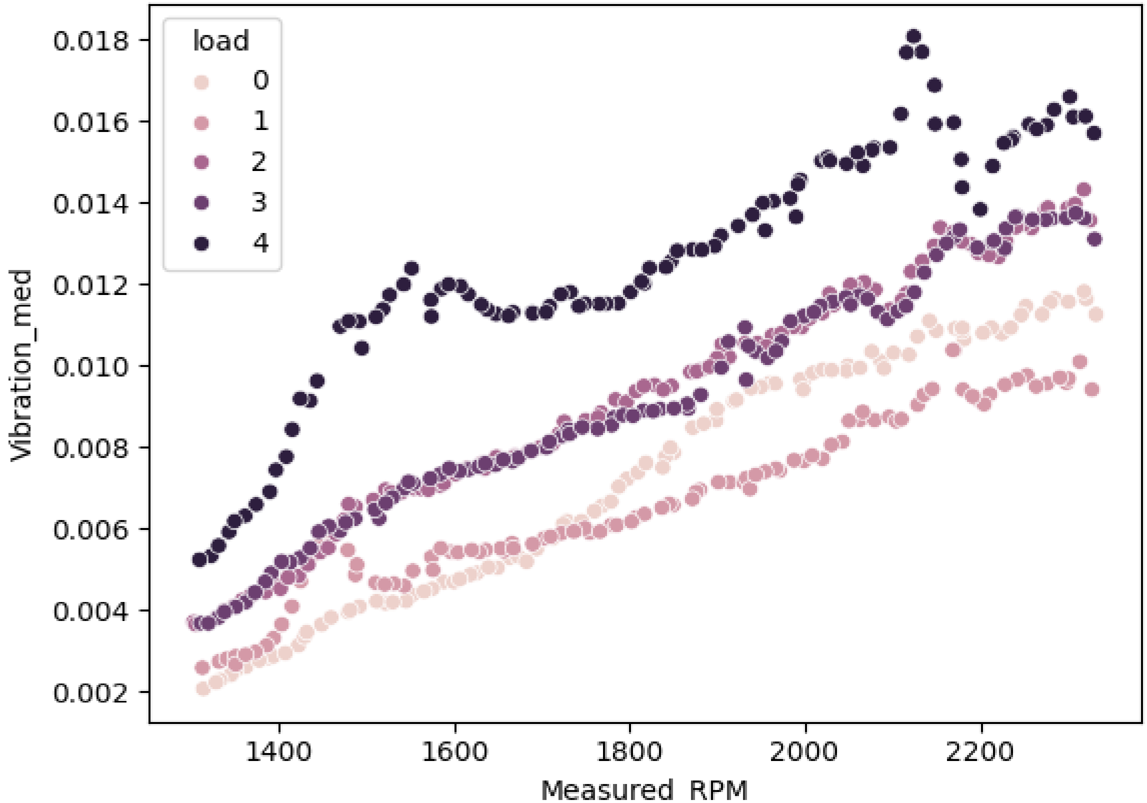
- Correlations between variables for the filtered and grouped data:



- We see that the vibration components are weakly related to load, but strongly correlated to each other
- The voltage and RPM components are very strongly related which we expect.
- **We can create a classification model with RPM and median vibration as inputs and 'load' as target variable**

EDA

- Characteristic plot of Vibration vs. RPM for different load levels



- *Based on the characteristic data, it was found that the 'D' and 'E' data had to be combined and re-split for successful analysis*
- It is almost impossible to resolve load levels 2 and 3
- There are several points for load level 0 that are higher than load level 1. This is unexpected

Model Steps

Favorable: The higher load levels are accurately predicted to prevent imminent damage (TP) - Sum of [(3,3), (4,4)].
Unfavorable 1: The higher load levels are predicted as lower loads (FN) - Sum of [(0,3), (1,3), (0,4), (1,4)]
Unfavorable 2: The lower load levels are mistaken as higher loads leading to false flags - Sum of [(3,0), (3,1), (4,1), (4,2)]

Where (a,b):(Predicted,True)

.

Based on the above, the following metric was used:

Model_score = favorable / (favorable + unfavorable_1 + unfavorable_2)

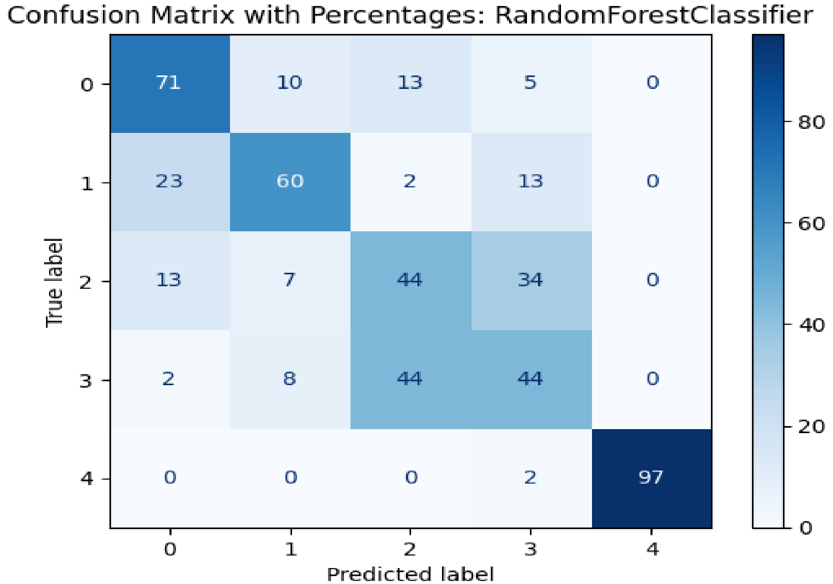
Model Score > .7

1. Pipeline: Scaling (MinMax), Model
2. Grid Search 10 fold CV
3. Confusion Matrix
4. Classification Report
5. Model Score

Model Performance

Random Forest Classifier (Score = .84):

```
RandomForestClassifier
Best Score, Best Params:  0.6251515151515152 {'Model__max_depth': 15, 'Model__n_estimators': 50}
```

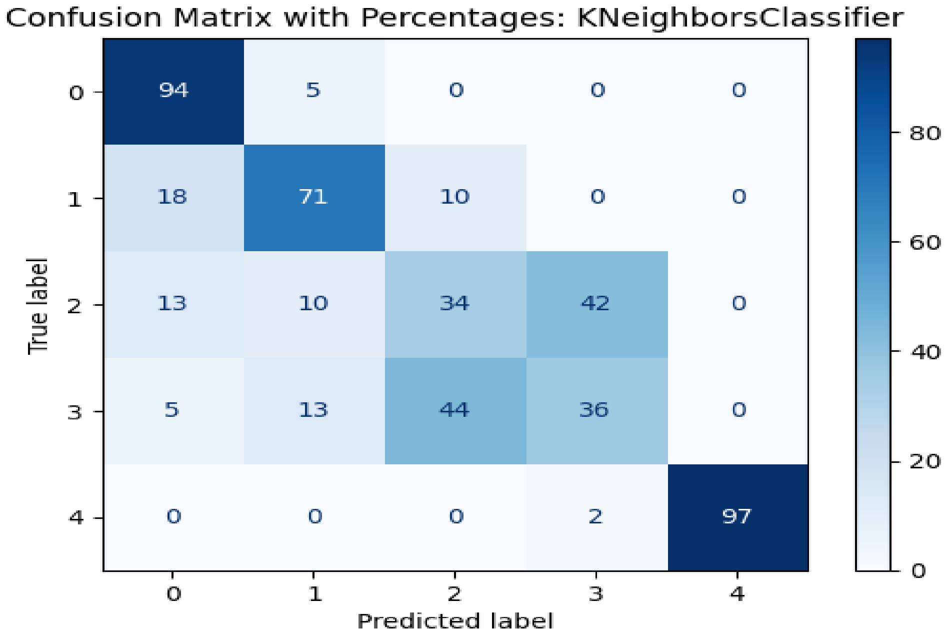


	precision	recall	f1-score	support
0	0.642857	0.710526	0.675000	38.000000
1	0.696970	0.605263	0.647887	38.000000
2	0.435897	0.447368	0.441558	38.000000
3	0.432432	0.444444	0.438356	36.000000
4	1.000000	0.976190	0.987952	42.000000
accuracy	0.645833	0.645833	0.645833	0.645833
macro avg	0.641631	0.636759	0.638151	192.000000
weighted avg	0.651277	0.645833	0.647519	192.000000

Model Score (Higher is better): 0.8382

Model Performance

KNN (Score = .89):



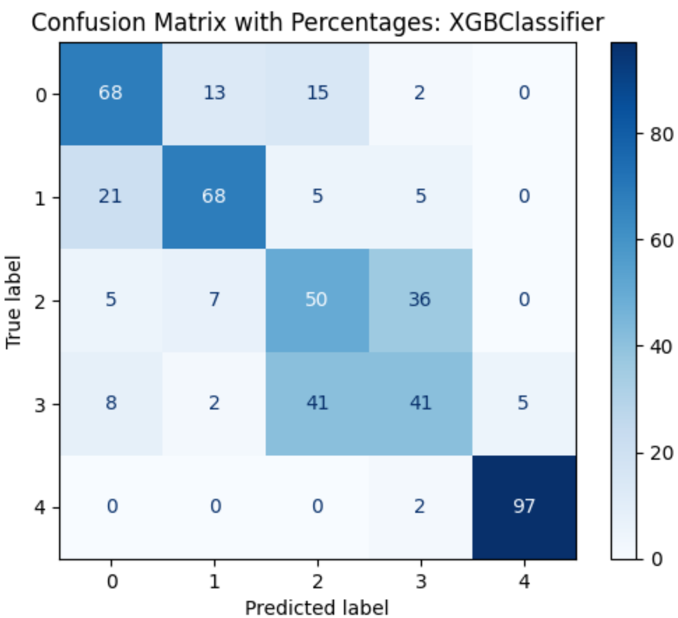
	precision	recall	f1-score	support
0	0.720000	0.947368	0.818182	38.000000
1	0.710526	0.710526	0.710526	38.000000
2	0.393939	0.342105	0.366197	38.000000
3	0.433333	0.361111	0.393939	36.000000
4	1.000000	0.976190	0.987952	42.000000
accuracy	0.677083	0.677083	0.677083	0.677083
macro avg	0.651560	0.667460	0.655359	192.000000
weighted avg	0.661092	0.677083	0.665011	192.000000

Model Score (Higher is better): 0.8852

Model Performance

XGB Classifier (Score = .89):

```
XGBClassifier
Best Score, Best Params:  0.5961111111111111 {'Model__booster': 'gbtree', 'Model__n_estimators': 100}
```

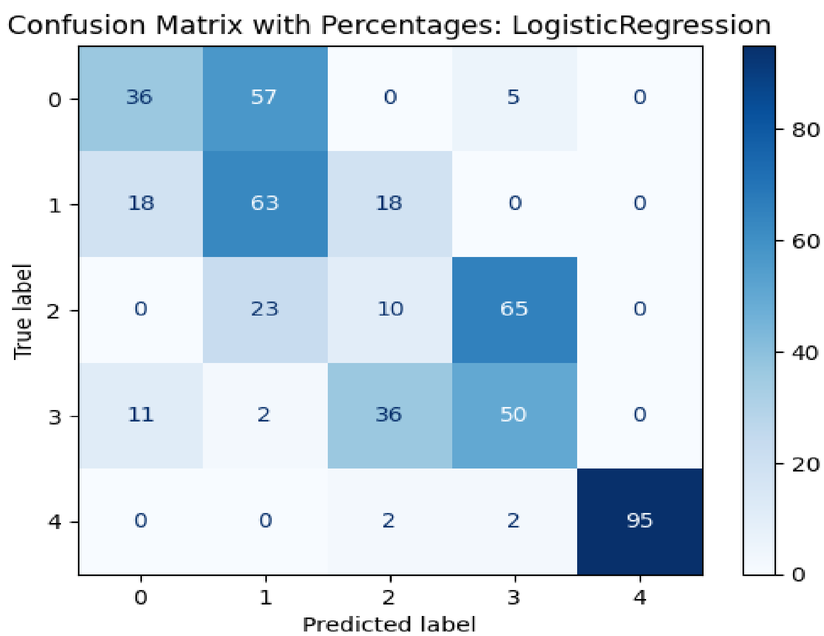


	precision	recall	f1-score	support
0	0.666667	0.684211	0.675325	38.000000
1	0.742857	0.684211	0.712329	38.000000
2	0.452381	0.500000	0.475000	38.000000
3	0.454545	0.416667	0.434783	36.000000
4	0.953488	0.976190	0.964706	42.000000
accuracy	0.661458	0.661458	0.661458	0.661458
macro avg	0.653988	0.652256	0.652428	192.000000
weighted avg	0.662305	0.661458	0.661201	192.000000

Model Score (Higher is better): 0.8889

Model Performance

Logistic Regressor (Score = .88):



	precision	recall	f1-score	support
0	0.560000	0.368421	0.444444	38.000000
1	0.428571	0.631579	0.510638	38.000000
2	0.160000	0.105263	0.126984	38.000000
3	0.391304	0.500000	0.439024	36.000000
4	1.000000	0.952381	0.975610	42.000000
accuracy	0.520833	0.520833	0.520833	0.520833
macro avg	0.507975	0.511529	0.499340	192.000000
weighted avg	0.519441	0.520833	0.509891	192.000000

Model Score (Higher is better): 0.8788

Model Performance Matrix

Model	Score
RF	.84
KNN Classifier	.89
XGB Classifier	.89
Logistic Regressor	.88

Although KNN, XGB and Logistic Regressor give very similar scores and better than RF, indicating all reasonable models, we can use a combination of model outputs for best results:

1. While KNN has significantly higher accuracy than others at level 0 (no unbalance), XGB outperforms KNN at almost all other levels, while managing to not falsely predict higher levels. Due to its higher accuracy at level 0, KNN can be used to overrule XGB if it predicts a zero.

2. XGB regressor is better than others in the critical region of classes 2 and 3, where the predictions are poor (due to low resolution as seen before). Hence XGB can be used for the other levels.

3. Logistic Regressor has poor TP at level

Conclusions

- We’ve obtained and transformed the data and used different models to try and predict the unbalance levels for a rotating equipment.
- Based on the model summary, we find the XGB Classifier to be the more reliable choice due to its score and its capacity to not overpredict level 2 unbalance. **It also meets the benchmark of 70% accuracy** we set in the proposal. However, the KNN model performs better at lower loads.
- This makes the XGB Classifier model choice for the task. The KNN model can be used to complement for predicting a shaft with no unbalance

Recommendations

- We can use the XGB regressor but consider KNN for the case of no unbalance prediction

Challenges

- Due to the very similar Vibration vs. RPM characteristics of levels 2 and 3, it was almost impossible to resolve them enough to classify them properly.
- Vibration levels for load level 0 exceeds that for load level 1. This is not expected and should be explored further
- The equipment must operate regularly above 1300 RPM (cutoff RPM) for the model to work successfully

Suggestions and Improvements

1. Explore reasons for why the level 0 vibrations exceed those of level 1 at higher RPMs. Probably a pre-existing unbalance? This might call for a repeat of the experiments depending on the findings
2. Vibration sensors can be placed at different radial orientations to see if it helps with getting more information to make better predictions especially at difficult to resolve levels such as 2 and 3.
3. Additional data techniques such as smoothing, in combination with more involved modeling such as neural networks might also help give better resolution ability.
4. To use the model, batch data with the same set of transformations that we used for the 'D' and 'E' data can be used to generate model input
5. Other model types could be explored to predict loading levels below cutoff RPM.