Springboard Data Science Capstone Project - Predicting Machine Failures

Ramaa Vissa

Mar 16, 2020

Contents

1. Introduction

Most manufacturing factories have a quality control unit that oversees the predictive maintenance. On the floor, machine failures can occur due to wear and tear of the machine over prolonged use. The problem is to capture the failed unit before it can cause more failures downstream.

The data presented here is obtained from simulation methods and this is a use case for machine predictive maintenance.
We would like to implement a machine learning algorithm to detect a bad device during a particular time frame and prevent problems before they occur.

This is a Multi-class classification problem. A machine learning algorithm is used to create the predictive model that is trained from historical data.

Problem: “what is the prediction that a particular machine will fail in a given timeframe, say next week/month/quarter.?”

- label/predictions is the ‘failure’.
- predictors/independent variables:

The Numerical features are Voltage, pressure, rotation, vibration and age of the machine.

The Categorical features are component and error. Volt, pressure, rotation, vibration are very interesting features that are suitable for feature transformation. The mean and std sampling are calculated over a rolling window of 24 and 3 hours and new features for volt, pressure,rotation and vibration are created. Total of 16 new features are created. The total features before feeding into the model is 30.

See slide

2. Data

The data has time based columns, numerical as well as categorical independent variables.

The acquired data sets were created by data simulation methods outside of this project. There are 5 data sets that are available for data analysis.

1. Telemetry

   The first data source is the telemetry time-series data which consists of voltage, rotation, pressure, and vibration measurements collected from 100 machines in real time averaged over every hour collected during the year 2015.

2. Errors

   The second data source is the error logs. These are non-critical errors that surface, while the machine is still operational and thus do not constitute as machine failure. The error date and times are rounded to the closest hour since the telemetry data is collected at an hourly rate.

3. Maintenance

   These are the scheduled and unscheduled maintenance records which correspond to both regular inspection of components as well as failures. A record is generated if a component is replaced during the scheduled inspection or replaced due to a breakdown. The records that are created due to breakdowns are called failures. The Maintenance data has both 2014 and 2015 records.

4. Failures

These are the records of component replacements due to failures. Each record has a date and time, machine ID, and failed component.

5. Machine

This data set includes some information about the machines like model type and age, the age here is the years in service.

First step would be to save all four data sets into the proper directories on the local machine.

3. Data Cleaning

There were no missing values when I inspected the data.

4. Experiment with different attribute combinations

First all the 5 datasets are inspected and joined to form one data set. The failures data set is joined and merged with the machine ID and timestamp of the telemetry data. (preserve the failures and match with the telemetry data).

Build out the error data set by making the categorical attributes into numeric by using get_dummies.

Build out the maint data set using the same transformer.

Build out failure data set.

The final feature set will look like this -

```
print(features.head())
features.describe()
```

```
  machineID            datetime  volt_mean_3h  rotate_mean_3h  \
0         1 2015-01-02 06:00:00    180.133784      440.608320
1         1 2015-01-02 09:00:00    176.364293      439.349655
2         1 2015-01-02 12:00:00    160.384568      424.385316
3         1 2015-01-02 15:00:00    170.472461      442.933997
4         1 2015-01-02 18:00:00    163.263806      468.937558

   pressure_mean_3h  vibration_mean_3h  volt_sd_3h  rotate_sd_3h  \
0         94.137969          41.551544   21.322735     48.770512
1        101.553209          36.105580   18.952210     51.329636
2         99.598722          36.094637   13.047080     13.702496
3        102.380586          40.483002   16.642354     56.290447
4        102.726648          40.921802   17.424688     38.680380

   pressure_sd_3h  vibration_sd_3h  volt_mean_24h  rotate_mean_24h  \
0        2.135684        10.037208     169.733809       445.179865
1       13.789279         6.737739     170.614862       446.364859
2        9.988609         1.639962     169.893965       447.009407
3        3.305739         8.854145     171.243444       444.233563
4        9.105775         3.060781     170.792486       448.440437

   pressure_mean_24h  vibration_mean_24h  volt_sd_24h  rotate_sd_24h  \
```

0	96.797113	40.385160	11.233120	48.717395
1	96.849785	39.736826	12.519402	48.385076
2	97.715600	39.498374	13.370357	42.432317
3	96.666060	40.229370	13.299281	41.346121
4	95.766838	40.055214	13.954518	43.490234

	pressure_sd_24h	vibration_sd_24h	error1count	error2count	error3count \
0	10.079880	5.853209	0.0	0.0	0.0
1	10.171540	6.163231	0.0	0.0	0.0
2	9.471669	6.195076	0.0	0.0	0.0
3	8.731229	5.687944	0.0	0.0	0.0
4	8.061653	5.898069	0.0	0.0	0.0

	error4count	error5count	comp1	comp2	comp3	comp4	model	age
0	0.0	0.0	20.000	215.000	155.000	170.000	model3	18
1	0.0	0.0	20.125	215.125	155.125	170.125	model3	18
2	0.0	0.0	20.250	215.250	155.250	170.250	model3	18
3	0.0	0.0	20.375	215.375	155.375	170.375	model3	18
4	0.0	0.0	20.500	215.500	155.500	170.500	model3	18

To find a good attribute combination, it is advisable to try different combinations, for example, should we consider another feature that could be a mean or std of the original feature.

5. Data Transformation

Even though the original data sets did not have any missing values, good judgement was used to eliminate Nans or fill with appropriate values as part of the data wrangling/data analysis while merging and joins. New variables were created before feeding to the model.

These were created for each volt, rotate, pressure, vibration.

Transformation steps need to be executed in the right order. We can use scikit-learn's Pipeline class to automate the pipelines once the transformers are finalised.

6. Exploratory Data Analysis

6.1. Introduction to the cleaned data

The purpose of the EDA is to find out the relationship between the predictors and labels/dependent variables. We can plot scatter plots between Voltage and rotate, voltage and pressure, or vibration to find the relationship between those variables. A bar graph can be plotted for the categorical attributes. These plots are important so that we know the distribution of the predictors and labels.

There are several predictor columns of interest, some needed transformations. All the object data types were converted to either string or numeric. The Columns are either numeric or categorical, the categorical columns were transformed to represent the respective classes.

The target column is the failure prediction. The data is divided into training set by the ‘maintenance time’, so if the data is trained on this month’s information, the prediction is validated in the next month's data. The target column of interest is the failure column which contains ‘none’ or the ‘failed component’.
After the data exploration we should have clearly defined target columns and feature columns.

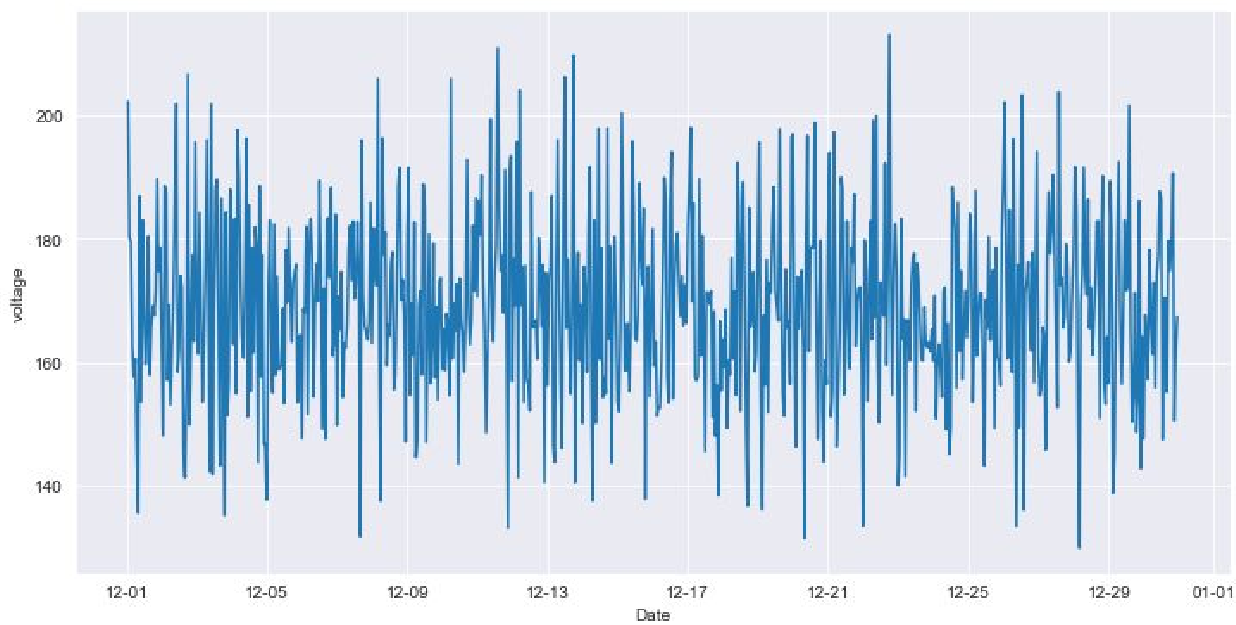
The correlation matrix gives us an intuition on correlation of the features and shows the features that are highly correlated. It is advisable to not have correlated features trickle into the model estimation, which can undoubtedly in theory mask as ‘noise’. The following are some corr_matrix results.

```
rotate       1.000000
machineID    0.000171
pressure    -0.000688
volt        -0.001511
vibration   -0.003056
Name: rotate, dtype: float64
```

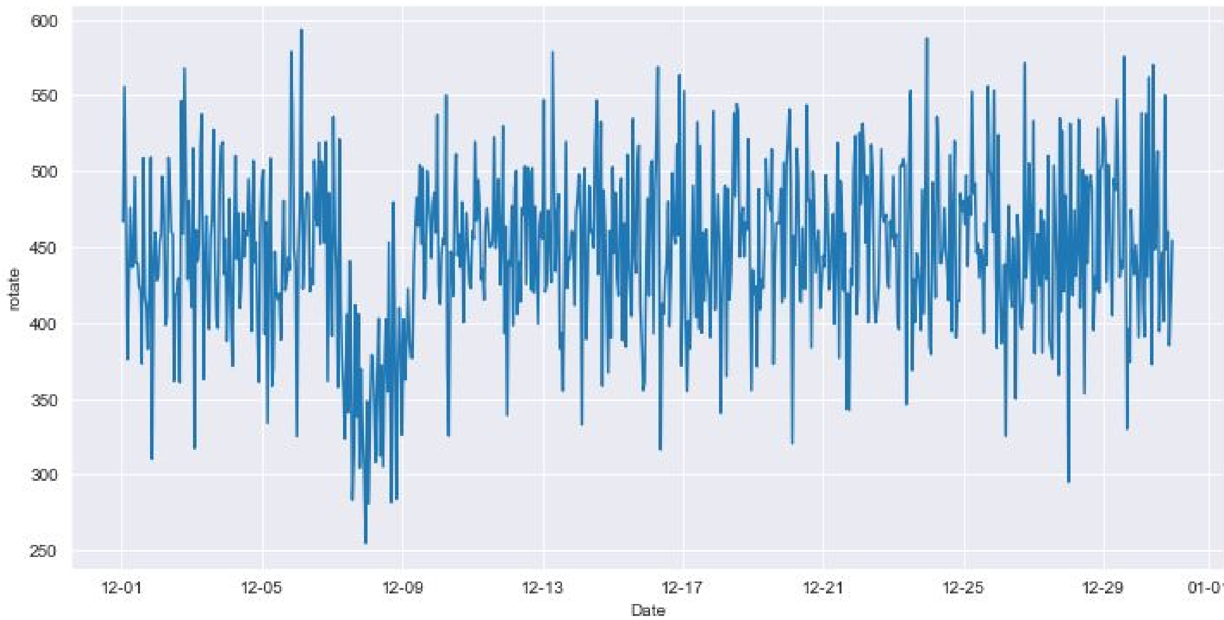
```
vibration    1.000000
volt         0.002390
pressure     0.001395
machineID   -0.000922
rotate      -0.003056
Name: vibration, dtype: float64
```

6.2. Plots

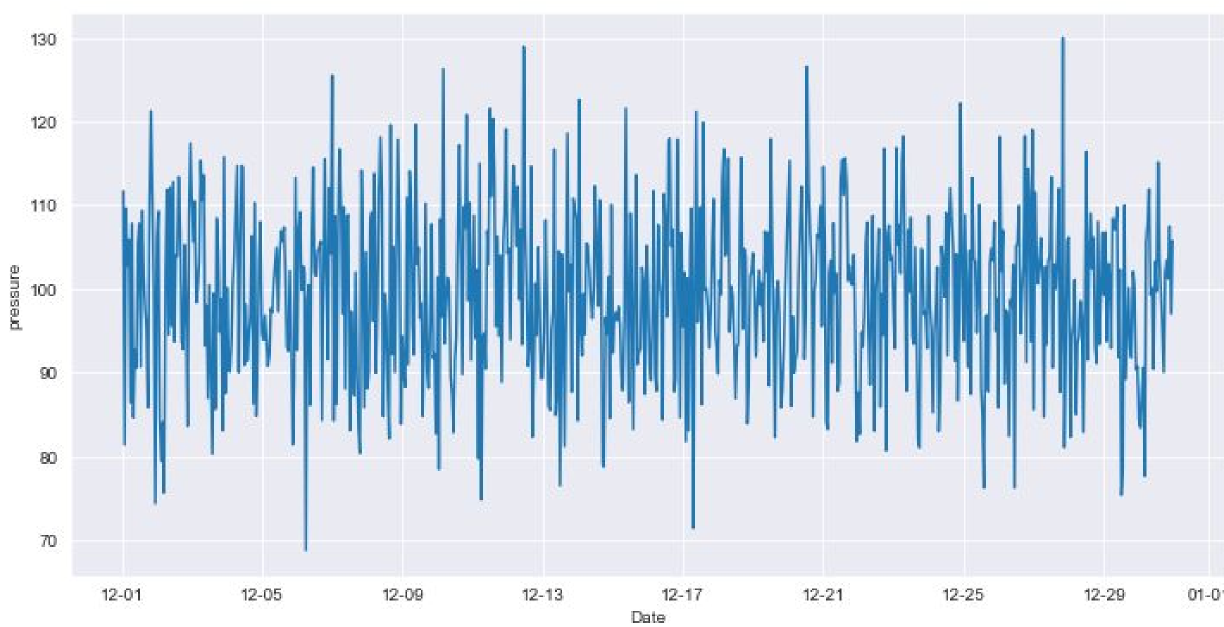
The telemetry data is a time series data. The following is a plot of the telemetry data - voltage for 1 machine with machine id 100 and between time periods for 1 month from 12/01 to 12/31.



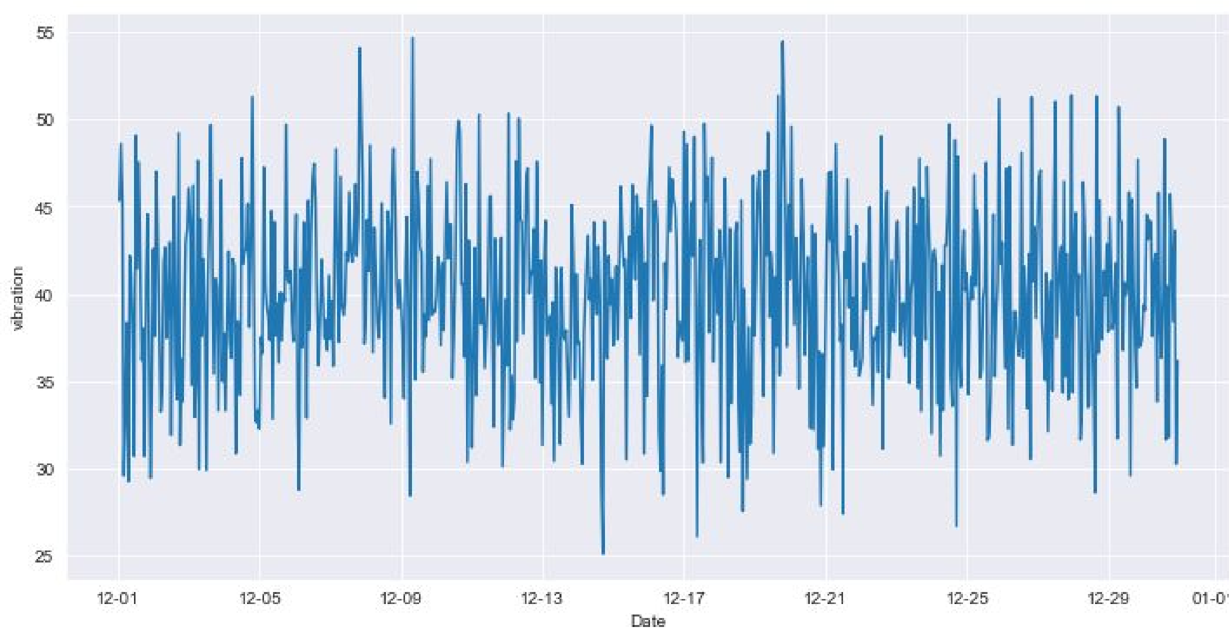
The following is a plot of the telemetry data - rotate for 1 machine with machine id 100 and between time periods for 1 month from 12/01 to 12/31.



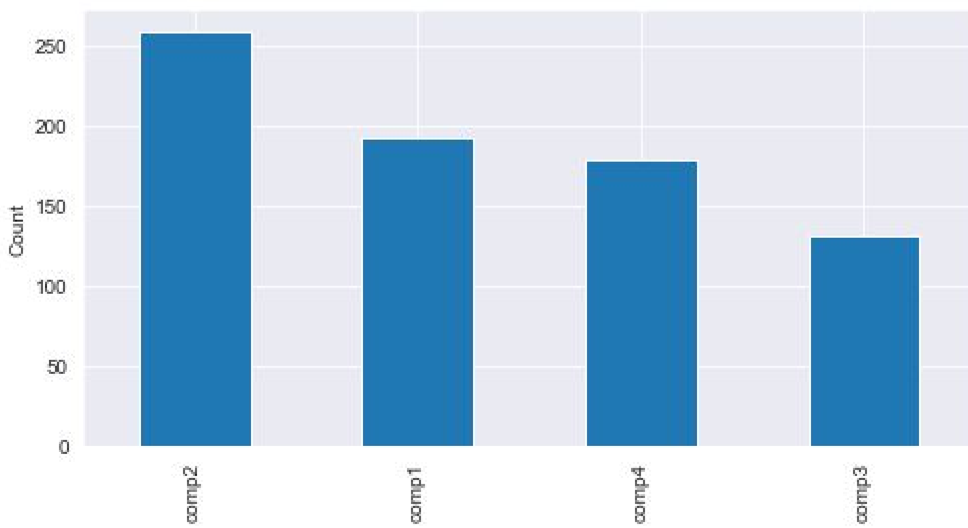
The following is a plot of the telemetry data - pressure for 1 machine with machine id 100 and between time periods for 1 month from 12/01 to 12/31.



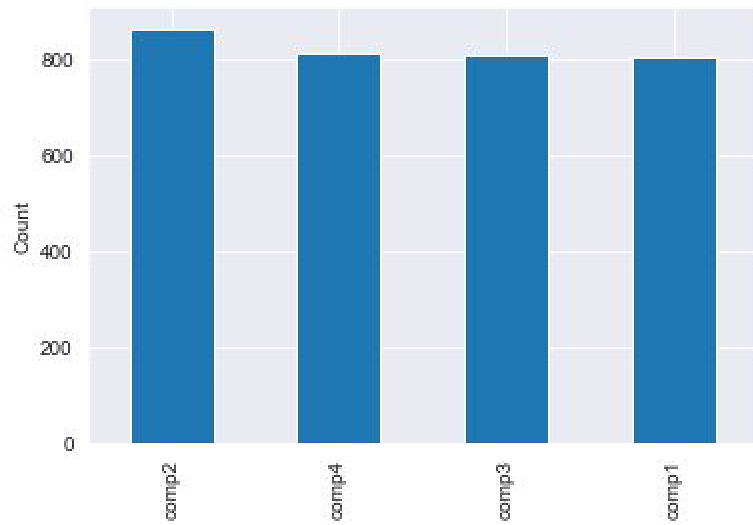
The following is a plot of the telemetry data - vibration for 1 machine with machine id 100 and between time periods for 1 month from 12/01 to 12/31.



The following is the bar graph showing counts of each type of components failure.

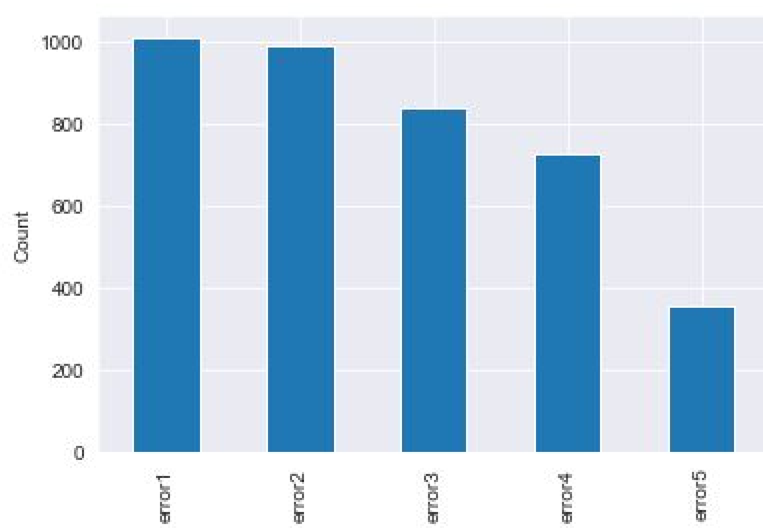


The following is the bar graph showing counts of each type of component maintenance.

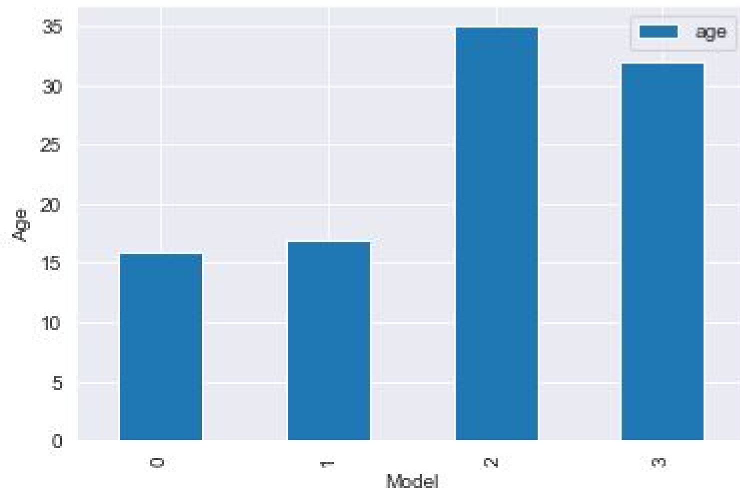


By intuition the failures occurring on component 2 are causing more maintenance leading to either repair or replacement.

The following is the bar graph showing counts of each type of error, errors are not critical like failures.



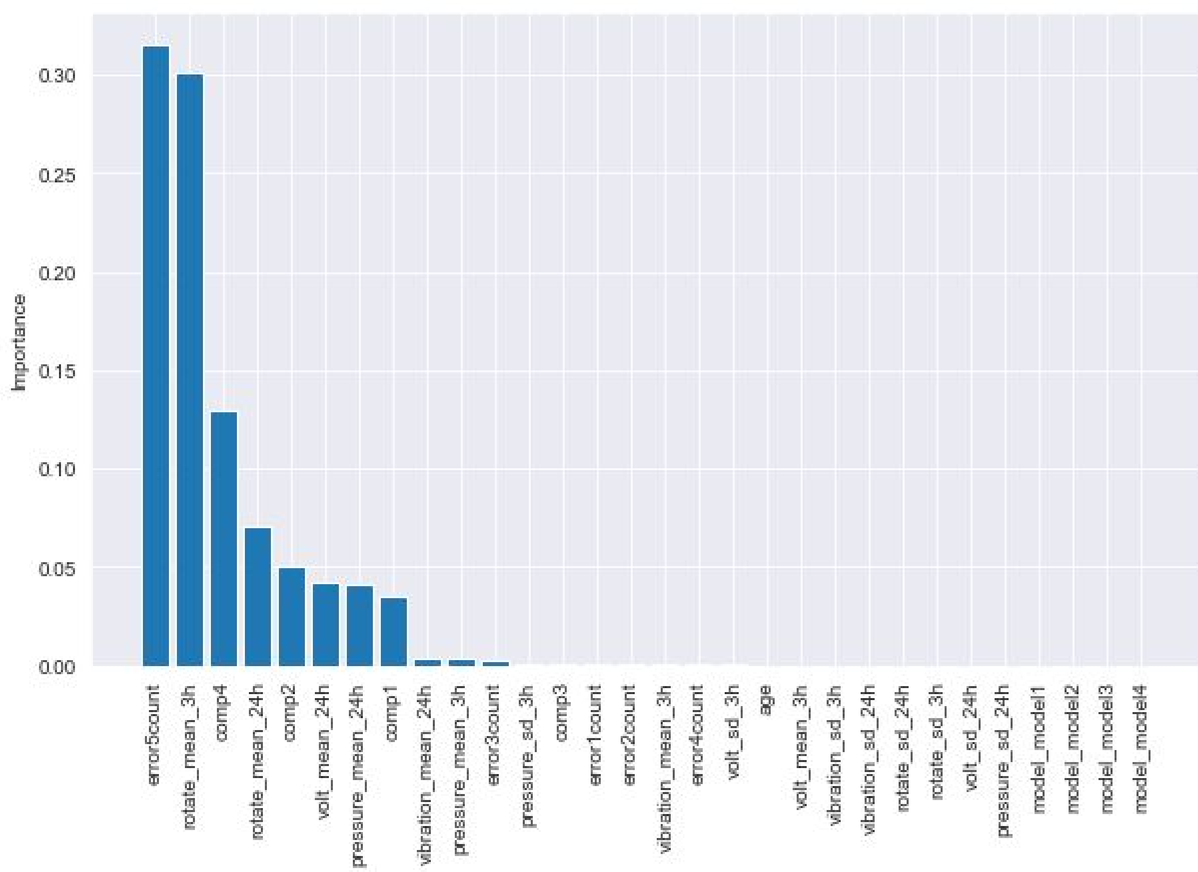
The following is the bar graph showing counts of each type of machine grouped by age.



7. Feature Selection

It should be clear from a domain expert perspective, given the features, we can assume that the volt, pressure and rotation are the most important characteristics of the machine performance.

The feature importance matrix, confirms our choice and transformation of attributes.



8. Split the data into train and test data

In this Use case the training data is a month behind the predictions month. So if the model is trained on a month's worth of data, the predictions are obtained for the following month. This is a hypothetical scenario, in practise I have made the executive decision to go with my own split of train/test data set and can change until the prototype is finalised and stabilised. If the use case warrants for a different window, this will change.

More efficient is scikit-learn's K-fold cross validation.

9. Model

Knowing the labels for the machine failures, 0 for component 1 failure, 1 for component 2 failure, 2 for component 3 failure and finally 3 for component 4 failure, we use the supervised machine learning algorithms to build a predictive model. Since there are only 4 classes in the data, we use the multi class classification algorithms. The first step involves splitting the data, for the test model, the data is split depending on the month of the date and time of failure.

Several models were evaluated and the best classifier was evaluated. After evaluation, gradient Boosting was chosen as the dominant classifier.

We can use randomised search or grid search to further fine tune the model and look for hyper parameters that could present a model that can enhance in accuracy, recall or precision scores.

10. Evaluation and Optimization

Confusion matrix was used to evaluate the Precision and Recall, this use case requires a high Recall. In Predictive maintenance it's important to know how many of the actual failures were predicted by the model..aka Recall. Here the recall rates for all components as well as none -aka no failures are all above 90%, meaning the model was able to capture above 90% of the failures correctly. (Recall becomes more important as the consequences of false negatives, aka true failures that the model did not predict, exceed the consequences of false positives, aka false predictions of future failure.)

Best Practices

- It's good to save the models while tuning as pickle files. The training data can be saved in temporary files.
- The feature importance should be in the inner loop and not in the data cleaning phase, so that all the features are relevant in the model selection.
- Use of pipelines for efficiency, and reuse of code.

Further Improvements.

Productionize the model: The model needs to be trained and make sure the evaluation metrics are still valid on new test data. There could be a possibility of overfitting on production /testing if the model is not refreshed and saved with a different version. A scheduling job like crontab can be employed to make sure the model is refreshed. The ‘hotswitch’ could be used to ensure seamless operation in production without any disruption to any services. Make sure you employ a version control to ensure that you can compare the metrics from an earlier model to later model.