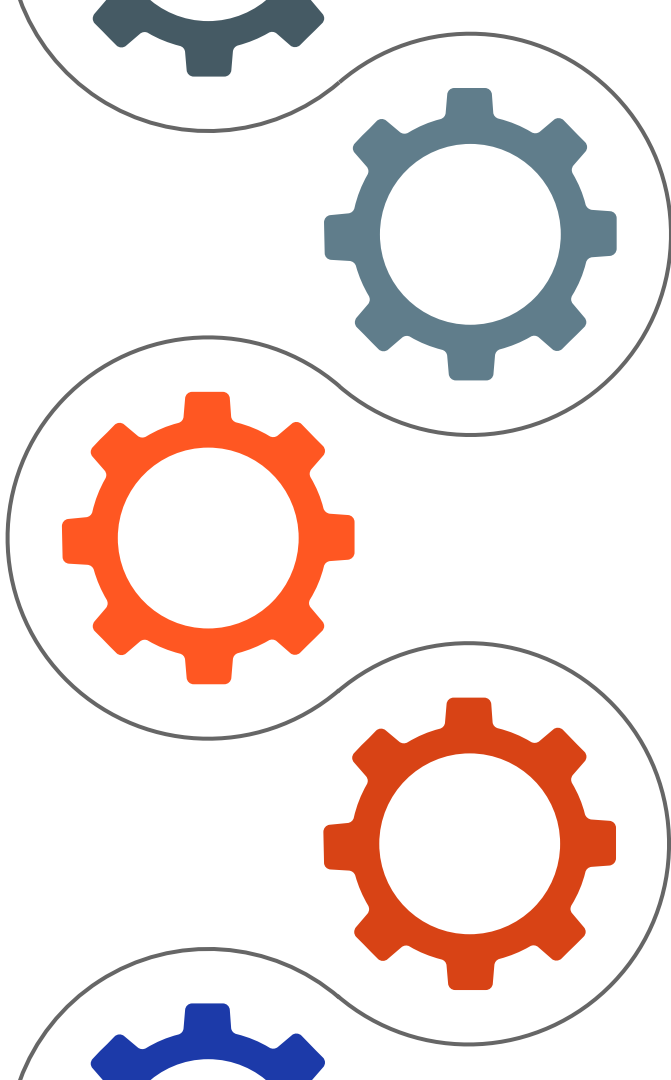
Gearbox Fault Diagnosis Using Machine Learning Techniques

Presented by
Rishav Kumar(B19ME066)
Sanodariya Kshitij Ashvinchandra (B19ME075)



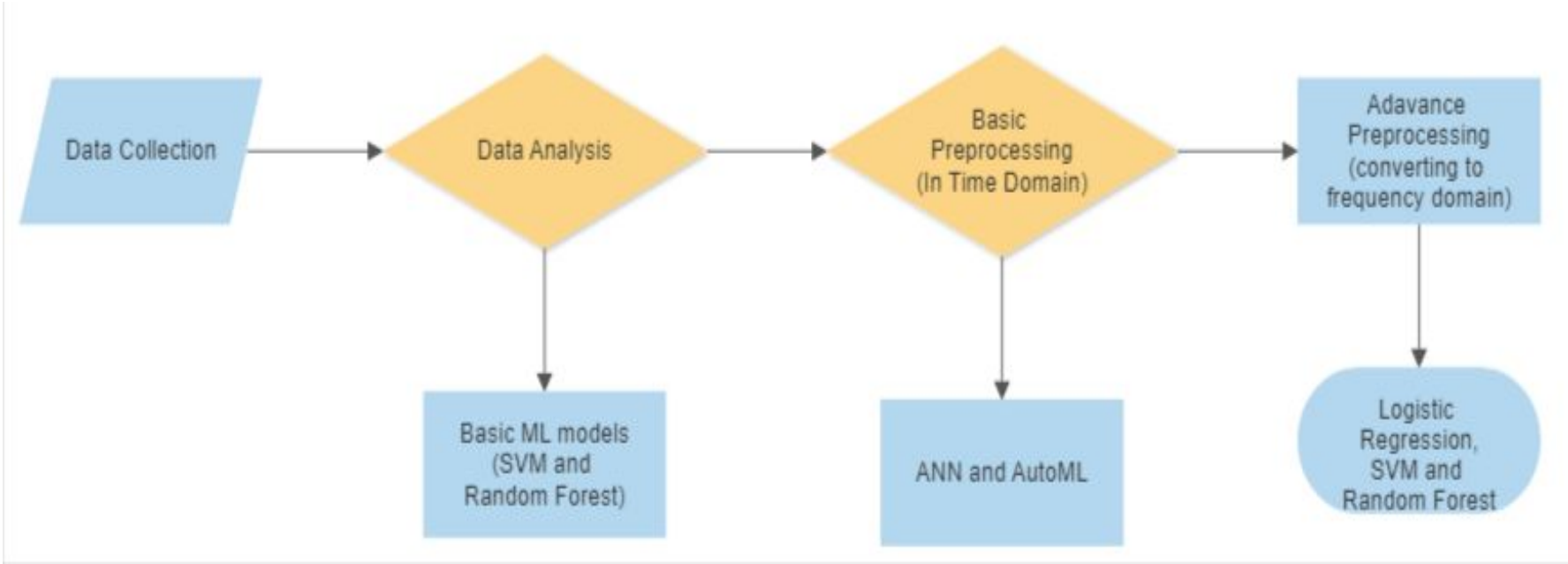
Introduction

- The use of gears is crucial in the world of rotating machinery and mechanical transmission systems
- It is essential to properly monitor the gear system to guarantee the effectiveness of mechanical transmission systems
- Any producing industry will experience production losses and higher maintenance costs as a result of the failure of such essential components.
- Regular maintenance and feedback are required to avoid gearbox faults
- These flaws can be found by analyzing the vibration data of the gearbox system that was gathered by electromechanical sensors.

OBJECTIVES

- To design a machine learning model for Gearbox fault diagnosis
- To design a model which can run without the internet and the whole process can be carried out locally.

Methodology



Dataset

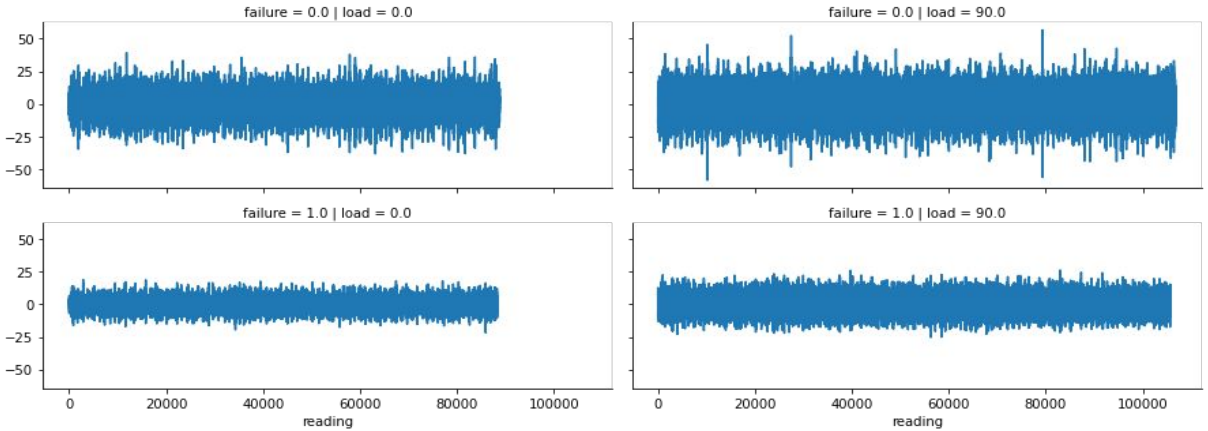
- The dataset used for the project is open-sourced. It is taken from Open Energy Data Initiative(OEDI)
- Data set include the vibration dataset recorded by using SpectraQuest Gearbox Fault Diagnostics Simulator
- Four accelerometer(vibration sensors), positioned in four distinct directions and with a load variation of "0" to "90," were used to record the data set
- A 30Hz data rate was used to obtain vibration data
- There are two distinct scenarios in the entire dataset:

  1) A healthy state and

  2) a broken tooth state

Data Analysis

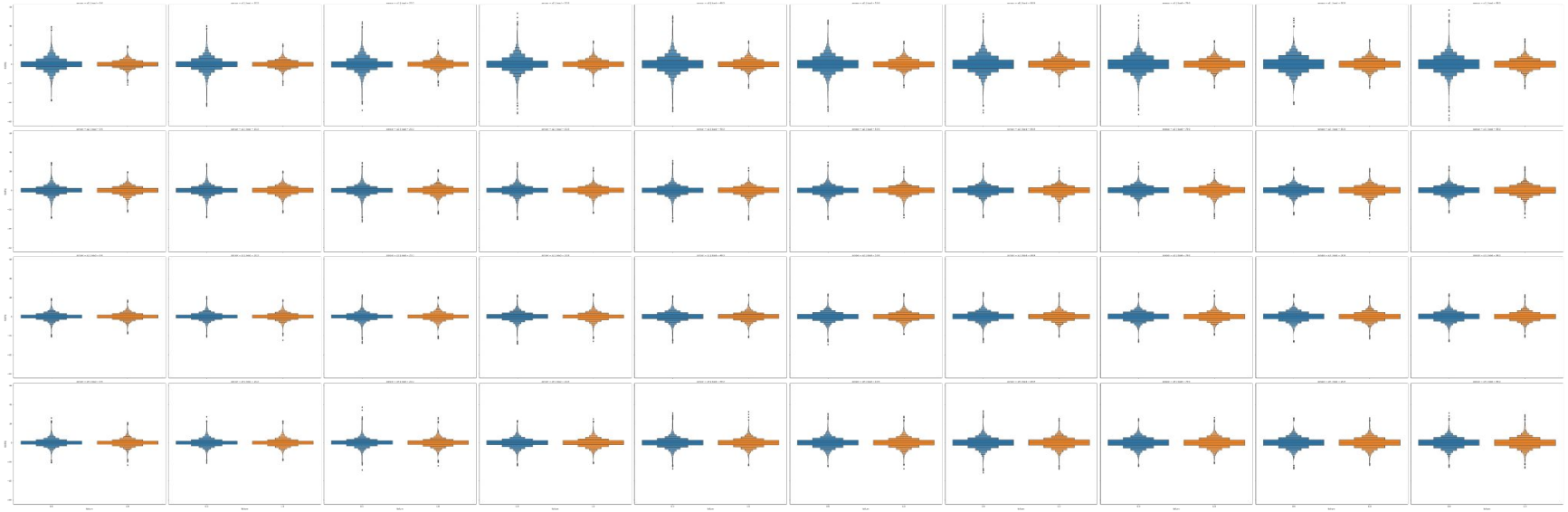
Plot Amplitude vs Reading



We can observe here that amplitude for healthy gearbox is more than the unhealthy.

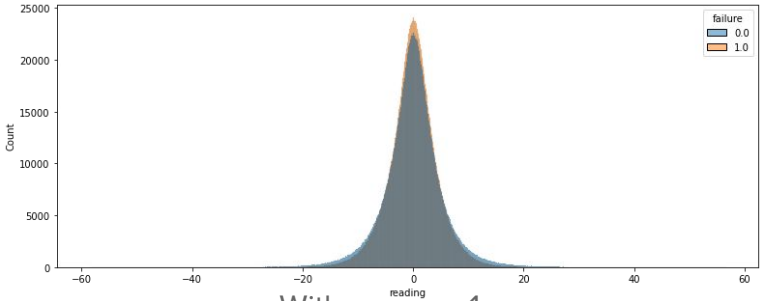
Plotting Boxplot

There are 10 column with increasing load from 0 to 90% and 4 rows which contain data of 4 sensor

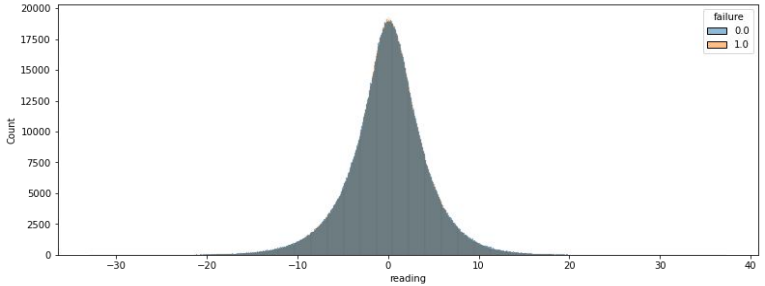


It is clear from the plot above that sensor a1 has a much larger amplitude difference, which gets larger with increasing load. There is a slight but distinguishable difference in sensor a2, as well. There doesn't seem to be much of a difference between sensors a3 and a4.

Plotting Amplitude Distribution



With sensor a1



Without sensor a1

- When sensor a1 is removed, it can be seen that the two distributions almost perfectly overlap
- It can be concluded that the gearbox had some other issue at the side of the a1 sensor that was causing the increased reading amplitude
- On calculating the std deviation we get more deviation on data with exclusion of sensor a1
- Excluding the dataset corresponding to sensor a1 can result in improving the accuracy of the model.

Experimental Work

SVM(Support Vector Machine)

- SVM is a Supervised Learning algorithm, which is used for Classification as well as Regression problems but majorly it is used for classification
- It creates decision boundary that can divide n-dimensional space into classes so that we can quickly classify new data points in the future

	precision	recall	f1-score	support
0.0	0.58	0.58	0.58	10061
1.0	0.57	0.57	0.57	9739
accuracy			0.58	19800
macro avg	0.58	0.58	0.58	19800
weighted avg	0.58	0.58	0.58	19800

Random Forest

- It is a supervised learning technique based on the idea of ensemble learning which is the act of integrating several classifiers to solve a complicated issue and enhance the performance of the model
- More trees in the forest result in increased accuracy and reduce the overfitting issue

	precision	recall	f1-score	support
0.0	0.71	0.35	0.47	10083
1.0	0.56	0.85	0.67	9717
accuracy			0.60	19800
macro avg	0.63	0.60	0.57	19800
weighted avg	0.64	0.60	0.57	19800

Basic Preprocessing

- Initially we have only 5 feature vectors i.e 4 sensors reading a1,a2,a3,a4, and one load value
- We increase the feature vector by adding mean, std deviation, Kurt, skew and moment by grouping 1000 samples together
- This will increase the feature vector and reduce the size of the dataset
- After preprocessing we applied both ANN and automl to the dataset generated to check can we get better results

AutoMl

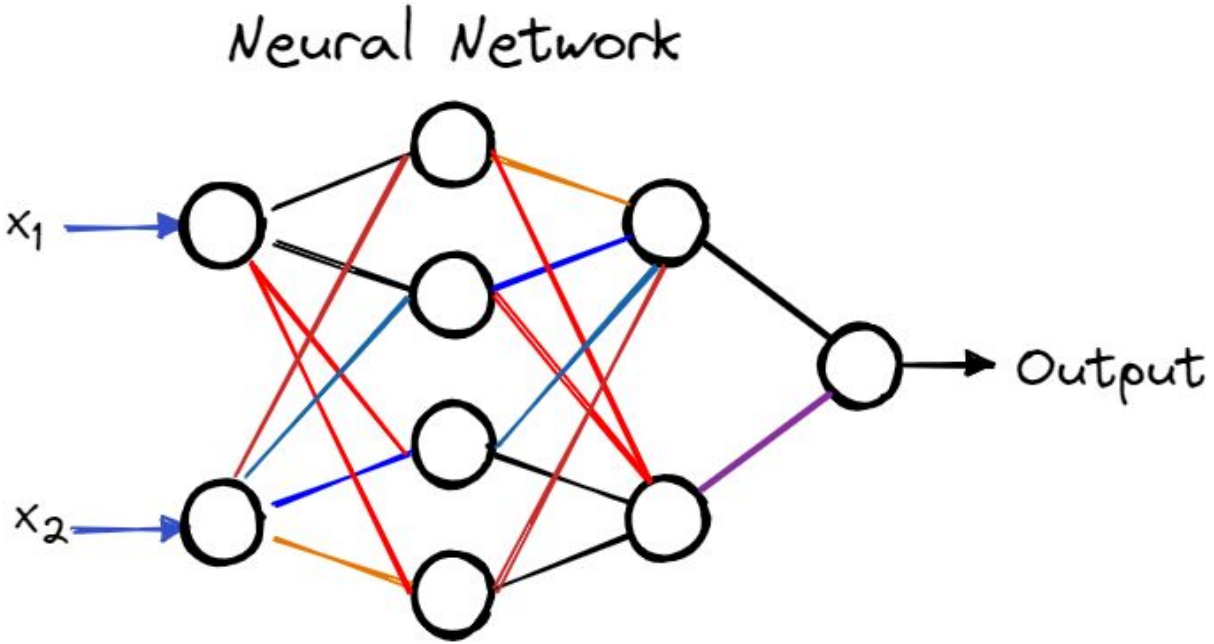
- It is the process of automating the time-consuming, iterative tasks of machine learning model development
- AutoML automatically finds and employs the best machine-learning algorithm for a certain task
- It is designed to enable us to easily analyze our data and determine which models, features, and characteristics work better than others

```
from tpot import TPOTClassifier

%%time
tpot_td = TPOTClassifier(generations=5, random_state=42, max_time_mins=2)
tpot_td.fit(X_train, y_train)
print(f'Best accuracy score: {tpot_td.score(X_test, y_test):0.3%}')
```

```
Best accuracy score: 86.381%
CPU times: user 2min 11s, sys: 47.4 s, total: 2min 59s
Wall time: 2min 9s
```

ANN

- The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain.
- The topology, weight vector, and activation functions of ANNs define them.
- They are composed of three layers: an input layer that receives signals from the outside world; a hidden layer that processes those signals; and an output layer that communicates the processed signals to the outside world.



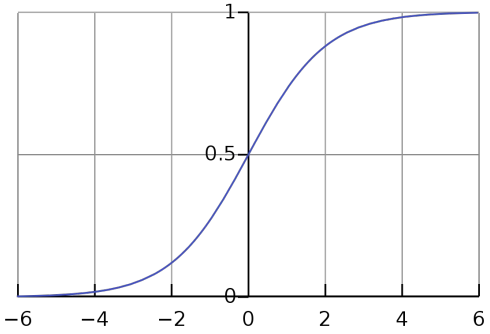
ANN

- We created an ANN model by taking 5 dense hidden layers with an activation function as ‘relu’ and one output layer with ‘sigmoid’ as activation function.
- The optimizer chosen is ‘adam’(Adaptive Moment Estimation) and for loss, we have taken binary cross entropy.

```
[17] import tensorflow as tf
     from tensorflow import keras


     model = keras.Sequential([
         keras.layers.Dense(100, input_shape=(9,), activation='relu'),
         keras.layers.Dense(50, activation='relu'),
         keras.layers.Dense(25, activation='relu'),
         keras.layers.Dense(12, activation='relu'),
         keras.layers.Dense(6, activation='relu'),
         keras.layers.Dense(3, activation='relu'),
         keras.layers.Dense(1, activation='sigmoid')
     ])

     # opt = keras.optimizers.Adam(learning_rate=0.01)

     model.compile(optimizer='adam',
                   loss='binary_crossentropy',
                   metrics=['accuracy'])

     hist=model.fit(X_train, y_train, epochs=100)
```

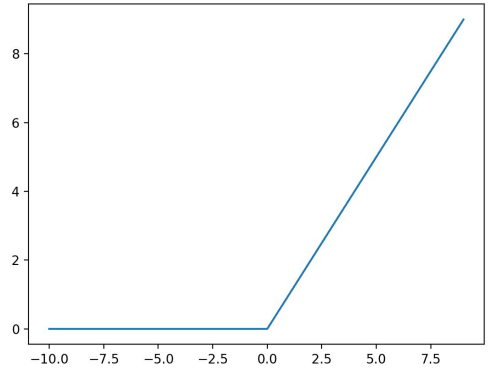
Sigmoid

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} = 1 - S(-x).$$



Relu

$$f(x) = x^+ = \max(0, x),$$



ANN

- From ANN we get an accuracy of around 88% which is much more than the classical machine-learning method.

```
Epoch 98/100
101/101 [==============================] - 0s 3ms/step - loss: 0.2474 - accuracy: 0.8832
Epoch 99/100
101/101 [==============================] - 0s 3ms/step - loss: 0.2410 - accuracy: 0.8910
Epoch 100/100
101/101 [==============================] - 0s 3ms/step - loss: 0.2459 - accuracy: 0.8838
```

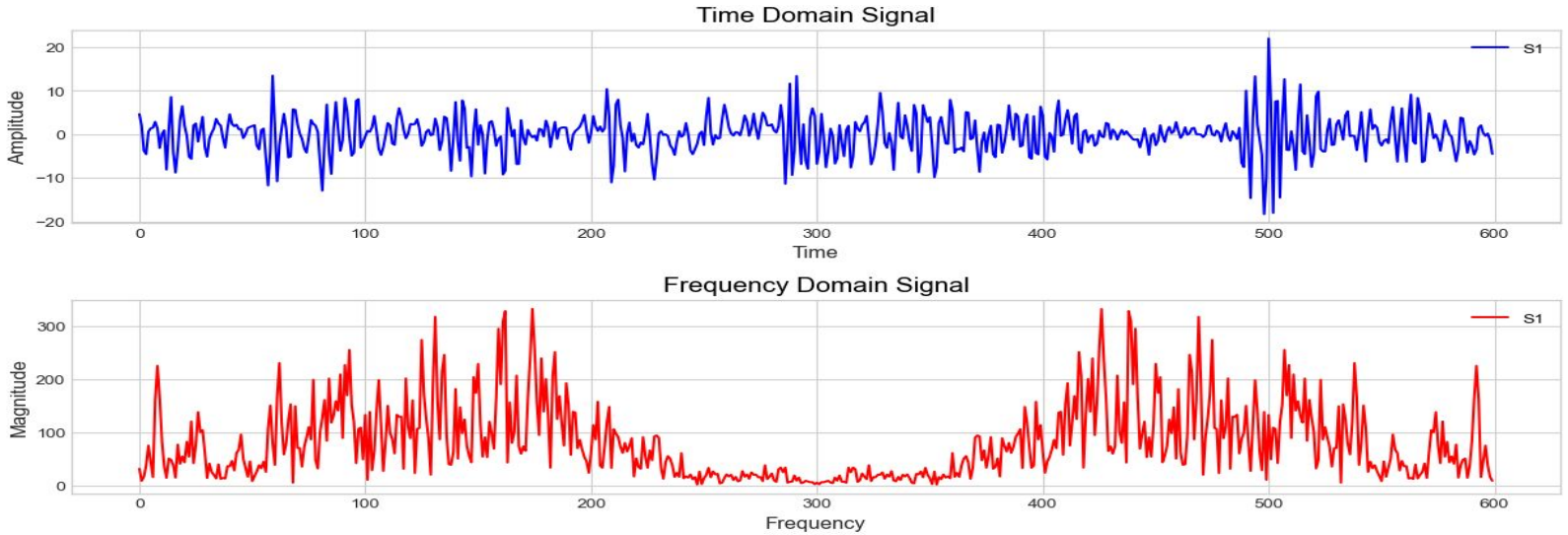
Better Pre-Processing

- As mentioned we did process the data but the results were not that satisfactory, the reason being not enough parameters to understand the data. So we did some more research and came across the idea of using the **Fourier Transform** to convert the time domain data to the frequency domain and applied RMS (Root Mean Square to find the effective value). We also sampled an equal amount of data from all columns.
- Dataset consists of 10 files of faulty gearbox data and healthy gearbox data each at a different load percentage. Each of these files has a different number of data points because of the different time lengths of the experiment. Hence, to make each load percentage experiment data of the same length, we need to bring all the datasets at the same dimension. Hence, this function will select the first **88200** data points from each file. As it is perfectly divisible by **300** and none file had a lesser amount of data points.

- The shape of the dataset after trimming is **(1764000, 7)**. The dataset shall be separated based on fault, i.e. "Healthy" or "Broken". From this data, we have to create a feature that shall be used to train our machine-learning algorithms to predict the type of fault.

Better Pre-Processing

1. FFT: - FFT function is to convert a signal from the time domain to the frequency domain. This function takes a pandas Dataframe as an input argument and spits out an absolute value of FFT in Pandas Dataframe format.



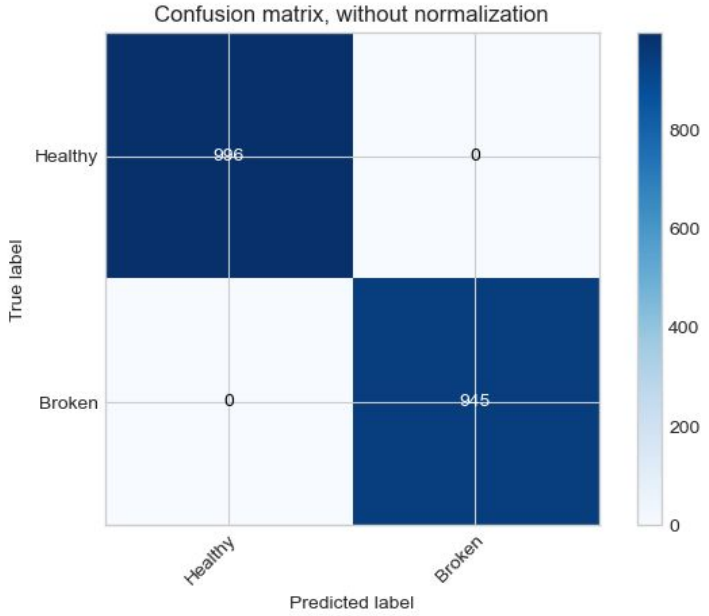
2. RMS: - This function is responsible to take the RMS values of all the data point that lies in the frequency bin of a single frequency. In this setting, each bin contains 10 data points.

3. Arrange Dataframe: - This function is used to rearrange the RMS values of the frequency of each sensor and rename the columns depending upon the frequency number (ranging from 1 to 15 Hz).

Logistic Regression

- Logistic regression is a statistical method used to predict the outcome of a dependent variable based on previous observations. It's a type of regression analysis and is a commonly used algorithm for solving binary classification problems. Logistic regression is a classification algorithm that predicts a binary outcome based on a series of independent variables. For our problem, we have two possible outcomes "Healthy" or "Broken". Logistic regression works by measuring the relationship between the dependent variable (what we want to predict) and one or more independent variables (the features).

- We used the processed data (mentioned above) and applied Logistic Regression to train the classifier. The accuracy of the model is **100%** but we can't always say that as the input database was not that big but in our test, it worked every time.



Using The processed Data with Previous Model

Random Forest:

	precision	recall	f1-score	support
0.0	0.71	0.35	0.47	10083
1.0	0.56	0.85	0.67	9717
accuracy			0.60	19800
macro avg	0.63	0.60	0.57	19800
weighted avg	0.64	0.60	0.57	19800

Raw Data

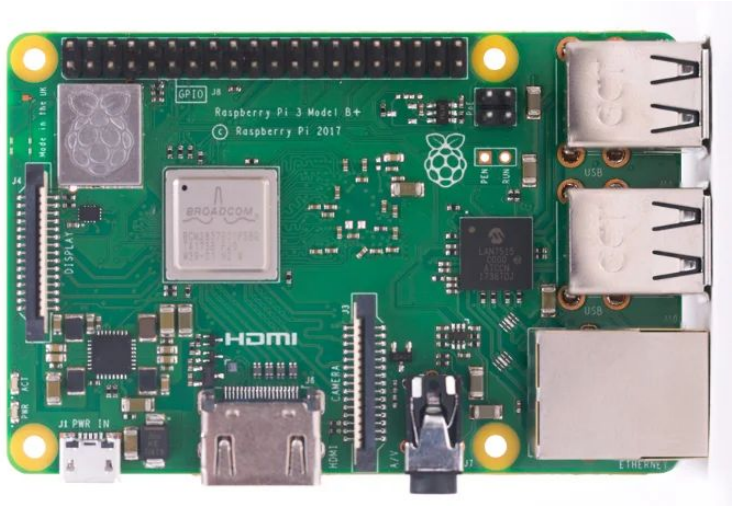
Processed Data

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100, max_depth=2,random_state=0)
clf.fit(X_train, y_train)
y_predict = clf.predict(X_test)
print("The Random Forest Classifier Score is: -   " + str(clf.score(X_test, y_test)))


plot_confusion_matrix(y_test, y_predict, classes=class_names, title='Confusion matrix, without normalization')
plt.show()
```

```
The Random Forest Classifier Score is: -   0.9895506792058516
```

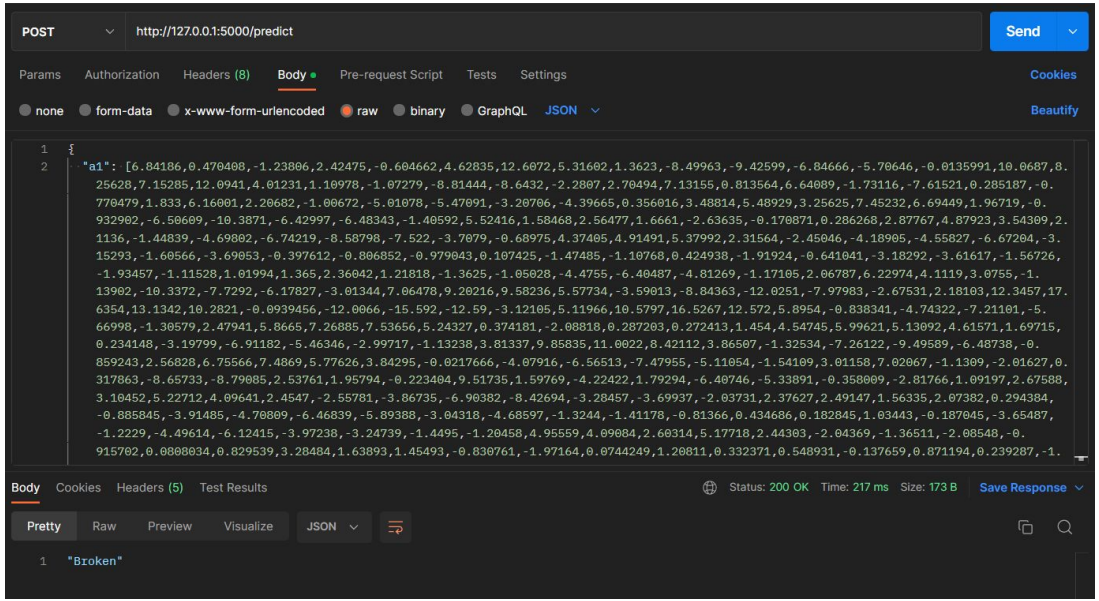
Real World

- We can not use the dedicated machine for every task and it will not be economically good choice for the company.
- With the development of new microcontrollers and computer vision algorithms, quality testing has dramatically improved.
- We need to go with good performing microcontroller and the best choice is Raspberry PI.
- We choose Raspberry PI 4 as older versions of PI didn't had onboard WiFi network so we can operate the PI from anywhere even with the phone(as we did).



Real World

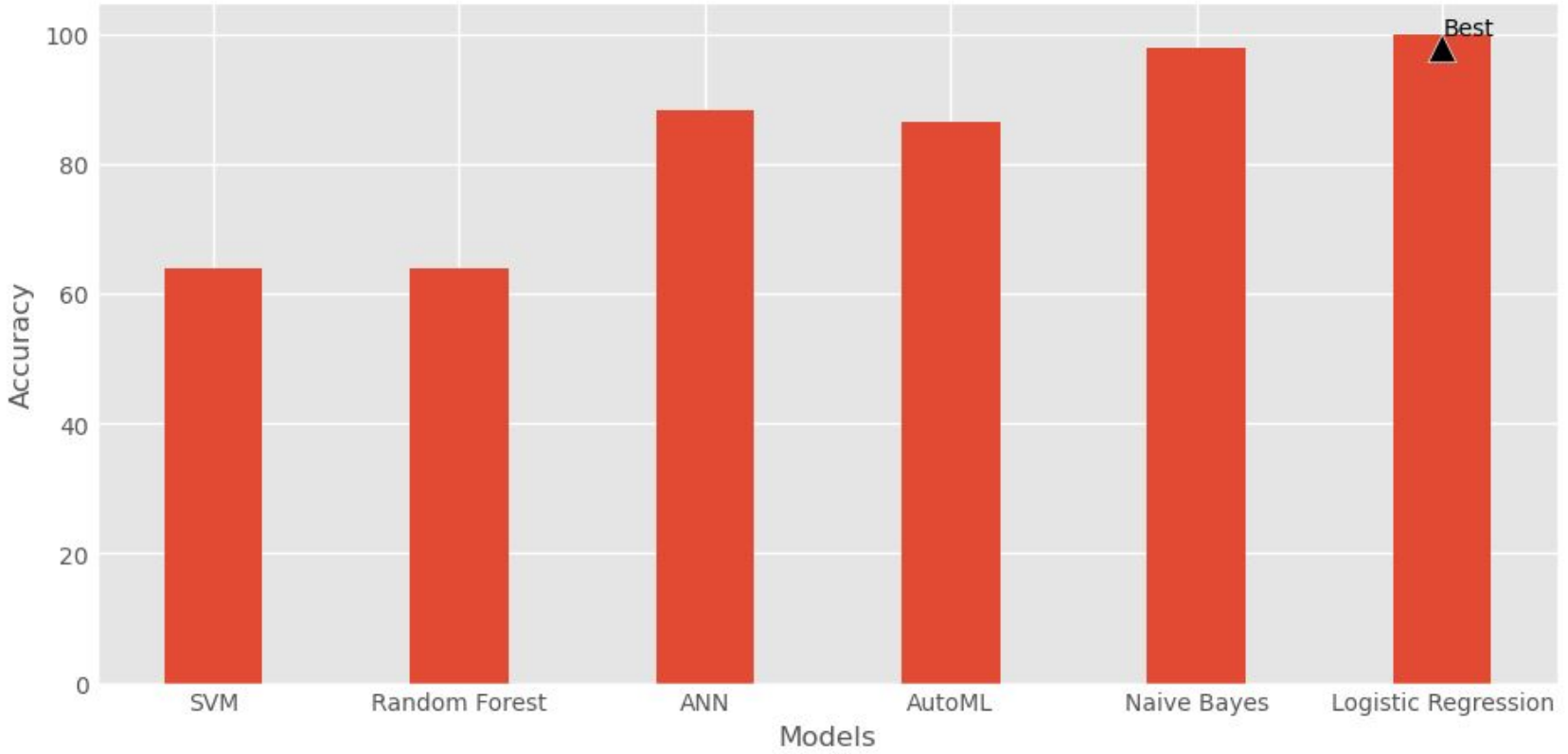
- We created a flask server to run in the local area network.
- Flask is a micro web framework written in Python.
- Created a route to get data from requests and use the data to predict the result.
- As the model was trained on 300 samples we need to provide at least 300 samples to get accurate results. The result will be either "Healthy" or "Broken".



Results

Sr. No	Algorithm	Preprocessing	Precision	Recall	Accuracy
1	SVM	—---	0.57	0.57	0.58
2	Random Forest	—---	0.56	0.85	0.60
3	ANN	Basic	0.87	0.90	0.88
4	AutoML	Basic	0.88	0.84	0.86
5	LogisticRegression	Advanced	1	1	1
6.	Random Forest	Advanced	0.98	0.99	0.98
7	SVM	Advanced	1	1	1

Bar graph comparing results of different algorithms



Thank You