Report for Data Challenge

Atrayee Neog

January 2020

1 Introduction

The use of Machine Learning for Predictive Maintenance Analysis using Vibration data was not something I had worked on earlier and hence the first step was to read the literature provided for the task. The references provided helped me get a hold of the problem domain and understand the main objectives of the Challenge. Along with the literature review, having gone through the AiSight website helped understand the Company's field of work. In the project, I have tried to meet the objectives of the Data Challenge by classifying the data into different Machine States by applying two Machine Learning algorithms.

2 Resources

The Dataset that I have chosen was the project_fan.csv with 900 sample points. *Python* was the language of preference and *Google Colab* was used as the platform to code. The Data given for the challenge was structured but unlabelled. The description mentioned that the data belonged to five classes namely: Unix time, Amount of samples, Time period (milliseconds), Sampling Rate, Sensor data (millivolt reads). An understanding of these features were required to understand which data would be most useful for the Challenge.The libraries used were:

- Tensorflow 2.1.0-rc1
- Keras
- Numpy
- Pandas
- ScikitLearn
- Matplotlib
- Seaborn

3 Training

3.1 Data Preprocessing

The data was loaded using Pandas Dataframe and summarised. Based on the information given, the data was divided into the five classes 'Unix_time','Sample_Size','Time_Period', 'Sampling_Rate' and 'Sensor_Data'. As the feature Sensor Data was in the form a string, it needed to be converted to float datatype to be used. For this, each datapoint in the Sensor Data column was converted into a list and then split into 16539 pandas columns containing 16539 float values for each Unix timestamp with an average interval of 340 milliseconds. For feature extraction, the Sensor Data column seemed most relevant for the analysis and hence this was

extracted into another dataset '*train*'. The shape of the train data hence was 900*16539. The missing Nan values in the dataset were then filled using the 'pad' method which propagates the last valid observation forward to the next valid missing value place. I used different methods for the *fillna* method, but *pad* or *ffill* worked the best. The data was then normalised using the scikitlearn package *StandardScalar* to remove any distortions in the range of the dataset.

3.2 Unsupervised Learning

The task being a classification task,the dataset needed to have definite labels. Since the dataset was unlabelled, *Unsupervised Learning* needed to be implemented to find clusters of data most similar to each other in their frequency distributions (Sensor Data) to distinguish different Machine States based on the vibrations. Intially, the join operation from pandas DataFrames was used to append the 16539 target labels into the 'train' as well as 'data' dataset. Kmeans clustering algorithm was then used to find the relevant clusters using an *auto* algorithm which chooses 'EM algorithm' for sparse data and *elkan* for dense data. In order to determine the optimal number of clusters, the *Silhouette Score* was used which optimally calculated the best value to be 17(approx. 52). The Score reduces further for k=21, but I have taken k=17 to reduce any chances for overfitting.

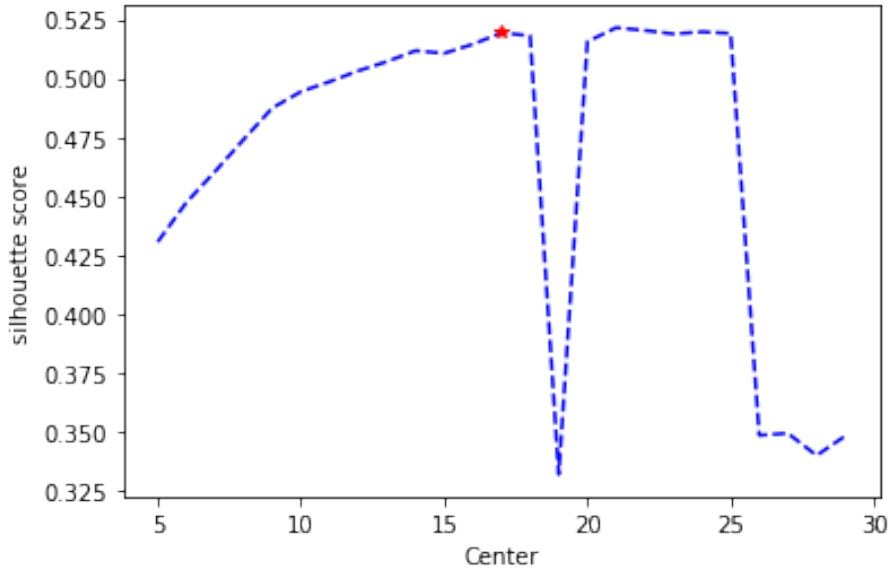

Figure 1: Silhouette Score with k=17 having the highest value

The prediction for the training model was done using the kmeans algorithm with k=17 as it shows the highest Silhouette Score as can be seen from Figure 1. A new column with the Target label (cluster labels) was saved into the *train* and *data* datasets for further classification and saved to disk as: *DataFinal.csv* and *TrainDataFinal.csv*.

3.3 Classification

3.3.1 Deep Learning:

Now that we have our labels, a *Deep Neural Network Model* was built using Tensorflow package to classify the dataset into the various labelled Machine States. The dataset was split into training and testing data with a 80:20 ratio. A *Sequential Model* was built using an input layer (with 16539 neurons as per the 16359 sensor data), a dense hidden layer (with 512 neurons), one dropout layer (20% dropout), another dense

layer (with 128 neurons) and finally an output layer (with 17 neurons as per the number of target clusters). The activation method used was the *relu* function for the hidden layers and a *softmax* activation function for the output layer. *Adam Optimizer* was used for optimization and loss was calculated using *Categorical Cross-Entropy*. *EarlyStopping* callback methods was used by monitoring validation with patience=10 to avoid overfitting. In order to train the data, the data was split into training and validation set (20%) during the traing and the data labels were mapped into *one hot encoded matrix* to avoid any bias towards any particular label. The training was done using 128 batch size and 200 epochs.Due to the callback, the training was stopped at 14 Epochs. The predicted values were then compared to the test labels and the evaluation was done based on this.

3.3.2 Evaluation

Accuracy and Loss measures are used for evaluating the model. The two measures are plotted to compare between the training and the validation datasets. The accuracy values for the validation is slightly lower than the training dataset. This could suggest a lack of sufficient sample size, leading to a overfitting in the model. The prediction accuracy is also calculated which comes out to be approximately 94%. A confusion matrix was also built using the scikitlearn package 'metrics' which describes the various evaluation measures for each and every Machine State. The evaluation measures suggest that the model is quite proficient in predicting the Machine States based on the frequency of the vibrations from the Sensors.

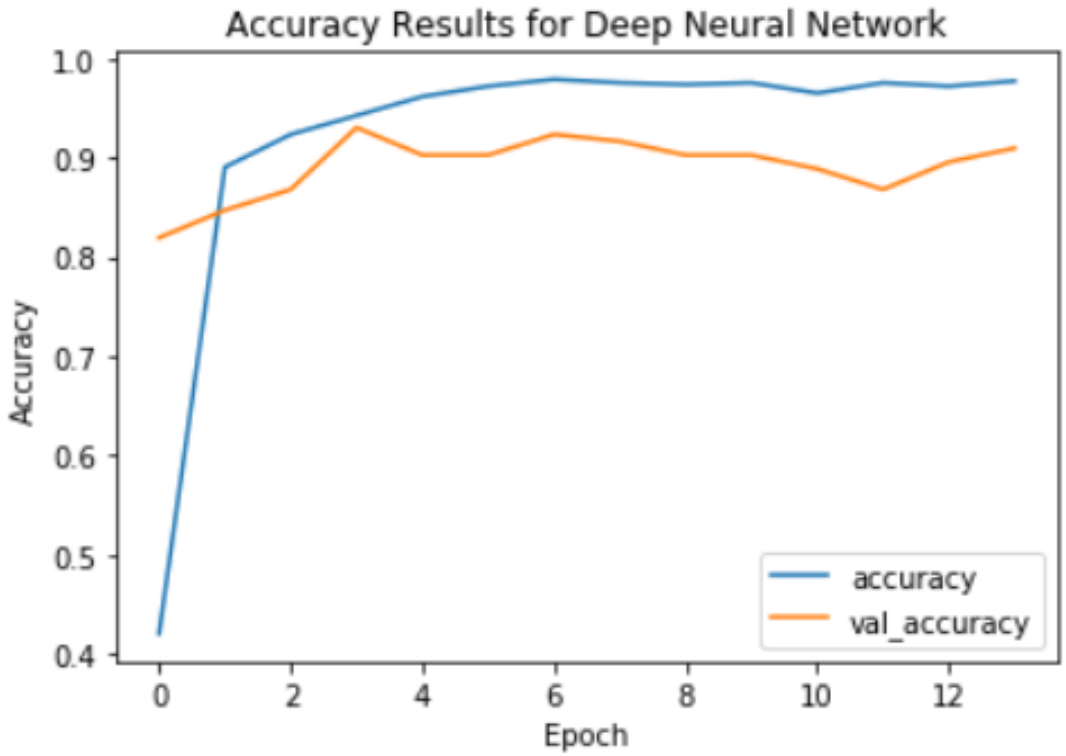

Figure 2: Accuracy Results for Deep Neural Network

3.3.3 Random Forest

Even though the Deep Learning model shows a prediction accuracy of 94%, the ambiguity in the training an validation accuracy and loss rates leads me to apply another algorithm, namely, Random Forest Classifier to try to make the prediction results better. The Random Forest starts with 100 decision trees and calculates the predicted values for the dataset based on a majority voting scheme.

3.3.4 Evaluation

Accuracy measured for the prediction increases by 2% from the Deep Learning model to 96%. A heatmap was plotted which showed the Confusion Matrix to be quite similar to that for the Deep Learning with maximum entities predicted for Machine State 1. A validation curve was plotted for the Random Forest model, to show the difference between the Training and the Cross-Validation scores for decision trees starting from 1 to 100, with the validation accuracy being slightly lower than the training accuracy, again leaving a scope for overfitting of the model due to lesser samples.

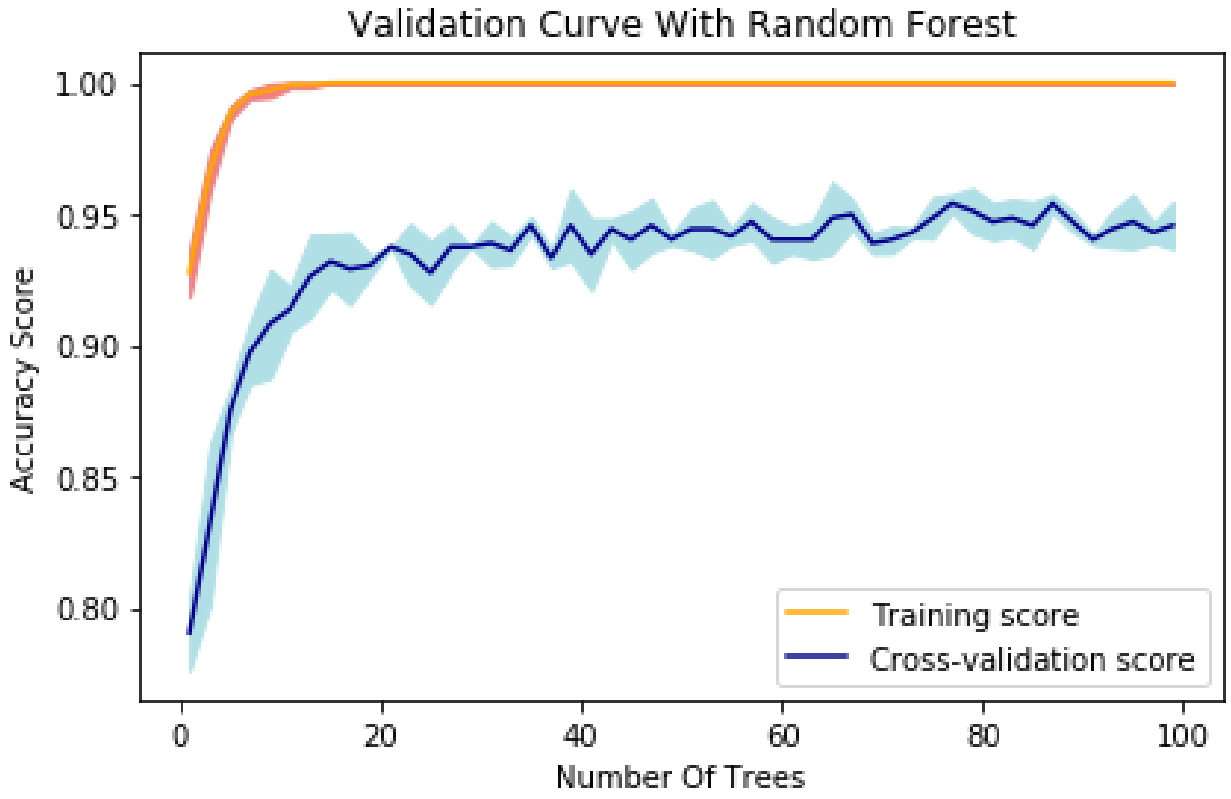

Figure 3: Validation Curve with Random Forest

4 Results

Below are the histogram plots for the prediction results for both the models which shows maximum number of items in Machine State 1. However, the distribution of items in the other Machine States is the same in the Random Forest Model and the Deep Neural Network Model.

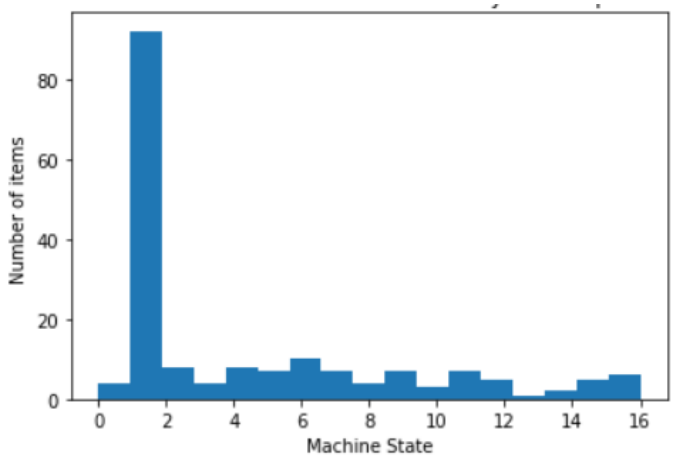

Figure 4: Histogram Results for Deep Neural Network Model

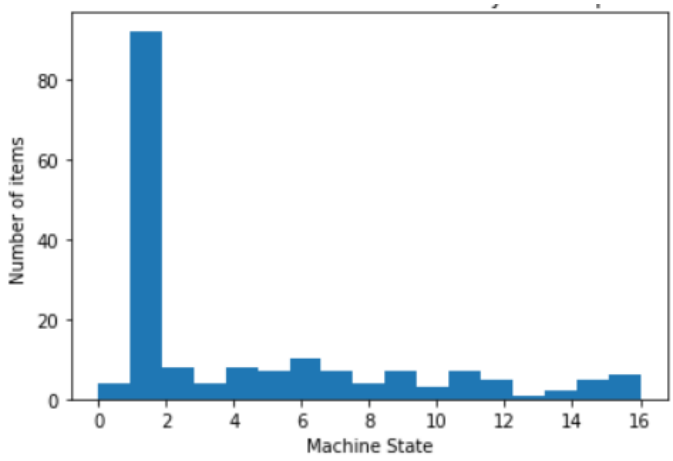

Figure 5: Histogram Results for Random Forest Classifier Model

	precision	recall	f1-score	support
Machine_state 1	1.00	1.00	1.00	4
Machine_state 2	1.00	0.98	0.99	94
Machine_state 3	0.88	1.00	0.93	7
Machine_state 4	0.75	1.00	0.86	3
Machine_state 5	0.75	1.00	0.86	6
Machine_state 6	1.00	1.00	1.00	7
Machine_state 7	0.70	0.88	0.78	8
Machine_state 8	0.86	0.86	0.86	7
Machine_state 9	0.75	0.60	0.67	5
Machine_state 10	0.86	0.86	0.86	7
Machine_state 11	1.00	1.00	1.00	3
Machine_state 12	1.00	0.88	0.93	8
Machine_state 13	0.60	0.75	0.67	4
Machine_state 14	1.00	0.50	0.67	2
Machine_state 15	1.00	1.00	1.00	2
Machine_state 16	1.00	0.62	0.77	8
Machine_state 17	0.83	1.00	0.91	5
accuracy			0.93	180
macro avg	0.88	0.88	0.87	180
weighted avg	0.94	0.93	0.93	180

Figure 6: Classification Report for Deep Neural Network Model

	precision	recall	f1-score	support
Machine_state 1	1.00	1.00	1.00	4
Machine_state 2	1.00	0.99	0.99	94
Machine_state 3	1.00	1.00	1.00	7
Machine_state 4	0.75	1.00	0.86	3
Machine_state 5	0.75	1.00	0.86	6
Machine_state 6	1.00	1.00	1.00	7
Machine_state 7	0.78	0.88	0.82	8
Machine_state 8	1.00	0.86	0.92	7
Machine_state 9	1.00	0.60	0.75	5
Machine_state 10	0.88	1.00	0.93	7
Machine_state 11	1.00	1.00	1.00	3
Machine_state 12	1.00	0.88	0.93	8
Machine_state 13	1.00	1.00	1.00	4
Machine_state 14	1.00	1.00	1.00	2
Machine_state 15	1.00	1.00	1.00	2
Machine_state 16	1.00	0.88	0.93	8
Machine_state 17	0.83	1.00	0.91	5
accuracy			0.96	180
macro avg	0.94	0.95	0.94	180
weighted avg	0.97	0.96	0.96	180

Figure 7: Classification Report for Random Forest Classifier Model

As can be seen from the Classification reports, The Accuracy for the Random Forest Classifier is 3% more than the Deep Neural Network, because for the DNN, only one model is rectified and evaluated again and again and hence it needs sufficient amount of data to overcome any bias and for correct classification. But in case of Random Forest, since comparisons are made between hundreds of decision trees, and classification is done based on majority voting, it is easier to overcome these obstacles, with lesser data samples.