Leverage Unlabelled data to improve Supervised Model performance

Project Background

Business Overview

Uptake provides SaaS (Software as a Service) platform using industrial AI to deliver fast financial outcomes that impact their client's business. This includes lowering client's Operation & Maintenance costs, improving their asset reliability, and providing new findings and efficiencies from client's operational data. Uptake deals with machine-level time series data such as sensor data, maintenance records, mechanical data and contextual data for industrial applications.

Problem Statement

In industrial applications, unplanned equipment downtime and service interruptions can have a large financial impact on business performance. Supervised machine learning is an effective tool in predicting and preventing these failure instances.

There are two types of data that are primarily used for this task:

- sensor and performance data that represents the normal operation of the machinery.
- maintenance records that cover how and when failure occurs.

With full access to well-maintained dataset, effective models can be built to predict these failure instances. However, such datasets are not the norm. Companies are often unwilling to release all their operational data for analysis. Maintenance records are often poorly maintained. Additionally, failure instances are rare which creates a large class imbalance and makes building models problematic.

Problem Objective

Unlabelled signal data was leveraged by using it to train Autoencoders in order to generate estimated features and use them to improve the performance of a supervised model trained with labels.

Data Source

The data being used by us in project is openly available at Backblaze . Backblaze data center daily collects snapshots of each operational drive.

Data Description

Uptake deals with two kinds of time-series data when it comes to prediction models:

- **Signal Data**
  It is time stamped data from sensors, used as inputs or features for failure models. In our case, signal data is overall data collected over 6 years from 2013 to 2018.
- **Maintenance Data**
  These are the records when machine malfunctioned in the past. Currently, the failure column is interpreted as maintenance records available for labelling in Supervised ML classifiers.

Data Size

Approximately 36 GB of storage capacity, our dataset therefore contains around 165 million records with each year divided into four quarters. Each quarter holds data for 3 months time period.

Data Type

These snapshots are then processed and stored in stats file in form of CSVs. Each day CSV file contains 85-129 columns and 40,000 to 1,00,000 records which indicates the stats related to operational drives that day.

The label data for this dataset is called Failure. It's represented as a 0 or 1, where 1 indicates the last day a drive was operational before failure. There are 4 other identification fields (data, serial number, model and capacity in bytes).

The remaining fields represent SMART stats data. SMART in this instance stands for self-monitoring, analysis and reporting technology. They are a standard set of data points collected for hard drive monitoring. For each SMART stat there is a raw and normalized field.

Problem and Data Pipeline

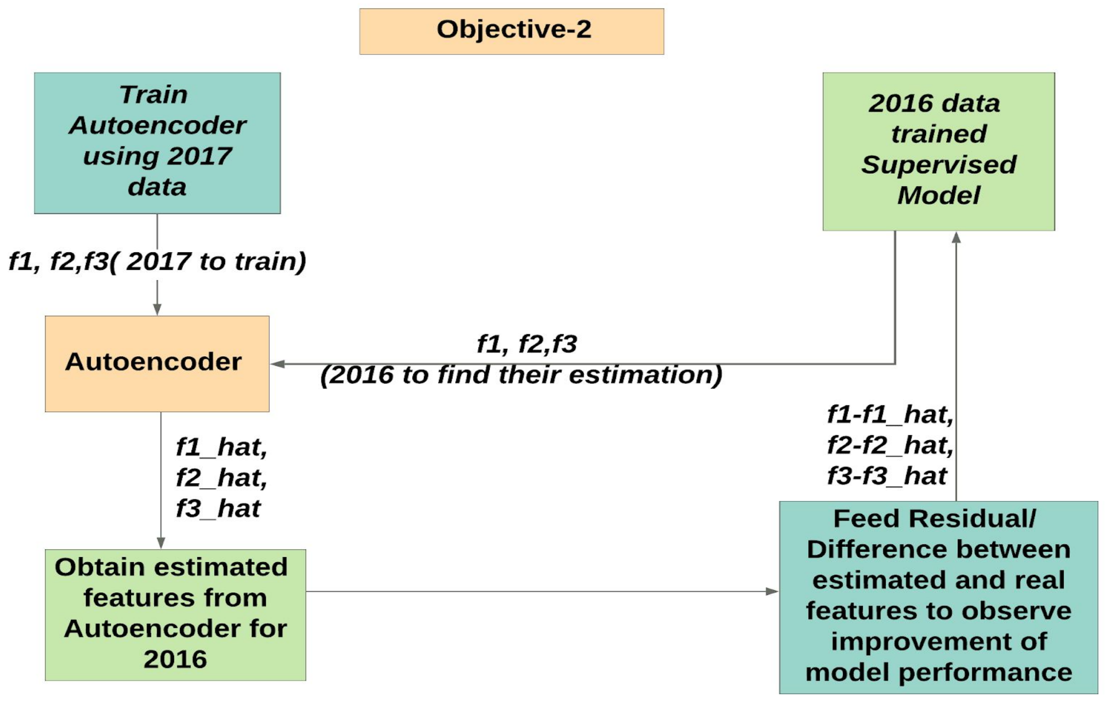
The problem was that signal data is available of equipment but maintenance data is not. Hence we need to leverage the left unlabeled signal data to improve performance of supervised model classifier predicting failure of equipment.

In our project, maintenance data is Failure column in drive data, and that is assumed to be limited for this objective. First Autoencoder is trained to over normal behaviour of operational drive data by dropping the rows with label of Failure (label 1). This data which only represents normal behaviour of drive is used to train Autoencoder.

Then Supervised Classifier (Random Forest) is trained over features on other dataset to predict failure of drives. Actual features used in Supervised Classifier is then fed to trained Autoencoder to estimate features, calculate residuals by subtracting actual features and estimated features.

Now, new dataset which contains actual features along with all the residuals is fed to Supervised Classifier and changes in its performance is observed.

The procedure used is explained visually in below lucid chart -

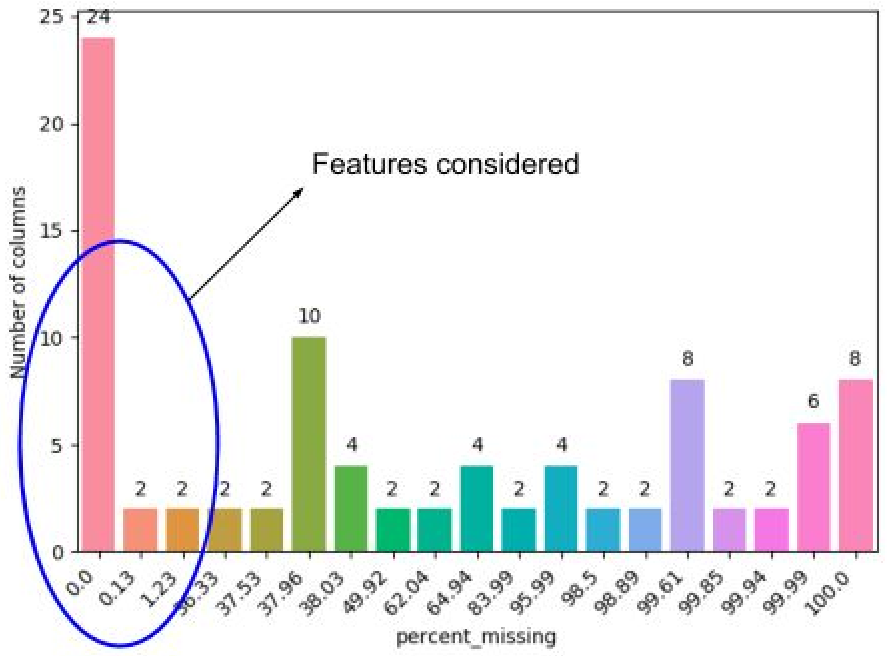

Objective-2
Train Autoencoder using 2017 data
f1, f2,f3(2017 to train)
Autoencoder
f1, f2,f3
(2016 to find their estimation)
2016 data trained Supervised Model
f1_hat,
f2_hat,
f3_hat
f1-f1_hat,
f2-f2_hat,
f3-f3_hat
Obtain estimated features from Autoencoder for 2016
Feed Residual/ Difference between estimated and real features to observe improvement of model performance

Project Execution

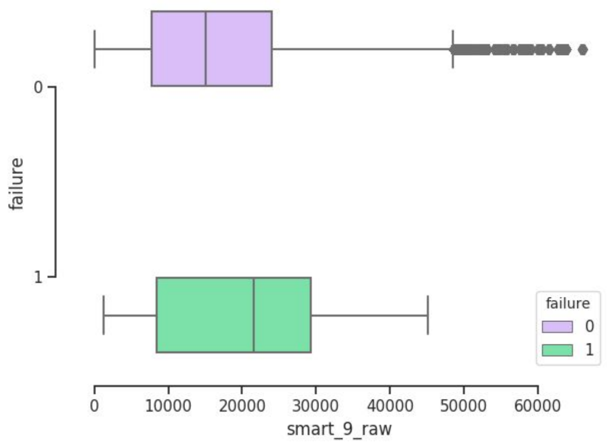
Step 1: EDA for year 2017 data

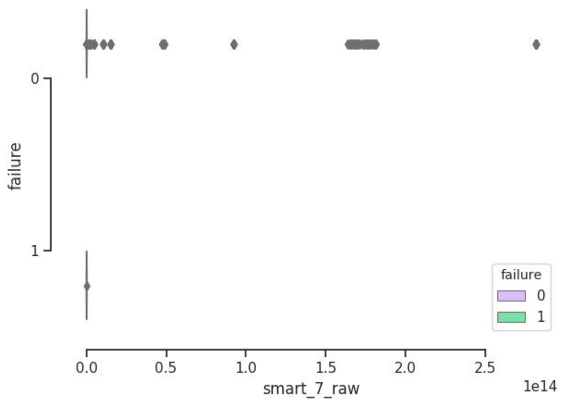
Tasks Completed

The data is highly imbalanced, for the first month of 2017 there are 1989462 - 1 million 989 thousand and 462 records, with 73,885 drives data, failure cases are just 101.
Then percentage of emptiness of all the columns with each smart stat represented with two columns of raw and normalised, were plotted as shown in below bar plot indicating 14 smart stats (28/2) with a threshold of less than 30% emptiness .

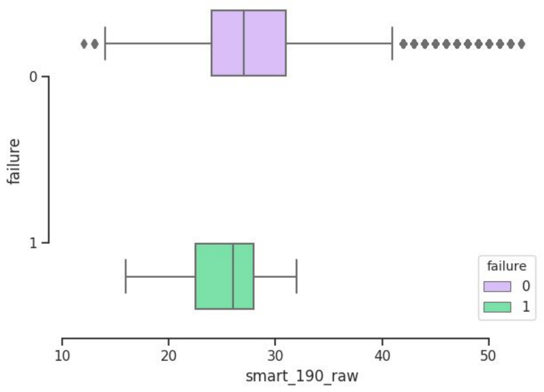


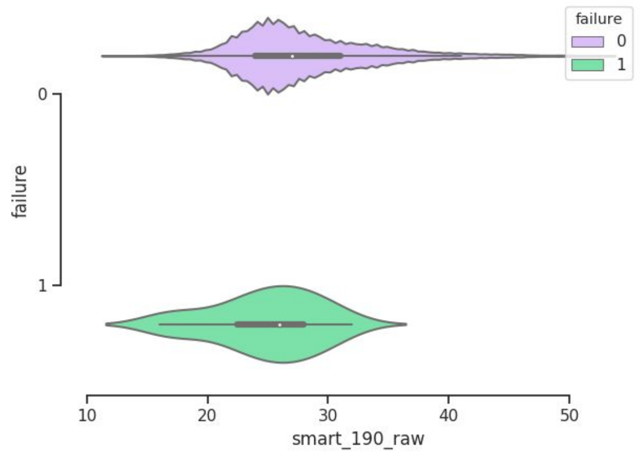
Further EDA was carried by plotting whisker plots representing medians and ranges for Smart Stats. As we can see in below two whisker plots, some features showed different variations, ranges and median values for failure and non-failure data. While for many features such as Smart Stat 7, Smart Stat 188, Smart Stat 189, Smart 191, etc. no variations were found indicating only constant values or outliers for these features.



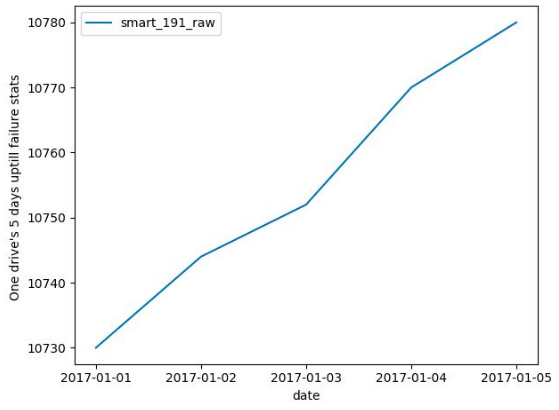


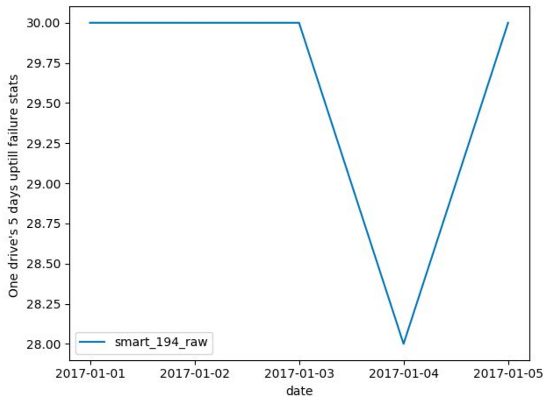
There were few features such as Smart Stat 190 showing overlapping variations in feature values for failure and non-failure for which violin plots were also used in order to find out the data distribution and density estimation over the feature values. Through the violin plot we can observe that data distribution for failure has lower values of Smart Stat 190 while higher for operational drives.

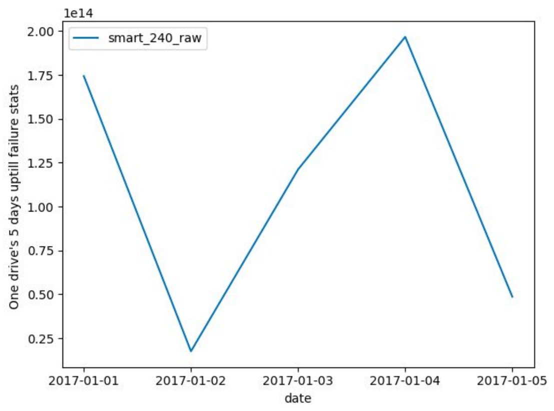


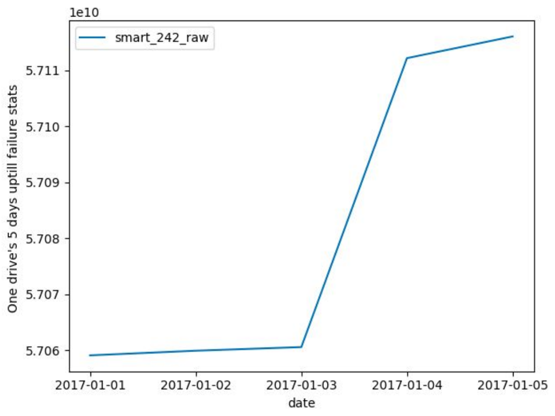


Then, analysis of smart stat values for a fixed time frame before failure happened was done using 5 days prior data for one particular drive which failed on 5th January, 2017 and observing its values of various features for the first 5 days of January.



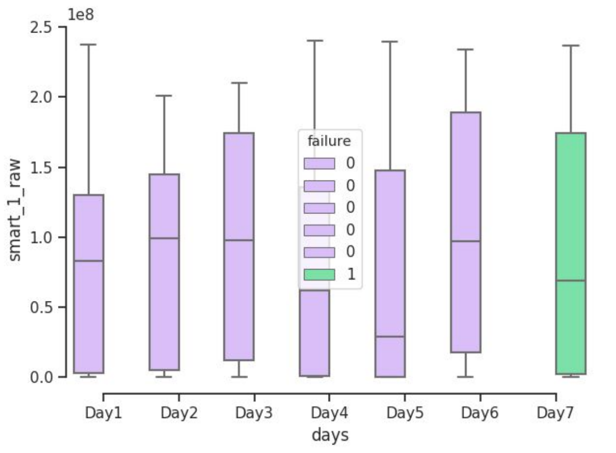


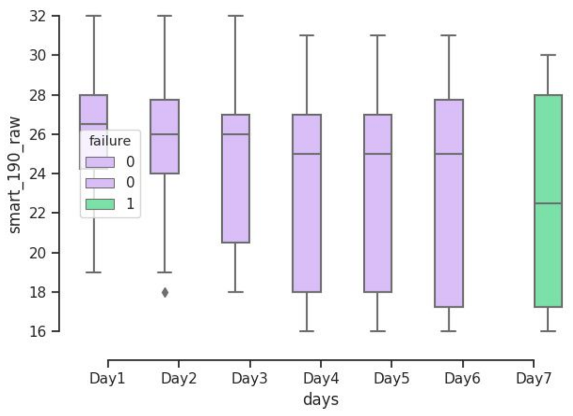


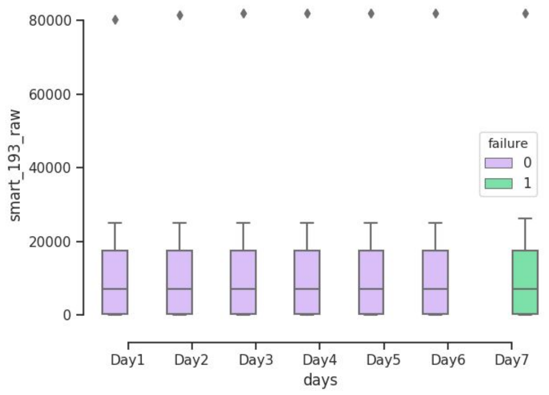


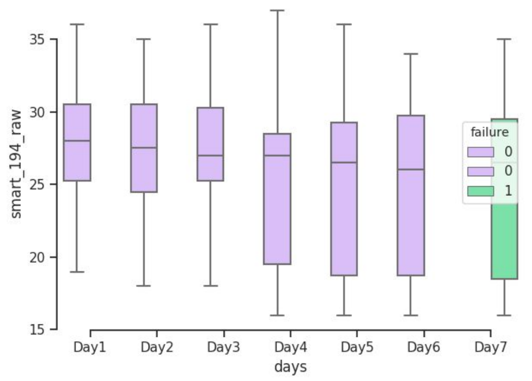
We can notice in the above plots that values of these features changed before drives failure, for some features the value was continuously rising or decreasing before failure, for some features it remained constant indicating not helpful in failure prediction. For many features the slope change was quite significant in plots such as for Smart Stat 242 in above figures.

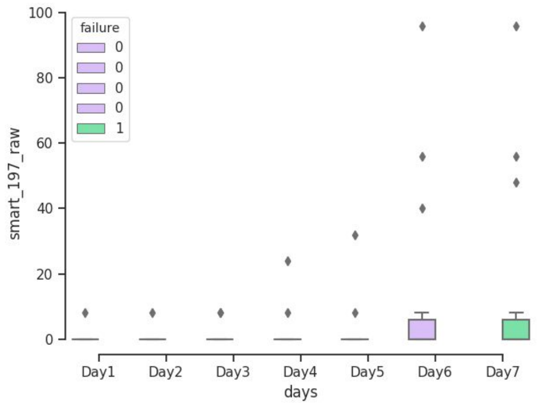
Lastly group of drives were plotted by using data of 14 drives together for 7 days in order to observe changes in Smart stats values and changes before their failure on 7th day.

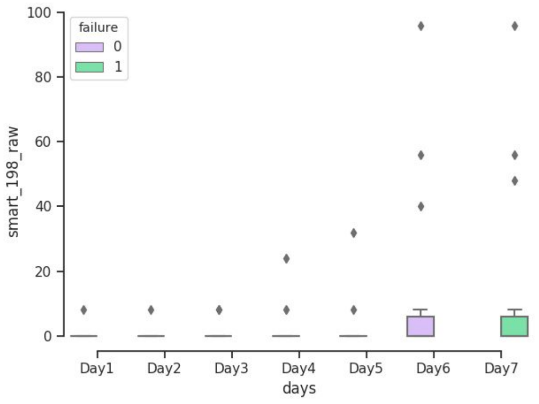












Few features such as Smart Stat 194, 197, 198 proved to be relevant by showing that median value increases/decreases every day before failure. Some features were discarded as in above plots they show no change in entire range of values for complete time frame of 7 days. While tracking behaviour for a group of drives before failure in the above plot of Smart Stat 1 and 194, we can notice that the average of the first 3 days are different than the average values of 3 days before failure which indicates that back labelling of data with 3 days as label 1 is a good choice.

Results

Summarizing the results from all the plots of EDA, it was concluded that below 7 features[1] are relevant in order to carry out further steps.
Smart 1 Read Error Rate
Smart 3 Spin Up Time
Smart 9 Power On Hours
Smart 192 Unsafe Shutdown Count
Smart 194 Temperature or Temperature Celsius
Smart 197 Current Pending Sector Count
Smart 198 Uncorrectable Sector Count

Availability of Deliverables

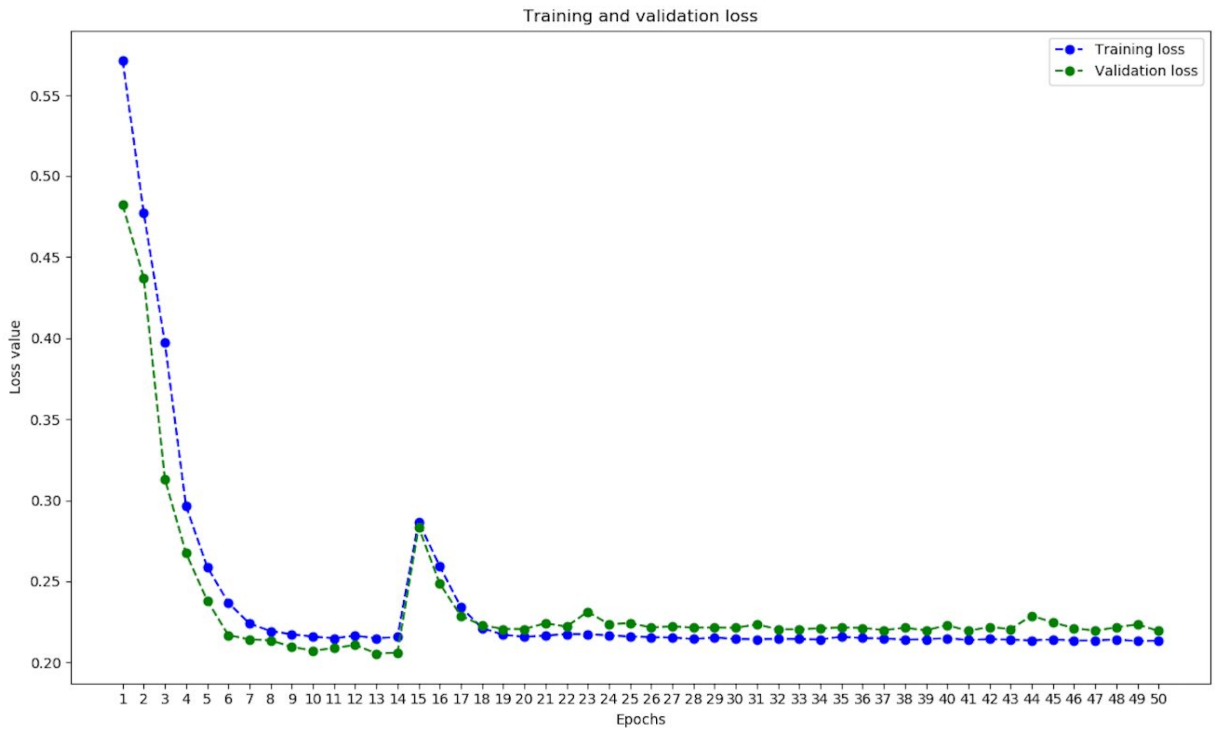
All the plots related to EDA of 2017 data can be found at below github link -
https://github.com/Shikhas/iit-uptake-capstone/blob/master/failure_modelling/reports/reports_eda_2017.pdf

Step 2: Training Autoencoders

Tasks Completed

Autoencoder was implemented with two layers of encoder, one hidden layer and two layers of decoder. Now using the above 7 features and back labelled data with 3 days before failure was used (around 1,40,000 data points from the first month of 2017) to train and tune hyperparameters of Autoencoder for normal behaviour of drives. But, the Autoencoder's validation data was producing lower loss value than training data's loss value, due to which it was decided to retrain Autoencoder with cross-validation.
The average loss for training data is - 0.3164

The average loss for validation data is - 0.3128



Results

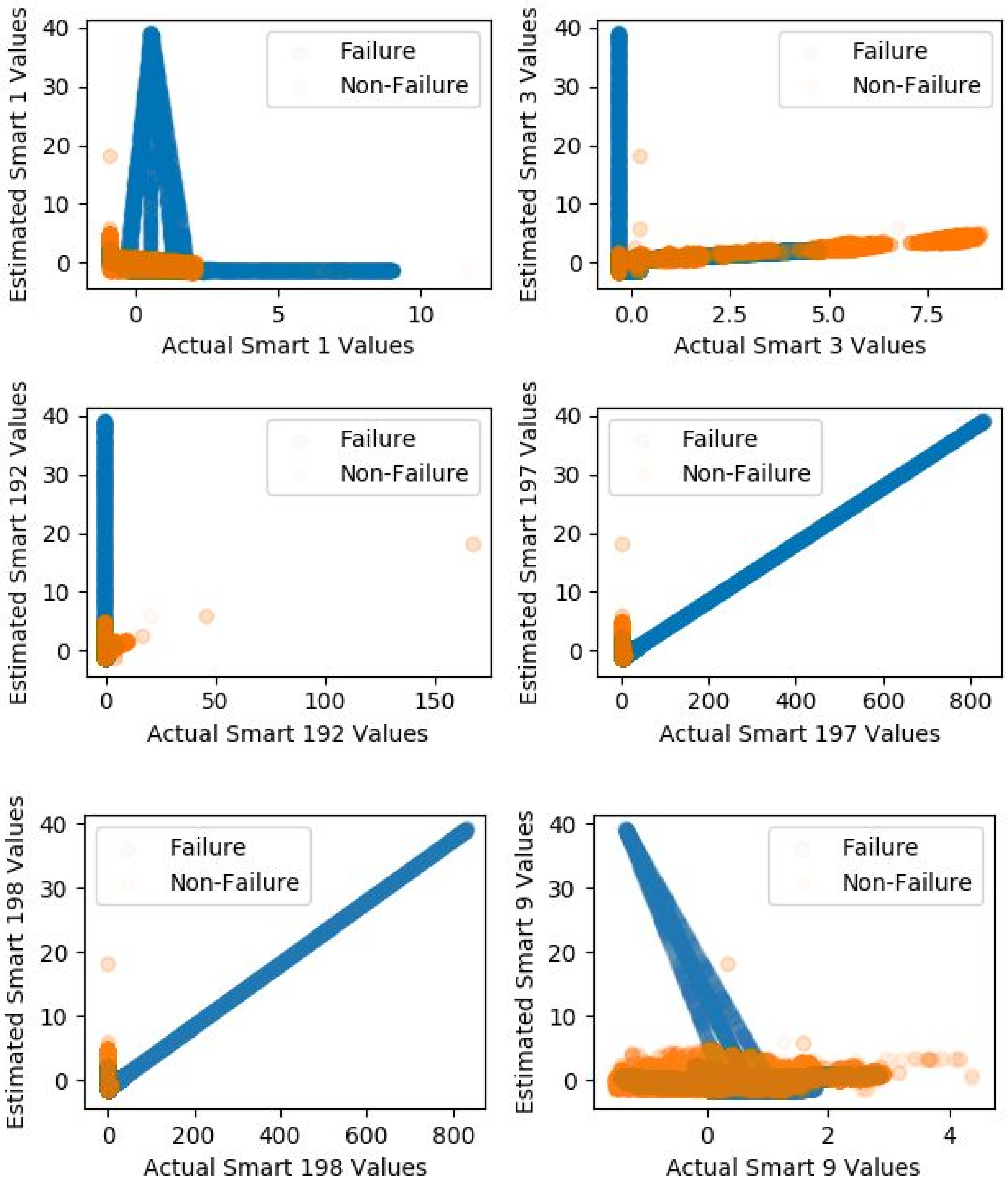
The Autoencoder was successfully trained and saved in file along with its weights so that it can be used to estimate or predict features without retraining again to recognize normal behaviour of drives. The validation data and training data loss value decreases as epoch value increases Moreover, as shown by spike in the above plot, it can be considered to use lower learning rate for future work.

Step 3: Exploring estimated features from Autoencoders

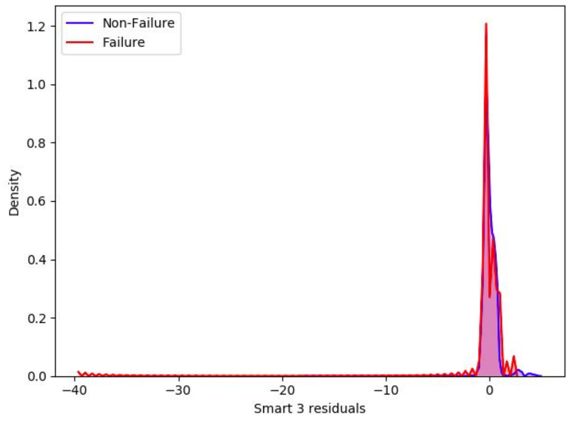
Tasks Completed

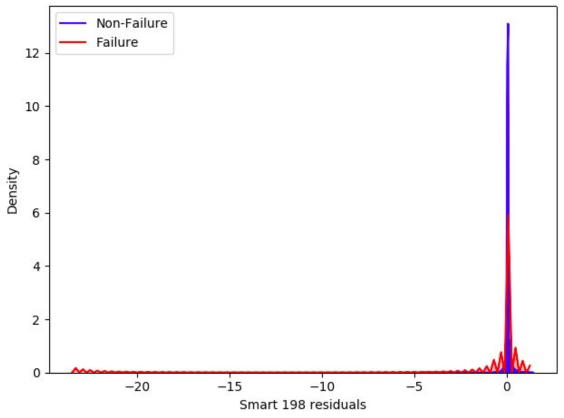
For exploring whether Autoencoders is recognizing abnormalities for failure data points three kinds of plots were used.

1. **Estimated versus Actual features** plots were explored by creating scatter plot to establish how each relevant features for failure drives and non failure drives differ in estimated values by autoencoders from actual feature values. Incase not clear in figures below orange represents non-failure data points and blue failure ones.

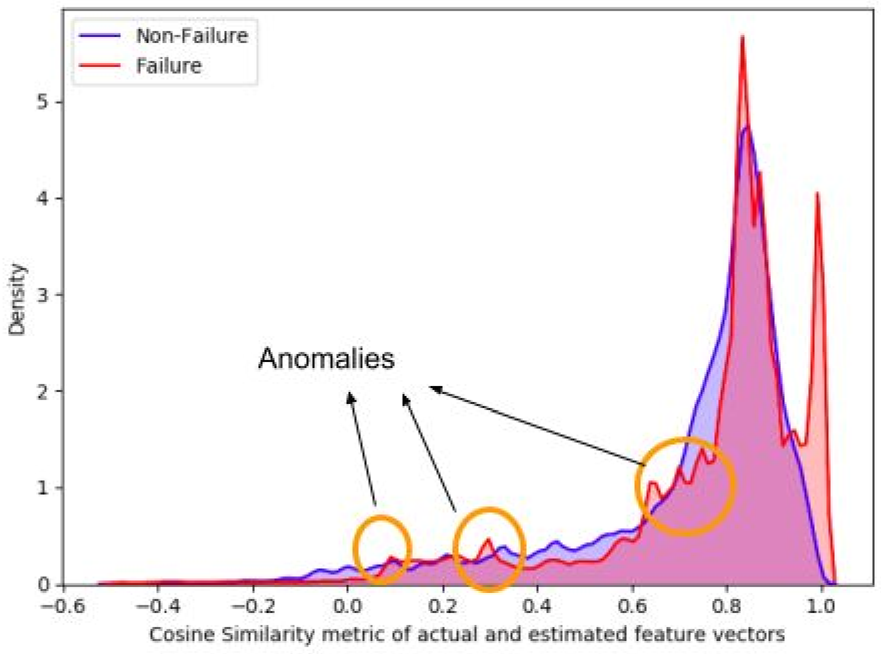


2. **Residual Plot** was explored by plotting density curve for residuals of failure and non failure drive data points of each feature. In below residual plots we can see there are spikes for failure data points in tail which represent abnormal feature value indicating different values of residuals than the one in blue curve.





3. **Cosine Similarity plot** was built treating estimated features and actual features as vectors and then calculating cosine similarity metric between those vectors. In below plot we can notice than many failure values have cosine similarity and spikes in negative scale as for failure drives, estimated and actual features must have less similarity hence cosine angle with angle 36 degree and more.



Results

We can see spikes in residual plots and cosine similarities where anomalies are recognized by Autoencoders.

Availability of Deliverables

All plots of autoencoders explained in Step 2 and 3 in detail are available at the below git link- https://github.com/Shikhas/iit-uptake-capstone/blob/master/failure_modelling/reports/final_plots_for_autoencoders.pdf

Step 4: Using residuals to improve Supervised Model Performance

Tasks Completed

Random Forest performance was observed without residuals with SMOTE oversampled 1million and 400,000 data points and its performance changes were noted after feeding residuals (estimated features - actual features) to Random Forest classifier as shown in below two confusion matrix. Important thing to note is that the classifiers with and without residuals is each tuned separately for hyperparameters as features fed to them are different and also it supports to ignore any noise.

Random Forest without residuals

Actual **Predicted**	**Class-0/ Negative**	**Class-1/ Positive (Failure)**	**Precision :-** TP/(TP+FP) **0.9079** **Recall:-** TP/(TP+FN) **0.8554**
Class-0/ Negative	134184 (TN)	21291 (FN / Missed Alarms)	
Class-1/ Positive (Failure)	12772 (FP/ False Alarms)	125967 (TP)	

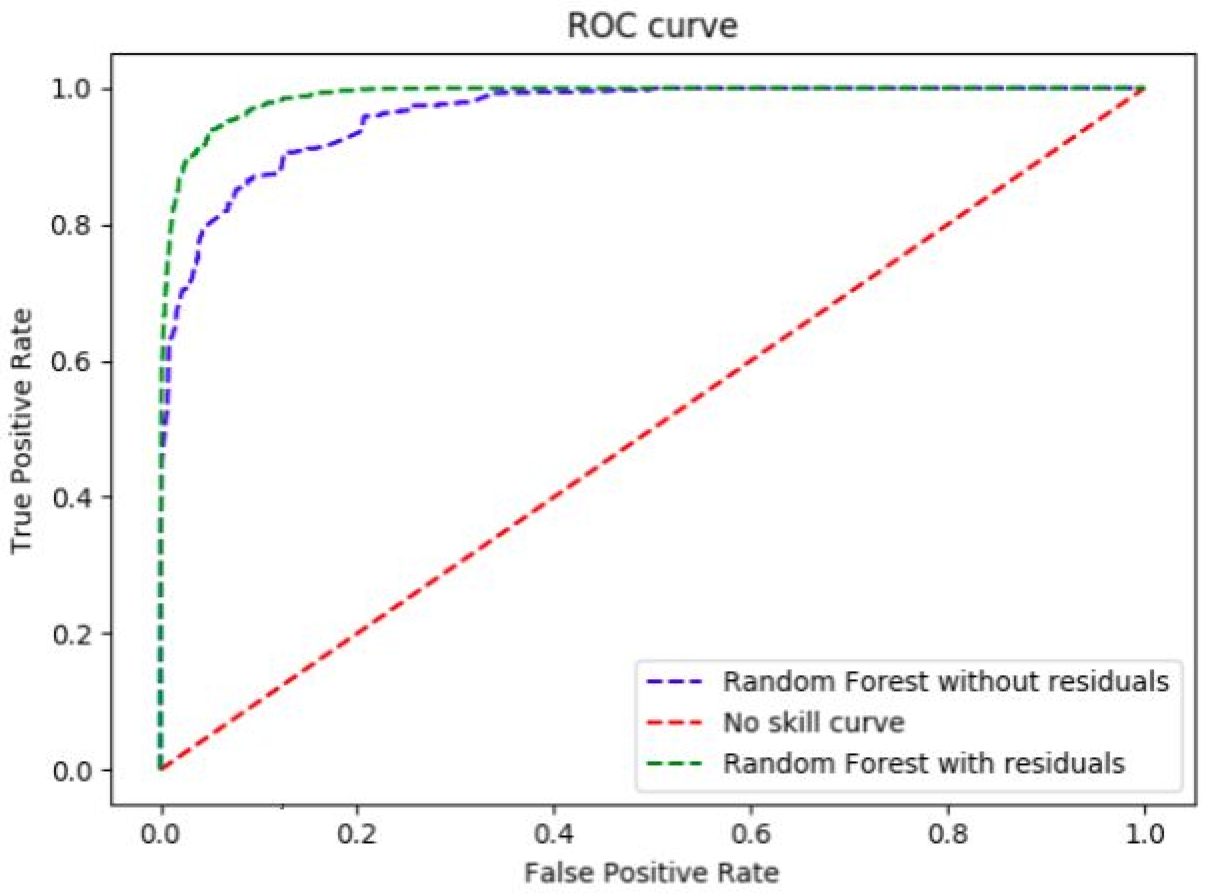
Random Forest with residuals

Actual **Predicted**	**Class-0/ Negative**	**Class-1/ Positive (Failure)**	**Precision** :- TP/(TP+FP) **0.9464** **Recall:-** TP/(TP+FN) **0.9382**
Class-0/ Negative	139134 (TN)	9097(FN / Missed Alarms)	
Class-1/ Positive (Failure)	7822 (FP/ False Alarms)	138161 (TP)	

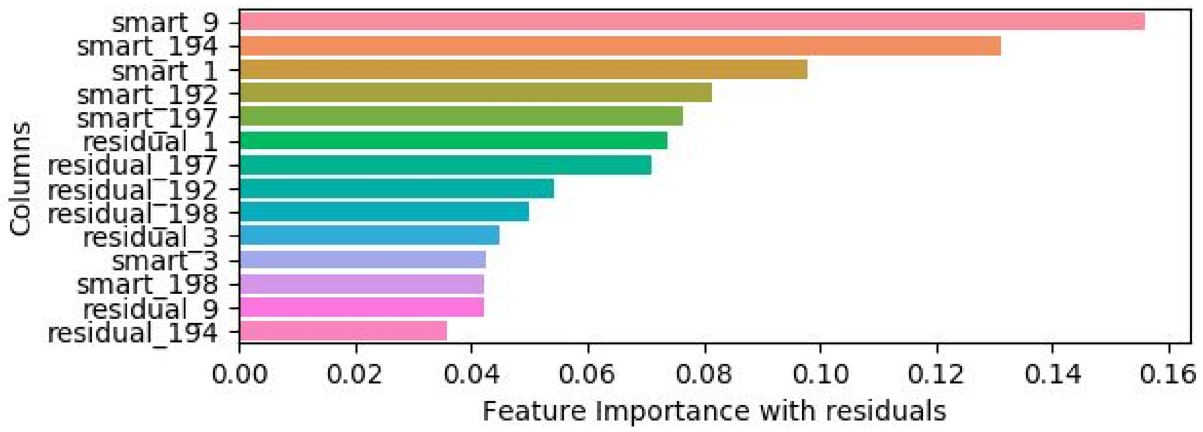
From these two confusion matrices we can notice that after feeding residuals to our classifier the Precision, Recall [2]and True Positive counts have increased and False Negative (missed alarms) which are very important to be less to save clients from monetary loss has also decreased.

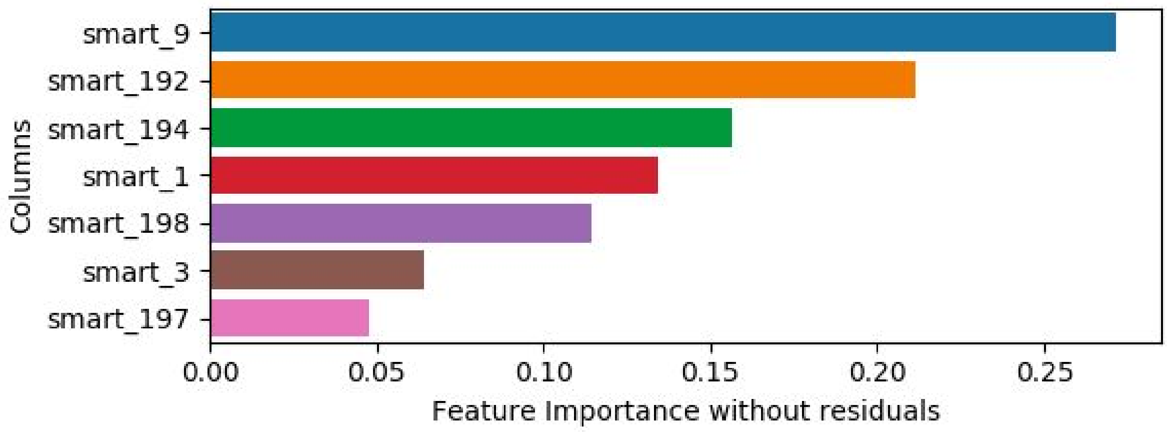
Even though Precision and Recall have been mentioned in above confusion matrix, however we need to note that there is an important distinction between our pointwise and eventwise values. The pointwise values are simply the ones generated from the standard confusion matrix as shown above. These values didn’t capture exactly what is happening in our models. Because, the way we are approaching our time series data, we ended up adding or removing labeled points from our dataset. To account for this we needed to compare our predicted labels from our model to our data original labels. If our model flagged a failure within the given time window before a failure in our original data then we consider that failure successfully predicted. Summing all the failures correctly predicted in this way gives us Event-wise Accuracy.

Below plot contains ROC curves of Random Forest before and after feeding residuals which also shows that for ROC curve with residuals in green has True Positive Rate is at higher than ROC curve without residuals in blue at lower value of False Positive rate.



Moreover, feature importance was taken out for Random Forest with and without residuals. We can notice than for feature importance few of residuals have even surpassed actual features for example residual 3 and residual 198, and others are also proving to be significant such as residual corresponding to features Smart Stat 1, 197 and 198 for classification with improved results.





Results

Random Forest performance was improved with residuals as recall and precision is increased with model recognizing True alarms better and ignoring false ones. Moreover, the false negatives (missed alarms) were decreased which was very important as missed alarms will lead to monetary loss of client costing due to failure without an alarm.

Availability of Deliverables

Github link for overall code of autoencoder part -
https://github.com/Shikhas/iit-uptake-capstone/tree/master/failure_modelling

Step 5: Possible Improvements for Autoencoders

For future work, all the plots shown above and results related to trained Autoencoders have room for improvement by using Autoencoders with more layers and even trying with LSTM Autoencoders [3,4,5] and including different features as well as experimenting with different year data. Need to observe event-wise accuracy improvements which would be even better.

References

1. "S.M.A.R.T. - Wikipedia." https://en.wikipedia.org/wiki/S.M.A.R.T.. Accessed 10 Aug. 2019.
2. "How and When to Use ROC Curves and Precision-Recall Curves for ...." 31 Aug. 2018, https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/. Accessed 10 Aug. 2019.
3. "Engineering Extreme Event Forecasting at ...." 9 Jun. 2017, https://eng.uber.com/neural-networks/. Accessed 10 Aug. 2019.
4. "Deep and Confident Prediction for Time Series at Uber." 6 Sep. 2017, https://arxiv.org/abs/1709.01907. Accessed 10 Aug. 2019.
5. "Dataset: Rare Event Classification in Multivariate Time Series." 27 Sep. 2018, https://arxiv.org/abs/1809.10717. Accessed 10 Aug. 2019.