Project Report
Predictive Maintenance For TurboFan Engine
Chandra Prakash Saini(160010027)

Introduction

In this project we are applying Machine learning /Data science techniques to turbofan engine data for Predictive Maintenance.

Problem definition:-

We will try to model RUL(defined in later paragraph) for a turbofan engine in this project.

The development of data-driven prognostics models requires the availability of datasets with run-to-failure trajectories. These trajectories need to be comprised of a set of time series of CM data along with the corresponding time-to-failure labels. While CM data are often available in abundance, they typically lack the corresponding time-to-failure labels due to the rarity of occurring failures in safety-critical systems and the excessive preventive maintenance.

Data availability:-

Due to the sensitive nature of failures and the potential legal implications, manufacturers and operators have been reluctant to share prognostics datasets of their assets openly. We have data provided on NASA website(link available as Ref(1).

RUL (Remaining Useful life)

The problem of predicting how long a particular industrial asset/machine is going to operate until a system failure occurs, i.e., predicting RUL, is also referred to as prognostics.
Deploying successful prognostic methods in real-life applications would enable the design of intelligent maintenance strategies to determine with a sufficiently long lead time before failure when interventions need to be performed. Such maintenance strate- gies have the potential of reducing costs, machine downtime, and the risk of potentially catastrophic consequences if the systems are not maintained in time.

Importance:

Failures of safety-critical systems such as aircraft engines can cause significant eco- nomic disruptions and have potentially high social costs. The prediction of the system's failure time is therefore of great importance for maintaining the functionality of safety-critical systems and society.

Motivation:

My primary motivation for this project is to explore more about a course (**SC640-Applied Predictive analytics**) which I did last semester. In that course we were taught about various predictive maintenance techniques/methods for engineering systems.
In that course we first learn basic statistical methods and machine learning techniques. Then move on to some system where sir explained how these methods are used in the real world.
So this project makes **PyCK** useful for me to explore the course and this field.

Implementation:

We used these libraries

1. Pandas (for data import and process)
2. Seaborn and matplotlib.pyplot (for visualisation and Plotting)
3. Sklearn (for machine learning techniques/algorithms)

My preparation :
I did some courses on the DataCamp portal to learn about the sklearn library and more on these machine learning techniques. Which I used in this project.
And learn about the data from research papers(ref.).

Dataset (CMAPSS Data)

The data are provided as a zip-compressed text file with 26 columns of numbers, separated by spaces. Each row is a snapshot of data taken during a single operational cycle; each column is a different variable. The columns correspond to:
1) unit number
2) time, in cycles
3) operational setting 1
4) operational setting 2
5) operational setting 3
6) sensor measurement 1
7) sensor measurement 2
...
26) sensor measurement 26

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   unit_num     20631 non-null  int64
 1   cycle_time   20631 non-null  int64
 2   os1          20631 non-null  float64
 3   os2          20631 non-null  float64
 4   os3          20631 non-null  float64
 5   sensor_01    20631 non-null  float64
 6   sensor_02    20631 non-null  float64
 7   sensor_03    20631 non-null  float64
 8   sensor_04    20631 non-null  float64
 9   sensor_05    20631 non-null  float64
 10  sensor_06    20631 non-null  float64
 11  sensor_07    20631 non-null  float64
 12  sensor_08    20631 non-null  float64
 13  sensor_09    20631 non-null  float64
 14  sensor_10    20631 non-null  float64
 15  sensor_11    20631 non-null  float64
 16  sensor_12    20631 non-null  float64
 17  sensor_13    20631 non-null  float64
 18  sensor_14    20631 non-null  float64
 19  sensor_15    20631 non-null  float64
 20  sensor_16    20631 non-null  float64
 21  sensor_17    20631 non-null  int64
 22  sensor_18    20631 non-null  int64
 23  sensor_19    20631 non-null  float64
 24  sensor_20    20631 non-null  float64
 25  sensor_21    20631 non-null  float64
 26  sensor_22    0 non-null      float64
 27  sensor_23    0 non-null      float64
 28  sensor_24    0 non-null      float64
 29  sensor_25    0 non-null      float64
 30  sensor_26    0 non-null      float64
dtypes: float64(27), int64(4)
memory usage: 4.9 MB
```

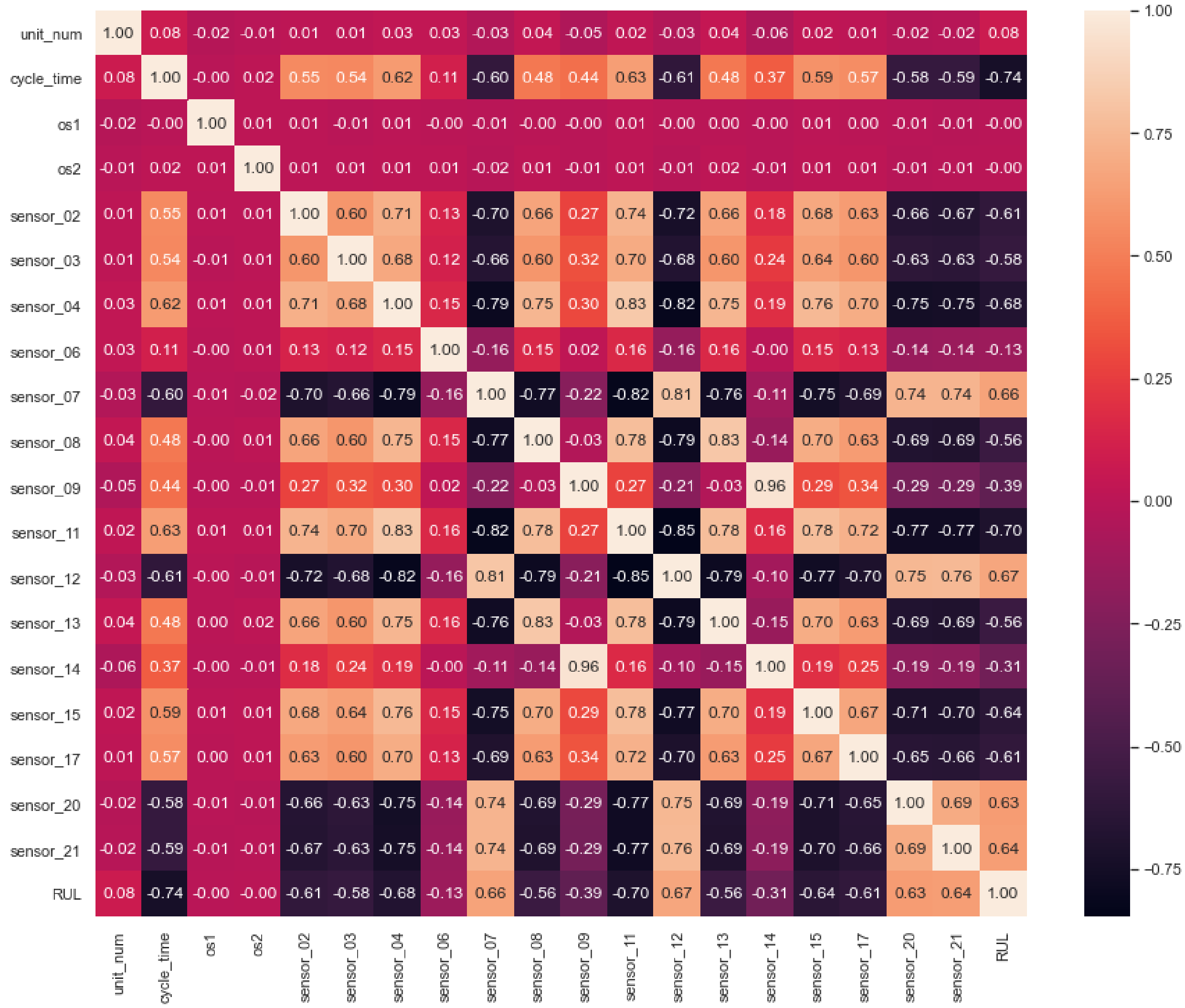
From this data set we explored properties of the features.

When we explored the data some of the column values were constant so we removed them and we also removed null(NaN) values/entry columns.

After these our remaining data is presented with describe function of Pandas library.

	unit_num	cycle_time	os1	os2	sensor_02	sensor_03	sensor_04	sensor_06	sensor_07	sensor_08	sensor_09	sensor_11	sensor_12	sensor_13	sensor_14	sensor_15	sensor_17	sensor_20	sensor_21	RUL
count	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000	20631.000000
mean	51.506568	108.807862	-0.000009	0.000002	642.680934	1590.523119	1408.933782	21.609803	553.367711	2388.096652	9065.242941	47.541168	521.413470	2388.096152	8143.752722	8.442146	393.210654	38.816271	23.289705	107.807862
std	29.227633	68.880990	0.002187	0.000293	0.500053	6.131150	9.000605	0.001389	0.885092	0.070985	22.082880	0.267087	0.737553	0.071919	19.076176	0.037505	1.548763	0.180746	0.108251	68.880990
min	1.000000	1.000000	-0.008700	-0.000600	641.210000	1571.040000	1382.250000	21.600000	549.850000	2387.900000	9021.730000	46.850000	518.690000	2387.880000	8099.940000	8.324900	388.000000	38.140000	22.894200	0.000000
25%	26.000000	52.000000	-0.001500	-0.000200	642.325000	1586.260000	1402.360000	21.610000	552.810000	2388.050000	9053.100000	47.350000	520.960000	2388.040000	8133.245000	8.414900	392.000000	38.700000	23.221800	51.000000
50%	52.000000	104.000000	0.000000	0.000000	642.640000	1590.100000	1408.040000	21.610000	553.440000	2388.090000	9060.660000	47.510000	521.480000	2388.090000	8140.540000	8.438900	393.000000	38.830000	23.297900	103.000000
75%	77.000000	156.000000	0.001500	0.000300	643.000000	1594.380000	1414.555000	21.610000	554.010000	2388.140000	9069.420000	47.700000	521.950000	2388.140000	8148.310000	8.465600	394.000000	38.950000	23.366800	155.000000
max	100.000000	362.000000	0.008700	0.000600	644.530000	1616.910000	1441.490000	21.610000	556.060000	2388.560000	9244.590000	48.530000	523.380000	2388.560000	8293.720000	8.584800	400.000000	39.430000	23.618400	361.000000

Further We plot heatmap of this data using Seaborn library’s heatmap function.



From this we can see some of the features are highly correlated with each other.
We calculated the high correlated columns (|corr| > 0.8) which are mentioned below

```
[('sensor_04', 'sensor_11', 0.8301356963159815),
 ('sensor_04', 'sensor_12', -0.815590516105214),
 ('sensor_07', 'sensor_11', -0.8228050249957691),
 ('sensor_07', 'sensor_12', 0.812712601325414),
 ('sensor_08', 'sensor_13', 0.8260843322333569),
 ('sensor_09', 'sensor_14', 0.9631566003059776),
 ('sensor_11', 'sensor_12', -0.8468835930051095)]
```

Removed some of the features based on these correlations and some plotting of the data.
["os1","os2","sensor_07","sensor_12","sensor_09",]

Now we are ready with our data to perform machine learning techniques.

Regression analysis:

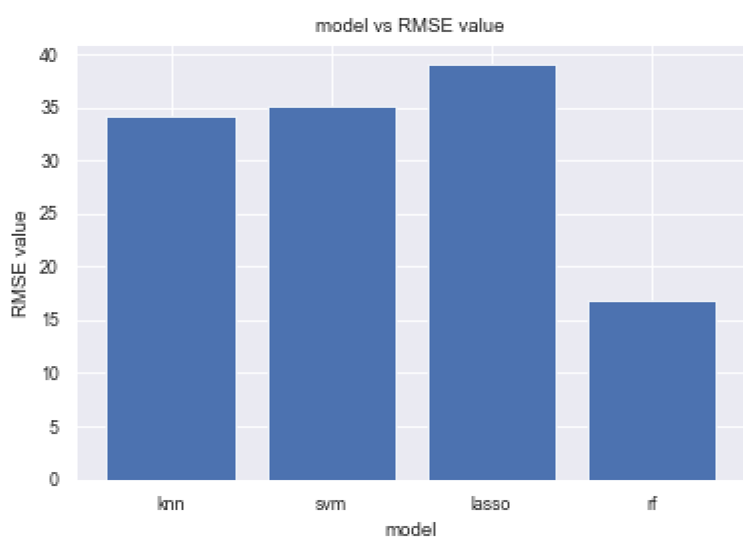
In this we will try to model RUL from the available data. Our main objective is to minimize the RMSE score.

Applied algorithms :

1. KNearest Neighbor(knn)
2. SVM(svm)
3. Lasso regression(lasso)
4. Random forest regression(rf)

Result/score of these algorithms:

We are considering RMSE(lower better) as our performance score:



Random forest gives us the best performance here.

	knn	svm	lasso	rf
MAE	24.175830	23.721424	30.240944	11.219696
MSE	1174.304721	1228.091970	1520.815819	287.413009
RMSE	34.268130	35.044143	38.997639	16.953260
R2_score	0.743623	0.731880	0.667971	0.937251

(other values)

Classification analysis:
Now we want to know if the engine is ready to use or not.
So we consider if (RUL <30) RUL is low then it is not safe to use that engine.
We will classify these low RUL engines from better RUL engines.

Applied algorithms:

1. KNearest Neighbor Classifier(knn)
2. LogisticRegression(lr)
3. Radial Basis Function SVM(svm)
4. RandomForestClassifier(rf)

Accuracy is our prefer performance parameter here

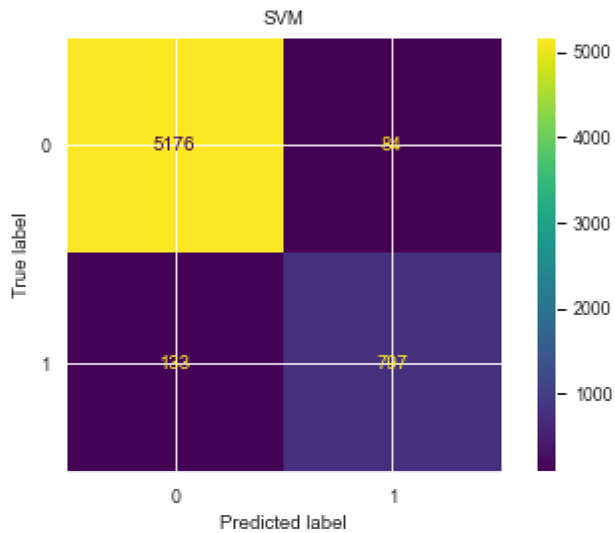
LogisticRegression gives best performance.

(other scores)

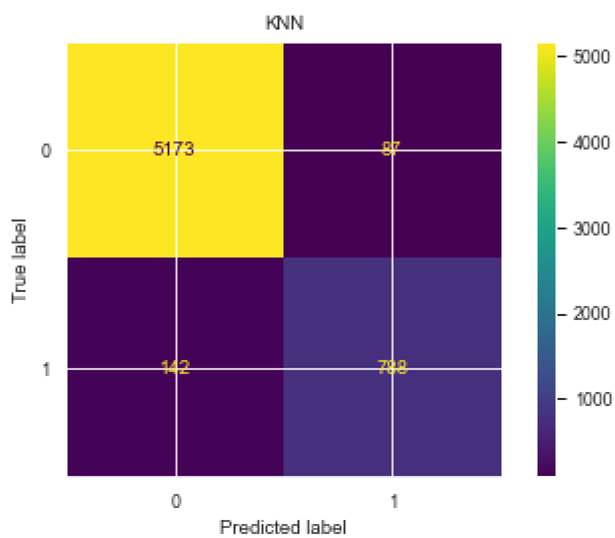
Result/score of these algorithms :

	knn	lr	svm	rf
Accuracy Score	0.963005	0.959935	0.964943	0.962197
Precision Score	0.900571	0.875551	0.904654	0.881579
Recall Score	0.847312	0.854839	0.856989	0.864516
F1 Score	0.873130	0.865071	0.880177	0.872964
Auc Score	0.915386	0.916678	0.920510	0.921992

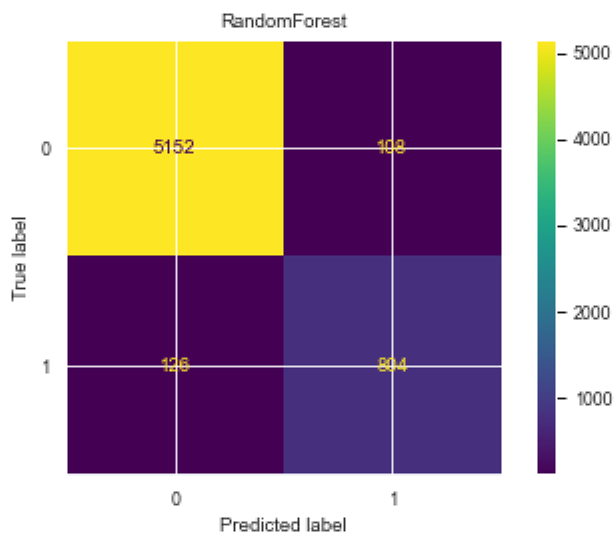
Confusion_matrix for these algorithms:
SVM :



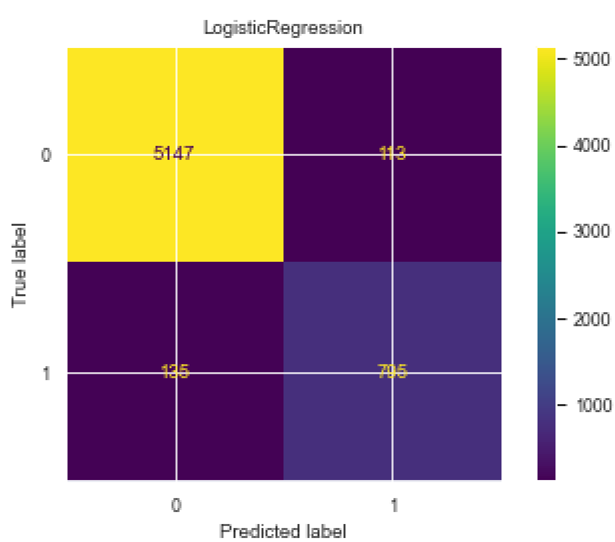
KNeighborsClassifier



RandomForest



LogisticRegression

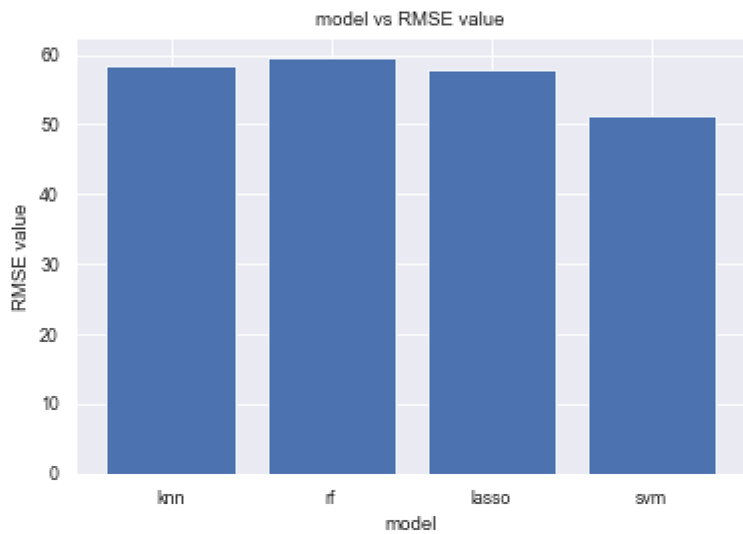


Unseen data performance

Now using these models to predict the RUL for new unseen data.

Regression analysis:
Result/score of these algorithms :

We are considering RMSE(lower better) as our performance score:

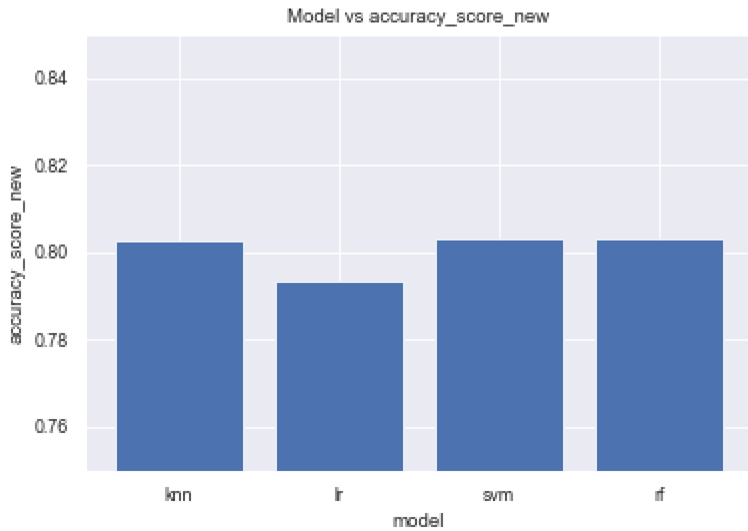


SVM gives us the best performance here but in training we got best performance from random forest.
Other parameters scores:

	knn	rf	lasso	svm
MAE	44.536367	44.726059	45.397228	39.120883
MSE	3411.669644	3539.254016	3350.586909	2624.295740
RMSE	58.409500	59.491630	57.884254	51.227880
R2_score	-0.212000	-0.257324	-0.190300	0.067716

Classification analysis:

Accuracy is our prefer performance parameter here

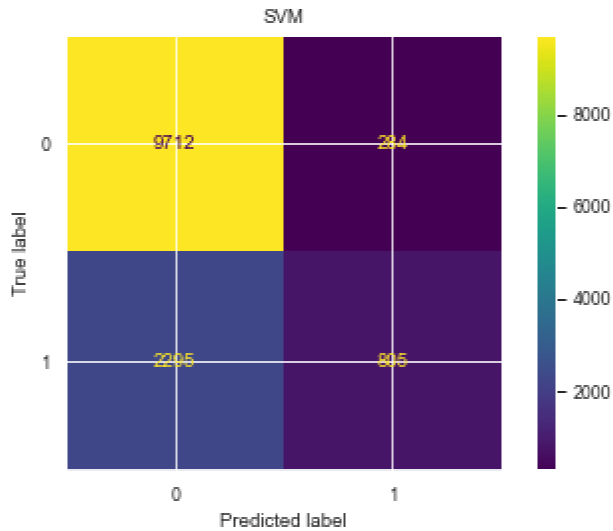


LogisticRegression gives best performance.(same as training)

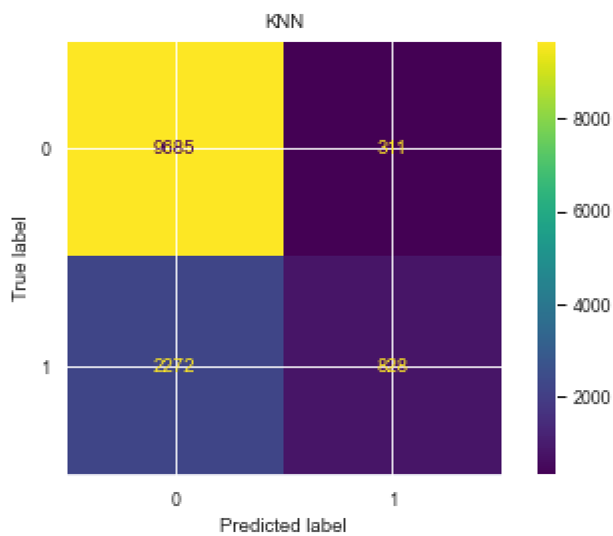
	knn	lr	svm	rf
Accuracy Score	0.802764	0.793601	0.803070	0.802917
Precision Score	0.726953	0.666109	0.739210	0.712531
Recall Score	0.267097	0.256774	0.259677	0.280645
F1 Score	0.390658	0.370664	0.384340	0.402685
Auc Score	0.617992	0.608429	0.615633	0.622766

Confusion_matrix for these algorithms:

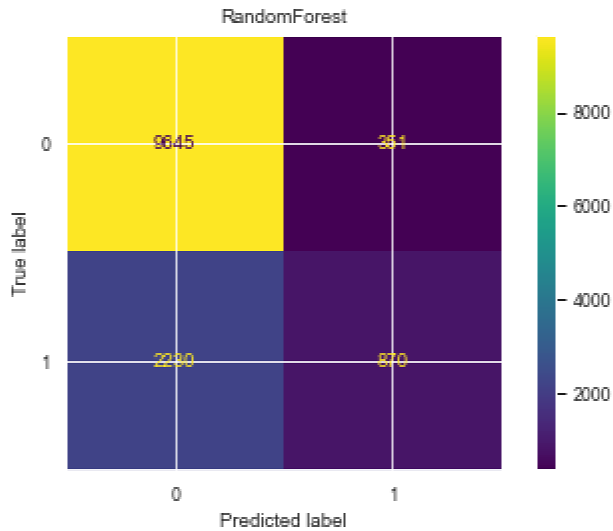
1. SVM



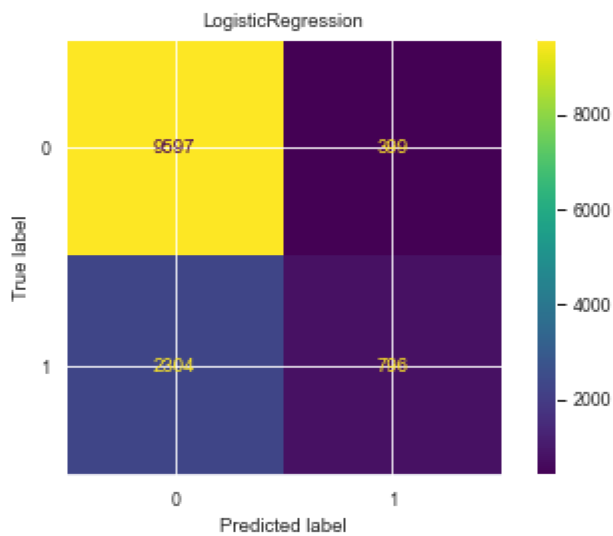
2. Knn



3. RandomForest



4. LogisticRegression



Conclusion:

Applied these techniques to model the RUL of turbofan engines got some interesting results.

We got classification results close to 80% accuracy.

As we can see when we apply our model to unseen data performance decreases which can be due to less data because we are using only 100 engines data to train.

So if we increase our training data we may get better results.

Future work:
Will apply more algorithms/methods on this data to improve performance.
(Xgboost,Neural network etc.)

References:

1.https://ti.arc.nasa.gov/c/13/ (data link)
2. A. Saxena and K. Goebel (2008). "PHM08 Challenge Data Set", NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA
3. A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation", in the Proceedings of the 1st International Conference on Prognostics and Health Management (PHM08), Denver CO, Oct 2008.
3.Performance Benchmarking and Analysis of Prognostic Methods for CMAPSS Datasets "Emmanuel Ramasso and Abhinav Saxena"