COSC2667 – Data Science Post Graduate Project

Vikas Virani (s3715555)

Table of Contents

Introduction

(Background): Ford Motor Company of Australia, the popular corporation in automobile market aiming to use their existing data to draw different insights and minimize the manual work that needs to be done by employee. Every day, Ford receives a lot of warranty claims data, which is currently processed by DataRobot; enterprise AI platform and automated machine learning platform which makes it fast and easy to build and deploy accurate predictive models. It gives top ML models (In terms of Accuracy) based on the target attribute selected (i.e. Failure mode) from data. Failure mode is multicategory attribute which represent which category of warranty claim a record goes into. This data is then fed to Alteryx platform, a leader in data science and self-service analytics which can prepare, blend, enrich, and analyse data.

(Introduction): The Aim of the project is to automize this process by combining the previous data as well as oncoming data to train a ML model including a hyperparameter tuning and feature selection to find the best ML model. It should improve accuracy and automate the data feeding to Alteryx along with these predicted failure modes & visualize data to derive meaningful insights.

Warranty claims at Ford are currently analysed manually to group by “failure modes”. Our main aim is to automize the failure mode binning for the highest occurring problems through AI.

The deliverables of the project are as follows: -

- Understand and demonstrate the difference of multiple classifiers (Multiclass) vs single classifier (Binary) in machine learning
- How well can machine learning explain which attribute contributes most to certain key failure modes
- How well does machine learning predict the failure modes based on claims data
- Which threshold to select for predictions with respect to confidence of correct prediction (e.g. prediction probability)
- Recommendations how to improve the data set to improve machine learning for better prediction results
- Recommendation if machine learning can be used as a substitute of a human being
- Cost benefit of machine learning vs manual claims processing under consideration of machine learning maintenance, software cost, etc

Some Bonus deliverables of the project are as follows: -

- Implement machine learning code into data processing software to automatically process incoming new customers claims
- Highlight trends or significant changes in newly processed data

There was no significant change in the original aim of the project, however approach was more focused on in depth analysis of above deliverables and removed analysis of time series component to the bonus deliverable parts.

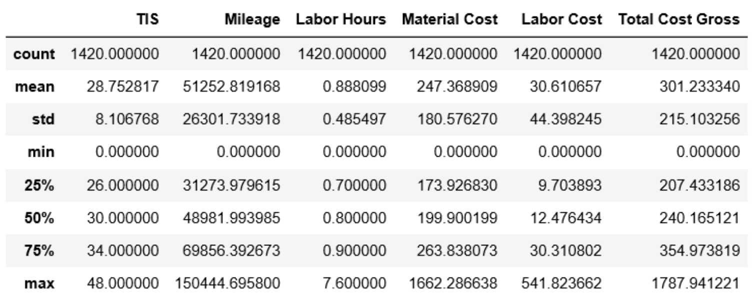
This report aims to fulfil all these deliverables by analysing data and exploring it with 3 different ML approaches which are explained in detail in subsequent sections. More details regarding data are given in the next section.

Approach / Methodology

Data Pro-processing

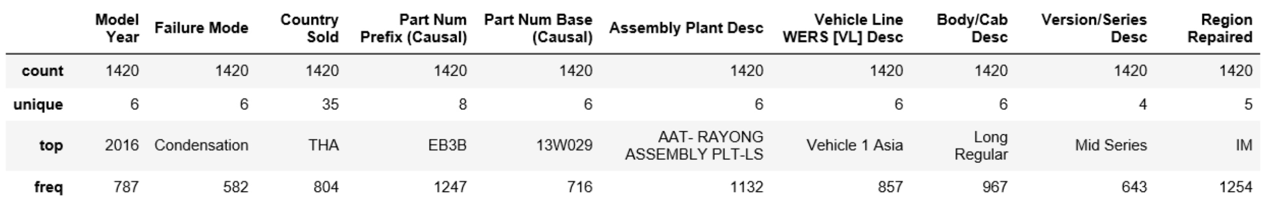
The sample data to be worked on was given by ford which had 7936 records of warranty claims data out of which, 1420 records had manually binned/identified Failure mode for that claim record and the rest were empty to be predicted on. Here, it is important to note that this data had only 6 failure modes related to only one specific part of the vehicle (for example headlamp), the idea of ford is to generalize this model across all categories and subcategories of failure modes if this model succeeds on this data. As it was sent directly, we read dataset from the local storage only. Also, since the data set contained the attribute names, we do not need to explicitly specify those during loading our dataset. The provided dataset has 23 features and one target column, again 1 feature is verbatim/comments on defect provided by customer and other 22 features are attributes regarding the vehicle like, model year, assembly line of vehicle, claim date, total time to service the vehicle since bought (TIS), labour cost, material cost etc., and more of these attributes can be added by ford for the vehicle to more accurately predict the Failure Modes. To check the data quality; data types and null counts of each of the attributes are checked first. There were less than 10 record with null value of feature, so I decided to drop them. To further make it Machine learning suitable, I have converted some of the features to categories and then checked value counts of each of them to see the distribution of values. To further investigate, I have plotted a summary statistic for all features to get an idea of distribution of values and any outliers. Here, only 15 features and 1 target variable are summarised as I have dropped other 7 features/columns which were not useful for our case. Below is a summary of sample dataset;

Summary Table of continuous features



	TIS	Mileage	Labor Hours	Material Cost	Labor Cost	Total Cost Gross
count	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000	1420.000000
mean	28.752817	51252.819168	0.888099	247.368909	30.610657	301.233340
std	8.106768	26301.733918	0.485497	180.576270	44.398245	215.103256
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	26.000000	31273.979615	0.700000	173.926830	9.703893	207.433186
50%	30.000000	48981.993985	0.800000	199.900199	12.476434	240.165121
75%	34.000000	69856.392673	0.900000	263.838073	30.310802	354.973819
max	48.000000	150444.695800	7.600000	1662.286638	541.823662	1787.941221

Summary Table of categorical features



	Model Year	Failure Mode	Country Sold	Part Num Prefix (Causal)	Part Num Base (Causal)	Assembly Plant Desc	Vehicle Line WERS [VL] Desc	Body/Cab Desc	Version/Series Desc	Region Repaired
count	1420	1420	1420	1420	1420	1420	1420	1420	1420	1420
unique	6	6	35	8	6	6	6	6	4	5
top	2016	Condensation	THA	EB3B	13W029	AAT- RAYONG ASSEMBLY PLT-LS	Vehicle 1 Asia	Long Regular	Mid Series	IM
freq	787	582	804	1247	716	1132	857	967	643	1254

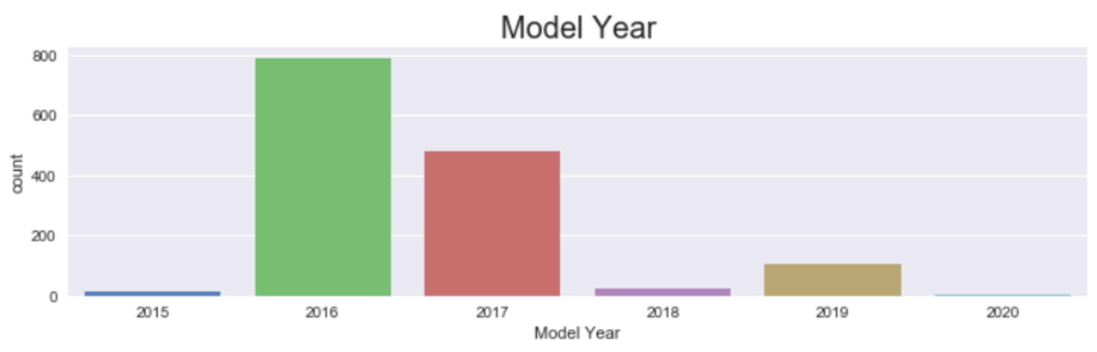
Some random verbatim records from the sample data are given in the below image;

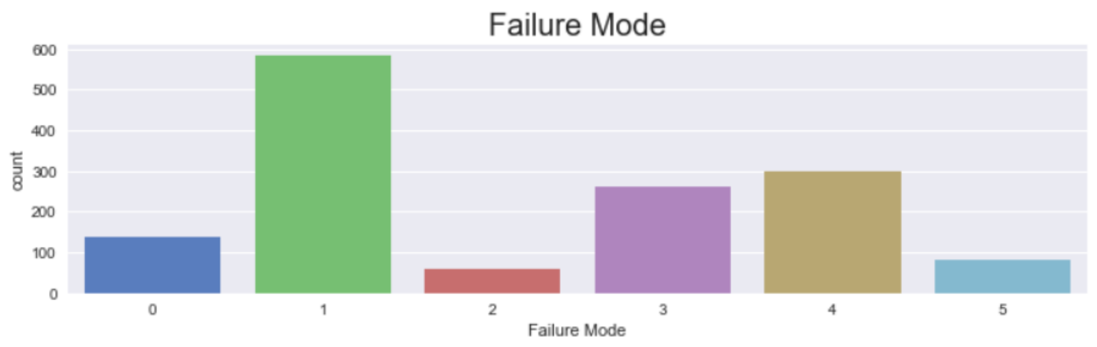
```
36     water inside of the headlamp causing for me th...
37     had me sealing in the lens of the front headla...
51     01farol right side with me sealing enters the ...
63     c c top edges of both head light s faded going...
64     premature discolouration deterioration of head...
88     ap s to accomplish test of condenses the as co...
119    it fails in sealing enters the lens and the ca...
144    headlight lens material de laminated and dull ...
147    cause headlamp lens left and right who was hol...
227    headlamp right side with me sealing causing in...
272    def evidenced ap s complains it of the custome...
285    cause projector the cup of the front left and ...
300    is front headlamp right side with sealing defe...
336    we verify infiltrates the of water in the head...
355    it does not work regulating of sptica lefthand...
356    inspected lights and found there has been mois...
361    problem lens over the south china sea there is...
370    cause the headlights was the misty solution re...
372                                                name
373    an inspection by the mechanic found that the f...
Name: Verbatim Combined, dtype: object
```

In the next section, we will continue with exploring our dataset features.

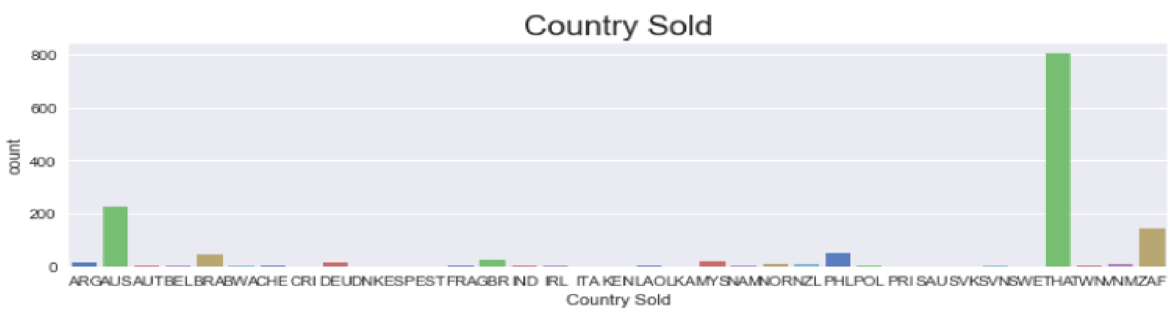
Data Exploration

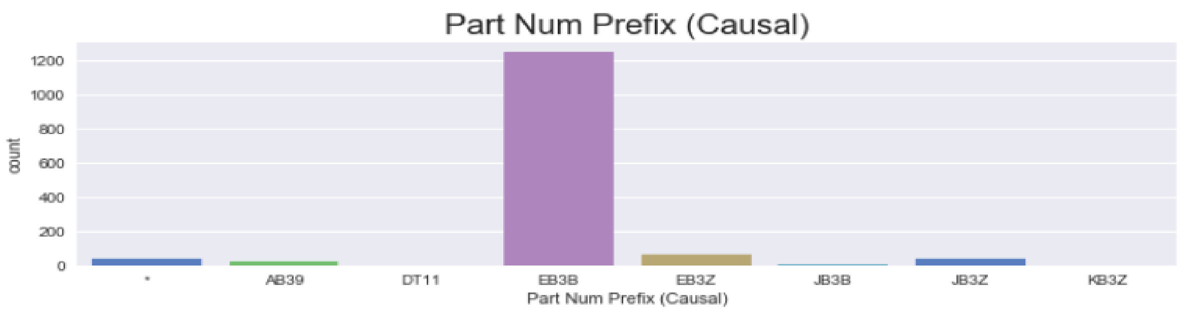
To avoid code repetition, I have defined two functions named ‘BarPlot(x)’ and ‘BoxAndHistogramPlot(x)’ for categorical and numerical features respectively. For given an input categorical column ‘x’, ‘BarPlot(x)’ returns a bar chart. A bar chart is useful to present the proportions by categories. For given an input numerical column ‘x’, ‘BoxAndHistogramPlot (x)’ plots a histogram and a box plot. A histogram is useful to visualize the shape of the underlying distribution whereas a box plot tells the range of the attribute and helps detect any outliers. The visualisation plots are as follows, here, I have encoded the failure modes using label encoder. It is important to note that model year will not have any significance for our original problem definition as the data is not a full dataset and model year can’t decide the failure mode, although we can identify if some model year had a specific failure mode with high frequency just by visualising failure modes grouped by model year for historic data. Another thing to note here is some of the features’ labels are encoded along with the failure modes before visualisation to adhere to dataset disclosure guideline and to prevent displaying whole dataset except from the summary and some sample values of data.

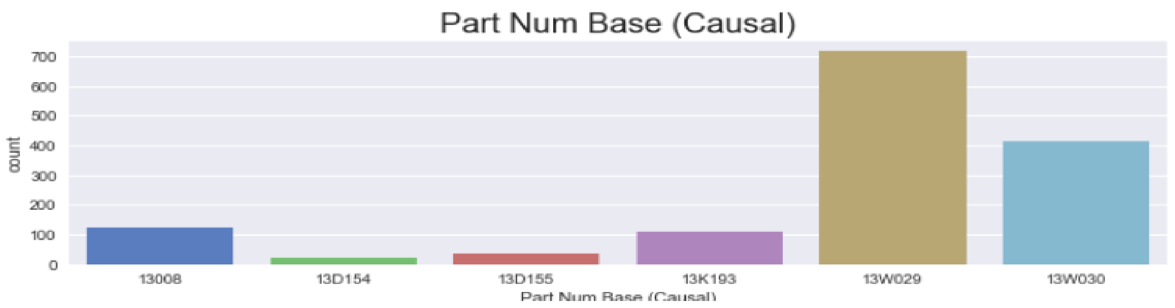




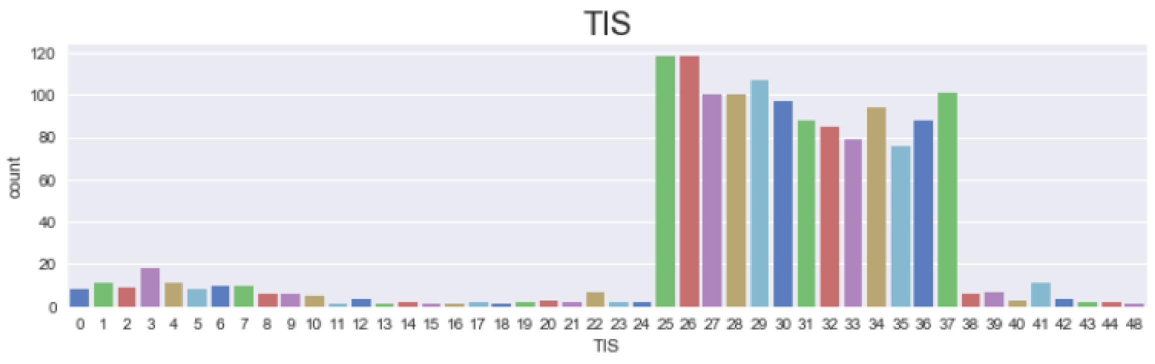
Next part shows in which country the vehicle was sold and part number used for the repairing.



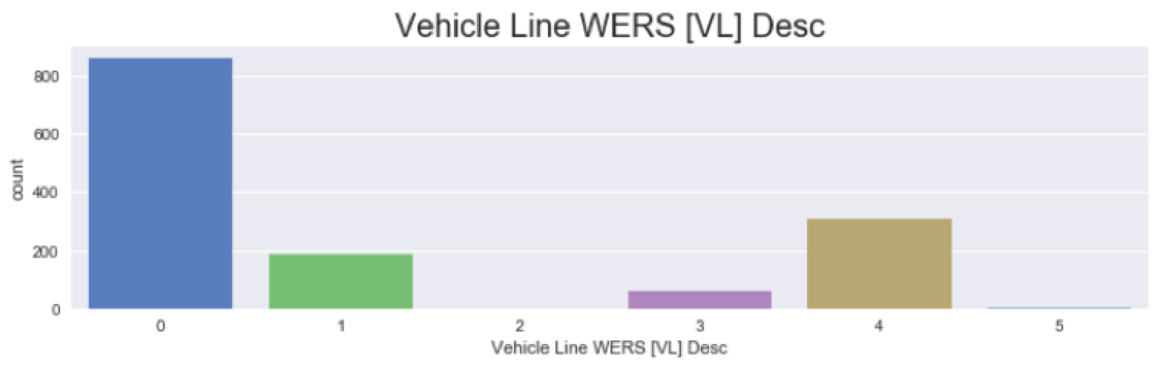


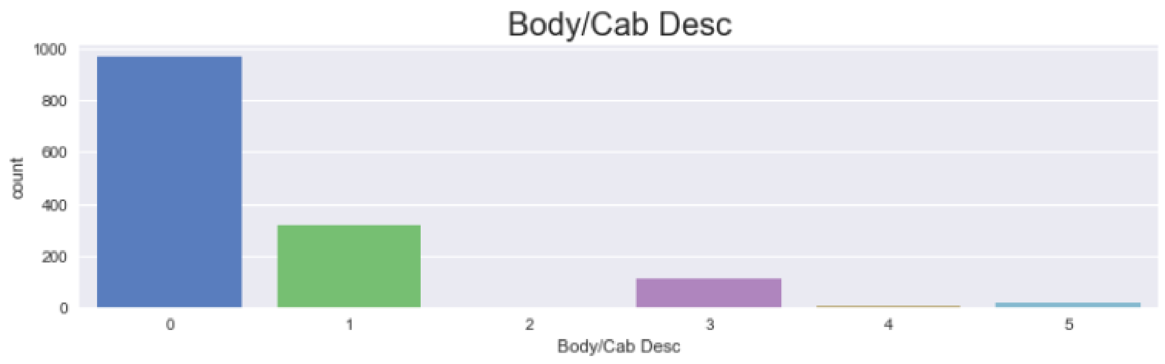


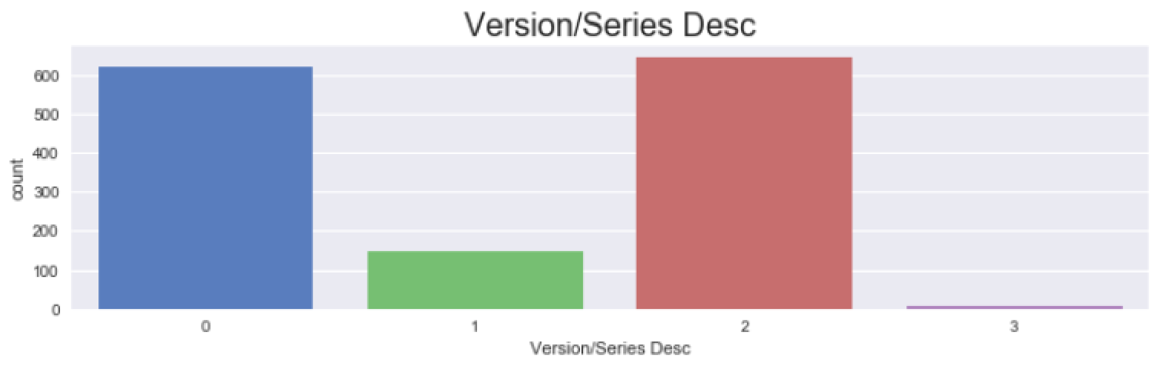
Next chart is for TIS- time in service for the vehicle, it can be seen from the graph that most of the vehicles are going in service in their 3rd year, i.e. from 25h month to 37th month.



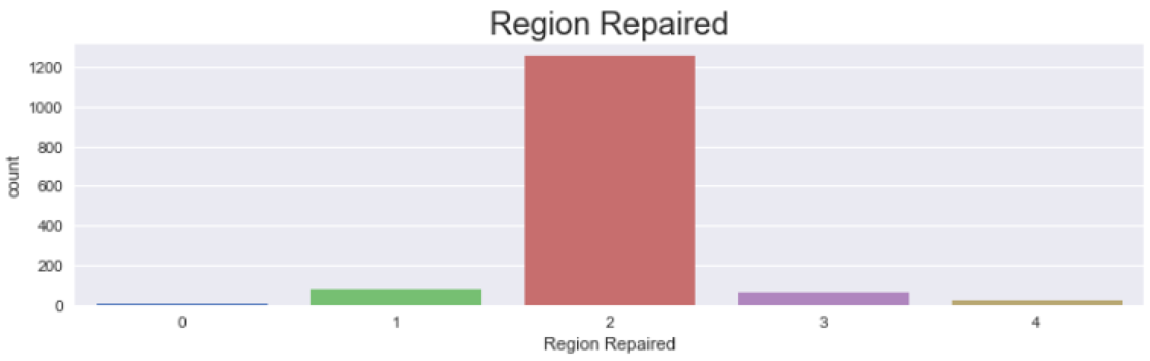
Following chart shows the distribution of types and versions/series of vehicles which were claimed for warranty. Note that all these categorical variables are label encoded already.





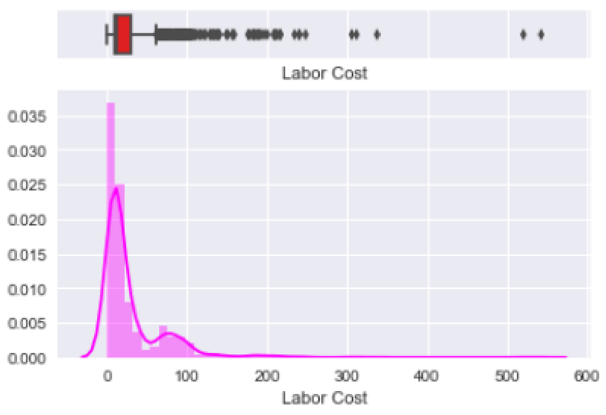


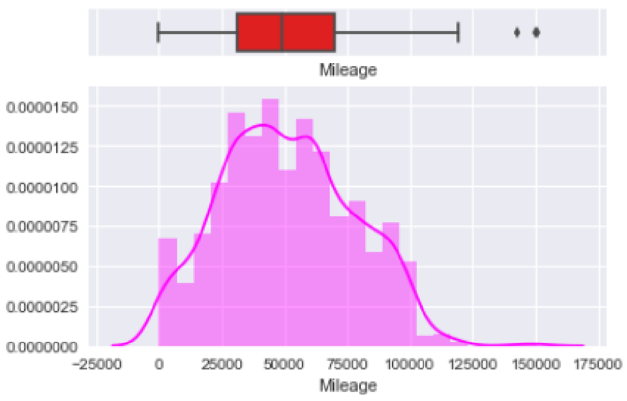
The next chart shows which region the claimed vehicle was repaired in,

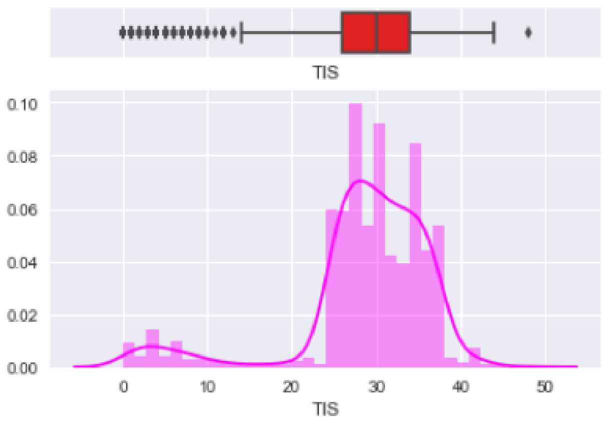


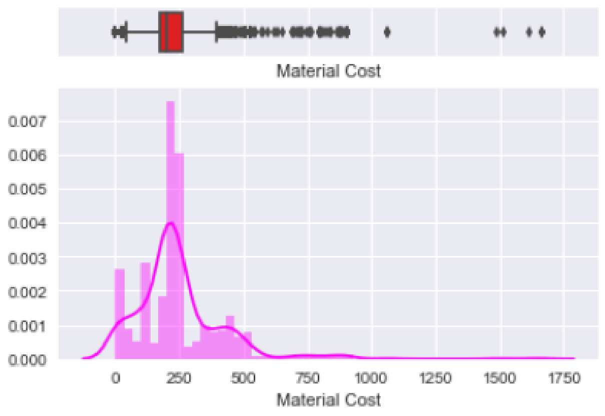
As stated earlier, it is not a whole data of ford, so, we cannot derive that region 2 has most occurring cases of repairs but by visualising this, **ford can check** whether a **particular assembly line is responsible for a particular failure mode** or if for example some particular **body type or vehicles series** from **specific assembly** line is responsible for a **particular failure mode** by grouping these features by failure modes and visualising them.

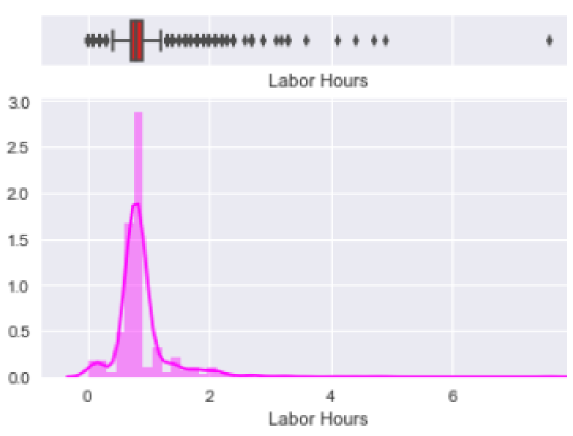
Next, I have tried to plot histogram and boxplot for all numeric features to identify the distribution of values in them,

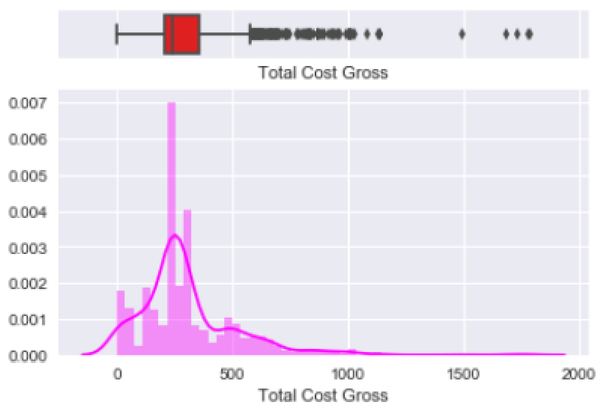












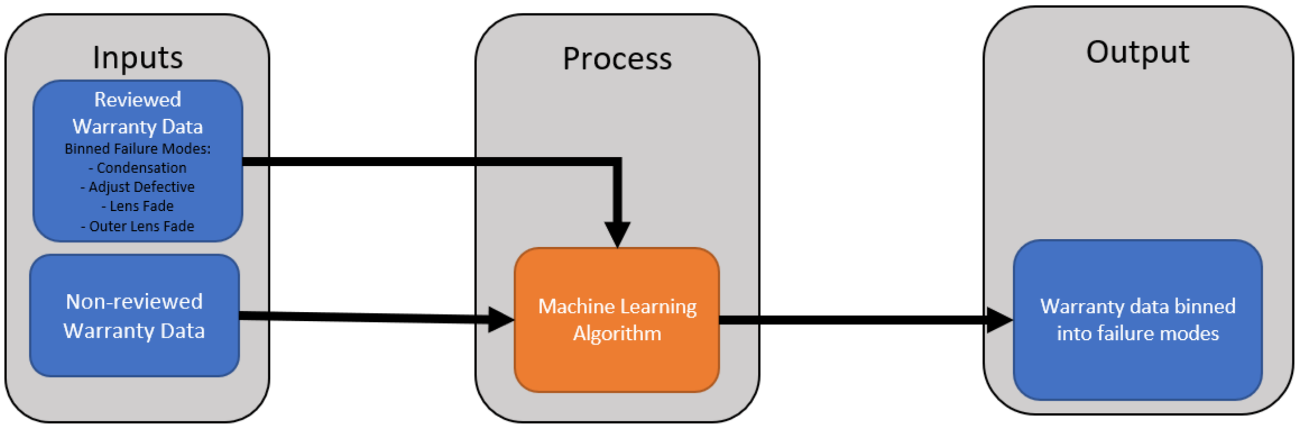
Here, as can be seen there are many outliers, but it won't be considered as outliers, instead it can used in our ML model and can be the important predicting variables different failure modes may have different values for above variables like, some failure modes have higher labour or material cost associated with it, while other failure modes might be occurring with lower Time in Service/Mileage values.

Apart from this, I have also visualised some multivariate visualisation as stated earlier, like Histogram of all numeric features segregated by Target (Failure Mode) and All categorical attributes segregated by Failure Mode to find the dependence/correlation among them in order to select them for our machine learning approach.

Combining all of these, I've decided to drop some of the variables/features from our data which are not of use for our use case/aim of the project. After dropping those features, there were 16 features in total (15 regular features + 1 verbatim/user comments) and 1 target variable - Failure Mode, in our final dataset.

Methodology

From here, the approach was divided into 3 separate explorations for ML and to mimic the DataRobot like functionality to identify best features and build a final ML model which can predict failure modes. The key objective was to identify how well machine learning can 'bin' failure modes based on raw warranty claims data. The basic diagram for what it should do in the end is provided with example failure modes as below;



The exploration was divided into: -

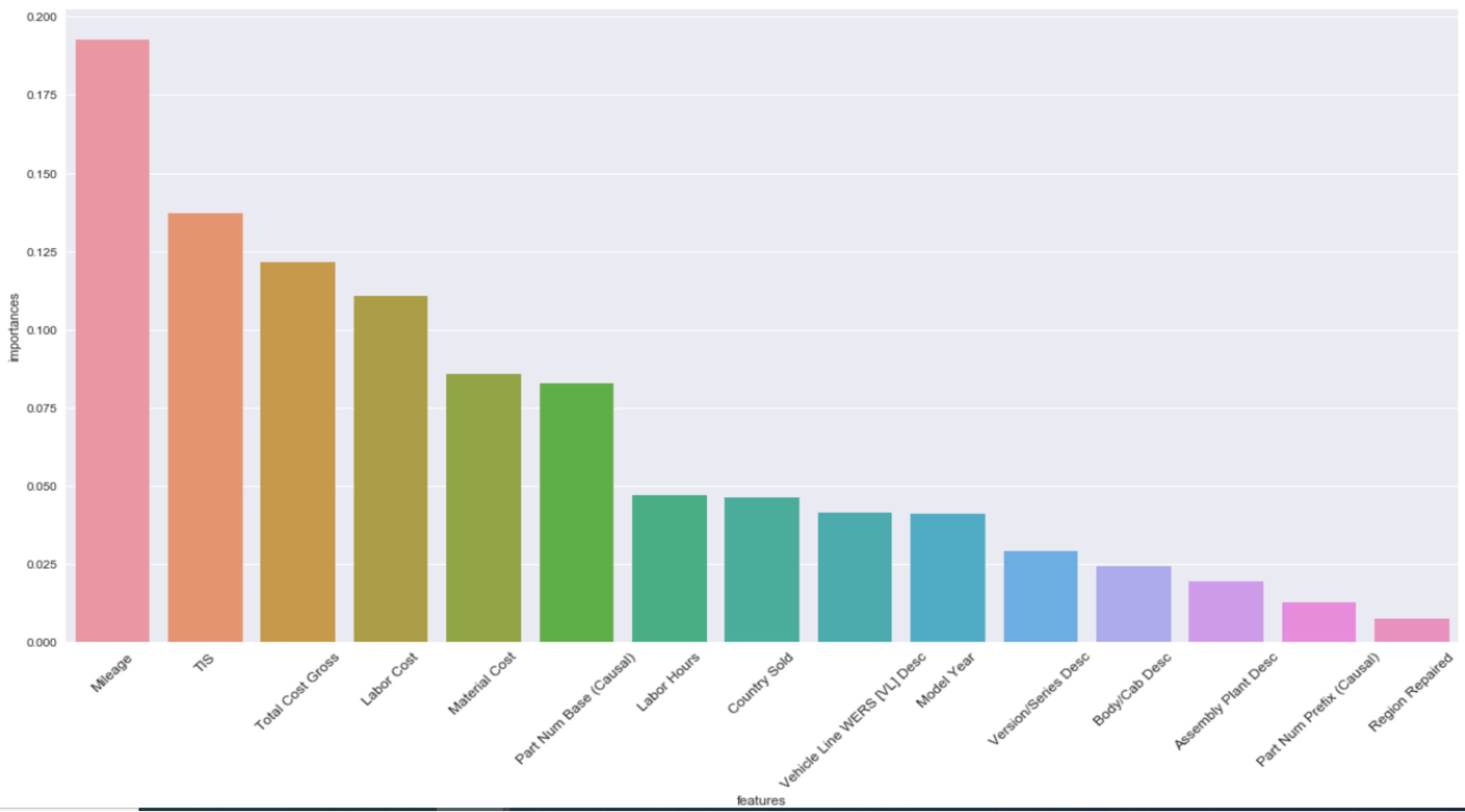
1) Using all categorical and numeric features except from Verbatim (customer comments/feedbacks) on different multiclass classification algorithms.
2) Use only verbatim and train binary classification as one vs. other failure modes, i.e. for each failure mode; classify records into whether they belong to that failure mode or other failure modes, repeated till "number of failure mode" times.
3) Combine all categorical and numerical features as well as verbatim provided from customer and build a one vs. other failure modes, ML model to predict on raw data.

I have built the following multiclass classifiers to predict the target feature for exploration 1:

- K-Nearest Neighbours (KNN), Bagging, XGBoosting, SVM, (Gaussian) Naive Bayes (NB), Random Forest, BernouliNB and MLP (Multi-Layer Perceptron) Classifier [1]. The idea here was to extensively explore (like what DataRobot does) most known multiclass classifiers to see which one, whether performs best or not. As these are the popular models for multiclass classification which gives higher accuracy as per various researches published, they were selected for this approach. I have selected this approach to compare it with other approach as mentioned in first deliverable.

Before fitting a model, we select the best features using the powerful Random Forest Importance method with 100 estimators inside a pipeline. A trick here is that we need a bit of coding so that we can make RFI feature selection as part of the pipeline. For this reason, I have defined the custom 'RFIFeatureSelector' class to pass in RFI as a "step" to the pipeline. I have considered 5, 10 and the full set of features (with 15 features) after encoding of categorical features. Hyperparameter tuning part for each of the model is done by using GridSeachCV.

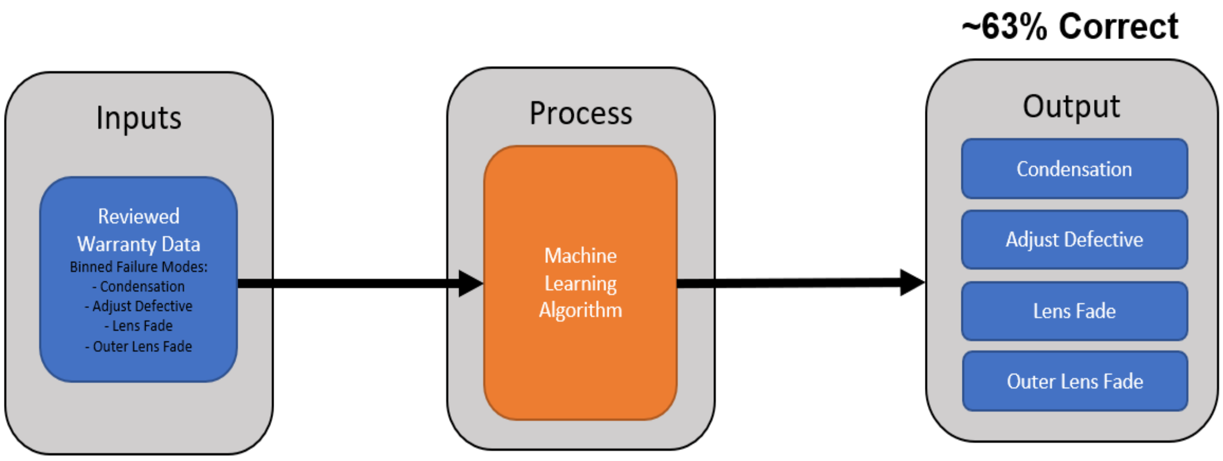
Since NB requires each descriptive feature to follow a Gaussian distribution, I have done a power transformation on the input data before model fitting for that model. For scaling, I have used MinMaxScalar as it holds the distribution among the values as it is and just scales it



between certain value range. I have plotted a feature importance plot for all our features except from verbatim using this class. The importance plot is as below;

Using feature selection together with hyperparameter search inside a single pipeline, we conduct a 5-fold stratified cross-validation to fine-tune hyperparameters of each classifier using area under curve (AUC) as the performance metric. We build each model using parallel processing with "-2" cores (Useful when scaling to large amount to data). Since the target has **unbalanced target class issue**, **stratification is crucial** to ensure that each validation set has the same proportion of classes as in the original dataset. I also examined sensitivity of each model with respect to its hyperparameters during the search.

Once the best model is identified for each of the three classifier types using a hyperparameter search on the training data, I conducted a 10-fold cross-validation on the test data and perform a paired t-test to see if any performance difference is statistically significant. In addition, I have compared the classifiers with respect to their recall, f1-scores and confusion matrices on the test data. The models are further fine-tuned by selecting different hyper parameter range if the best model has the hyperparameter values from the extreme ends of original parameters. Almost all classifiers have an average accuracy of 50 to 60% with highest accuracy of 63%, which is not that accurate. Below is a general diagram of the process flow with example failure modes;

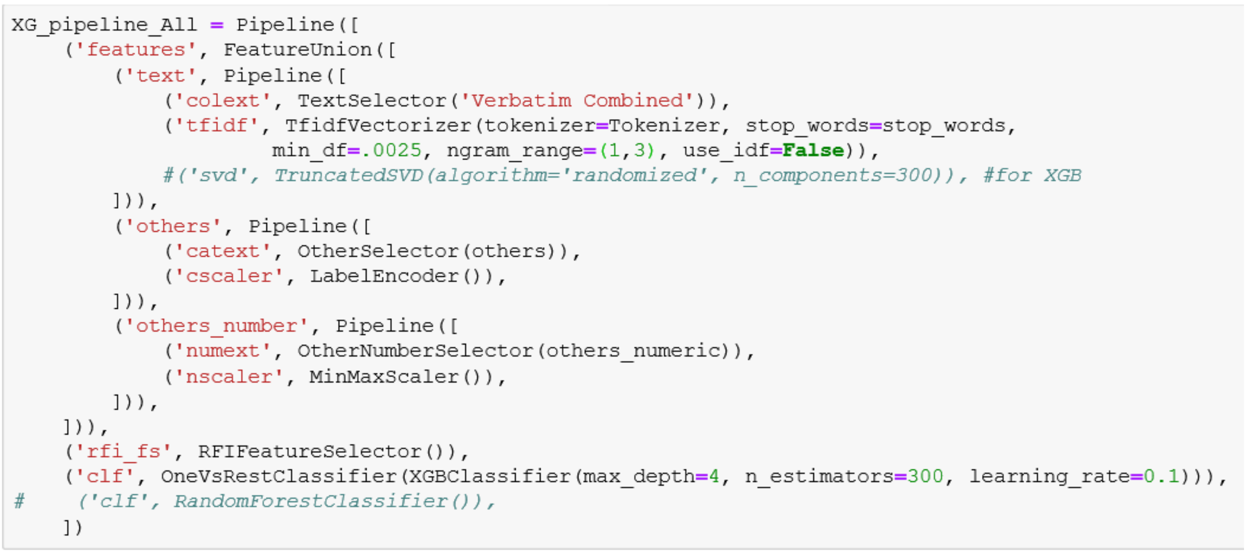


So, we will explore option 2 of our project. I have selected approach 2 and 3 in order to complete deliverable 1. Also, whichever approach has higher accuracy among all of these, it will be used to complete deliverables 2, 3 and 4 because it relates to the final machine learning model. Exploration 2 is to be done using one classifier for each failure mode and using verbatim only as a feature to predict the failure modes. Since, customer comments are very important, and it can generalise a failure mode by using the Natural language processing (NLP) techniques to identify features. Since verbatim is a customer feedback on the defect/failure, it will need to be processed first in order to use it or make it machine readable. So, I have made a custom function called "clean_text" which does all the necessary processing to pre-process customer comments like, convert to lower case, converting apostrophe words, removing punctuations and other symbols, trimming extra spaces etc. I have also defined some custom functions like 'print_stats' to print all statistics about model and 'plot_classification_report' to visualise classification accuracy.

In order to process verbatim, I've selected TfidfVectorizer method (Equivalent to CountVectorizer followed by TfidfTransformer), which is common method known in text analysis like in social media or one of the core logics behind google searches. It gives term frequency of a word across whole corpus (all customer comments in our case) and frequency of document in which that term occurs, values for each word are calculated with these TF-IDF ratios. While doing these, I've also leveraged the stop words dictionary provided inbuilt in nltk package to ignore stop words to be processed from our verbatim. I have customised TfidfVectorizer method's parameter like "**norm**"- for normalization, "**use_idf**"- inverse document frequency reweighting which gives importance to rare words, "**smooath_idf**" – to prevent zero divisions for new word, "**sublinear_tf**" – log values of term frequency to prevent overpowering by some common words. Apart from that, since it is a verbal text data, for similar failure modes, it won't be able to identify with a single word, so I took an "**ngram_range**" to (1,3) to consider up to group of 3 words together. These values are then fed to different classifiers which are wrapped in "OneVsRestClassifier" [1] & [2] method of sklearn library which exactly does the thing that we want to do for our project. Apart from that, I've also made a custom class "TextSelector" which only elects verbatim feature from the whole dataset. All these "TextSelector", "TfidfVectorizer" and our ML models wrapped inside

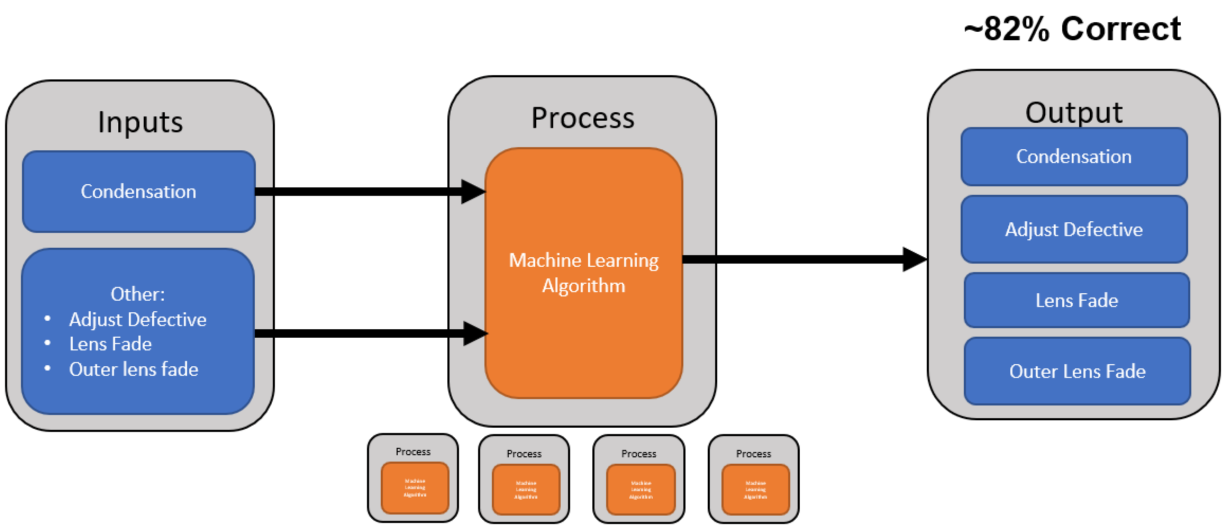
"OneVsRestClassifier" are put together by building a single pipeline to automate all these things which reduces efforts of human work. Again, grid search was used for all combination of parameters for all these tasks in pipeline and best model was derived. Essentially, 4 models are used to compare for this task; NB, XGBoosting, SVC and Logistic Regression, since these 3 gave higher accuracies compared to other models as seen earlier. Creating this pipeline was a challenging task since it was the first encounter for me on dealing with RAW text data as well as categorical and numerical features together in a single model to train.

For exploration 3, the approach taken was the same as the 2nd one, but instead of building a model based on only verbatim, all the other categorical features as well as numeric features are also considered in training a machine learning model. In addition, I have created 2 more custom class names "OtherNumberSelector"- to select numeric columns from data based on list and "OtherSelector"- to select categorical features of data from the list. Same models from earlier task; NB, XGBoosting, SVC and Logistic Regression are used to train the model. The pipeline building to automate an end to end task was a challenging thing here. Below is a screenshot of sample pipeline created for XGBoosting algorithm with all features and verbatim combined;



```
XG_pipeline_All = Pipeline([
    ('features', FeatureUnion([
        ('text', Pipeline([
            ('colext', TextSelector('Verbatim Combined')),
            ('tfidf', TfidfVectorizer(tokenizer=Tokenizer, stop_words=stop_words,
                     min_df=.0025, ngram_range=(1,3), use_idf=False)),
            #('svd', TruncatedSVD(algorithm='randomized', n_components=300)), #for XGB
        ])),
        ('others', Pipeline([
            ('catext', OtherSelector(others)),
            ('cscaler', LabelEncoder()),
        ])),
        ('others_number', Pipeline([
            ('numext', OtherNumberSelector(others_numeric)),
            ('nscaler', MinMaxScaler()),
        ])),
    ])),
    ('rfi_fs', RFIFeatureSelector()),
    ('clf', OneVsRestClassifier(XGBClassifier(max_depth=4, n_estimators=300, learning_rate=0.1))),
#    ('clf', RandomForestClassifier()),
    ])
```

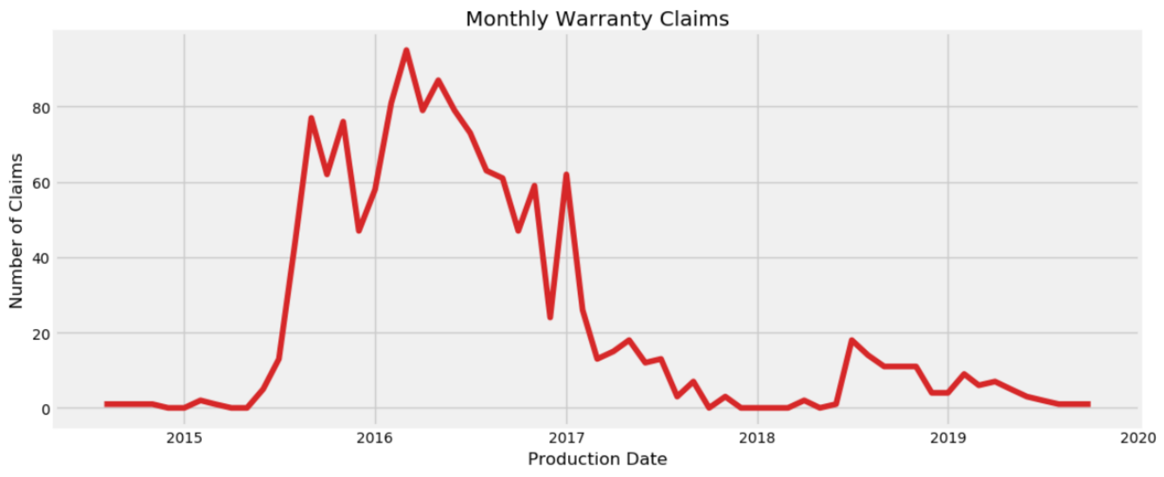
Out of all 3 approaches, this 3rd approach has higher accuracy among all, and XGBoosting performs well compared to other models in this approach. So, we decided to go with XGBoosting combining with all numeric and categorical features as well as customer comments/feedback (i.e. Verbatim Combined feature). The general approach of a final model is as below with example failure modes; (also, deliverables 5, 6 and 7 are general recommendations and observed along the way during the project execution so, there is no approach justification is given tying to these deliverables)



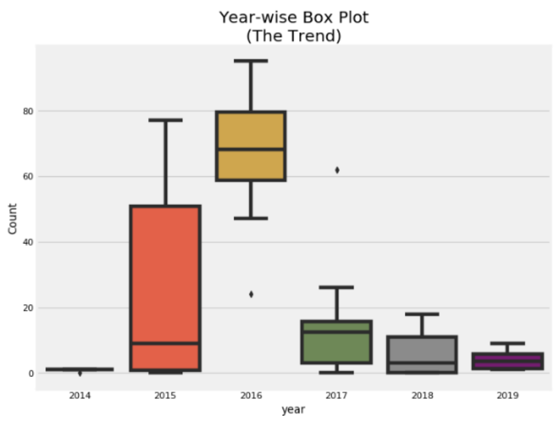
Process repeated for each failure mode

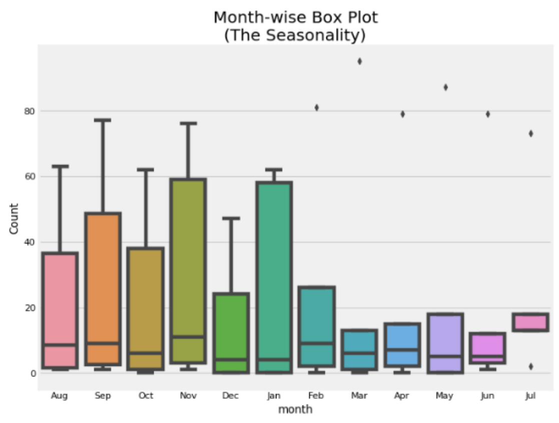
(Bonus Deliverables – Extra work)

This part is from the bonus deliverables regarding time series/trends in data and it was to be done if all deliverables are met and any extra time is remaining. Time series are widely used for non-stationary data, like stock price, economic, retail sales and weather etc. so it is a fair choice to use to identify trends in data and forecast. All deliverables were met, and I have started with exploration stage for this approach, but the data was not adequate to do this component as it was a sampled data and it is not appropriate to find the trends from sample data instead of whole dataset. For example, I have processed data for monthly warranty counts and when visualizing time series for number of claims in each month for each year, the following graph was plotted,

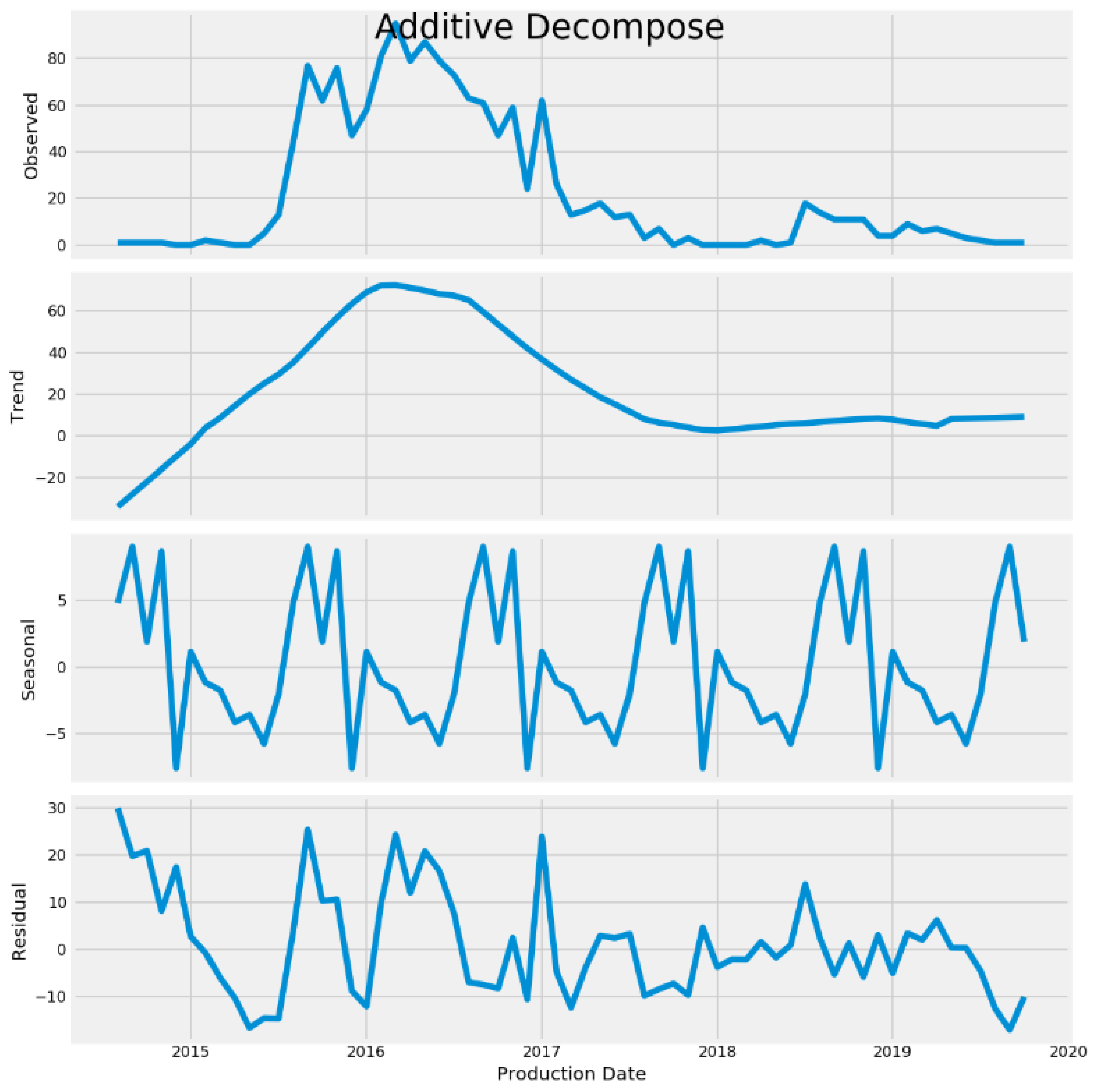


As can be seen, this data has more examples from 2016-17 and less example from other years and no data for 2014, 2013 or before. As it was sampled from whole data, all claims records of 2015 or 2018 might not even be there in our dataset, so it is hard to detect a pattern/trend without al data. The same problem occurs with the box plot when trying to find a trend & seasonality in data as below, these counts are not accurate values of monthly/yearly count.





Also, same thing happens when plotting additive decompose for time series to identify trend and seasonality which should be eliminated in order to make a time series stationary to further fit the model, which is shown in below graph, (It doesn't have a significant observed pattern due to the lack of data)



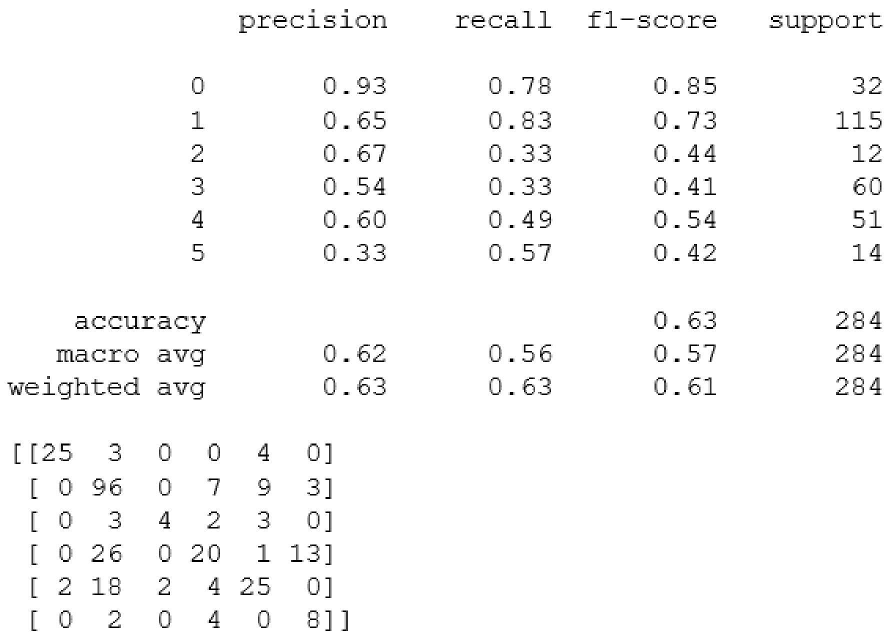
So, there is no result, analysis, evaluation/discussion or conclusion given in this regard as it was not in the original aim or deliverables of the project.

Results, Evaluation & Analysis

One of the aims was to provide ford users with DataRobot like functionalities which executes multiple algorithms when data file is provided and finds out the best model with highest accuracy. Also, configuration of hyperparameter tuning is to be added in the file too, similar to what users/engineers were doing through UI in DataRobot. To do this, Exploration one as stated earlier was used. So now the result is, user can provide data file to run different algorithms as well as do hyperparameter tuning for each of them from jupyter file. I have also explained how these model training and parameter tuning can be configured and how models can be changed to test the data on different models. So, users can easily replicate the scripts to apply it to other sectors at ford or scale it to larger amount of data even after this project is over. For all the approaches taken, instead of looking at just accuracy, alternative metrics like Precision, Recall and F1-score [4] are used to get more robust results for algorithm. Also, failure mode number 4 is nothing but the value “Other” as in the failure mode is not anything from 0, 1, 2, 3 and 5, so its accuracy is not given more important as such.

As for the multiclass classification models, the time taken to train models were low, but accuracy was also not that good. The classification report of the top 3 models are given below:

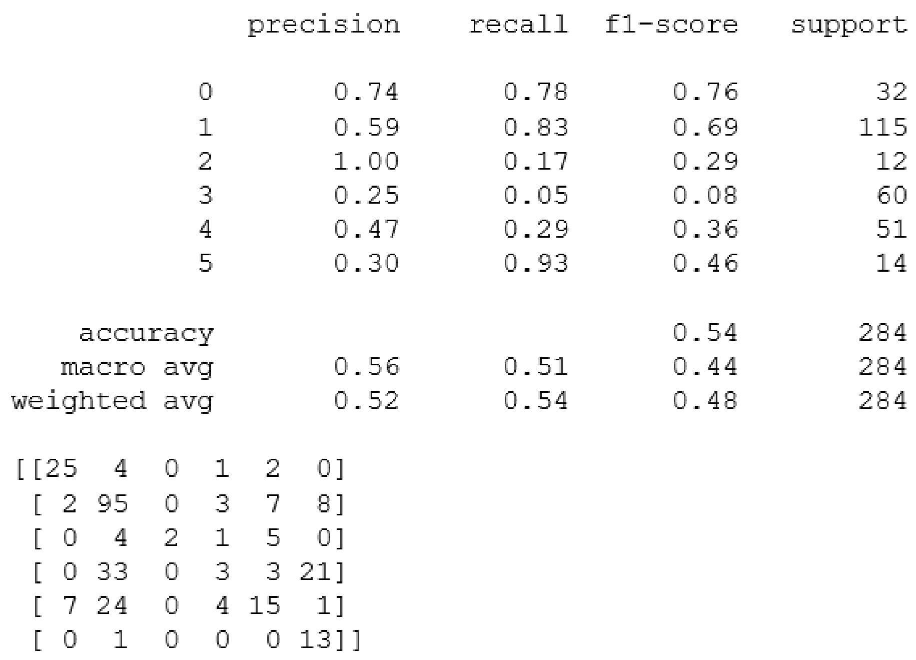
XGBoosting:



```
              precision    recall  f1-score   support

           0       0.93      0.78      0.85        32
           1       0.65      0.83      0.73       115
           2       0.67      0.33      0.44        12
           3       0.54      0.33      0.41        60
           4       0.60      0.49      0.54        51
           5       0.33      0.57      0.42        14

    accuracy                           0.63       284
   macro avg       0.62      0.56      0.57       284
weighted avg       0.63      0.63      0.61       284

[[25  3  0  0  4  0]
 [ 0 96  0  7  9  3]
 [ 0  3  4  2  3  0]
 [ 0 26  0 20  1 13]
 [ 2 18  2  4 25  0]
 [ 0  2  0  4  0  8]]
```

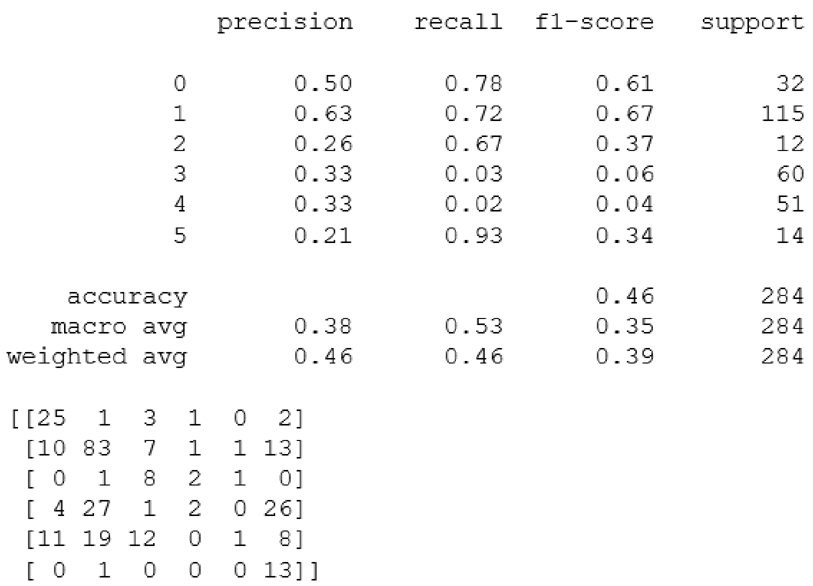
As can be seen from these metrics, XGBoosting has highest accuracy among all for multiclass classification. But it is not that good though, so I have leveraged the customer provided feedback as a feature and used NLP technique to extract information from it in the next exploration part.

SVM:



```
              precision    recall  f1-score   support

           0       0.74      0.78      0.76        32
           1       0.59      0.83      0.69       115
           2       1.00      0.17      0.29        12
           3       0.25      0.05      0.08        60
           4       0.47      0.29      0.36        51
           5       0.30      0.93      0.46        14

    accuracy                           0.54       284
   macro avg       0.56      0.51      0.44       284
weighted avg       0.52      0.54      0.48       284

[[25  4  0  1  2  0]
 [ 2 95  0  3  7  8]
 [ 0  4  2  1  5  0]
 [ 0 33  0  3  3 21]
 [ 7 24  0  4 15  1]
 [ 0  1  0  0  0 13]]
```

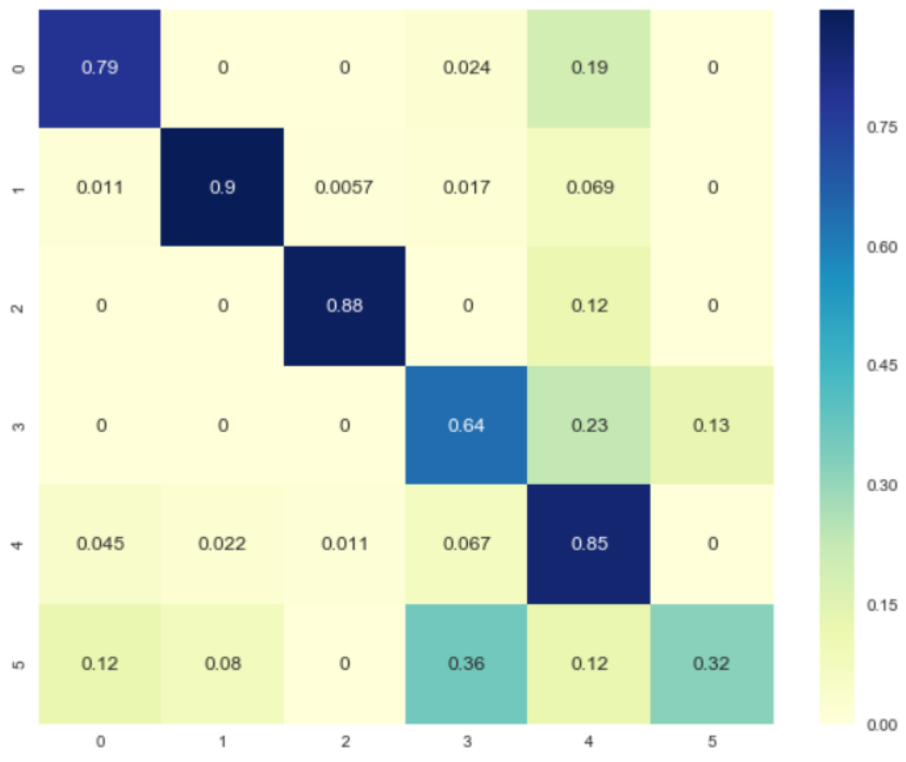
Naive Bayes: as can be seen, NB has low accuracy, it can be even worse as naïve byes works based on probability and it assumes conditional independence between features.



```
              precision    recall  f1-score   support

           0       0.50      0.78      0.61        32
           1       0.63      0.72      0.67       115
           2       0.26      0.67      0.37        12
           3       0.33      0.03      0.06        60
           4       0.33      0.02      0.04        51
           5       0.21      0.93      0.34        14

    accuracy                           0.46       284
   macro avg       0.38      0.53      0.35       284
weighted avg       0.46      0.46      0.39       284

[[25  1  3  1  0  2]
 [10 83  7  1  1 13]
 [ 0  1  8  2  1  0]
 [ 4 27  1  2  0 26]
 [11 19 12  0  1  8]
 [ 0  1  0  0  0 13]]
```

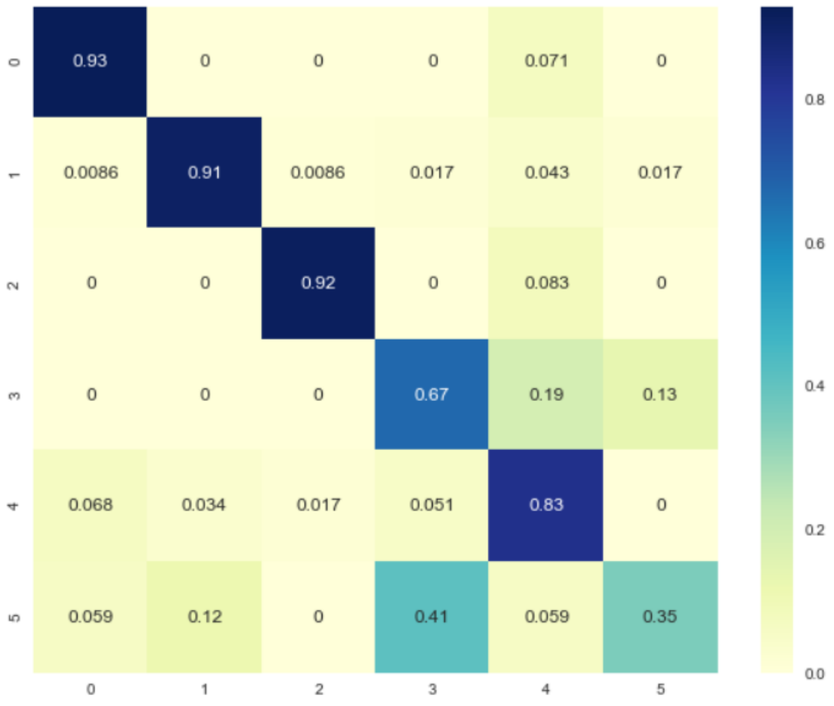
For the exploration 2, I got good results for all classifiers. Since **XGBoosting had highest accuracy** among all models that are used in exploration, I'll be showing the results of this algorithm only. Below are the screenshots for classification report, confusion metrics and visualisation of it. As can be seen, it achieves significantly higher accuracy than previous approach. As can be seen from the report, failure mode 3 and 5 have less accuracy compared to others. That is because both are almost similar defects and in a way sub defects of a defect type. For example, "outer lens fade" and "lens fade" defects are almost like each other and customer comments will also be about same for these defects so it will be hard to recognize between them. This is an intended behaviour to show that even though machine learning can be applied for identifying failure modes, human intervention is needed to further drill down it into different sub categories;

```
Accuracy = 0.796
----------------------------------------
Classification report:
              precision    recall  f1-score   support

           0       0.79      0.79      0.79        42
           1       0.98      0.90      0.93       175
           2       0.88      0.88      0.88        17
           3       0.72      0.64      0.68        78
           4       0.64      0.85      0.73        89
           5       0.44      0.32      0.37        25

    accuracy                           0.80       426
   macro avg       0.74      0.73      0.73       426
weighted avg       0.81      0.80      0.80       426

----------------------------------------
Confusion matrix
[[ 33   0   0   1   8   0]
 [  2 157   1   3  12   0]
 [  0   0  15   0   2   0]
 [  0   0   0  50  18  10]
 [  4   2   1   6  76   0]
 [  3   2   0   9   3   8]]
```



In our 3rd exploration, I have extended the previous model of approach 2 by adding all categorical as well as numeric feature to the model and trained it on the same data and the results are as shown below for this model,

```
Accuracy = 0.817
----------------------------------------
Classification report:
              precision    recall  f1-score   support

           0       0.81      0.93      0.87        28
           1       0.96      0.91      0.93       116
           2       0.85      0.92      0.88        12
           3       0.74      0.67      0.71        52
           4       0.72      0.83      0.77        59
           5       0.40      0.35      0.38        17

    accuracy                           0.82       284
   macro avg       0.75      0.77      0.76       284
weighted avg       0.82      0.82      0.82       284

----------------------------------------
Confusion matrix
[[ 26   0   0   0   2   0]
 [  1 105   1   2   5   2]
 [  0   0  11   0   1   0]
 [  0   0   0  35  10   7]
 [  4   2   1   3  49   0]
 [  1   2   0   7   1   6]]
```

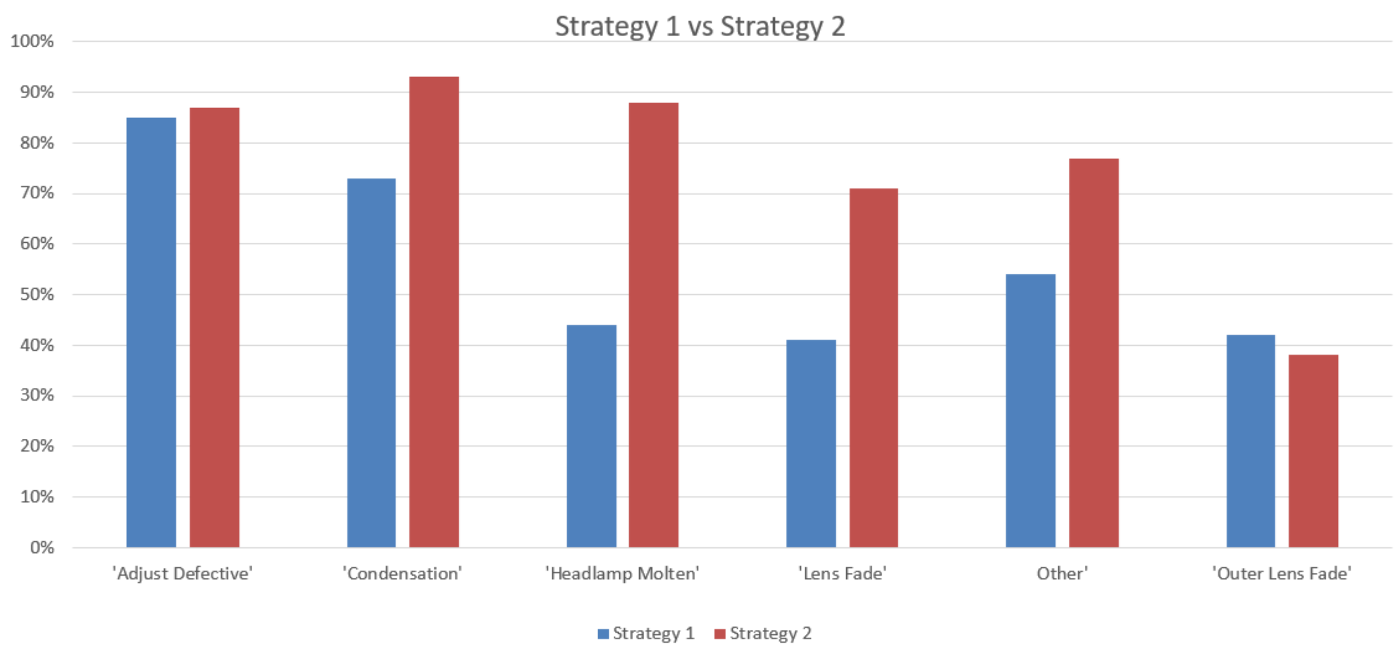


As stated earlier, model evaluation was done using not only accuracy but Precision, Recall as well as F1-score on a separate test data to get better accurate results and deal with class imbalance. Additionally, manual evaluation was also performed on another raw data. In that I have provided model predictions on other raw data to my manager and he randomly sampled some of the predictions and cross verified it with what should be the right failure mode for that claims data. It was observed that the model was correctly predicting failure modes approximately 8-9 times out of each 10 sampled records. So, our **ultimate judgement** was to use **XGBoosting with exploration 3** as our **final model**.

Next, I'll add original deliverables of this project and provide what was done in order to achieve it,

Deliverables

1) **Understand and demonstrate the difference of multiple classifiers (Multiclass) vs single classifier (Binary) in machine learning.**
    - Exploration 1 and Exploration 2 as discussed earlier was done in order to achieve it and explained the rationale and working behind it to manager in order to be able to replicate it by ford users for various other use cases. Also, I've made a bar chart to compare the accuracy of both, multiclass classifier and OneVsRest classifier; (Here OneVsRest classifier is in an essence binary classifier only, with two targets – one is failure mode itself and 2nd is rest all failure modes combined as Rest; which is repeated for all failure modes.)

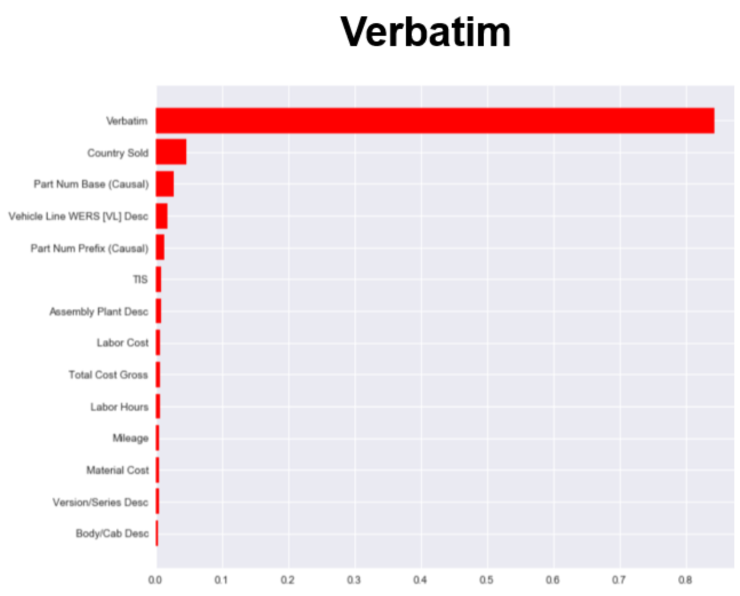


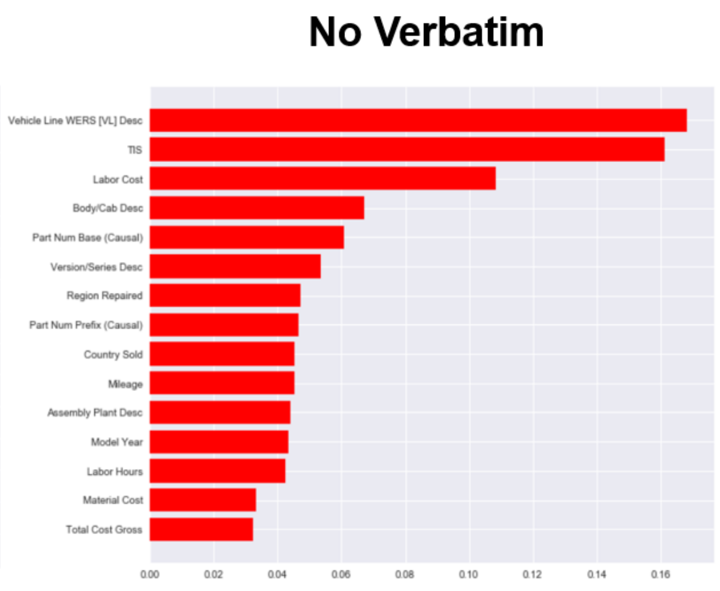
Here, strategy 1 is multiclass and strategy 2 is OneVsRest classification.

- **Analysis/Discussion:** It is obvious that 2nd approach has a higher accuracy than former one.

2) **How well can machine learning explain which attribute contributes most to certain key failure modes.**

- Since the XGB model was giving best accuracy among all, I have plotted a feature importance graph for both model when verbatim is used vs. when verbatim is not used while training a model, in order to give an understanding to higher management audience regarding which feature contributes more and which less. Below is a feature importance graph from the best models;



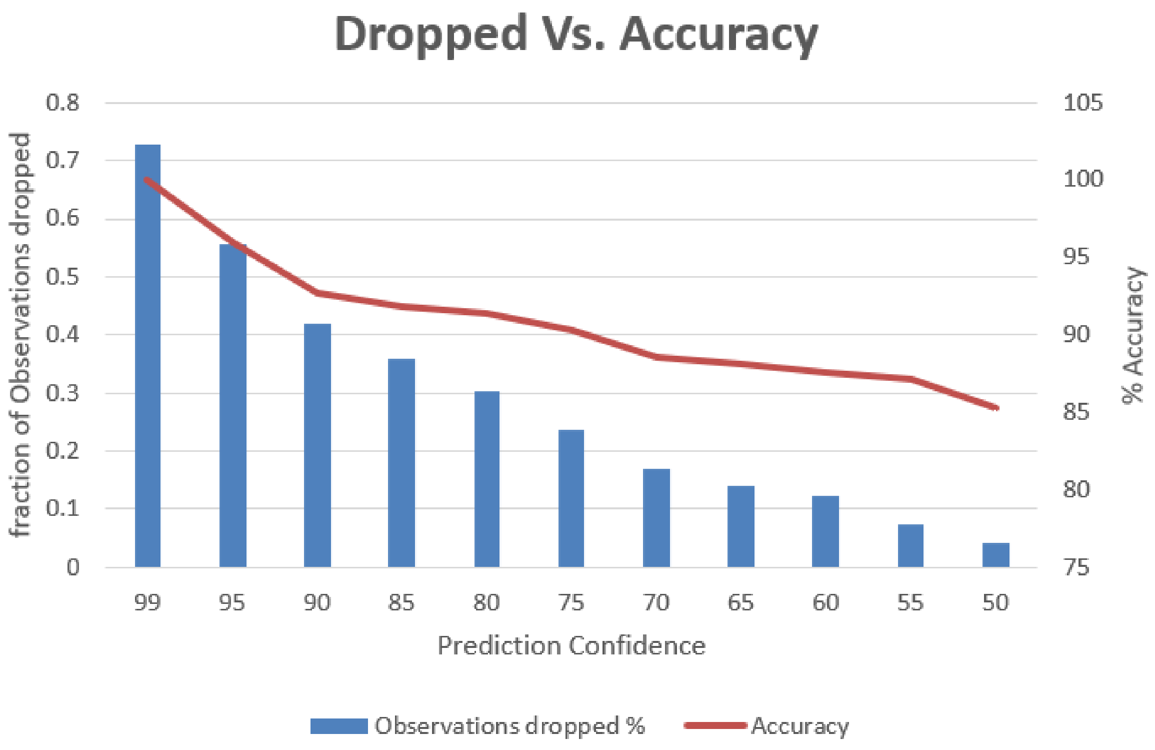


- **Analysis/Discussion:** It is obvious that 2nd approach has a higher accuracy than former one. Verbatim is most important feature when binning failure modes compared to other features.

3) **How well does machine learning predict the failure modes based on claims data**
   - As explained in the deliverable-1 bar chart, I have provided an individual accuracy of each of the failure mode to show how well our model predicts the failure modes based on claim data.
   - **Analysis/Discussion:** Works well on easily distinguishable failure modes. Struggling to correctly distinguish between similar failure modes.

4) **Which threshold to select for predictions with respect to confidence of correct prediction (e.g. prediction probability)**
   - This has to do with how much confidence does our mode predict each of the record of claims data, as we have used the OneVsRest classifier as our final model, it generates probabilities for each failure mode for each record and then failure mode with maximum probability out of all will be selected as predicted failure mode for that record. No by default, it selects the maximum probability out of all, but we can customize it to change the threshold like if probability is greater than 70% than select it as final prediction. To explain how changing the confidence threshold can change our model (i.e. improve accuracy but drops more values, which in turn requires more human efforts), I have set different threshold and plotted bar chart with line for accuracy as shown below;

- **Analysis/Discussion:** As can be seen from the above bar graph, when we improve the prediction confidence/probability threshold of our algorithm, it improves the accuracy because we are more confident in every failure mode as confidence threshold increases, but at the same time certain percentage of observations are dropped as they have less than threshold probabilities which needs to be processed by humans manually to assign a failure mode to them.



5) Recommendations how to improve the data set to improve machine learning for better prediction results

- As can be seen from the feature importance graph, verbatim is the most important feature when binning/classifying failure modes.
- **Analysis/Discussion:** So, one of the recommendations is to improve the quality of verbatim to more accurately predict & distinguish the failure modes. As not all features are used, I would recommend adding more vehicle information (features) to data and train it based on that to improve accuracy as certain feature values might have a correlation to certain failure modes, which will give more accurate predictions.

6) Recommendation if machine learning can be used as a substitute of a human being

- As stated earlier, it can easily distinguish different failure modes, but struggles when there are similar failure modes.
- **Analysis/Discussion:** So, I would recommend using machine learning to predict failure modes, but not as a substitute, instead, it can be used as pre-filtering for warranty claims binning which will reduce the manual work

significantly and then from there users can take manual approach on remaining data.

7) **Cost benefit of machine learning vs manual claims processing under consideration of machine learning maintenance, software cost, etc**
    - It was removed from original deliverables due to the data restrictions on manual claims processing cost.

Bonus deliverables

- **Implement machine learning code into data processing software to automatically process incoming new customers claims**
    - I've additionally created two separate scripts with respect to final model for model training and model prediction.
    - **Analysis/Discussion:** The model training file can be replicated to use different features and train model on different failure modes by ford users with minimal changes done. Model prediction file uses the final trained model, and this will be added into the Alteryx software used by ford, where it will predict on new incoming raw data and find failure modes and visualise those results in software itself.

- **Highlight trends or significant changes in newly processed data**
    - This part is bonus deliverables and explained in methodology section as analysis on time series data, which was just explored but could not be finished fully before the deadline of the project.

Recommendations from above results and analysis

- OneVsRest classification takes more time to train model but more accurate than multiclass classification in our case
- Verbatim is significantly important feature compared to others
- Confidence threshold can be set by ford as per the need or use case (how well the predicted failure modes need to be correct or with what confidence), but then as threshold is increased, accuracy vs. observations dropped trade off needed to be dealt with.
- Improving verbatim quality or adding additional vehicle information to dataset can increase accuracy of our model.
- As stated earlier, it works well in distinguishing failure modes except for when these failure modes are almost similar with similar description, so it should be used as a prefilter to reduce manual efforts as opposed to human substitution.

Self-reflection

- As stated earlier, working on real customer comments together with other numerical and categorical attributes for classification was a challenging part to solve for me as I have come across this type of situation first time. Applying NLP techniques separately to these verbatim comments and how to make the output of this as a feature for classification task was something to research upon for me.
- My idea of making a single pipeline to cater and process all the things in one single go to make it more automatic worked in providing outputs of verbatim processing as input to next stage of pipeline. The screenshot of the sample pipeline was provided in methodology section earlier. So, it went well and worked as expected. I will try to improve predictive power of the model by considering other algorithms like neural networks for text processing or find another way to deal with how two different failure modes of same category can be distinguish more accurately.

Conclusion

- Warranty claims data at ford is processed manually by user to bin them into different failure modes or defect, which takes a lot of time. So, the objective of the project was to identify how well machine learning can predict/bin failure modes and to automate this prediction of raw data and feeding it to their data processing and visualisation software. 3 different approaches are selected to be explored, Multiclass classification, one vs. other class classification (i.e. binary classification of each failure mode and combined single prediction) without NLP/text processing, one vs. other class with verbatim (essentially combined customer comments on defect for warranty claims data). By employing data science techniques of data pre-processing like scaling, removing null values, text processing of verbatim etc., the data was made suitable to train on machine learning models. Out of these three approaches; when taking into consideration all results, analysis, evaluation measures (accuracy, f1-score etc.) and by performing hyper parameter tuning on each model; it was found that 3rd approach of one vs. other class machine learning model with verbatim included, along with XGBoosting model, gives highest values for our evaluation metrics **(82% accuracy)**. Hence this was chosen as the best model with best hyper parameters suggested from grid search. Report justifies the techniques used in their respective sections. As there are data sharing restrictions, not all details on data are explained here, just the approach is explained. This accuracy was when two similar sub category of a main defect are included in data for training a machine learning model, this was to show that machine learning should be used as a prefilter to reduce manual work instead of a human substitution as these kinds of defect requires manual check by humans. While plotting a feature importance graph for our final model, it was found that verbatim is a most importance feature in prediction of warranty claims data. So, improving verbatim quality and adding more vehicles information (i.e. features) to dataset was recommended to improve the quality and accuracy of machine learning model.
- As stated earlier, this project was done on just the sample of failure modes, in future it can be scaled on a higher level to predict each failure mode of each category of defects.

Although, the final model training file was built in such a way to handle more failure modes as well as new features that are added to dataset.

- Another possible extension can be (not exactly in terms of machine learning), currently customer comments/verbatim entry into database is a manual process by ford users. Instead, some other techniques can be used like google speech to text functionality or combination of python packages like **SpeechRecognition**, **pyaudio etc.** to save verbatim directly which can reduce human efforts/manual intervention and automate the end to end process.

References

[1] https://scikit-learn.org/stable/modules/multiclass.html#

[2] https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html

[3] https://scikit-learn.org/stable/index.html

[4] https://scikit-learn.org/stable/modules/model_evaluation.html

[5] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html