

Project Report

Fangren Ji

Data Preprocessing – Cleaning Data

- The first step is to clean the data set due to data recording error.
 - Data definition says “repair_type – if a part was required, this is a hard repair; otherwise, a soft repair”. However, there are records in the data set where no part was sent on either initial visit or additional visit but the “repair_type” was labelled as “Hard”. For example, asset ID 7 is one of the examples.
 - Data points with this type of problem will be treated as potential outliers and removed before conducting data analysis to avoid misleading conclusion.

Asst_ID	Product_Type	...	Issue_Type	...	Parts_Sent	Repair_Type	...	Repeat_Parts_Sent
7	Desktops		Contract or Service Issue			Hard		

- There are data points in the data set where there was no additional visit but there was part sent out on additional visit. Asset ID 215 is one of them. This type of records will also be treated as potential outliers and removed.

Asst_ID	...	Repeat_CT	...	Repeat_Parts_Sent
215		0		Keyboard

- There are records where values for “agent_tenure_indays” are missing such as Asset ID 1682. Observations like this will be removed as well.


Asst_ID	Product_Type	...	Agent_Tenure_Indays
1682	Laptops		

Data Preprocessing – Filling Missing Values

- Next step is to fill in missing data points.

- For data points where there was no part sent on the initial visit but part(s) was/were sent on additional visit, the “repair_type” of “Hard” will still be legit and the “parts_sent” will be set to “None” for convenience. This also applies to records where “repair_type” is “Soft” since no part is required in this case.


Asst_ID	...	Parts_Sent	Repair_Type	...	Repeat_Parts_Sent
495			Hard		Cables, Adapter, AC



Asst_ID	...	Parts_Sent	Repair_Type	...	Repeat_Parts_Sent
495		None	Hard		Cables, Adapter, AC

- Column “repeat_parts_sent” has some blanks. It may mean there was no part needed for additional visit or simply there was no additional visit for this service ticket. These blanks will be filled in with keyword “None”.


Asst_ID	...	Repeat_CT	...	Repeat_Parts_Sent
0		0		
640		1		



Asst_ID	...	Repeat_CT	...	Repeat_Parts_Sent
0		0		None
640		1		None

- There are records in the dataset where “issue_type” and “topic_category” are blank such as Asset ID 0. This may not be a recording error. It may simply mean the reported issue did not fit in any category in the system. For the purpose of convenience the type of blanks will be filled in with keyword “Other”.

Asst_ID	Product_Type	...	Issue_Type	Topic_Category
0	Laptops			

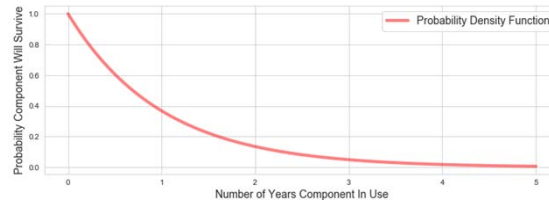


Asst_ID	Product_Type	...	Issue_Type	Topic_Category
0	Laptops		Other	Other

- Similarly, for other columns such as “product_type”, “region”, “country” and so on where there is a blank it will be replaced with “Other” as well.

Creating Additional Column

- Create an additional column in the dataset for later use, and name it as “duration_wk”. It represents the number of weeks between the contract start week and the customer contact week. This would give us a rough idea how long the device has been in use by the customer before a service call is made, especially when the lifetime of a hardware component usually follows exponential distribution in statistics. Exponential distribution states that the longer a hardware component is being used, the lower probability it will survive, or the more likely it will fail.



- Here is how to calculate additional column “duration_wk” by using information from Asset ID 0.

Asst_ID	...	Contract_ST	...	Contact_WK	Duration_WK
0		201726		201840	66

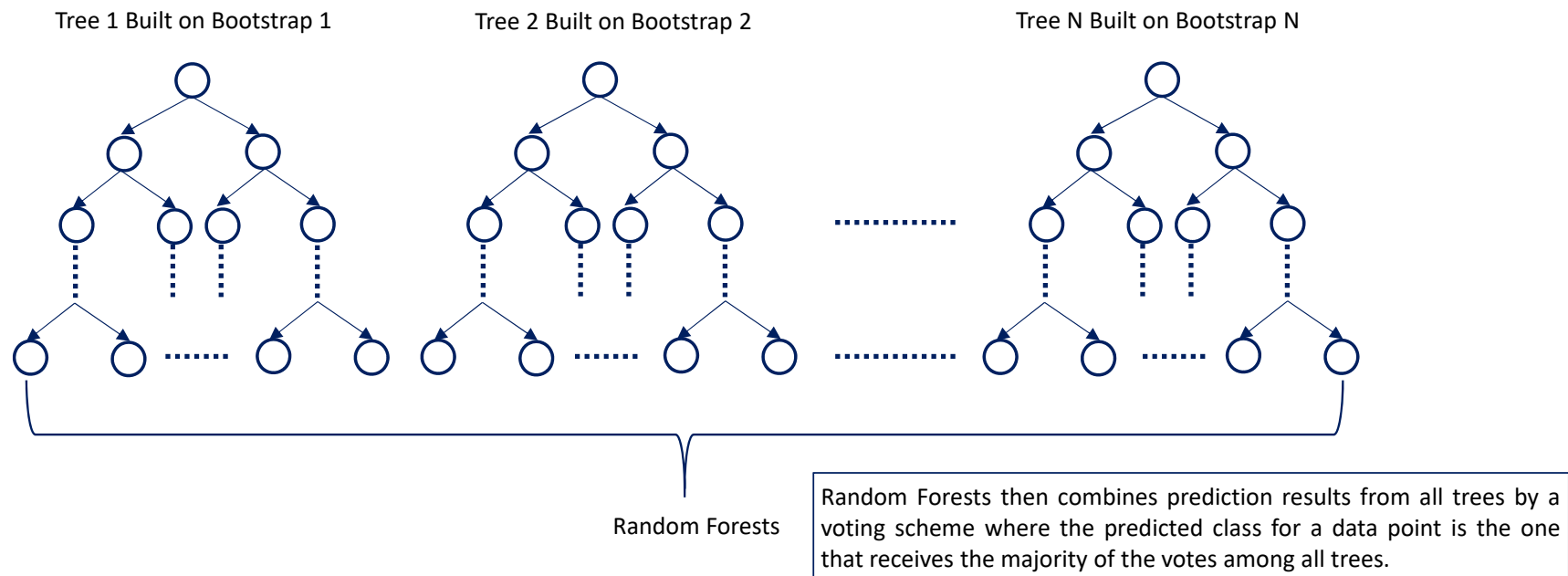
Week difference between last 2 digits: $40 - 26 = 14$
 Year difference between the first 4 digits: $2018 - 2017 = 1$
 Given 52 weeks per year, for Asset ID 0:
 $\text{duration_wk} = 1 \times 52 + 14 = 66$

- After creating this additional column it seems that there are more outliers in the data that need to be removed, because some records show service call was made before contract start week or even before device was manufactured with negative duration.

Asst_ID	Product_Type	Mnfture_WK	Contract_ST	...	Contact_WK
46	Laptops	201844	201844		201840
241	Laptops	201848	201848		201840

Classification – Introduction to Random Forests

- After cleaning out the data, it is time to apply machine learning and statistical methods to analyze the data.
- Since the majority of the columns are categorical variables, we first explore the data by Random Forests which is an ensemble classification algorithm.
- A bootstrap is a data sample created by randomly sampling the original data set with replacement, then used to build a classification tree or model. This process repeats among multiple bootstrap samples to create multiple classification trees that form Random Forests.



Classification – Applying Random Forests to Cleaned Data Set

- Based on subject-matter knowledge an asset ID will most likely make no contribution to predicting hardware failure. We first choose “repeat_ct” as dependent variable and the rest of the columns except “asst_id” as independent variables, meaning we want to predict “repeat_ct” based on the other features.
- We also split the cleaned data set into a training set of 70% and a test set of 30% of all the data points.
- “repeat_ct” has 4 categories namely 0, 1, 2, and 3. In initial run of deep learning we set the number of bootstrap samples as 1,000 and achieved a prediction accuracy of ~ 99.8%.

Accuracy of Predicting Number of Additional Visits Required to Fix The Problem: 0.9983111277913305

← Screenshot of Python program output

- We can also take a look at the first 10 observed “repeat_ct” values in the test set and compare them with those predicted by Random Forests model. They are exactly identical.

Point	1	2	3	4	5	6	7	8	9	10
Observed Repeat Count	0	0	0	0	0	0	1	0	0	1
Predicted Repeat Count	0	0	0	0	0	0	1	0	0	1

- Depending on the choice of response/dependent variable we can have multiple classification models with different accuracy. Next, let us try to predict the “parts_ct” using the same strategy.

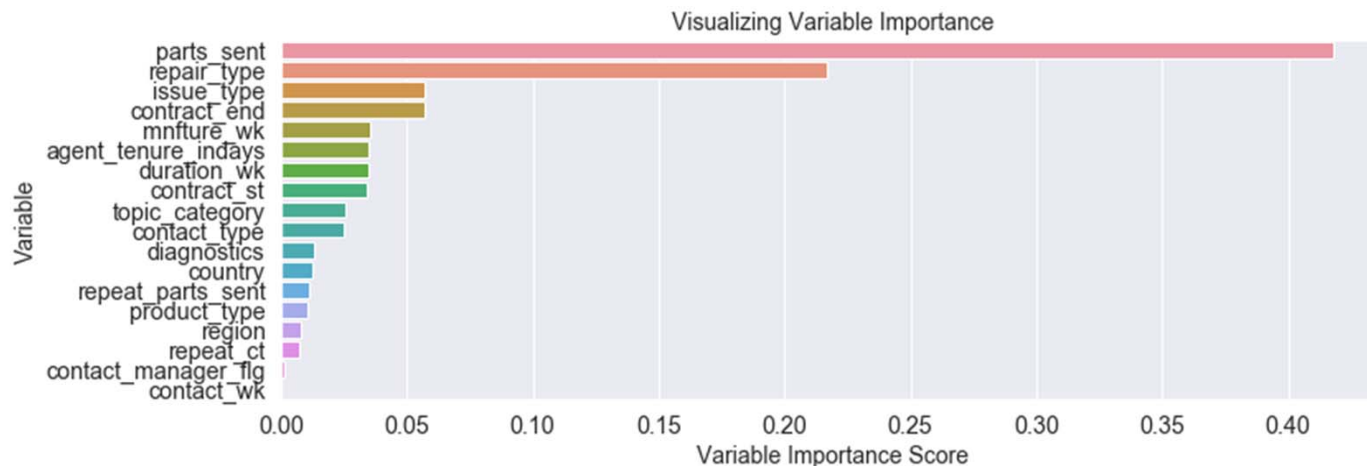
Predicting Parts Count

- We now use “parts_ct” as the response variable to be predicted and the rest of the columns as independent variables except “asst_id”.
- “parts_ct” takes 27 possible values between 0 and 51.
- With the same machine learning strategy used previously we now achieved an accuracy of ~92% in predicting the number of parts sent to fix the problem.

Accuracy of Predicting Number of Parts Sent to Fix The Problem: 0.9224620003753049

← Screenshot of Python program output

- Next let us explore the variable importance among all the independent variables we used to build this model and see if we can improve the prediction accuracy.



There are a few independent variables that have low importance scores, meaning they probably made low contribution in the training process. Sometimes these variables may bring noise to the model with negative influence. Let us remove them from the independent variable set and retrain the model to see if we can optimize it in the next slide. In the meantime, we notice that the newly created variable “duration_wk” seems moderately important.

Retraining Random Forests

- Let us removed the 4 least important variables observed from the variable importance bar plot in the previous slide, namely “contact_wk”, “contact_manager_flg”, “repeat_ct”, “region”, and retrain the model.
- After retraining the model we now achieved a prediction accuracy of also ~ 92%.

Accuracy of Predicting Number of Parts Sent to Fix The Problem: 0.9216738600112592



Screenshot of Python program output

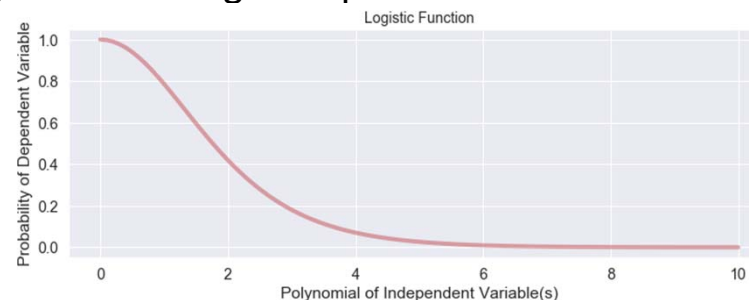
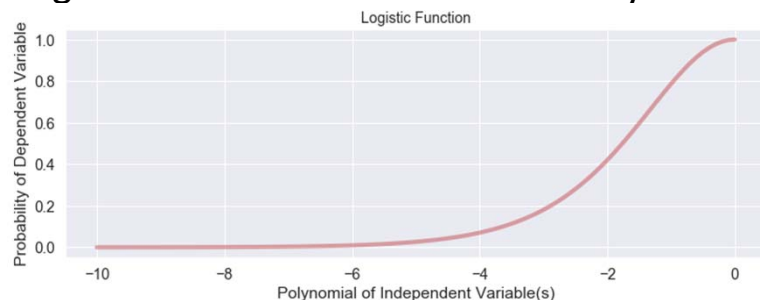
- Therefore, removing least important variables does not improve the model accuracy for this specific data.
- There are other ways to explore in order to improve the model accuracy. Also, there are some other variables we can use as response variable to be predicted by using Random Forests. But for now, let us move onto the next topic.

Introduction to Logistic Regression

- Given that the “contact_manager_flg” feature is a binary variable (0 or 1), we can explore a possible solution to predict whether a service call will raise a flag to bring in a manager to solve the problem under certain circumstance.
- A simple example can be, a tech support agent who has a smaller value in attribute “agent_tenure_indays” will more likely raise a contact manager flag than the others due to lack of experience.
- Logistic Regression is a type of GLM(Generalized Linear Model) that can make this type of binary prediction.
- Logistic function has the following format

$$Y = \frac{1}{1 - e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \dots + \beta_n X_n)}}, \text{ where } X_1, X_2, \dots, X_n \text{ are the independent variable and } \beta_0, \beta_1, \dots, \beta_n \text{ are the coefficients to be estimated.}$$

- Logistic function has a monotonically increasing or decreasing S shape.



- The predicted Y response is a probability between 0 and 1. But if we set a threshold value such as 0.5 then we can classify the predict probability into one of the two classes. For example,
 - If *predicted probability* > 0.5 → *Bring in a Manager* ; If *predicted probability* ≤ 0.5 → *No Need to Bring in a Manager*

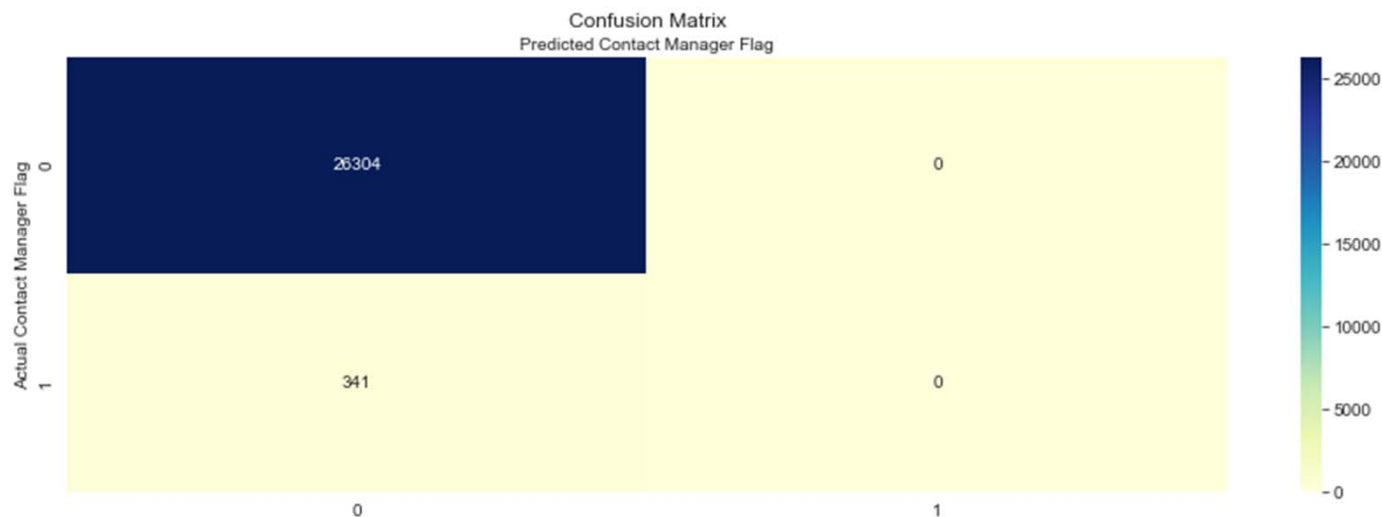
Logistic Regression Simulation

- We will use the same data split strategy as before setting aside 70% of the data points as training data set and 30% for the test data set.
- Once Logistic Regression model is built we use the model to make prediction on “contact_manager_flg” with the test data set to evaluate its accuracy. Here we achieved the accuracy of ~98.7%.

Accuracy of Predicting Contact Manager Flag: 0.9872021017076374

← Screenshot of Python program output

- We can also evaluate the accuracy in more detail by Confusion Matrix plotted in a Heatmap.



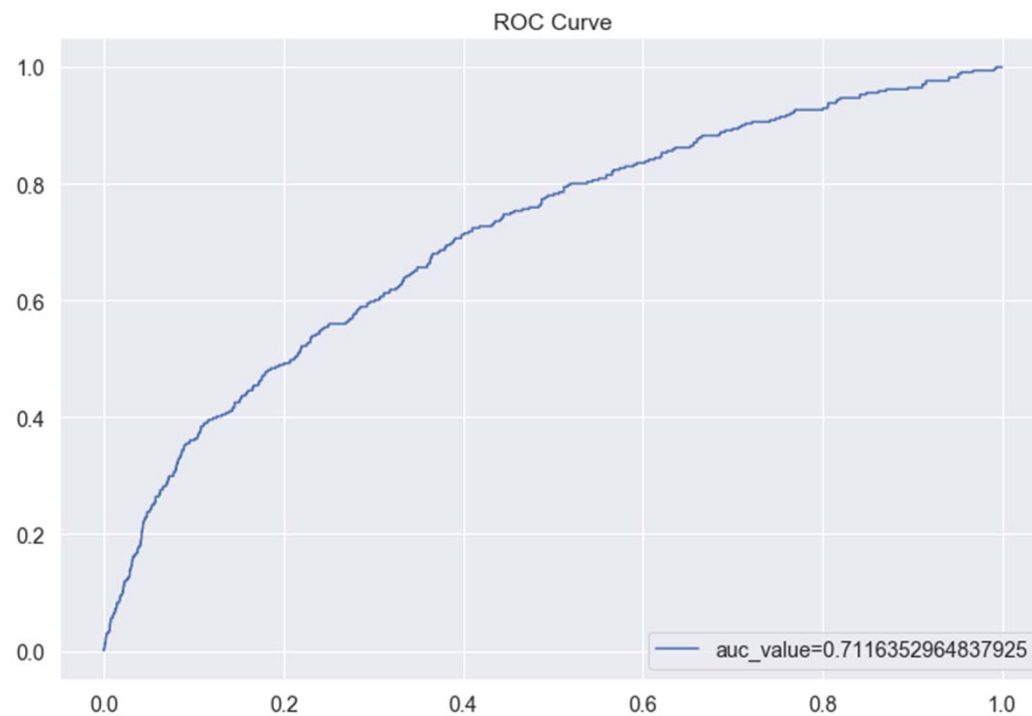
Diagonal numbers represent correct predictions while off-diagonal ones represent incorrect predictions. Our model tells us that 26,304 service call records were correctly predicted in terms of “contact_manager_flg” as response variable, while 341 records were incorrectly predicted. Also, this is based on the test data set we have.

ROC-AUC

- Since it is an imbalanced classification it is more appropriate to also look the AUC value and see how well the logistic regression model distinguished the 2 classes, 0 and 1.

The ROC-AUC value for Logistic Regression classifier is : 0.7116352964837925

- ROC-AUC Plot.



Thank You!