RollNumbers: 2019026, 2019051

# Assumptions

* The data does not have any outliers.

# The parameters used and the results for each of the question are shown.

# Preprocessing and data splitting

Preprocessing

* Checked for null values
* Removed duplicate values
* Not identified outliers

Splitting the dataset

* Random attribute set so that same results are displayed.
* Stratify is set to the label to keep the distribution equal as the original data set.

# (A) Training

# Q3. Varying hyperparameters

1. Criterion

The results obtained using Entropy are better than Gini. Entropy is a measure of information that indicates the disorder of the features with the target while the gini impurity measures the frequency at which any element of the dataset will be mislabelled when it is randomly labeled.

The entropy takes values between 0 and 1 while Gini is half of it: it takes value between 0 and 0.5

2. Splitter

Splitter='random' gives a degradation of result. The accuracy, precision and recall are less than the base model. This is because 'random' splitter uses a random uniform function while 'best' splitter evaluates all splits using the criterion before splitting.

3. Min_samples_split

It is the minimum number of samples required to split an internal node. We can see that all the measures are high when we set this to some value. This attribute can be used to prevent overfitting. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree.

4. Max depth

As seen from the above accuracy vs depth graph, the accuracy increases as the depth increases. This is because with increasing depth the model becomes more and more complex and captures more information about the data. However, with high depth the model tends to overfit and keeping the restricting the max_depth of the tree can help combat overfitting.

5. Min_Samples_Leaf

With this parameter there is an increase in the performance of the model. According to scikit-learn, we can use min_samples_leaf to ensure that multiple samples inform every decision in the tree, by controlling which splits will be considered.

6. Max Features

Max_features tells number of features to consider when looking for the best split. By default it looks at all the features for the split. However, here it looks at the limited features due to which we see a decreased performance scores in terms of accuracy, precision and recall. If we keep it default, it can lead to overfitting.

7. Class Weight

class_weight is used to provide a weight or bias for each output class. But what does this actually mean, see when the algorithm calculates the entropy or gini impurity to make the split at a node,

8. Max Leaf Nodes

It tells the maximum of leaf nodes that we want to have in the tree. The accuracy is similar to the base model but there are variations in the recall and precision. Since recall has decreased and precision has increased, this implies that there is an increase in the number of false negatives in the model.

The most increased accuracy as compared with the base model comes after varying 'class weight'. However, since we do not have to alter the class weight for any further question. We are going with the max_depth = 6 for the further question as this is the only hyperparameter apart from class weight giving greater accuracy than the base model.

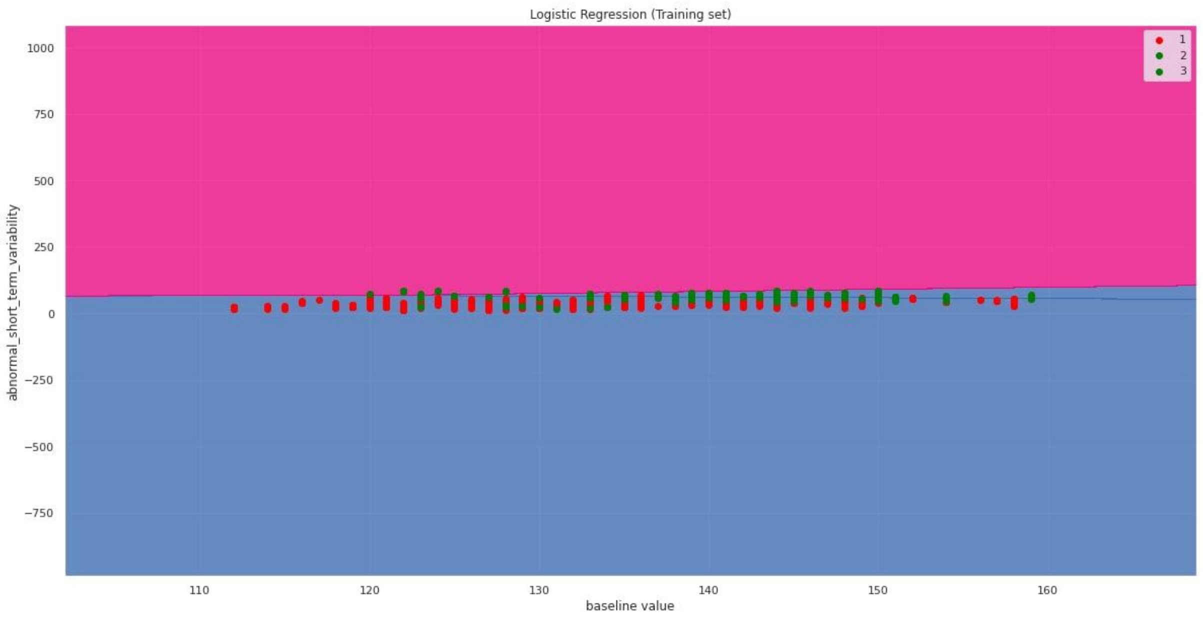
# (C) Experiments

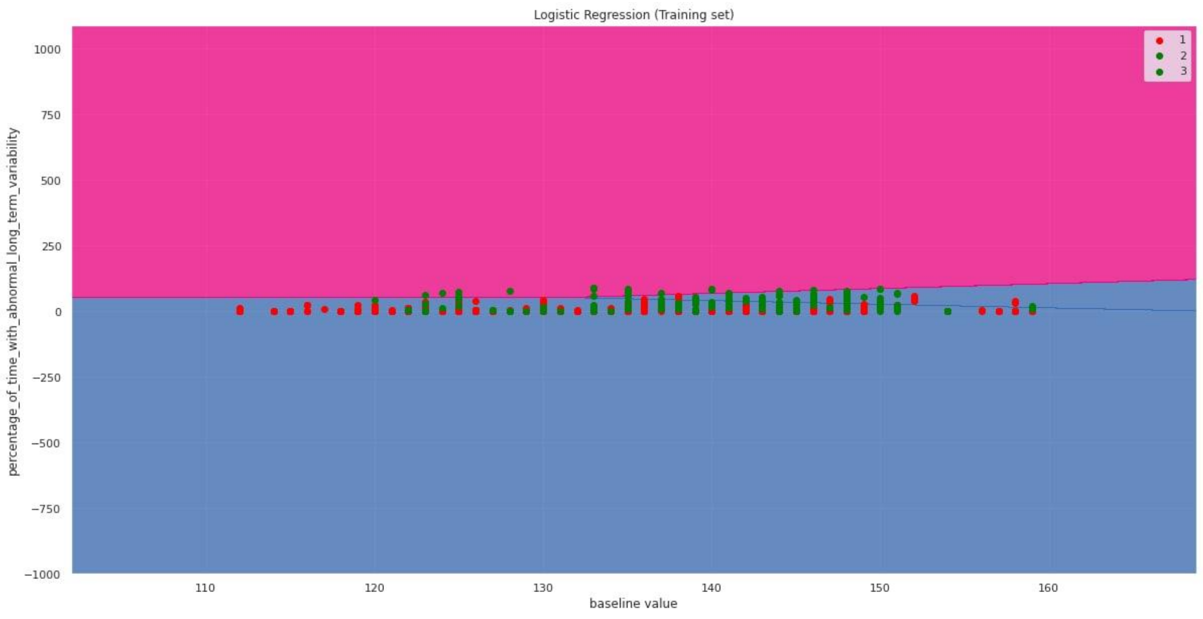
# Q1. Data Augmentation

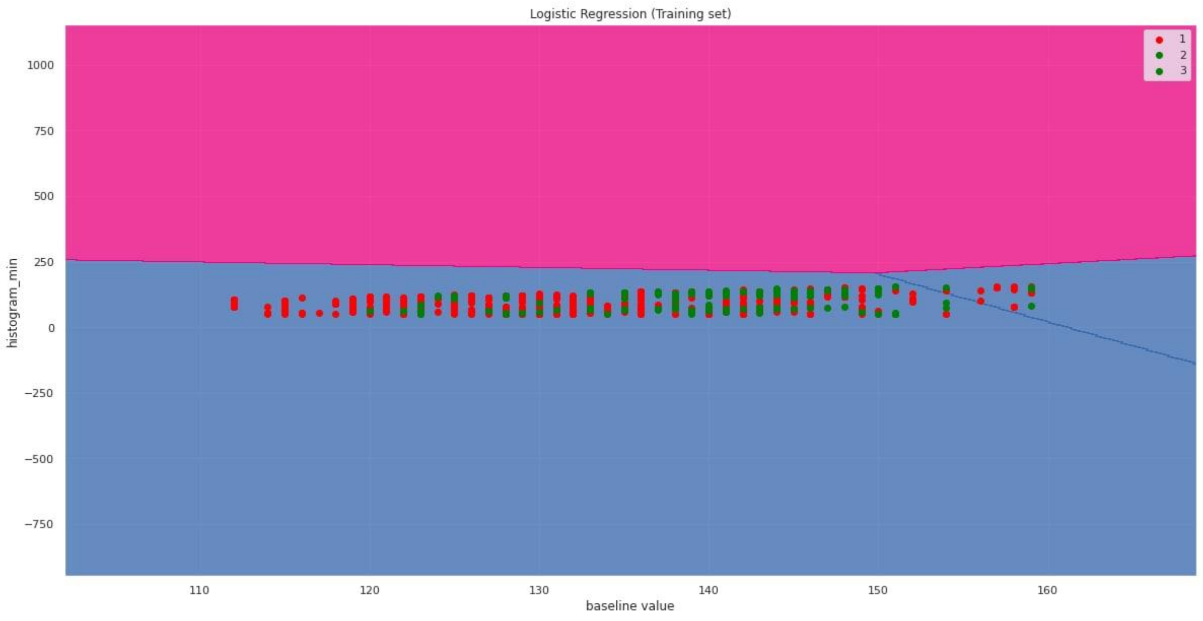
EFDT stands for Extremely Fast Decision Tree. We used this algorithm and skmultiflow.trees.ExtremelyFastDecisionTreeClassifier class to build the decision tree. It keeps on taking in data from a data stream and then modifies the decision tree accordingly rather than building it from scratch.

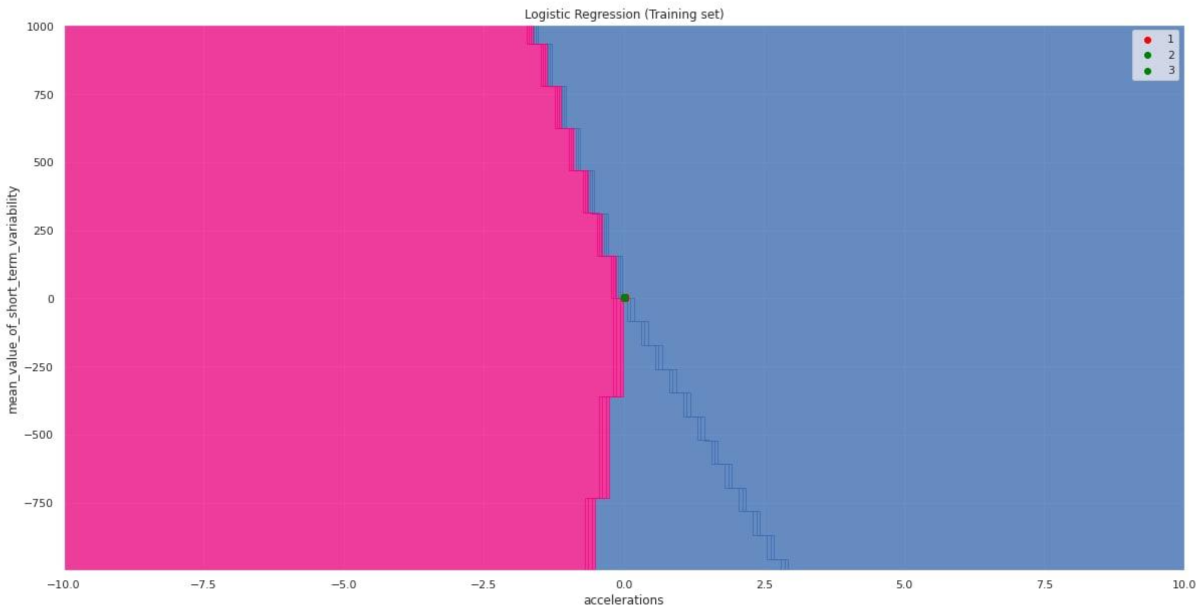
The performance as compared with the base model as shown in the output. The decision tree updates its knowledge as the new data comes in and the ROC curve has been plotted accordingly.

Q2.


Logistic Regression (Training set)
1
2
3
1000
750
500
250
0
−250
−500
−750
abnormal_short_term_variability
110
120
130
140
150
160
baseline value


Logistic Regression (Training set)
1
2
3
1000
750
500
250
0
−250
−500
−750
−1000
percentage_of_time_with_abnormal_long_term_variability
110
120
130
140
150
160
baseline value





# Learnings

* The assignment was highly challenging. However, it gave us the opportunity to explore multiple libraries like sklearn and skmultiflow.

* We also learnt about serialization through this assignment when we dumped the C1 model.

* It gave an insight to the iconsistencies a data may have and how we should handle them in order to reach to a correct result.

Sources Used

https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

https://towardsdatascience.com/data-pre-processing-techniques-you-should-know-8954662716d6

https://www.datacamp.com/community/tutorials/decision-tree-classification-python

https://mljar.com/blog/visualize-decision-tree/

https://quantdare.com/decision-trees-gini-vs-entropy/

https://towardsdatascience.com/finding-and-removing-duplicate-rows-in-pandas-dataframe-c6117668631f

https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/

https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/

https://github.com/doubleplusplus/incremental_decision_tree-CART-Random_Forest

https://towardsdatascience.com/hands-on-guide-to-plotting-a-decision-surface-for-ml-in-python-149710ee2a0e

https://www.analyticsvidhya.com/blog/2020/10/cost-complexity-pruning-decision-trees/

https://stackoverflow.com/questions/51397109/prune-unnecessary-leaves-in-sklearn-decisiontreeclassifier