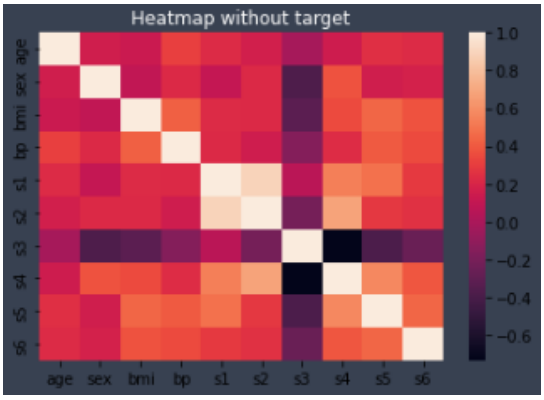
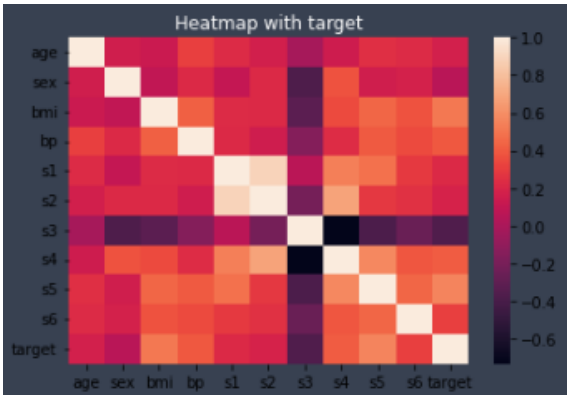
Programming Assignment 1 - Report

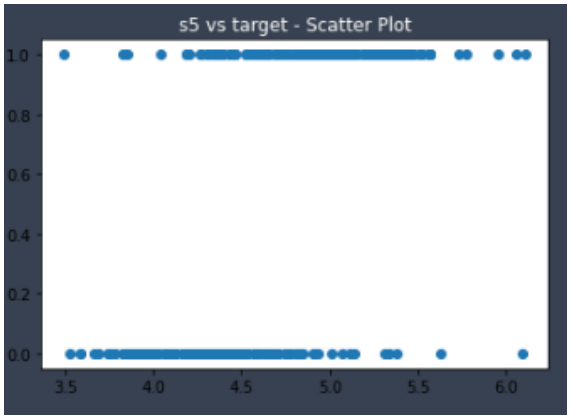
Blake Downey - 2127448 - April 10th, 2022

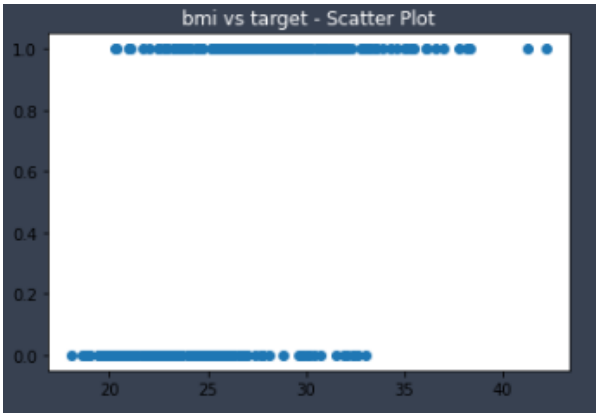
Answer 1)

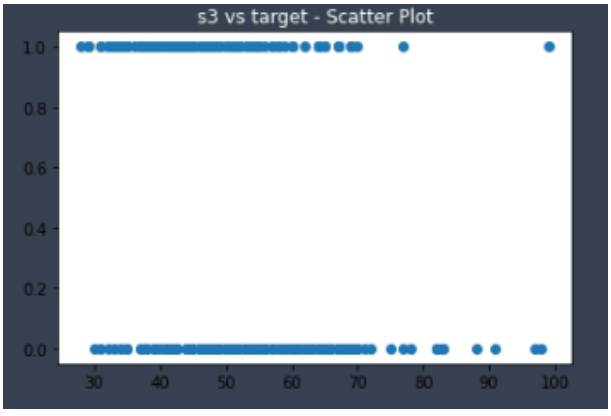


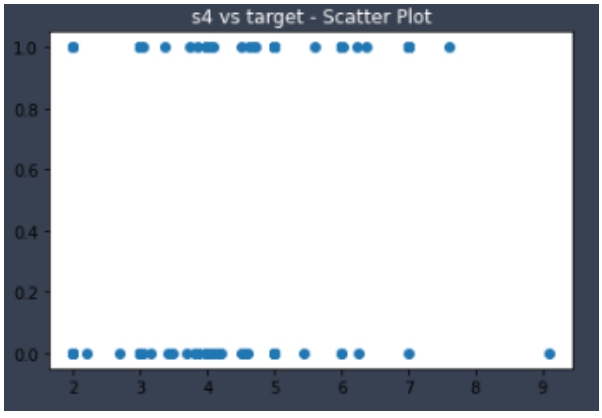


Heatmaps with features vs features and features vs target









Scatter Plots of top features by (color on the heatmaps) vs target

The heatmaps display the feature correlation to other features and the feature correlation between the features and the target. The most negative and the most positive values are the most correlated, represented by positive as white/light orange, and negative as dark purple. Simply from observation of the heatmap we can say that more correlated features are s3, s4, s5, and bmi. Using these I generated scatter plots shown above between these features and the target. In the scatter plots for s4, s5, and bmi we can see a pattern that for target equal to 0, we have left shifted feature values and for the target equal to 1 we have right shifted feature values. For s3 we can see a slight opposite shift, with target being 1 as a left shift. These left/right shifts indicate that there is a correlation between the feature and the target, because if there was no shift present, the feature would not be useful in determining prediction.

Answer 2)

	Precision	Recall	F1	Accuracy
LR	0.8333	0.8929	0.8621	0.8621
DT	0.7188	0.8214	0.7667	0.7586
Kaggle - LR	NA	NA	0.87356	NA

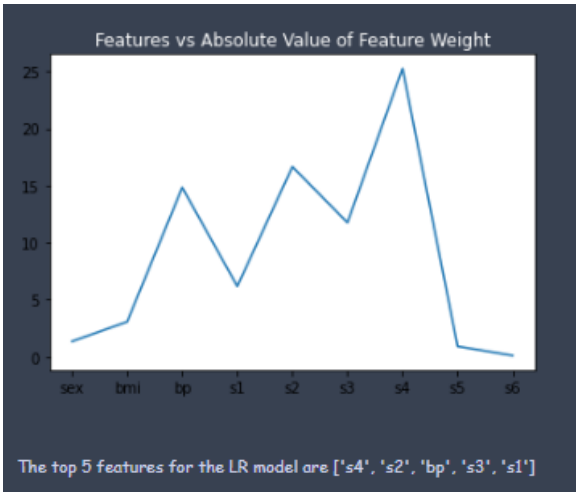
Table 1. Metrics for LR and DT

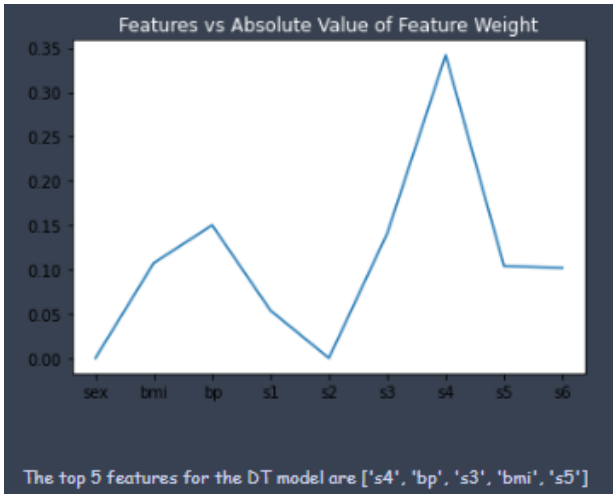
Hyper-parameter tuning was extensive on this dataset. In order to achieve the best results I performed a few different comprehensive searches for LR hyper-parameters: data split (random state seed for the train test split function), damping value, and class 1 threshold. Since the size of this dataset is relatively small, the LR and DT results varied heavily depending on the data split.

I ended with the following hyper-parameters for LR: l1_ratio=0.4, penalty='elasticnet', C=c, and solver='saga' (where c is passed in as 67). I chose to use elasticnet which comprises of both l1 and l2 regression with a l1 ratio of 0.4, meaning the regression is leaned towards l2. For this programming assignment I spent more time trying to optimize LR than DT and that is reflected in the metrics table above.

For DT, I found good results with the hyper-parameters: criterion='entropy', and max_depth=5. I found that when increasing the max_depth hyper-parameter to 6 or 7 the DT overfit to the training data, but when decreasing it lower than 5 the tree wasn't able to classify train or validation well.

Answer 3)





Graphs of feature importances for LR and DT models

The feature selection method that I used for LR was to get the coefficient values of the weights and take the absolute values of them to get all positive values as charted in the left plot above. By getting the coefficient values of the weights that represent the value of the weights, the greater the value the greater the feature importance. From there I found the top k features where k is 5.

For my DT feature selection method I used the function call feature_importances of the DT model. This method returns a list of values with equal length to the feature count where each index in the list corresponds to the importance of the feature, the higher the value the higher the importance. Since I chose to use entropy as my criterion in the creation of the DT, the feature importances returns the information gain for each feature as the list of importances.

As stated above the top features for the data I identified as s3, s4, s5, and bmi based on the heatmap. The top 5 features identified by information gain in my DT model are s4, bp, s3, bmi and s5. So, the features that I identified relate to the features that the information gain of the DT is providing as there is an overlap with 3 of the 4 features that I previously stated were top features by correlation.

Since the FeatureSelection method uses the model weights to calculate feature importance, yes the information gain methods used correlate with the model weights.

Answer 4)

For this I chose to split the dataset into its respective target classes, get the indices of each, and generate random indices based on those subsampled dataframe indices. By doing this I ensured that there were 3 class 1 and 3 class 0 samples that weren't repeated! I generated decision path sentence descriptions which can be found in the page below. These decision path descriptions were created based on the decision_path method of the DT model and the thresholds, feature splits, left and right children of the DT model.

Path for predicting sample 0 with dataframe index 124
Decision at node 0: split on s4 with threshold 0.155. The feature value at this split is 0.143, so the left path is taken.
Decision at node 1: split on s5 with threshold 0.018. The feature value at this split is 0.019, so the right path is taken.
Decision at node 9: split on bmi with threshold 0.004. The feature value at this split is 0.004, so the left path is taken.
Leaf node reached! The DT predicted 1.0 and the true value was 1.0 so it was a correct prediction!

Path for predicting sample 1 with dataframe index 182
Decision at node 0: split on s4 with threshold 0.155. The feature value at this split is 0.151, so the left path is taken.
Decision at node 1: split on s5 with threshold 0.018. The feature value at this split is 0.014, so the left path is taken.
Decision at node 2: split on bmi with threshold 0.008. The feature value at this split is 0.003, so the left path is taken.
Decision at node 3: split on s3 with threshold 0.525. The feature value at this split is 0.516, so the left path is taken.
Leaf node reached! The DT predicted 1.0 and the true value was 1.0 so it was a correct prediction!

Path for predicting sample 2 with dataframe index 164
Decision at node 0: split on s4 with threshold 0.155. The feature value at this split is 0.147, so the left path is taken.
Decision at node 1: split on s5 with threshold 0.018. The feature value at this split is 0.016, so the left path is taken.
Decision at node 2: split on bmi with threshold 0.008. The feature value at this split is 0.003, so the left path is taken.
Decision at node 3: split on s3 with threshold 0.525. The feature value at this split is 0.438, so the left path is taken.
Leaf node reached! The DT predicted 1.0 and the true value was 1.0 so it was a correct prediction!

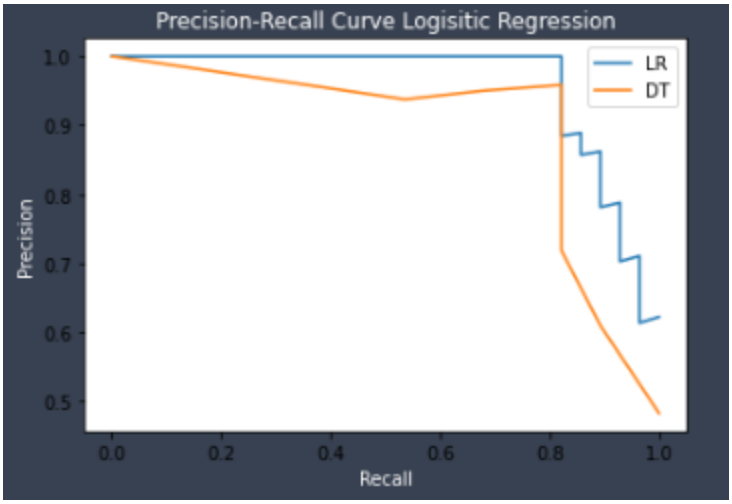
Path for predicting sample 3 with dataframe index 99
Decision at node 0: split on s4 with threshold 0.155. The feature value at this split is 0.247, so the right path is taken.
Decision at node 16: split on bp with threshold 0.099. The feature value at this split is 0.100, so the right path is taken.
Decision at node 30: split on bmi with threshold 0.004. The feature value at this split is 0.004, so the right path is taken.
Decision at node 36: split on s6 with threshold 0.022. The feature value at this split is 0.018, so the left path is taken.
Decision at node 37: split on s4 with threshold 0.193. The feature value at this split is 0.247, so the right path is taken.
Leaf node reached! The DT predicted 0.0 and the true value was 0.0 so it was a correct prediction!

Path for predicting sample 4 with dataframe index 204
Decision at node 0: split on s4 with threshold 0.155. The feature value at this split is 0.223, so the right path is taken.
Decision at node 16: split on bp with threshold 0.099. The feature value at this split is 0.090, so the left path is taken.
Decision at node 17: split on s4 with threshold 0.238. The feature value at this split is 0.223, so the left path is taken.
Decision at node 18: split on s5 with threshold 0.014. The feature value at this split is 0.010, so the left path is taken.
Decision at node 19: split on s3 with threshold 0.471. The feature value at this split is 0.403, so the left path is taken.
Leaf node reached! The DT predicted 1.0 and the true value was 0.0 so it was an incorrect prediction!

Path for predicting sample 5 with dataframe index 42
Decision at node 0: split on s4 with threshold 0.155. The feature value at this split is 0.172, so the right path is taken.
Decision at node 16: split on bp with threshold 0.099. The feature value at this split is 0.093, so the left path is taken.
Decision at node 17: split on s4 with threshold 0.238. The feature value at this split is 0.172, so the left path is taken.
Decision at node 18: split on s5 with threshold 0.014. The feature value at this split is 0.015, so the right path is taken.
Decision at node 22: split on s3 with threshold 0.490. The feature value at this split is 0.451, so the left path is taken.
Leaf node reached! The DT predicted 0.0 and the true value was 0.0 so it was a correct prediction!

Patterns that I see here (simply analyzing the above samples) are that when the ground truth value is class 0, the DT always starts down the right side of the tree. Whereas with ground truth value of class 1, the DT always starts down the left side of the tree. Another point of note is that the last feature split was often feature s3, whether it was a class 1 or class 0 ground truth.

Answer 5)



Precision-Recall Curve

As shown above, the precision-recall curve is quite jagged, but still downward trending. This graph also varied heavily with data split. Over the course of hyper-parameter tuning I generated many graphs that looked better and some that looked worse, but I was able to get better f1 scores with the data split I ended with and this is the resulting auc curve. To generate this I found a method under sklearn metrics for generating this exact curve and decided to use it for simplicity. According to this graph, the best results for precision/recall (where precision is 1.0 and recall is 0.821) can be identified at a threshold value equal to 0.506158 for LR. For DT precision and recall max intersection we can see that happens when recall is 0.821 and precision is 0.9583, this happens at threshold value of 0.6667.

To confirm this threshold value for LR I swept the thresholds and manually calculated the classes and generated the below curve. It's confirming that the best threshold is indeed ~0.5 for this particular tuning of the hyper parameters for the validation set f1 score. In the kaggle data I found that 0.435 as a threshold value I was able to achieve better results.