Report
Programming Assignment 3

Blake Downey - 2127448

Introduction

Programming assignment 3 involved a dataset of our choosing within the scope of illness prediction / diagnosis. The dataset had to be one that one was capable of achieving consistent high accuracy results using deep learning and not solely classical machine learning algorithms like decision trees or random forests. This assignment focused on the explainability of deep learning 'black box' models that are neural networks, and how using an explainable model, we can build a path to the solution and explain the result of the network to the patient.

Dataset

The dataset that I chose to use was one I found of kaggle, Breast Cancer Prediction Dataset [1]. This breast cancer dataset was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg, and it is in the format of a csv with 5 features and a target column with classes 0 and 1.

Problem 1: Explain It

In problem one we were to pick any public data set on cancer prediction with enough data. I found the dataset mentioned above to be sufficient for my needs. Using this dataset we were instructed to build a deep learning model with at least 3 hidden layers, build an explainable model to explain the predictions of the deep learning model, and then take 5 examples from each class and explain them.

For my deep learning model I opted for a simple Keras dense network as this data set had 5 features and I made the assumption that a small dense network would be sufficient for optimizing to great results, it was. My network takes the structure of an input layer, 3 hidden layers, and an output layer. All of the layers except the last have 3 neurons and ReLU activations, where the last has 1 output which is activated by a sigmoid function. For the objective loss function I am using binary cross entropy as this is a binary classification problem. I used Keras Model Checkpoint callbacks to save and then load the best model found during training, based on validation loss being lowest.

For the explainable model I chose a decision tree and trained it on the same X_train, y_train data split as the dense network. After varying the max_depth to find suitable results, the max f1 I could achieve with this DT was 0.9355 on the test set. This model did not perform as well on the validation set, coming in at 0.88 f1 score. A criteria for using this dataset for this programming assignment was the deep learning model had to

achieve better results than the explainable model could, and it does for both validation and train sets alike, this can be seen in the table below.

Split	Model	F1	Accuracy
Train	Dense Net	0.9347	0.9147
	Decision Tree	0.9892	0.9862
	Random Forest	0.9947	0.9931
Validation	Dense Net	**0.9649**	**0.9535**
	Decision Tree	0.8799	0.8605
	Random Forest	0.9038	0.8837
Test	Dense Net	**0.9538**	**0.9388**
	Decision Tree	0.9355	0.9184
	Random Forest	0.9333	0.9184

As for the explainability of the Decision Tree, I took 5 randomly selected class 0 samples and 5 randomly selected class 1 samples. Below is the step by step depiction of why the input sample produced the output prediction for a subsample of 4 samples. These generated sentences could be used or modified to explain to a patient why the prediction is the way it is. The decision tree does indeed give good explanations of the deep learning model, to the extent of lowered accuracy in the decision tree versus deep learning, which will present some discrepancies between the models predictions.

Path for predicting sample 0 with dataframe index 467
Decision at node 0: split on mean_perimeter with threshold 0.322. The feature value at this split is -1.273, so the left path is taken.
Decision at node 1: split on mean_perimeter with threshold -0.066. The feature value at this split is -1.273, so the left path is taken.
Decision at node 2: split on mean_texture with threshold 0.471. The feature value at this split is -0.277, so the left path is taken.
Decision at node 3: split on mean_smoothness with threshold 2.995. The feature value at this split is -0.943, so the left path is taken.
Decision at node 4: split on mean_area with threshold -0.640. The feature value at this split is -1.048, so the left path is taken.
Leaf node reached! The DT predicted 1 and the true value was 1 so it was a correct prediction!

Path for predicting sample 2 with dataframe index 299
Decision node 0: split on mean_perimeter with threshold 0.322. The feature value at this split is -1.035, so the left path is taken.
Decision node 1: split on mean_perimeter with threshold -0.066. The feature value at this split is -1.035, so the left path is taken.
Decision node 2: split on mean_texture with threshold 0.471. The feature value at this split is 0.884, so the right path is taken.
Decision node 12: split mean_smoothness with threshold -0.422. The feature value at this split is 0.366, so the right path is- taken.
Decision at node 18: split on mean_area with threshold -0.558. The feature value at this split is -0.912, so the left path is taken.
Decision at node 19: split on mean_texture with threshold 0.486. The feature value at this split is 0.884, so the right path is taken.
Leaf node reached! The DT predicted 1 and the true value was 1 so it was a correct prediction!

Path for predicting sample 5 with dataframe index 280
Decision at node 0: split on mean_perimeter with threshold 0.322. The feature value at this split is 1.410, so the right path is taken.
Decision at node 40: split on mean_texture with threshold -0.658. The feature value at this split is 1.701, so the right path is taken.

Decision at node 46: split mean_smoothness with threshold -1.541. The feature value at this split is 0.401, so the right path is taken.
Decision at node 50: split on mean_texture with threshold -0.212. The feature value at this split is 1.701, so the right path is taken.
Leaf node reached! The DT predicted 0 and the true value was 0 so it was a correct prediction!

Path for predicting sample 6 with dataframe index 321
Decision at node 0: split on mean_perimeter with threshold 0.322. The feature value at this split is 1.612, so the right path is taken.
Decision at node 40: split on mean_texture with threshold -0.658. The feature value at this split is 0.086, so the right path is taken.
Decision node 46: split mean_smoothness with threshold -1.541. The feature value at this split is -1.150, so the right path is taken.
Decision at node 50: split mean_texture with threshold -0.212. The feature value at this split is 0.086, so the right path is taken.
Leaf node reached! The DT predicted 0 and the true value was 0 so it was a correct prediction!

Problem 2: Random Forest

For problem two of this programming assignment, we were to train a random forest classifier on the dataset, compare the metrics of the deep learning model in problem 1 with the metrics of the random forest classifier, and use the randomly selected samples and the best decision tree in the random forest to describe the predictions.

For the Random Forest classifier, I utilized the sklearn RandomForestClassifier. After some light hyper-parameter tuning I came to settle on 25 estimators with a depth of 10 and min samples split of 3. From the metrics table in problem 1, we can note that the Random Forest performed better than the decision tree on validation but not as well as the dense network, and performed about the same as the decision tree for the test set as the number of samples were smaller for this set.

Metrics comparison of deep learning model and random forest model. The deep learning model was able to achieve 0.9649 f1 and 0.9535 accuracy on validation and 0.9538 f1 and 0.9388 accuracy on the test set. This is in comparison to the lower scores of the random forest classifier, with validation coming in at 0.9038 f1 and 0.8837 accuracy, and with test set coming in at 0.9333 f1 and 0.9184 accuracy.

In order to explain the prediction of the randomly selected samples, let's look at what a random forest is: a collection of decision trees. These trees are all different, and achieve different metrics across the data. To determine the best decision tree in the random forest, I made a function called get_best_tree_index() which takes in a random forest, and iterates the estimators of the random forest and tests the X_test dataset on each decision tree. The tree with the highest accuracy is then deemed the best tree and the index of that tree in the random forest is returned. In my code I have an example of this code running where the accuracy of the trees in the random forest are printed out to show the search for highest accuracy. After finding the tree with the best accuracy, I reused a chunk of code to generate the sentences from problem 1. Using this decision tree and the 5 samples from each class I generated the explanations of the predictions shown below.

Path for predicting sample 0 with dataframe index 467

Decision at node 0: split on mean_radius with threshold 0.121. The feature value at this split is -1.267, so the left path is taken.
Decision at node 1: split on mean_texture with threshold 0.471. The feature value at this split is -0.277, so the left path is taken.
Decision at node 2: split on mean_area with threshold -0.359. The feature value at this split is -1.048, so the left path is taken.
Leaf node reached! The DT predicted 1 and the true value was 1 so it was a correct prediction!

Path for predicting sample 2 with dataframe index 299
Decision at node 0: split on mean_radius with threshold 0.121. The feature value at this split is -1.027, so the left path is taken.
Decision at node 1: split on mean_texture with threshold 0.471. The feature value at this split is 0.884, so the right path is taken.
Decision node 15: split on mean_smoothness with threshold 0.487. The feature value at this split is 0.366, so the left path is taken.
Decision node 16: split mean_smoothness with threshold -0.422. The feature value at this split is 0.366, so the right path is taken.
Decision node 18: split mean_smoothness with threshold -0.346. The feature value at this split is 0.366, so the right path is taken.
Decision at node 20: split on mean_area with threshold -0.219. The feature value at this split is -0.912, so the left path is taken.
Decision at node 21: split on mean_texture with threshold 0.671. The feature value at this split is 0.884, so the right path is taken.
Leaf node reached! The DT predicted 1 and the true value was 1 so it was a correct prediction!

Path for predicting sample 5 with dataframe index 280
Decision at node 0: split on mean_radius with threshold 0.121. The feature value at this split is 1.429, so the right path is taken.
Decision at node 30: split on mean_radius with threshold 1.066. The feature value at this split is 1.429, so the right path is taken.
Leaf node reached! The DT predicted 0 and the true value was 0 so it was a correct prediction!

Path for predicting sample 6 with dataframe index 321
Decision at node 0: split on mean_radius with threshold 0.121. The feature value at this split is 1.713, so the right path is taken.
Decision at node 30: split on mean_radius with threshold 1.066. The feature value at this split is 1.713, so the right path is taken.
Leaf node reached! The DT predicted 0 and the true value was 0 so it was a correct prediction!

Something really interesting to note is that the samples did not change, but the features on which the samples were split on did change. For example, sample idx 467 is a class 1 prediction and ground truth 1 as well. In problem 1 it split on the features perimeter-> perimeter-> texture-> smoothness-> area. But in this decision tree the path was more concise and consisted of radius-> texture -> area. This means that the decision tree from problem 1 put the mean_perimeter as the most important feature for information gain, in comparison to the decision tree here, the most important feature is mean_radius.

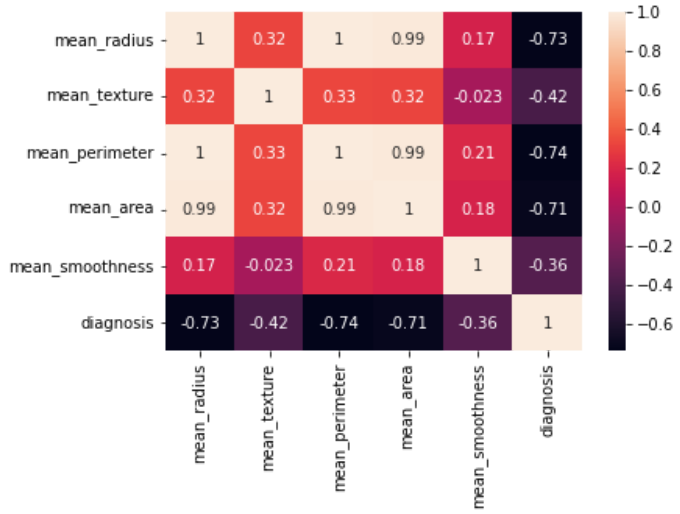
Problem 3: Feature Correlations

For problem 3 for assignment 3 we were to use the same dataset, and explain the random forest model using feature correlations, specifically identifying the top input features that correlate the most with the prediction for the 5 examples, and to compare this explanation to the explainable model.

feature	importance
mean_perimeter	0.3171
mean_radius	0.2159
mean_area	0.1932
mean_texture	0.1478
mean_smoothness	0.1260

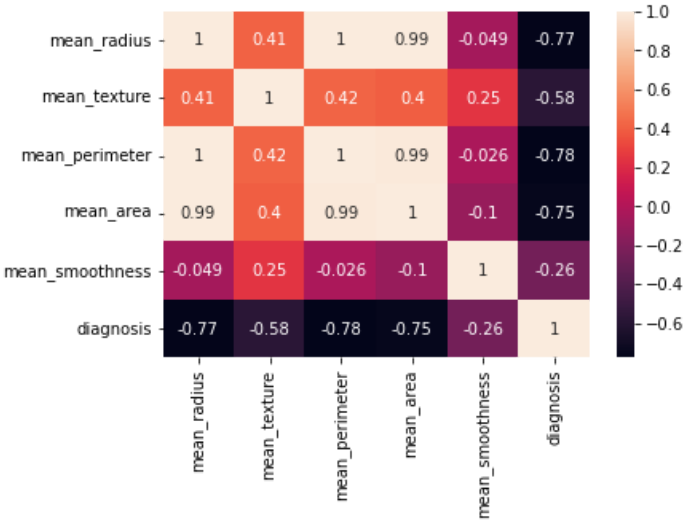
To extract the random forest models feature importance I used a simple bit of code to get the feature_importances_ from the random forest model and organize them by importance in a dataframe for easy viewing. On the left is the result of doing this, we can note that the most important feature on average is the mean_perimeter, with mean_radius

being the second most important. This is not feature correlation, but thought it was an interesting bit of code that showed feature importance, and was simple to implement. After implementing this and the correlation matrix I actually realized that even though these are not feature correlations, we can draw the same conclusions as we can from the correlations.



To dive into actual correlations, we can look at the feature correlations I performed at the start of this programming assignment. It is posted here for convenience. We can draw a conclusion off the bat from this correlation heatmap: mean perimeter and mean radius will be vital in the prediction of samples. I say this because the absolute value of correlation between these two features and the target ‘diagnosis’ is the highest of them all.

However, this correlation matrix is for all of the data. The question specifically states feature correlation for the 10 randomly selected samples, so let's see if it holds.



As we can see with the correlation heatmap here, we have the exact same order of feature correlations for this small sample size: perimeter -> radius -> area -> texture -> smoothness. To explain a prediction to a patient, one could explain the importance of the size (perimeter and radius/area) of the clump and how that correlates to a positive / negative sample. Here the best explanation from correlation would be to explain how the size has a negative correlation with the target, describe the size that the patient has, and then describe how that size fits into the correlation, either positively or negatively.

In comparison to the explainable model from problem 2, the top splits from the chosen decision tree were first mean radius, and then left was surprisingly mean texture, and right was again mean radius. This is giving high priority to mean radius, as expected, but also to texture which was not expected. Given this was just one of the 25 trees, I'm sure others had varying splits as well. Being that the top two feature correlations were perimeter and radius (with such a small difference between them), we can expect that one of these two features end up on the top of the decision trees as it did in this example from problem 2. To further this point actually, we can look at the explainable model from problem 1, and we see the top feature to split on was indeed mean perimeter, so it's a possibility that this decision tree from problem 2 didn't have perimeter as a feature in the sample dataset after bagging / feature sampling. We can also note an interesting point that mean texture shows up high in the feature splits as well for problem 1 decision tree. So in terms of correlation and information gain, it's insignificant, but still shows up frequently in the trees, maybe as a good way to weed out a few examples early on that are edge cases!

References

[1] https://www.kaggle.com/datasets/merishnasuwal/breast-cancer-prediction-dataset
[2] How to Read a Correlation Matrix - Statology