1 Comparison between some models

Answer the following questions

1.1 (10 Points)

What are the advantages of Random Forests over decision trees?

Random forest reduces the risk of overfitting by reducing the variance of the final model without massively increasing error due to bias. So the accuracy of the model would be higher and it would have higher performance.
Moreover random forests normally perform better when the target we want to predict is continuous such as regression problems. Additionally, random forest's predictions are more stable than decision trees since they use an average prediction of many decision trees for the final model prediction. So overall Random forests combine the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy.

1.2 (10 Points)

What are the advantages of XGBoost over random forest, adaboost and gradient boosting?

XGBoost always gives more weight to functional space when reducing the cost of a model, whereas random forest gives more weight to hyperparameters in order to optimize the model. Moreover XGBoost straight away prunes the tree with a score called similarity score. XGBoost performs well on unbalanced datasets forexample in applications like fraud detection. Also we can have parallel processing in XGBoost which we can't have in adaboost or gradient boosting since they are sequential algorithms.
Overall XGBoost is an algorithm widely used in different competitions like kaggle and has proved to be really advantageous over other algorithms.

2 Fitting a model

In this question, we want to fit a model for diabetes according to a given data.

2.1 (30 Points)

How Random Forest and AdaBoost fit a model to the following data?

Blocked Arteries	Weight	Exercise	Genetics	Has Diabetes
No	210	Yes	Yes	No
No	125	No	No	No
Yes	180	Yes	Yes	Yes
Yes	167	No	Yes	Yes

The random forest algorithm is an extension of the bagging method in which we use bootstrap sampling and create some bootstrap samples and try to fit one decision tree to each bootstrap sample. At each step we only consider a random subset of features and we find the best split among these features. By using feature randomness we ensure low correlation between different decision trees. At the end we use an aggregation method (majority voting for classification or averaging for regression) to predict the final output of the random forest model.
In this example we have 4 features and we would like to predict whether the patient has diabetes or not so this problem is binary classification here. In the process of constructing the random forest, we take bootstrap samples for each tree we would like to fit the data, and we select random features at each split and we try to fit the tree to it. We do the same for the number of trees specified as a hyperparameter to the model. The optimal hyperparameters of random forest like the number of trees or the number of features taken randomly in each bootstrap dataset can be selected by using out-of-bag samples error. Out-of-bag samples are simply the samples which are not used in any decision trees of random forest. It is proved that for large datasets typically about 1/3 of the original data does not end up in a bootstrapped dataset. So we use these data samples as a validation set for each dataset and we seek the best hyperparameters. Finally when we want to predict the output of a test sample we give this sample to each decision tree and either we use hard voting or soft voting to declare the target value of this test sample.

The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. Because these trees are so short and only contain one decision for classification, they are often called decision stumps. In Adaboost we assign a weight to each sample (which is 1/N at the beggining) and at each step we try to fit the samples to a weak learner which here is a stump and we update the weights of the samples (if the sample is wrongly predicted we increase its weight and if not we reduce its weight).
We do this process for like a limited number of K or we even may stop earlier if the error of weak learner gets higher than 1/2 which is the expected error of just a random model.
With the provided data forexample, the first=weak learner would be as following:
First Weak learner: First the weights should be normalised (they are already). Then the data should be fitted into a stump by choosing the best feature to split the data based on it. This feature could be selected by forexample the best gini index we get for each split. It is obvious that spliting the data based on the "Blocked Arteries" feature is the best option here as the gini index in the result groups would be both zero. Hence the error would be zero and would stop the iteration. So the final model would be just a single stump which predicts the target just based on "Blocked Arteries" feature which is not good at all!

2.2 (30 Points)

Now suppose that two entries of the data table("Blocked Arteries" and "Weight" of the 4^{th} data) are missing. Write step by step how random forest can fill the missing data?

First the missing values are filled with some initial guess that could be bad and then we gradually refine the guess until it is a good guess. For the initial guess for "Blocked Arteries" we look at the samples which has diabetes just like the fourth data sample and we consider the most common value for "Blocked Arteries" in those samples as the initial guess. Here there is only the third sample with diabetes and this data has "Blocked Arteries" so we set initial guess for "Blocked Arteries" of the fourth data sample to "Yes". We do the same for "Weight" missing value except that this time we take median of the "Weights" of the samples with diabetes, so the initial guess for "Weight" for the fourth data sample would be simply 180.
Then we build a random forest based on the completed data samples and for each tree we compute the similarity of all samples with respect to other samples and we store the results in a similiarity matrix. We call two samples similar in a decision tree if we end up in the same leaf node for two samples when we run these samples on a decision tree.
Every time two samples like s1 and s2 are similar we add one to similarity_matrix[s1,s2] and similarity_matrix[s2,s1] and in the end we devide the values of the matrix by the number of decision trees.
For example imagine after running the data on all the decision trees the similarity matrix would be like following:

	1	2	3	4
1		0.2	0.1	0.2
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.2	0.1	0.8	

Now use these similarity values for sample 4 to make better guesses about the missing data and we update the missing values. For example we update the "Blocked Arteries" value with:

	1	2	3	4
1		0.2	0.1	0.2
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.2	0.1	0.8	

Weighted frequency for "Yes" = $\frac{1}{3} \times$ similarity(4,3) = 0.26

Weighted frequency for "No" = $\frac{2}{3} \times$(similarity(4,2) + similarity(4,1)) = 0.2

⇒ Value "Blocked Arteries" <= "Yes"

Then we do the same for "Weight":

Weighted average = $180 \times$ similarity(4,3) + $125 \times$ similarity(4,2) + $210 \times$ similarity(4,1)

= 198.5

So we revised our guesses for a little bit and we need to repeat the process many more times. Again we need to build a random forest and run the data through the trees, recalculate the similarities and the missing values. We do this until the missing values converge (the missing values don't change for a specific rounds of repeating the algorithm).

3 Gradient Boost (20 Points)

Write step by step how Gradient Boost fits a model to the following data?

Gender	Height (m)	Blood Pressure	Weight (kg)
Female	1.6	Medium	76
Male	1.6	Low	88
Female	1.5	Low	56
Female	1.4	Low	57
Male	1.8	High	73
Male	1.5	Medium	77

First the average of the target is computed as the base first model h0. Then the residuals are computed and a fixed sized decision tree is fitted into these residuals. At each leave of the tree the average of the values of weights is computed and we would have for h1 the following formula:
h1(x_i)=h0(x_i)+alpha * y1_i
We follow the same algorithm but this time residuals of "Weight"s with respect to h1 as the target feature and after T times of iteration we would compute all the yt_j and combine trees for the prediction and we would have
h0(x)+alpha*y1_j+alpha*y2_j+...+alpha*yT_j as the prediction output.
alpha is the learning rate.

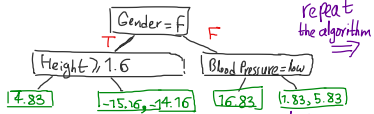
Note that the trees at each step are built greedy meaning that we fit the best decision result tree that we can to each residuals. This best tree is the tree which at each split the result split group would have more similarity which similarity is defined as:
sum(w_left)^2/count(node_left) + sum(w_right)^2/count(node_right)

For a better insight we go through some first steps of this algorithm:

average "weight" = 71.16

residuals: 4.83, 16.83, -15.16, -14.16, 1.83, 5.83

⇒ fit the limited decision tree



repeat the algorithm ⇒

new_residuals
76 - (71.16 + (0.1 × 4.83))
88 - (71.16 + (0.1 × 16.83))
56 - (71.16 + (0.1 × (-14.66)))
57 - (71.16 + (0.1 × (-14.66)))
73 - (71.16 + (0.1 × 3))
77 - (71.16 + (0.1 × 3))
...

For example

$\alpha = 0.1 \Rightarrow$ Prediction (sample 3) = $71.16 + \alpha \times (-14.66) = 69.69$

4 Descriptive Questions (35 Points)

Give a brief explanation for the following statements.

4.1 (5 Points)

Explain the Exploding Gradient problem. Propose a solution to prevent this issue.

In the training process of a neural network when we want to compute the gradient of loss function with respect to the weights we backpropagate and we use chain rule to compute each weights derivative. According to chain rule we multiply many partial derivatives to get the desired derivative and when these partial derivatives get high and accumulate this results in a very large update to neural networks model during training and this makes the model unstable and unable to learn from the data. In worst case, the value of gradients of weights can become so large that it might overflow and the result would ba NaN values.
One solution to this problem could be using gradient clipping in which bounds the gradient which we want to update between a specific range.

4.2 (5 Points)

What are the Pros. and Cons. of adding more layers to a deep neural network?

One of the advantages of adding more layers to a neural network is that it increases the model's complexity so we might be able to predict more complex functions. However, adding more layers without a large training set could lead to overfitting and hence it decreases the model's accuracy. Moreover, when adding more layers, the number of weights for which we should compute the gradient in each iteration rises and so the required time for training would increase. Additionally, if we increase the number of layers in a deep neural network, there is a good chance problems like Vanishing or Exploding gradient problem might occur.

4.3 (5 Points)

In Stochastic Gradient Descent algorithm , why does the first step of the procedure require shuffling the training dataset?

The main reason for shuffling the training data before starting the training process of the neural network is that there could be some cyclic patterns in data and this could lead the model to learn in a biased way. Forexample imagine we want to use a neural network to classify some pictures into dogs and cats. Assume that we have 1000 pictures and the first 500 pictures are all cats and the rest are dogs. Imagine we use Mini batch stochastic gradient descent without shuffling and batches have the size of 500. So in each epoch we would have two iterations and in the first iteration the weights are updated just by cat pictures and in the second iteration the weights are updated respected to just dog pictures. This makes the neural network to traverse the search space in a single direction and restricts the possible states that our model could achieve. So the neural network's loss function might end up in a local optima and not the global optima.

4.4 (5 Points)

Consider Sigmoid activation function. What will be the gradient value of this function for a very large input? What problem does this cause for neural network training? How could this be solved?

The gradient value of Sigmoid function for a very large input is 0 and this could lead to Vanishing gradient problem when training the neural network model specially in a deep neural network with many layers. The simplest solution is to use other activation functions such as ReLU or Leaky ReLU which only saturate on one direction and thus are more resilient to the vanishing of gradients. Another solution is to use ResNets.

4.5 (10 Points)

Consider a Multilayer Perceptron which is used to provide a 2-class classifier. Consider the output of the last neuron to be z, and the output of the neural network as:

$$y = \sigma(RELU(z))$$

The Outputs which are greater than 0.5 are considered as class 1. What is the problem of this neural network?

When we apply sigmoid activation function to a ReLU activation function we get a function whose y values are all bigger than 0.5. (The sketch of the function is drawn)
And since the outputs which are greater than 0.5 are considered as class 1 so this model predicts all the classification inputs to be one and it is obious how terrible this model is :).
layer.

4.6 (5 Points)

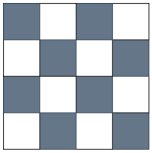
What would happen to MLPs if we did not have activation functions?

If we did not have activation functions in MLP simply the model would be linear. Forexample lets say that there are two layers in the MLP and the first layer weight matrix is A with bais b. Then the output of the first layer would be Ax+b. Then consider the weight matrix of the second layer to be C with bias d. So the output of this layer would be C(Ax+b)+d=CAx+(Cb+d) which is again a linear model of features. So if we don't have any activation funcitons in our MLP we won't be able to solve non linear problems like xor problem with this model.

5 Intuitive Questions (25 Points)

5.1 (20 Points)

Consider a 4*4 image (16 pixels) like the below figure. The black squares have the value 0, and the white ones have the value of 1.
In this task, you have to find the following chess-like pattern by designing a MLP. You have to pick the suitable weights, biases and activation functions(Step, RELU, etc) for the network.



Hint: The input Layer should contain 16 Neurons. The output node specifies whether if the image matches the pattern or not.

We assume we want to find the chess-like patterns in this question hence the board can be either like the pattern provided in the previous picture or it could be exactly vice versa with black cells being white and white cells being black.
However, if only the provided pattern is demanded the answer would be just the top part of the final neural network drawn at the bottom of the page and we would only have one input layer and one output layer with bias -7 and with ReLU activation function.

if we name the pixels from top left to down right as $x_1,\dots,x_{16}$ its obvious that the output function is as follows:

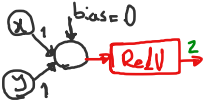
$$f(x)= (x_1=0 \wedge x_2=1 \wedge x_3=0 \wedge \dots \wedge x_{15}=0 \wedge x_{16}=1) \vee (x_1=1 \wedge x_2=0 \wedge \dots \wedge x_{15}=1 \wedge x_{16}=0)$$

So first we should implement $x \wedge y'$ and $x \vee y$ with SLP and then we can design the final MLP.

For $x \wedge y'$ we can have:

$$z=\begin{cases}1 & \text{if } x=1 \text{ and } y=0\\ 0 & \text{else}\end{cases}$$

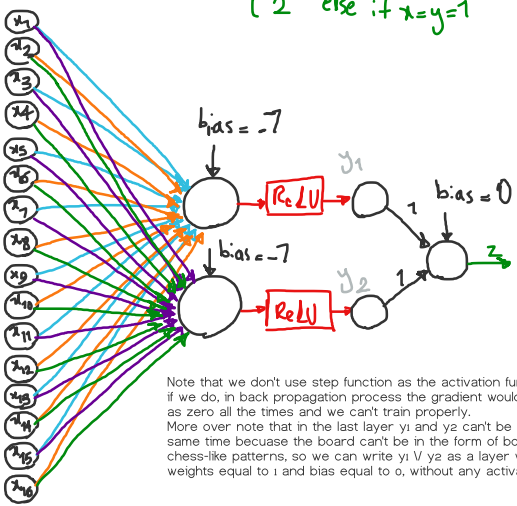
And for $x \vee y$ we have:



$$z=\begin{cases}0 & \text{if } x=0 \text{ and } y=0\\ 1 & \text{if } x \text{ xor } y=1\\ 2 & \text{else if } x=y=1\end{cases}$$

So for the final MLP we have:

$w=-1$
$w=1$
$w=-1$
$w=1$



Note that we don't use step function as the activation function because if we do, in back propagation process the gradient would be calculated as zero all the times and we can't train properly.
More over note that in the last layer y_1 and y_2 can't be both 1 at the same time becuase the board can't be in the form of both possible chess-like patterns, so we can write $y_1 \vee y_2$ as a layer with both weights equal to 1 and bias equal to 0, without any activation functions.

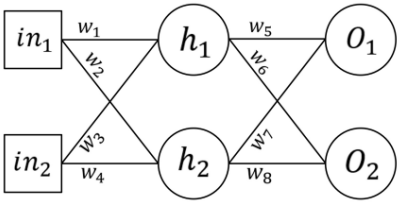
5.2 (5 Points)

What are the problems of using MLP as an Image/Pattern Classifier?

One of the most important problems when using MLP to classify images is that the number of weights which we should find rapidly becomes unmanageable for large images. As a result training would be very difficult and it can even lead to overfitting which makes it lose its ability to generalize.
Another problem is that MLP reacts differently to inputs(images) if the images are shifted or transformed. Forexample if there are many images in which there is a picture of a dog at the top right, after MLP trains the data it expects always to see the dog picture at the top right of the image and hence if in test data we have a dog picture at bottom left of the image, MLP probably can't classify it as a dog image. So we need to use a model that can leverage the spatial correlation of the image such as CNNs.

6 Computational Question (40 Points)

Take a look at this Neural Network. (The activation function is sigmoid)



Consider these parameters:

$$w_i = i \times 0.1$$

$$in_1 = 0.1$$

$$in_2 = 0.5$$

$$b_{h1} = 0.25$$

$$b_{h2} = 0.25$$

$$b_{o2} = 0.35$$

$$b_{o2} = 0.35$$

$$lr = 1$$

$$t_1 = 0.05$$

$$t_2 = 0.95$$

$$E = \frac{1}{2}\sum(t_1 - o_2)^2$$

Calculate the updated weights after 1 iteration of forward pass and backward pass. Consider decimal accuracy up to 3 digits.

First we calculate the neurons value after forward pass.
h1=Sigmoid(w1*in1+w3*in2+bh1)=Sigmoid(0.41)= 0.601
h2=Sigmoid(w2*in2+w4*in4+bh2)=Sigmoid(0.47)= 0.615
O1=Sigmoid(w5*h1+w7*h2+bo1)=Sigmoid(1.081)=0.746
O2=Sigmoid(w6*h1+w8*h2+bo2)=Sigmoid(1.202)=0.768
Then if we backpropagate we will have:

$$h_1 = \frac{1}{1+\exp(-(w_1 \times in_1 + w_3 \times in_2 + b_{h_1}))}$$

$$\Rightarrow \frac{\partial h_1}{\partial w_1} = h_1(1-h_1) \times in_1$$

$$L = \frac{1}{2}\sum_{i=1}^{2}(t_i - O_i)^2 \Rightarrow \frac{\partial L}{\partial O_1} = O_1 - t_1 \;,\; \frac{\partial L}{\partial O_2} = O_2 - t_2$$

$$\Rightarrow \frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial O_1} \times \frac{\partial O_1}{\partial w_5} = (O_1 - t_1) \times (O_1(1-O_1) \times h_1) = 0.079$$

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial w_6} = (O_2 - t_2) \times (O_2(1-O_2) \times h_1) = -0,019$$

$$\frac{\partial L}{\partial w_7} = \frac{\partial L}{\partial O_1} \times \frac{\partial O_1}{\partial w_7} = (O_1 - t_1) \times (O_1(1-O_1) \times h_2) = 0,087$$

$$\frac{\partial L}{\partial w_8} = \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial w_8} = (O_2 - t_2) \times (O_2(1-O_2) \times h_2) = -0,019$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial O_1} \times \frac{\partial O_1}{\partial h_1} \times \frac{\partial h_1}{\partial w_1} + \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial h_1} \times \frac{\partial h_1}{\partial w_1}$$

$$= (O_1 - t_1) \times (w_5 \times O_1 \times (1-O_1)) \times (in_1 \times h_1 \times (1-h_1)) + ((O_2 - t_2) \times (w_6 \times O_2 \times (1-O_2)) \times (in_1 \times h_1 \times (1-h_1))$$

$$= 0.001$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial O_1} \times \frac{\partial O_1}{\partial h_2} \times \frac{\partial h_2}{\partial w_2} + \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial h_2} \times \frac{\partial h_2}{\partial w_2}$$

$$= (O_1 - t_1) \times (w_7 \times O_1 \times (1-O_1)) \times (in_1 \times h_2 \times (1-h_2)) + ((O_2 - t_2) \times (w_8 \times O_2 \times (1-O_2)) \times (in_1 \times h_2 \times (1-h_2))$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial O_1} \times \frac{\partial O_1}{\partial h_1} \times \frac{\partial h_1}{\partial w_3} + \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial h_1} \times \frac{\partial h_1}{\partial w_3} = 0.007$$

$$= (O_1 - t_1) \times (w_5 \times O_1 \times (1-O_1)) \times (in_2 \times h_1 \times (1-h_1)) + ((O_2 - t_2) \times (w_6 \times O_2 \times (1-O_2)) \times (in_2 \times h_1 \times (1-h_1))$$

$$\frac{\partial L}{\partial w_4} = \frac{\partial L}{\partial O_1} \times \frac{\partial O_1}{\partial h_2} \times \frac{\partial h_2}{\partial w_4} + \frac{\partial L}{\partial O_2} \times \frac{\partial O_2}{\partial h_2} \times \frac{\partial h_2}{\partial w_4} = 0.005$$

$$= (O_1 - t_1) \times (w_7 \times O_1 \times (1-O_1)) \times (in_2 \times h_2 \times (1-h_2)) + ((O_2 - t_2) \times (w_8 \times O_2 \times (1-O_2)) \times (in_2 \times h_2 \times (1-h_2))$$

$$= 0.007$$

$$w = w - \eta \nabla_w L \Rightarrow w_1 = 0.099 \quad w_2 = 0.199 \quad w_3 = 0.295$$

$$w_4 = 0.393 \quad w_5 = 0.421 \quad w_6 = 0.619$$

$$w_7 = 0.619 \quad w_8 = 0.819$$