Name:- Kalyan Kumar Paladugula

NetID:- Kpalad4

UIN:- 679025059.

Final Exam

1) a) No. Generally speaking, Regression problems are easier than classification problems because Regression is continuous optimization and classification is discrete optimization. continuous optimization is easier than the discrete. The constraint of the classification is that the o/p of the classification should not take an arbitrary value.

Ex:- $y = w^T x$ - Regression

$y = \text{sign}(w^T x)$ - classification.

The sign function makes the associated cost function a non-convex. So, we could end up in the local minima.

~~Even when we need to apply Kernel transformation for Linear~~

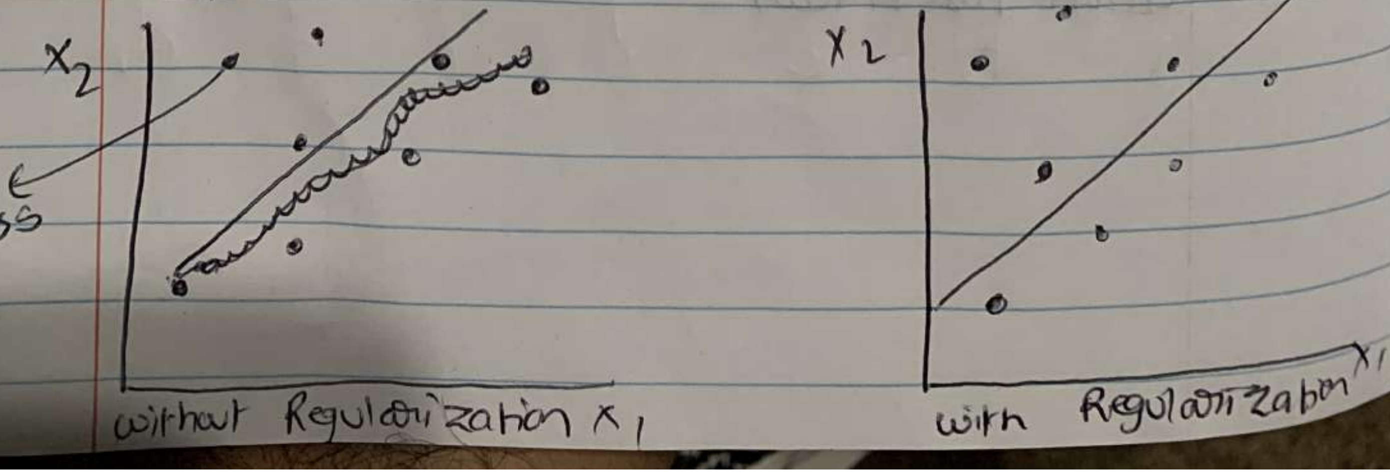
b) False.

With higher C, the margin would become small (the error / hinge loss ~~st~~ would be less) So, the No of support vectors would be lower than the original SVM.

c) False.

Regularization is done to increase the accuracy / decrease the error in the testing dataset. i.e to make model more generalizable which might increase the error in training set

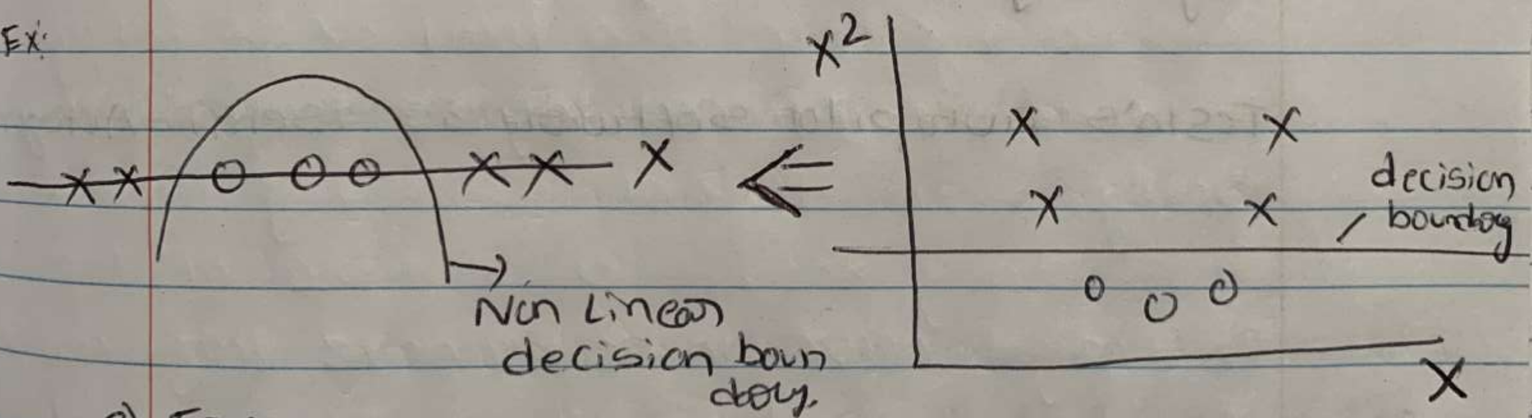
Ex:



d) True:-

Kernel transformation produces higher dimensional version of the data in which ~~a~~ the data could become linearly seperable and SVM produces ~~a~~ linear decision boundaries in the transformed space. If we project the data along with decision boundary, back to the original dimensions, the discriminators would be nonlinear

Ex:



e) False.

KNN is a lazy learner - no training.
so, the training time has no correlation with the 'K' value

2) a) Self-driving car programmes by tech giants like Google, Tesla etc

Ex:

In 2018, a tesla car crashed by hitting ~~the~~ a highway barrier and tesla auto-pilot system found probably at fault in this crash. In this case, the Tesla software failed to detect the high way barrier.

Tesla's autopilot software is semi-Autonomy

b) Leave-one-out-cross-validation:- (LOOCV)

In this, we leave one data point for validation/testing and use the rest for training.

LOOCV can be used for the KNN. as KNN is a lazy learner - No training, LOOCV would be very efficient for KNN.

c) ~~Very less/zer~~

c) i) ~~zero~~ very low training error and high testing error
ii) high variance, Low bias.
iii) when training loss continues to decrease but validation loss started to increase
... boundaries

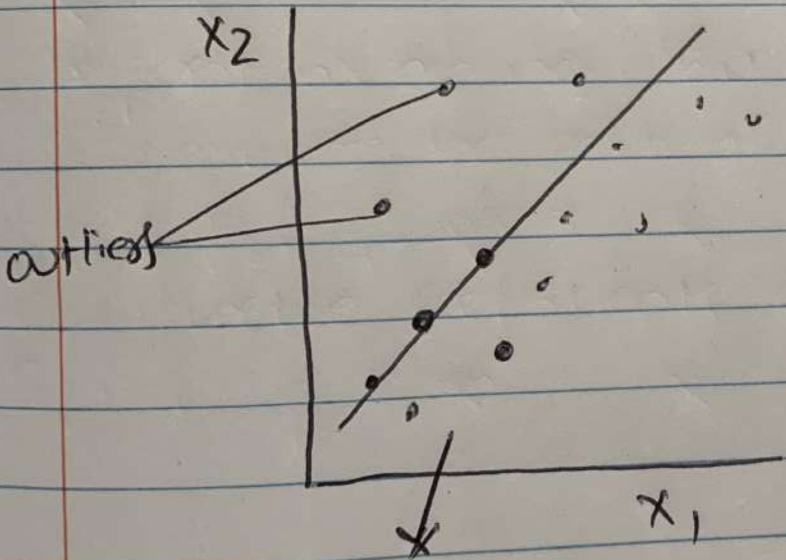
2) c) i) Very less/zero training error and high testing error

- Low generalizability.

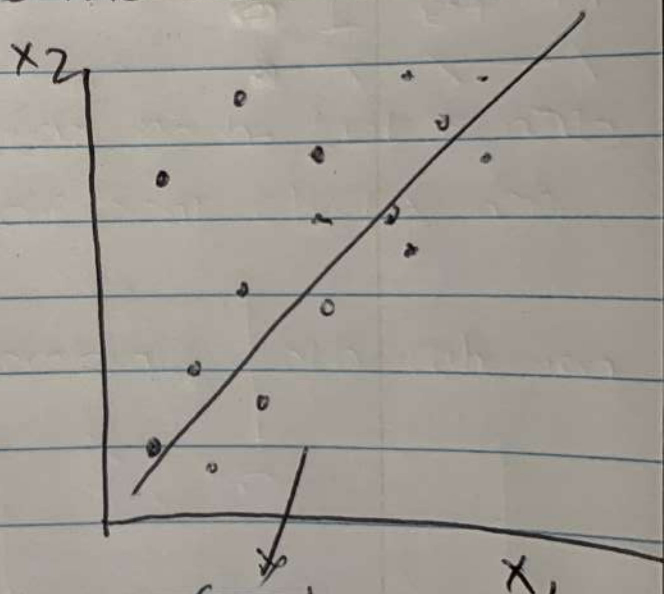
ii) In the error vs Complexity graph, if the training error continues to decrease but the testing/validation error starts increasing

- overfitting region

iii) When the model fits the outliers also



overfitting



Good model

iv) Very complex decision boundaries

v) If the model has high variance and low bias, then we can say it is overfitting.

d) <u>Vanishing Gradient</u>

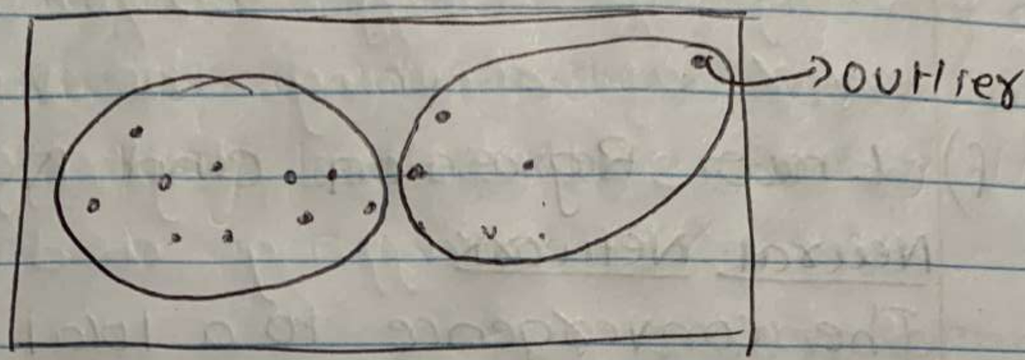
when the |gradient update| < 1, during the backpropagation, since we multiply the sensitivity of the connected nodes, ~~by~~ the gradient updates would become very small for the Initial layers of the network. This is called "Vanishing Gradient".

→ This causes Initial layers to learn very slowly.

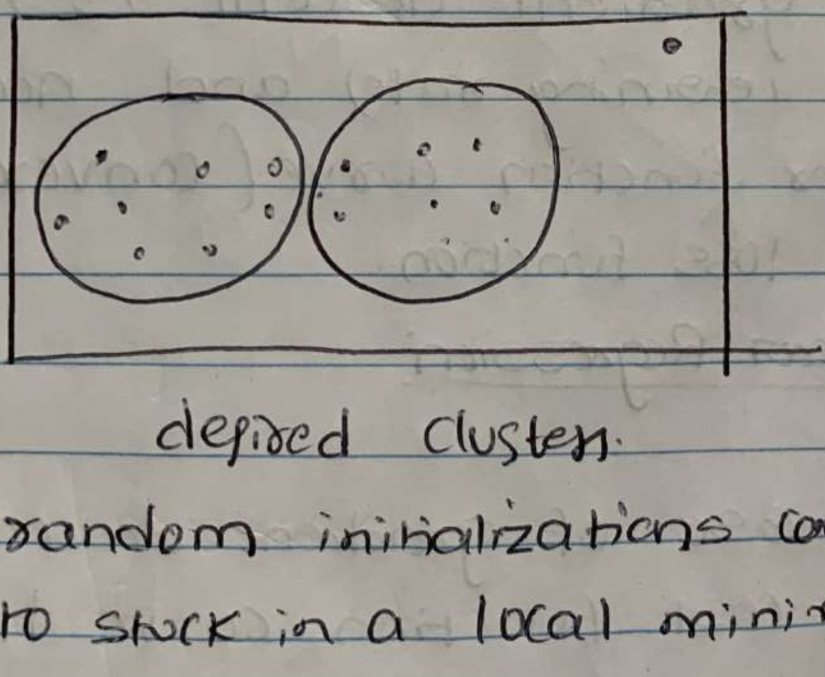
This might occur when we have large number of hidden Layers/ Nodes. i.e. Deep Neural Networks or when |gradient update| ≈ 0. or ~~when~~ due to the sigmoid Activation function

e) KMeans is vulnerable to the Random start points.

i) If we choose the outliers as starting points, KMeans takes a long time to converge or might produce incorrect clusters



Predicted clusters.



desired Clusters.

ii) Some random initializations cause Kmeans to stuck in a local minima rather than Global minima

f) Linear Regression and Neural Networks.

Neural Network

The convergence to a local or global optima is contingent on the learning rate (as gradient descent is heavily dependent on learning rate) and nature of the error function curve (convex or non-convex) ie loss function.

Linear Regression:-

In case of gradient descent for Linear regression the cost function curve has only one minimum that is global minimum. Hence, here gradient descent always converge to Global minimum keeping learning rate to neither too small or too large.

g) By seperating the data into training and testing, we are avoiding "Data snooping" which causes good accuracy in the Seen data but poor accuracy in the unseen data.

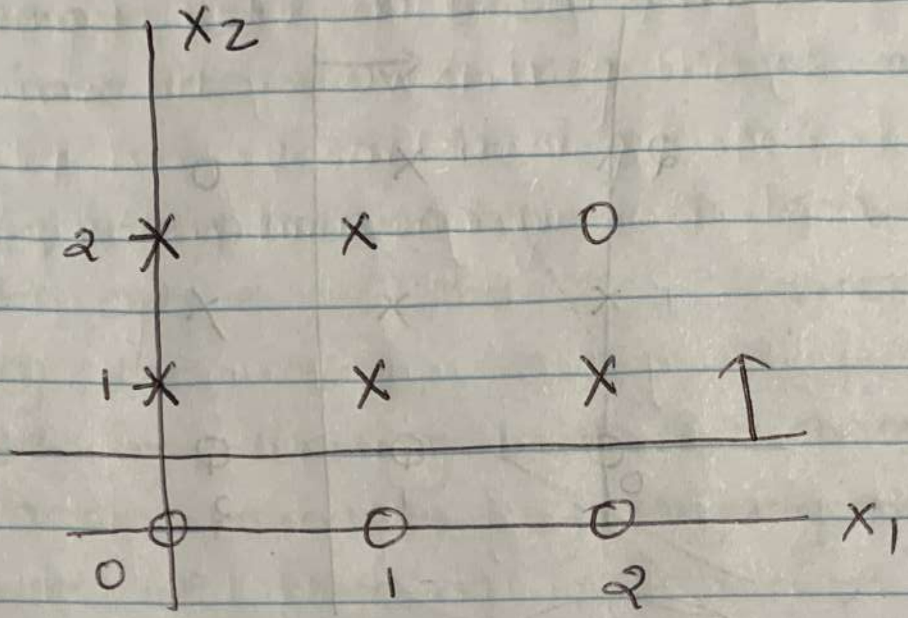
By building the model on the training data, we can check it's generalizability by applying it on the testing data.

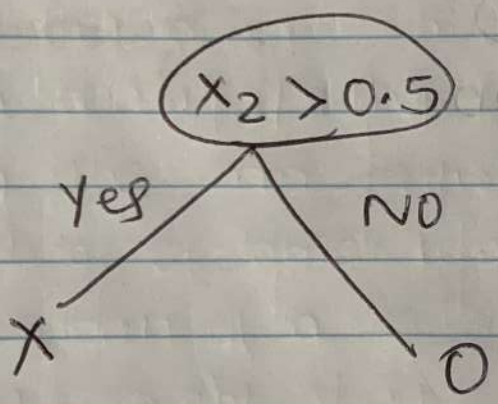
We can also find the best hyperparameters by applying the model on Validation/testing data.

h) Ensemble methods use unstable learners as base learners to make them independent.

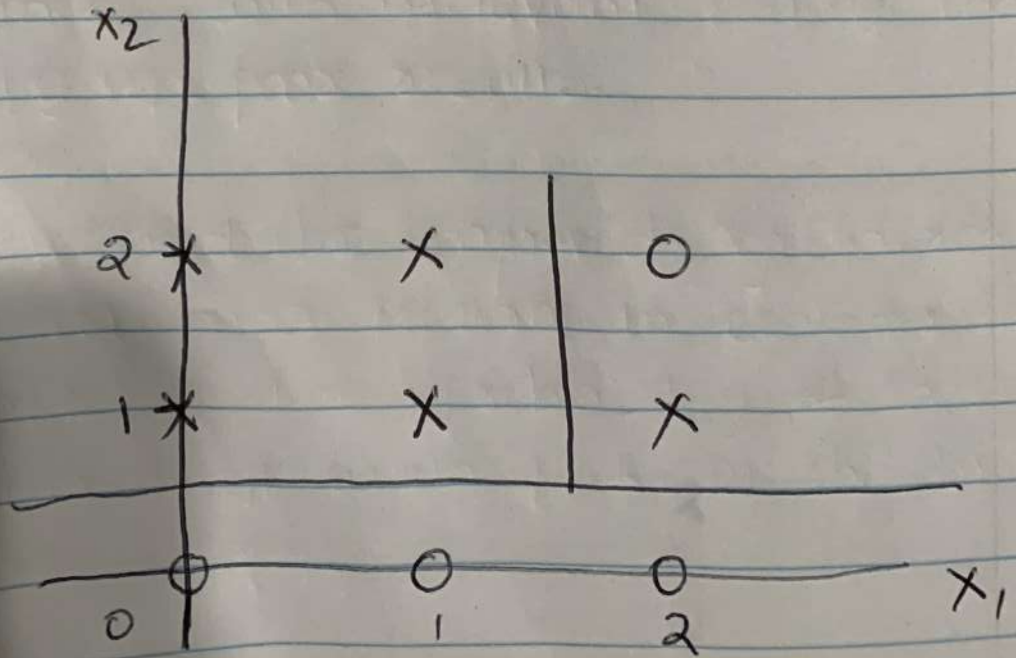
Since, there would be a lot of variance in the outputs of the base learners, by taking the voting, we would reduce the error by reducing the variance of the base learners by aggregation/ensembling in case of bagging and bias in case of boosting.

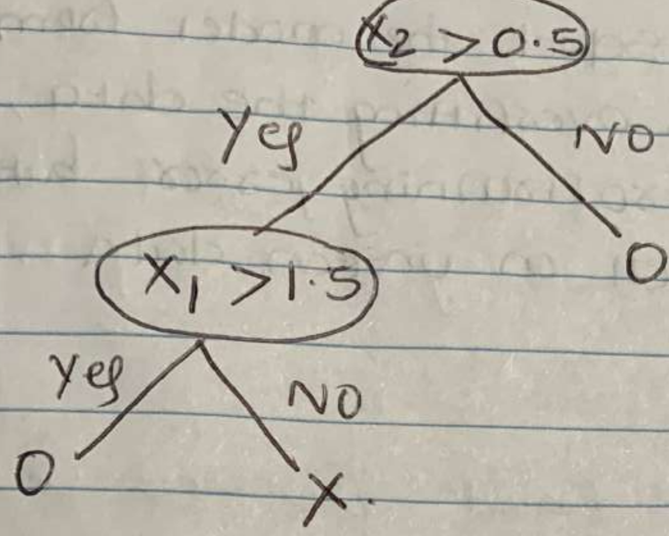
3) a)



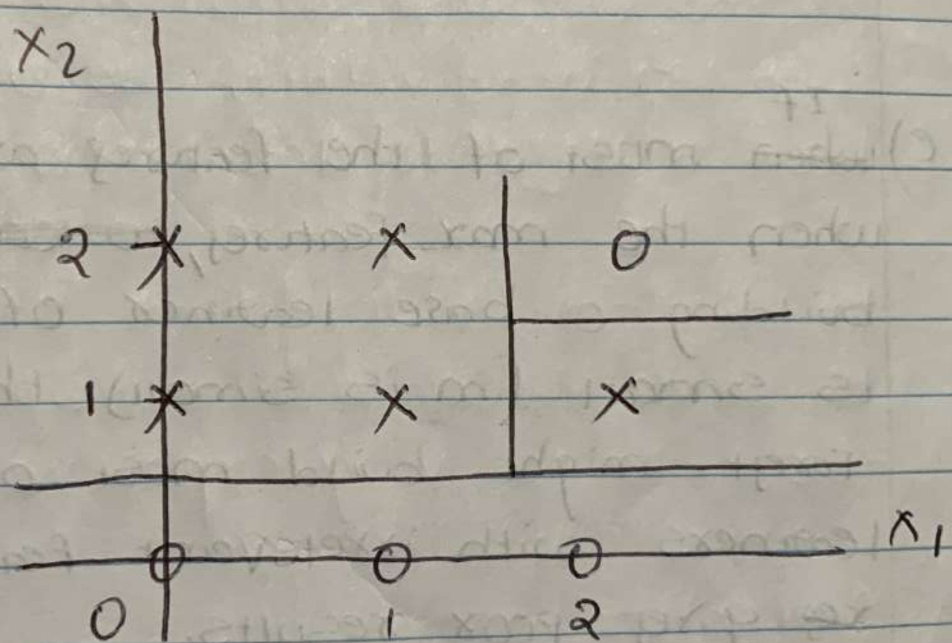


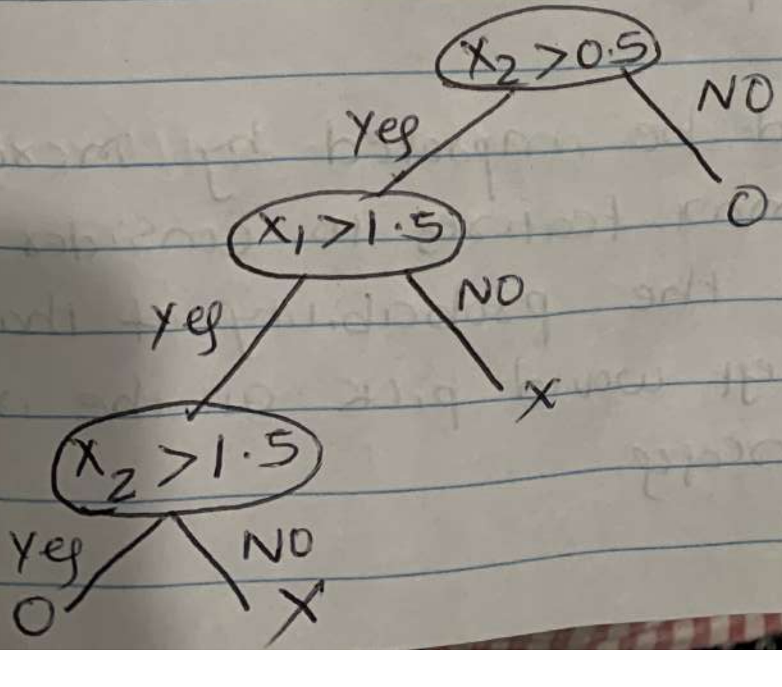
b)




x2 > 0.5
yes
NO
x1 > 1.5
O
yes
NO
O
X

c)


x2
2
1
0
1
2
x1


x2 > 0.5
yes
NO
x1 > 1.5
O
yes
NO
X
x2 > 1.5
yes
NO
O
X

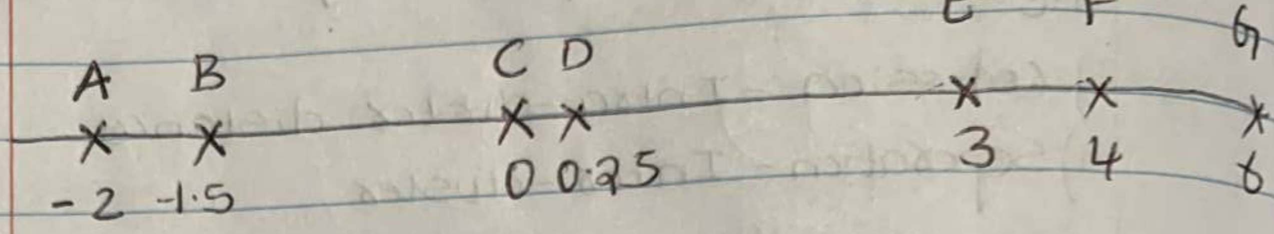
d) I might not select the model from c because it is overfitting the data, this model has zero training error but might not perform well on unseen data

e) ~~when~~ If most of the features are noise, when the max_features, ~~to consider~~ for building a base learner of Random Forest, is small (m is small) then, Random-Forest might build most of it's base learners with irrelevant features that ~~rel~~ gives poor results.

This could be improved by increasing the m (max_features to consider for base learners) as the probability ~~of~~ that the Random forest would pick all the irrelevant features decreases

4) unsupervised Learning

a)



AD = 2.25

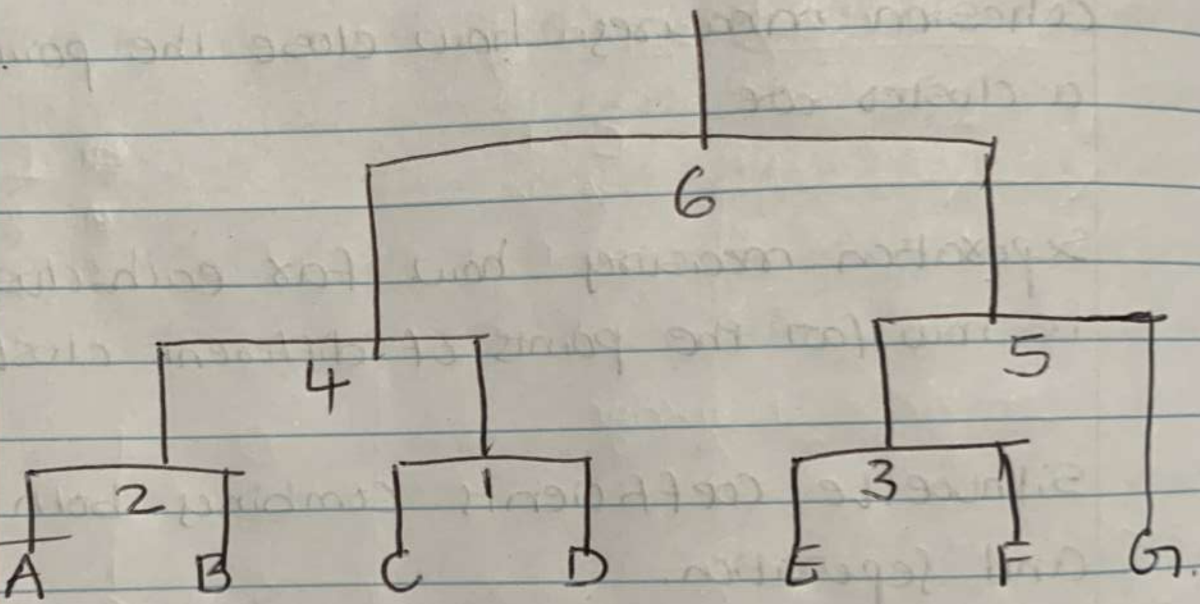
AG = 8

AF = 6

CF = 4

CG = 6

EG = 3.



AF = 6

AG = 8

EG = 3

4) b) Semi-supervised learning

Semi-supervised Learning falls between unsupervised Learning and Supervised Learning.

In SSL, we first use the unlabeled dataset. we first use unsupervised learning on the unlabled data and cluster the Similar data. Then we use the existing labeled data to label the unlabeled data

It is most useful when the acquistion of unlabeled data is cheap and labeling the data is very expensive

Applications

i) Alexa
ii) Movie Recommendation - Netflix, prime
iii) Book Recommendation

c) We can judge/assess the quality of an unsuper-
-vised learning approach by calculating
the

i) cohesion - Intra-cluster distance
ii) seperation - Inter-cluster "

cohesion measures how close the points in
a cluster are

seperation measures how far each clusters are
ie how far the points of different clusters are

silhuoette coefficients combines both cohesion
and seperation.

Q) 5) a):

The hyperparameters for my final neural network are:

No of Hidden Layers = 1, No of hidden nodes = 10, Learning Rate = 0.1, no of iterations/epochs = 1000, solver = 'adam', activation function = 'relu'

To make the algorithm simple, I used rectangular weight matrix i.e. the number of nodes in all hidden layers is same

I used the following grid for the Grid Search method to select the best hyperparameters:

1. hidden = [1,2,5]
2. nodes = [2,5,10]
3. lr =[0.1,1]
4. af =['tanh', 'relu']
5. solver =['sgd','adam']

I tried every combination of the above (72 combinations) and calculated the cross-validation error. Then, I picked the hyperparameters which gave the least cross validation error.

I applied the best model on the test dataset and got **77** percent accuracy

While doing the search, I got a few convergence warnings which say that the no of iterations = 1000 is not enough for the model to converge. This may be due to the 0.1 learning rate

Q) 5) b):

The concentration bound obtained at 95% confidence interval using Hoeffding bound is **0.096**

Q) 5) c):

I am getting different results (accuracy) every time I run my neural network because of

1) Random initialization of weights
2) Randomness in the training data (I set shuffle = True in the MLPClassifier)
3) Due to the adaptive Learning Rate (I set Learning Rate = 'adaptive' in the MLP Classifier)
4) Multiple Local Minima (in case of Non-Convex cost functions)

Q) 5) d):

We can say whether a model is overfitting or not by checking the Training error.

I saved the training error of all the models I built using the cross_validate function of the Sklearn.model_selection. To get the training error, I set the return_train_score parameter of cross_validate to True.

For the model with the least validation error (the model that I picked as the best) has training error of **0.17.**

Q) 5) c)

```
print('The training error obtained with my best model = ' +str(t_error))
```

```
The training error obtained with my best model = 0.17041666666666666
```

Since, the training error is not zero, I can say that my model is not overfitting.

I checked the testing accuracy of the model that has low training error:

```
b_hl2, b_hn2, b_lr2, b_af2, b_s2, error,t_error2 = minimum7(d,e,f,g,h,c,b)
print(t_error2)
b_hid2 = [b_hn2 for n in range(b_hl2)]
classifier2  = MLPClassifier(hidden_layer_sizes= b_hid2, activation = b_af2, learning_rate_init = b_lr2, learning_rate = 'adaptive',
classifier2.fit(X_train, y_train.values.ravel())
y_pred2 = classifier2.predict(X_test)
nn_accuracy2 = accuracy_score(y_test, y_pred2)
print('The testing accuracy obtained with the model that has lower training error = ' + str(nn_accuracy2))
```

```
0.015972222222222221
The testing accuracy obtained with the model that has lower training error = 0.755
```

I got lesser testing accuracy of 0.755 with the model that has low training error of 0.016.

Note: I used random_state =100 in the MLP classifier in the code to check the testing accuracies of my best model and the model that has low training error of 0.016