WINE DATASET

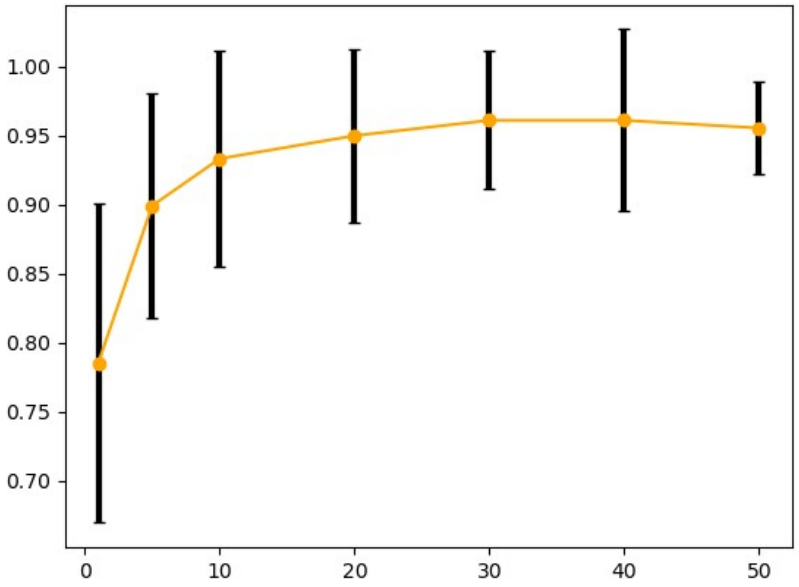
Q1,Q2) Printing Accuracy, Precision, Recall, F1Score for each value of nTree

```
Random Forest Accuracy nTree=1 = 0.7854166666666668
Random Forest Precision nTree=1 = 0.7895634920634921
Random Forest Recall nTree=1 = 0.7808333333333334
Random Forest F1 Score nTree=1 = 0.7851412775556353
Random Forest Accuracy nTree=5 = 0.8993055555555556
Random Forest Precision nTree=5 = 0.9074867724867726
Random Forest Recall nTree=5 = 0.8903174603174605
Random Forest F1 Score nTree=5 = 0.8987080363289837
Random Forest Accuracy nTree=10 = 0.9333333333333333
Random Forest Precision nTree=10 = 0.9444444444444444
Random Forest Recall nTree=10 = 0.9352380952380953
Random Forest F1 Score nTree=10 = 0.9396909529175016
Random Forest Accuracy nTree=20 = 0.95
Random Forest Precision nTree=20 = 0.959920634920635
Random Forest Recall nTree=20 = 0.9533333333333331
Random Forest F1 Score nTree=20 = 0.9564767992875511
Random Forest Accuracy nTree=30 = 0.961111111111111
Random Forest Precision nTree=30 = 0.9662698412698413
Random Forest Recall nTree=30 = 0.962857142857143
Random Forest F1 Score nTree=30 = 0.9645366241132504
Random Forest Accuracy nTree=40 = 0.961111111111111
Random Forest Precision nTree=40 = 0.9715873015873017
Random Forest Recall nTree=40 = 0.9666666666666668
Random Forest F1 Score nTree=40 = 0.9690279211722727
Random Forest Accuracy nTree=50 = 0.9555555555555555
Random Forest Precision nTree=50 = 0.9621031746031745
Random Forest Recall nTree=50 = 0.9592063492063494
Random Forest F1 Score nTree=50 = 0.9606328617979065
```

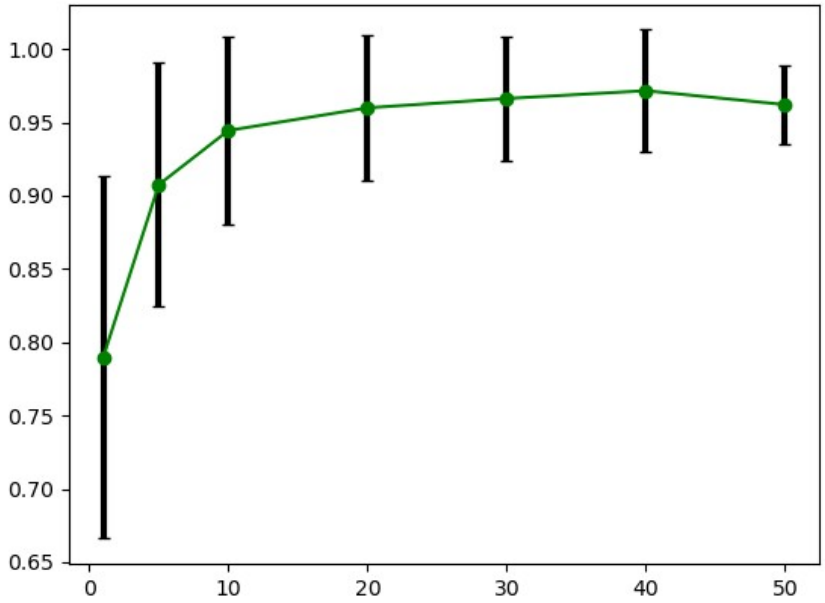
```
Final Accuracies [0.7854166666666668, 0.8993055555555556, 0.9333333333333333, 0.95, 0.961111111111111, 0.961111111111111
, 0.9555555555555555]
Final Precision [0.7895634920634921, 0.9074867724867726, 0.9444444444444444, 0.959920634920635, 0.9662698412698413, 0.97
15873015873017, 0.9621031746031745]
Final Recall [0.7808333333333334, 0.8903174603174605, 0.9352380952380953, 0.9533333333333331, 0.962857142857143, 0.96666
66666666668, 0.9592063492063494]
Final F1 Score [0.7851412775556353, 0.8987080363289837, 0.9396909529175016, 0.9564767992875511, 0.9645366241132504, 0.96
90279211722727, 0.9606328617979065]
```

Final Accuracies, Final Precision, Final Recall and Final F1 Score are recorded in order of nTree values [1,5,10,20,30,40,50]

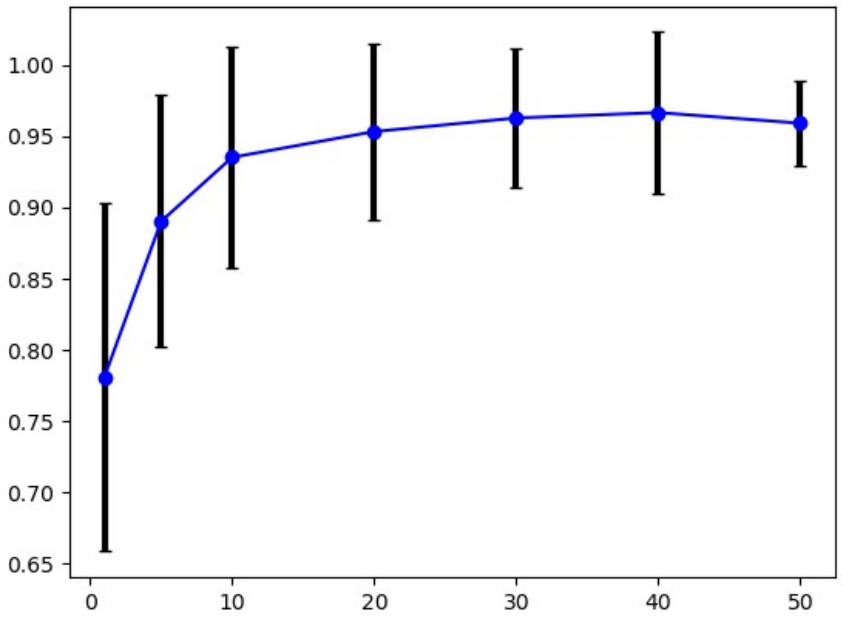
Q3) WINE DATASET Accuracy -



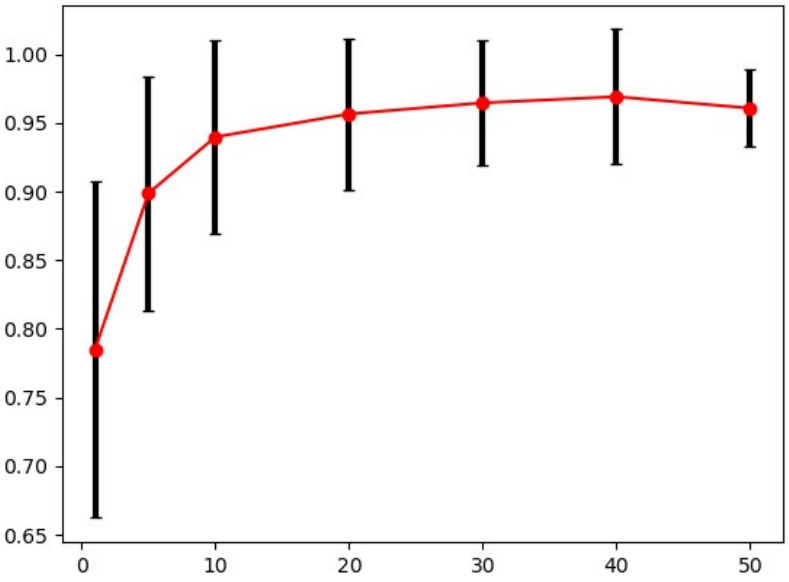
WINE DATASET Precision -



WINE DATASET Recall -



WINE DATASET F1 Score -



Q4)

WINE DATASET Accuracy –

For high accuracy, I would pick a higher value of **nTree.** This is because individual decision trees have very high variance. So, if we pick a smaller nTree value, we have a high chance of overfitting the training data. This will result in bad predictions for the incoming test data. This is evident from the accuracy readings printed above fluctuating from **77% for nTree=1** to **96% for nTree=50**. In real life we should go for a higher number of decision trees because we can expect lots of rows of data, and since we are using randomized attributes for splitting, higher value of nTree makes sure all attributes are accommodated. I would choose **nTree=30** for lesser computational time. But if possible, I would also consider **nTree=50** if we don't have any computational limitations because the measurements have lower variance and also it will help in real life if we have several more rows of data. Also for lower nTree values there is higher variance in accuracy readings between the 10 cross-validation sets. This variation reduces with higher nTree value which indicates to pick a higher value.

WINE DATASET Precision –

For high precision, similar to accuracy, there are several good nTree values here. One thing to consider when optimizing for high precision is the precision-recall tradeoff. High precision is good but sometimes it might mean our model is strictly focused on predicting one subtype of a class and can miss all other subtypes because of this. So it might not always be a good indictor if we are compromising other metrics. **nTree=30,40** look like a good value as it gives high precision.

WINE DATASET Recall –

Similarly for high recall we can pick **nTree=30(**does not add too much computational time**) or 50** because this tree value also corresponds with high precision readings so this ensures there is no tradeoff. Also the variation in measurements in lesser than for **nTree=40**.

WINE DATASET F1 Score –

F1 score gives a better measure of the incorrectly classified instances. Basically, if the FNs and FPs in a dataset are very important. For example, in the case of detecting cancer(I think accuracy is a pretty solid metric in our case). **nTree=30,50** gives a pretty good F1 score also with solid readings for all the other metrics.

Q5) WINE DATASET

The **nTree** value affects all metrics. We can see that **for nTree=1** , all the metrics suffer. Even though the mean accuracy for **nTree=1** is 78%, there is very high variance between performance metrics (some trees have very low accuracy, some trees have 100% accuracy) of individual decision trees(from the 10 cross-validation splits) used for training forest with **nTree=1** model. The variance between accuracy measurements significantly reduces with increase in nTree value**.** This is because by increasing the number of trees, we are reducing the overall variance of the entire random forest. Because even though one tree may be biased towards a particular dataset, other trees are influenced by different datasets and this evens out. Thus the overall error of the random forest reduces linearly with increase in decision trees. **All the metrics** are affected equally by this change. If I had to pick, I would say accuracy is more slightly more important for this dataset. From nTree=30 onwards, all the metrics have good/high measurements overall. The **F1 Score** is also significantly improved by increasing the number of trees in the ensemble. I think after a point say **nTree=30** in our case, there is no point in taking higher nTree values. This is because there is no significant change in performance from that point onwards. We do need more nTree values with the greater number of attributes we have because since we are using randomized splits to calculate information gain for our decision trees, having more trees will ensure that all attributes get fair chance or representation. However, increasing the nTree values by a lot (at some point say >50,100 etc.) will drastically affect the computational time which we do not want. So this is essentially negatively impacting our performance.

HOUSE VOTES DATASET

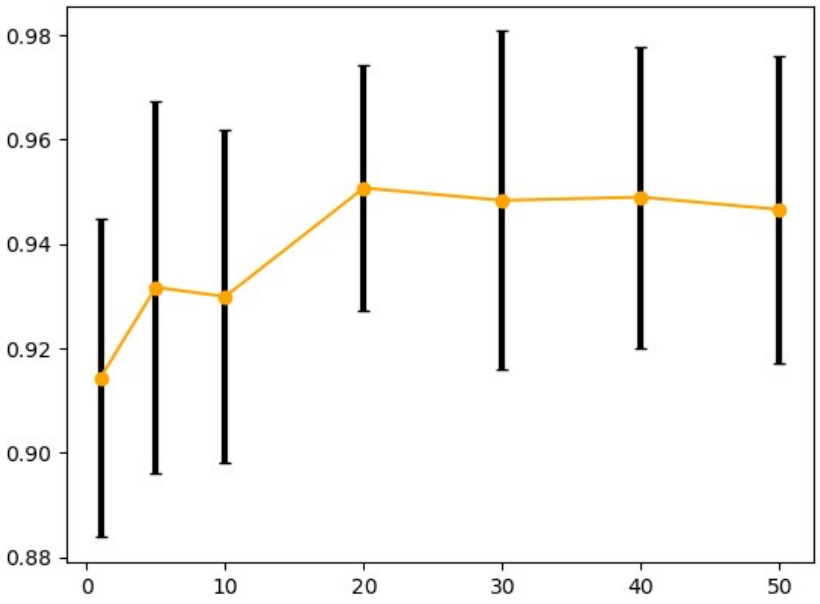
Q1,2)

```
Random Forest Accuracy nTree=1 = 0.9144110275689223
Random Forest Precision nTree=1 = 0.8968346771907143
Random Forest Recall nTree=1 = 0.8833333333333334
Random Forest F1 Score nTree=1 = 0.8881395056556347
Random Forest Accuracy nTree=5 = 0.931704260651629
Random Forest Precision nTree=5 = 0.9064270152505447
Random Forest Recall nTree=5 = 0.9270833333333334
Random Forest F1 Score nTree=5 = 0.9139667069173711
Random Forest Accuracy nTree=10 = 0.9299498746867169
Random Forest Precision nTree=10 = 0.9039006830802496
Random Forest Recall nTree=10 = 0.9270833333333334
Random Forest F1 Score nTree=10 = 0.9111384567361798
Random Forest Accuracy nTree=20 = 0.950751879699248
Random Forest Precision nTree=20 = 0.926884531590414
Random Forest Recall nTree=20 = 0.9520833333333334
Random Forest F1 Score nTree=20 = 0.937864412627221
Random Forest Accuracy nTree=30 = 0.9483709273182956
Random Forest Precision nTree=30 = 0.9149704326174914
Random Forest Recall nTree=30 = 0.9645833333333333
Random Forest F1 Score nTree=30 = 0.9368803683794196
Random Forest Accuracy nTree=40 = 0.9489974937343358
Random Forest Precision nTree=40 = 0.9212523342670401
Random Forest Recall nTree=40 = 0.9583333333333334
Random Forest F1 Score nTree=40 = 0.9369482796878433
Random Forest Accuracy nTree=50 = 0.9466165413533835
Random Forest Precision nTree=50 = 0.9239432527396924
Random Forest Recall nTree=50 = 0.9458333333333334
Random Forest F1 Score nTree=50 = 0.9328726152089531
```

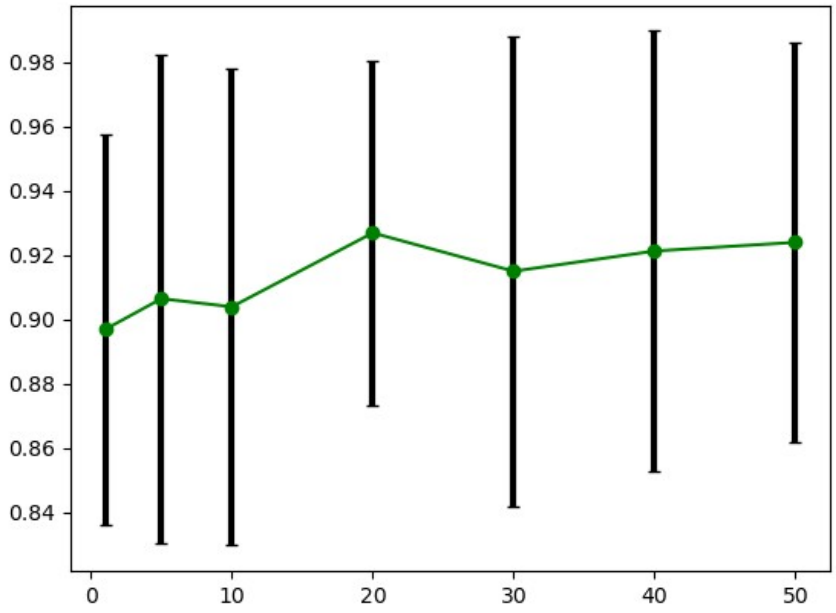
```
Final Accuracies [0.9144110275689223, 0.931704260651629, 0.9299498746867169, 0.950751879699248, 0.9483709273182956, 0.94
89974937343358, 0.9466165413533835]
Final Precision [0.8968346771907143, 0.9064270152505447, 0.9039006830802496, 0.926884531590414, 0.9149704326174914, 0.92
12523342670401, 0.9239432527396924]
Final Recall [0.8833333333333334, 0.9270833333333334, 0.9270833333333334, 0.9520833333333334, 0.9645833333333333, 0.9583
333333333334, 0.9458333333333334]
Final F1 Score [0.8881395056556347, 0.9139667069173711, 0.9111384567361798, 0.937864412627221, 0.9368803683794196, 0.936
9482796878433, 0.9328726152089531]
```

Final Accuracies, Final Precision, Final Recall and Final F1 Score are recorded in order of nTree values [1,5,10,20,30,40,50]

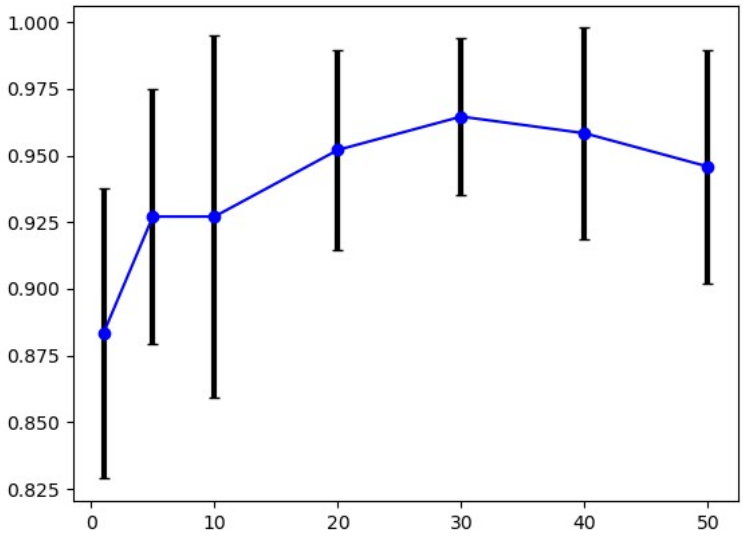
Q3) HOUSE VOTES Accuracy –



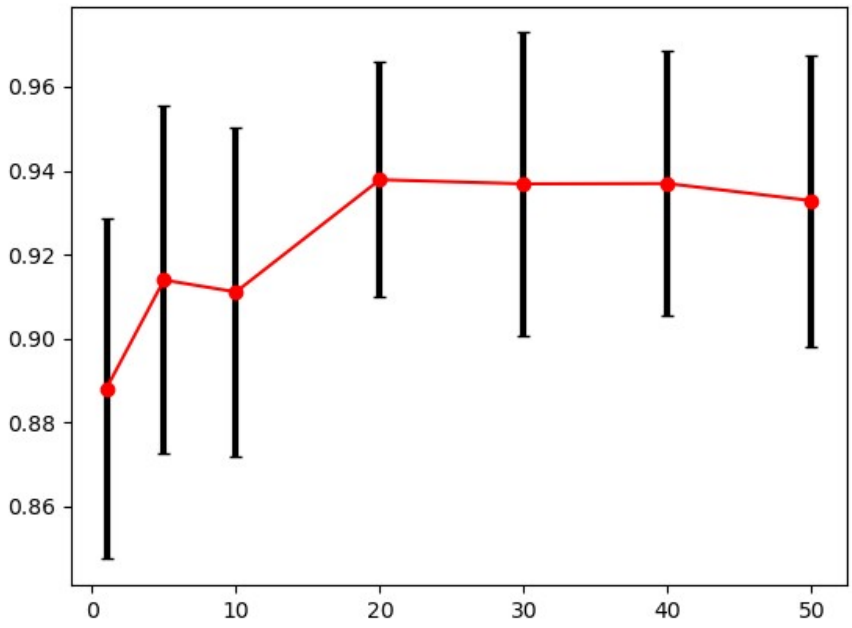
HOUSE VOTES Precision -



HOUSE VOTES Recall -



HOUSE VOTES F1 Score-



Q4) House Votes Accuracy –

For higher accuracy, I would pick a higher value of nTree. For the graph we notice that there is a spike in accuracy from nTree value 1 to 5. From there the accuracy drops slightly and then increases upto nTree=20. From there it remains pretty steady. Since accuracies from nTree=20 and onwards are pretty even; we can take into account the standar deviation measurements as well to choose a suitable nTree value. Since **nTree=20** has the lowest deviation, and has pretty high accuracy, I think it is a good nTree value to use in our case. Picking higher nTree values will also affect computational performance so that can be avoided in this case. Also another thing to consider is the accuracy might still go higher for higher nTree values which we haven't computed. Thus we would have to consider the tradeoff between computation time here and nTree values.

House Votes Precision –

The precision measurements also follow a similar pattern to accuracy. The average precisions measurements for the all nTree values are all within a very small range of one another. This means that we do not necessarily have to go for the highest nTree value as it is not significantly alterning this metric. We can see **nTree=20,50** have the highest readings. But nTree=20 has lower deviation in measurements.

Hence I would pick **nTree=20** here. However, if we have many rows of input data and features in real life; it might not hurt to consider higher nTree value as it gives a higher chance of all attributes being represented in the information gain.

House Votes Recall –
The graph shows good recall for all values. We have to ensure there is no precision tradeoff when picking recall and vice-versa. Candidates for good recall here are **nTree=20,30**. It really depends on whether you are training your model to prioritize recall(I don't think it is important for this dataset). As we can see from the graphs higher value for **nTree=30** is accompanied by drop in precision for the same nTree value.

House Votes F1 Score–
The F1 Score is also a very good metric here to gauge model performance. This is because this dataset is slightly unbalanced and a high F1 score is a good indicator of balance in Precision-Recall tradeoff. The F1 scores for 20 to 40 are very similar. This indicates to me that **nTree=20** is a good value to pick for deploying this model in real time as lower nTree value can same computational time.

Q5) HOUSE VOTES DATASET

All metrics were impacted equally for this dataset. I think there is certain bias in this dataset though. This is because there is one column – Physician_fee_freeze which has very high information gain. Since we are performing randomized splits for all the decision trees, whether this attribute gets picked or not greatly impacts the performance of an individual decision tree. Picking a high nTree value will help with the variance for the individual nTrees.
The **accuracy spike** from nTree =1 to 5 is also serves as a clear indicator of how variance of random forest is being reduced by increasing the nTree values. Afterwards, accuracy increases gradually.
Also after **nTree=20**, there is no (significant)increase in accuracy in our case, this indicates that **nTree=20** is a good value to train our Random Forests. Even if there was a small increase in accuracy with higher nTree values over 50, it will affect our computational performances, so it might not be worth it to go on increasing nTree value over some point. There is significant impact on **recall** in our case with increase in nTree value. This indicates our model is identifying more of all the available instances of the positive class. This is expected however as all the other metrics are also increasing as we increase nTree value upto 50. Thus we are able to identify the instances of the class correctly without significantly affecting other performance metrics. Also trying nTree values past 50 might increase the accuracy and F1 score even more, but I think it will come at the cost of computational performance. So that is a tradeoff which we need to consider when we are trying to select optimal nTree value.