Discussions and Figures

1. What is an ensemble classifier? Name some of the popular ensemble methods (at least three) and which one you prefer and why?

A: An ensemble classifier is essentially a cascade of classifier. This consists of several classifiers and predicts the results based on some variation of averaging of constituent classifiers. This is based on the idea that the combined intelligence of a group exceeds the intelligence of any one individual. Using this idea, generally the classifier outperforms the constituent classifiers.

Some popular ensemble methods are:

- Random Forest
- Bagging Classifier
- Boosting classifiers (Adaboost, Grad Boost, XGBoost)
- Voting based ensemble: Based on result of majority of classifiers.
- Stacked generalization – This does not average the results, rather combines the results. Main disadvantage is that it is computationally expensive.

The preference of classifier depends on the type of application and data. If speed is very important then I would go with Grad Boost, but if there is a possibly of overfitting then I would switch to Random Forest as they are harder to overfit as compared to Grad Boost.

2. Let's assume we have a noisy dataset. You want to build a classifier model. Which classifier is appropriate for your dataset and why?

A: This depends on the type of noise that is present in data. If it is noise caused by imbalance then I would go with ensemble (bagging/Random Forest) as it will aggregate models with high variance and reduce the overall variance..
If the noise present is because of high amount of outliers and less data points then I will design a pipeline based on SVM as they are more robust in such cases, although Random Forest also works well in this case. So, overall Random Forest would be a better classifier..

3. Load and pre-process the dataset if necessary. Explain steps that you have taken. Are there any alternative ways for doing that? Explain.

A: For pre-processing the data, I have taken following steps:

- Loaded the data and checked for null values, none found.
- Checked for duplicate rows, none.
- Followed this by binary label encoding the target variable "Response" and preserved the original labels in a list.
- Converted the date to datetime instance, then decomposed it into constituent year, month, day etc.
- Used one hot encoding to handle remaining categorical values. We can also use label binarizer but that does not preserve the information as much as one hot encoding.
- After this divided the encoded data into test and train set.

An alternate way of doing this would be to handle class imbalance as the dataset suffers severely from class imbalance. To handle this without changing the original data, I have used stratified splitting so that the class ratio is maintained even after splitting:

```
Imbalance ratio: after split 5.983180428134556
Imbalance ratio before split:  5.983180428134556
```

An alternate and ideal way would be to use stratified splitting followed by resampling the training set so that the class imbalance is removed in training.

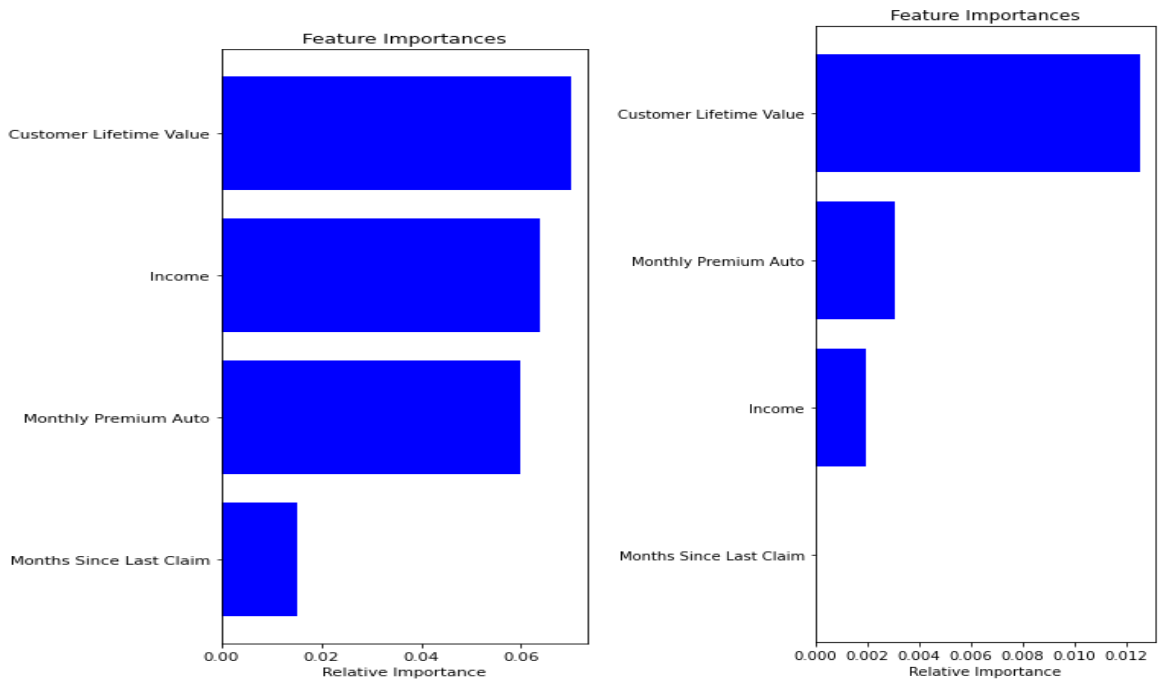
4. Analyse the importance of the features for predicting customer response using two different approaches. Explain the similarity/difference between outcomes.

A: To analyse feature importance I have used two sklearn methods

(a) Based on impurity: This is a property of Random Forest classifier instance. This is calculated based on mean and standard deviation of cumulative impurity decrease as the tree splits downwards to its leaf nodes.

(b) Based on permutation: This is like cross validation technique; a feature is randomly shuffled and the relationship with the target variable is broken. Hence, after the complete process we have the highest-ranking feature based on this permutation.

The major similarity between both is that they find the highest correlated feature with the target value and the biggest dissimilarity is that feature importance based on impurity if highly affected by the number of samples whereas feature importance based on permutation is not.



(a) Based on impurity **(b)** Based on permutation

This shows that both the approaches have given the highest importance to “Customer Lifetime Value” to predict customer response.

5. Create three supervised machine learning (ML) models except any ensemble approach for predicting customer response.

A: To predict customer response, I have used three models:

- SVM (RBF and Linear)
- KNN
- Decision Trees

a. Report performance score using a suitable metric. Is it possible that the presented result is an overfitted one? Justify.

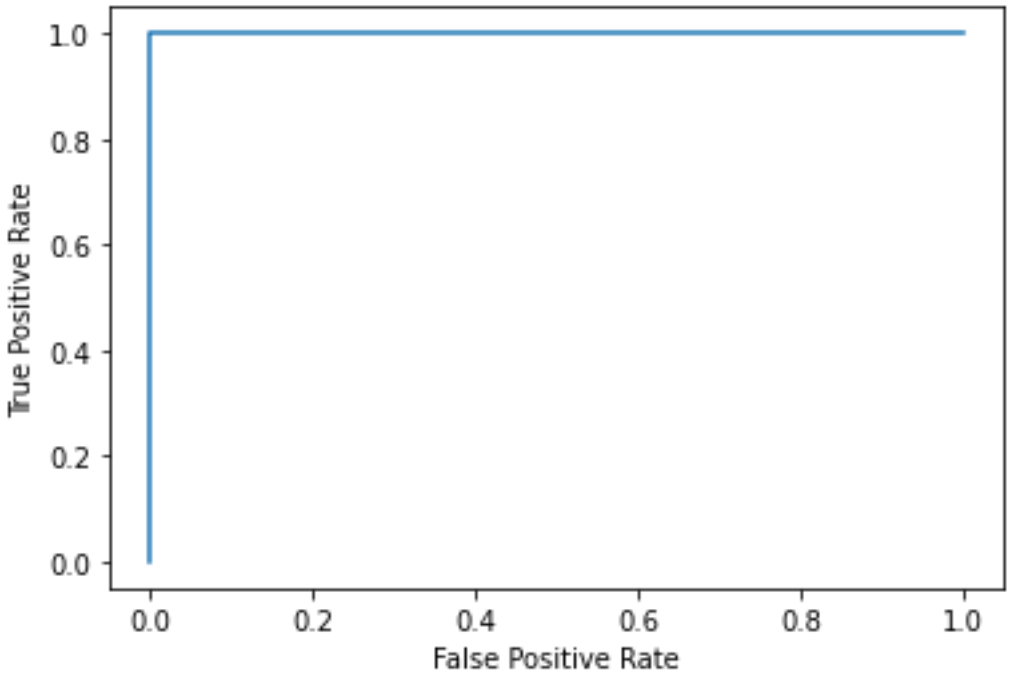
A: Classification performance analysis of models show that SVM with RBF was the best performing model when the model parameter “max_iter” was tweaked. This shows the ability of SVMs to transforming the data into higher dimension and finding the best possible fit.

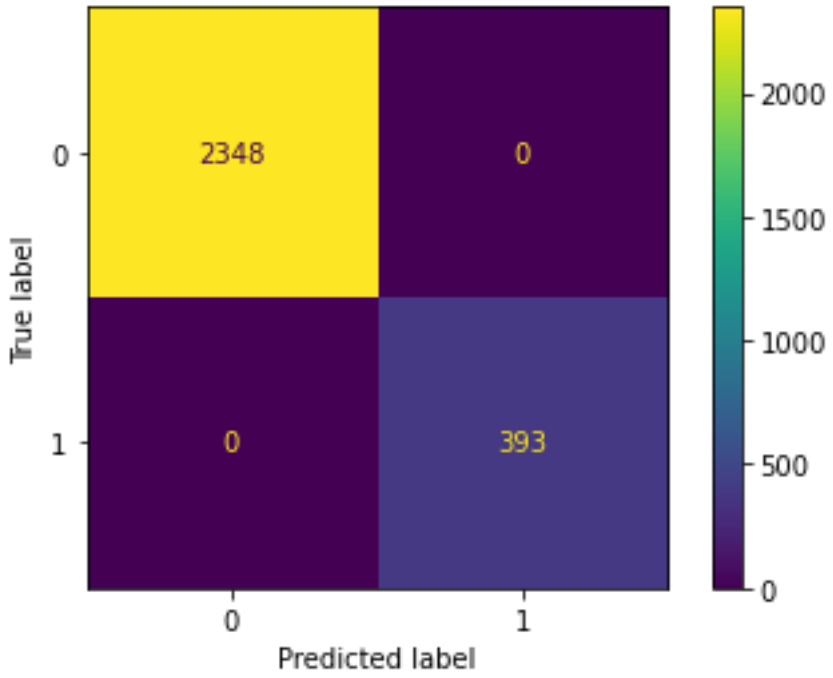
It is highly possible that the presented data is overfitted as there is a severe class imbalance present. This would mean that the models are perfectly trained on seen data but might fail on unseen data, to remove this either data augmentation via resampling or organically collecting new data is required. Although, the models trained here are performing well on cross validation tests which can mean that there is no overfitting, but to say conclusively class imbalance must be removed.

Performance statistics of all trained models is based on following metrics:

- **ROC curve** for Response “Yes”
- **Confusion matrix** showing distribution of predictions
- **Average time taken, accuracy, precision, recall and f1 score**
- **The time taken is average time for predicting on one sample.**

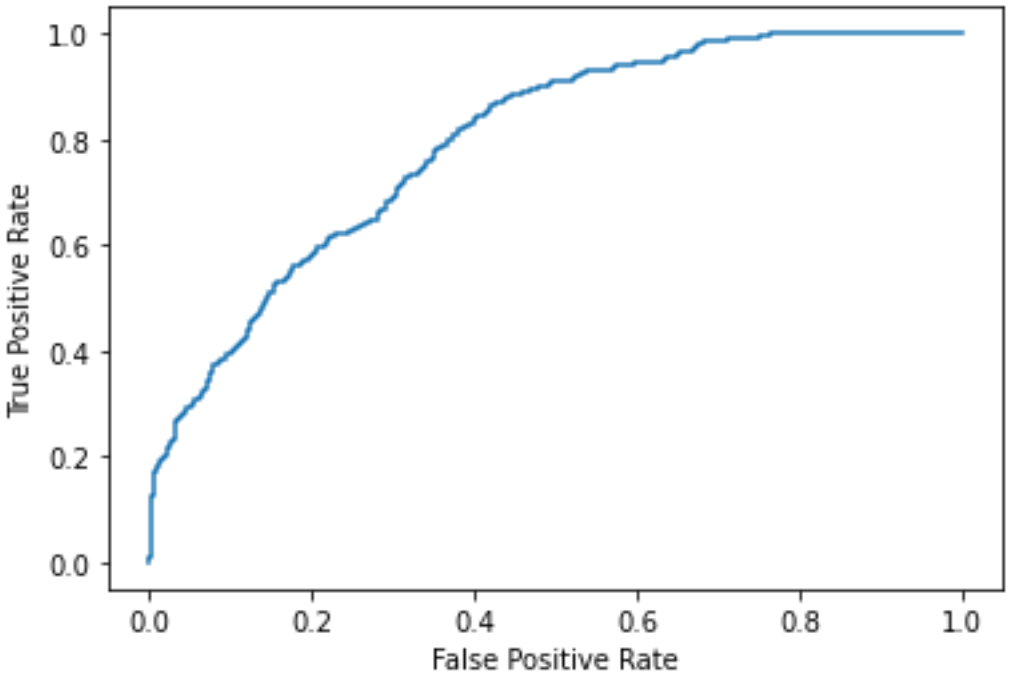
- **SVC – RBF:**

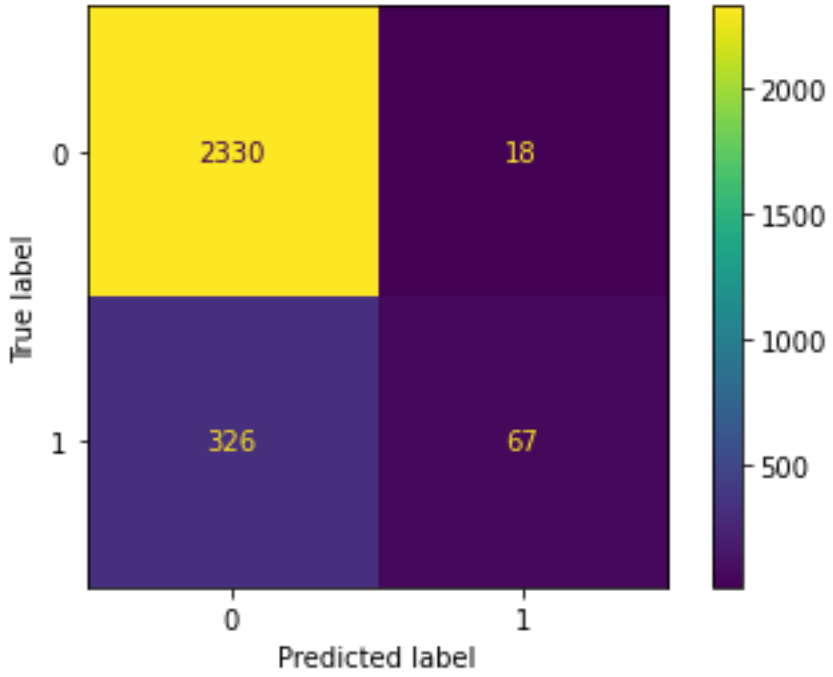





```
Time taken SVM rbf:  0.11929952572053995
1.0
[0.99843628 0.99296325 0.99843628 0.99843505 0.99608764]
              precision    recall  f1-score   support

          No       1.00      1.00      1.00      2348
         Yes       1.00      1.00      1.00       393

    accuracy                           1.00      2741
   macro avg       1.00      1.00      1.00      2741
weighted avg       1.00      1.00      1.00      2741
```

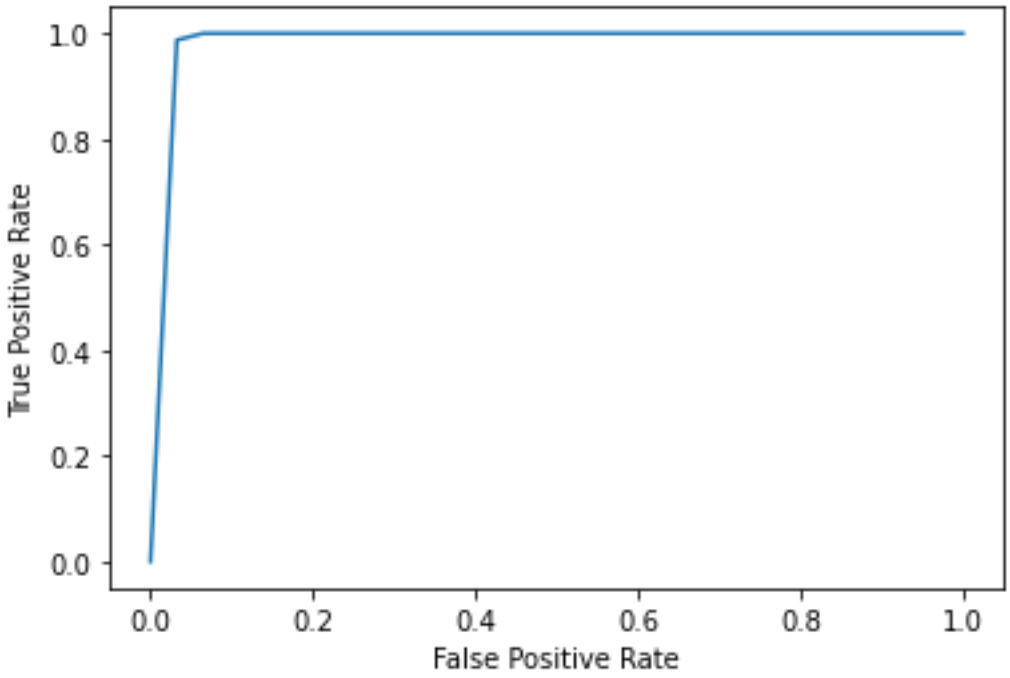
- **SVC linear**

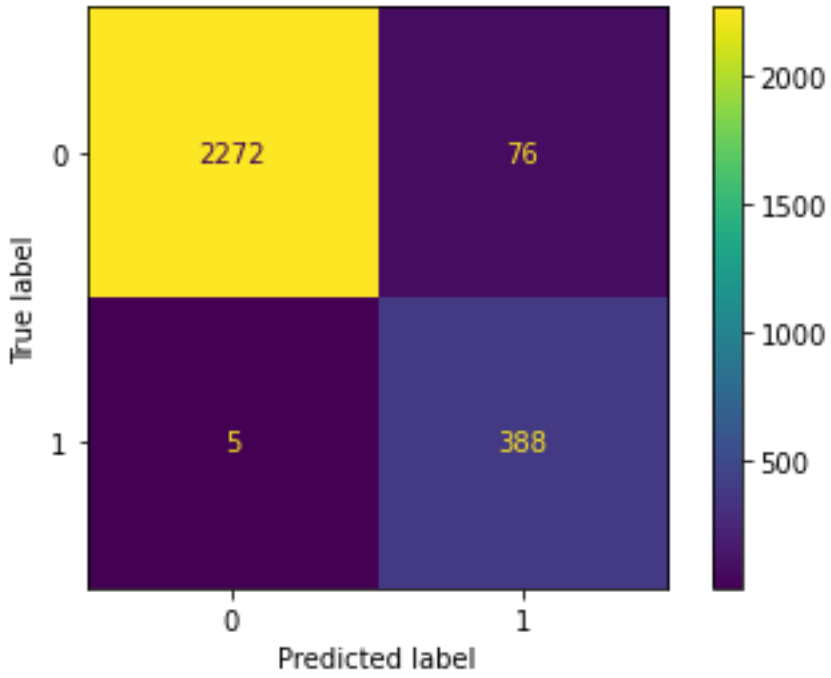

True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


2330
18
326
67
True label
Predicted label
0
1
2000
1500
1000
500

```
Time taken SVM linear:  0.002918642831083546
0.8744983582634075
[0.86942924 0.87177482 0.87255668 0.87245696 0.87089202]
              precision    recall  f1-score   support

          No       0.88      0.99      0.93      2348
         Yes       0.79      0.17      0.28       393

    accuracy                           0.87      2741
   macro avg       0.83      0.58      0.61      2741
weighted avg       0.86      0.87      0.84      2741
```

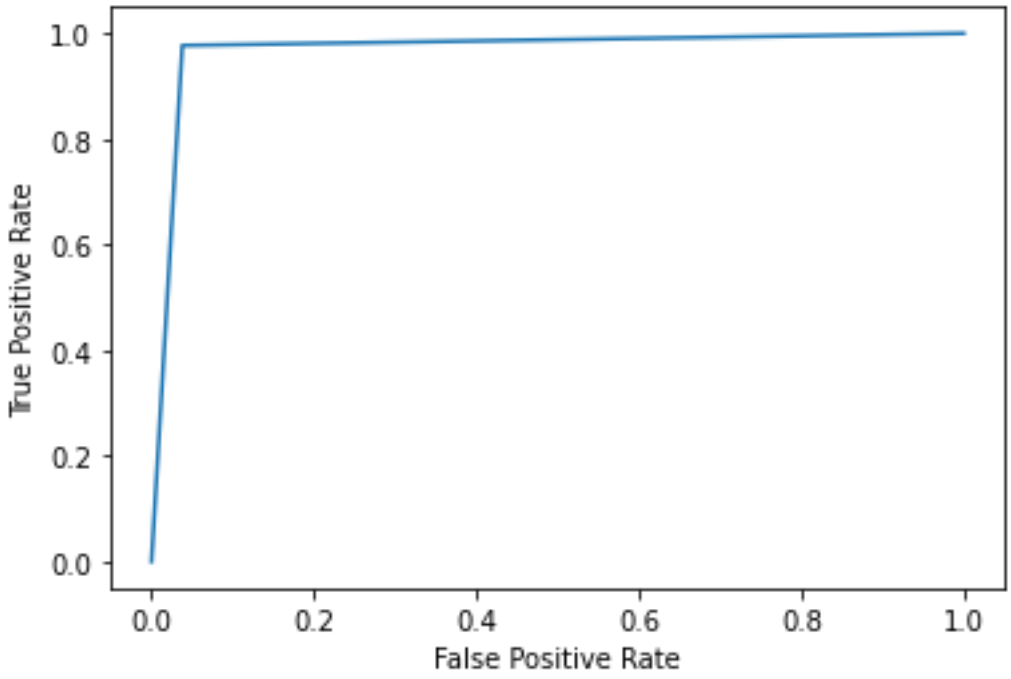
- **KNN**

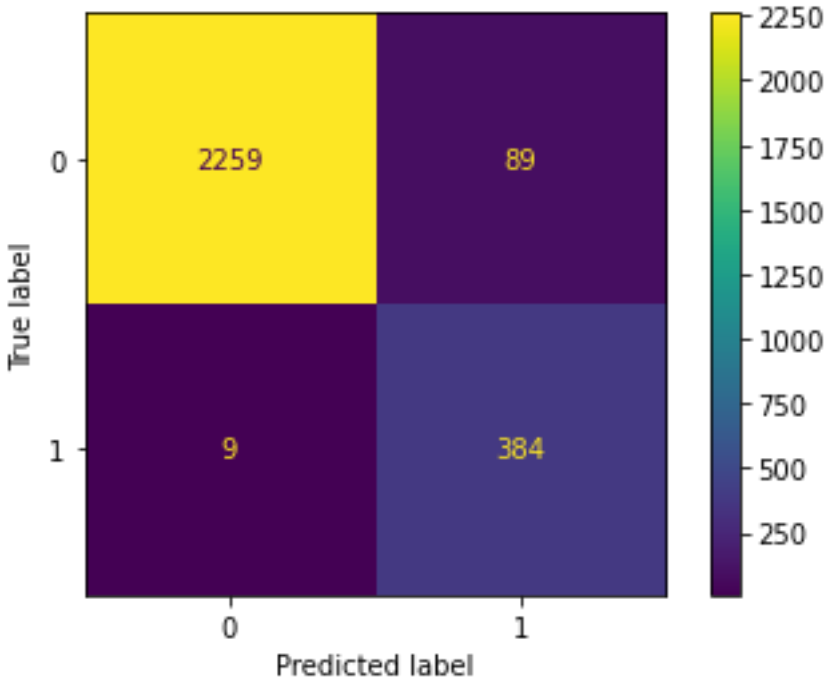





```
Time taken KNN:  0.12404232032105071
0.9704487413352791
[0.95230649 0.94370602 0.95856138 0.9514867  0.94679186]
0.950570489219742
              precision    recall  f1-score   support

          No       1.00      0.97      0.98      2348
         Yes       0.84      0.99      0.91       393

    accuracy                           0.97      2741
   macro avg       0.92      0.98      0.94      2741
weighted avg       0.97      0.97      0.97      2741
```

- **Decision Tree**





```
Time taken Decision Tree:  0.001459321415541773
0.9693542502736228
[0.95777952 0.94448788 0.94839719 0.95070423 0.94131455]
0.9693542502736228
              precision    recall  f1-score   support

          No       1.00      0.97      0.98      2348
         Yes       0.84      0.98      0.90       393

    accuracy                           0.97      2741
   macro avg       0.92      0.97      0.94      2741
weighted avg       0.97      0.97      0.97      2741
```

b. Justify different design decisions for each ML model used to answer this question.

A: SVM: For linear SVC the modifications are to feed in encoded data (does not work on categorical data) and try with different combinations of C, gamma and max_iter. The default value of max_iter did not converge so I had to modify it to 5000.

For RBF SVC the max_iter default is 1 which was changed to 230 to get convergence.

KNN: KNN cannot handle categorical data (distance cannot be calculated between categorical values) that is why it takes in one hot encoded data. Another design choice is to evaluate different values of K against cross validation accuracy vs training accuracy. This yields in the best K and minimizes variance and bias.

Decision Trees: For each of the models I have used one hot encoded data. Although, for decision trees we can also use data with categorical values but sklearn does not support categorical variables as of now for classification. This leads to sparse tree generation because the splitting happens more frequently in one hot encoded data.

A common pattern across all models used was to have stratified K Fold cross validation which would give realistic classification scores (as much as possible) and maintain similar class ration in all the runs of cross validation.

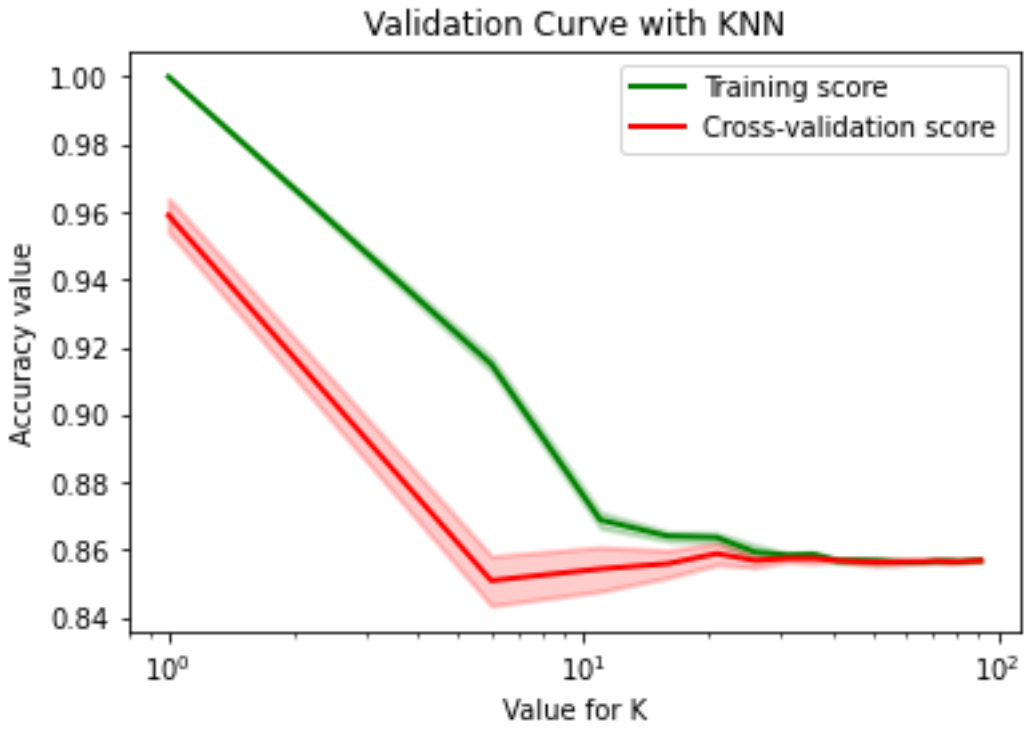
c. Have you optimised any hyper-parameters for each ML model? What are they? Why have you done that? Explain.

A: SVC – RBF: After trying different values of max_iter, the SVM converged at 230 (Highest value across all metrics). The intuition behind using high value of max_iter was that as the data which had 62 feature columns after encoding would become even more high dimensional after the kernel function maps it to higher dimension. The default value of 1 max_iter would not be sufficient to reach convergence.

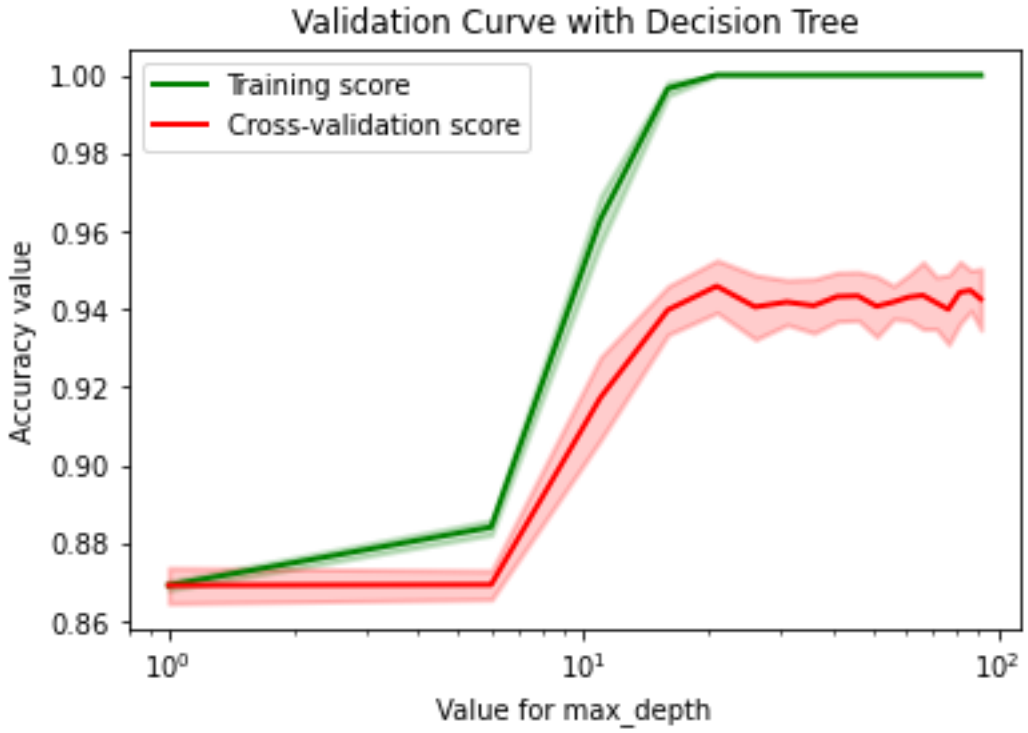
SVC – Linear: In this scenario the SVC could not attain higher values as the separating curve was in lower dimension and the data was not clearly separable. This caused the model to give lesser precision even when the value of C was increased. Apart from

this, I set dual =False as the number of features < number of samples, this reduces the computation time as n_samples x n_samples matrix is not computed.

KNN: For KNN, I plotted the cross-validation score with training score to get the best value of K. Here, we can see that the min gap between the cross validation and training score is at 30, though I have selected neighbours as 2 because lower K values generally result in more generic model as well as the cross validation and testing metrics were high for this K.



Decision Tree: For decision tree I plotted the max_depth parameter against the accuracy score. I did this because the depth of tree determines how much bias and variance would be present in the trained model. High depth = high variance/low bias, shallow tree = low variance/large bias. To achieve optimum balance I chose the combination with best scores across all metrics. This shows that the optimum depth is 6, but I chose 20 as it has higher cross validation scores.



d. Finally, make a recommendation based on the reported results and justify it.

A: If computation time is not a factor, then the recommended model would be SVM with RBF as it has highest precision-recall, accuracy and F1 score across all the used models. But if real world applications are concerned then decision tree would be more suited to predicting customer response as we can see that it also has a high precision-recall, accuracy and F1 score with very low computation time. This low computation time makes Decision Tree the best out of all reported models in a real-world scenario. Also, it might be the only model which can generalize well on removing class imbalance and not overfit.

The time taken is average time for predicting on one sample.

```
Time taken Decision Tree:  0.001459321415541773
0.9693542502736228
[0.95777952 0.94448788 0.94839719 0.95070423 0.94131455]
0.9693542502736228
              precision    recall  f1-score   support

          No       1.00      0.97      0.98      2348
         Yes       0.84      0.98      0.90       393

    accuracy                           0.97      2741
   macro avg       0.92      0.97      0.94      2741
weighted avg       0.97      0.97      0.97      2741
```

6. Build three ensemble models for predicting customer response

A: I have built following three models:

- Random Forest
- Bagging (Decision Tree as base classifier)
- Gradient Boosting

a. When do you want to use ensemble models over other ML models?

A: Most common scenario to use ensemble models is when there is a requirement to boost accuracy and we have a collection of weak classifiers. This makes the ensemble model stronger than the weaker models and there is a boost in accuracy. The other major factor is when we want to reduce variance of prediction errors by the constituent models. However, one must be careful not to mix good single models with bad performing single models otherwise the resultant ensemble will be worse than the best performing model.

b. What are the similarities or differences between these models?

A: Similarity between all ensemble models is that they reduce variance of prediction errors and boost accuracy of low variance and high bias models (convert weak learners to strong learners).
The dissimilarity is in the way of aggregation employed by different ensemble models:

Bagging: It averages the different classifier which constitute a bagging classifier ensemble.

Boosting: This ensemble methods takes in the results of previous classifiers and feeds the error to the successor. In other words, eahch following classifier tries to correct the previous classifier.

Random Forest: Random Forest is a modification of Bagging algorithm specifically designed to work best with averaging decision trees. The major difference from bagging is that Random Forest decorrelates the constituent trees.

c. Is there any preferable scenario for using any specific model among set of ensemble models?

A: Bagging: We can generally use bagging when we must give regression values or when we want to average results of non Decision Tree models. This can also be used when we want to minimize variance.

Boosting: Boosting can be used when there is a need to improve accuracy by combining weaker models, this can generally result in higher boost in accuracy than averaging based models.

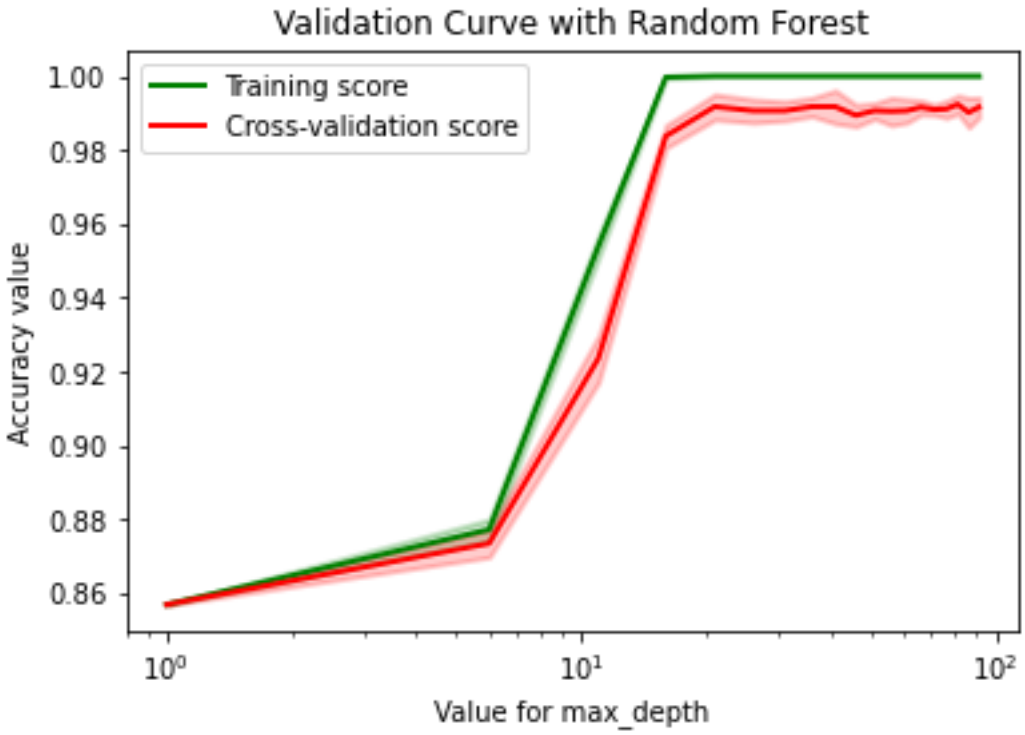
Random Forest: Random forests work best if we have there is noisy data or class imbalance in data.

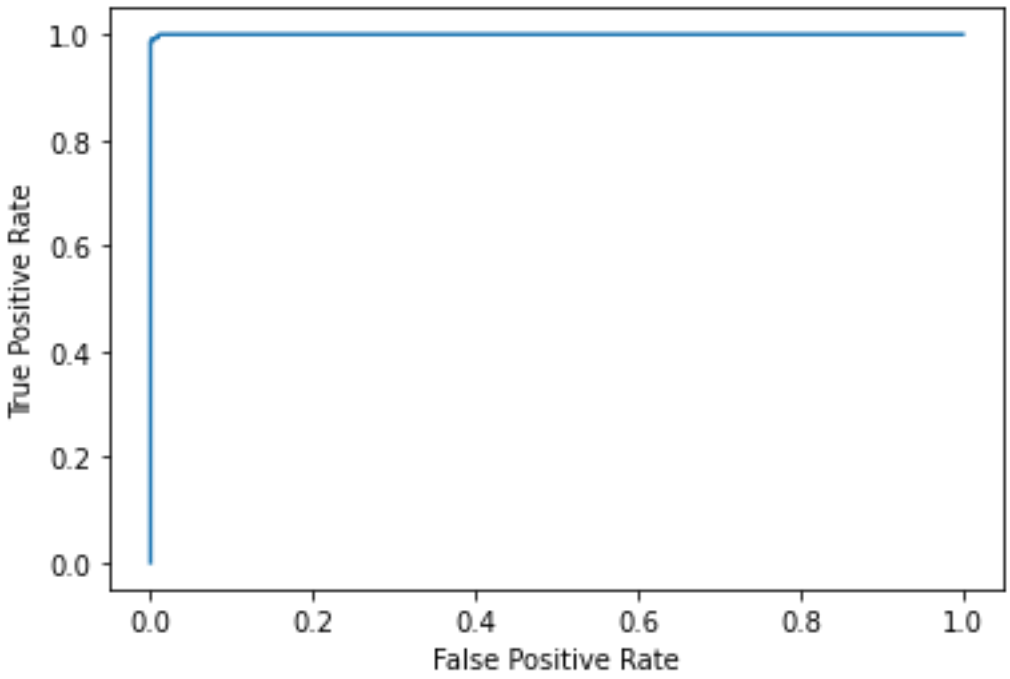
d. Write a report comparing performances of models built in question 5 and 6. Report the best method based on model complexity and performance.

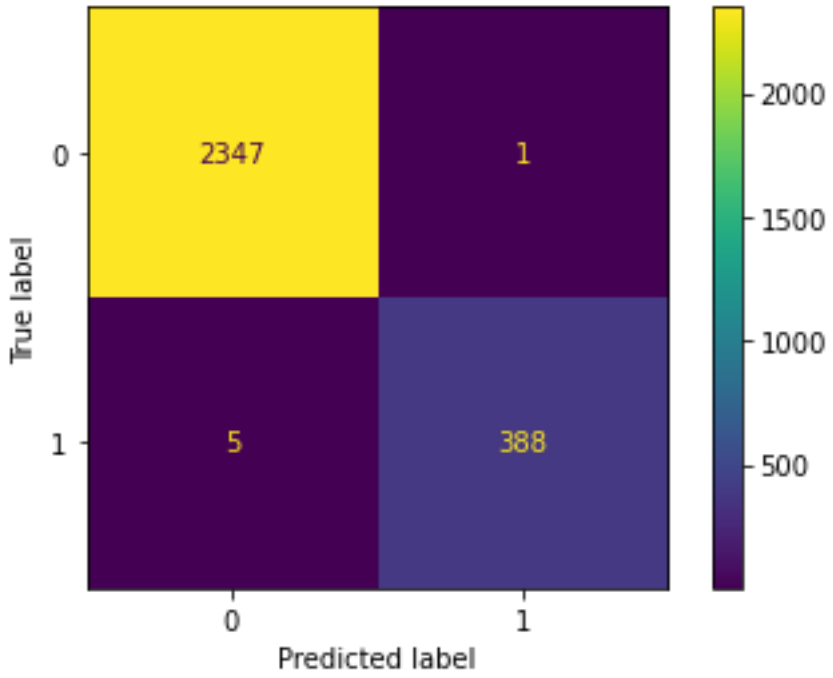
A: Combining the performance scores of the ensemble models across same metrics in 5 (a)

- **ROC curve** for Response “Yes”
- **Confusion matrix** showing distribution of predictions
- **Average time taken, accuracy, precision, recall and f1 score**
- **The time taken is average time for predicting on one sample**

Random Forest:

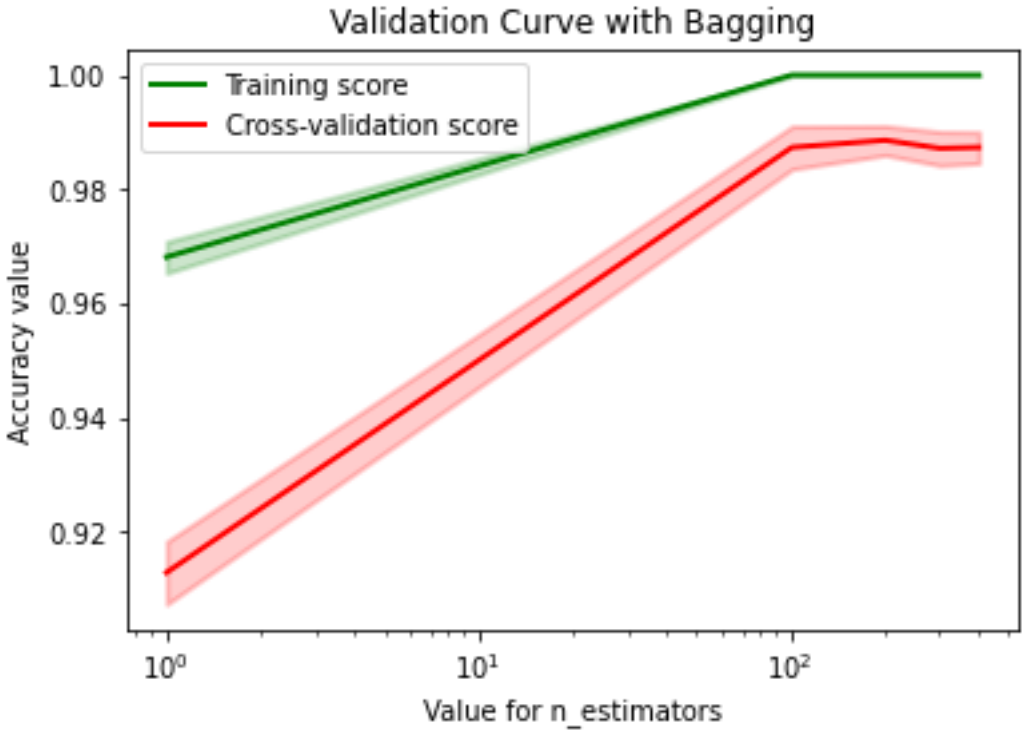


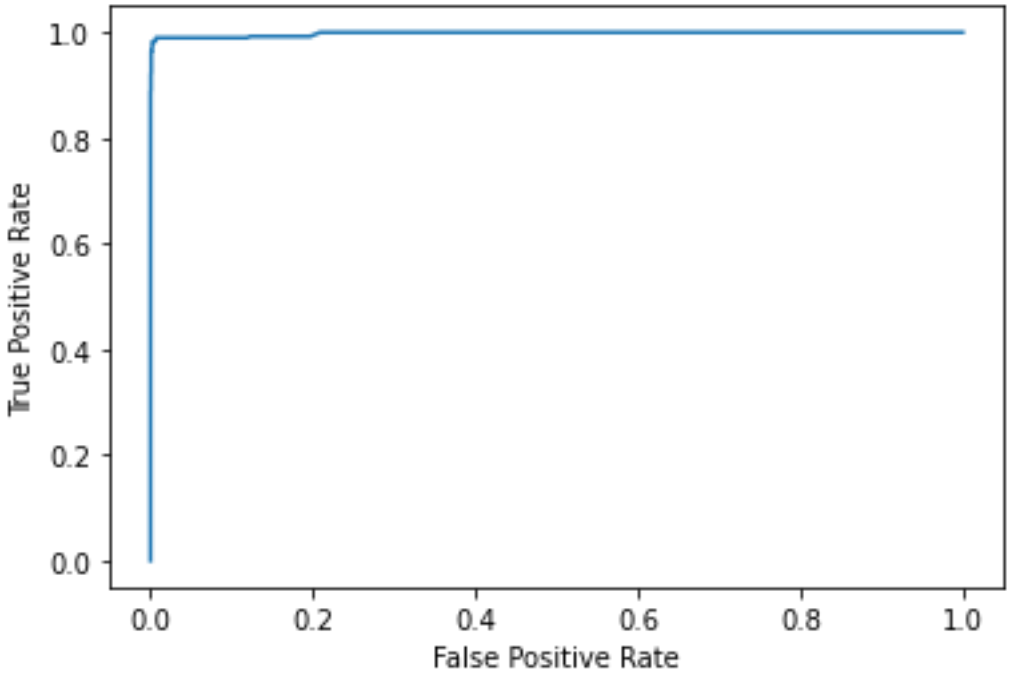


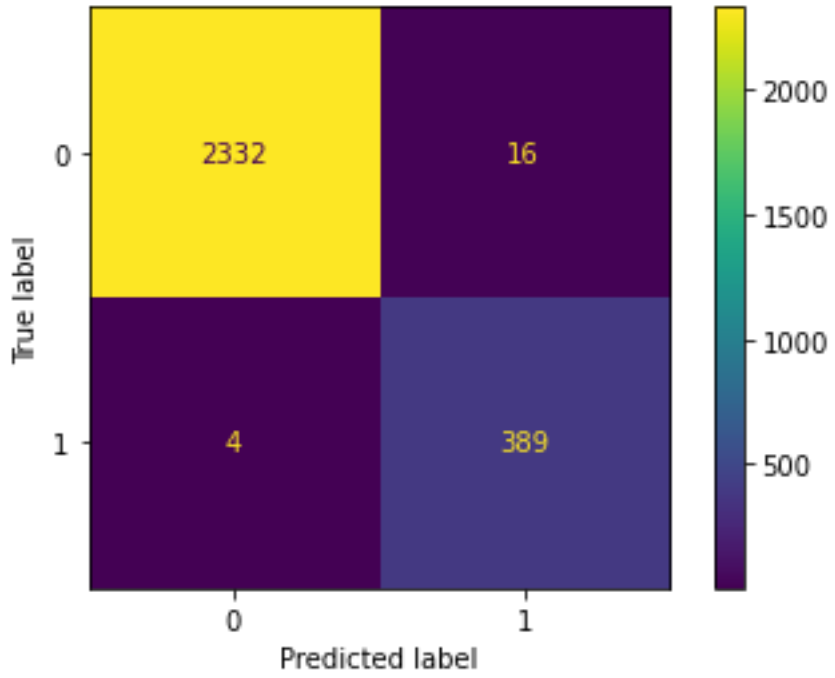



```
Time taken Random Forest:  0.015322874863188618
0.9978110178766874
[0.98983581 0.98592651 0.99374511 0.98982786 0.98669797]
0.9892066498548233
              precision    recall  f1-score   support

          No       1.00      1.00      1.00      2348
         Yes       1.00      0.99      0.99       393

    accuracy                           1.00      2741
   macro avg       1.00      0.99      1.00      2741
weighted avg       1.00      1.00      1.00      2741
```

Bagging:

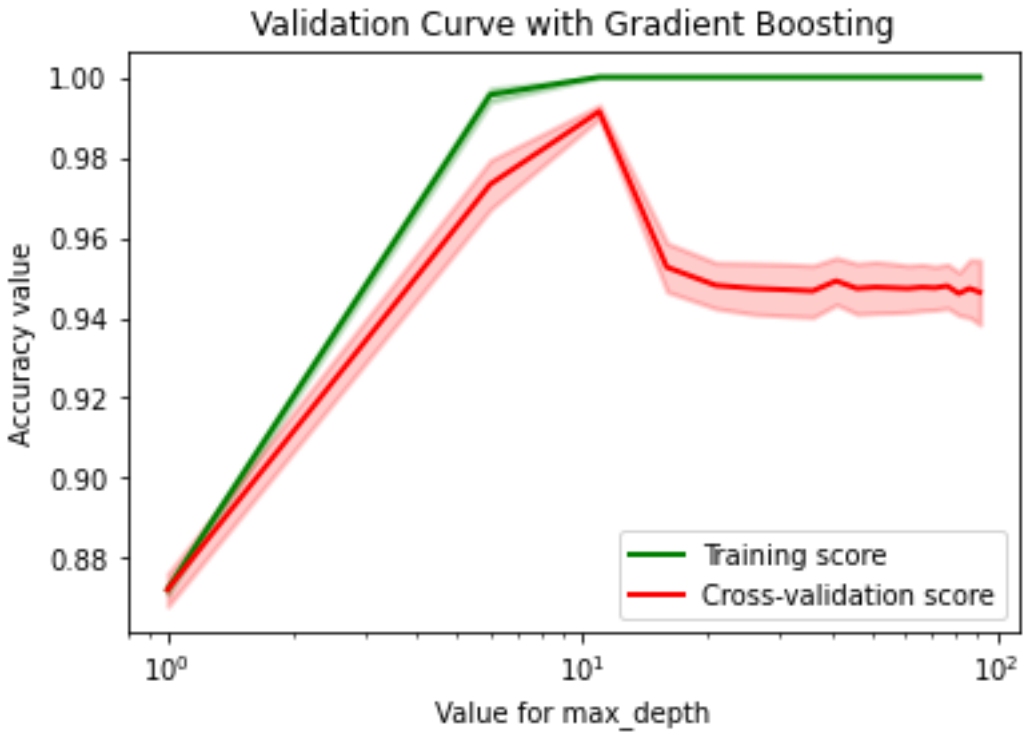

Validation Curve with Bagging
Training score
Cross-validation score
Accuracy value
1.00
0.98
0.96
0.94
0.92
10^0
10^1
10^2
Value for n_estimators

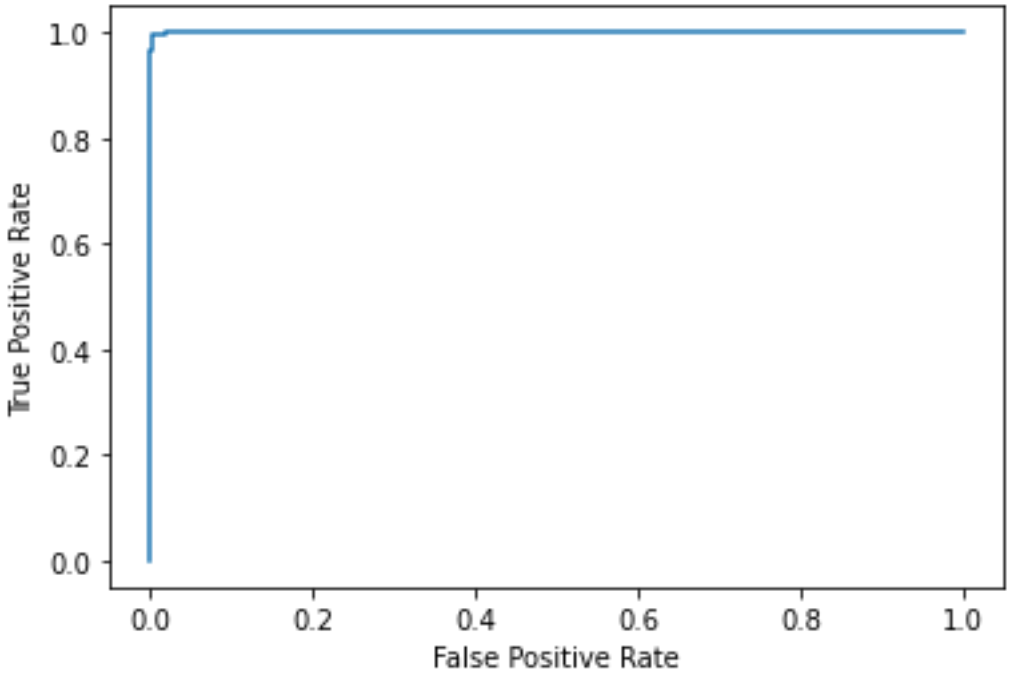

True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

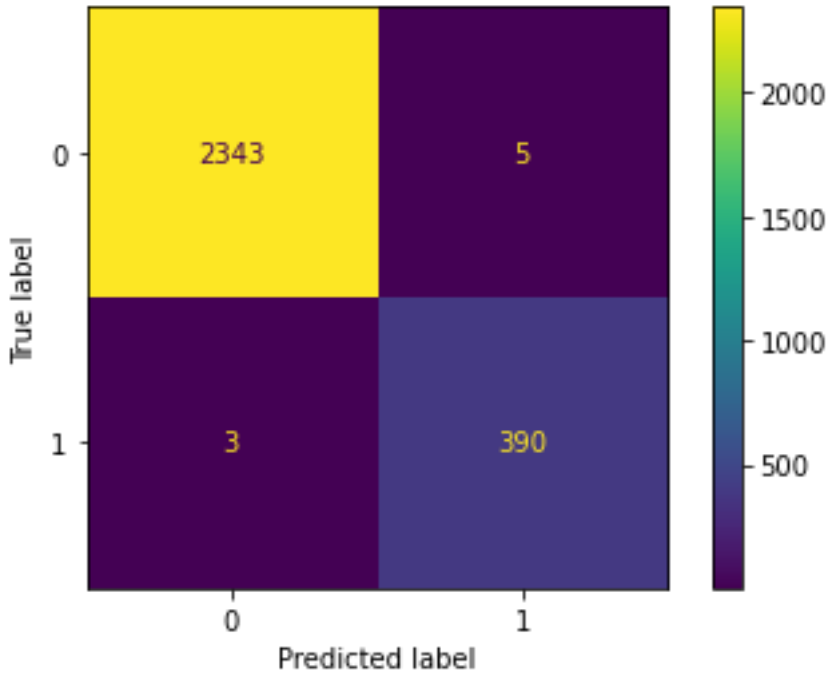



```
[0.98592651 0.97967162 0.99061767 0.9913928  0.9827856 ]
              precision    recall  f1-score   support

          No       1.00      1.00      1.00      2348
         Yes       0.99      0.99      0.99       393

    accuracy                           1.00      2741
   macro avg       0.99      1.00      0.99      2741
weighted avg       1.00      1.00      1.00      2741

Time taken Bagging:  0.0791681867931412
Accuracy:  0.9927033929222912
```

Gradient Boosting:








```
Time taken Gradient Boost:  0.010215249908792412
0.9970813571689164
[0.99452697 0.99374511 0.99139953 0.98982786 0.98356808]
0.9906135099188651
              precision    recall  f1-score   support

          No       1.00      1.00      1.00      2348
         Yes       0.99      0.99      0.99       393

    accuracy                           1.00      2741
   macro avg       0.99      1.00      0.99      2741
weighted avg       1.00      1.00      1.00      2741
```

We can see that how ensemble models outperform single ML models by averaging out the results. So, it stands to reason that one of the ensemble models is used. We can also see from the experiments that cross validation can be a great tool to select the optimum parameters for nearly all ML models.

In this scenario, based on the execution time (fastest for Gradient boost and Random Forest in ensemble category) we should either choose Grad boost or Random Forest as the classification metric is nearly similar. Now, as there is a imbalance in the dataset and RF is less likely to overfit than GradBoost, so in this scenario **Random Forest** has an edge over all the other models and comes out as **the best model**.

e. Is it possible to build ensemble model using ML classifiers other than decision tree? If yes, then explain with an example.

A: Yes. There are multiple ways to create ensemble model without using decision tree:

Bagging: (Example in code). We can use bagging with SVC. This would train multiple SVC and then aggregate the results of all the constituent SVCs. I have not optimized the parameters, just created a sample to demonstrate how bagging can be used with SVC.

Other methods are:

Voting classifiers: Different classifiers vote to give final output.

Stacking classifiers: Different classifiers are stacked to create a single ensemble model.