HW2

Task 1

1.
The dataset contains 12 features. Among them, “PassengerId”, “Name”, “Ticket”, and “Cabin” should not have any impact on the survival factor of a passenger. So, these features are removed.

Class data are converted to numerical data, since the scikit-learn library expects numerical data in it’s input and target. In the “Sex” column, “male” and “female” are converted to -1 and 1.

In the “Embarked” column, C, Q, and S are replaced with 1, 2, and 3, respectively.

Two new features were calculated based on the suggestions from the original dataset source. One is “Family size” (Fsize) and the other one is called “Fare per person” (FareP). The family size is defined as, “Sibling/Spouse” (SibSp) + “Parent children” (Parch). The fare per person is the fare divided by the total number of people in a family.

And finally the rows with absent values are removed from the final dataset.

2.
A decision tree classifier model is trained on a subset of the preprocessed dataset and tested on a remaining subset. The train-test split is done with 80-20 share. These data points are sampled randomly to get a even sampling throughout the dataset. This train-test split is done to do the fine-tuning of the model training process.

At the fine-tuning stage, the model depth is tuned. Maximum train accuracy was found at 15 max depth. But the test accuracy was found maximum when the model depth was set to be 13. So, 13 was selected for the model depth.

Finally the train and test accuracy was found to be 96.66% and 76.22% respectively. The tree found from the training process is given below.

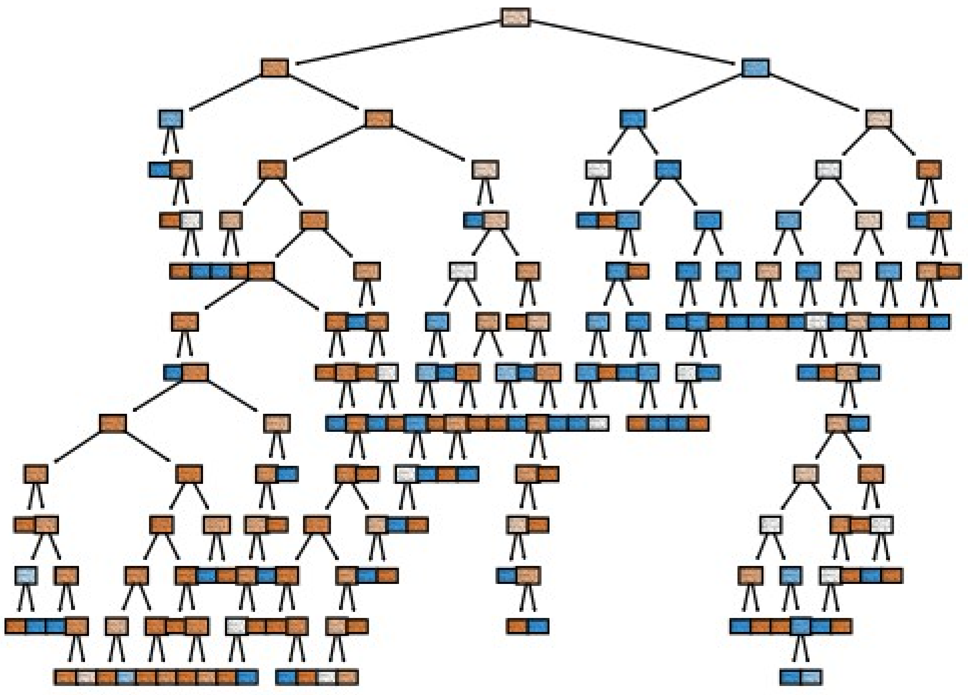


Figure 1: Decision tree after training using the Titanic dataset. (Please see the “DT.svg” file from the link provided at the end of this report for a better resolution.)

3.

After applying 5-fold CV, the average training and testing accuracy for the decision tree model were found to be 97.40% and 77.25%. The accuracy in each split is reported below -

```
Split:  0
Train accuracy:  97.36379613356766 %
Test accuracy:  72.72727272727273 %
Split:  1
Train accuracy:  96.66080843585237 %
Test accuracy:  82.51748251748252 %
Split:  2
Train accuracy:  96.49122807017544 %
Test accuracy:  74.64788732394366 %
Split:  3
Train accuracy:  98.42105263157895 %
Test accuracy:  79.5774647887324 %
Split:  4
Train accuracy:  98.0701754385965 %
Test accuracy:  76.76056338028168 %
```

4.
Applying 5-fold CV for a random forest model, the average training and testing accuracy values were found to be 98.88% and 79.08%, respectively. The number of trees for training the random forest model was selected to be 100. The accuracy metric in each fold is reported below -

```
Split:  0
Train accuracy:  99.29701230228471 %
Test accuracy:  73.42657342657343 %
Split:  1
Train accuracy:  98.76977152899823 %
Test accuracy:  80.41958041958041 %
Split:  2
Train accuracy:  98.7719298245614 %
Test accuracy:  80.98591549295774 %
Split:  3
Train accuracy:  98.7719298245614 %
Test accuracy:  80.28169014084507 %
Split:  4
Train accuracy:  98.7719298245614 %
Test accuracy:  80.28169014084507 %
```

5.
Comparing these two algorithms, it can be confirmed that the Random Forest algorithm is better than the Decision tree algorithm. Though the statement is evident from the experiments, the random forest model is far more robust compared to the decision tree model. Because, the random forest model generates a number of decision trees along with other techniques to produce the result using ensemble methods.

Task 2

(a)
To calculate the training error rate for the tree, we need to calculate the number of misclassified samples and total number of data points in all the leaf nodes.

Number of misclassified samples = 5+6+2+6+5+5 [misclassified samples from the left most to the right most leaf nodes]

= 29

Total number of samples = 19+13+12+14+22+20 = 100

So, training error rate = (No. of misclassified samples) / (No. of total samples)

= 29/100

= 0.29 = 29%

(b)
Given the test point, T = {A=0, B=1, C=1, D=1, E=0} we need to traverse from the root node of the decision tree towards a leaf node. The root node of the tree decides if A is equal to 0 or 1. Since in the test data, A = 0, we need to move to the left child node (B). Then using the same procedure all the nodes need to be traversed. And finally we will reach a leaf node having majority of samples for a specific class. In this test case the nodes A, B, and then E is traversed and then the left child node is found from the tree. That leaf node highest number of samples for the (-) class. Hence the decision from that tree should be (-).

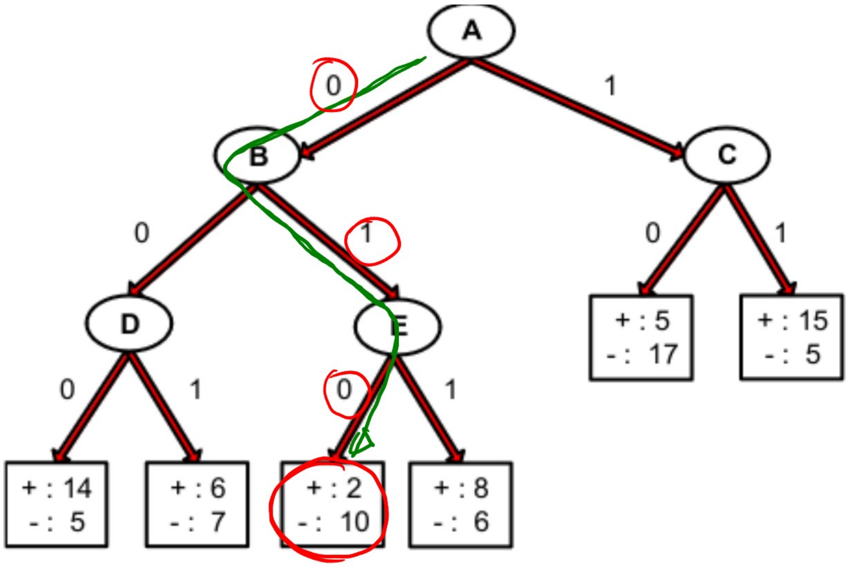
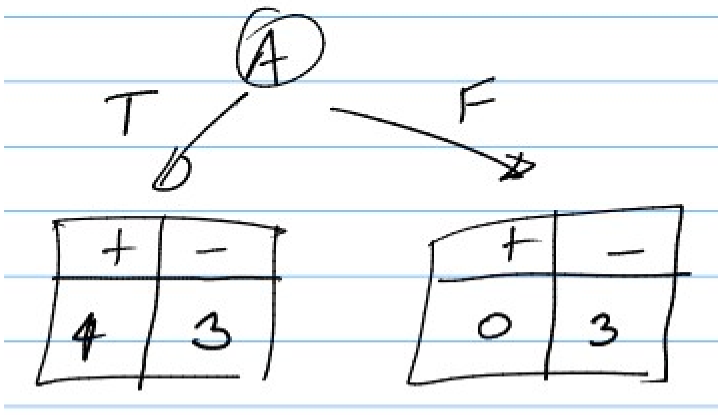


Figure 2: Decision process for a given test data point.

Task 3

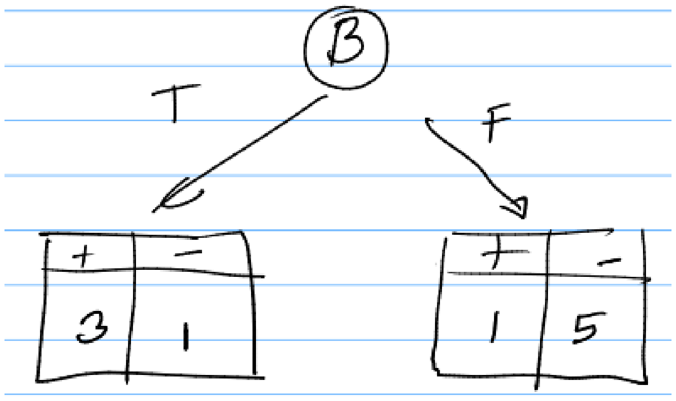
Q1.
Before splitting the overall gini is = $1 - (4/10)^2 - (6/10)^2 = 0.48$

Q2.
Gini gain after splitting on A



Gini for A-true would be = $1 - (4/7)^2 - (3/7)^2 = 0.48979591$
Gini for A-false would be = $1 - (0/3)^2 - (0/3)^2 = 0.0$
So, the gini gain for splitting on A = $0.48 - (7/10)*0.48979591 - (3/10)*0$
$= 0.137142$

Q3.
In a similar way we can calculate the gini gain after splitting on B = $0.48 - (4/10)*(1 - (3/4)^2 - (1/4)^2) - (6/10)*(1 - (1/6)^2 - (5/6)^2)$
$=0.163333$



Q4.
Since the gini gain is higher for splitting on B, the decision tree should choose B to split.

Task 4

Q1.
No, decision trees are non-linear classifiers. In a decision tree each node corresponds to a feature based decision and hence creates a subset of the complete dataset. Then the tree work on the subset of the dataset to branch out on that dataset again, resulting in a piecewise like classification method.

Q2.
Both of the error metrics, Gini and misclassification error have the range of 0 to 0.5. But the Gini is sensitive to impurity. So, the Gini is better than misclassification error.

Task 5

In bagging, the features and samples are uniformly sampled (with replacement) to split a node in a tree. This results in complex trees with overfitting problem. Hence, changing the dataset with completely change the tree structure. And complex tree structure will also need more time to train and evaluate.

Random forest mainly reduces the overfitting problem of bagging by introducing randomness into the bootstrapping (sampling) process. Contrary to the default bagging process, RF creates a subset of features m of the whole dataset features M. This introduces randomness not only in the data sample selection process, but also in the process of feature selection reducing the overfitting effect seen in bagging.

Task 6

After calculating x1x2 for the given dataset we can get,

x1	**x2**	**x1x2**	**Class**
-1	-1	+1	Negative
-1	+1	-1	Positive
+1	-1	-1	Positive
+1	+1	+1	Negative

We can show the SVC designed for the x1 – x1x2 space as follows,

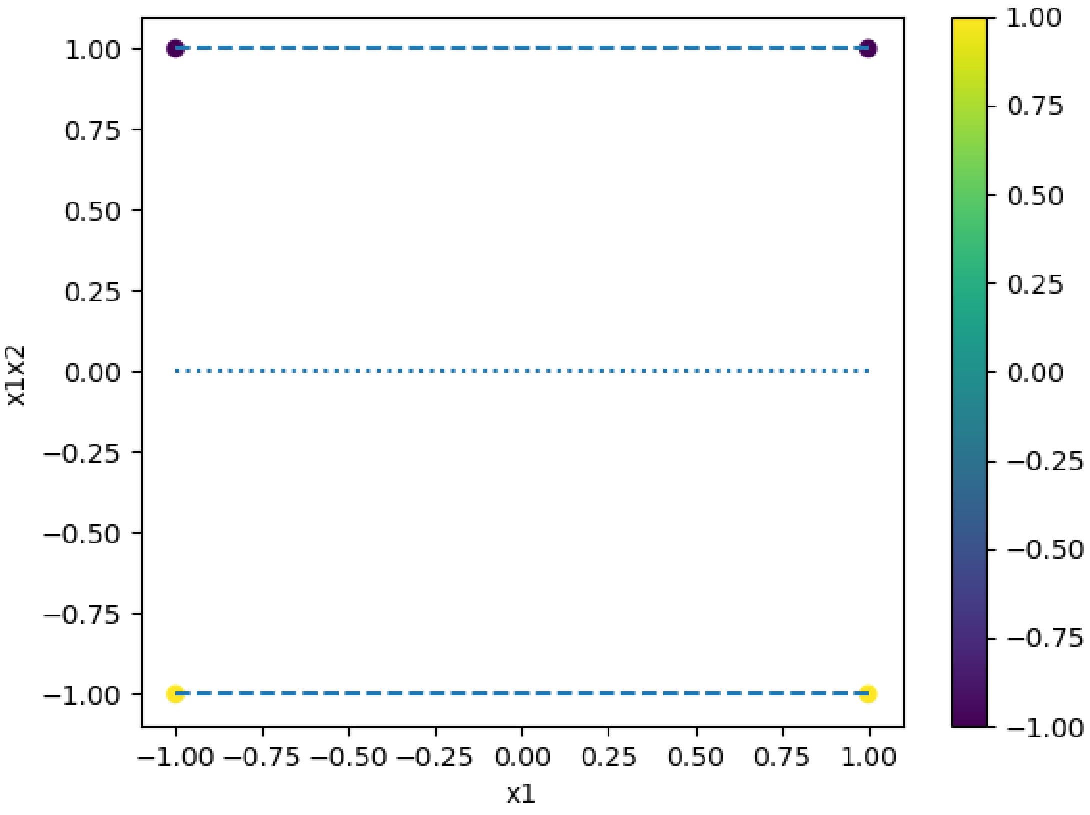


Figure 3: SVC margin. The dotted line shows the hyperplane and dashed lines are the support vectors.

The margin which is the distance from the hyperplane to the support vectors would be 1.

Task 7

The given equation of circle is (x1-a)^2 + (x2-b)^2 – r^2 = 0

Expanding the equation yields,
=> x1^2 – 2x1a + a^2 + x2^2 – 2x2b + b^2 – r^2 = 0
=> x1^2 + x2^2 – 2x1a – 2x2b + (a^2 + b^2 – r^2) = 0

This expanded form can be seen to have quadratic terms, linear terms of x1 and x2. And the equation is itself linear with these terms. Hence, having a feature space with x1, x2, x1^2, x2^2 can be linearly separable using a linear classifier (e.g., linear SVC).

Task 8

Using a polynomial kernel, the data from the equation of ellipse can be mapped into the kernel feature space.

Given a polynomial kernel of order 2, we can get,

$$K(u,v)=(1+u\cdot v)^2=\sum_{i,j=1}^{n}(u_i u_j)(v_i v_j)+\sum_{i=1}^{n}(\sqrt{2}u_i)(\sqrt{2}v_i)+1$$

Here, the n is the number of features in the original dataset. In this case, the number of features, n = 2. So, the feature mapping $\phi(x)$ generated from this kernel function is,

$$\phi(x)=[x_1^2, x_2^2, x_1 x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1]$$

Now, expanding the equation of ellipse yields,

$$c(x_1-a)^2+d(x_2-b)^2-1=0$$
$$c\,x_1^2-c\,x_1 a+c\,a^2+d\,x_2^2-d\,x_2 b+d\,b^2-1=0$$

This expanded equation contains a linear form of expression containing x1^2, x2^2, x1, x2, and constant terms. Hence, using the polynomial kernel the ellipse equation can be solved using linear SVC.

Moreover, using the polynomial kernel of second degree we can get the same feature spaces of [1, x1, x2, x1^2, x2^2, x1x2]. So, these two are equivalent feature spaces. And a linear SVC can separate elliptic region using these feature spaces.

Github link: https://github.com/ShifatHossain/CAP5610-23Spring0001/tree/main/HW2