ML_HW3

Javad Kavian
810100103

July 28, 2024

Question 1

To find the probability of truly determining crime of criminal, we have to first select the number of judges who decided truly and multiply it by its probability:

1-

$$p = \binom{5}{3} \times (0.7)^3 \times (0.3)^2 + \binom{5}{4} \times (0.7)^4 \times (0.3)^1 + \binom{5}{5} \times (0.7)^5 \times (0.3)^0 = 0.8369199$$

2-

$$\begin{aligned} p = \binom{9}{5} \times (0.7)^5 \times (0.3)^4 + \binom{9}{6} \times (0.7)^6 \times (0.3)^3 + \\ \binom{9}{7} \times (0.7)^7 \times (0.3)^2 + \binom{9}{8} \times (0.7)^8 \times (0.3)^1 \\ + \binom{9}{9} \times (0.7)^9 \times (0.3)^0 = 0.901191 \end{aligned}$$

3-

$$P_N = \binom{N}{\frac{N+1}{2}} (0.7)^{\frac{N+1}{2}} (0.3)^{\frac{N-1}{2}} + \binom{N}{\frac{N+3}{2}} (0.7)^{\frac{N+3}{2}} (0.3)^{\frac{N-3}{2}} + ... + \binom{N}{N} (0.7)^N (0.3)^0$$

$$\lim_{N \to \infty} P_N = ?$$

To find the above limit, we know that binomial distribution tends to normal when the number of samples go to infinity.thus we have the following distribution:

$$\mu = N \times p = 0.7N$$

$$\sigma^2 = N \times p \times q = 0.21N$$

$$p = \mathcal{N}(\mu, \sigma^2)$$

$$= \frac{1}{\sqrt{2\pi} 0.21N} exp(-\frac{(x - 0.7N)^2}{2 \times 0.21N})$$

$$\lim_{p \to \infty} p = 1$$

4-

$$p = \binom{5}{3} \times (0.5)^3 \times (0.5)^2 + \binom{5}{4} \times (0.5)^4 \times (0.5)^1 + \binom{5}{5} \times (0.5)^5 \times (0.5)^0 = 0.5$$

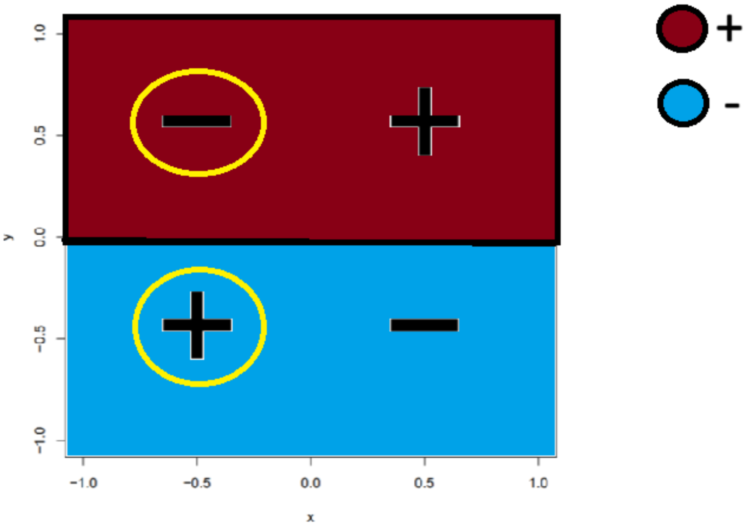
In this case, the value of N does not matter and we will always get the probability of 0.5

Question 2

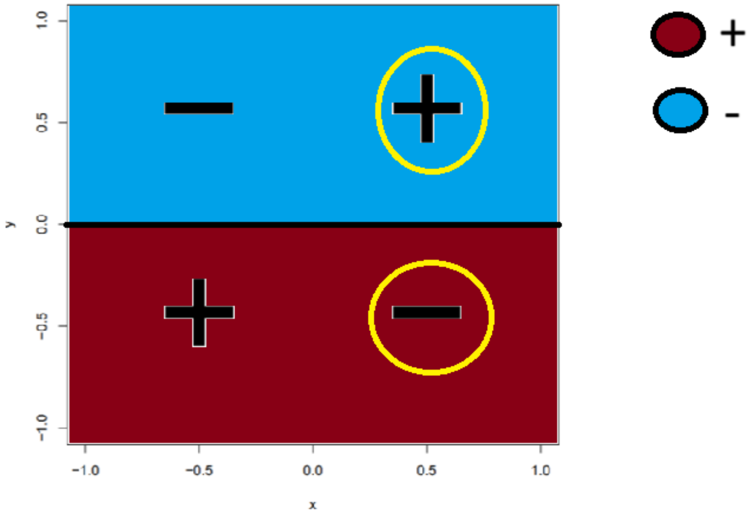
A-

1-

For this dataset, the decision stump will have 50% error;Because it will have to just select one feature between x and y and select a thereshold.For example suppose it chooses $y = 0$ as decision boundary.Then the feature space will look like:



As we can see, the points rounded by yellow circle are classified wrongly;though their weights will increase and the second classifier will do its best to classify them correctly.Consequently the decision boundary will be like:



As you can see, again the points rounded by yellow circles are misclassified and their weights increases and the decision boundary will be something like the previous part and this process is like an infinit loop and it does not matter how many classifiers we use, we will always have 50% learning error rate

2-

If we consider $\gamma = \frac{1}{2} - \epsilon_t$ as the edge of the classifier, we have:

$$learningError = \frac{1}{N}\sum_{i=1}^{N} I(H(x_i) \neq y_i) = \sum_{i=1}^{N} p_1(i) I(y_i F(x_i)) \leq \sum_{i=1}^{N} p_1(i)\exp(-y_i F(x_i))$$

Now lets write the update rule of p_t which is:

$$p_{T+1}(i) = p_1(i)\prod_{t=1}^{T}\frac{\exp(-y_i\alpha_t h_t(x_i))}{Z_t} = \frac{p_1(i)\exp(-y_i F(x_i))}{\prod_{t=1}^{T} Z_t}$$

So the learning error rate is bounded by $\prod_{t=1}^{T} Z_t$

$$Z_t = \sum_{i=1}^{N} p_t(i)\exp(-y_i\alpha_t h_t(x_i))$$

$$= \sum_{i\in C} p_t(i)\exp(-\alpha_t) + \sum_{i\in M} p_t(i)\exp(\alpha_t)$$

$$= (1 - \epsilon_t)\exp(-\alpha_t) + \epsilon_t \exp(\alpha_t)$$

$$= \sqrt{1 - 4\gamma_t^2} \leq \exp(-2\gamma_t^2).$$

When the error rate drops below $\frac{1}{N}$, we have 0 learning error rate.So we have:

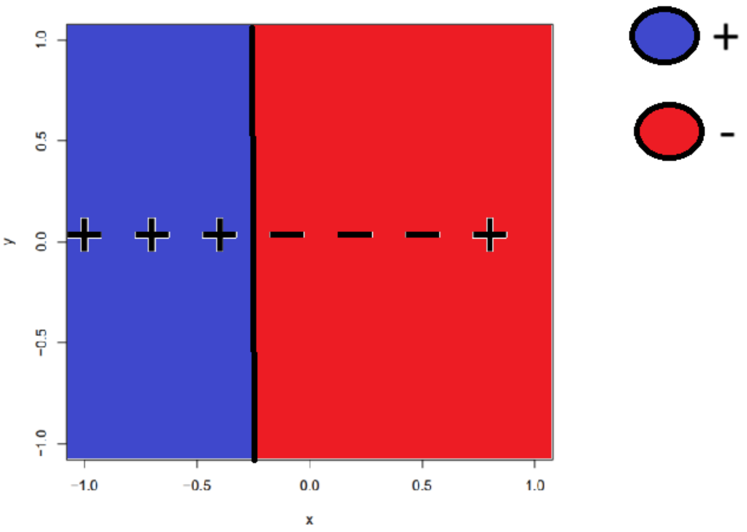
$$\exp(-2T\gamma^2) < \frac{1}{N}$$

$$\Rightarrow T > \frac{\ln(N)}{2\gamma^2}$$

B-

3-

It is obvious that decision boundary of h_1 will be like:



4-

As we can see in the above figure, the rightmost sample is misclassified.As we are in the first iteration, the weight are uniformly distributed and using the below formula, we can calculate learning error rate:

$$\epsilon_T = \frac{\sum_{i \in M} w_i^{T-1}}{\sum_{i \in C} w_i^{T-1}}$$

which M and C are set of misclassified and correctly classified samples by the T-th estimator:

$$\epsilon_1 = \frac{\frac{1}{7}}{1} = \frac{1}{7} = 0.14$$

and using the following formula:

$$\alpha_T = \frac{1}{2}\ln(\frac{1-\epsilon_T}{\epsilon_T})$$

$$\alpha_1 = \frac{1}{2}\ln(\frac{1-0.14}{0.14}) = 0.8958$$

and the accuracy of model is:

$$accuracy = \frac{6}{7} = 0.85$$

5-

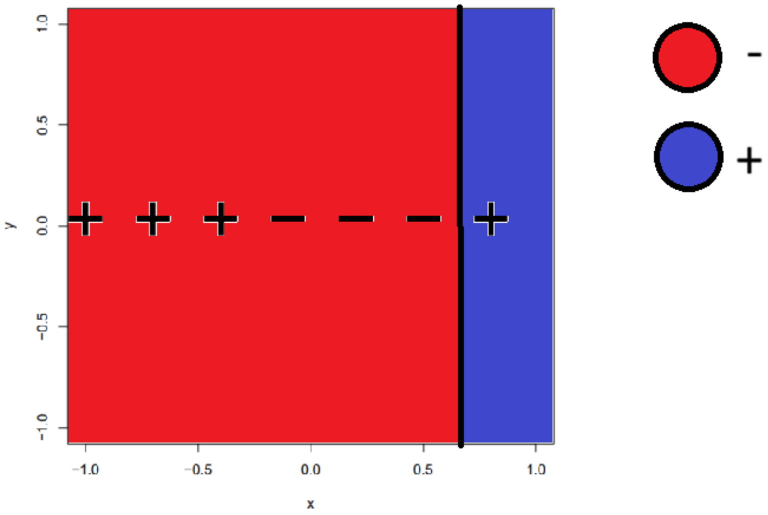
We know that weights of samples are updated by the formula below and then be normalized:

$$w_i* = \exp(\alpha_T) I(y_i \neq h_T(x_i))$$

using the above update rule, we get the following weights:
weights = [0.11835034, 0.11835034, 0.11835034, 0.11835034, 0.11835034, 0.11835034, 0.28989795]
as you see, the weight for the misclassified sample has increased.

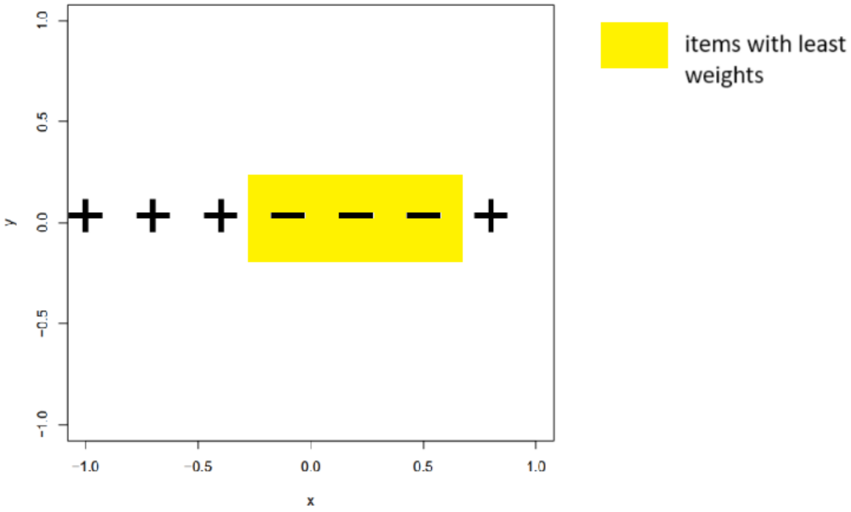
6-

As predicted, the weight of the last sample has increased and the decision stump will do its best to classify it correcty.thus we will have the following decision boundary:



7-

After the second iteration, the first three + samples are misclassified;thus we expect their weights to increase.But in both iterations, all - samples were correctly classified and thus we expect their weights to be less than others:

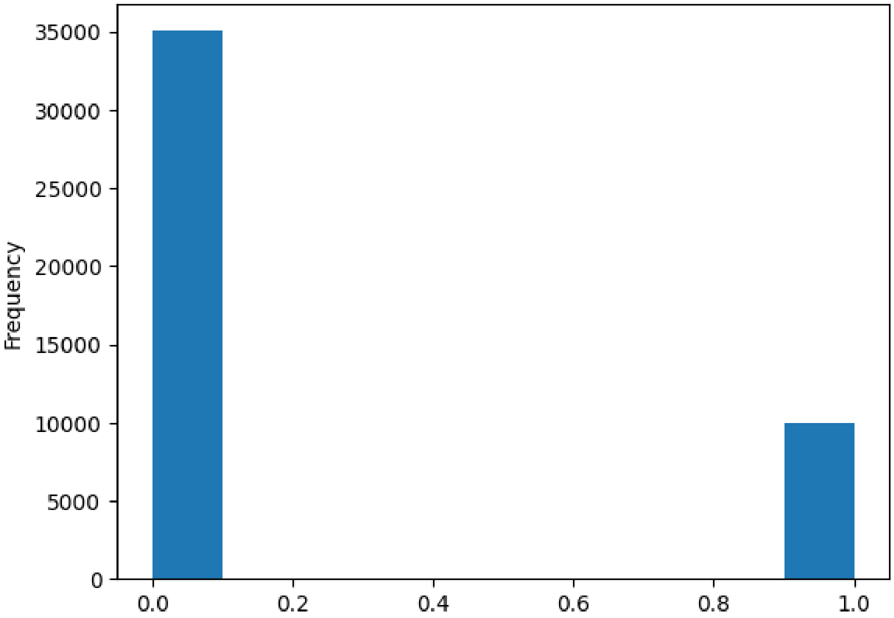


8-

As we saw, the number of misclassifications in the second iteration increased and thus we got lower accuracy.The reason is the useless attempt of model to correctly classify the last sample which is increasing the variance error of the model due to uneven distribution of + samples.

Question 3

In the following figure, we see the distribution of target variable using histogram:



BootStrapping

1-

To estimate the desired interval, we first need a hyper parameter which is number of iterations.Suppose we have N iterations.In each iteration we take samples from our dataframe with replacement(BootStrapping) and calculate the mean age of customers who didn't pay in 90 days and append this mean to a list.Afterwards, we have a list of means from which we will take period between the 5-th and 95-th percentile to get the confidence level of 90%.The estimated interval using this method is: $[45.73144553, 46.12021343]$

RandomForst

A model can have several hyperparameters which we need to find the optimal values of them for our costum dataset.There are several methods to find optimal hyperparameters.The simplest idea which comes to mind is to simply search through a specific domain for hyperparameters and find the one which fits best to our dataset.This method is called **GridSearch**.In this subsection, we are finding the best parameters for a random forest with 100 base estimators.One thing to note is the way in which we split our data into train and validation.Here we use stratified 5-fold validation in which we train the model 5 times on different $\frac{4}{5}$ portions of data and validate it using the other $\frac{1}{5}$.The set of best parametes we got in GridSearch is:
$max\_depth : 15, max\_features : 2, min\_samples_leaf : 3$ Now that we have the

optimal parameters, we can train the model and evaluate it using ROC and AUC scores.Training the model and finding these two metrics, I got the following results:
$ROCScore : 0.82676138581145 AUCScore : 0.82676138581145$
And also we can find the feature which has the least effect on the model using the **feature_importances_** attribute.Doing so we obtain that the feature **NumberOfDependents** has the weakest influence on the model.

Bagging

Searching between all states of parameters can be time consuming and sometimes impossible;thus we can use **Randomized Search** to find the optimal parameters.Doing so I got the following parameters as the best parameters:
n_estimators: 100,
max_samples: 0.9,
max_features: 4,
estimator: LogisticRegression(C=10, max$_iter = 500$)

and following scores were obtained after training:
$ROCScore : 0.7527685194565812 AUCScore : 0.7527685194565812$

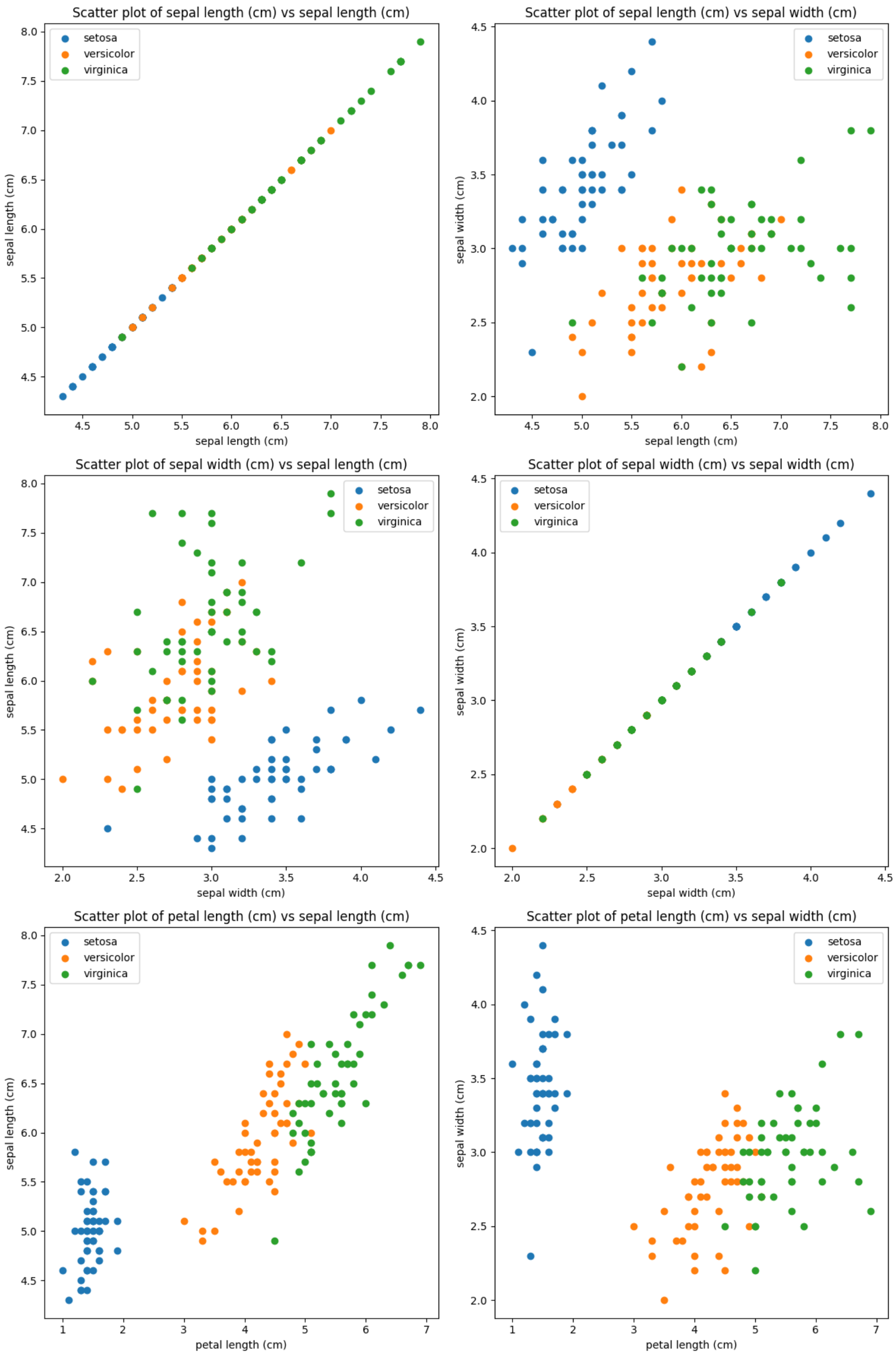
Question 4

In this question we are implementing the **SAMME** algorithm.Here is the sudocode of SAMME algorithm which is a generalization for binary adaboost classifier:

Algorithm 1 SAMME algorithm

```
Initialize the observation weights uniformly
for $m = 1 \to M$ do
    Fit classifier $T^m(x)$ to the training data using weights w$_i$
    compute error : $err^m = \frac{\sum_{i=1}^{n} w_i I(c_i \neq T^m(x_i))}{\sum_{i=1}^{n} w_i}$
    compute learner weight : $\alpha^m = \log(\frac{1-err^m}{err^m}) + \log(K-1)$ in which K is number of classes
    update weights: $w_i \leftarrow w_i.\exp(\alpha^m.I(c_i \neq T^m(x_i)))$ and renormalize w afterwards.
end for
Output : $C(x) = \underset{k}{argmax} \sum_{m=1}^{M} \alpha^m.I(T^m(x) = k)$
```

The only primary difference between this implementation and binary implementation is in the prediction phase which I made a made a mapping for each label and filled the mapped space of label with the weights of classifiers who voted for that label.Completing the provided template and training the model, I got the accuracy 1.The reason for which we got the maximum accuracy lays in the dataset.Lets visualize the scatter plot of the dataset:



From the above scatters, we can see that **setosa** can easily be seperated from others using only one feature.Exploring the other two species, we can see that they can be seperated using two features.Thus we can also get this accuracy using only three estimators in our ensemble method.I trained the model using three estimators and got the accuracy 1.But we need more train datas to perfectly seperate versicolor from virginica because we are using less estimators.So I changed the train-test ratio to 20-80 for getting one accuracy using three estimators.

Question 5

At first, lets find the root or our decision tree.To do so we have to find the information gain for each feature:

BloodPressure

YES

6+, 2-

NO

3+, 3-
Entropy in this case is 0.618

Cholesterol Level

Normal

3+, 2-

Critical

4+, 0-

High

2+, 3-
Entropy in this case is minimum

Smoking

YES

6+, 1-

NO

3+, 4-
Entropy in this case is 0.54

Weight

Overweight

4+, 2-

Normal

3+, 1-

Fat

2+, 2-
entropy in this case is 0.631

Thus the feature **Cholesterol Level** will be selected as the root of decision tree.when the cholesterol level is critical, the label will be YES.Now we have to find root of subtrees when cholesterol level is high or normal.let start with high cholesterol:

Blood Pressure

YES

1+, 2-

NO

1+, 1-
in this case the entropy is 0.23

Smoking

YES

2+, 0-

NO

0+, 3-
in this case, the entropy is minimum.

Weight

OverWeight

1+, 1-

Normal

1+, 0-

Fat

0+, 2-
in this case also the entropy is minimum but uncertainity is more than the case when we decide by Smoking because here we have subtrees that are not perfectly seperating the classes

Thus the root in this subtree will be **Smoking**.now lets find the case where cholesterol level is normal:

BloodPressure

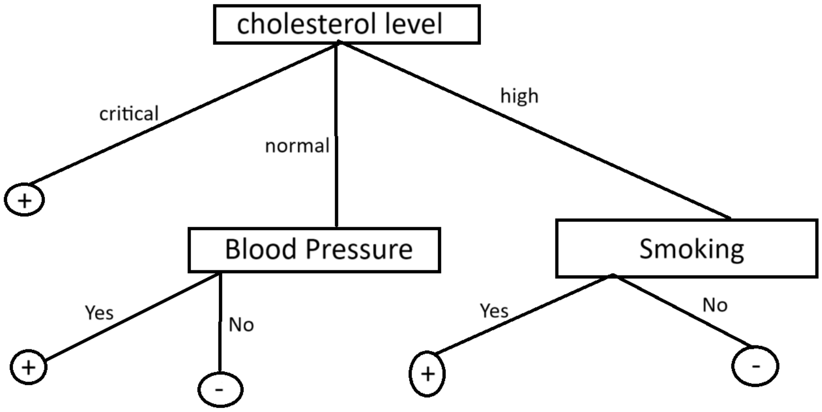
YES

3+, 0-

NO

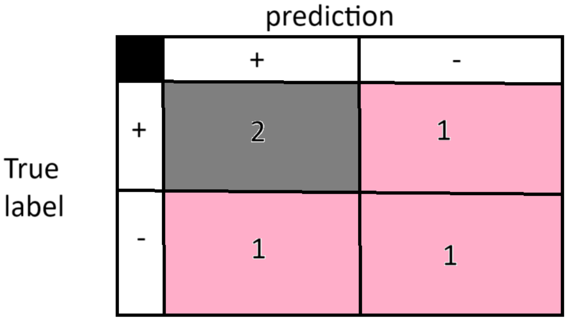
0+, 2-
we can stop here because this feature perfectly seperates classes.

finally the decision tree will look like:



Now lets predict the test data using the above treeL
15-Yes
16-Yes
17-No
18-Yes
19-No

We can see that we got 60% accuracy.Lets plot the confusion matrix:



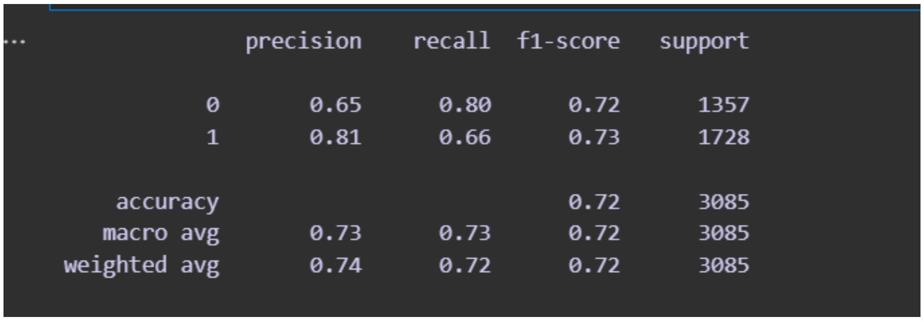
		prediction	
		+	-
True label	+	2	1
	-	1	1

Decision trees are prone to overfitting because they have the ability to create very complex trees that perfectly fit the training data, capturing noise and outliers.Lets explore two solutions for that:

- implement regularization techniques such as pruning. Pruning involves removing branches that do not significantly improve the model's performance on the validation set, leading to a simpler and more generalizable tree.
- use ensemble methods such as Random Forest or Gradient Boosting. By combining multiple decision trees and aggregating their predictions, ensemble methods can reduce overfitting and improve the model's accuracy and robustness. Random Forest, for example, creates an ensemble of decision trees by training each tree on a random subset of the training data and features, thereby decreasing the likelihood of overfitting.

Question 6

In this question, we are implementing decision tree from scratch.To do so, I find the root of the tree by traversing features and calculating information gain and when I find the root, I create the subtrees that should continue fitting the data by ommiting the feature of root and this process is done recursivly until either we reach the max depth or the datas are perfectly classified.Training the model and testing it, I got the following classification report:

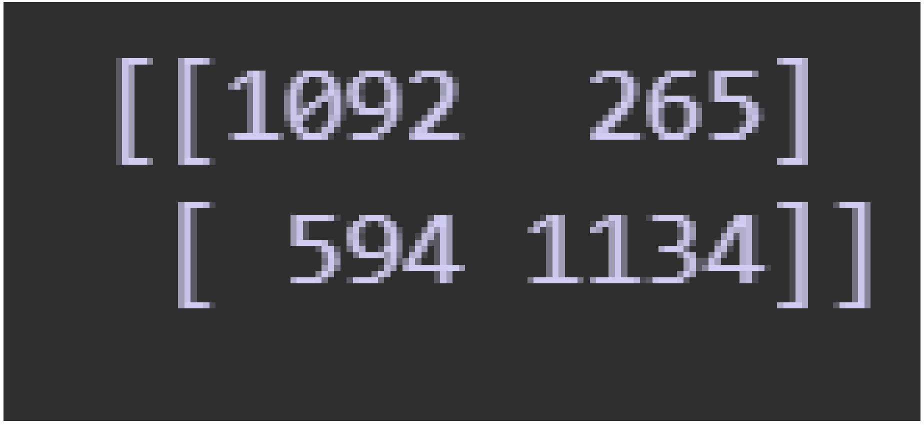


```
...              precision    recall  f1-score   support

           0       0.65      0.80      0.72      1357
           1       0.81      0.66      0.73      1728

    accuracy                           0.72      3085
   macro avg       0.73      0.73      0.72      3085
weighted avg       0.74      0.72      0.72      3085
```

and the following confusion matrix:



```
[[1092  265]
 [ 594 1134]]
```