Machine Learning
Assignment - 2

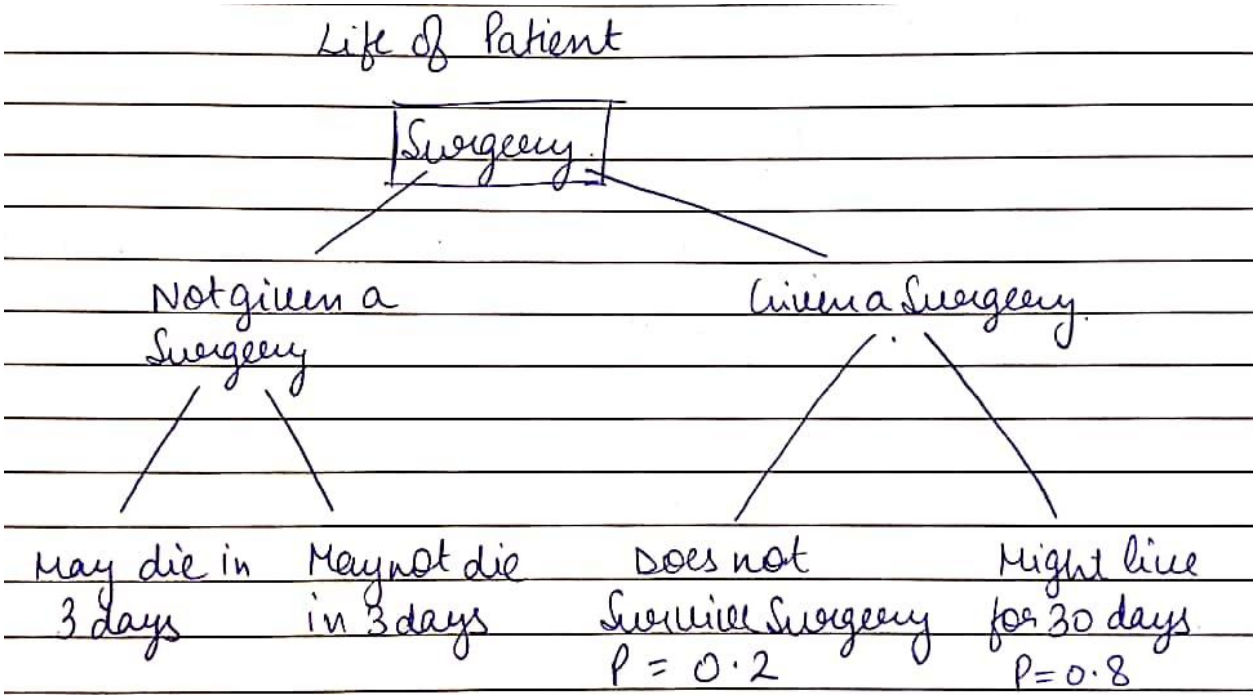
Ashwin Sheoran
2020288

Section-A

Ans 1)

a)

The decision Tree is given as follows



b)
L(x) denotes the patient's living function, where x represents the number of days he will survive. Assuming that L(30) = 1.0 and L(0) = 0
We have to estimate how low can the patient's utility for living 3 days and still have the surgery performed

If the Utility of patient is low , he will have to perform the surgery to increase his chances of living, So his utility should be close to 1. We can approximate his utility to be around 0.75 .

(PTO)

c)

Ans 3)

$T \rightarrow$ Patient Test Positive in the Test

$S \rightarrow$ Patient Survives surgery.

We are given

$P(T|S) = 0.95$

and

$P(T|\bar{S}) = 0.05$

We are also given

$P(S) = 1 - 0.2 = 0.8.$

We have to find

$P(S|T) = ?$

Using Bays theorem we know

$$P(S|T) = \frac{P(T|S)\,P(S)}{P(T)}$$

Now we know that

$$P(T) = P(T \cap S) + P(T \cap \bar{S})$$

#

Also

$$P(T|S) = \frac{P(T \cap S)}{P(S)}$$

$$\therefore P(T \cap S) = P(T|S)\,P(S)$$

Similiarly ~~P(T∩S)~~ $P(T|\bar{S}) = \frac{P(T \cap \bar{S})}{P(\bar{S})}$

$$P(T \cap \bar{S}) = P(T|\bar{S})\,P(\bar{S})$$

$\therefore$

$$P(T) = P(T|S)\,P(S) + P(T|\bar{S})\,P(\bar{S})$$

$$P(S|T) = \frac{P(T|S)\,P(S)}{P(T|S)\,P(S) + P(T|\bar{S})\,P(\bar{S})}$$

$$P(S|T) = \frac{(0.95)(0.8)}{(0.95)(0.8)+(0.05)(0.2)}$$

$$P(S|T) = 0.987$$

∴ Survivor has 0.98 probability of a successful surgery if he Tests positive.

(d)

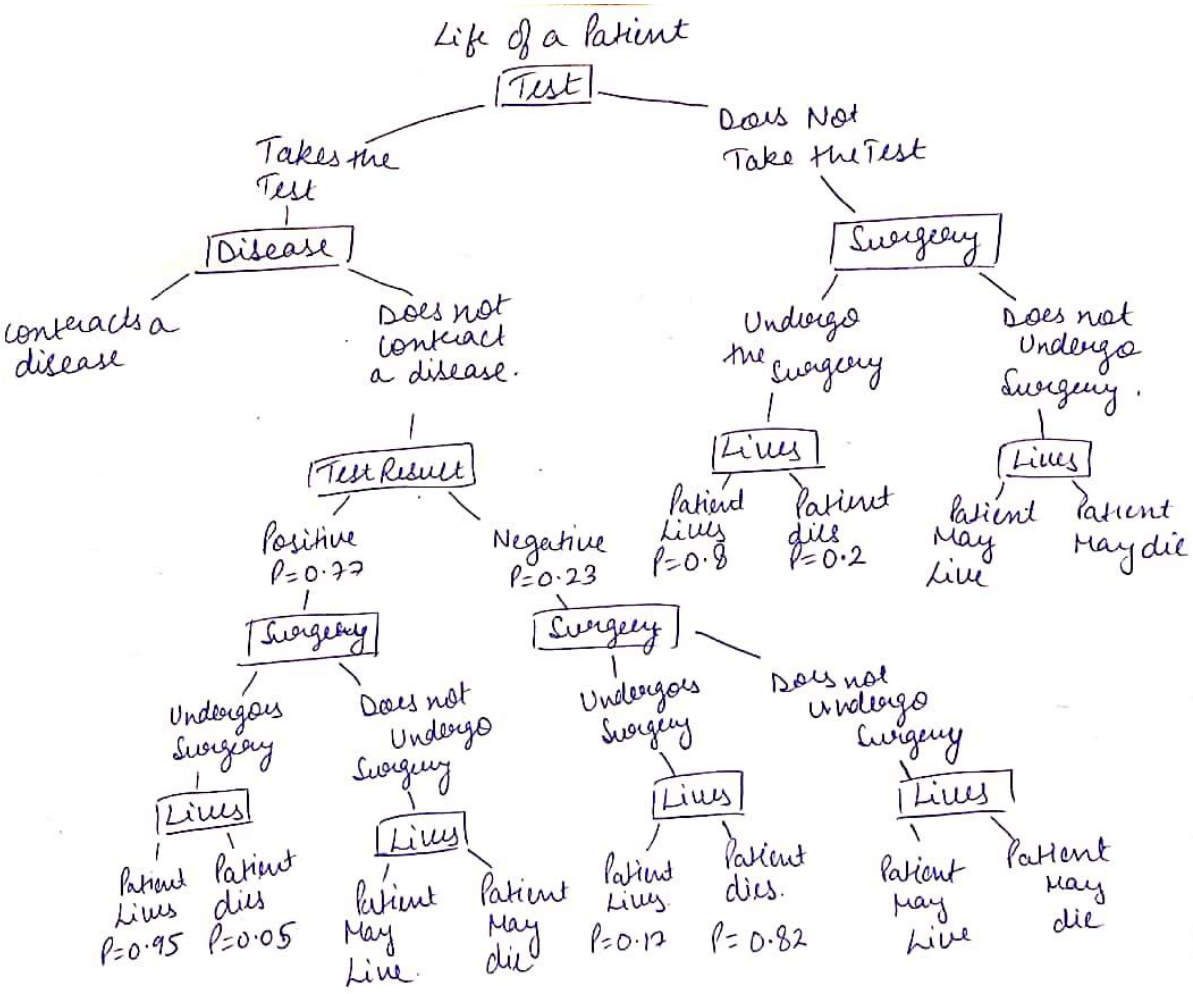
We are given that L(3) = 0.8 .

But we know that if a person tests positive and takes the test then he has high chances of surviving the surgery (probability = 0.98)

As 0.98 > 0.8 ,

<u>The surgery should be performed</u> if the person tests positive

(e)

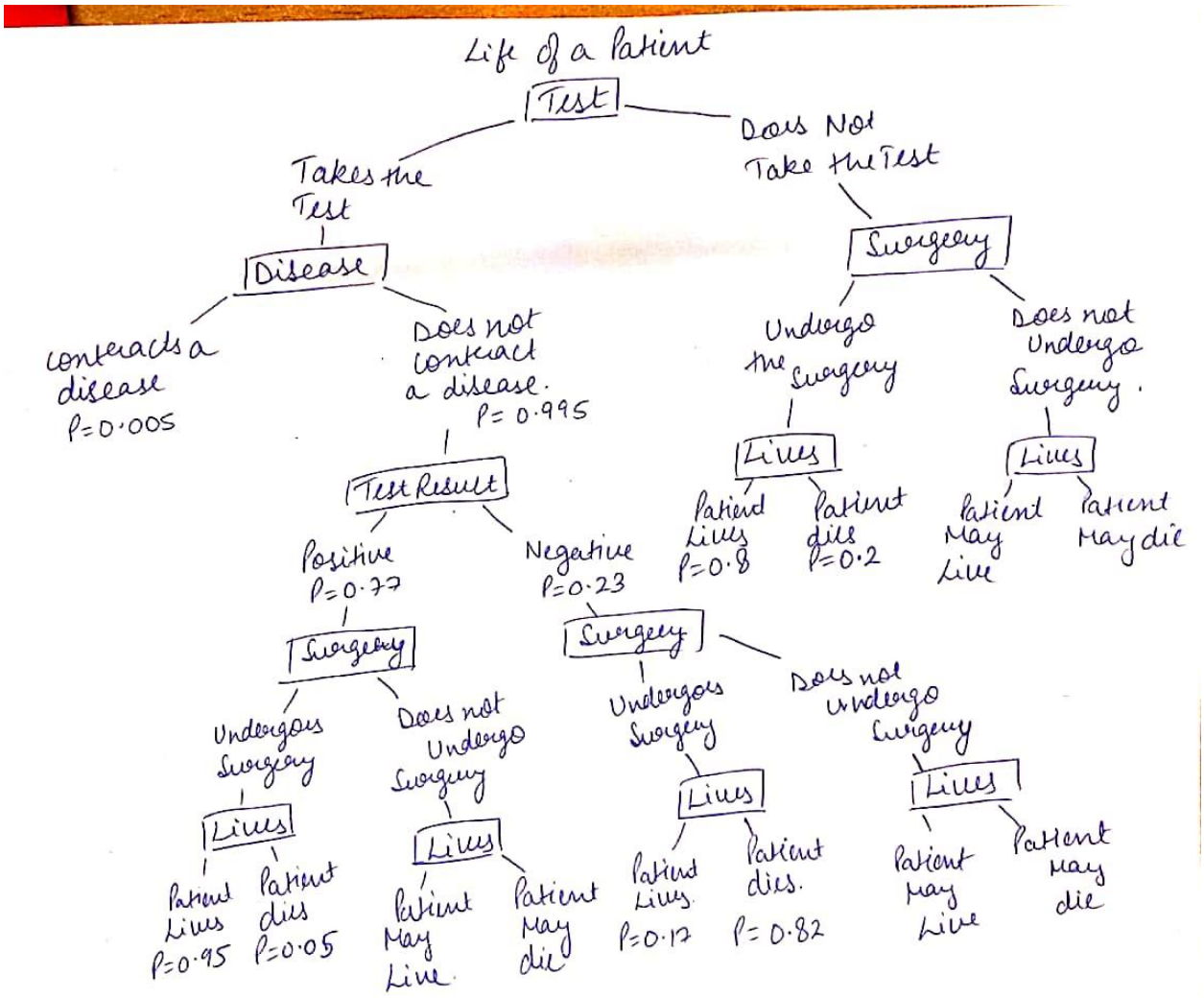


<u>Here we have made an assumption that if a patient contracts a new disease leading to his death as it is a fatal disease.</u>

(f)

P(new disease) = 0.005

<u>Yes the test should be conducted</u> as the test is very effective in prediction of successful surgery (Survivor has 0.98 probability of surviving surgery of test is positive) and the probability of getting a new disease is very less (Probablity = 0.005)



<u>Here we have made an assumption that if a patient contracts a new disease leading to his death as it is a fatal disease.</u>

Section-C

Ans A.

Trained a decision tree using both Gini Index and Entropy by changing the value of the max-depth [4, 8, 10, 15, 20].

We are getting the following results

Max depth = 4 Accuracy = 0.9859590497044612

Max depth = 8 Accuracy = 0.9862447600718619

Max depth = 10 Accuracy = 0.9874424579320055

Max depth = 15 Accuracy = 0.9883247315465388

Max depth = 20 Accuracy = 0.9862950450965244

We are getting best result with Max Depth = 15

(PTO)

Python Code

a) Train a decision tree using both the Gini index and the Entropy

Max depth 4

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(max_depth=4 , criterion='entropy') ## criteria has by default gini
clf = clf.fit(X_train , y_train)
predictions = clf.predict(X_test)
clf.score(X_test , y_test)

## Reference taken from scikit-learn.org
```

```
0.9859590497044612
```

Max depth 8

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(max_depth=8 , criterion='entropy') ## criteria has by default gini
clf = clf.fit(X_train , y_train)
predictions = clf.predict(X_test)
clf.score(X_test , y_test)

## Reference taken from scikit-learn.org
```

```
0.9862447600718619
```

Max depth 10

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(max_depth=10 , criterion='entropy') ## criteria has by default gini
clf = clf.fit(X_train , y_train)
predictions = clf.predict(X_test)
clf.score(X_test , y_test)

## Reference taken from scikit-learn.org
```

```
0.9874424579320055
```

Max depth 15

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(max_depth=15 , criterion='entropy') ## criteria has by default gini
clf = clf.fit(X_train , y_train)
predictions = clf.predict(X_test)
clf.score(X_test , y_test)

## Reference taken from scikit-learn.org
```

```
0.9883247315465388
```

Max depth 20

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(max_depth=20 , criterion='entropy') ## criteria has by default gini
clf = clf.fit(X_train , y_train)
predictions = clf.predict(X_test)
clf.score(X_test , y_test)

## Reference taken from scikit-learn.org
```

```
0.9862950450965244
```

Ans B.

```
Accuracy obtained is 0.9857715586600351
```

The accuracy coming from Ensembling by creating 100 different decision stumps on 50% data and predicting the test samples' labels by taking a majority vote of the output of the stumps is 0.9857715586600351

Python Code

b) Ensembling is a method to combine multiple not-so-good models to get a better performing model.

```
import numpy as np

## We have to create 100 decision stumps
stumps=[]
for i in range(100):
    stumps.append(DecisionTreeClassifier(criterion="entropy",max_depth=3))

## Now we have to train it on randomly selected 50% of data


for i in range(100):
    X_train_frac=X_train.sample(frac=0.5)
    y_train_frac=y_train.loc[X_train_frac.index]
    stumps[i].fit(X_train,y_train)
## Now we have Trained
## Now we have to check the prediction
```

```
import numpy as np

predicted_samples=[]
## The list of predicted samples
for i in range(100):
    predicts=stumps[i].predict(X_valid)
    predicted_samples.append(predicts)

new_predicted_samples=[]   ## List of new predicted Sample ( after majority voting of output of stumps)

##  Majority voting of output of stumps
for i in range(len(predicted_samples[0])):
    max_val={}   # creating a dictionary to store the key value pair of y
    for j in predicted_samples:
        if(j[i] in max_val):
            max_val[j[i]]+=1
        else:
            max_val[j[i]]=1
    max_key=max(max_val, key= lambda x: max_val[x])
    new_predicted_samples.append(max_key)   ## appeding the y with maximum value of dictionary

accuracy=np.sum(np.array(new_predicted_samples)==np.array(y_valid.to_list()))/len(y_valid)   ## Calculating Accuracy

print("Accuracy obtained is "+str(accuracy))
```

```
Accuracy obtained is 0.9857715586600351
```

Ans C.

Using AdaBoost

Using Max Depth = 15

n estimators = 4 Accuracy = 0.9723615218991283

n estimators = 8 Accuracy = 0.9765351789461173

n estimators = 10 Accuracy = 0.9801031300142169

n estimators = 15 Accuracy = 0.8836747381750193

n estimators = 20 Accuracy = 0.9844184994034367

We are getting the best result with n estimators = 20 and when max depth = 15

Comparison between Random Forest and Adaboost

The best Accuracy that we got with Random Forest was 0.9883247315465388

With Adaboost we are getting Accuracy 0.9844184994034367

Python Code for AdaBoost

C) Use the Adaboost algorithm on the above dataset and report the testing accuracy.

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=15)  , n_estimators=4 )
clf.fit(X_train, y_train)
clf.score(X_test, y_test)

## Reference taken from scikit-learn.org
```

```
0.9668713114791568
```

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=15) , n_estimators=8 )
clf.fit(X_train, y_train)
clf.score(X_test, y_test)

## Reference taken from scikit-learn.org
```

```
0.9737169318820771
```

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=15) ,    n_estimators=10)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
## Reference taken from scikit-learn.org
```

```
0.9765900353366582
```

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=15) , n_estimators=15)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)

## Reference taken from scikit-learn.org
```

```
0.9801739861853324
```

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
clf = AdaBoostClassifier( DecisionTreeClassifier(max_depth=15)  , n_estimators=20)
clf.fit(X_train, y_train)
clf.score(X_test, y_test)

## Reference taken from scikit-learn.org
```

```
0.984343071866443
```