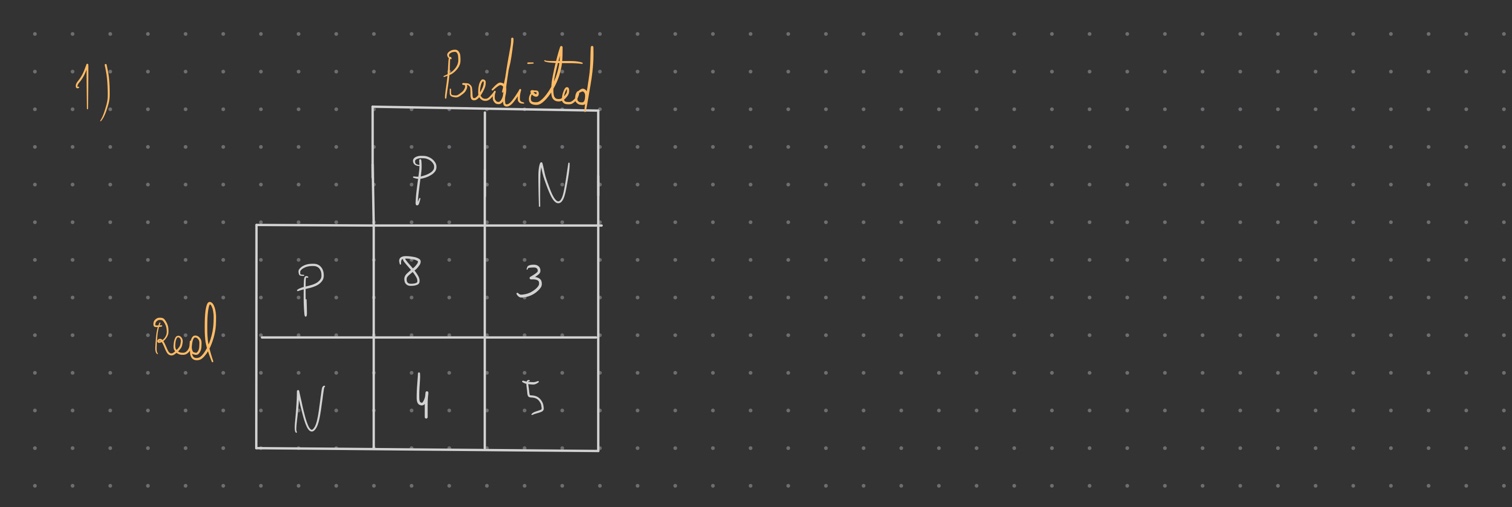
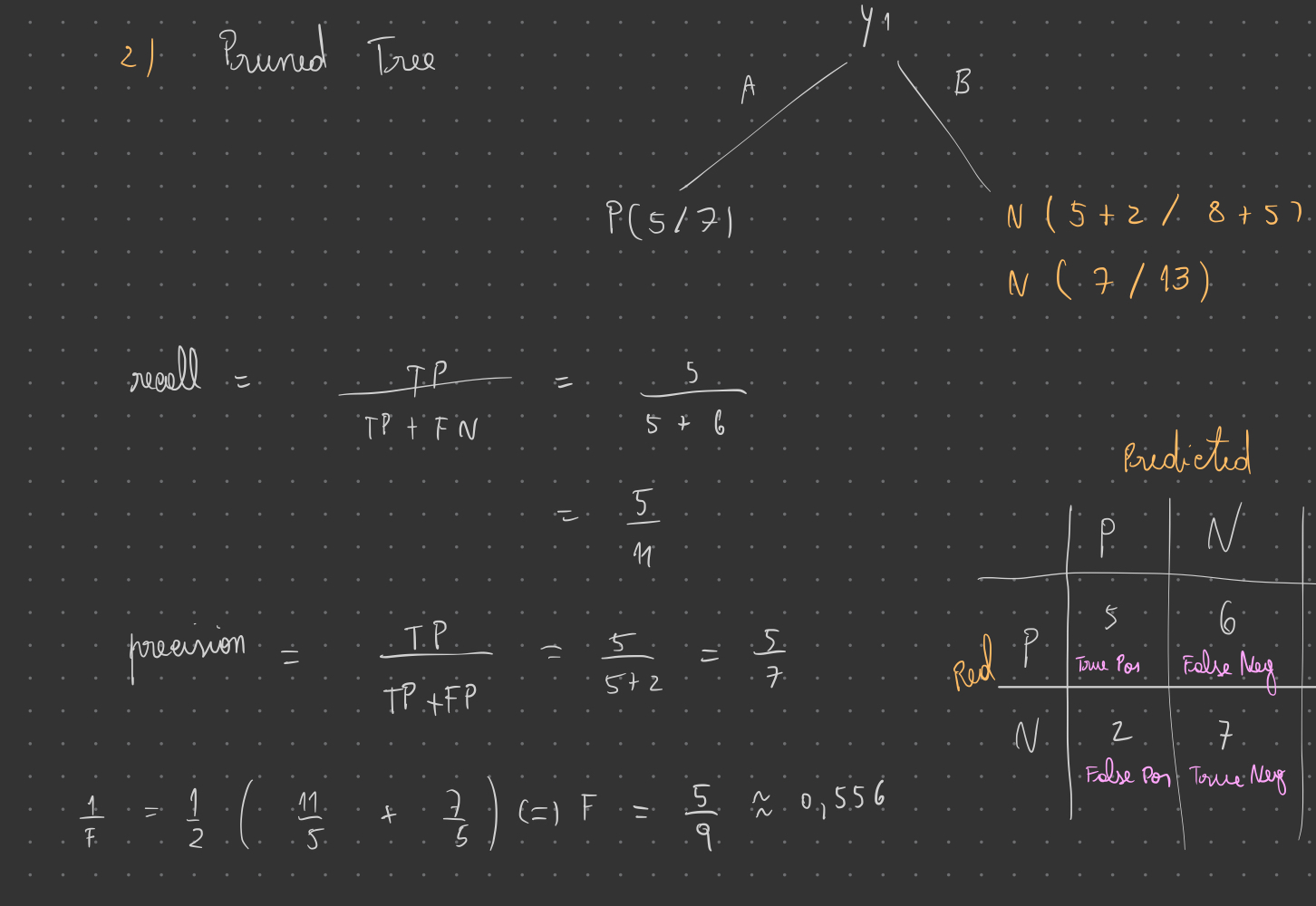
**I. Pen-and-paper**

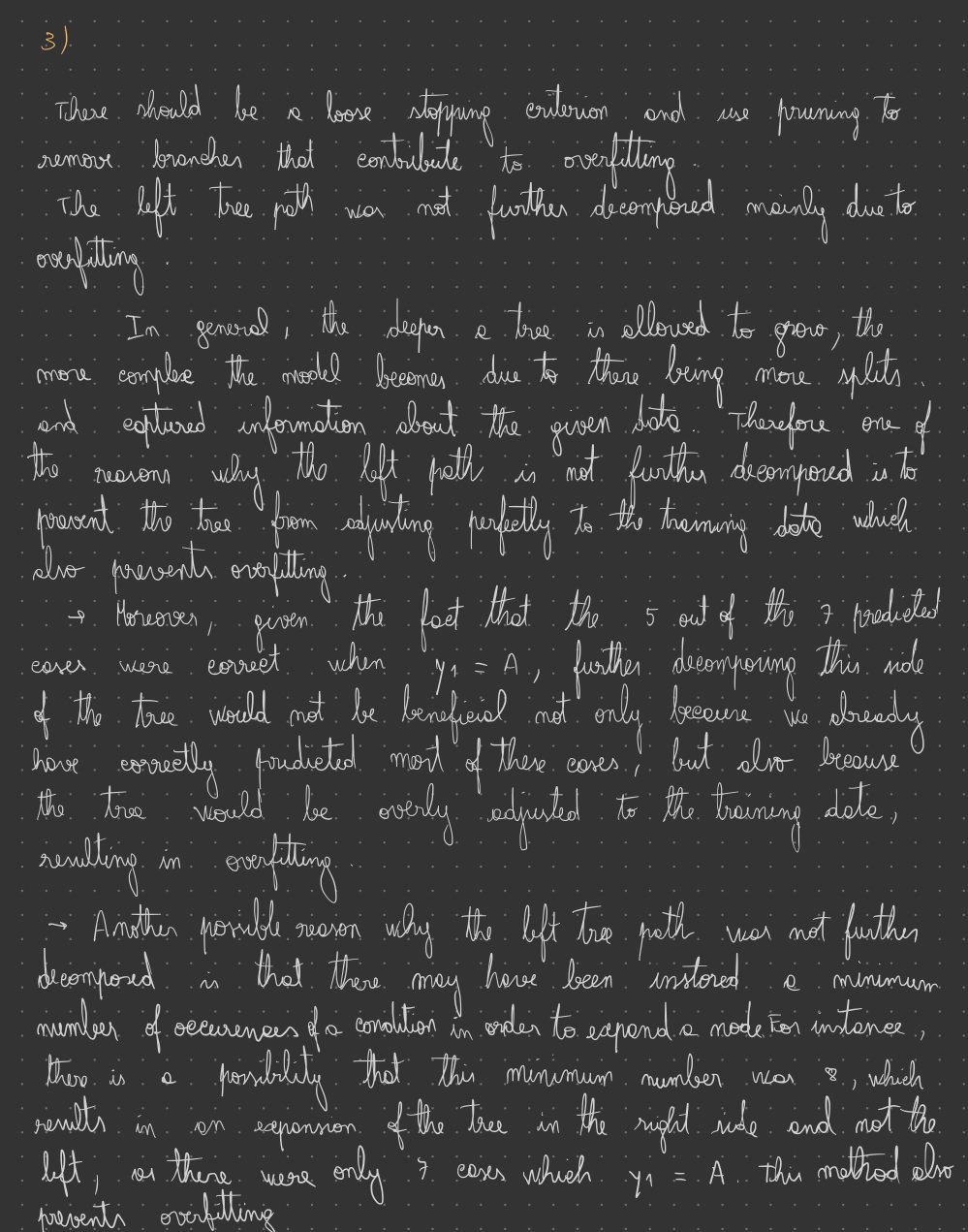
**1)**



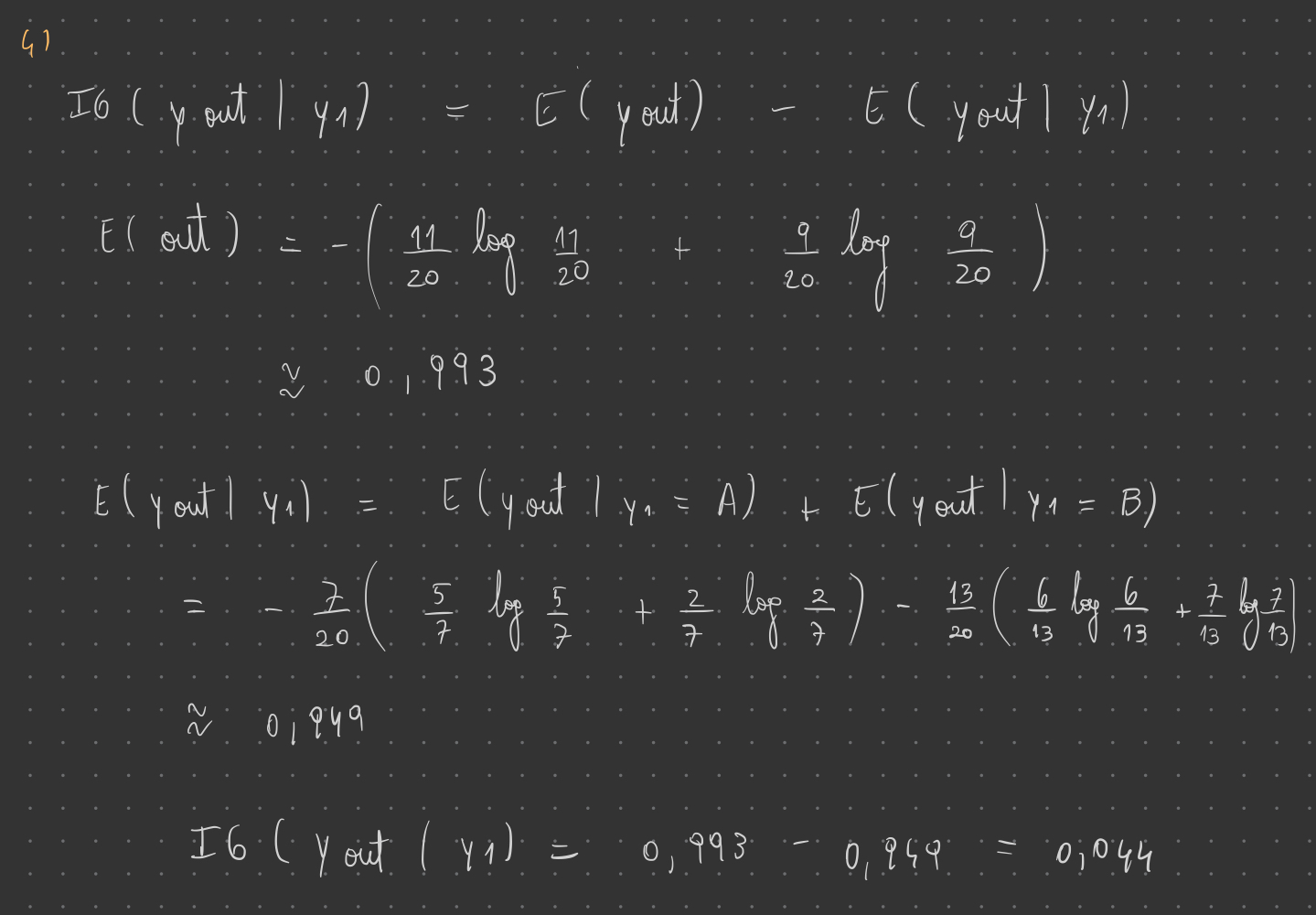
**2)**



**3)**

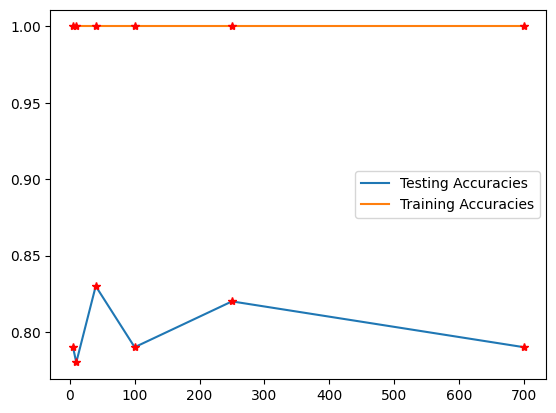


**4)**

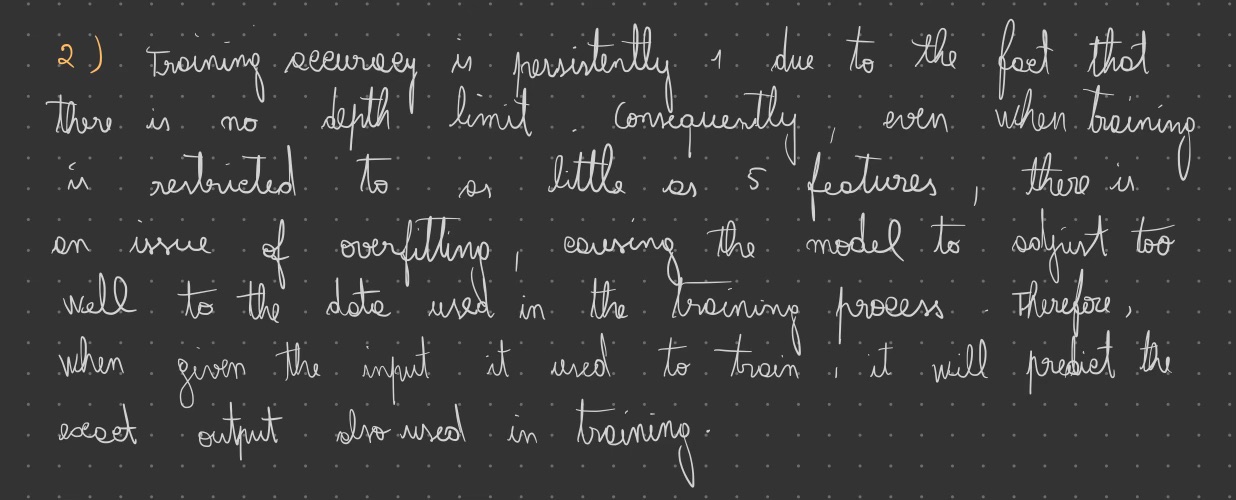


**II. Programming and critical analysis**

**1)**



**2)**



**III. APPENDIX**

**import** pandas **as** pd

**from** scipy.io.arff **import** loadarff

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.feature\_selection **import** mutual\_info\_classif, SelectKBest

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn **import** metrics, datasets, tree

data **=** loadarff('pd\_speech.arff')

df **=** pd.DataFrame(data[**0**])

df['class'] **=** df['class'].str.decode('utf-8')

testingAccuracies **=** []

trainingAccuracies **=** []

featureNums **=** (**5**, **10**, **40**, **100**, **250**, **700**)

target **=** df["class"]

variables **=** []

**for** i **in** df.columns:

variables.append(i)

X **=** df.copy()

X **=** X.drop('class', **axis** **=** **1**)

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, target, **test\_size** **=** **0.30**, **stratify** **=** target, **random\_state** **=** **1**)

**for** i **in** featureNums:

selector\_method **=** SelectKBest(mutual\_info\_classif, **k** **=** i)

selector\_method.fit(X\_train, y\_train)

X\_train\_i **=** selector\_method.transform(X\_train)

X\_test\_i **=** selector\_method.transform(X\_test)

predictor **=** tree.DecisionTreeClassifier()

predictor.fit(X\_train\_i, y\_train)

y\_predTesting **=** predictor.predict(X\_test\_i)

y\_predTraining **=** predictor.predict(X\_train\_i)

testingAccuracies.append(**round**(metrics.accuracy\_score(y\_test, y\_predTesting), **2**))

trainingAccuracies.append(**round**(metrics.accuracy\_score(y\_train, y\_predTraining), **2**))

**print**("accuracy on testing set:", **round**(metrics.accuracy\_score(y\_test, y\_predTesting), **2**))

plt.plot(featureNums, testingAccuracies, **label** **=** "Testing Accuracies")

plt.plot(featureNums, trainingAccuracies, **label** **=** "Training Accuracies")

**for** i **in** **range**(**0**, **len**(featureNums)):

plt.plot(featureNums[i],testingAccuracies[i], 'r\*')

plt.plot(featureNums[i],trainingAccuracies[i], 'r\*')

plt.legend()

plt.show()

**END**