TÉCNICO LISBOA

Aprendizagem 2021/22

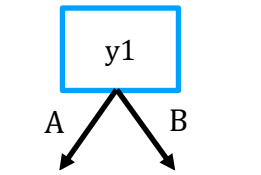
Homework I – Group 086

I. Pen-and-paper

1) TP = 5+3 = 8; FP = 7-5+5-3 = 4; TN = 5; FN = 8-5 = 3

Predicted/Actual	P_{actual}	N_{actual}
$P_{predicted}$	8_{TP}	4_{FP}
$N_{predicted}$	3_{FN}	5_{TN}

2) Pruning:



#(y1=B) = 8+5 = 13
#P = 3+(8-5) = 6
#N = 13-6 = 7

* Classified as N because when considering only y1=B, #N=7>6 =#P

$$Precision = \frac{TP_{prun}}{TP_{prun} + FP_{prun}} = \frac{5}{5 + (7-5)} = \frac{5}{7}$$

$$Recall = \frac{TP_{prun}}{TP_{prun} + FN_{prun}} = \frac{5}{5 + (13-7)} = \frac{5}{11}$$

$$Training\ F1 = \frac{2 \times P \times R}{P + R} = \frac{5}{9}$$

3) We found two possibilities:

- So as to avoid overfitting because the larger the tree the greater the chances of the tree capturing the noise of the training data and failing to generalize to the test data. So some chosen criterion might have stopped the further expansion of the tree to avoid overfitting.
- The considered features lacked disciminative power. So, for example, if information gain was chosen as the selection criteria to build the tree, then for all the considered features their information gain was either 0 or failed to pass some considered treshold.

4) $\#P_{Actual} = TP + FN = 8 + 3 = 11 \quad \#N_{actual} = TN + FP = 5 + 4 = 9 \quad \#TOTAL = TP + FP + TN + FN = 20$

$$I(table) = -\frac{11}{20}\log_2\frac{11}{20} - \frac{9}{20}\log_2\frac{9}{20} \cong 0.9928\ bits$$

$$I(C_A) = -\frac{5}{7}\log_2\frac{5}{7} - \frac{7-5}{7}\log_2\frac{7-5}{7} \cong 0.8631\ bits$$

$$I(C_B) = -\frac{7}{13}\log_2\frac{7}{13} - \frac{13-7}{13}\log_2\frac{13-7}{13} \cong 0.9957\ bits$$

$$I(y_1) = \frac{\#(y1=A)}{\#TOTAL} \times I(C_A) + \frac{\#(y1=B)}{\#TOTAL} \times I(C_B) = \frac{20-13}{20} \times I(C_A) + \frac{13}{20} \times I(C_B) = 0.9493\ bits$$

$$Gain(y1) = I(table) - E(y1) = 0.0435\ bits$$

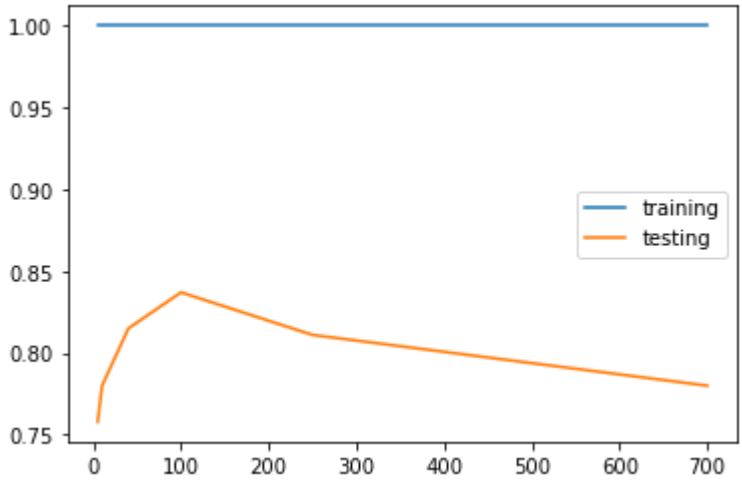
TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 086

II. Programming and critical analysis

1) We plotted the training and testing accuracies of our decision tree.



2) Why the training accuracy is consistently 1:

- We chose the K best features for the training data. This means that even with one feature, chosen from a pool of 753, the training accuracy is already a whopping 0.998. Also, since no limit was imposed on the depth of the tree and no criteria was used to trim it, the model overfits the training data. Thus, for the considered number of features the training accuracy is invariably 1.

Critical analysis of the gathered results:

- For a small number of features the bias is larger. As the number of features increased along with the model complexity, the bias went down and the variance and testing accuracy went up. Eventually, when going from 100 to 250 features, the variance got too big and we started overfitting the training data and getting worse accuracy on the test data, the model got too complex and failed to generalize so well. From 200 to 700 features it somewhat stabilized. We still see a decline in testing accuracy, though. About 100 features seem to offer the best tradeoff between bias and variance and for the considered number of features it gave the best testing accuracy (about 0.85).

TÉCNICO LISBOA

Aprendizagem 2021/22

Homework I – Group 086

III. APPENDIX

```
from scipy.io.arff import loadarff
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn import tree, metrics
from sklearn.feature_selection import SelectKBest, mutual_info_classif

data = loadarff('pd_speech.arff')
df = pd.DataFrame(data[0])
df['class'] = df['class'].str.decode('utf-8')
#df.head()

#print(df.value_counts('class'))
#print(df.isna().sum())

number_of_features = [5, 10, 40, 100, 250, 700]
x, y = df.drop('class', axis=1), np.ravel(df['class'])
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 0.7,
random_state=1)
data_accuracy_results = {'training':[], 'testing':[]}
for i in number_of_features:
    select = SelectKBest(score_func = mutual_info_classif, k=i)
    select.fit(x_train, y_train)
    x_train_new = select.transform(x_train)
    x_test_new = select.transform(x_test)
    predictor = tree.DecisionTreeClassifier()
    predictor.fit(x_train_new, y_train)
    y_pred_train = predictor.predict(x_train_new)
    y_pred_test = predictor.predict(x_test_new)
    data_accuracy_results['training'].append(metrics.accuracy_score(y_train,
y_pred_train))
    data_accuracy_results['testing'].append(metrics.accuracy_score(y_test,
y_pred_test))

df_accuracy_results = pd.DataFrame(data_accuracy_results, index = number_of_features)
df_accuracy_results.plot.line()
```

END

https://scikit-learn.org/stable/modules/tree.html