Khan Inan

HW 5

Course # 7743

Machine Learning HW 5:

Boosting Decision Trees & Random Forest

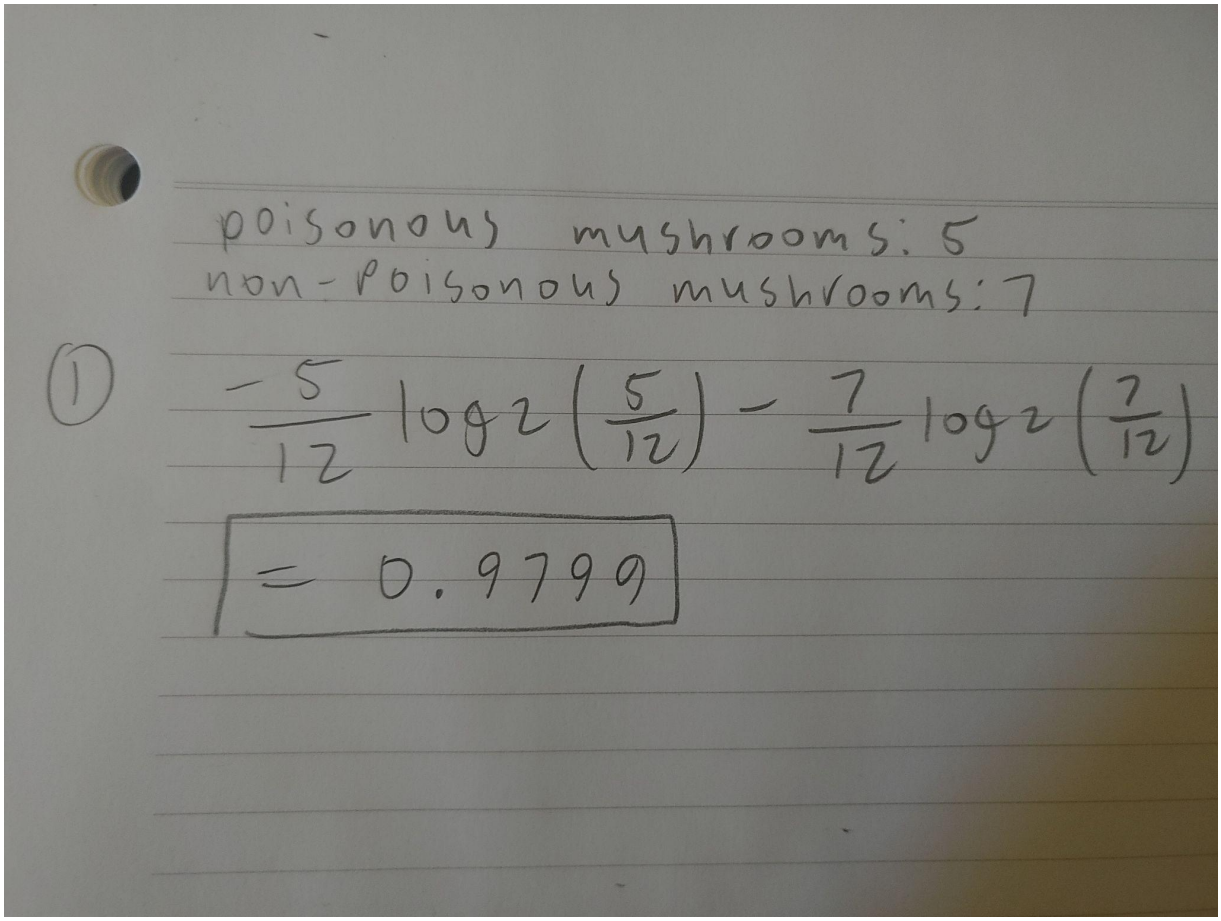
Determining Splits

Below is a sample dataset of mushrooms and it's attributes the intention is to classify them as poisonous or not.

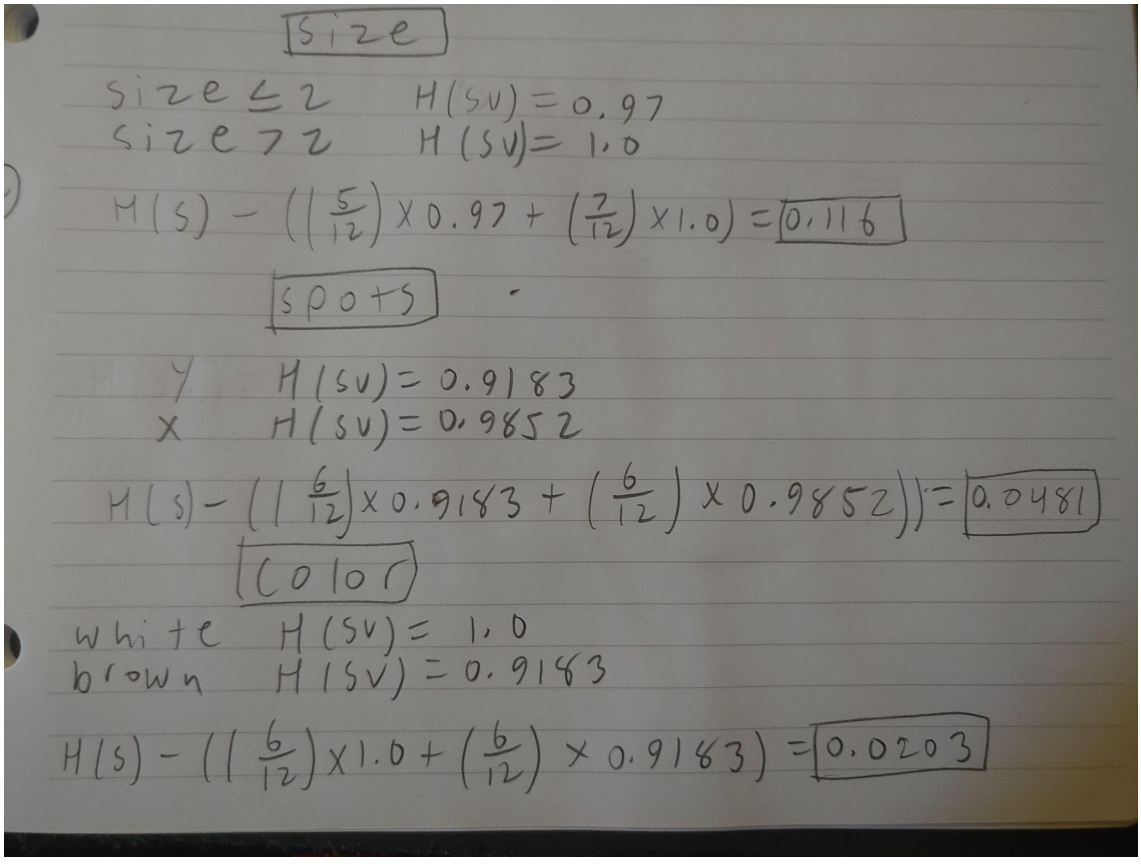
Build a decision tree by hand show all your calculations in either pen or notepad to classify

$y =$ Poisonous?	$x_1 =$ size (real-valued)	$x_2 =$ spots?	$x_3 =$ color
N	1	N	White
N	5	N	White
N	2	Y	White
N	2	N	Brown
N	3	Y	Brown
N	4	N	White
N	1	N	Brown
Y	5	Y	White
Y	4	Y	Brown
Y	4	Y	Brown
Y	1	Y	White
Y	1	Y	Brown

1. What is the entropy of the target variable, “poisonous”?



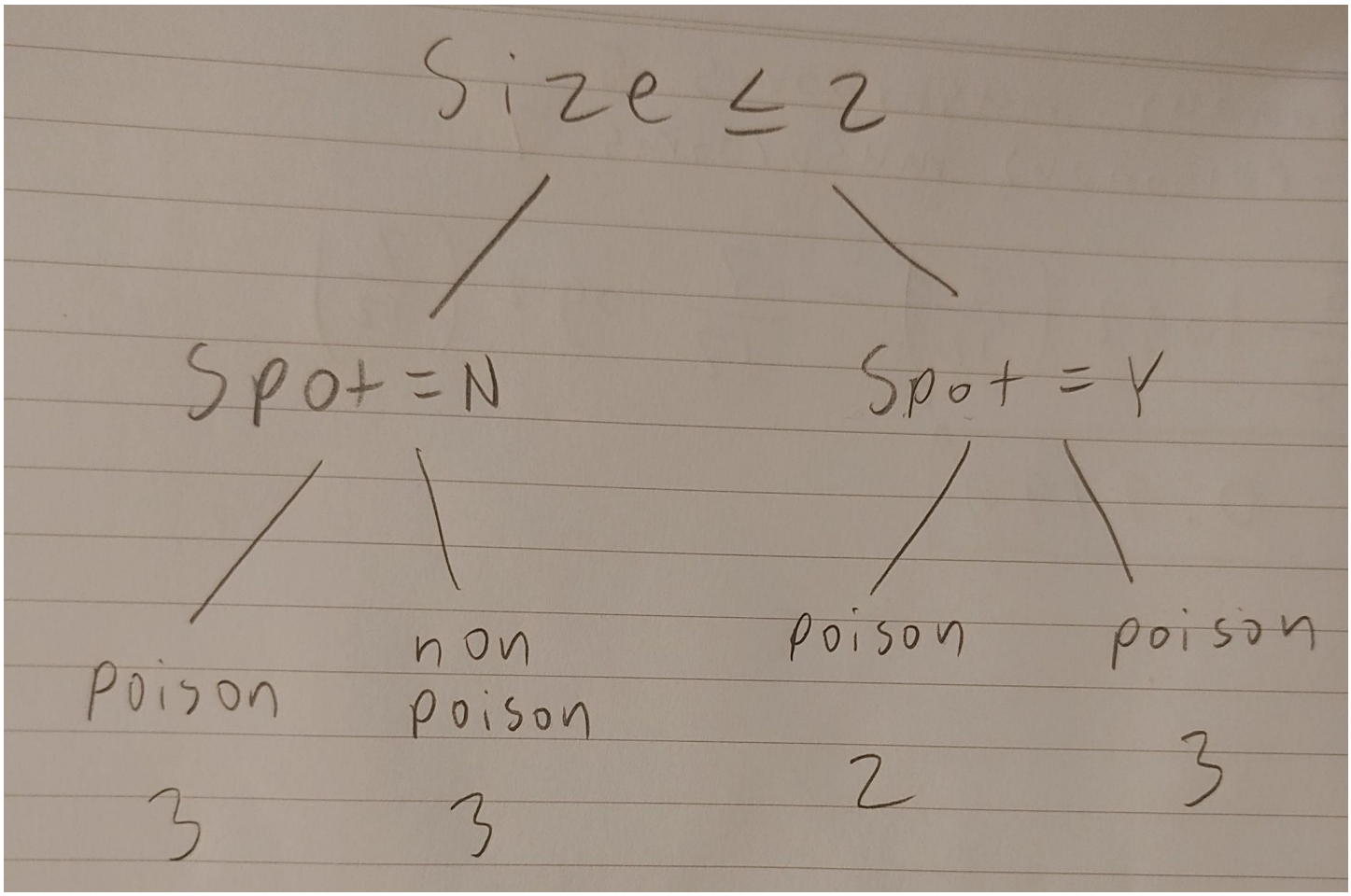
2. What is the first attribute a decision tree trained using the entropy and the information gain method



3. What is the information gain of this attribute?

Based on my calculations the size attribute has an information gain value of 0.116

4. Draw the full decision tree learned from this data set .



Part 2: Decision Tree using Python

Download the Breast cancer dataset from
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

1) Divide the data into train (80%) and test (20%)

```
[4] import pandas as pd
    from sklearn.tree import DecisionTreeClassifier, plot_tree
    from sklearn.model_selection import train_test_split, GridSearchCV
    from sklearn.metrics import accuracy_score
    from sklearn.ensemble import AdaBoostClassifier
    from sklearn import datasets

    breast_cancer = datasets.load_breast_cancer()
    data = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)
    target = pd.Series(breast_cancer.target)
```

```
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.2, random_state=42)
```

2) Using the Scikit-Learn Library train the Decision Tree Classifier using all the features of the data and test your model on the test data

```
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)

y_pred = dtc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy of Decision Tree Classifier: {accuracy}")
```

```
Accuracy of Decision Tree Classifier: 0.9298245614035088
```

3) Use the Grid Search method to run the model for trees of depth 1, 2, 3, 4, 5, and 6 and for the Gini Impurity and Entropy impurity measures.

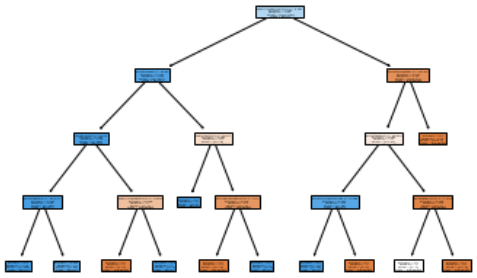
```
parameters = {'max_depth': range(1, 7), 'criterion': ['gini', 'entropy']}

grid_search = GridSearchCV(DecisionTreeClassifier(), parameters, cv=5)
grid_search.fit(X_train, y_train)

best_dtc = grid_search.best_estimator_
print(f"Best Model: {best_dtc}")

y_pred = best_dtc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy of Best Model: {accuracy}")
```

```
Best Model: DecisionTreeClassifier(criterion='entropy', max_depth=4)
Accuracy of Best Model: 0.956140350877193
```

4) Determine the best model use the plot.tree() method to visualize it.

```
plot_tree(best_dtc, feature_names=breast_cancer.feature_names, class_names=breast_cancer.target_names, filled=True)
```

```
[Text(0.5875, 0.9, 'mean concave points <= 0.051\nentropy = 0.952\nsamples = 455\nvalue = [169, 286]\nclass = benign'),
 Text(0.325, 0.7, 'worst radius <= 16.83\nentropy = 0.314\nsamples = 282\nvalue = [16, 266]\nclass = benign'),
 Text(0.2, 0.5, 'radius error <= 0.626\nentropy = 0.136\nsamples = 263\nvalue = [5, 258]\nclass = benign'),
 Text(0.1, 0.3, 'worst texture <= 30.145\nentropy = 0.091\nsamples = 260\nvalue = [3, 257]\nclass = benign'),
 Text(0.05, 0.1, 'entropy = 0.0\nsamples = 225\nvalue = [0, 225]\nclass = benign'),
 Text(0.15, 0.1, 'entropy = 0.422\nsamples = 35\nvalue = [3, 32]\nclass = benign'),
 Text(0.3, 0.3, 'mean compactness <= 0.063\nentropy = 0.918\nsamples = 3\nvalue = [2, 1]\nclass = malignant'),
 Text(0.25, 0.1, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0]\nclass = malignant'),
 Text(0.35, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]\nclass = benign'),
 Text(0.45, 0.5, 'mean texture <= 16.19\nentropy = 0.982\nsamples = 19\nvalue = [11, 8]\nclass = malignant'),
 Text(0.4, 0.3, 'entropy = 0.0\nsamples = 6\nvalue = [0, 6]\nclass = benign'),
 Text(0.5, 0.3, 'concave points error <= 0.01\nentropy = 0.619\nsamples = 13\nvalue = [11, 2]\nclass = malignant'),
 Text(0.45, 0.1, 'entropy = 0.0\nsamples = 11\nvalue = [11, 0]\nclass = malignant'),
 Text(0.55, 0.1, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]\nclass = benign'),
 Text(0.85, 0.7, 'worst perimeter <= 114.45\nentropy = 0.517\nsamples = 173\nvalue = [153, 20]\nclass = malignant'),
 Text(0.8, 0.5, 'worst texture <= 25.655\nentropy = 0.994\nsamples = 44\nvalue = [24, 20]\nclass = malignant'),
 Text(0.7, 0.3, 'worst concave points <= 0.166\nentropy = 0.454\nsamples = 21\nvalue = [2, 19]\nclass = benign'),
 Text(0.65, 0.1, 'entropy = 0.0\nsamples = 19\nvalue = [0, 19]\nclass = benign'),
 Text(0.75, 0.1, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0]\nclass = malignant'),
 Text(0.9, 0.3, 'perimeter error <= 1.558\nentropy = 0.258\nsamples = 23\nvalue = [22, 1]\nclass = malignant'),
 Text(0.85, 0.1, 'entropy = 1.0\nsamples = 2\nvalue = [1, 1]\nclass = malignant'),
 Text(0.95, 0.1, 'entropy = 0.0\nsamples = 21\nvalue = [21, 0]\nclass = malignant'),
 Text(0.9, 0.5, 'entropy = 0.0\nsamples = 129\nvalue = [129, 0]\nclass = malignant')]
```



5) Use Adaboost to improve the model and evaluate the performance using the test set.
6) What is the accuracy?

```
[8]  adaboost = AdaBoostClassifier(base_estimator=best_dtc, n_estimators=100)
     adaboost.fit(X_train, y_train)

     y_pred = adaboost.predict(X_test)
     accuracy = accuracy_score(y_test, y_pred)

     print(f"Accuracy of Adaboost Classifier: {accuracy}")
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/ensemble/_base.py:166: Futu
  warnings.warn(
Accuracy of Adaboost Classifier: 0.9736842105263158
```

Part 3: Random Forest using Python

Using the same dataset as above

1) Use the same training and test set as above.
2) Use the RandomForest classifier (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) to create a model.

```
[10] from sklearn.ensemble import RandomForestClassifier

     # create a random forest classifier with 100 trees
     rfc = RandomForestClassifier(n_estimators=100)

     # fit the model to the training data
     rfc.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

3) Compare the parameters that are provided for the Random Forest classifier and Decision Tree classifier. How many are the same and how many are different?

The max_depth, criterion, and min_samples_split are the same for both. The main difference between the two is that the random forest one has extra parameters like the # of trees and the max # of features

4) Use the Grid Search method to run the model for trees of depth 1, 2, 3, 4, 5, and 6 and for the Gini Impurity and Entropy impurity measures. Also set the parameter so it will use the "out-of-bag" samples for calculating accuracy.

```
from sklearn.model_selection import GridSearchCV

# define the parameter grid to search
param_grid = {
    'max_depth': [1, 2, 3, 4, 5, 6],
    'criterion': ['gini', 'entropy'],
    'oob_score': [True],
}

# create a random forest classifier
rfc = RandomForestClassifier(n_estimators=100)

# create the GridSearchCV object
grid_search = GridSearchCV(rfc, param_grid=param_grid)

# fit the GridSearchCV object to the training data
grid_search.fit(X_train, y_train)

# print the best hyperparameters found
print(grid_search.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 5, 'oob_score': True}
```

5) Test the accuracy of RandomForest using the Test set.

```
# predict the labels of the test set
y_pred = grid_search.predict(X_test)

# calculate the accuracy of the random forest classifier
accuracy = sum(y_pred == y_test) / len(y_test)

# print the accuracy
print("Random Forest accuracy:", accuracy)
```

```
Random Forest accuracy: 0.9649122807017544
```

6) Compare the performance of Decision Tree with Boost and Random Forest.

```
# evaluate the accuracy of the decision tree with boost
y_pred_boost = adaboost.predict(X_test)
accuracy_boost = sum(y_pred_boost == y_test) / len(y_test)
print("Decision Tree with Boost accuracy:", accuracy_boost)

# evaluate the accuracy of the random forest classifier
y_pred_rfc = grid_search.predict(X_test)
accuracy_rfc = sum(y_pred_rfc == y_test) / len(y_test)
print("Random Forest accuracy:", accuracy_rfc)

# evaluate the accuracy of the decision tree classifier
y_pred_dt = best_dtc.predict(X_test)
accuracy_dt = sum(y_pred_dt == y_test) / len(y_test)
print("Decision Tree accuracy:", accuracy_dt)
```

```
Decision Tree with Boost accuracy: 0.9736842105263158
Random Forest accuracy: 0.9649122807017544
Decision Tree accuracy: 0.956140350877193
```

As we can see, the Decision tree with boost seems to have the best performance/accuracy