Exercise 4 - Decision Tree

First name: Brian

Last name: Schweigler

Matriculation number: 16-102-071

(1) Take the titanic dataset and use all attributes to predict the class 'Survived'with a Decision tree classifier. (convert age and fare into classes ; exclude names from the attribute list)

(a) Find the best tree depth for the model

First some imports and preprocessing

In [1]:

```
%load_ext autoreload
%autoreload 2
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import preprocessing
from mlxtend.evaluate import accuracy_score

df = pd.read_csv("data/titanic.csv", index_col='Name')
pd.set_option('display.max_colwidth', None)
le = preprocessing.LabelEncoder()
bins = [0, 4, 18, 65, 100]
labels = ['Infant', 'Child', 'Adult', 'Elderly']
labels = [1, 2, 3, 4]
df['AgeGroup'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)
df["AgeGroup"] = le.fit_transform(df["AgeGroup"])
df['FareGroup'] = pd.qcut(x=df['Fare'], q=4)
df["FareGroup"] = le.fit_transform(df["FareGroup"])
df["Survived"] = le.fit_transform(df["Survived"])
df["Sex"] = le.fit_transform(df["Sex"])
print(df.head(3))
```

```
                                                     Survived  Pclass  Sex  \
Name
Mr. Owen Harris Braund                                      0       3    1
Mrs. John Bradley (Florence Briggs Thayer) Cumings          1       1    0
Miss. Laina Heikkinen                                       1       3    0

                                                      Age  \
Name
Mr. Owen Harris Braund                                22.0
Mrs. John Bradley (Florence Briggs Thayer) Cumings    38.0
Miss. Laina Heikkinen                                 26.0

                                                     Siblings/Spouses Aboard  \
Name
Mr. Owen Harris Braund                                                    1
Mrs. John Bradley (Florence Briggs Thayer) Cumings                        1
Miss. Laina Heikkinen                                                     0

                                                     Parents/Children Aboard  \
Name
Mr. Owen Harris Braund                                                    0
Mrs. John Bradley (Florence Briggs Thayer) Cumings                        0
Miss. Laina Heikkinen                                                     0

                                                        Fare  AgeGroup  \
```

```
Name
Mr. Owen Harris Braund                                 7.2500        2
Mrs. John Bradley (Florence Briggs Thayer) Cumings   71.2833        2
Miss. Laina Heikkinen                                  7.9250        2

                                                     FareGroup
Name
Mr. Owen Harris Braund                                       0
Mrs. John Bradley (Florence Briggs Thayer) Cumings           3
Miss. Laina Heikkinen                                        0
```

In [2]:

```
all_features = ['Pclass', 'Sex', 'Siblings/Spouses Aboard',
                'Parents/Children Aboard', 'AgeGroup', 'FareGroup']
max_depth = len(all_features) + 1


def decision_tree_accuracy(depth: int):
    x_train, x_test, y_train, y_test = train_test_split(
        df[all_features], df["Survived"], test_size=0.4, random_state=6)

    dt = DecisionTreeClassifier(max_depth=depth)
    dt.fit(x_train, y_train)
    y_pred = dt.predict(x_test)

    return accuracy_score(y_test, y_pred)

accuracies = pd.DataFrame(
    map(lambda d: [d, decision_tree_accuracy(d)], range(1, max_depth)),
    columns=['Decision Tree Depth', 'Accuracy'],
).sort_values(by='Accuracy', ascending=False)

accuracies
```

Out[2]:

	Decision Tree Depth	Accuracy
4	5	0.819718
5	6	0.819718
2	3	0.816901
3	4	0.814085
0	1	0.802817
1	2	0.777465

It seems like the best depth is 5, but as the values for 5, 6, 3, and 4 are very close in their accuracy.

3 might be the best choice for depth on a performance/computation trade-off.

(2) Build a Decision tree model with your selected stock / market index determine the number attributes that is capable of giving the best prediction of ‘daily returns’. (‘daily returns’ must first be converted into a decision class that will be used as the target(label))

(a) Find the best tree depth for the model with the selected attributes.

First simply import the dataset and set up the values:

In [3]:

```
stock_df = pd.read_csv("data/Nasdaq.csv", index_col='Date')
print(stock_df.head(3))


daily_return = np.empty(stock_df['Close'].shape)
#  From Slides: Daily return (r): Difference in percentage between
#  price at time t+1 and time t
```

```
daily_return[0] = float('NaN') # The first
daily_return[1:] = np.ediff1d(stock_df['Close']) / stock_df['Close'][:-1]
stock_df.insert(loc=len(stock_df.columns), column='Daily Return', value=daily_return)

binary = (daily_return > 0).astype(float)
stock_df.insert(loc=len(stock_df.columns), column='Binary Decision', value=binary)
stock_df["Binary Decision"] = le.fit_transform(stock_df["Binary Decision"])

stock_df['Rolling Mean 5'] = stock_df['Close'].rolling(5).mean()
stock_df['Rolling Mean 10'] = stock_df['Close'].rolling(10).mean()
stock_df['Rolling Mean 20'] = stock_df['Close'].rolling(20).mean()
stock_df['Rolling Mean 50'] = stock_df['Close'].rolling(50).mean()
stock_df['Rolling Mean 200'] = stock_df['Close'].rolling(200).mean()
stock_df = stock_df.fillna(0)  # NAs replaced with zero

print(stock_df.tail(3))
```

```
                  Open        High         Low       Close   Adj Close  Volume
Date
1971-02-05  100.000000  100.000000  100.000000  100.000000  100.000000       0
1971-02-08  100.839996  100.839996  100.839996  100.839996  100.839996       0
1971-02-09  100.760002  100.760002  100.760002  100.760002  100.760002       0
                    Open          High           Low         Close  \
Date
2021-09-17  15163.360352  15166.559570  14998.730469  15043.969727
2021-09-20  14758.139648  14841.820312  14530.070312  14713.900391
2021-09-21  14803.400391  14847.027344  14696.467773  14779.216797

               Adj Close      Volume  Daily Return  Binary Decision  \
Date
2021-09-17  15043.969727  6682650000     -0.009086                0
2021-09-20  14713.900391  4860630000     -0.021940                0
2021-09-21  14779.216797  3083208000      0.004439                1

            Rolling Mean 5  Rolling Mean 10  Rolling Mean 20  Rolling Mean 50  \
Date
2021-09-17    15106.151953     15191.898926     15143.947949     14875.465586
2021-09-20    15027.816016     15126.937012     15143.909961     14875.705195
2021-09-21    14976.107422     15067.425684     15135.738281     14876.624727

            Rolling Mean 200
Date
2021-09-17      13844.889136
2021-09-20      13856.711787
2021-09-21      13868.721973
```

Solving the task

In [4]:

```
stock_df["Class Daily Return"] = pd.qcut(stock_df["Daily Return"], q=3)
stock_df["Class Daily Return"], _ = stock_df["Class Daily Return"].factorize(sort=True)

all_stock_features = ['Volume', 'Rolling Mean 5', 'Rolling Mean 10',
                      'Rolling Mean 20', 'Rolling Mean 50', 'Rolling Mean 200']


def decision_tree_accuracy_stock(depth: int):
    x_train, x_test, y_train, y_test = train_test_split(
        stock_df[all_stock_features], stock_df["Class Daily Return"],
        test_size=0.4, random_state=6)

    dt = DecisionTreeClassifier(max_depth=depth)
    dt.fit(x_train, y_train)
    y_pred = dt.predict(x_test)

    return accuracy_score(y_test, y_pred)


accuracies = pd.DataFrame(
```

```python
    map(lambda d: [d, decision_tree_accuracy(d)], range(1, max_depth)),
    columns=['Decision Tree Depth', 'Accuracy'],
).sort_values(by='Accuracy', ascending=False)

accuracies
```

Out[4]:

	Decision Tree Depth	Accuracy
4	5	0.819718
5	6	0.819718
2	3	0.816901
3	4	0.814085
0	1	0.802817
1	2	0.777465

In the stock_df case, 3 seems like a good trade-off between performance and computation speed in terms of accuracy.

A depth of 5 would lead to the highest accuracy though.