Exercise 2 - Naive Bayes

First name: Brian

Last name: Schweigler

Matriculation number: 16-102-071

(1a) In the correlation visualization, select the two features that have the most significant correlation to the target feature, Survived.

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 PIL import Image
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing
import scipy
from mlxtend.evaluate import accuracy_score

df = pd.read_csv("data/titanic.csv", index_col='Name')
# df.describe(include='all')
pd.set_option('display.max_colwidth', None)
print(df.head(5))
le = preprocessing.LabelEncoder()
#df["Name"] = le.fit_transform(df["Name"])
df["Survived"] = le.fit_transform(df["Survived"])
df["Sex"] = le.fit_transform(df["Sex"])
```

```
                                                    Survived  Pclass     Sex  \
Name
Mr. Owen Harris Braund                                     0       3    male
Mrs. John Bradley (Florence Briggs Thayer) Cumings         1       1  female
Miss. Laina Heikkinen                                      1       3  female
Mrs. Jacques Heath (Lily May Peel) Futrelle                1       1  female
Mr. William Henry Allen                                    0       3    male

                                                     Age  \
Name
Mr. Owen Harris Braund                              22.0
Mrs. John Bradley (Florence Briggs Thayer) Cumings  38.0
Miss. Laina Heikkinen                               26.0
Mrs. Jacques Heath (Lily May Peel) Futrelle         35.0
Mr. William Henry Allen                             35.0

                                                    Siblings/Spouses Aboard  \
Name
Mr. Owen Harris Braund                                                    1
Mrs. John Bradley (Florence Briggs Thayer) Cumings                        1
Miss. Laina Heikkinen                                                     0
Mrs. Jacques Heath (Lily May Peel) Futrelle                               1
Mr. William Henry Allen                                                   0

                                                    Parents/Children Aboard  \
Name
Mr. Owen Harris Braund                                                    0
Mrs. John Bradley (Florence Briggs Thayer) Cumings                        0
Miss. Laina Heikkinen                                                     0
Mrs. Jacques Heath (Lily May Peel) Futrelle                               0
Mr. William Henry Allen                                                   0
```

```
                                                       Fare
Name
Mr. Owen Harris Braund                               7.2500
Mrs. John Bradley (Florence Briggs Thayer) Cumings  71.2833
Miss. Laina Heikkinen                                7.9250
Mrs. Jacques Heath (Lily May Peel) Futrelle         53.1000
Mr. William Henry Allen                              8.0500
```

In [2]:

```
correlation = df.corr()["Survived"]
correlation = correlation.apply(lambda entry: abs(entry))

print("Correlation of columns to column survived")
print(correlation.sort_values(ascending=False))

correlation.pop(correlation.idxmax())
temp_correlation = correlation
best = correlation.idxmax()
temp_correlation.pop(best)
second_best = temp_correlation.idxmax()

print(best + " has the highest correlation, " + second_best + " the second highest")
```

```
Correlation of columns to column survived
Survived                   1.000000
Sex                        0.542152
Pclass                     0.336528
Fare                       0.256179
Parents/Children Aboard    0.080097
Age                        0.059665
Siblings/Spouses Aboard    0.037082
Name: Survived, dtype: float64
Sex has the highest correlation, Pclass the second highest
```

(1b) Using Naive Bayes classifier and the most two significant features to predict the Survival of the travellers.

In [3]:

```
naive_bayes = GaussianNB()
x_train, x_test, y_train, y_test = train_test_split(df[[best, second_best]], df["Survived"], te
                                                    random_state=6)
naive_bayes.fit(x_train, y_train)
test_predictions = naive_bayes.predict(x_test)
accuracy = accuracy_score(y_test, test_predictions)
print("Accuracy for Naive Bayes ", round(accuracy, 3))
```

```
Accuracy for Naive Bayes  0.803
```

(1c) Compare the performance of your model when using all the attributes of the travellers.

In [4]:

```
naive_bayes_all = GaussianNB()
x_train_all, x_test_all, y_train_all, y_test_all = train_test_split(df.loc[:, df.columns != 'Su
                                                                     test_size=0.4, random_state
naive_bayes_all.fit(x_train_all, y_train_all)
test_predictions_all = naive_bayes_all.predict(x_test_all)
accuracy_all = accuracy_score(y_test_all, test_predictions_all)
print("Accuracy using all attributes for Naive Bayes ", round(accuracy_all, 3))
```

```
Accuracy using all attributes for Naive Bayes  0.752
```

Thus using all columns is not only less efficient, but also performs worse.

(2a) Select the two features that have the most signifcant correlation to the target feature, daily return.

First simply import the dataset and set up the values:

In [5]:

```
pd.set_option('display.max_colwidth', None)
stock_df = pd.read_csv("data/Nasdaq.csv", index_col='Date')
print(stock_df.head(5))
stock_df.describe(include='all')


daily_return = np.empty(stock_df['Close'].shape)
#  From Slides: Daily return (r): Difference in percentage between the price at time t+1 and at
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(5))
```

```
                  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
1971-02-10  100.690002  100.690002  100.690002  100.690002  100.690002       0
1971-02-11  101.449997  101.449997  101.449997  101.449997  101.449997       0
                    Open          High           Low         Close  \
Date
2021-09-15  15071.339844  15174.379883  14984.679688  15161.530273
2021-09-16  15120.089844  15205.500000  15047.139648  15181.919922
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
2021-09-15  15161.530273  4446270000      0.008231                1
2021-09-16  15181.919922  3681700000      0.001345                1
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
2021-09-15    15133.722070     15233.365918     15086.038477     14855.444590
2021-09-16    15120.456055     15220.619922     15118.838965     14865.781797
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
2021-09-15      13816.528940
2021-09-16      13831.444839
2021-09-17      13844.889136
2021-09-20      13856.711787
2021-09-21      13868.721973
```

Getting started with the task:

In [6]:

```
correlation_stock = stock_df.corr()["Binary Decision"]
correlation_stock = correlation_stock.apply(lambda entry: abs(entry))
```

```
print("Correlation of columns to column Binary Decision")
print(correlation_stock.sort_values(ascending=False))
```

```
Correlation of columns to column Binary Decision
Binary Decision     1.000000
Daily Return        0.660999
Volume              0.022448
Close               0.006831
Adj Close           0.006831
Low                 0.004667
Rolling Mean 200    0.003651
High                0.003244
Rolling Mean 50     0.002230
Open                0.001571
Rolling Mean 5      0.000909
Rolling Mean 20     0.000829
Rolling Mean 10     0.000093
Name: Binary Decision, dtype: float64
```

We have to ignore daily return, open, close; as all of those are directly related to the binary decision (daily return was used to create the binary decision).

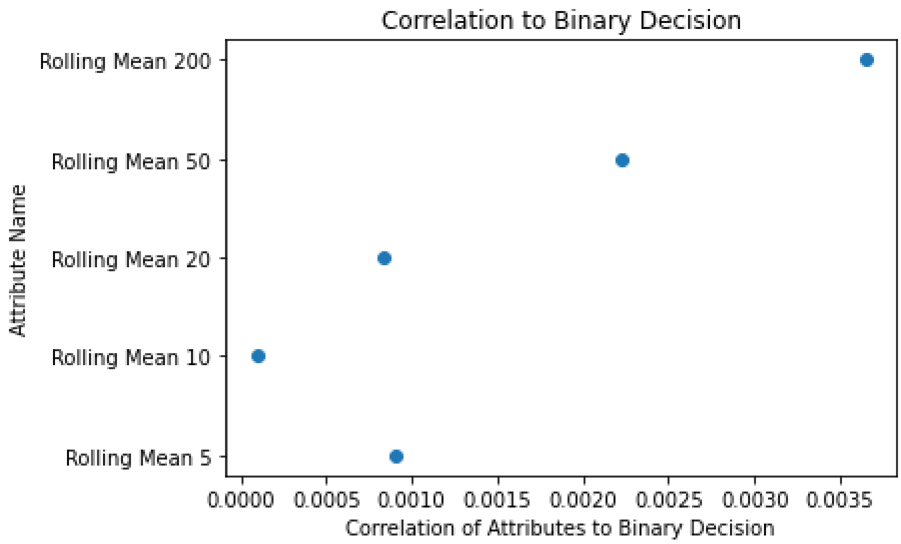
In [7]:

```
smaller_stock_df = stock_df[
    ['Binary Decision', 'Volume', 'Rolling Mean 5', 'Rolling Mean 10', 'Rolling Mean 20', 'Rol]
     'Rolling Mean 200']]

correlation_stock_fix = smaller_stock_df.corr()["Binary Decision"]
correlation_stock_fix = correlation_stock_fix.apply(lambda entry: abs(entry))

print("Correlation of columns to column Binary Decision")
temp_correlation_stock_fix = correlation_stock_fix
print(temp_correlation_stock_fix.sort_values(ascending=False))
temp_correlation_stock_fix.pop(temp_correlation_stock_fix.idxmax())
best_stock_attr_fix = temp_correlation_stock_fix.idxmax()
temp_correlation_stock_fix.pop(best_stock_attr_fix)
second_best_stock_attr_fix = temp_correlation_stock_fix.idxmax()

print(best_stock_attr_fix + " has the highest correlation, " + second_best_stock_attr_fix + " t
```

```
Correlation of columns to column Binary Decision
Binary Decision     1.000000
Volume              0.022448
Rolling Mean 200    0.003651
Rolling Mean 50     0.002230
Rolling Mean 5      0.000909
Rolling Mean 20     0.000829
Rolling Mean 10     0.000093
Name: Binary Decision, dtype: float64
Volume has the highest correlation, Rolling Mean 200 the second highest
```

In [8]:

```
plt.scatter(correlation_stock_fix,correlation_stock_fix.keys())
plt.ylabel("Attribute Name")
plt.xlabel("Correlation of Attributes to Binary Decision")
plt.title("Correlation to Binary Decision")
plt.show()
```



For some reason I just can't get 'Volume' and 'Binary Decision' to show up in the plot as somewhere their keys are lost:

In [9]:

```
print(correlation_stock_fix.keys())
```

```
Index(['Rolling Mean 5', 'Rolling Mean 10', 'Rolling Mean 20',
       'Rolling Mean 50', 'Rolling Mean 200'],
      dtype='object')
```

(2b) Using Naive Bayes classifier and the most two significant features predict daily return.

In [11]:

```
naive_bayes_stock = GaussianNB()
print("Length of DF: ", stock_df.shape[0])
test_size = 100/stock_df.shape[0]
print("Percentage of test_size to use last 100 days: ", test_size)
x_stock_train, x_stock_test, y_stock_train, y_stock_test = train_test_split(
    stock_df[[best_stock_attr_fix, second_best_stock_attr_fix]],
    stock_df["Binary Decision"], test_size=test_size,
    shuffle=False, random_state=6)
naive_bayes_stock.fit(x_stock_train, y_stock_train)
test_predictions_stock = naive_bayes_stock.predict(x_stock_test)
accuracy_stock = accuracy_score(y_stock_test, test_predictions_stock)
print("Accuracy for Naive Bayes ", round(accuracy_stock, 3))
```

```
Length of DF:  12402
Percentage of test_size to use last 100 days:  0.008063215610385421
Accuracy for Naive Bayes  0.57
```