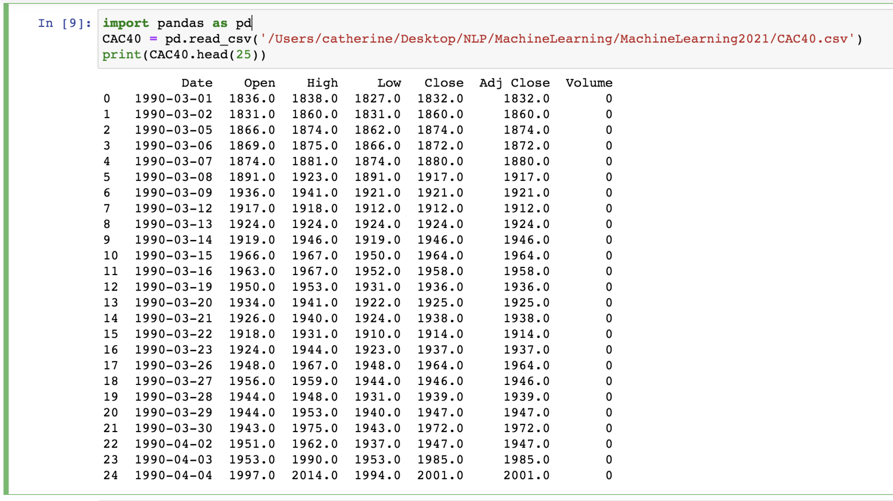
Some Python Libraries

October 3, 2022

Data exploration

- Can be done with pandas DataFrames.
- DataFrame is a 2-dimensional labelled data structure with columns of potentially different types.
- import pandas as pd or import numpy as np

Display the data



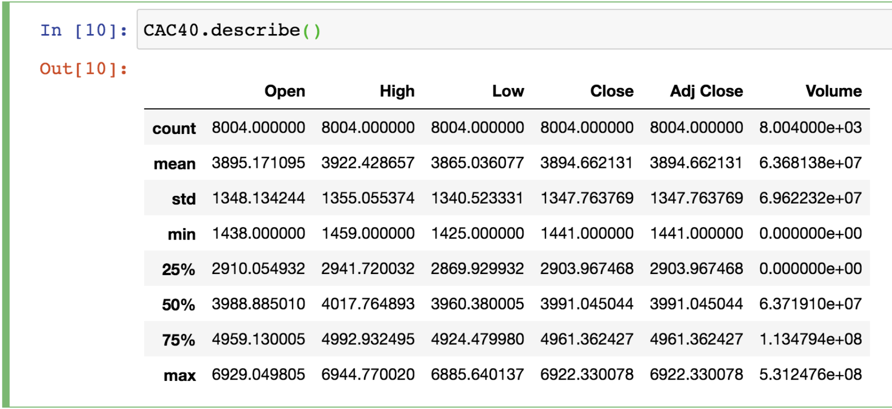
In [9]:

```
import pandas as pd
CAC40 = pd.read_csv('/Users/catherine/Desktop/NLP/MachineLearning/MachineLearning2021/CAC40.csv')
print(CAC40.head(25))
```

```
          Date    Open    High     Low   Close  Adj Close  Volume
0   1990-03-01  1836.0  1838.0  1827.0  1832.0     1832.0       0
1   1990-03-02  1831.0  1860.0  1831.0  1860.0     1860.0       0
2   1990-03-05  1866.0  1874.0  1862.0  1874.0     1874.0       0
3   1990-03-06  1869.0  1875.0  1866.0  1872.0     1872.0       0
4   1990-03-07  1874.0  1881.0  1874.0  1880.0     1880.0       0
5   1990-03-08  1891.0  1923.0  1891.0  1917.0     1917.0       0
6   1990-03-09  1936.0  1941.0  1921.0  1921.0     1921.0       0
7   1990-03-12  1917.0  1918.0  1912.0  1912.0     1912.0       0
8   1990-03-13  1924.0  1924.0  1924.0  1924.0     1924.0       0
9   1990-03-14  1919.0  1946.0  1919.0  1946.0     1946.0       0
10  1990-03-15  1966.0  1967.0  1950.0  1964.0     1964.0       0
11  1990-03-16  1963.0  1967.0  1952.0  1958.0     1958.0       0
12  1990-03-19  1950.0  1953.0  1931.0  1936.0     1936.0       0
13  1990-03-20  1934.0  1941.0  1922.0  1925.0     1925.0       0
14  1990-03-21  1926.0  1940.0  1924.0  1938.0     1938.0       0
15  1990-03-22  1918.0  1931.0  1910.0  1914.0     1914.0       0
16  1990-03-23  1924.0  1944.0  1923.0  1937.0     1937.0       0
17  1990-03-26  1948.0  1967.0  1948.0  1964.0     1964.0       0
18  1990-03-27  1956.0  1959.0  1944.0  1946.0     1946.0       0
19  1990-03-28  1944.0  1948.0  1931.0  1939.0     1939.0       0
20  1990-03-29  1944.0  1953.0  1940.0  1947.0     1947.0       0
21  1990-03-30  1943.0  1975.0  1943.0  1972.0     1972.0       0
22  1990-04-02  1951.0  1962.0  1937.0  1947.0     1947.0       0
23  1990-04-03  1953.0  1990.0  1953.0  1985.0     1985.0       0
24  1990-04-04  1997.0  2014.0  1994.0  2001.0     2001.0       0
```

Describe the data(mean, std and IQR)

uses the function describe(), which gives summary about numeric columns. describe(include='all') will include even non numeric values.



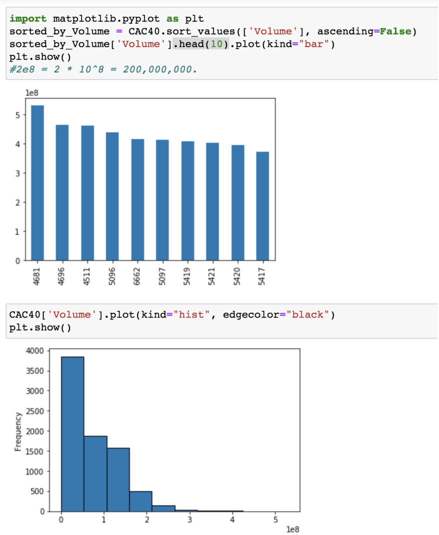
```
In [10]: CAC40.describe()
```

Out[10]:

	Open	High	Low	Close	Adj Close	Volume
count	8004.000000	8004.000000	8004.000000	8004.000000	8004.000000	8.004000e+03
mean	3895.171095	3922.428657	3865.036077	3894.662131	3894.662131	6.368138e+07
std	1348.134244	1355.055374	1340.523331	1347.763769	1347.763769	6.962232e+07
min	1438.000000	1459.000000	1425.000000	1441.000000	1441.000000	0.000000e+00
25%	2910.054932	2941.720032	2869.929932	2903.967468	2903.967468	0.000000e+00
50%	3988.885010	4017.764893	3960.380005	3991.045044	3991.045044	6.371910e+07
75%	4959.130005	4992.932495	4924.479980	4961.362427	4961.362427	1.134794e+08
max	6929.049805	6944.770020	6885.640137	6922.330078	6922.330078	5.312476e+08

plotting libraries

- Matplotlib
  - pip install matplotlib or conda install matplotlib
  - import matplotlib.pyplot as plt



```
import matplotlib.pyplot as plt
sorted_by_Volume = CAC40.sort_values(['Volume'], ascending=False)
sorted_by_Volume['Volume'].head(10).plot(kind="bar")
plt.show()
#2e8 = 2 * 10^8 = 200,000,000.
```

```
CAC40['Volume'].plot(kind="hist", edgecolor="black")
plt.show()
```

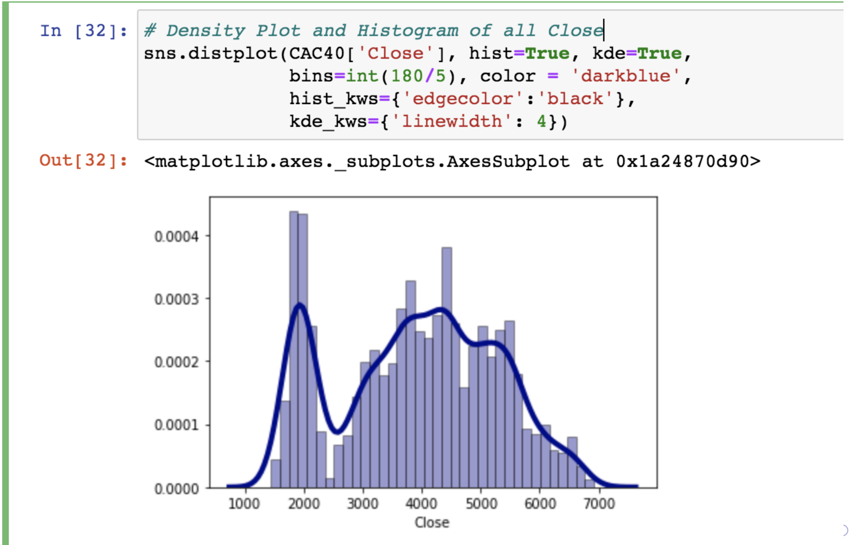
Other libraries

- Pandas
- Seaborn
- Plotly

Density of values

A smooth version of the histogram.

The y-axis is the of density, and the histogram is normalized by default so that it has the same y-scale as the density plot.

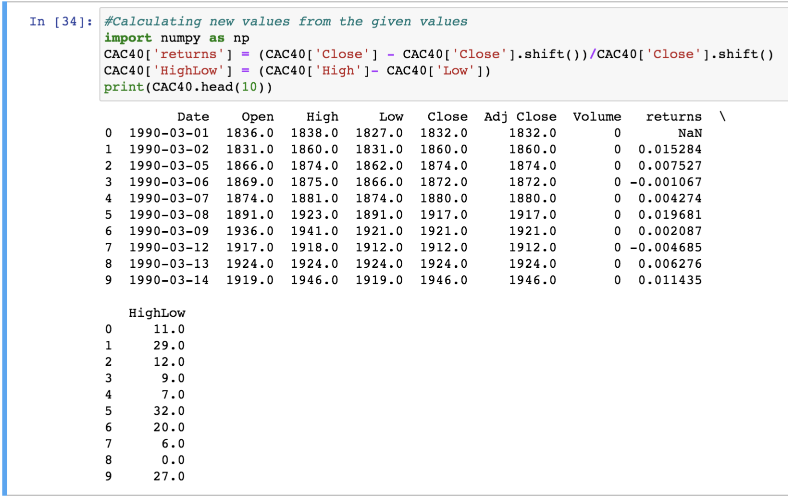


In [32]:
```
# Density Plot and Histogram of all Close
sns.distplot(CAC40['Close'], hist=True, kde=True,
             bins=int(180/5), color = 'darkblue',
             hist_kws={'edgecolor':'black'},
             kde_kws={'linewidth': 4})
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a24870d90>
```

Calculating new values

New columns of values can be created by calculating using given values.



```
In [34]: #Calculating new values from the given values
         import numpy as np
         CAC40['returns'] = (CAC40['Close'] - CAC40['Close'].shift())/CAC40['Close'].shift()
         CAC40['HighLow'] = (CAC40['High']- CAC40['Low'])
         print(CAC40.head(10))
```

```
         Date    Open    High     Low   Close  Adj Close  Volume   returns  \
0  1990-03-01  1836.0  1838.0  1827.0  1832.0     1832.0       0       NaN
1  1990-03-02  1831.0  1860.0  1831.0  1860.0     1860.0       0  0.015284
2  1990-03-05  1866.0  1874.0  1862.0  1874.0     1874.0       0  0.007527
3  1990-03-06  1869.0  1875.0  1866.0  1872.0     1872.0       0 -0.001067
4  1990-03-07  1874.0  1881.0  1874.0  1880.0     1880.0       0  0.004274
5  1990-03-08  1891.0  1923.0  1891.0  1917.0     1917.0       0  0.019681
6  1990-03-09  1936.0  1941.0  1921.0  1921.0     1921.0       0  0.002087
7  1990-03-12  1917.0  1918.0  1912.0  1912.0     1912.0       0 -0.004685
8  1990-03-13  1924.0  1924.0  1924.0  1924.0     1924.0       0  0.006276
9  1990-03-14  1919.0  1946.0  1919.0  1946.0     1946.0       0  0.011435

   HighLow
0     11.0
1     29.0
2     12.0
3      9.0
4      7.0
5     32.0
6     20.0
7      6.0
8      0.0
9     27.0
```

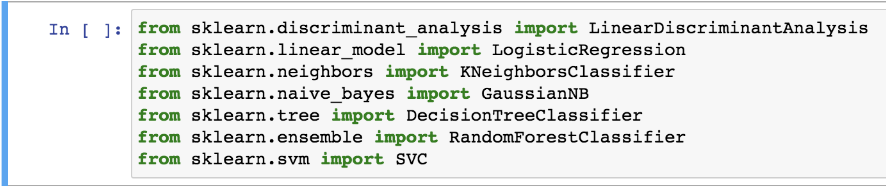
Machine Learning Algorithms

Here is the list of some commonly used machine learning algorithms. These algorithms can be applied to almost any data problem:

- Linear Regression
- Logistic Regression
- Decision Tree
- SVM
- Naive Bayes
- kNN
- K-Means
- Random Forest

Machine Learning Libraries

Most of the ML algorithms can be accessed using Scikit-Learn which is a library for Python



```python
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

Making prediction with Machine Learning

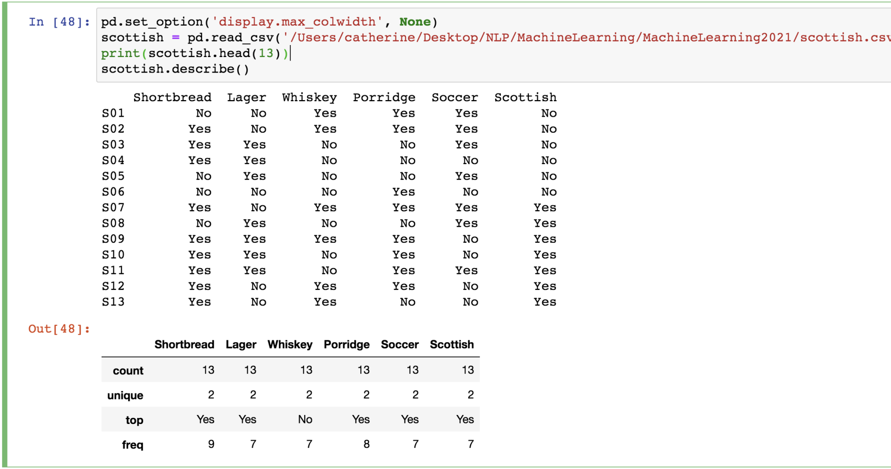
	Attributes					
Sample	Shortbread	Lager	Whiskey	Porridge	Soccer	Scottish
S01	No	No	Yes	Yes	Yes	No
S02	Yes	No	Yes	Yes	Yes	No
S03	Yes	Yes	No	No	Yes	No
S04	Yes	Yes	No	No	No	No
S05	No	Yes	No	No	Yes	No
S06	No	No	No	Yes	No	No
S07	Yes	No	Yes	Yes	Yes	Yes
S08	No	Yes	No	No	Yes	Yes
S09	Yes	Yes	Yes	Yes	No	Yes
S10	Yes	Yes	No	Yes	No	Yes
S11	Yes	Yes	No	Yes	Yes	Yes
S12	Yes	No	Yes	Yes	No	Yes
S13	Yes	No	Yes	No	No	Yes

Making prediction with Machine Learning

Decide whether Logan is Scottish based on the following attributes and using a Naïve Bayes classifier.
Logan likes shortbread, drinks whiskey and eats porridge but doesn't like lager and doesn't watch soccer.

	Attributes					
Sample	Shortbread	Lager	Whiskey	Porridge	Soccer	Scottish
Logan	Yes	No	Yes	Yes	No	?

Making prediction with Machine Learning



In [48]:

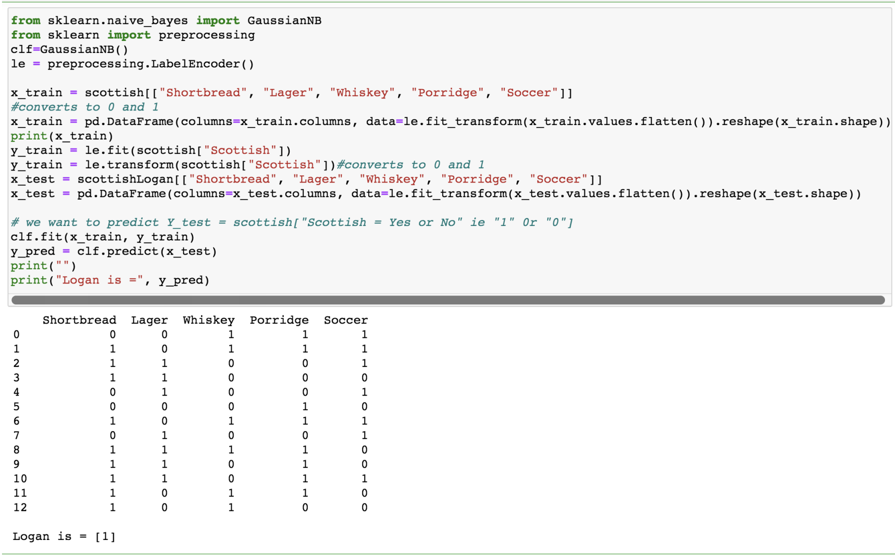
```
pd.set_option('display.max_colwidth', None)
scottish = pd.read_csv('/Users/catherine/Desktop/NLP/MachineLearning/MachineLearning2021/scottish.csv
print(scottish.head(13))
scottish.describe()
```

```
    Shortbread Lager Whiskey Porridge Soccer Scottish
S01         No    No     Yes      Yes    Yes       No
S02        Yes    No     Yes      Yes    Yes       No
S03        Yes   Yes      No       No    Yes       No
S04        Yes   Yes      No       No     No       No
S05         No   Yes      No       No    Yes       No
S06         No    No      No      Yes     No       No
S07        Yes    No     Yes      Yes    Yes      Yes
S08         No   Yes      No       No    Yes      Yes
S09        Yes   Yes     Yes      Yes     No      Yes
S10        Yes   Yes      No      Yes     No      Yes
S11        Yes   Yes      No      Yes    Yes      Yes
S12        Yes    No     Yes      Yes     No      Yes
S13        Yes    No     Yes       No     No      Yes
```

Out[48]:

	Shortbread	Lager	Whiskey	Porridge	Soccer	Scottish
count	13	13	13	13	13	13
unique	2	2	2	2	2	2
top	Yes	Yes	No	Yes	Yes	Yes
freq	9	7	7	8	7	7

Making prediction with Machine Learning



```
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing
clf=GaussianNB()
le = preprocessing.LabelEncoder()

x_train = scottish[["Shortbread", "Lager", "Whiskey", "Porridge", "Soccer"]]
#converts to 0 and 1
x_train = pd.DataFrame(columns=x_train.columns, data=le.fit_transform(x_train.values.flatten()).reshape(x_train.shape))
print(x_train)
y_train = le.fit(scottish["Scottish"])
y_train = le.transform(scottish["Scottish"])#converts to 0 and 1
x_test = scottishLogan[["Shortbread", "Lager", "Whiskey", "Porridge", "Soccer"]]
x_test = pd.DataFrame(columns=x_test.columns, data=le.fit_transform(x_test.values.flatten()).reshape(x_test.shape))

# we want to predict Y_test = scottish["Scottish = Yes or No" ie "1" Or "0"]
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
print("")
print("Logan is =", y_pred)
```

```
    Shortbread  Lager  Whiskey  Porridge  Soccer
0            0      0        1         1       1
1            1      0        1         1       1
2            1      1        0         0       1
3            1      1        0         0       0
4            0      1        0         0       1
5            0      0        0         1       0
6            1      0        1         1       1
7            0      1        0         0       1
8            1      1        1         1       0
9            1      1        0         1       0
10           1      1        0         1       1
11           1      0        1         1       0
12           1      0        1         0       0

Logan is = [1]
```