W4_ML&DM_vedasri

October 30, 2021

1 Week 4: Machine Learning and Data Mining

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from mlxtend.classifier import OneRClassifier
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

1.0.1 Loading Data and replacing 'Sex' attribute with categorical values

[5]:
```
pd.set_option('display.max_colwidth',None)
titanic = pd.read_csv('./titanic.csv')
print("Number of samples in original data: {}\n".format(len(titanic.index)))

columns = titanic.columns
print("Features present in dataset: \n", list(columns))

titanic.loc[titanic['Sex'] == 'male', 'Sex']=1
titanic.loc[titanic['Sex'] == 'female', 'Sex']=0
titanic.head(5)
```

```
Number of samples in original data: 887

Features present in dataset:
 ['Survived', 'Pclass', 'Name', 'Sex', 'Age', 'Siblings/Spouses Aboard',
'Parents/Children Aboard', 'Fare']
```

[5]:
```
   Survived  Pclass                                               Name Sex  \
0         0       3                             Mr. Owen Harris Braund   1
1         1       1  Mrs. John Bradley (Florence Briggs Thayer) Cumings   0
```

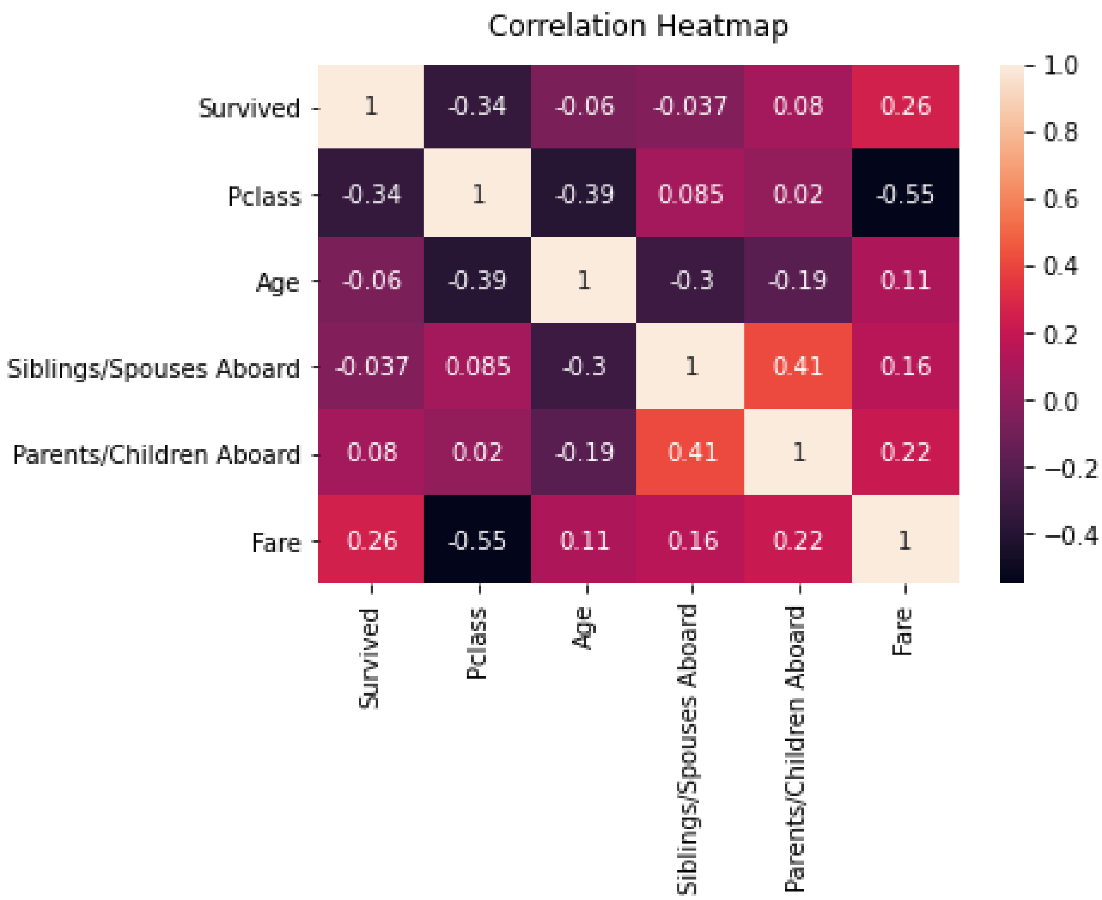
```
2         1       3                              Miss. Laina Heikkinen   0
3         1       1         Mrs. Jacques Heath (Lily May Peel) Futrelle   0
4         0       3                            Mr. William Henry Allen   1

    Age  Siblings/Spouses Aboard  Parents/Children Aboard     Fare
0  22.0                        1                        0   7.2500
1  38.0                        1                        0  71.2833
2  26.0                        0                        0   7.9250
3  35.0                        1                        0  53.1000
4  35.0                        0                        0   8.0500
```

1.1 Problem 1:

Q1. Selecting two features that have the most significant correlation to the target feature, Survived from correlation visualization

Ans. Basing on below plot, we choose 'Pclass' and 'Fare' that has high correlation (magnitude wise) with 'Survived' label

[8]:
```python
heat_map = sns.heatmap(titanic.corr(), annot=True);
heat_map.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12);
```



[9]:
```
Selecting_features = titanic[['Survived', 'Pclass', 'Fare']]
Selecting_features.head(5)
```

[9]:

	Survived	Pclass	Fare
0	0	3	7.2500
1	1	1	71.2833
2	1	3	7.9250
3	1	1	53.1000
4	0	3	8.0500

1.1.1 Q2. Using Naive Bayes classifier and the most two significant features to predict the Survival of the travellers.

1.1.2 Ans. Accuracy_score on test set with top 2 features: 64.86%

[23]:
```
clf = GaussianNB()
le = preprocessing.LabelEncoder()

x = titanic[["Pclass", "Fare"]]
y = le.fit(titanic["Survived"])
y = le.transform(titanic["Survived"])

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25)
print("x training samples  : {}".format(x_train.shape))
print("x test samples      : {}".format(x_test.shape))
print("y training samples  : {}".format(y_train.shape))
print("y test samples      : {}".format(y_test.shape))
```

```
x training samples  : (665, 2)
x test samples      : (222, 2)
y training samples  : (665,)
y test samples      : (222,)
```

[24]:
```
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
accu = 100*np.mean(y_pred ==y_test)
print("Accuracy on test samples with only top 2 features: {:.2f}%".format(accu))
print("\n confusion_matrix: \n", confusion_matrix(y_test, y_pred))
```

```
Accuracy on test samples with only top 2 features: 71.17%

 confusion_matrix:
 [[127  15]
 [ 49  31]]
```

1.1.3 Q3. Comparing the performance of titanic model when using all the attributes of the travellers.

1.1.4 Ans. Accuracy_score on test set with all features: 80.18%

[25]:
```
xd = titanic[["Pclass","Sex","Age","Siblings/Spouses Aboard","Parents/Children␣
↪Aboard","Fare"]]
yd = le.fit(titanic["Survived"])
yd = le.transform(titanic["Survived"])
```

```
xd_train, xd_test, yd_train, yd_test = train_test_split(xd, yd, test_size = 0.
↪25)
print("No of training samples  : {}".format(xd_train.shape))
print("No of test samples      : {}".format(xd_test.shape))
print("y training samples      : {}".format(yd_train.shape))
print("y test samples          : {}".format(yd_test.shape))
```

```
No of training samples  : (665, 6)
No of test samples      : (222, 6)
y training samples      : (665,)
y test samples          : (222,)
```

[27]:
```
clf.fit(xd_train, yd_train)
yd_pred = clf.predict(xd_test)
acc = accuracy_score(yd_test, yd_pred)
print("Accuracy_score on test set with all features: {:.2f}%".format(100 * acc))
print("\n confusion_matrix: \n", confusion_matrix(yd_test, yd_pred))
```

```
Accuracy_score on test set with all features: 80.18%

 confusion_matrix:
 [[119  12]
 [ 32  59]]
```

1.2 Problem 2:

Loading and Calculating dailt returns

[71]:
```
df = pd.read_csv('./IBM.txt', delimiter = " ")
df_raw = df
print("Number of rows in original data: {}".format(len(df.index)))
print("Features: ", list(df.columns))
df.head(5)
```

```
Number of rows in original data: 3692
Features:  ['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adjusted']
```

[71]:
```
         Date       Open        High        Low       Close    Volume  \
0  2007-01-03  97.180000   98.400002  96.260002   97.269997   9196800
1  2007-01-04  97.250000   98.790001  96.879997   98.309998  10524500
2  2007-01-05  97.599998   97.949997  96.910004   97.419998   7221300
3  2007-01-08  98.500000   99.500000  98.349998   98.900002  10340000
4  2007-01-09  99.080002  100.330002  99.070000  100.070000  11108200

    Adjusted
0  63.127567
1  63.802544
```

```
2   63.224930
3   64.185463
4   64.944771
```

Calculate daily returns using previous day's close price

[76]:
```
df['Daily_returns'] = 100*((df['Close'] - df['Close'].shift())/ df['Close'].
 ↪shift())
conditions = [(df['Daily_returns'] >= 0.0),(df['Daily_returns'] < 0.0)]

# 1 for UP. -1 for Down
values = [1, -1]
df['Decision'] = np.select(conditions, values)

df['Decision(next_day)'] =  df['Decision'].shift(-1)
print("Number of rows in processed data: {}".format(len(df.index)))

# remove the first row and last row, since it is not possible to
# calculate first daily return and last day ground truth
df_new = df[1:-2]
df_new['Decision(next_day)'] = df_new['Decision(next_day)'].astype('int32')


df_new.head(8)
```

```
Number of rows in processed data: 3692
```

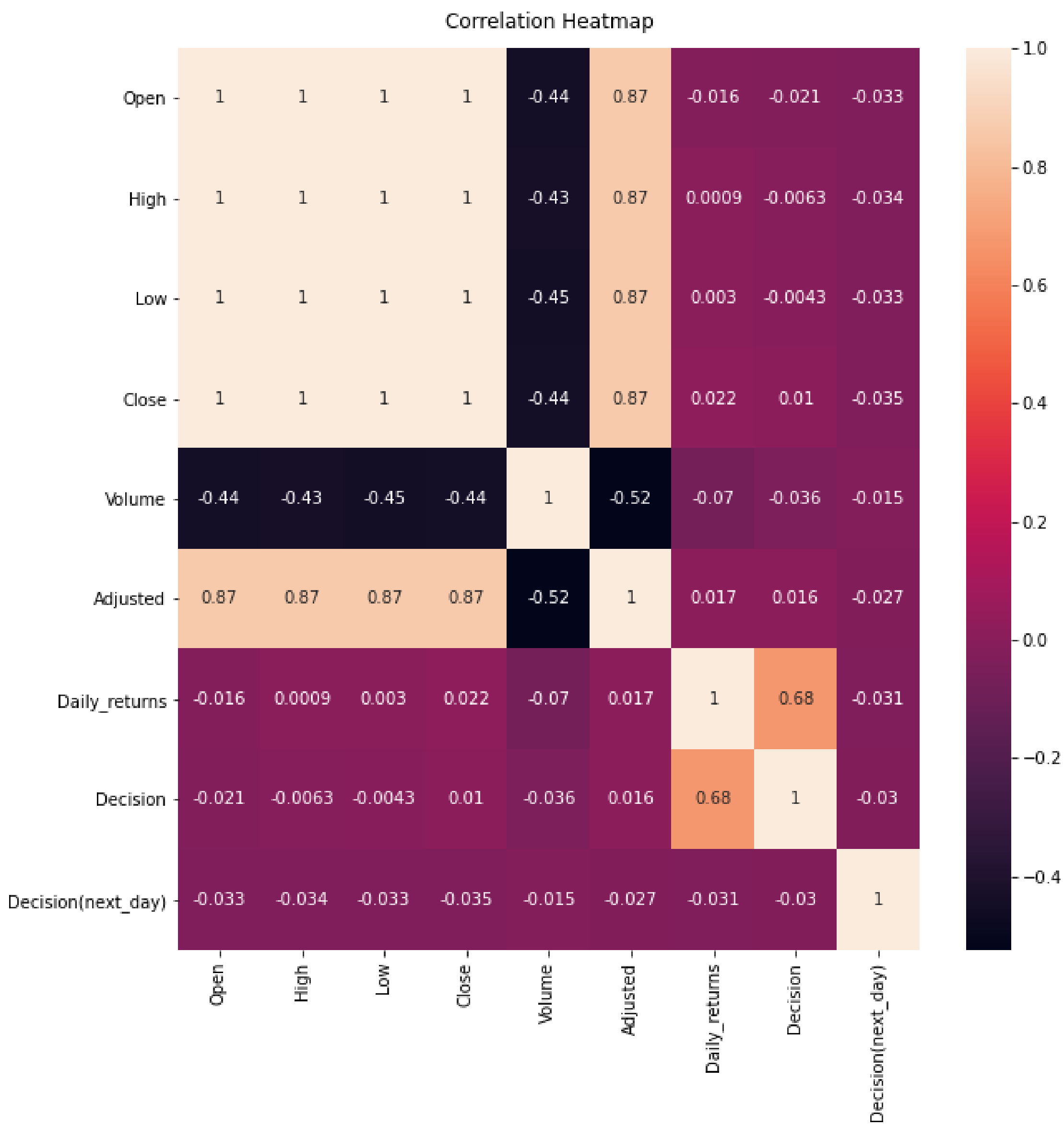
```
/Users/krishna/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[76]:
```
         Date       Open        High        Low       Close    Volume  \
1  2007-01-04  97.250000   98.790001  96.879997   98.309998  10524500
2  2007-01-05  97.599998   97.949997  96.910004   97.419998   7221300
3  2007-01-08  98.500000   99.500000  98.349998   98.900002  10340000
4  2007-01-09  99.080002  100.330002  99.070000  100.070000  11108200
5  2007-01-10  98.500000   99.050003  97.930000   98.889999   8744800
6  2007-01-11  99.000000   99.900002  98.500000   98.650002   8000700
7  2007-01-12  98.989998   99.690002  98.500000   99.339996   6636500
8  2007-01-16  99.400002  100.839996  99.300003  100.820000   9602200

    Adjusted  Daily_returns  Decision  Decision(next_day)
```

1	63.802544	1.069190	1	-1
2	63.224930	-0.905300	-1	1
3	64.185463	1.519199	1	1
4	64.944771	1.183011	1	-1
5	64.178978	-1.179176	-1	-1
6	64.023201	-0.242691	-1	1
7	64.471024	0.699436	1	1
8	65.431503	1.489837	1	-1

Generating a correlation visualization of Volume

[79]:
```
fig, ax = plt.subplots(figsize=(10,10))
heat_map_IBM = sns.heatmap(df_new.corr(), annot=True);
heat_map_IBM.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12);
```



1.2.1 Q1. select the two features that have the most significant correlation to the target feature, daily return.

1.2.2 Ans. We choose 'Close' and 'High' as the top features to target feature 'Decision(next_day')

[81]:
```
Selecting_features_IBM = df_new[['Daily_returns', 'Close', 'High']]
Selecting_features_IBM.head(5)
```

[81]:

	Daily_returns	Close	High
1	1.069190	98.309998	98.790001
2	-0.905300	97.419998	97.949997
3	1.519199	98.900002	99.500000
4	1.183011	100.070000	100.330002
5	-1.179176	98.889999	99.050003

Generating a correlation visualization of the moving average (with a period of 5, 10, 20, 50 or 200).

[93]:

```
df_raw = pd.read_csv('./IBM.txt', delimiter = " ")

for i in [5,10, 20, 50,200]:
    df_raw['MA_{}'.format(i)] = df_raw['Close'].rolling(window=i).mean()
```

[94]:

```
df_raw.head(20)
```

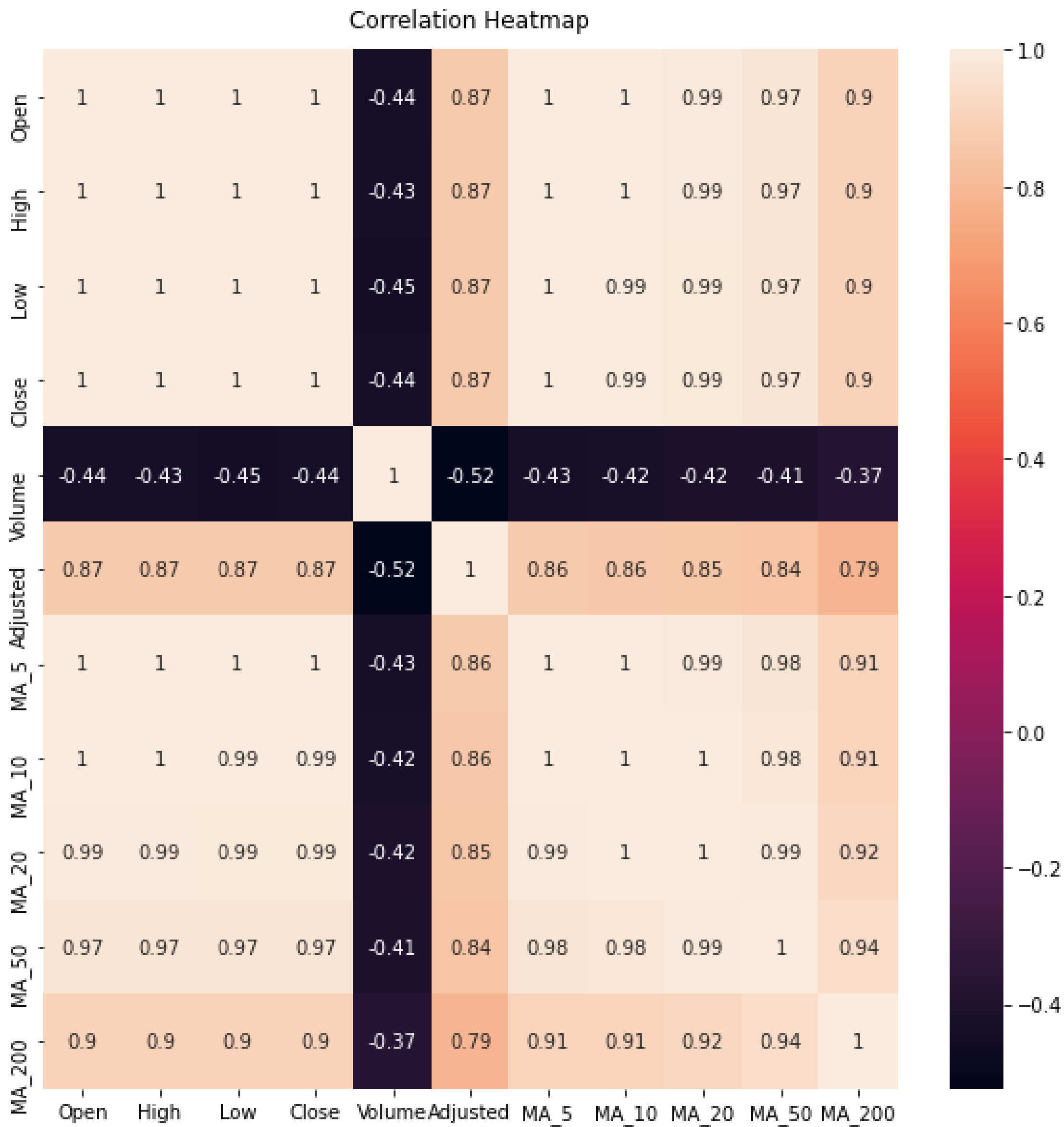
[94]:

	Date	Open	High	Low	Close	Volume	\
0	2007-01-03	97.180000	98.400002	96.260002	97.269997	9196800	
1	2007-01-04	97.250000	98.790001	96.879997	98.309998	10524500	
2	2007-01-05	97.599998	97.949997	96.910004	97.419998	7221300	
3	2007-01-08	98.500000	99.500000	98.349998	98.900002	10340000	
4	2007-01-09	99.080002	100.330002	99.070000	100.070000	11108200	
5	2007-01-10	98.500000	99.050003	97.930000	98.889999	8744800	
6	2007-01-11	99.000000	99.900002	98.500000	98.650002	8000700	
7	2007-01-12	98.989998	99.690002	98.500000	99.339996	6636500	
8	2007-01-16	99.400002	100.839996	99.300003	100.820000	9602200	
9	2007-01-17	100.690002	100.900002	99.900002	100.019997	8200700	
10	2007-01-18	99.800003	99.949997	98.910004	99.449997	14636100	
11	2007-01-19	95.000000	96.849998	94.550003	96.169998	26035800	
12	2007-01-22	96.419998	97.230003	96.120003	97.110001	13539300	
13	2007-01-23	96.910004	97.379997	96.199997	97.080002	10337400	
14	2007-01-24	97.080002	97.580002	96.580002	97.400002	5700000	
15	2007-01-25	97.220001	97.919998	97.220001	97.510002	6201300	
16	2007-01-26	97.519997	97.830002	96.839996	97.449997	5771100	
17	2007-01-29	97.699997	98.660004	97.449997	98.540001	7294800	
18	2007-01-30	98.570000	99.449997	98.500000	99.370003	7177900	
19	2007-01-31	98.800003	99.480003	98.349998	99.150002	6432600	

	Adjusted	MA_5	MA_10	MA_20	MA_50	MA_200
0	63.127567	NaN	NaN	NaN	NaN	NaN
1	63.802544	NaN	NaN	NaN	NaN	NaN
2	63.224930	NaN	NaN	NaN	NaN	NaN
3	64.185463	NaN	NaN	NaN	NaN	NaN
4	64.944771	98.393999	NaN	NaN	NaN	NaN
5	64.178978	98.717999	NaN	NaN	NaN	NaN
6	64.023201	98.786000	NaN	NaN	NaN	NaN

7	64.471024	99.170000	NaN	NaN	NaN	NaN
8	65.431503	99.553999	NaN	NaN	NaN	NaN
9	64.912323	99.543999	98.968999	NaN	NaN	NaN
10	64.542397	99.655998	99.186999	NaN	NaN	NaN
11	62.413712	99.159998	98.972999	NaN	NaN	NaN
12	63.023762	98.713999	98.941999	NaN	NaN	NaN
13	63.004272	97.965999	98.759999	NaN	NaN	NaN
14	63.211952	97.442000	98.492999	NaN	NaN	NaN
15	63.283344	97.054001	98.355000	NaN	NaN	NaN
16	63.244381	97.310001	98.234999	NaN	NaN	NaN
17	63.951797	97.596001	98.155000	NaN	NaN	NaN
18	64.490479	98.054001	98.010000	NaN	NaN	NaN
19	64.347694	98.404001	97.923001	98.446	NaN	NaN

[95]:
```
fig, ax = plt.subplots(figsize=(10,10))
heat_map_IBM = sns.heatmap(df_raw.corr(), annot=True);
heat_map_IBM.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12);
```



1.2.3 Q2. Finding Naive Bayes classifier Using Naive Bayes classifier and the most two significant features predict daily return.

[89]:
```
# predicting daily returns of 'NEXT DAY' using shift(-1) operation.
df_new_IBM = df_new.copy()
xd_IBM = df_new_IBM[[ "High", "Close"]]
le = preprocessing.LabelEncoder()
decision = le.fit(df_new_IBM["Decision"].shift(-1))
decision = le.transform(df_new_IBM["Decision"].shift(-1))
```

[96]:

```
# Split the IBM data into training and testing

xd_train_IBM =  xd_IBM[:-102]
xd_test_IBM  =  xd_IBM[-102:-2]

yd_train_IBM =  decision[:-102]
yd_test_IBM  =  decision[-102:-2]

print("No of training samples : {}".format(xd_train_IBM.shape))
print("No of test samples     : {}\n".format(xd_test_IBM.shape))
print("y training samples : {}".format(yd_train_IBM.shape))
print("y test samples     : {}\n".format(yd_test_IBM.shape))
```

```
No of training samples : (3587, 2)
No of test samples     : (100, 2)

y training samples : (3587,)
y test samples     : (100,)
```

[97]:

```
# Finding Gaussian Classifier

clf.fit(xd_train_IBM, yd_train_IBM)
yd_pred_IBM = clf.predict(xd_test_IBM)
acc = accuracy_score(yd_test_IBM, yd_pred_IBM)

print("Accuracy_score on test samples with Gaussian NB: {:.2f}%".format(100 *␣
↪acc))
print("\n confusion_matrix: \n", confusion_matrix(yd_test_IBM, yd_pred_IBM))
```

```
Accuracy_score on test samples with Gaussian NB: 56.00%

 confusion_matrix:
 [[ 0 44]
 [ 0 56]]
```