Naive Bayes classifier

October 10, 2022

Making prediction with Naive Bayes classifier

	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 Naive Bayes classifier

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 Naive Bayes classifier

p(C=yes) = 7/13

p(c=no) = 6/13

Logan : Shortbread=yes, Lager=No, Whiskey=Yes, Porridge=Yes, Soccer=No

p(shortbread=yes | C=yes) = 6/7	p(shortbread=yes | C=no) = 3/6
p(lager=no | C=yes) = 3/7	p(lager=no | C=no) = 3/6
p(whiskey=yes | C=yes) = 4/7	p(whiskey=yes | C=no) = 2/6
p(porridge=yes | C=yes) = 5/7	p(porridge=yes | C=no) = 3/6
p(soccer=no | C=yes) = 4/7	p(soccer=no | C=no) = 2/6

p(C=yes | Logan) = 7/13 x 6/7 x 3/7 x 4/7 x 5/7 x 4/7 = 0.0461

p(C=no | Logan) = 6/13 x 3/6 x 3/6 x 2/6 x 3/6 x 2/6 = 0.006409

p(C=yes | Logan) = 0.0461 / (0.0461 + 0.006409) = 0.8779 = **87.79%**

p(C=no | Logan) = 0.006409 / (0.0461 + 0.006409) = 0.12205 = 12.205%

Logan is Scottish because p(C=yes | Logan) > p(C=no | Logan)

Making prediction with Naive Bayes classifier

Smoothing

p(C=yes) = 7+1/13+2

p(c=no) = 6+1/13+2

p(shortbread=yes | C=yes) = 6+1/7+2 p(shortbread=yes | C=no) = 3+1/6+2

p(lager=no | C=yes) = 3+1/7+2 p(lager=no | C=no) = 3+1/6+2

p(whiskey=yes | C=yes) = 4+1/7+2 p(whiskey=yes | C=no) = 2+1/6+2

p(porridge=yes | C=yes) = 5+1/7+2 p(porridge=yes | C=no) = 3+1/6+2

p(soccer=no | C=yes) = 4+1/7+2 p(soccer=no | C=no) = 2+1/6+2

p(C=yes | Logan) = 7+1/13+2 x 6+1/7+2 x 3+1/7+2 x 4+1/7+2 x 5+1/7+2 x 4+1/7+2 = 0.0379

p(C=no | Logan) = 6+1/13+2 x 3+1/6+2 x 3+1/6+2 x 2+1/6+2 x 3+1/6+2 x 2+1/6+2 = 0.0082

p(C=yes | Logan) = 0.0379/ (0.0379+ 0.0082) = 0.822 = 82.2%

p(C=no | Logan) = 0.0082/ (0.0379+ 0.0082) = 0.178 = 17.8%

Logan is Scottish because p(C=yes | Logan) > p(C=no | Logan)

Making prediction with Naive Bayes classifier

```
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
```

	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

```
scottishLogan = pd.read_csv('/Users/catherine/Desktop/NLP/MachineLearning/MachineLearning2021/scottishLogan.csv')
print(scottishLogan.head(1))
scottishLogan.describe()
```

```
      Shortbread Lager Whiskey Porridge Soccer
Logan        Yes    No     Yes       Yes     No
```

	Shortbread	Lager	Whiskey	Porridge	Soccer
count	1	1	1	1	1
unique	1	1	1	1	1
top	Yes	No	Yes	Yes	No
freq	1	1	1	1	1

Making prediction with Naive Bayes classifier

```
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_trai
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.sh

# we want to predict Y_test = scottish["Scottish = Yes or No" ie "1" 0r "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]
```

Classification with Naive Bayes classifier

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

Xd_train_nb, Xd_test_nb, y_train_nb, y_test_nb = train_test_split(x_train, y_train, test_size=0.35)
print(Xd_train_nb)
print("y_train = ",y_train_nb, "\n")
print(Xd_test_nb)
print("y_test = ",y_test_nb)
```

```
    Shortbread  Lager  Whiskey  Porridge  Soccer
5            0      0        0         1       0
8            1      1        1         1       0
11           1      0        1         1       0
7            0      1        0         0       1
1            1      0        1         1       1
4            0      1        0         0       1
3            1      1        0         0       0
10           1      1        0         1       1
y_train =  [0 1 1 1 0 0 0 1]

    Shortbread  Lager  Whiskey  Porridge  Soccer
6            1      0        1         1       1
0            0      0        1         1       1
12           1      0        1         0       0
2            1      1        0         0       1
9            1      1        0         1       0
y_test =  [1 0 1 0 1]
```

```
clf.fit(Xd_train_nb, y_train_nb)

y_pred = clf.predict(Xd_test_nb)
NB_Accuracy = accuracy_score(y_test_nb, y_pred)

print("y_test = ",y_test_nb)
print("y_pred = ",y_pred,"\n")
print("NB_Accuracy = ", NB_Accuracy, "\n")
print("confusion_matrix \n", confusion_matrix(y_test_nb, y_pred))
```

```
y_test =  [1 0 1 0 1]
y_pred =  [1 0 0 0 1]

NB_Accuracy =  0.8

confusion_matrix
 [[2 0]
 [1 2]]
```

Exercise comments

1. Group continuous data in range so that they can be places in classes ie Age=Age_range(Child, Adult)
2. When calculating returns and Moving averages there is NAN which can be handled in many ways such as:
   - Replacing NAN with zeros,
   - Replacing NAN with the mean for the series,
   - Replacing NAN with values with some pseudo values.
   - Drop all the rows which have NaN values and use a slightly smaller dataset
3. Compare the correlation between attributes and class to see which data is most likely the best predictor. This can be done on any graph type and checking if there is a clear separation between classes of a given attribute(works best for oneR)

Correlation between attributes and target(class)

Other approaches include:

- Pearson's Correlation Coefficient: f_regression()
- ANOVA: f_classif()
- Chi-Squared: chi2()
- Mutual Information: mutual_info_classif() and mutual_info_regression()
  All the listed functions are found in the scikit-learn library

Correlation between attributes and target(class)

In [31]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif, f_regression,  chi2
best_features = SelectKBest(score_func=f_classif, k=3)
fit = best_features.fit(Xd_train_nb,y_train_nb)
df_scores = pd.DataFrame(fit.scores_)
df_columns = pd.DataFrame(Xd_train_nb.columns)
# concatenate dataframes
feature_scores = pd.concat([df_columns, df_scores],axis=1)
feature_scores.columns = ['Feature_Name','Score']  # name output columns
print(feature_scores.nlargest(3,'Score'))  # print 3 best features
```

```
  Feature_Name     Score
3     Porridge  0.428571
4       Soccer  0.136364
0   Shortbread  0.026786
```