Gokul-Naive-Practice

November 11, 2024

[1]:
```
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings("ignore")
```

[3]:
```
df_play = pd.read_csv('play.csv')
df_play.head()
```

[3]:

	Unnamed: 0	Outlook	Temperature	Humidity	Wind	Play
0	0	Sunny	Hot	High	Weak	No
1	1	Sunny	Hot	High	Strong	No
2	2	Overcast	Hot	High	Weak	Yes
3	3	Rain	Mild	High	Weak	Yes
4	4	Rain	Cool	Normal	Weak	Yes

[7]:
```
#  Calculate Prior Probabilities

# Obtain total 'Yes' and 'No' values for 'Play'
num_yes = (df_play['Play'] == "Yes").sum()
num_no = (df_play['Play'] == "No").sum()
```

[9]:
```
# Calculate prior probabilities from Yes and No occurrences in the 'Play' Column
P_Yes = num_yes / df_play.shape[0]
P_No = num_no / df_play.shape[0]
```

[11]:
```
# Print the probability
print("P(Yes): ", P_Yes)
print("P(No): ", P_No)
```

```
P(Yes):  0.6428571428571429
P(No):  0.35714285714285715
```

[13]:
```
#  Calculate Conditional Probability or Likelihood for 'Outlook'
P_Sunny_Yes = len(df_play[(df_play['Outlook'] == "Sunny") & (df_play['Play'] ==␣
 ↪"Yes")]) / num_yes
P_Sunny_No = len(df_play[(df_play['Outlook'] == "Sunny") & (df_play['Play'] ==␣
 ↪"No")]) / num_no
```

```
print("P(Sunny|Yes):", P_Sunny_Yes)
print("P(Sunny|No):", P_Sunny_No)
```

```
P(Sunny|Yes): 0.2222222222222222
P(Sunny|No): 0.6
```

[15]:
```
# Calculate Conditional Probability or Likelihood for 'Temperature'
P_Cool_Yes = len(df_play[(df_play['Temperature'] == "Cool") & (df_play['Play']␣
↪== "Yes")]) / num_yes
P_Cool_No = len(df_play[(df_play['Temperature'] == "Cool") & (df_play['Play']␣
↪== "No")]) / num_no
print("P(Cool|Yes):", P_Cool_Yes)
print("P(Cool|No):", P_Cool_No)
```

```
P(Cool|Yes): 0.3333333333333333
P(Cool|No): 0.2
```

[17]:
```
# Calculate Conditional Probability or Likelihood for 'Humidity'
P_High_Yes = len(df_play[(df_play['Humidity'] == "High") & (df_play['Play'] ==␣
↪"Yes")]) / num_yes
P_High_No = len(df_play[(df_play['Humidity'] == "High") & (df_play['Play'] ==␣
↪"No")]) / num_no
print("P(High|Yes):", P_High_Yes)
print("P(High|No):", P_High_No)
```

```
P(High|Yes): 0.3333333333333333
P(High|No): 0.8
```

[19]:
```
# Calculate Conditional Probability or Likelihood for 'Wind'
P_Strong_Yes = len(df_play[(df_play['Wind'] == "Strong") & (df_play['Play'] ==␣
↪"Yes")]) / num_yes
P_Strong_No = len(df_play[(df_play['Wind'] == "Strong") & (df_play['Play'] ==␣
↪"No")]) / num_no
print("P(Strong|Yes):", P_Strong_Yes)
print("P(Strong|No):", P_Strong_No)
```

```
P(Strong|Yes): 0.3333333333333333
P(Strong|No): 0.6
```

[21]:
```
# Calculate final likelihood or conditional probabilities for 'Yes' and 'No'␣
↪values
P_X_c1 = P_Sunny_Yes * P_Cool_Yes * P_High_Yes * P_Strong_Yes
P_X_c2 = P_Sunny_No * P_Cool_No * P_High_No * P_Strong_No
print("P(X|c1):", P_X_c1)
print("P(X|c2):", P_X_c2)
```

```
P(X|c1): 0.008230452674897118
P(X|c2): 0.0576
```

[23]:
```
#  Evaluate Game Play Probabilities Using Bayes' Theorem

# Probability for having a game
P_c1_X = P_X_c1 * P_Yes
print("Probability of having a game : ", P_c1_X)
```

```
Probability of having a game :  0.005291005291005291
```

[25]:
```
# Probability for not having a game
P_c2_X = P_X_c2 * P_No
print("Probability of not having a game : ", P_c2_X)
```

```
Probability of not having a game :  0.02057142857142857
```

[27]:
```
#  Apply the 'info()' function on the 'df_play' DataFrame.
df_play.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Unnamed: 0   14 non-null     int64
 1   Outlook      14 non-null     object
 2   Temperature  14 non-null     object
 3   Humidity     14 non-null     object
 4   Wind         14 non-null     object
 5   Play         14 non-null     object
dtypes: int64(1), object(5)
memory usage: 804.0+ bytes
```

[29]:
```
#  Encode the categorical values

from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()
for column in df_play.columns:
  df_play[column] = label.fit_transform(df_play[column])

df_play
```

[29]:

	Unnamed: 0	Outlook	Temperature	Humidity	Wind	Play
0	0	2	1	0	1	0
1	1	2	1	0	0	0
2	2	0	1	0	1	1
3	3	1	2	0	1	1
4	4	1	0	1	1	1
5	5	1	0	1	0	0
6	6	0	0	1	0	1

7	7	2	2	0	1	0
8	8	2	0	1	1	1
9	9	1	2	1	1	1
10	10	2	2	1	0	1
11	11	0	2	0	0	1
12	12	0	1	1	1	1
13	13	1	2	0	0	0

[31]:
```
#  Create separate DataFrames for feature and target

features_df = df_play.drop('Play', axis = 1)
target_df = df_play['Play']

print(features_df.shape)
print(target_df.shape)
```

```
(14, 5)
(14,)
```

[33]:
```
#  Import train_test_split function
from sklearn.model_selection import train_test_split
```

[35]:
```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(features_df, target_df,␣
 ↪test_size = 0.3,random_state = 2)
```

[37]:
```
# Print the shape of train and test sets.
print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)
print("Shape of y_train:", y_train.shape)
print("Shape of y_test:", y_test.shape)
```

```
Shape of X_train: (9, 5)
Shape of X_test: (5, 5)
Shape of y_train: (9,)
Shape of y_test: (5,)
```

[39]:
```
# Implement Naive Bayes Classifier

# Import the required library
from sklearn.naive_bayes import CategoricalNB

# Model the NB Classifier
nb_clf = CategoricalNB()
nb_clf.fit(X_train, y_train)
```

[39]:
```
CategoricalNB()
```

[41]:
```
# Predict the train and test sets
y_train_predict_nb = nb_clf.predict(X_train)
y_test_predict_nb = nb_clf.predict(X_test)
```

[43]:
```
# Evaluate the accuracy scores
print('Accuracy on the training set: {:.2f}'.format(nb_clf.score(X_train,␣
 ↪y_train)))
print('Accuracy on the test set: {:.2f}'.format(nb_clf.score(X_test, y_test)))
```

```
Accuracy on the training set: 1.00
Accuracy on the test set: 0.60
```

[47]:
```
# Predict the outcome for Day 15
nb_clf.predict([[1, 0, 0, 0,0]])
```

[47]:
```
array([1])
```

[ ]:
```

```