10th April Assignment

April 24, 2023

1 Assignment 65

Q1. A company conducted a survey of its employees and found that 70% of the employees use the company's health insurance plan, while 40% of the employees who use the plan are smokers. What is the probability that an employee is a smoker given that he/she uses the health insurance plan?

Ans.This can be solve using bayes theorem let A denote the event that employee is smoker.

B denotes the event that employees uses health insuarance.

P(A|B) = P(B|A) * P(A) | P(B)

P(B) = 70% = 0.70, P(A|H) = 40% = 0.40

P(A) = p(A|B) * P(B) + P(A|B') * P(B')

= 0.40 * 0.70 + 0.20 * 0.30

= 0.34

P(A|B) = P(B|A) * P(A)/P(B)

= 0.40 * 0.34 / 0.70

= 19.5 are somker uses the health insuarance

Q2. What is the difference between Bernoulli Naive Bayes and Multinomial Naive Bayes?

Ans.The main difference between Bernoulli Naive Bayes and Multinomial Naive Bayes is the type of data they are designed to handle. Bernoulli Naive Bayes is used for binary data (i.e., data that can take on only two values), while Multinomial Naive Bayes is used for count data (i.e., data that represents the frequency of occurrence of multiple discrete events).

Q3. How does Bernoulli Naive Bayes handle missing values?

Ans. The bernoulli naive bayes handle missing values using following ways:

- Drop the missing values
- Impute the missing values

- Treat missing values as separate category

Q4. Can Gaussian Naive Bayes be used for multi-class classification?

Ans.Yes, Gaussian Naive Bayes can be used for multi-class classification. In the case of multi-class classification, Gaussian Naive Bayes estimates the parameters of a Gaussian distribution for each class and uses these estimates to compute the probability of each class given the feature vector.

Q5. Assignment:

- Data preparation:Download the "Spambase Data Set" from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Spambase). This dataset contains email messages, where the goal is to predict whether a message is spam or not based on several input features.
- Implementation:Implement Bernoulli Naive Bayes, Multinomial Naive Bayes, and Gaussian Naive Bayes classifiers using the scikit-learn library in Python. Use 10-fold cross-validation to evaluate the performance of each classifier on the dataset. You should use the default hyperparameters for each classifier.
- Results: Report the following performance metrics for each classifier(Accuracy,Precision,Recal l,F1 score
- Discussion:Discuss the results you obtained. Which variant of Naive Bayes performed the best? Why do you think that is the case? Are there any limitations of Naive Bayes that you observed?
- Conclusion: Summarise your findings and provide some suggestions for future work.

Ans.

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

[12]:
```
attributes=['word_freq_1','word_freq_2','word_freq_3','word_freq_4','word_freq_5','word_freq_6
```

[14]:
```
df=pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/
↪spambase/spambase.data",names=attributes)
```

[18]:
```
df.head()
```

[18]:
```
   word_freq_1  word_freq_2  word_freq_3  word_freq_4  word_freq_5  \
0         0.00         0.64         0.64          0.0         0.32   
1         0.21         0.28         0.50          0.0         0.14   
2         0.06         0.00         0.71          0.0         1.23   
3         0.00         0.00         0.00          0.0         0.63   
4         0.00         0.00         0.00          0.0         0.63   

   word_freq_6  word_freq_7  word_freq_8  word_freq_9  word_freq_10  …  \
0         0.00         0.00         0.00         0.00          0.00  …
```

```
1          0.28         0.21         0.07         0.00           0.94  …
2          0.19         0.19         0.12         0.64           0.25  …
3          0.00         0.31         0.63         0.31           0.63  …
4          0.00         0.31         0.63         0.31           0.63  …

   char_freq_1  char_freq_2  char_freq_3  char_freq_4  char_freq_5  \
0         0.00        0.000          0.0        0.778        0.000
1         0.00        0.132          0.0        0.372        0.180
2         0.01        0.143          0.0        0.276        0.184
3         0.00        0.137          0.0        0.137        0.000
4         0.00        0.135          0.0        0.135        0.000

   char_freq_6  capital_run_length_average  capital_run_length_longest  \
0        0.000                       3.756                          61
1        0.048                       5.114                         101
2        0.010                       9.821                         485
3        0.000                       3.537                          40
4        0.000                       3.537                          40

   capital_run_length_total  output
0                       278       1
1                      1028       1
2                      2259       1
3                       191       1
4                       191       1

[5 rows x 58 columns]
```

[20]:
```
#independent and dependent featuress
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

[21]:
```
##train test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=42,test_size=0.
 ↪3)
```

[22]:
```
from sklearn.naive_bayes import GaussianNB,BernoulliNB,MultinomialNB
```

1.1 Gaussian

[25]:
```
gb=GaussianNB()
```

[26]:
```
gb.fit(X_train,y_train)
```

[26]:
```
GaussianNB()
```

[28]:
```
y_pred_gb=gb.predict(X_test)
```

[29]:
```
from sklearn.metrics import classification_report,accuracy_score
print(accuracy_score(y_test,y_pred_gb))
print(classification_report(y_test,y_pred_gb))
```

```
0.8247646632874729
              precision    recall  f1-score   support

           0       0.95      0.74      0.83       804
           1       0.72      0.95      0.82       577

    accuracy                           0.82      1381
   macro avg       0.84      0.84      0.82      1381
weighted avg       0.86      0.82      0.83      1381
```

1.2 Bernoulli

[30]:
```
bb=BernoulliNB()
```

[31]:
```
bb.fit(X_train,y_train)
```

[31]:
```
BernoulliNB()
```

[32]:
```
y_pred_bb=bb.predict(X_test)
```

[33]:
```
from sklearn.metrics import classification_report,accuracy_score
print(accuracy_score(y_test,y_pred_bb))
print(classification_report(y_test,y_pred_bb))
```

```
0.8790731354091238
              precision    recall  f1-score   support

           0       0.87      0.93      0.90       804
           1       0.89      0.81      0.85       577

    accuracy                           0.88      1381
   macro avg       0.88      0.87      0.87      1381
weighted avg       0.88      0.88      0.88      1381
```

1.3 Mulinomial

[34]:
```
mb=MultinomialNB()
```

[35]:
```
mb.fit(X_train,y_train)
```

[35]: MultinomialNB()

[36]:
```
y_pred_mb=mb.predict(X_test)
```

[37]:
```
from sklearn.metrics import classification_report,accuracy_score
print(accuracy_score(y_test,y_pred_mb))
print(classification_report(y_test,y_pred_mb))
```

```
0.782041998551774
              precision    recall  f1-score   support

           0       0.79      0.84      0.82       804
           1       0.76      0.69      0.73       577

    accuracy                           0.78      1381
   macro avg       0.78      0.77      0.77      1381
weighted avg       0.78      0.78      0.78      1381
```

1.4 Conclusion

Bernoulli Naive bayes is suitable for this problem statement