File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown

This project needs get_dummy and CountVectorizer / TfidfVectorizer before or after tts

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

Load the Amazon Review data

In [2]:
```
file_path = r"D:\LIUZHICHENG\Udemy\Machine Learning\Classification Bootmap\ML Classification Package\6. Decision Trees ar
df = pd.read_csv(filepath_or_buffer=file_path, sep="\t")
df.head()
```

Out[2]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	Love my Echo!	1
1	5	31-Jul-18	Charcoal Fabric	Loved it!	1
2	4	31-Jul-18	Walnut Finish	Sometimes while playing a game, you can answer...	1
3	5	31-Jul-18	Charcoal Fabric	I have had a lot of fun with this thing. My 4 ...	1
4	5	31-Jul-18	Charcoal Fabric	Music	1

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
rating              0
date                0
variation           0
verified_reviews    0
feedback            0
dtype: int64
```

EDA [...]

Data Preprocessing

In [9]:
```
df.drop(columns=["date", "rating"], inplace=True)
# get dummy for "variation"
df = pd.get_dummies(data=df, columns=["variation"], drop_first=True) # drop_first=True -> dummy trap
```

In [10]:
```
# CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer

cv = CountVectorizer()
review_cv = cv.fit_transform(df["verified_reviews"])
```

In [11]:
```
review_cv.shape
```

Out[11]: (3150, 4044)

In [12]:
```
print(cv.get_feature_names())
```

```
kie', 'telephone', 'television', 'tell', 'telling', 'tells', 'temp', 'temperature', 'temps', 'tempting', 'ten', 'tend',
'tends', 'terminology', 'terrible', 'terrific', 'test', 'tested', 'testing', 'texas', 'text', 'texts', 'tg', 'tge', 'tha
n', 'thank', 'thanks', 'that', 'thats', 'the', 'theater', 'theecho', 'their', 'theirs', 'them', 'themes', 'themselves',
'then', 'theories', 'there', 'therefore', 'thermostat', 'these', 'thestand', 'thete', 'they', 'thick', 'thing', 'things',
'think', 'thinking', 'third', 'this', 'thongs', 'thorough', 'thoroughly', 'those', 'thou', 'though', 'thought', 'thought
s', 'thousands', 'three', 'thrilled', 'through', 'throughout', 'throw', 'thrown', 'thru', 'thu', 'thumb', 'thumbs', 'thun
derstorm', 'thursday', 'ti', 'tickled', 'tiempo', 'tiene', 'ties', 'til', 'till', 'time', 'timer', 'timers', 'times', 'ti
ming', 'tin', 'ting', 'tinker', 'tinkering', 'tinny', 'tiny', 'tipping', 'tips', 'tired', 'title', 'tivo', 'to', 'toda',
'today', 'toddler', 'together', 'toilet', 'told', 'tomorrow', 'tomy', 'ton', 'tones', 'tons', 'tony', 'too', 'took', 'too
l', 'tools', 'tooth', 'top', 'topic', 'tosca', 'total', 'totallly', 'totally', 'tou', 'touch', 'touching', 'touted', 'tow
ard', 'towards', 'tower', 'town', 'toy', 'tp', 'track', 'traditional', 'traffic', 'trailer', 'trailers', 'trained', 'trai
nees', 'training', 'transferring', 'travel', 'traveling', 'travelling', 'través', 'treadmill', 'treat', 'treble', 'trek',
'tremendous', 'trending', 'trial', 'tricks', 'tricky', 'tried', 'tries', 'trigger', 'trip', 'trivia', 'trouble', 'trouble
shooting', 'troublesome', 'troubling', 'true', 'truly', 'trust', 'try', 'trying', 'tube', 'tubi', 'tune', 'tunein', 'tune
s', 'turn', 'turned', 'turning', 'turns', 'tv', 'tvs', 'tweeter', 'tweeters', 'twice', 'twist', 'twitter', 'two', 'ty',
'type', 'typed', 'types', 'typical', 'typically', 'typing', 'títulos', 'udefulness', 'ugly', 'uhyour', 'ummm', 'un', 'una
ble', 'unacceptable', 'unavailable', 'unbelievable', 'uncle', 'under', 'underestimated', 'understand', 'understanding',
'understands', 'understood', 'unexpected', 'unfortunately', 'unhappy', 'unhelpful', 'unico', 'unimportant', 'uninstall',
'unique', 'unit', 'units', 'universal', 'unless', 'unlike', 'unlimited', 'unlocking', 'unnannounced', 'unnecessary', 'uno
btrusive', 'unplug', 'unplugged', 'unresponsive', 'unsettling', 'untapped', 'until', 'unusable', 'unused', 'unwitty', 'un
wrapped', 'up', 'upcoming', 'update', 'updated', 'updates', 'updating', 'upgrade', 'upgraded', 'upgrades', 'upgrading',
```

In [13]:
```
type(review_cv)
```

Out[13]: scipy.sparse.csr.csr_matrix

In [14]:
```
review_cv.toarray()
```

Out[14]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [15]:
```
df = df.drop(columns="verified_reviews")
df.head()
```

Out[15]:

variation_Black	variation_Black	variation_Black	variation_Black	variation_Charcoal	variation_Configuration:	variation_Heather	variation_Oak

	feedback	variation_Black Dot	variation_Black Plus	variation_Black Show	variation_Black Spot	variation_Charcoal Fabric	variation_Configuration: Fire TV Stick	variation_Heather Gray Fabric	variation_C Finish
0	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0	0
2	1	0	0	0	0	0	0	0	0
3	1	0	0	0	0	1	0	0	0
4	1	0	0	0	0	1	0	0	0

In [16]:
```
pd.DataFrame(review_cv.toarray())
```

Out[16]:

	0	1	2	3	4	5	6	7	8	9	...	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3145	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3146	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3147	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3148	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3149	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

3150 rows × 4044 columns

In [17]:
```
df = pd.concat(objs=[df, pd.DataFrame(review_cv.toarray())], axis=1)
df
```

Out[17]:

	feedback	variation_Black Dot	variation_Black Plus	variation_Black Show	variation_Black Spot	variation_Charcoal Fabric	variation_Configuration: Fire TV Stick	variation_Heather Gray Fabric	variation_C Fini
0	1	0	0	0	0	1	0	0	
1	1	0	0	0	0	1	0	0	
2	1	0	0	0	0	0	0	0	
3	1	0	0	0	0	1	0	0	
4	1	0	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...
3145	1	1	0	0	0	0	0	0	
3146	1	1	0	0	0	0	0	0	
3147	1	1	0	0	0	0	0	0	
3148	1	0	0	0	0	0	0	0	
3149	1	1	0	0	0	0	0	0	

3150 rows × 4060 columns

In [18]:
```
X = df.drop(columns="feedback").to_numpy()
X
```

Out[18]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [1, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [1, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [19]:
```
y = df["feedback"].to_numpy()
y
```

Out[19]:
```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

Model Training

CountVectorizer before tts

In [20]:
```
# Decision Trees
from sklearn.model_selection import train_test_split
# stratify=y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)

from sklearn.tree import DecisionTreeClassifier
clf_dt = DecisionTreeClassifier()
clf_dt.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = clf_dt.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.59      0.53      0.56        51
           1       0.96      0.97      0.96       579

    accuracy                           0.93       630
   macro avg       0.77      0.75      0.76       630
weighted avg       0.93      0.93      0.93       630

0.9317460317460318
```

```
[[ 27  24]
 [ 19 560]]
```

In [21]:
```
# Random Forest
from sklearn.model_selection import train_test_split
# stratify=y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)

from sklearn.ensemble import RandomForestClassifier
clf_rf = RandomForestClassifier()
clf_rf.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = clf_rf.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      0.25      0.41        51
           1       0.94      1.00      0.97       579

    accuracy                           0.94       630
   macro avg       0.97      0.63      0.69       630
weighted avg       0.94      0.94      0.92       630

0.9396825396825397
[[ 13  38]
 [  0 579]]
```

Model Training

CountVectorizer after tts

In [22]:
```
# read csv
df = pd.read_csv(filepath_or_buffer=file_path, sep="\t")
# drop columns, "rating", "date"
df.drop(columns=["rating", "date"], inplace=True)
```

In [23]:
```
# get_dummies for "variation"
df = pd.get_dummies(data=df, columns=["variation"], drop_first=True)
```

In [24]:
```
X = df.drop(columns="feedback")
y = df["feedback"]
```

In [25]:
```
from sklearn.model_selection import train_test_split
# stratify=y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)
```

In [26]:
```
X_train.shape
```

Out[26]: (2520, 16)

In [27]:
```
X_test.shape
```

Out[27]: (630, 16)

In [28]:
```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer()
X_train_review_cv = cv.fit_transform(raw_documents=X_train["verified_reviews"])
X_test_review_cv = cv.transform(raw_documents=X_test["verified_reviews"])
```

In [29]:
```
X_train_review_cv
```

Out[29]:
```
<2520x3696 sparse matrix of type '<class 'numpy.int64'>'
        with 48733 stored elements in Compressed Sparse Row format>
```

In [30]:
```
X_train_review_cv.shape
```

Out[30]: (2520, 3696)

In [31]:
```
X_test_review_cv.shape
```

Out[31]: (630, 3696)

In [32]:
```
X_train_review_cv.toarray()
```

Out[32]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [33]:
```
X_test_review_cv.toarray()
```

Out[33]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [34]:
```
X_train = X_train.drop(columns="verified_reviews").reset_index() # reset_index() for X_train
# concatenate X_train and "review" sparse matrix
X_train = pd.concat(objs=[X_train, pd.DataFrame(X_train_review_cv.toarray())], axis=1)
X_train
```

Out[34]:

index	variation_Black Dot	variation_Black Plus	variation_Black Show	variation_Black Spot	variation_Charcoal Fabric	variation_Configuration: Fire TV Stick	variation_Heather Gray Fabric	variation_Oak Finish

		Dot	Plus	Show	Spot	Fabric	Fire TV Stick	Gray Fabric	Finish
0	1655	0	0	0	0	0	0	0	0
1	177	0	0	0	0	0	0	0	1
2	2381	0	0	0	0	0	1	0	0
3	273	0	0	0	0	0	0	1	0
4	1344	0	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...
2515	2295	0	0	0	0	0	1	0	0
2516	1276	0	0	0	0	0	0	0	0
2517	3113	1	0	0	0	0	0	0	0
2518	367	0	0	0	0	0	0	0	0
2519	3007	1	0	0	0	0	0	0	0

2520 rows × 3712 columns

In [35]:
```
X_test = X_test.drop(columns="verified_reviews").reset_index() # reset_index() for X_test
# concatenate X_test and "review" sparse matrix
X_test = pd.concat(objs=[X_test, pd.DataFrame(X_test_review_cv.toarray())], axis=1)
X_test
```

Out[35]:

	index	variation_Black Dot	variation_Black Plus	variation_Black Show	variation_Black Spot	variation_Charcoal Fabric	variation_Configuration: Fire TV Stick	variation_Heather Gray Fabric	variation_Oak Finish
0	876	0	0	0	0	1	0	0	0
1	1172	0	0	0	0	0	0	0	0
2	149	0	0	0	0	1	0	0	0
3	232	0	0	0	0	0	0	0	0
4	1562	0	0	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
625	2509	1	0	0	0	0	0	0	0
626	898	0	0	0	0	1	0	0	0
627	2737	1	0	0	0	0	0	0	0
628	2210	0	0	0	0	0	1	0	0
629	1957	0	1	0	0	0	0	0	0

630 rows × 3712 columns

In [36]:
```
# Decision Trees
from sklearn.tree import DecisionTreeClassifier

clf_dt = DecisionTreeClassifier()
clf_dt.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = clf_dt.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.66      0.53      0.59        51
           1       0.96      0.98      0.97       579

    accuracy                           0.94       630
   macro avg       0.81      0.75      0.78       630
weighted avg       0.93      0.94      0.94       630

0.9396825396825397
[[ 27  24]
 [ 14 565]]
```

In [37]:
```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

clf_rf = RandomForestClassifier()
clf_rf.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = clf_rf.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.93      0.25      0.40        51
           1       0.94      1.00      0.97       579

    accuracy                           0.94       630
   macro avg       0.93      0.63      0.68       630
weighted avg       0.94      0.94      0.92       630

0.9380952380952381
[[ 13  38]
 [  1 578]]
```

Model Training

TfidfVectorizer before tts

In [38]:
```
# read csv
df = pd.read_csv(filepath_or_buffer=file_path, sep="\t")
```

```
# drop columns, "rating", "date"
df.drop(columns=["rating", "date"], inplace=True)

# get_dummies for "variation"
df = pd.get_dummies(data=df, columns=["variation"], drop_first=True)

from sklearn.feature_extraction.text import TfidfVectorizer

tv = TfidfVectorizer()
review_tv = tv.fit_transform(raw_documents=df["verified_reviews"])

df = pd.concat(objs=[df, pd.DataFrame(review_tv.toarray())], axis=1)
df.drop(columns="verified_reviews", inplace=True)

X = df.drop(columns="feedback")
y = df["feedback"]

from sklearn.model_selection import train_test_split
# stratify=y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)
```

In [39]:

```
# Decision Trees
from sklearn.tree import DecisionTreeClassifier

clf_dt = DecisionTreeClassifier()
clf_dt.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = clf_dt.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.55      0.59      0.57        51
           1       0.96      0.96      0.96       579

    accuracy                           0.93       630
   macro avg       0.75      0.77      0.76       630
weighted avg       0.93      0.93      0.93       630

0.926984126984127
[[ 30  21]
 [ 25 554]]
```

In [40]:

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

clf_rf = RandomForestClassifier()
clf_rf.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = clf_rf.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      0.25      0.41        51
           1       0.94      1.00      0.97       579

    accuracy                           0.94       630
   macro avg       0.97      0.63      0.69       630
weighted avg       0.94      0.94      0.92       630

0.9396825396825397
[[ 13  38]
 [  0 579]]
```

Model Training

TfidfVectorizer after tts

In [41]:

```
# read csv
df = pd.read_csv(filepath_or_buffer=file_path, sep="\t")
# drop columns, "rating", "date"
df.drop(columns=["rating", "date"], inplace=True)

# get_dummies for "variation"
df = pd.get_dummies(data=df, columns=["variation"], drop_first=True)

X = df.drop(columns="feedback")
y = df["feedback"]

from sklearn.model_selection import train_test_split
# stratify=y
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)

from sklearn.feature_extraction.text import TfidfVectorizer

tv = TfidfVectorizer()
X_train_review_tv = tv.fit_transform(raw_documents=X_train["verified_reviews"])
X_test_review_tv = tv.transform(raw_documents=X_test["verified_reviews"])

X_train = X_train.drop(columns="verified_reviews").reset_index() # reset_index() for X_train
# concatenate X_train and "review" sparse matrix
X_train = pd.concat(objs=[X_train, pd.DataFrame(X_train_review_cv.toarray())], axis=1)

X_test = X_test.drop(columns="verified_reviews").reset_index() # reset_index() for X_test
# concatenate X_test and "review" sparse matrix
X_test = pd.concat(objs=[X_test, pd.DataFrame(X_test_review_cv.toarray())], axis=1)
```

In [42]:

```
# Decision Trees
from sklearn.tree import DecisionTreeClassifier

clf_dt = DecisionTreeClassifier()
```

```
clf_dt.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = clf_dt.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.62      0.47      0.53        51
           1       0.95      0.97      0.96       579

    accuracy                           0.93       630
   macro avg       0.78      0.72      0.75       630
weighted avg       0.93      0.93      0.93       630

0.9333333333333333
[[ 24  27]
 [ 15 564]]
```

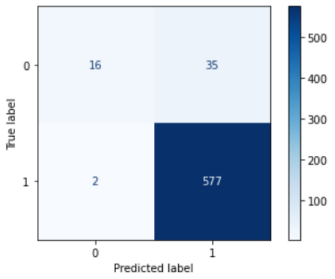
In [43]:

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier

clf_rf = RandomForestClassifier()
clf_rf.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

predictions = clf_rf.predict(X_test)
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.89      0.31      0.46        51
           1       0.94      1.00      0.97       579

    accuracy                           0.94       630
   macro avg       0.92      0.66      0.72       630
weighted avg       0.94      0.94      0.93       630

0.9412698412698413
[[ 16  35]
 [  2 577]]
```

In [44]:

```
from sklearn.metrics import plot_confusion_matrix

display = plot_confusion_matrix(estimator=clf_rf, X=X_test, y_true=y_test, cmap="Blues", values_format=".3g");
```



In [45]:

```
display.confusion_matrix
```

Out[45]:

```
array([[ 16,  35],
       [  2, 577]], dtype=int64)
```