5dwxy3iea

September 23, 2023

[ ]:
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading
hmmlearn-0.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (160
kB)
                           160.4/160.4
kB 3.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-
packages (from hmmlearn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0,>=0.16->hmmlearn) (3.2.0)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.0
```

[ ]:
```
!gdown 1HuaFjmhYjp23EAc3_x8tmhgheLRiLCpV
```

```
Downloading…
From: https://drive.google.com/uc?id=1HuaFjmhYjp23EAc3_x8tmhgheLRiLCpV
To: /content/wdbc.data
100% 124k/124k [00:00<00:00, 102MB/s]
```

[ ]:
```
#MultinomialHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report,confusion_matrix,␣
 ↪accuracy_score
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import math
# WDBC DATASET
# 3.3.1 MultinomialHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wdbc.data",header=None )
df.columns =␣
 ↪['1','Class','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19',
X = df.drop(['1','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.MultinomialHMM(n_components=4)
#random_state=15,n_iter=10,algorithm='viterbi',params='ste')
row = len(X_train)
col = len(X_train[0])
new = [1] * 30
for i in range(row):
  for j in range(col):
    X_train[i][j] = X_train[i][j]*10
    X_train[i][j] = math.floor(X_train[i][j])
  x = X_train[i].astype(np.int32)
new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_train = y
row = len(X_test)
col = len(X_test[0])
new
for i in range(row):
  for j in range(col):
    X_test[i][j] = X_test[i][j]*10
    X_test[i][j] = math.floor(X_test[i][j])
  x = X_test[i].astype(np.int32)
new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_test = y
classifier.fit(X_train)
y_pred = classifier.predict(X_test%101)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
```

```
  if y_pred[i] == 1:
    strings[i] = ("M")
  else :
    strings[i] = ("B")
strings
strings = strings[0:171]
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-----------------------------------------------------")
print("-----------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-----------------------------------------------------")
print("-----------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
WARNING:hmmlearn.hmm:MultinomialHMM has undergone major changes. The previous
version was implementing a CategoricalHMM (a special case of MultinomialHMM).
This new implementation follows the standard definition for a Multinomial
distribution (e.g. as in
https://en.wikipedia.org/wiki/Multinomial_distribution). See these issues for
details:
https://github.com/hmmlearn/hmmlearn/issues/335
https://github.com/hmmlearn/hmmlearn/issues/340
WARNING:hmmlearn.base:Fitting a model with 131 free scalar parameters with only
60 data points will result in a degenerate solution.
```

```
Confusion Matrix:
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-62c5b7a535a0> in <cell line: 60>()
     58 strings = strings[0:171]
     59 print("Confusion Matrix:")
---> 60 print(confusion_matrix(y_test, strings))
     61 print("-----------------------------------------------------")
     62 print("-----------------------------------------------------")

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py in␣
 ↪confusion_matrix(y_true, y_pred, labels, sample_weight, normalize)
    315     (0, 2, 1, 1)
    316     """
--> 317     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    318     if y_type not in ("binary", "multiclass"):
```

```
    319         raise ValueError("%s is not supported" % y_type)

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py in␣
 ↪_check_targets(y_true, y_pred)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in␣
 ↪check_consistent_length(*arrays)
    395     uniques = np.unique(lengths)
    396     if len(uniques) > 1:
--> 397         raise ValueError(
    398             "Found input variables with inconsistent numbers of samples␣
 ↪%r"
    399             % [int(l) for l in lengths]

ValueError: Found input variables with inconsistent numbers of samples: [171, 3
```

[ ]:
```
#MultinomialHMM(with tuning).py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report,confusion_matrix,␣
 ↪accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import math
# WDBC DATASET
# 3.3.6 MultinomialHMM(With Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wdbc.data",header=None )
df.columns =␣
 ↪['1','Class','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19',
X = df.drop(['1','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪3,test_size=0.7,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.MultinomialHMM(n_components=4,init_params="e")
```

```
classifier.startprob_ = np.array([0.6, 0.2, 0.1, 0.1])
classifier.transmat_ = np.array([[0.6, 0.2, 0.1, 0.1],[0.3, 0.4, 0.2, 0.1],[0.
↪3, 0.3, 0.2, 0.2],[0.1, 0.5, 0.3, 0.1]])
classifier.emissionprob_ = np.array([[0.3,0.2,0.3,0.2],[0.1, 0.4, 0.4, 0.1],[0.
↪6, 0.2, 0.1, 0.1],[0.5, 0.2, 0.2, 0.1]])
row = len(X_train)
col = len(X_train[0])
new = [1] * 30
for i in range(row):
  for j in range(col):
    X_train[i][j] = X_train[i][j]*10
    X_train[i][j] = math.floor(X_train[i][j])
  x = X_train[i].astype(np.int32)
new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_train = y
row = len(X_test)
col = len(X_test[0])
new
for i in range(row):
  for j in range(col):
    X_test[i][j] = X_test[i][j]*10
    X_test[i][j] = math.floor(X_test[i][j])
  x = X_test[i].astype(np.int32)
new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_test = y
classifier.fit(X_train)
y_pred = classifier.predict(X_test%101)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("M")
  else :
    strings[i] = ("B")
strings
strings = strings[0:399]
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-------------------------------------------------")
print("-------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-------------------------------------------------")
```

```
print("-------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
WARNING:hmmlearn.hmm:MultinomialHMM has undergone major changes. The previous
version was implementing a CategoricalHMM (a special case of MultinomialHMM).
This new implementation follows the standard definition for a Multinomial
distribution (e.g. as in
https://en.wikipedia.org/wiki/Multinomial_distribution). See these issues for
details:
https://github.com/hmmlearn/hmmlearn/issues/335
https://github.com/hmmlearn/hmmlearn/issues/340
WARNING:hmmlearn.base:Fitting a model with 131 free scalar parameters with only
60 data points will result in a degenerate solution.

Confusion Matrix:
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-4-bfd147a304bb> in <cell line: 62>()
     60 strings = strings[0:399]
     61 print("Confusion Matrix:")
---> 62 print(confusion_matrix(y_test, strings))
     63 print("-------------------------------------------------")
     64 print("-------------------------------------------------")

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py in␣
 ↪confusion_matrix(y_true, y_pred, labels, sample_weight, normalize)
    315     (0, 2, 1, 1)
    316     """
--> 317     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    318     if y_type not in ("binary", "multiclass"):
    319         raise ValueError("%s is not supported" % y_type)

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py in␣
 ↪_check_targets(y_true, y_pred)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in␣
 ↪check_consistent_length(*arrays)
    395     uniques = np.unique(lengths)
```

```
    396     if len(uniques) > 1:
--> 397         raise ValueError(
    398             "Found input variables with inconsistent numbers of samples
 ↪%r"
    399             % [int(l) for l in lengths]

ValueError: Found input variables with inconsistent numbers of samples: [399, 3
```

uifdaeaxh

September 23, 2023

[ ]:
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading
hmmlearn-0.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (160
kB)
                           160.4/160.4
kB 2.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-
packages (from hmmlearn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0,>=0.16->hmmlearn) (3.2.0)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.0
```

[ ]:
```
!gdown 1HuaFjmhYjp23EAc3_x8tmhgheLRiLCpV
```

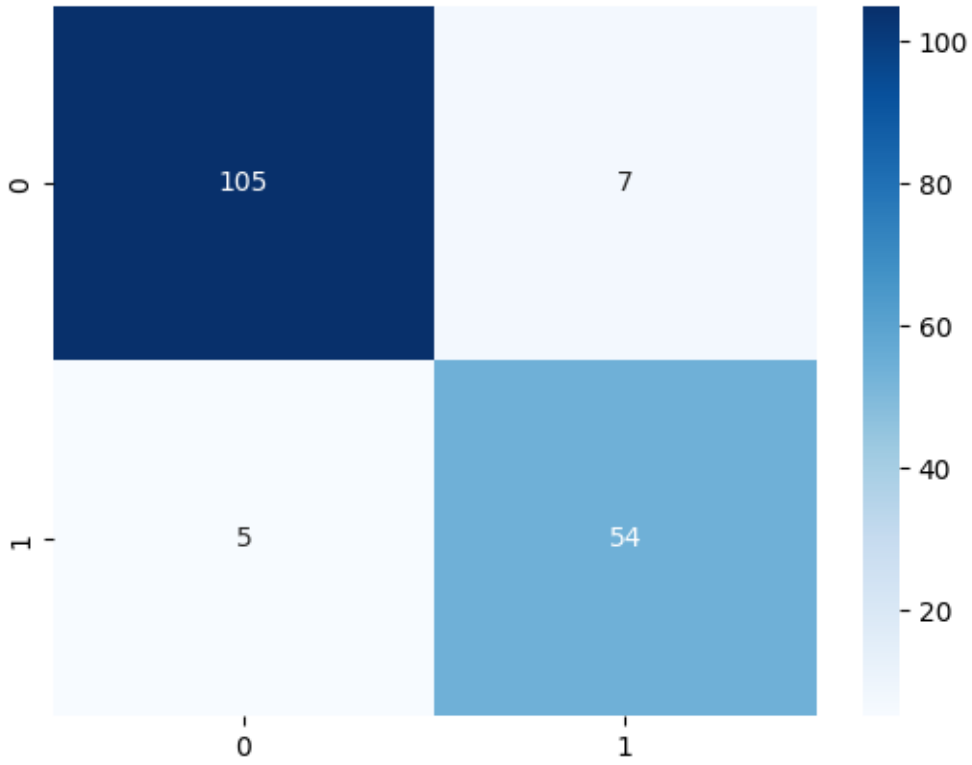
```
Downloading…
From: https://drive.google.com/uc?id=1HuaFjmhYjp23EAc3_x8tmhgheLRiLCpV
To: /content/wdbc.data
100% 124k/124k [00:00<00:00, 68.8MB/s]
```

[ ]:
```
#GMMHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report,confusion_matrix,␣
 ↪accuracy_score
```

```
import matplotlib.pyplot as plt
import seaborn as sns
# WDBC DATASET
# 3.2.1 GMMHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wdbc.data",header=None )
df.columns =␣
 ↪['1','Class','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19',
X = df.drop(['1','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GMMHMM(n_components=2)
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("M")
  else :
    strings[i] = ("B")
strings
#strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-------------------------------------------------")
print("-------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-------------------------------------------------")
print("-------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
Confusion Matrix:
[[105   7]
 [  5  54]]
-------------------------------------------------
-------------------------------------------------
Performance Evaluation
```

```
              precision    recall  f1-score   support

           B       0.95      0.94      0.95       112
           M       0.89      0.92      0.90        59

    accuracy                           0.93       171
   macro avg       0.92      0.93      0.92       171
weighted avg       0.93      0.93      0.93       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9298245614035088
```

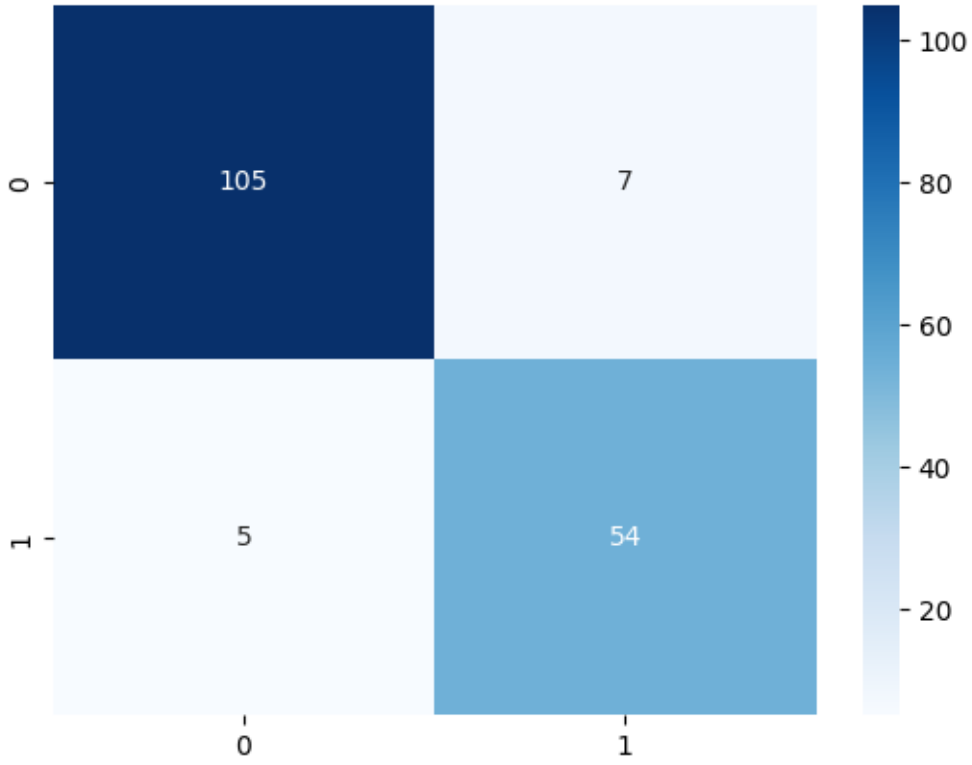


[ ]:
```
#GMMHMM(with tuning).py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
```

```
import matplotlib.pyplot as plt
import seaborn as sns
# WDBC DATASET
# 3.2.1 GMMHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wdbc.data",header=None )
df.columns =␣
 ↪['1','Class','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19',
X = df.drop(['1','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GMMHMM(n_components=2, init_params="m")
classifier.startprob_ = np.array([0.6, 0.4])
classifier.transmat_ = np.array([[0.7, 0.3],
[0.5, 0.5]])
classifier.covars_ = np.tile(np.identity(1),(2, 1, 30))
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("M")
  else :
    strings[i] = ("B")
strings
#strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("------------------------------------------------------")
print("------------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("------------------------------------------------------")
print("------------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
Confusion Matrix:
[[105   7]
```

```
 [  5  54]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.95      0.94      0.95       112
           M       0.89      0.92      0.90        59

    accuracy                           0.93       171
   macro avg       0.92      0.93      0.92       171
weighted avg       0.93      0.93      0.93       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9298245614035088
```



cl2wg1ene

September 23, 2023

[ ]:
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading
hmmlearn-0.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (160
kB)
                           160.4/160.4
kB 3.3 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-
packages (from hmmlearn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0,>=0.16->hmmlearn) (3.2.0)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.0
```

[ ]:
```
!gdown 1HuaFjmhYjp23EAc3_x8tmhgheLRiLCpV
```

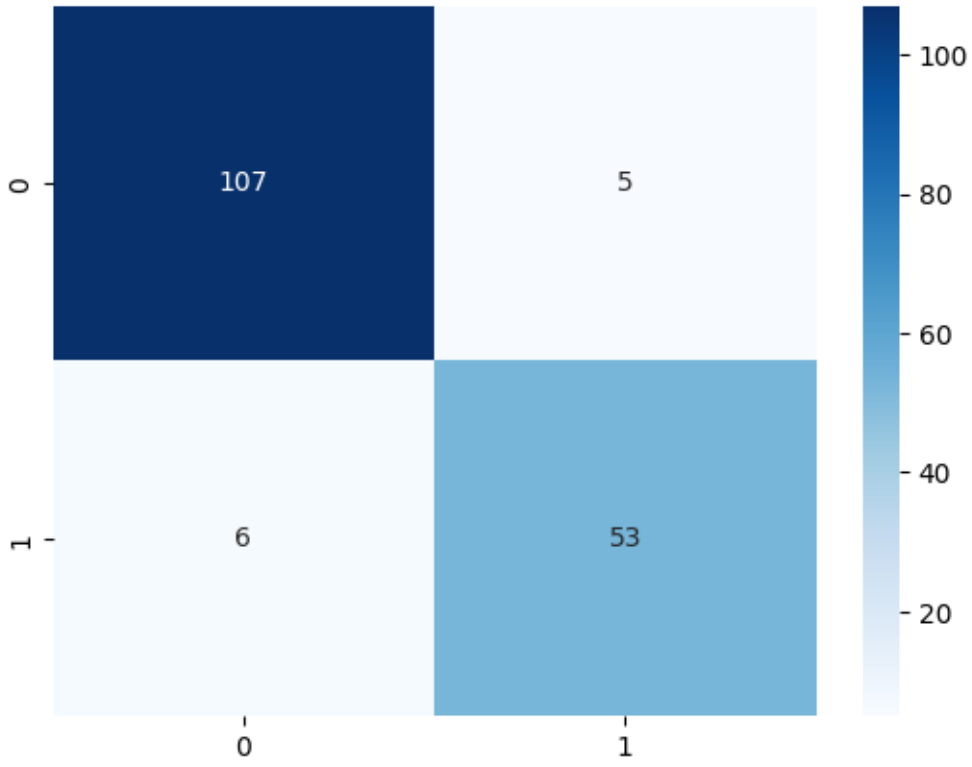
```
Downloading…
From: https://drive.google.com/uc?id=1HuaFjmhYjp23EAc3_x8tmhgheLRiLCpV
To: /content/wdbc.data
100% 124k/124k [00:00<00:00, 24.4MB/s]
```

[ ]:
```
#GaussianHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
```

```
import matplotlib.pyplot as plt
import seaborn as sns
# WDBC DATASET
# 3.1.1 GaussianHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wdbc.data",header=None )
df.columns =␣
 ↪['1','Class','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19',
X = df.drop(['1','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GaussianHMM(n_components=3,
covariance_type="full")
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("M")
  else :
    strings[i] = ("B")
#strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-------------------------------------------------")
print("-------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-------------------------------------------------")
print("-------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
Confusion Matrix:
[[107   5]
 [  6  53]]
-------------------------------------------------
-------------------------------------------------
Performance Evaluation
```

```
              precision    recall  f1-score   support

           B       0.95      0.96      0.95       112
           M       0.91      0.90      0.91        59

    accuracy                           0.94       171
   macro avg       0.93      0.93      0.93       171
weighted avg       0.94      0.94      0.94       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.935672514619883
```



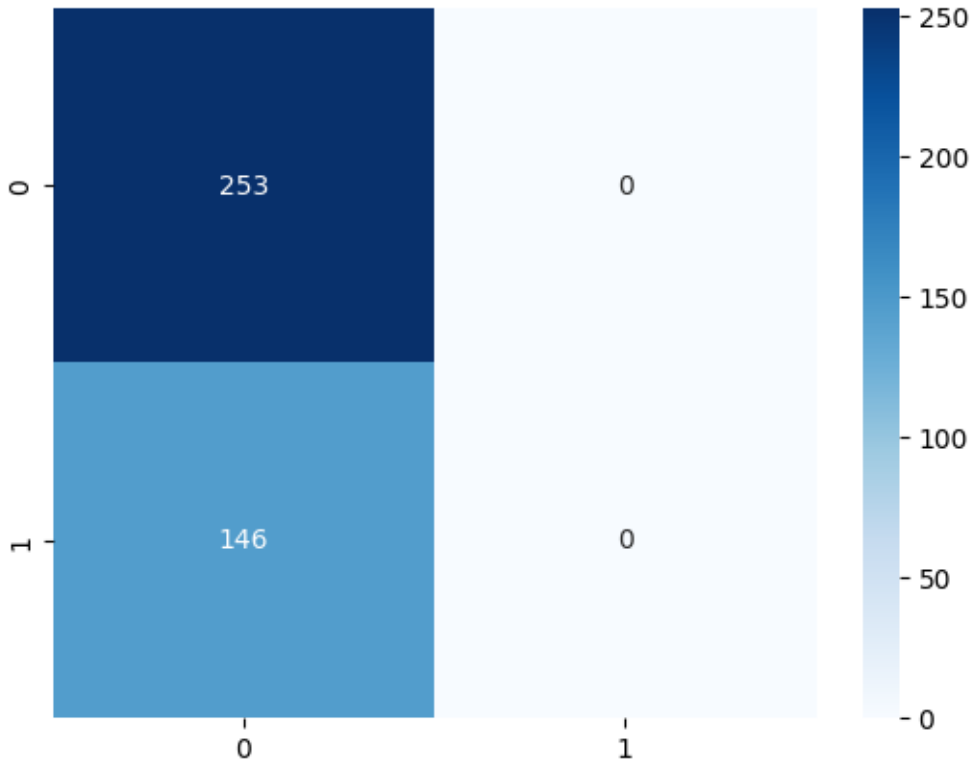
[ ]:
```
#GaussianHMM(with tuning).py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
```

```
import matplotlib.pyplot as plt
import seaborn as sns
# WDBC DATASET
# 3.1.6 GaussianHMM(With Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wdbc.data",header=None )
df.columns =␣
 ↪['1','Class','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19',
X = df.drop(['1','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪3,test_size=0.7,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GaussianHMM(n_components=3,
covariance_type="full", init_params="m")
classifier.startprob_ = np.array([0.6, 0.3, 0.1])
classifier.transmat_ = np.array([[0.7, 0.2, 0.1], [0.3, 0.5, 0.2],[0.3, 0.3, 0.
 ↪4]])
classifier.covars_ = np.tile(np.identity(30), (3, 1, 1))
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("M")
  else :
    strings[i] = ("B")
#strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-------------------------------------------------------")
print("-------------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-------------------------------------------------------")
print("-------------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
Confusion Matrix:
[[253   0]
```

```
 [146   0]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.63      1.00      0.78       253
           M       0.00      0.00      0.00       146

    accuracy                           0.63       399
   macro avg       0.32      0.50      0.39       399
weighted avg       0.40      0.63      0.49       399

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.6340852130325815
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```


253
0
146
0
0
1
0
1
250
200
150
100
50
0

x2x5vq4j7

September 23, 2023

[ ]:
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading
hmmlearn-0.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (160
kB)
                           160.4/160.4
kB 5.7 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-
packages (from hmmlearn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0,>=0.16->hmmlearn) (3.2.0)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.0
```

[ ]:
```
!gdown 1EwHiw0cfjxYynvCT2QfpICZWy2p5NhxH
```

```
Downloading…
From: https://drive.google.com/uc?id=1EwHiw0cfjxYynvCT2QfpICZWy2p5NhxH
To: /content/ionosphere.data
100% 76.5k/76.5k [00:00<00:00, 80.3MB/s]
```

[ ]:
```
#MultinomialHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import math
# IONOSPHERE DATASET
# 2.3.1 MultinomialHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("ionosphere.data",header=None )
df.columns =␣
 ↪['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20'
 ↪'29','30','31','32','33','34','Class']
X = df.drop(['1','2','Class'], axis=1)
y = df['Class']
X = df.drop(['1','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.MultinomialHMM(n_components=3)
row = len(X_train)
col = len(X_train[0])
new = [1] * 33
for i in range(row):
  for j in range(col):
    X_train[i][j] = X_train[i][j]*10
    X_train[i][j] = math.floor(X_train[i][j])
  x = X_train[i].astype(np.int32)
new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_train = y
row = len(X_test)
col = len(X_test[0])
new #= [1] * 67
for i in range(row):
  for j in range(col):
    X_test[i][j] = X_test[i][j]*10
    X_test[i][j] = math.floor(X_test[i][j])
  x = X_test[i].astype(np.int32)
new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_test = y
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
```

```
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("g")
  else :
    strings[i] = ("b")
#strings = strings.astype(np.int32)
strings = strings[0:106]
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-----------------------------------------------------")
print("-----------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-----------------------------------------------------")
print("-----------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
WARNING:hmmlearn.hmm:MultinomialHMM has undergone major changes. The previous
version was implementing a CategoricalHMM (a special case of MultinomialHMM).
This new implementation follows the standard definition for a Multinomial
distribution (e.g. as in
https://en.wikipedia.org/wiki/Multinomial_distribution). See these issues for
details:
https://github.com/hmmlearn/hmmlearn/issues/335
https://github.com/hmmlearn/hmmlearn/issues/340
WARNING:hmmlearn.base:Fitting a model with 104 free scalar parameters with only
66 data points will result in a degenerate solution.
```

```
Confusion Matrix:
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-6-1afcc06ef021> in <cell line: 61>()
     59 strings = strings[0:106]
     60 print("Confusion Matrix:")
---> 61 print(confusion_matrix(y_test, strings))
     62 print("-----------------------------------------------------")
     63 print("-----------------------------------------------------")

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py in␣
 ↪confusion_matrix(y_true, y_pred, labels, sample_weight, normalize)
    315     (0, 2, 1, 1)
    316     """
```

```
--> 317     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    318     if y_type not in ("binary", "multiclass"):
    319         raise ValueError("%s is not supported" % y_type)

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py in␣
 ↪_check_targets(y_true, y_pred)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in␣
 ↪check_consistent_length(*arrays)
    395     uniques = np.unique(lengths)
    396     if len(uniques) > 1:
--> 397         raise ValueError(
    398             "Found input variables with inconsistent numbers of samples␣
 ↪%r"
    399             % [int(l) for l in lengths]

ValueError: Found input variables with inconsistent numbers of samples: [106, 3
```

[ ]:
```
#MultinomialHMM(with tuning).py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import math
# IONOSPHERE DATASET
# 2.3.6 MultinomialHMM(With Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("ionosphere.data",header=None )
df.columns =␣
 ↪['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20'
X = df.drop(['1','2','Class'], axis=1)
y = df['Class']
X = df.drop(['1','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.MultinomialHMM(n_components=3,init_params="e")
classifier.startprob_ = np.array([0.6, 0.3, 0.1])
classifier.transmat_ = np.array([[0.7, 0.2, 0.1], [0.3, 0.5, 0.2], [0.3, 0.3, 0.
 ↪4]])
classifier.emissionprob_ = np.array([[0.3,0.2,0.5], [0.1, 0.4, 0.5],[0.6, 0.3,␣
 ↪0.1]])
row = len(X_train)
col = len(X_train[0])
new = [1] * 33
for i in range(row):
  for j in range(col):
    X_train[i][j] = X_train[i][j]*10
    X_train[i][j] = math.floor(X_train[i][j])
  x = X_train[i].astype(np.int32)
new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_train = y
row = len(X_test)
col = len(X_test[0])
new #= [1] * 67
for i in range(row):
  for j in range(col):
    X_test[i][j] = X_test[i][j]*10
    X_test[i][j] = math.floor(X_test[i][j])
  x = X_test[i].astype(np.int32)
new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_test = y
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("g")
  else :
    strings[i] = ("b")
#strings = strings.astype(np.int32)
strings = strings[0:106]
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("------------------------------------------------------")
```

```
print("-------------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-------------------------------------------------------")
print("-------------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
WARNING:hmmlearn.hmm:MultinomialHMM has undergone major changes. The previous
version was implementing a CategoricalHMM (a special case of MultinomialHMM).
This new implementation follows the standard definition for a Multinomial
distribution (e.g. as in
https://en.wikipedia.org/wiki/Multinomial_distribution). See these issues for
details:
https://github.com/hmmlearn/hmmlearn/issues/335
https://github.com/hmmlearn/hmmlearn/issues/340
WARNING:hmmlearn.base:Fitting a model with 104 free scalar parameters with only
66 data points will result in a degenerate solution.
```

```
Confusion Matrix:
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-7-a5d0c5265778> in <cell line: 64>()
     62 strings = strings[0:106]
     63 print("Confusion Matrix:")
---> 64 print(confusion_matrix(y_test, strings))
     65 print("-------------------------------------------------------")
     66 print("-------------------------------------------------------")

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py in␣
 ↪confusion_matrix(y_true, y_pred, labels, sample_weight, normalize)
    315     (0, 2, 1, 1)
    316     """
--> 317     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    318     if y_type not in ("binary", "multiclass"):
    319         raise ValueError("%s is not supported" % y_type)

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py in␣
 ↪_check_targets(y_true, y_pred)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in␣
 ↪check_consistent_length(*arrays)
    395     uniques = np.unique(lengths)
    396     if len(uniques) > 1:
--> 397         raise ValueError(
    398             "Found input variables with inconsistent numbers of samples␣
 ↪%r"
    399             % [int(l) for l in lengths]

ValueError: Found input variables with inconsistent numbers of samples: [106, 3
```

oxsmaybgm

September 23, 2023

[ ]:
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading
hmmlearn-0.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (160
kB)
                           160.4/160.4
kB 3.0 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-
packages (from hmmlearn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0,>=0.16->hmmlearn) (3.2.0)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.0
```

[ ]:
```
!gdown 1EwHiw0cfjxYynvCT2QfpICZWy2p5NhxH
```

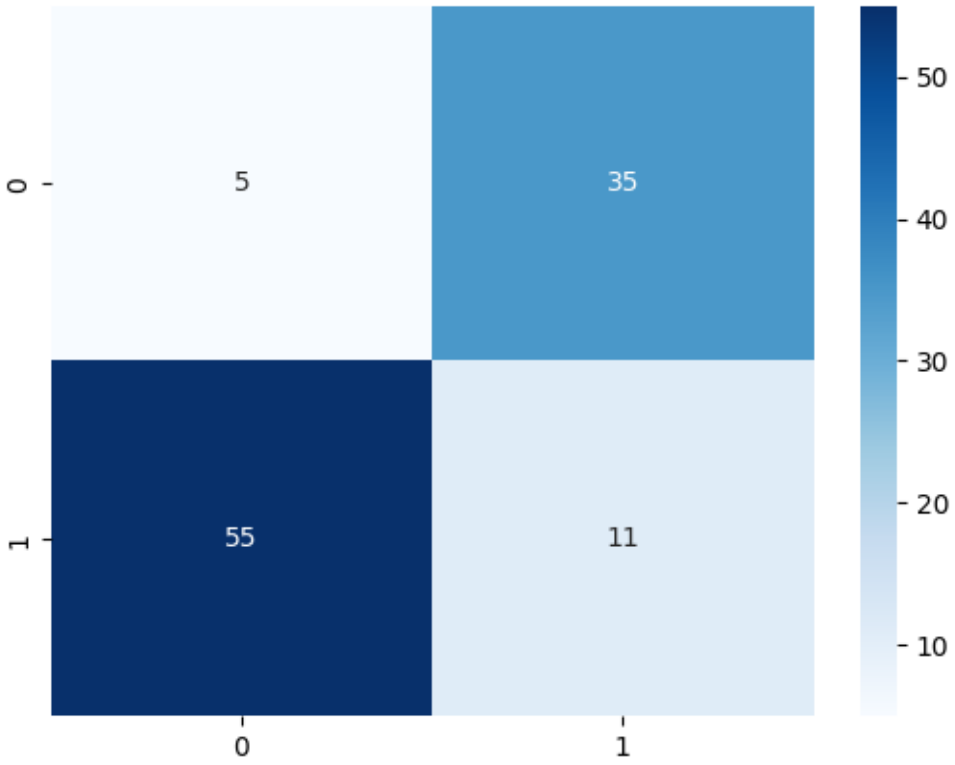
```
Downloading…
From: https://drive.google.com/uc?id=1EwHiw0cfjxYynvCT2QfpICZWy2p5NhxH
To: /content/ionosphere.data
100% 76.5k/76.5k [00:00<00:00, 87.7MB/s]
```

[ ]:
```
#GMMHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
```

```
import matplotlib.pyplot as plt
import seaborn as sns
# IONOSPHERE DATASET
# 2.2.1 GMMHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("ionosphere.data",header=None )
df.columns = ['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15'␣
 ↪,'16','17','18','19','20','21','22','23','24','25','26','27','28',␣
 ↪'29','30','31','32','33','34','Class']
X = df.drop(['1','2','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GMMHMM(n_components=3, covariance_type="full")
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("g")
  else :
    strings[i] = ("b")
#strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-------------------------------------------------")
print("-------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-------------------------------------------------")
print("-------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
Confusion Matrix:
[[ 5 35]
 [55 11]]
-------------------------------------------------
-------------------------------------------------
Performance Evaluation
```

```
              precision    recall  f1-score   support

           b       0.08      0.12      0.10        40
           g       0.24      0.17      0.20        66

    accuracy                           0.15       106
   macro avg       0.16      0.15      0.15       106
weighted avg       0.18      0.15      0.16       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.1509433962264151
```



[ ]:
```
#GMMHMM(with tuning).py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
```

```
import matplotlib.pyplot as plt
import seaborn as sns
# IONOSPHERE DATASET
# 2.2.6 GMMHMM(With Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("ionosphere.data",header=None )
df.columns =␣
 ↪['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20'
X = df.drop(['1','2','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GMMHMM(n_components=3, covariance_type="full", init_params="m")
classifier.startprob_ = np.array([0.6, 0.3, 0.1])
classifier.transmat_ = np.array([[0.7, 0.2, 0.1], [0.3, 0.5, 0.2], [0.3, 0.3, 0.
 ↪4]])
classifier.covars_ = np.tile(np.identity(32), (3, 1, 1))
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("g")
  else :
    strings[i] = ("b")
#strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-------------------------------------------------------")
print("-------------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-------------------------------------------------------")
print("-------------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-5-6052ac533ead> in <cell line: 25>()
```

```
     23 classifier.transmat_ = np.array([[0.7, 0.2, 0.1], [0.3, 0.5, 0.2], [0.3
 ↪0.3, 0.4]])
     24 classifier.covars_ = np.tile(np.identity(32), (3, 1, 1))
---> 25 classifier.fit(X_train)
     26 y_pred = classifier.predict(X_test)
     27 size = len(y_pred)

/usr/local/lib/python3.10/dist-packages/hmmlearn/base.py in fit(self, X, length
    467
    468         self._init(X, lengths)
--> 469         self._check()
    470         self.monitor_._reset()
    471

/usr/local/lib/python3.10/dist-packages/hmmlearn/hmm.py in _check(self)
    674         needed_shape = needed_shapes[self.covariance_type]
    675         if covars_shape != needed_shape:
--> 676             raise ValueError(
    677                 f"{self.covariance_type!r} mixture covars must have
 ↪shape "
    678                 f"{needed_shape}, actual shape: {covars_shape}")

ValueError: 'full' mixture covars must have shape (3, 1, 32, 32), actual shape:
 ↪(3, 32, 32)
```

httjcueu6

September 23, 2023

[ ]:
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading
hmmlearn-0.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (160
kB)
                           160.4/160.4
kB 3.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-
packages (from hmmlearn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0,>=0.16->hmmlearn) (3.2.0)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.0
```

[ ]:
```
!gdown 1EwHiw0cfjxYynvCT2QfpICZWy2p5NhxH
```

```
Downloading…
From: https://drive.google.com/uc?id=1EwHiw0cfjxYynvCT2QfpICZWy2p5NhxH
To: /content/ionosphere.data
100% 76.5k/76.5k [00:00<00:00, 3.25MB/s]
```

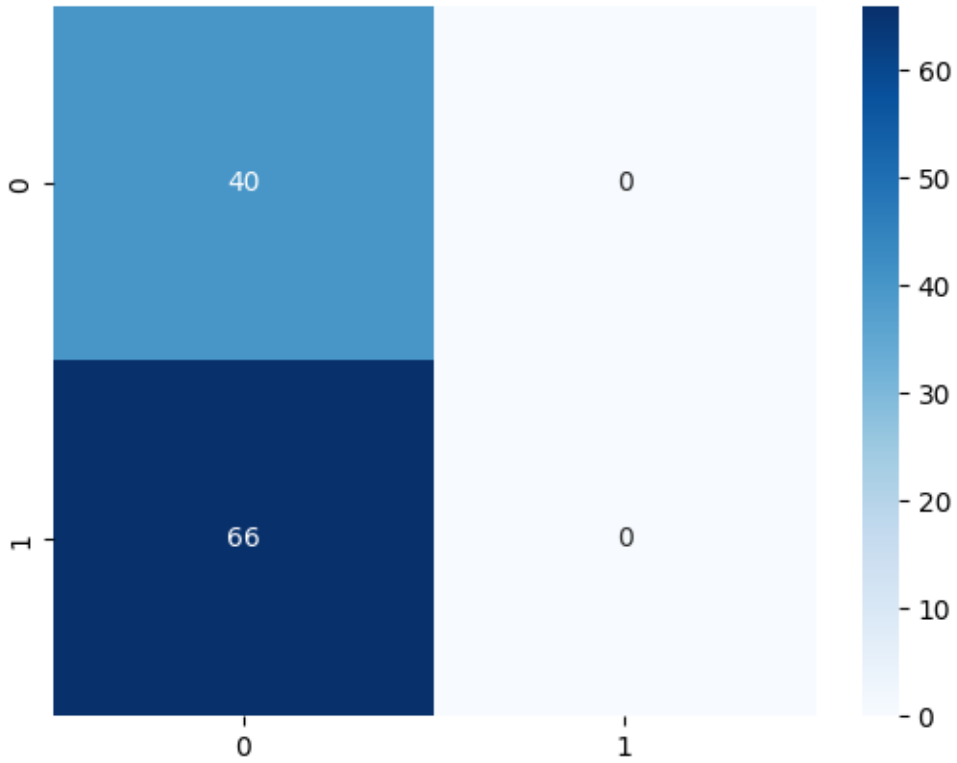
[ ]:
```
#GaussianHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn import metrics
```

```
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
# IONOSPHERE DATASET
# 2.1.1 GaussianHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("ionosphere.data",header=None )
df.columns =␣
 ↪['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20'
 ↪30','31','32','33','34','Class']
X = df.drop(['1','2','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GaussianHMM(n_components=3, covariance_type="full")
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("g")
  else :
    strings[i] = ("b")
#strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-----------------------------------------------------")
print("-----------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-----------------------------------------------------")
print("-----------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
Confusion Matrix:
[[40  0]
 [66  0]]
-----------------------------------------------------
```

```
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.38      1.00      0.55        40
           g       0.00      0.00      0.00        66

    accuracy                           0.38       106
   macro avg       0.19      0.50      0.27       106
weighted avg       0.14      0.38      0.21       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.37735849056603776
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

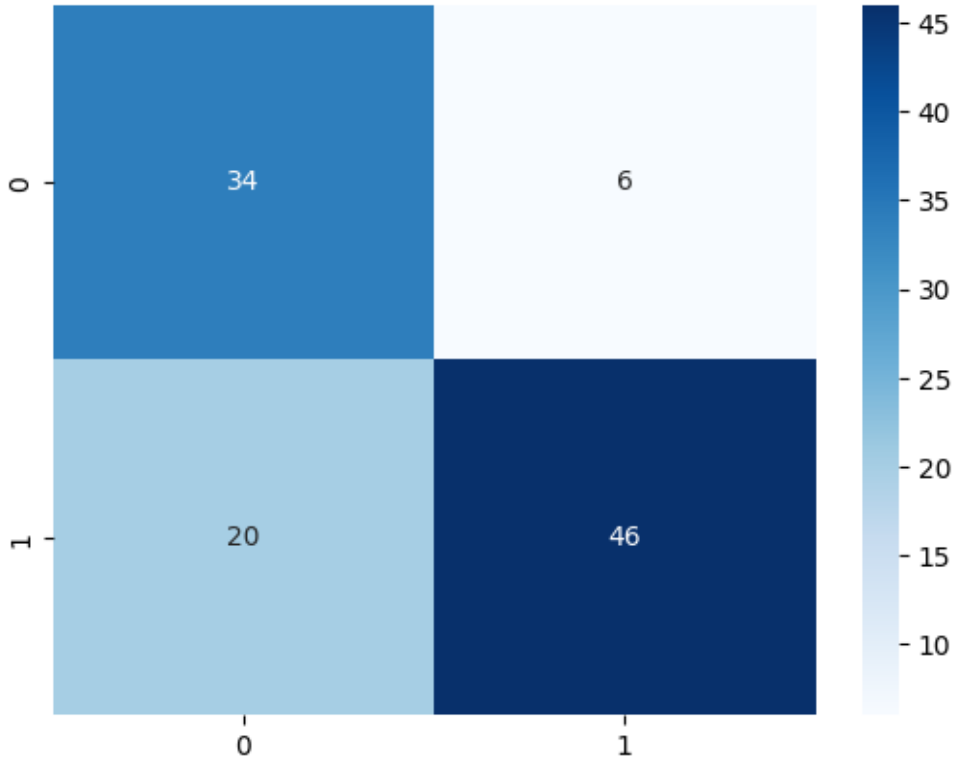


[ ]:
```
#GaussianHMM(with tuning).py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
# IONOSPHERE DATASET
# 2.1.6 GaussianHMM(With Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("ionosphere.data",header=None )
df.columns =␣
 ↪['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20'
X = df.drop(['1','2','Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
```

```
X_test = sc.transform(X_test)
classifier = hmm.GaussianHMM(n_components=3,
covariance_type="full", init_params="m")
classifier.startprob_ = np.array([0.6, 0.3, 0.1])
classifier.transmat_ = np.array([[0.7, 0.2, 0.1],
[0.3, 0.5, 0.2],
[0.3, 0.3, 0.4]])
classifier.covars_ = np.tile(np.identity(32), (3, 1, 1))
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 1:
    strings[i] = ("g")
  else :
    strings[i] = ("b")
#strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("-------------------------------------------------")
print("-------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-------------------------------------------------")
print("-------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
Confusion Matrix:
[[34  6]
 [20 46]]
-------------------------------------------------
-------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.63      0.85      0.72        40
           g       0.88      0.70      0.78        66

    accuracy                           0.75       106
   macro avg       0.76      0.77      0.75       106
weighted avg       0.79      0.75      0.76       106
```

```
--------------------------------------------------
--------------------------------------------------
Accuracy:
0.7547169811320755
```



cmybzvmzl

September 23, 2023

[ ]:
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading
hmmlearn-0.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (160
kB)
                           160.4/160.4
kB 3.7 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-
packages (from hmmlearn) (1.11.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0,>=0.16->hmmlearn) (3.2.0)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.0
```

[ ]:
```
!gdown 1dQlEhfWb15RwU7IxBbqtdzqur0UHFqMG
```

```
Downloading…
From: https://drive.google.com/uc?id=1dQlEhfWb15RwU7IxBbqtdzqur0UHFqMG
To: /content/wine.data
100% 10.8k/10.8k [00:00<00:00, 24.6MB/s]
```

[ ]:
```
#MultinomialHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn import metrics
```

```
from sklearn.metrics import classification_report,confusion_matrix,␣
 ↪accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import math
# WINE DATASET
# 1.3.1 MultinomialHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wine.data",header=None )
df.columns = ['Class','Alcohol','Malic acid','Ash','Alcalinity of␣
 ↪ash','Magnesium','Total phenols','Flavanoids', 'Nonflavanoid␣
 ↪phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted␣
 ↪wines','Proline']
X = df.drop(['Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.MultinomialHMM(n_components=3)
row = len(X_train)
col = len(X_train[0])
new = [1] * 13
for i in range(row):
  for j in range(col):
    X_train[i][j] = X_train[i][j]*10
    X_train[i][j] = math.floor(X_train[i][j])
  x = X_train[i].astype(np.int32)
  new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_train = y
row = len(X_test)
col = len(X_test[0])
#new #= [1] * 13
for i in range(row):
  for j in range(col):
    X_test[i][j] = X_test[i][j]*10
    X_test[i][j] = math.floor(X_test[i][j])
  x = X_test[i].astype(np.int32)
  new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_test = y
classifier.fit(X_train)
y_pred = classifier.predict(X_test,row).reshape(row,col)
```

```
size = len(y_pred)
#print(y_pred.shape)
y_pred_proba = classifier.predict_proba(X_test,row).reshape(row,col*3)[0:row,0:
  ↪3]
for i in range(row):
  s=0
for j in range(2):
  s+=y_pred_proba[i][j]
y_pred_proba[i][2]=1.0-s
#print(y_pred_proba)
#print(y_test)
fpr, tpr, _ = metrics.roc_curve(y_test, classifier.predict_proba(X_test,row).
  ↪reshape(row,col*3)[:,1],pos_label=1)
auc = metrics.roc_auc_score(y_test, y_pred_proba, multi_class='ovr')
#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i].any() == 0:
    strings[i] = 1
  elif y_pred[i].any() == 1:
    strings[i] = 2
else :
    strings[i] = 3
strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("------------------------------------------------------")
print("------------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("------------------------------------------------------")
print("------------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
WARNING:hmmlearn.hmm:MultinomialHMM has undergone major changes. The previous
version was implementing a CategoricalHMM (a special case of MultinomialHMM).
This new implementation follows the standard definition for a Multinomial
distribution (e.g. as in
```

https://en.wikipedia.org/wiki/Multinomial_distribution). See these issues for details:
https://github.com/hmmlearn/hmmlearn/issues/335
https://github.com/hmmlearn/hmmlearn/issues/340

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-3-a574cbe6cf34> in <cell line: 49>()
     47 X_test = y
     48 classifier.fit(X_train)
---> 49 y_pred = classifier.predict(X_test,row).reshape(row,col)
     50 size = len(y_pred)
     51 #print(y_pred.shape)

/usr/local/lib/python3.10/dist-packages/hmmlearn/base.py in predict(self, X,␣
 ↪lengths)
    361             Labels for each sample from ``X``.
    362         """
--> 363         _, state_sequence = self.decode(X, lengths)
    364         return state_sequence
    365

/usr/local/lib/python3.10/dist-packages/hmmlearn/base.py in decode(self, X,␣
 ↪lengths, algorithm)
    336         log_prob = 0
    337         sub_state_sequences = []
--> 338         for sub_X in _utils.split_X_lengths(X, lengths):
    339             # XXX decoder works on a single sample at a time!
    340             sub_log_prob, sub_state_sequence = decoder(sub_X)

/usr/local/lib/python3.10/dist-packages/hmmlearn/_utils.py in split_X_lengths(X␣
 ↪lengths)
     22         n_samples = len(X)
     23         if cs[-1] != n_samples:
---> 24             raise ValueError(
     25                 f"lengths array {lengths} doesn't sum to {n_samples}␣
 ↪samples")
     26         return np.split(X, cs)[:-1]

ValueError: lengths array 54 doesn't sum to 179 samples
```

[ ]:
```
#MultinomialHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
from hmmlearn import hmm
from sklearn import metrics
from sklearn.metrics import classification_report,confusion_matrix,␣
 ↪accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import math
```

[ ]:
```
# WINE DATASET
# 1.3.1 MultinomialHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wine.data",header=None )
df.columns = ['Class','Alcohol','Malic acid','Ash','Alcalinity of␣
 ↪ash','Magnesium','Total phenols','Flavanoids', 'Nonflavanoid␣
 ↪phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted␣
 ↪wines','Proline']
X = df.drop(['Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.MultinomialHMM(n_components=3)
```

```
WARNING:hmmlearn.hmm:MultinomialHMM has undergone major changes. The previous
version was implementing a CategoricalHMM (a special case of MultinomialHMM).
This new implementation follows the standard definition for a Multinomial
distribution (e.g. as in
https://en.wikipedia.org/wiki/Multinomial_distribution). See these issues for
details:
https://github.com/hmmlearn/hmmlearn/issues/335
https://github.com/hmmlearn/hmmlearn/issues/340
```

[ ]:
```
row = len(X_train)
col = len(X_train[0])
new = [1] * 13
for i in range(row):
  for j in range(col):
    X_train[i][j] = X_train[i][j]*10
    X_train[i][j] = math.floor(X_train[i][j])
  x = X_train[i].astype(np.int32)
  new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_train = y
row = len(X_test)
```

```
col = len(X_test[0])
```

[ ]:
```
for i in range(row):
  for j in range(col):
    X_test[i][j] = X_test[i][j]*10
    X_test[i][j] = math.floor(X_test[i][j])
  x = X_test[i].astype(np.int32)
  new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_test = y
classifier.fit(X_train)
```

```
WARNING:hmmlearn.base:Even though the 'startprob_' attribute is set, it will be
overwritten during initialization because 'init_params' contains 's'
WARNING:hmmlearn.base:Even though the 'transmat_' attribute is set, it will be
overwritten during initialization because 'init_params' contains 't'
```

[ ]:
```
MultinomialHMM(n_components=3,
               n_trials=array([ 13,  97, 106,  91,  98, 124,  79, 125, 118, 147,
163, 103, 107,
       122,  87, 101, 113,  77,  72, 107,  78, 118, 100, 103, 126, 111,
       163, 104,  93, 154,  75, 101, 116,  58, 117, 136,  91,  85, 118,
       114,  90,  95, 130, 101, 131,  86, 114,  88,  84,  94,  85, 104,
       117, 113,  84,  66, 122,  97, 142, 142, 127,  92,  71, 116,  92,
        89, 131, 110, 125,  90, 120, 103, 117,  90,  79, 162, 135, 134,
       144, 129,  94, 104, 134, 106, 140,  73,  79,  83, 102,  72, 103,
       109, 108,  98, 101,  81,  95, 145, 101, 138,  93, 118, 100, 106,
        87,  79,  97,  90,  98, 101, 142, 122, 100, 128, 164, 106, 127,
       134, 120, 123,  90, 109, 109,  94, 109]),
               random_state=RandomState(MT19937) at 0x7B0CC0B87640)
```

[ ]:
```
y_pred = classifier.predict(X_test,row).reshape(row,col)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-9aa472262cdf> in <cell line: 1>()
----> 1 y_pred = classifier.predict(X_test,row).reshape(row,col)

/usr/local/lib/python3.10/dist-packages/hmmlearn/base.py in predict(self, X,␣
 ↪lengths)
    361             Labels for each sample from ``X``.
    362         """
--> 363         _, state_sequence = self.decode(X, lengths)
    364         return state_sequence
    365
```

```
/usr/local/lib/python3.10/dist-packages/hmmlearn/base.py in decode(self, X,␣
 ↪lengths, algorithm)
    336         log_prob = 0
    337         sub_state_sequences = []
--> 338         for sub_X in _utils.split_X_lengths(X, lengths):
    339             # XXX decoder works on a single sample at a time!
    340             sub_log_prob, sub_state_sequence = decoder(sub_X)

/usr/local/lib/python3.10/dist-packages/hmmlearn/_utils.py in split_X_lengths(X ␣
 ↪lengths)
     22         n_samples = len(X)
     23         if cs[-1] != n_samples:
---> 24             raise ValueError(
     25                 f"lengths array {lengths} doesn't sum to {n_samples}␣
 ↪samples")
     26         return np.split(X, cs)[:-1]

ValueError: lengths array 54 doesn't sum to 233 samples
```

[ ]:
```
print(y_pred.shape)
y_pred_proba = classifier.predict_proba(X_test,row).reshape(row,col*3)[0:row,0:
 ↪3]
for i in range(row):
  s=0
for j in range(2):
  s+=y_pred_proba[i][j]
y_pred_proba[i][2]=1.0-s
#print(y_pred_proba)
#print(y_test)
fpr, tpr, _ = metrics.roc_curve(y_test, classifier.predict_proba(X_test,row).
 ↪reshape(row,col*3)[:,1],pos_label=1)
auc = metrics.roc_auc_score(y_test, y_pred_proba, multi_class='ovr')
#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i].any() == 0:
    strings[i] = 1
  elif y_pred[i].any() == 1:
    strings[i] = 2
else :
    strings[i] = 3
```

```
strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("------------------------------------------------------")
print("------------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("------------------------------------------------------")
print("------------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

[ ]:

[ ]:
```
#MultinomialHMM(with tuning).py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
import math
# WINE DATASET
# 1.3.6 MultinomialHMM(With Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wine.data",header=None )
df.columns = ['Class','Alcohol','Malic acid','Ash','Alcalinity of␣
 ↪ash','Magnesium','Total phenols','Flavanoids', 'Nonflavanoid␣
 ↪phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted␣
 ↪wines','Proline']
X = df.drop(['Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.MultinomialHMM(n_components=3,init_params="e")
classifier.startprob_ = np.array([0.6, 0.3, 0.1])
classifier.transmat_ = np.array([[0.7, 0.2, 0.1],
```

```
[0.3, 0.5, 0.2],
[0.3, 0.3, 0.4]])
classifier.emissionprob_ = np.array([[0.3,0.2,0.5],
[0.1, 0.4, 0.5],
[0.6, 0.3, 0.1]
])
row = len(X_train)
col = len(X_train[0])
new = [1] * 13
for i in range(row):
  for j in range(col):
    X_train[i][j] = X_train[i][j]*10
    X_train[i][j] = math.floor(X_train[i][j])
  x = X_train[i].astype(np.int32)
  new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_train = y
"""
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
"""
#print(len(X_test))
#print(str(len(X_test[0]))+' '+str(len(X_test[1])))
row = len(X_test)
col = len(X_test[0])
#new #= [1] * 13
for i in range(row):
  for j in range(col):
    X_test[i][j] = X_test[i][j]*10
    X_test[i][j] = math.floor(X_test[i][j])
  x = X_test[i].astype(np.int32)
  new = np.vstack([new,x])
y = new
y = np.absolute(y)
X_test = y
classifier.fit(X_train)
y_pred = classifier.predict(X_test,row).reshape(row,col)
size = len(y_pred)
#print(y_pred.shape)
y_pred_proba = classifier.predict_proba(X_test,row).reshape(row,col*3)[0:row,0:
  ↪3]
for i in range(row):
  s=0
for j in range(2):
```

```
  s+=y_pred_proba[i][j]
y_pred_proba[i][2]=1.0-s
#print(y_pred_proba)
#print(y_test)
fpr, tpr, _ = metrics.roc_curve(y_test, classifier.predict_proba(X_test,row).
 ↪reshape(row,col*3)[:,1],pos_label=1)
auc = metrics.roc_auc_score(y_test, y_pred_proba, multi_class='ovr')
#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i].any() == 0:
    strings[i] = 1
  elif y_pred[i].any() == 1:
    strings[i] = 2
  else :
    strings[i] = 3
strings = strings.astype(np.int32)
#y_test.reshape(row,col)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("------------------------------------------------------")
print("------------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("------------------------------------------------------")
print("------------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```

```
WARNING:hmmlearn.hmm:MultinomialHMM has undergone major changes. The previous
version was implementing a CategoricalHMM (a special case of MultinomialHMM).
This new implementation follows the standard definition for a Multinomial
distribution (e.g. as in
https://en.wikipedia.org/wiki/Multinomial_distribution). See these issues for
details:
https://github.com/hmmlearn/hmmlearn/issues/335
https://github.com/hmmlearn/hmmlearn/issues/340
```

```
--------------------------------------------------------------------------
```

```
ValueError                                Traceback (most recent call last)
<ipython-input-13-d102b9c9ac24> in <cell line: 65>()
     63 X_test = y
     64 classifier.fit(X_train)
---> 65 y_pred = classifier.predict(X_test,row).reshape(row,col)
     66 size = len(y_pred)
     67 #print(y_pred.shape)

/usr/local/lib/python3.10/dist-packages/hmmlearn/base.py in predict(self, X,␣
 ↪lengths)
    361             Labels for each sample from ``X``.
    362         """
--> 363         _, state_sequence = self.decode(X, lengths)
    364         return state_sequence
    365

/usr/local/lib/python3.10/dist-packages/hmmlearn/base.py in decode(self, X,␣
 ↪lengths, algorithm)
    336         log_prob = 0
    337         sub_state_sequences = []
--> 338         for sub_X in _utils.split_X_lengths(X, lengths):
    339             # XXX decoder works on a single sample at a time!
    340             sub_log_prob, sub_state_sequence = decoder(sub_X)

/usr/local/lib/python3.10/dist-packages/hmmlearn/_utils.py in split_X_lengths(X␣
 ↪lengths)
     22         n_samples = len(X)
     23         if cs[-1] != n_samples:
---> 24             raise ValueError(
     25                 f"lengths array {lengths} doesn't sum to {n_samples}␣
 ↪samples")
     26         return np.split(X, cs)[:-1]

ValueError: lengths array 54 doesn't sum to 179 samples
```

rx9ib9izb

September 23, 2023

[ ]:
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading
hmmlearn-0.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (160
kB)
                           160.4/160.4
kB 3.2 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-
packages (from hmmlearn) (1.11.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0,>=0.16->hmmlearn) (3.2.0)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.0
```

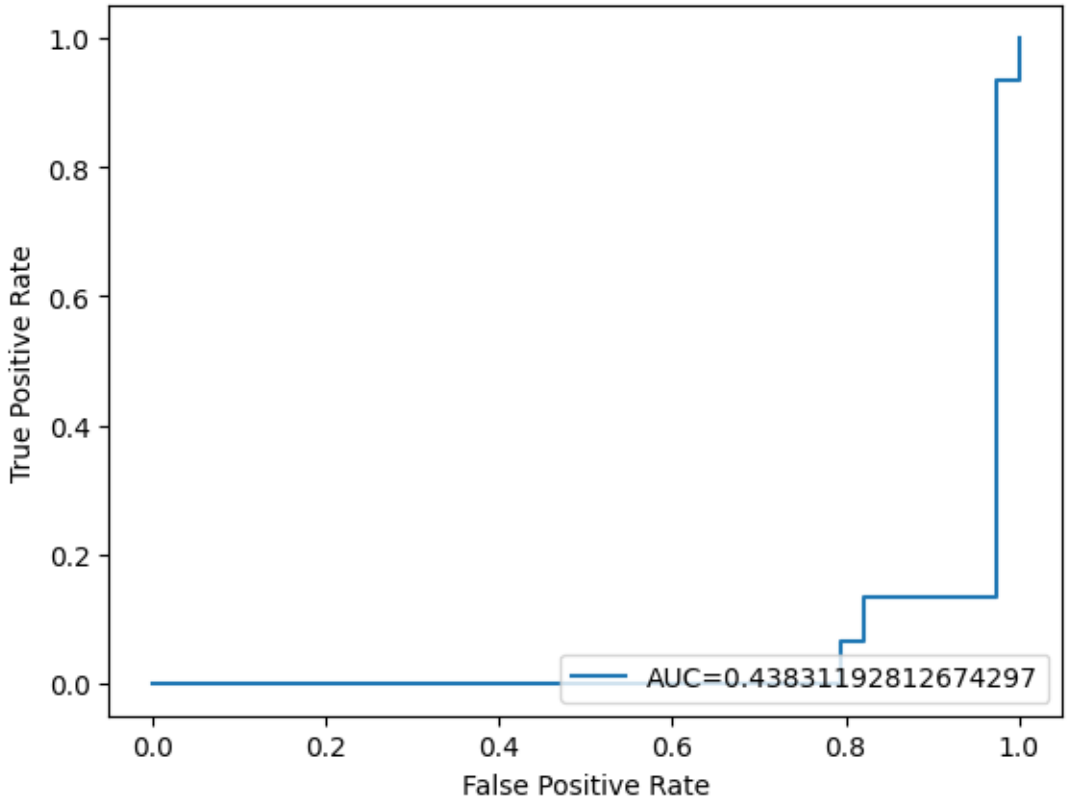
[ ]:
```
!gdown 1dQlEhfWb15RwU7IxBbqtdzqur0UHFqMG
```

```
Downloading…
From: https://drive.google.com/uc?id=1dQlEhfWb15RwU7IxBbqtdzqur0UHFqMG
To: /content/wine.data
100% 10.8k/10.8k [00:00<00:00, 24.8MB/s]
```

[ ]:
```
#GMMHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn import metrics
```

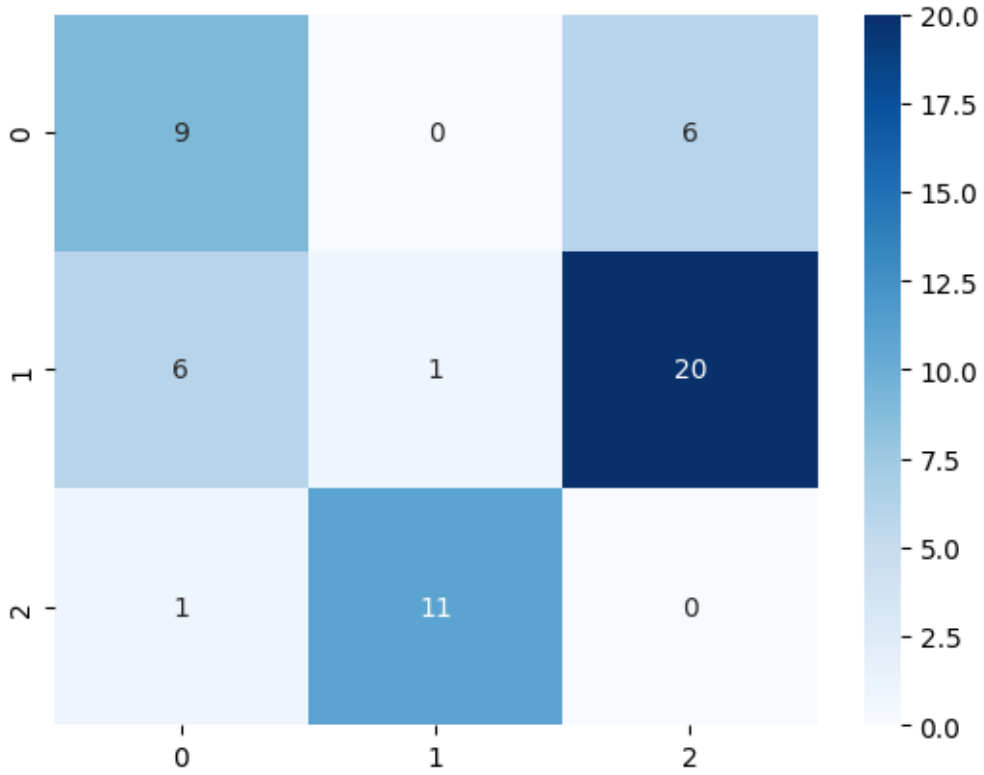
```
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
# WINE DATASET
# 1.2.1 GMMMHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wine.data",header=None )
df.columns = ['Class','Alcohol','Malic acid','Ash','Alcalinity of␣
 ↪ash','Magnesium','Total phenols','Flavanoids','Nonflavanoid␣
 ↪phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted␣
 ↪wines','Proline']
X = df.drop(['Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GMMHMM(n_components=3, covariance_type="full")
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
y_pred_proba = classifier.predict_proba(X_test)
fpr, tpr, _ = metrics.roc_curve(y_test,
classifier.predict_proba(X_test)[:,1],pos_label=1)
auc = metrics.roc_auc_score(y_test, y_pred_proba,
multi_class='ovr')
#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 0:
    strings[i] = 1
  elif y_pred[i] == 1:
    strings[i] = 2
  else :
    strings[i] = 3
strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("------------------------------------------------------")
print("------------------------------------------------------")
```

```
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-----------------------------------------------------")
print("-----------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```




```
Confusion Matrix:
[[ 9  0  6]
 [ 6  1 20]
 [ 1 11  0]]
-----------------------------------------------------
-----------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.56      0.60      0.58        15
           2       0.08      0.04      0.05        27
```

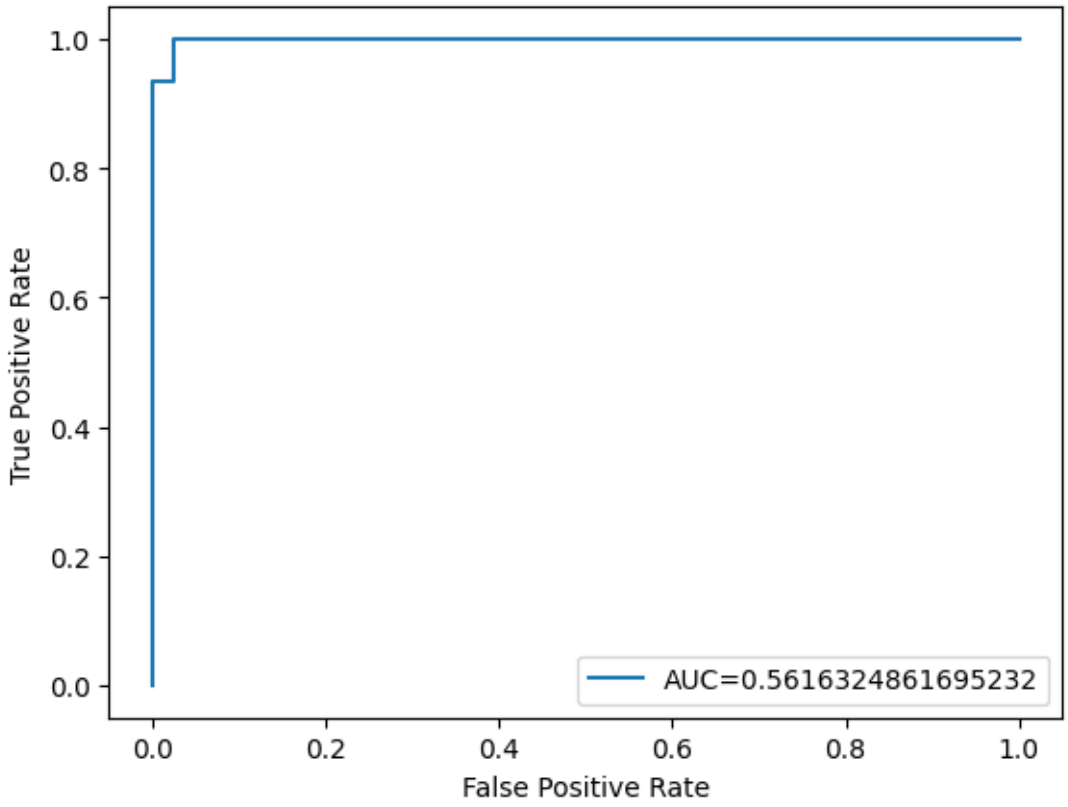
```
           3       0.00      0.00      0.00        12

    accuracy                           0.19        54
   macro avg       0.22      0.21      0.21        54
weighted avg       0.20      0.19      0.19        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.18518518518518517
```



[ ]:
```python
#GMMHMM(with tuning).py
import warnings
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
import matplotlib.pyplot as plt
```

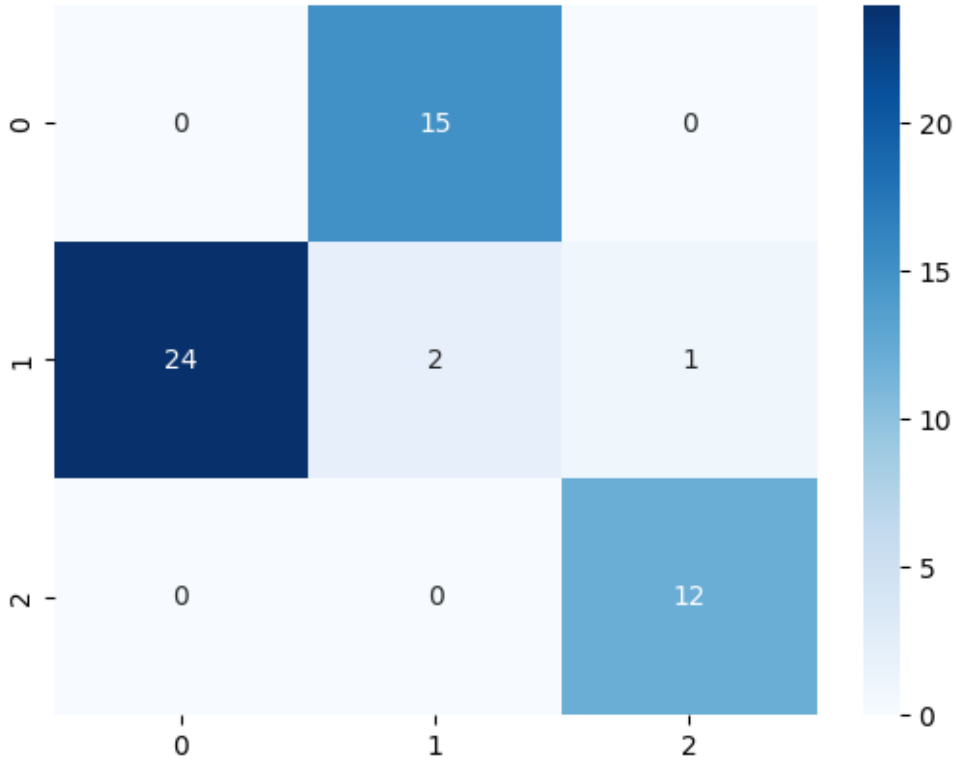
```
import seaborn as sns
warnings.filterwarnings('ignore') # "error", "ignore", "always",
"default", "module" or "once"
# WINE DATASET
# 1.2.6 GMMHMM(With Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wine.data",header=None )
df.columns = ['Class','Alcohol','Malic acid','Ash','Alcalinity of␣
 ↪ash','Magnesium','Total phenols','Flavanoids', 'Nonflavanoid␣
 ↪phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted␣
 ↪wines','Proline']
X = df.drop(['Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GMMHMM(n_components=3, covariance_type="full",␣
 ↪init_params="mw")
classifier.startprob_ = np.array([0.6, 0.3, 0.1])
classifier.transmat_ = np.array([[0.7, 0.2, 0.1],
[0.3, 0.5, 0.2],
[0.3, 0.3, 0.4]])
classifier.covars_ = np.tile(np.identity(13), (3, 1, 1, 1))
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
y_pred_proba = classifier.predict_proba(X_test)
fpr, tpr, _ = metrics.roc_curve(y_test,
classifier.predict_proba(X_test)[:,1],pos_label=1)
auc = metrics.roc_auc_score(y_test, y_pred_proba,
multi_class='ovr')
#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 0:
    strings[i] = 1
  elif y_pred[i] == 1:
    strings[i] = 2
  else :
    strings[i] = 3
```

```
strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("--------------------------------------------------")
print("--------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("--------------------------------------------------")
print("--------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```




```
Confusion Matrix:
[[ 0 15  0]
 [24  2  1]
 [ 0  0 12]]
--------------------------------------------------
--------------------------------------------------
```

```
Performance Evaluation
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        15
           2       0.12      0.07      0.09        27
           3       0.92      1.00      0.96        12

    accuracy                           0.26        54
   macro avg       0.35      0.36      0.35        54
weighted avg       0.26      0.26      0.26        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.25925925925925924
```



qegcf8shn

September 23, 2023

[ ]:
```
pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading
hmmlearn-0.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (160
kB)
                           160.4/160.4
kB 5.3 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.10 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in
/usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-
packages (from hmmlearn) (1.11.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-
learn!=0.22.0,>=0.16->hmmlearn) (3.2.0)
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.0
```

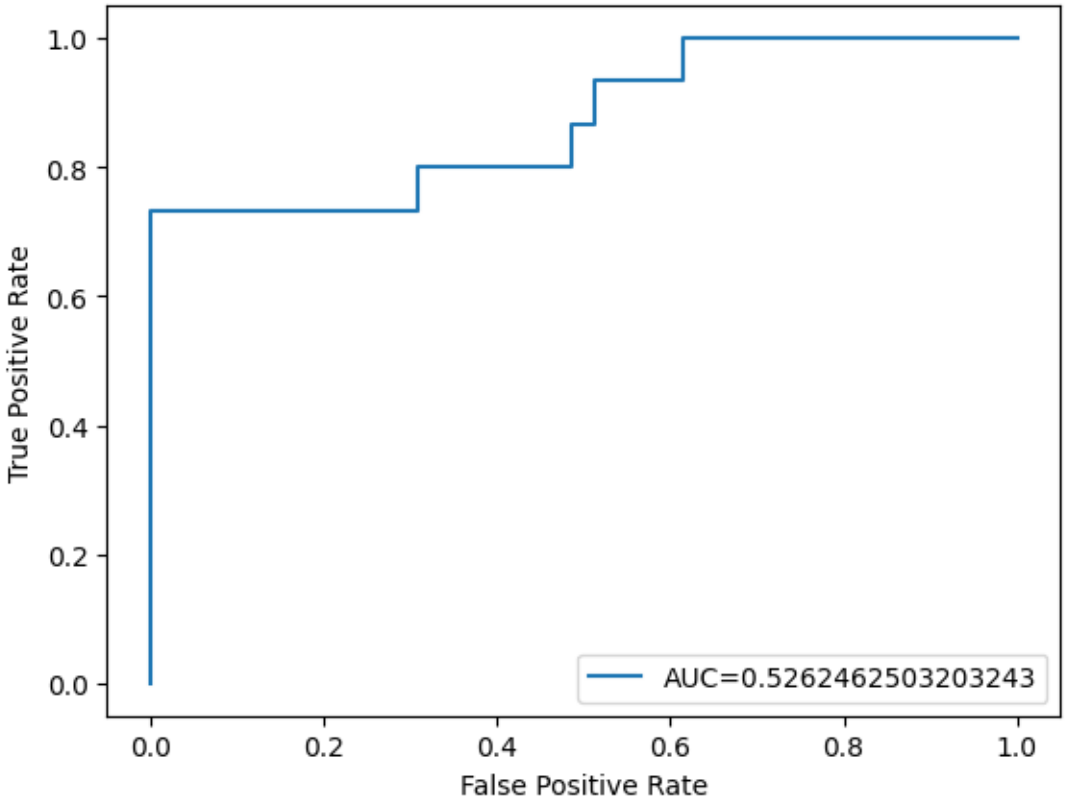
[ ]:
```
!gdown 1dQlEhfWb15RwU7IxBbqtdzqur0UHFqMG
```

```
Downloading…
From: https://drive.google.com/uc?id=1dQlEhfWb15RwU7IxBbqtdzqur0UHFqMG
To: /content/wine.data
100% 10.8k/10.8k [00:00<00:00, 23.0MB/s]
```

[ ]:
```
# GaussianHMM.py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn import metrics
```

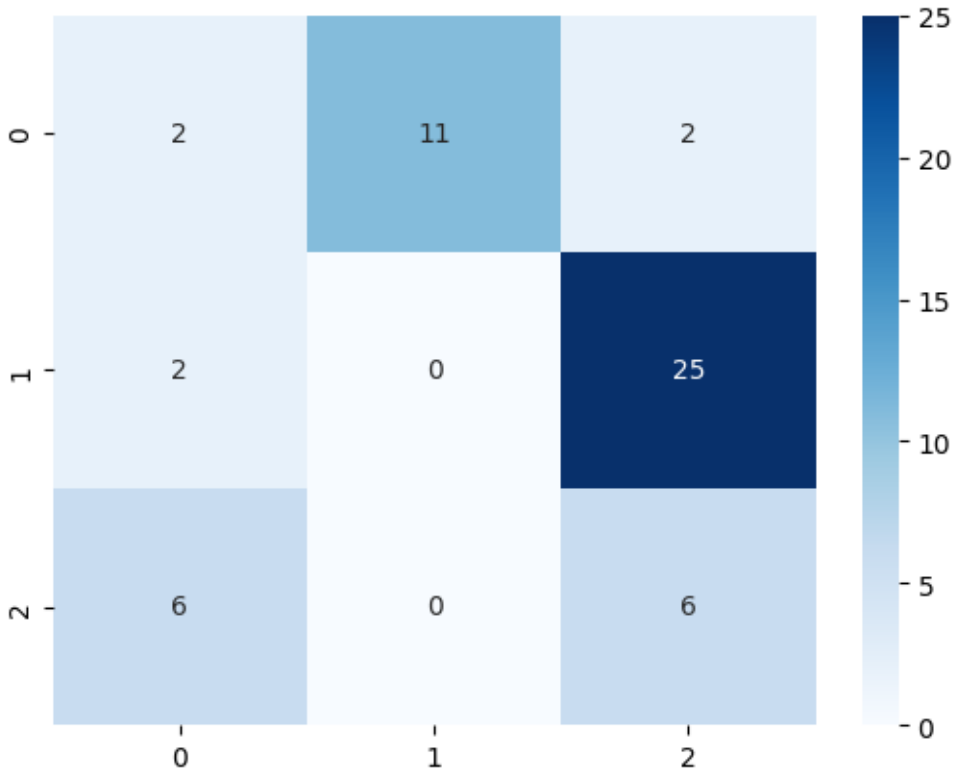
```
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
# WINE DATASET
# 1.1.1 GaussianHMM(Without Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wine.data",header=None )
df.columns = ['Class','Alcohol','Malic acid','Ash','Alcalinity of␣
 ↪ash','Magnesium','Total phenols','Flavanoids', 'Nonflavanoid␣
 ↪phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted␣
 ↪wines','Proline']
X = df.drop(['Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GaussianHMM(n_components=3,
covariance_type="full")
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
y_pred_proba = classifier.predict_proba(X_test)
fpr, tpr, _ = metrics.roc_curve(y_test,
classifier.predict_proba(X_test)[:,1],pos_label=1)
auc = metrics.roc_auc_score(y_test, y_pred_proba,
multi_class='ovr')
#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 0:
    strings[i] = 1
  elif y_pred[i] == 1:
    strings[i] = 2
  else :
    strings[i] = 3
strings = strings.astype(np.int32)
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("------------------------------------------------------")
```

```
print("-----------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("-----------------------------------------------------")
print("-----------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```




```
Confusion Matrix:
[[ 2 11  2]
 [ 2  0 25]
 [ 6  0  6]]
-----------------------------------------------------
-----------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.20      0.13      0.16        15
```

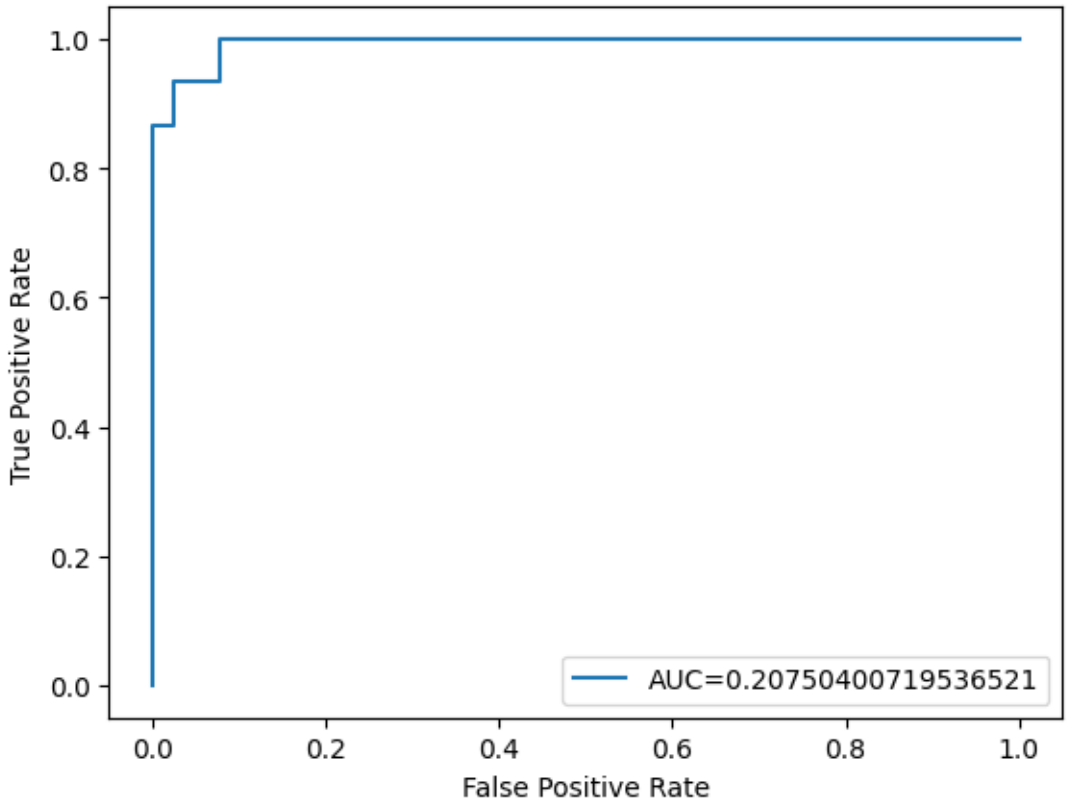
```
           2       0.00      0.00      0.00        27
           3       0.18      0.50      0.27        12

    accuracy                           0.15        54
   macro avg       0.13      0.21      0.14        54
weighted avg       0.10      0.15      0.10        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.14814814814814814
```



[ ]:
```
#GaussianHMM(with tuning).py
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from hmmlearn import hmm
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
import matplotlib.pyplot as plt
```

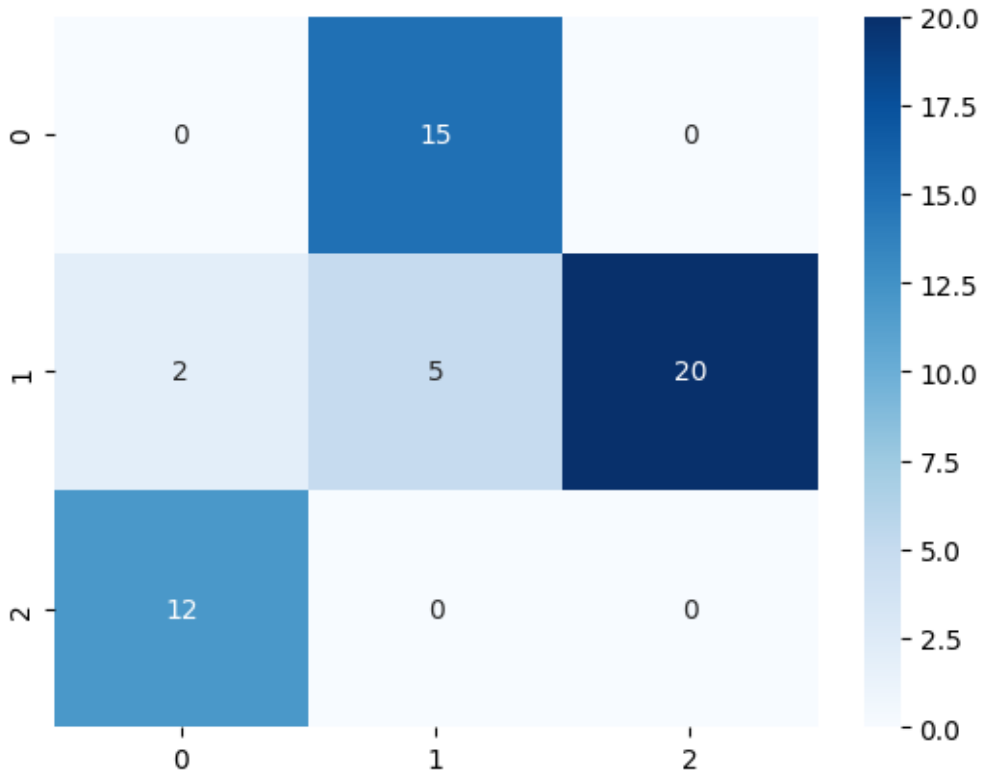
```
import seaborn as sns
# WINE DATASET
# 1.1.6 GaussianHMM(With Tuning)[70-30 split]
# Dataset Preparation
df = pd.read_csv("wine.data",header=None )
df.columns = ['Class','Alcohol','Malic acid','Ash','Alcalinity of␣
 ↪ash','Magnesium','Total phenols','Flavanoids', 'Nonflavanoid␣
 ↪phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted␣
 ↪wines','Proline']
X = df.drop(['Class'], axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.
 ↪7,test_size=0.3,random_state=10)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = hmm.GaussianHMM(n_components=3,
covariance_type="full", init_params="m")
classifier.startprob_ = np.array([0.6, 0.3, 0.1])
classifier.transmat_ = np.array([[0.7, 0.2, 0.1],
[0.3, 0.5, 0.2],
[0.3, 0.3, 0.4]])
classifier.covars_ = np.tile(np.identity(13), (3, 1, 1))
classifier.fit(X_train)
y_pred = classifier.predict(X_test)
size = len(y_pred)
y_pred_proba = classifier.predict_proba(X_test)
fpr, tpr, _ = metrics.roc_curve(y_test,
classifier.predict_proba(X_test)[:,1],pos_label=1)
auc = metrics.roc_auc_score(y_test, y_pred_proba,
multi_class='ovr')
#create ROC curve
plt.plot(fpr,tpr,label="AUC="+str(auc))
#plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc=4)
plt.show()
strings = np.empty(size, np.unicode_)
for i in range (size):
  if y_pred[i] == 0:
    strings[i] = 1
  elif y_pred[i] == 1:
    strings[i] = 2
  else :
    strings[i] = 3
strings = strings.astype(np.int32)
```

```
print("Confusion Matrix:")
print(confusion_matrix(y_test, strings))
print("--------------------------------------------------")
print("--------------------------------------------------")
print("Performance Evaluation")
print(classification_report(y_test, strings))
print("--------------------------------------------------")
print("--------------------------------------------------")
print("Accuracy:")
print(accuracy_score(y_test, strings))
cm = confusion_matrix(y_test, strings)
sns.heatmap(cm, annot=True , fmt="d",cmap='Blues')
plt.show()
```




```
Confusion Matrix:
[[ 0 15  0]
 [ 2  5 20]
 [12  0  0]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
```

```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        15
           2       0.25      0.19      0.21        27
           3       0.00      0.00      0.00        12

    accuracy                           0.09        54
   macro avg       0.08      0.06      0.07        54
weighted avg       0.12      0.09      0.11        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.09259259259259259
```



qauolx5je

September 23, 2023

[ ]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.applications import VGG19
from tensorflow.keras.utils import to_categorical
from keras.datasets import cifar10
```

[ ]:
```
(x_train,y_train), (x_test,y_test) = cifar10.load_data()
print("Shape of x_train is ",x_train.shape)
print("Shape of y_train is ",y_train.shape)
print("Shape of x_test is ",x_test.shape)
print("shape of y_test is",y_test.shape)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 2s 0us/step
Shape of x_train is  (50000, 32, 32, 3)
Shape of y_train is  (50000, 1)
Shape of x_test is  (10000, 32, 32, 3)
shape of y_test is (10000, 1)
```

[ ]:
```
def resize_img(img):
  numberOfImage = img.shape[0]
  new_array = np.zeros((numberOfImage, 48,48,3))
  for i in range(numberOfImage):
    new_array[i] = cv2.resize(img[i,:,:,:],(48,48))
  return new_array
```

[ ]:
```
x_test = resize_img(x_test)
x_train = resize_img(x_train)
print("New shape of x_train is ",x_train.shape)
print("New shape of x_test is ",x_test.shape)
```

```
New shape of x_train is  (50000, 48, 48, 3)
New shape of x_test is  (10000, 48, 48, 3)
```

[ ]:
```
# one hot encoding
y_train = to_categorical (y_train,num_classes=10)
y_test = to_categorical (y_test,num_classes=10)
print("New shape of y_train is ",y_train.shape)
print("New shape of y_test is ",y_test.shape)
```

```
New shape of y_train is  (50000, 10)
New shape of y_test is  (10000, 10)
```

[ ]:
```
# Include top = add fully connected layers to layer.
# Weights = use pretrained weights (trained in imagenet)
vgg = VGG19(include_top=False, weights="imagenet", input_shape=(48,48,3))
vgg.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 [==============================] - 0s 0us/step
Model: "vgg19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 48, 48, 3)]       0

 block1_conv1 (Conv2D)       (None, 48, 48, 64)        1792

 block1_conv2 (Conv2D)       (None, 48, 48, 64)        36928

 block1_pool (MaxPooling2D)  (None, 24, 24, 64)        0

 block2_conv1 (Conv2D)       (None, 24, 24, 128)       73856

 block2_conv2 (Conv2D)       (None, 24, 24, 128)       147584

 block2_pool (MaxPooling2D)  (None, 12, 12, 128)       0

 block3_conv1 (Conv2D)       (None, 12, 12, 256)       295168

 block3_conv2 (Conv2D)       (None, 12, 12, 256)       590080

 block3_conv3 (Conv2D)       (None, 12, 12, 256)       590080

 block3_conv4 (Conv2D)       (None, 12, 12, 256)       590080

 block3_pool (MaxPooling2D)  (None, 6, 6, 256)         0

 block4_conv1 (Conv2D)       (None, 6, 6, 512)         1180160

 block4_conv2 (Conv2D)       (None, 6, 6, 512)         2359808
```

```
 block4_conv3 (Conv2D)       (None, 6, 6, 512)         2359808

 block4_conv4 (Conv2D)       (None, 6, 6, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 3, 3, 512)         0

 block5_conv1 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_conv2 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_conv3 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_conv4 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 1, 1, 512)         0

=================================================================
Total params: 20024384 (76.39 MB)
Trainable params: 20024384 (76.39 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```python
model = Sequential()
# Adding layers to the blank model
for layer in vgg.layers:
  model.add(layer)
# Don't train layers again, because they are already trained
for layer in model.layers:
  layer.trainable = False
# Adding fully connected layers
model.add(Flatten())
model.add(Dense(128))
model.add(Dense (10, activation="softmax"))
# Checking model
model.summary()
```

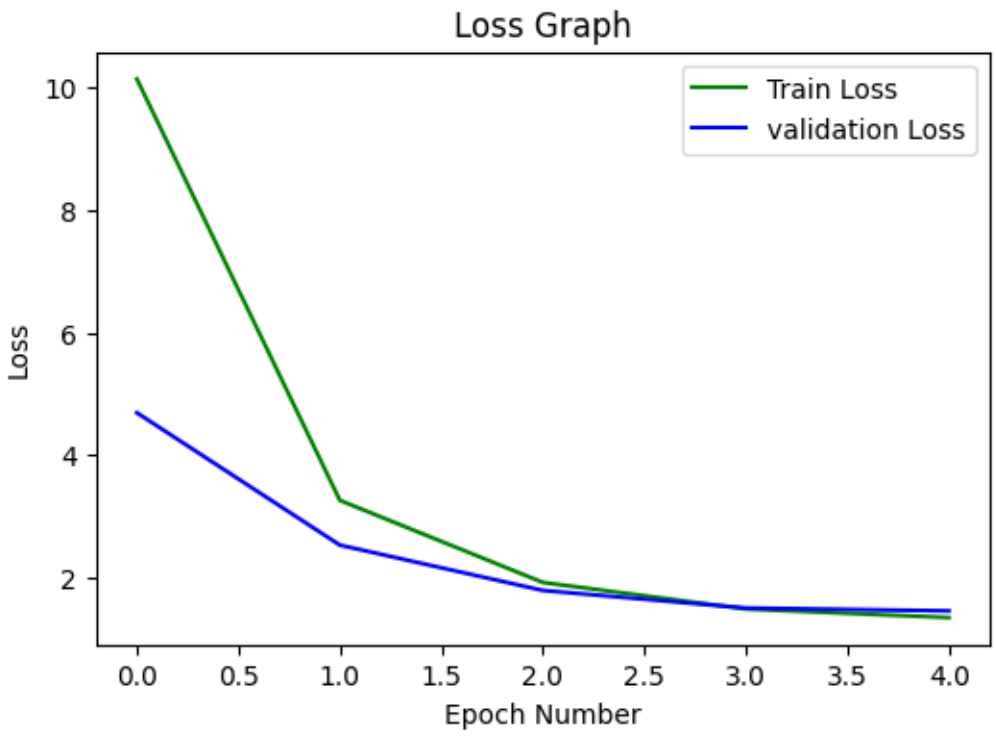
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 block1_conv1 (Conv2D)       (None, 48, 48, 64)        1792

 block1_conv2 (Conv2D)       (None, 48, 48, 64)        36928

 block1_pool (MaxPooling2D)  (None, 24, 24, 64)        0

 block2_conv1 (Conv2D)       (None, 24, 24, 128)       73856
```

```
 block2_conv2 (Conv2D)       (None, 24, 24, 128)       147584

 block2_pool (MaxPooling2D)  (None, 12, 12, 128)       0

 block3_conv1 (Conv2D)       (None, 12, 12, 256)       295168

 block3_conv2 (Conv2D)       (None, 12, 12, 256)       590080

 block3_conv3 (Conv2D)       (None, 12, 12, 256)       590080

 block3_conv4 (Conv2D)       (None, 12, 12, 256)       590080

 block3_pool (MaxPooling2D)  (None, 6, 6, 256)         0

 block4_conv1 (Conv2D)       (None, 6, 6, 512)         1180160

 block4_conv2 (Conv2D)       (None, 6, 6, 512)         2359808

 block4_conv3 (Conv2D)       (None, 6, 6, 512)         2359808

 block4_conv4 (Conv2D)       (None, 6, 6, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 3, 3, 512)         0

 block5_conv1 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_conv2 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_conv3 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_conv4 (Conv2D)       (None, 3, 3, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 1, 1, 512)         0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 128)               65664

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 20091338 (76.64 MB)
Trainable params: 66954 (261.54 KB)
Non-trainable params: 20024384 (76.39 MB)
_________________________________________________________________
```

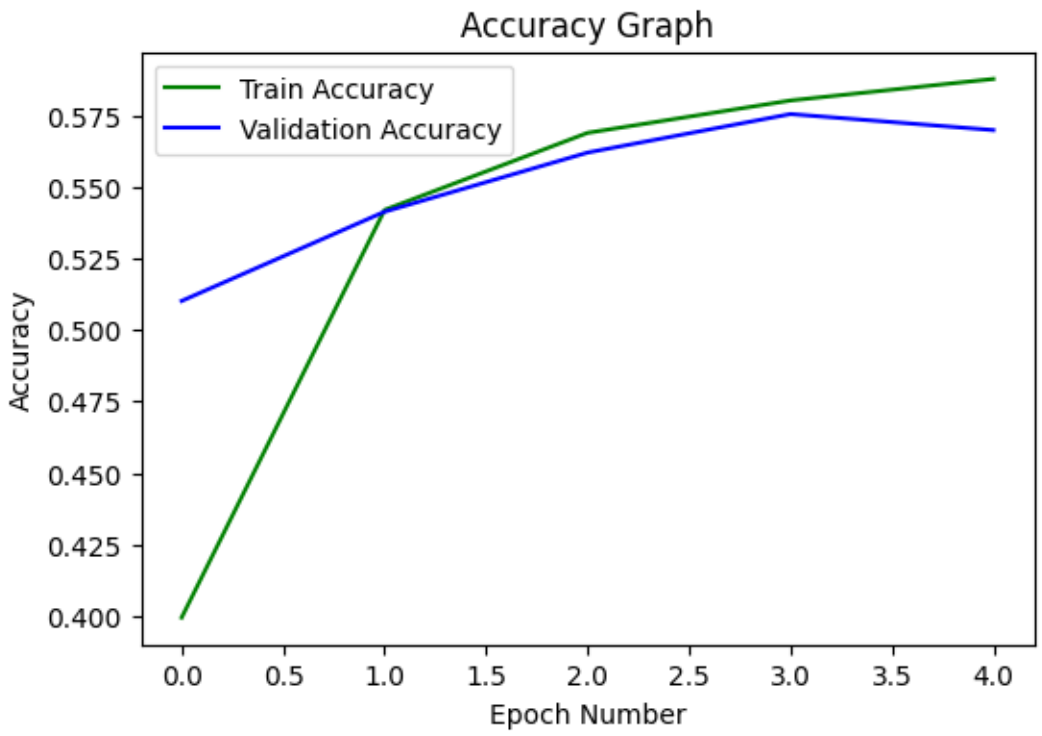
[ ]:
```
model.compile(optimizer="Adam", loss="categorical_crossentropy",␣
 ↪metrics=["accuracy"])
# Let's train.
hist = model.fit(x_train,y_train, validation_split=0.15, epochs=5,␣
 ↪batch_size=1000)
# We use %15 of the train set as validation set.
```

```
Epoch 1/5
43/43 [==============================] - 32s 410ms/step - loss: 10.1428 -
accuracy: 0.3995 - val_loss: 4.6913 - val_accuracy: 0.5101
Epoch 2/5
43/43 [==============================] - 13s 314ms/step - loss: 3.2601 -
accuracy: 0.5420 - val_loss: 2.5295 - val_accuracy: 0.5413
Epoch 3/5
43/43 [==============================] - 14s 316ms/step - loss: 1.9198 -
accuracy: 0.5689 - val_loss: 1.7893 - val_accuracy: 0.5620
Epoch 4/5
43/43 [==============================] - 13s 304ms/step - loss: 1.4856 -
accuracy: 0.5802 - val_loss: 1.5025 - val_accuracy: 0.5755
Epoch 5/5
43/43 [==============================] - 14s 321ms/step - loss: 1.3463 -
accuracy: 0.5877 - val_loss: 1.4559 - val_accuracy: 0.5699
```

[ ]:
```
plt.subplots(figsize=(6,4))
plt.plot(hist.epoch, hist.history["loss"],color="green", label="Train Loss")
plt.plot(hist.epoch, hist.history["val_loss"],color="blue", label="validation␣
 ↪Loss")
plt.xlabel("Epoch Number")
plt.ylabel("Loss")
plt.legend()
plt.title("Loss Graph")
plt.show()
```



[ ]:
```
plt.subplots(figsize=(6,4))
plt.plot(hist.epoch, hist.history["accuracy"], color="green", label="Train␣
 ↪Accuracy")
plt.plot(hist.epoch, hist.history["val_accuracy"], color="blue",␣
 ↪label="Validation Accuracy")
plt.xlabel("Epoch Number")
plt.ylabel("Accuracy")
plt.legend()
plt.title("Accuracy Graph")
plt.show()
```



[ ]:
```
model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 6s 18ms/step - loss: 1.4454 -
accuracy: 0.5684
```

[ ]:
```
[1.4454282522201538, 0.5684000253677368]
```

mfonkikph

September 23, 2023

[ ]:
```
#MNIST.py
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

[ ]:
```
#dataset preparation
from tensorflow.keras import datasets,layers,models
```

[ ]:
```
#loading dataset
(train_images, train_labels), (test_images, test_labels) = datasets.mnist.
 ↪load_data()
```

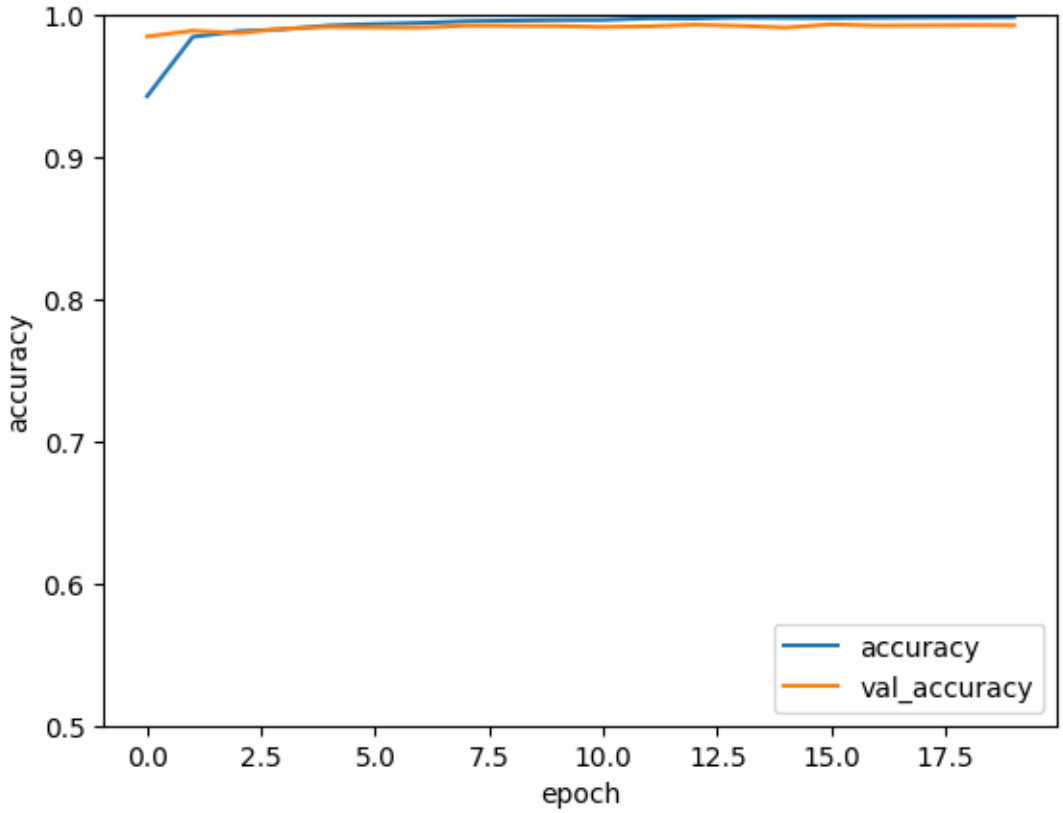
[ ]:
```
#Normalise pixel values to be within 0,1
train_images, test_images = train_images/255.0, test_images/255.0

train_images = np.reshape(train_images, train_images.shape + (1,))
test_images = np.reshape(test_images, test_images.shape + (1,))
print(train_images[0].shape)

model = models.Sequential()
model.add(layers.Conv2D(32,(3,3),activation='relu',input_shape=(28,28,1)))
model.add(layers.MaxPool2D(2,2))
model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.MaxPool2D(2,2))
model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64,activation ='relu'))
model.add(layers.Dense(10))
model.summary()

model.compile(optimizer='adam', loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),metrics=['accuracy'])
history = model.
 ↪fit(train_images,train_labels,epochs=20,batch_size=64,validation_data=(test_images,test_lab

plt.plot(history.history['accuracy'],label='accuracy')
```

```
plt.plot(history.history['val_accuracy'],label='val_accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.ylim([0.5,1])
plt.legend(loc='lower right')
plt.show()

test_loss, test_acc = model.evaluate(test_images,test_labels,verbose=2)

print(test_loss)
print(test_acc)
```

```
(28, 28, 1)
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_5 (Conv2D)           (None, 3, 3, 64)          36928

 flatten_1 (Flatten)         (None, 576)               0

 dense_2 (Dense)             (None, 64)                36928

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
938/938 [==============================] - 6s 5ms/step - loss: 0.1872 -
accuracy: 0.9429 - val_loss: 0.0455 - val_accuracy: 0.9849
Epoch 2/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0512 -
accuracy: 0.9847 - val_loss: 0.0318 - val_accuracy: 0.9890
Epoch 3/20
```

```
938/938 [==============================] - 5s 5ms/step - loss: 0.0365 -
accuracy: 0.9886 - val_loss: 0.0410 - val_accuracy: 0.9873
Epoch 4/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0310 -
accuracy: 0.9900 - val_loss: 0.0308 - val_accuracy: 0.9905
Epoch 5/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0234 -
accuracy: 0.9927 - val_loss: 0.0260 - val_accuracy: 0.9916
Epoch 6/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0194 -
accuracy: 0.9937 - val_loss: 0.0274 - val_accuracy: 0.9913
Epoch 7/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0167 -
accuracy: 0.9945 - val_loss: 0.0284 - val_accuracy: 0.9912
Epoch 8/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0128 -
accuracy: 0.9957 - val_loss: 0.0246 - val_accuracy: 0.9925
Epoch 9/20
938/938 [==============================] - 4s 4ms/step - loss: 0.0114 -
accuracy: 0.9962 - val_loss: 0.0265 - val_accuracy: 0.9925
Epoch 10/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0104 -
accuracy: 0.9965 - val_loss: 0.0274 - val_accuracy: 0.9923
Epoch 11/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0098 -
accuracy: 0.9966 - val_loss: 0.0325 - val_accuracy: 0.9916
Epoch 12/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0067 -
accuracy: 0.9977 - val_loss: 0.0309 - val_accuracy: 0.9921
Epoch 13/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0072 -
accuracy: 0.9976 - val_loss: 0.0305 - val_accuracy: 0.9932
Epoch 14/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0058 -
accuracy: 0.9983 - val_loss: 0.0347 - val_accuracy: 0.9924
Epoch 15/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0057 -
accuracy: 0.9981 - val_loss: 0.0400 - val_accuracy: 0.9912
Epoch 16/20
938/938 [==============================] - 5s 5ms/step - loss: 0.0056 -
accuracy: 0.9980 - val_loss: 0.0329 - val_accuracy: 0.9935
Epoch 17/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0057 -
accuracy: 0.9982 - val_loss: 0.0341 - val_accuracy: 0.9926
Epoch 18/20
938/938 [==============================] - 4s 4ms/step - loss: 0.0050 -
accuracy: 0.9983 - val_loss: 0.0378 - val_accuracy: 0.9927
Epoch 19/20
```

```
938/938 [==============================] - 5s 5ms/step - loss: 0.0048 -
accuracy: 0.9984 - val_loss: 0.0355 - val_accuracy: 0.9929
Epoch 20/20
938/938 [==============================] - 4s 5ms/step - loss: 0.0049 -
accuracy: 0.9985 - val_loss: 0.0417 - val_accuracy: 0.9928
```




```
313/313 - 1s - loss: 0.0417 - accuracy: 0.9928 - 1s/epoch - 3ms/step
0.04167570173740387
0.9927999973297119
```

tnydkqkbb

September 23, 2023

[ ]:
```
#CIFAR-10.py
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

[ ]:
```
#dataset preparation
from tensorflow.keras import datasets,layers,models
```

[ ]:
```
#loading dataset
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.
 ↪load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 2s 0us/step
```

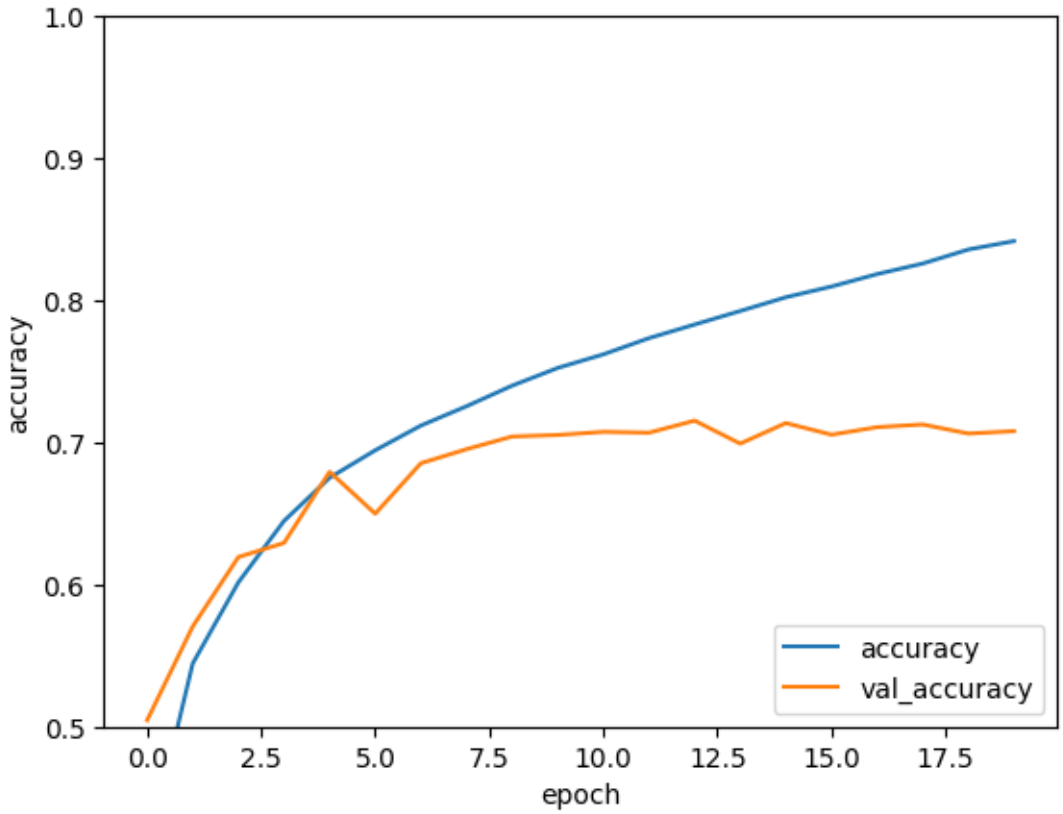
[ ]:
```
#pre-processing
train_images, test_images = train_images/255.0, test_images/255.0
```

[ ]:
```
#Normalise pixel values to be within 0,1
input_shape = train_images[0].shape
model = models.Sequential()
model.add(layers.Conv2D(32,(3,3),activation='relu',input_shape=input_shape))
model.add(layers.MaxPool2D(2,2))
model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.MaxPool2D(2,2))
model.add(layers.Conv2D(64,(3,3),activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64,activation ='relu'))
model.add(layers.Dense(10))
model.summary

model.compile(optimizer='adam', loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),metrics=['accuracy'])
history = model.
 ↪fit(train_images,train_labels,epochs=20,batch_size=64,validation_data=(test_images,test_lab
```

```
plt.plot(history.history['accuracy'],label='accuracy')
plt.plot(history.history['val_accuracy'],label='val_accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.ylim([0.5,1])
plt.legend(loc='lower right')
plt.show()

test_loss, test_acc = model.evaluate(test_images,test_labels,verbose=2)

print(test_loss)
print(test_acc)
```

```
Epoch 1/20
782/782 [==============================] - 7s 6ms/step - loss: 1.6103 -
accuracy: 0.4103 - val_loss: 1.3455 - val_accuracy: 0.5045
Epoch 2/20
782/782 [==============================] - 4s 5ms/step - loss: 1.2686 -
accuracy: 0.5444 - val_loss: 1.2047 - val_accuracy: 0.5705
Epoch 3/20
782/782 [==============================] - 4s 5ms/step - loss: 1.1206 -
accuracy: 0.6015 - val_loss: 1.0868 - val_accuracy: 0.6193
Epoch 4/20
782/782 [==============================] - 4s 6ms/step - loss: 1.0128 -
accuracy: 0.6447 - val_loss: 1.0509 - val_accuracy: 0.6292
Epoch 5/20
782/782 [==============================] - 4s 5ms/step - loss: 0.9290 -
accuracy: 0.6751 - val_loss: 0.9280 - val_accuracy: 0.6793
Epoch 6/20
782/782 [==============================] - 4s 5ms/step - loss: 0.8730 -
accuracy: 0.6944 - val_loss: 0.9996 - val_accuracy: 0.6498
Epoch 7/20
782/782 [==============================] - 4s 6ms/step - loss: 0.8231 -
accuracy: 0.7118 - val_loss: 0.9204 - val_accuracy: 0.6853
Epoch 8/20
782/782 [==============================] - 4s 6ms/step - loss: 0.7806 -
accuracy: 0.7254 - val_loss: 0.8852 - val_accuracy: 0.6952
Epoch 9/20
782/782 [==============================] - 4s 5ms/step - loss: 0.7462 -
accuracy: 0.7399 - val_loss: 0.8742 - val_accuracy: 0.7041
Epoch 10/20
782/782 [==============================] - 5s 6ms/step - loss: 0.7108 -
accuracy: 0.7524 - val_loss: 0.8621 - val_accuracy: 0.7052
Epoch 11/20
782/782 [==============================] - 4s 5ms/step - loss: 0.6789 -
accuracy: 0.7619 - val_loss: 0.8708 - val_accuracy: 0.7074
Epoch 12/20
```

```
782/782 [==============================] - 4s 6ms/step - loss: 0.6480 -
accuracy: 0.7733 - val_loss: 0.8603 - val_accuracy: 0.7068
Epoch 13/20
782/782 [==============================] - 4s 5ms/step - loss: 0.6204 -
accuracy: 0.7829 - val_loss: 0.8531 - val_accuracy: 0.7153
Epoch 14/20
782/782 [==============================] - 4s 5ms/step - loss: 0.5921 -
accuracy: 0.7924 - val_loss: 0.9323 - val_accuracy: 0.6991
Epoch 15/20
782/782 [==============================] - 5s 6ms/step - loss: 0.5658 -
accuracy: 0.8021 - val_loss: 0.8685 - val_accuracy: 0.7136
Epoch 16/20
782/782 [==============================] - 4s 5ms/step - loss: 0.5436 -
accuracy: 0.8097 - val_loss: 0.9269 - val_accuracy: 0.7054
Epoch 17/20
782/782 [==============================] - 4s 5ms/step - loss: 0.5183 -
accuracy: 0.8184 - val_loss: 0.9186 - val_accuracy: 0.7107
Epoch 18/20
782/782 [==============================] - 5s 6ms/step - loss: 0.4949 -
accuracy: 0.8258 - val_loss: 0.9135 - val_accuracy: 0.7126
Epoch 19/20
782/782 [==============================] - 4s 5ms/step - loss: 0.4719 -
accuracy: 0.8357 - val_loss: 0.9552 - val_accuracy: 0.7062
Epoch 20/20
782/782 [==============================] - 4s 5ms/step - loss: 0.4508 -
accuracy: 0.8416 - val_loss: 0.9833 - val_accuracy: 0.7079
```




```
313/313 - 1s - loss: 0.9833 - accuracy: 0.7079 - 683ms/epoch - 2ms/step
0.9833196997642517
0.7078999876976013
```