Summary of all analysis performed can be found in the last five pages.

```
import os

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from google.colab import drive

from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier as RFC

from scipy.stats import ttest_1samp # !pip install --upgrade scipy

drive.mount("/content/drive", force_remount=True)
os.chdir('/content/drive/MyDrive/Colab Notebooks/PSY3100')
```

```
Mounted at /content/drive
```

Making it easy to iterate over different files and type of image easily.

```
seed = 0
user = 'S01'
area = 'RSC'
image_type = 'original'

df = pd.read_csv(f'fMRI_Scenes/{user}_{area}.csv',sep=r',',skipinitialspace = True, ir
df = df.loc[image_type]
df = df.sample(frac=1)
n_classes = len(df.category.unique())


summary = list()

def make_ttest(x):
  # function to make easy calculation of t test for each classification method
 return ttest_1samp(x, 1/n_classes, axis=0, nan_policy='propagate', alternative='great

def append_summary(method_name, strategy, ttest):
  # function to make easy append results of t test to a list
  summary.append({"area": area, "image type": image type, "method": method name, "stra
```

```
def show_confusion_matrix(model, X, y):
  # making it easier to show confusion matrix for each subject
  predictions = model.predict(X)
  cm = confusion_matrix(y, predictions, labels=model.classes_)
  disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                                display_labels=model.classes_)
  disp.plot()
  plt.show()
```

First Analysis: RSC and original images

Data-splitting strategy

The goal was to select approximatelly 20% of the trials, while avoiding to split runs. To do so, I selected one run at random.

```
np.random.seed(seed)
test_run = np.random.choice(df.run.unique(), 1).tolist()

df_train = df.loc[~df.run.isin(test_run)]
df_test = df.loc[df.run.isin(test_run)]

print('Train size:', len(df_train))
print('Test size:', len(df_test))
```

```
Train size: 288
Test size: 48
```

Checking if proportions are correct

```
df_train.category.value_counts(normalize=True)
```

```
beaches      0.166667
city         0.166667
highways     0.166667
offices      0.166667
forests      0.166667

mountains    0.166667
Name: category, dtype: float64
```

```
df_test.category.value_counts(normalize=True)
```

```
city         0.166667
mountains    0.166667
```

```
forests      0.166667
offices      0.166667
beaches      0.166667
highways     0.166667
Name: category, dtype: float64
```

Making sure only the variables that start with "vox" are in X and the categories in Y. Also, encoding variables to make sure all methods work

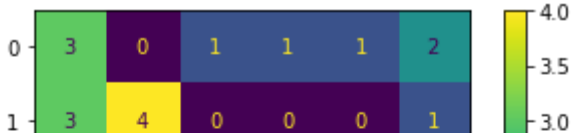
```
X_train, y_train = df_train.loc[:, [x for x in df_train.columns if x.startswith('vox')
X_test, y_test = df_test.loc[:, [x for x in df_test.columns if x.startswith('vox')]].t

encoder = LabelEncoder().fit(y_train)
y_train, y_test = encoder.transform(y_train), encoder.transform(y_test)

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

pca = PCA(n_components=60).fit(X_train)
X_train_pca, X_test_pca = pca.transform(X_train), pca.transform(X_test)
print('PCA explained variance ratio: %.3f' % pca.explained_variance_ratio_.sum())
```

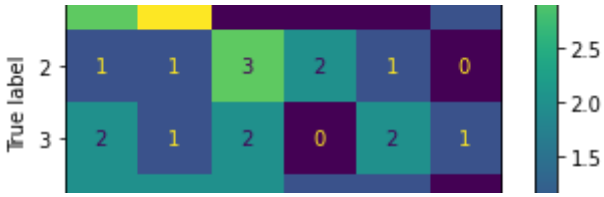
```
PCA explained variance ratio: 0.800
```

First model: Logistic Regression

```
# without PCA
model = LogisticRegression().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("LogisticRegression", None, make_ttest(model.predict(X_test) == y_test)
```

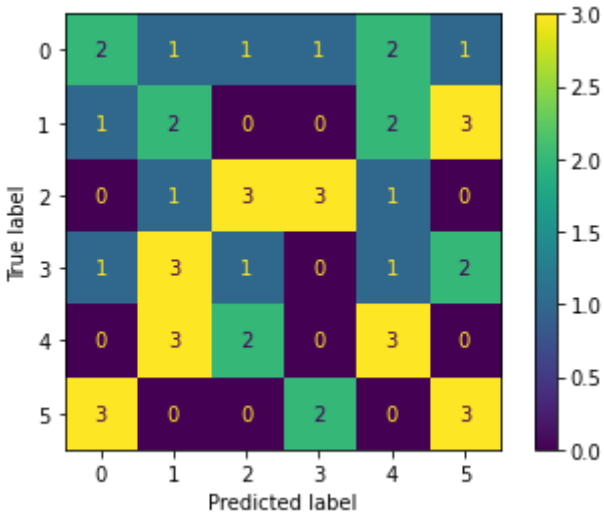
```
Train acc.: 1.000
Test acc.: 0.271
```






```
# with PCA
model = LogisticRegression().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("LogisticRegression", "PCA", make_ttest(model.predict(X_test_pca) == y_
```

```
Train acc.: 0.809
Test acc.: 0.271
```



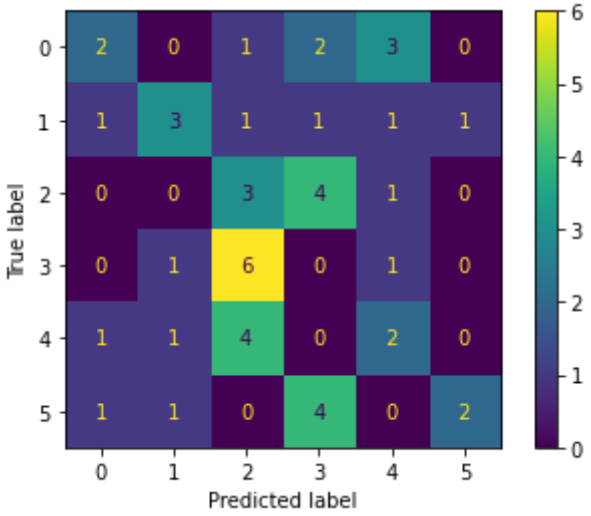
Second model: Support Vector Machine (SVM)

Note to self: "SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall."

```
# without PCA
model = SVC().fit(X_train, y_train)


print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("SVM", None, make_ttest(model.predict(X_test) == y_test))
```

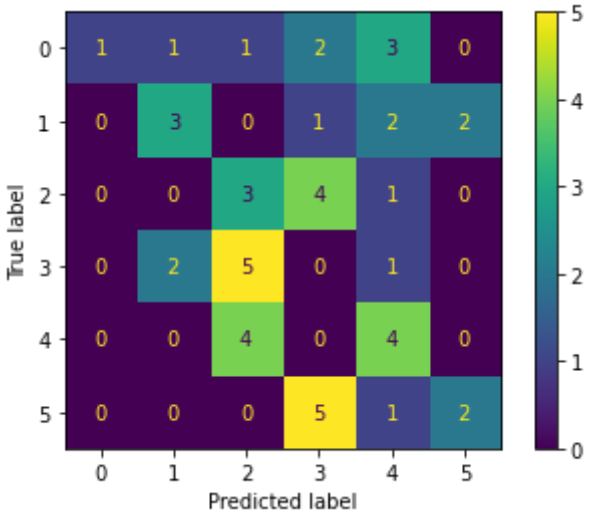
```
Train acc.: 0.990
```

```
Test acc.: 0.250
```




```
# with PCA
model = SVC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("SVM", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 0.976
Test acc.: 0.271
```

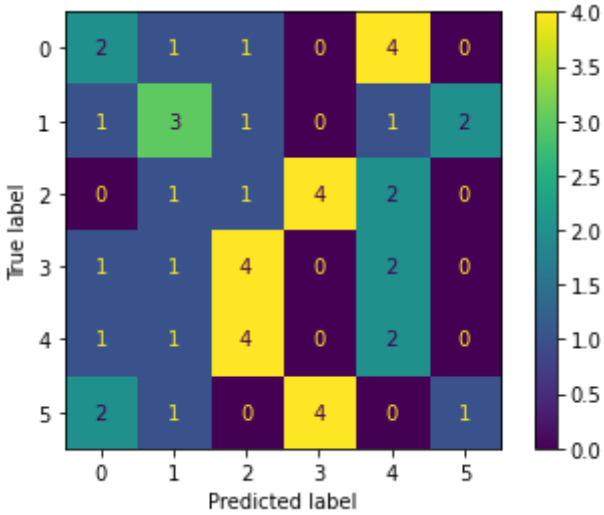


Third model: Random Forest Classifier (RFC)

```
# without PCA
model = RFC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
```

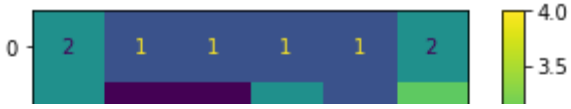
```
print('Train acc.: %.3f' % model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.188
```




```
# with PCA
model = RFC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("RFC", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.188
```

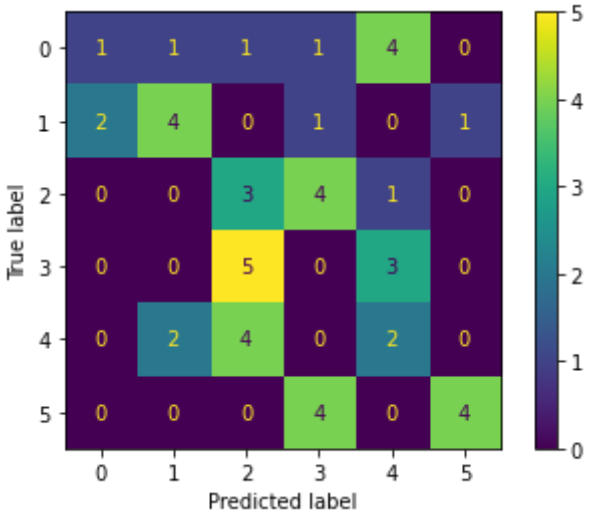


Random Search

Randomly picks 10 parameters, builds prediction, and then selects the best option of random choices.


```
# without PCA + RandomSearch
from sklearn.model_selection import RandomizedSearchCV

params = {'bootstrap': [True, False],
 'max_depth': [10, 30, 60, 90, None],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 800, 1000, 2000]}

model = RandomizedSearchCV(RFC(), params, n_iter=20, cv=3).fit(X_train, y_train).best_

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", "RandomSearch", make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.292
```



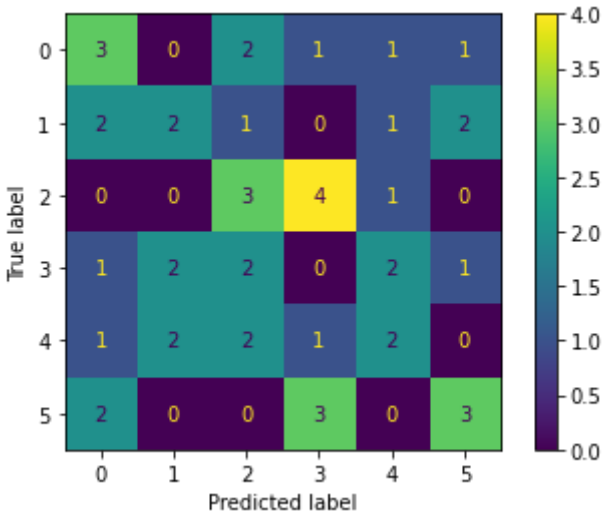
Fourth model: Neural Network - Multilayer Perceptron (MLP)

Note to self: "An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function."

```
# without PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train, y_train)
```

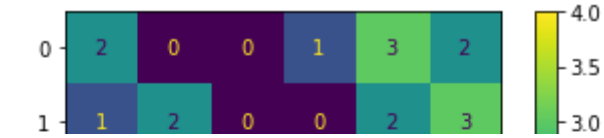
```
print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("MLP", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.271
```




```
# with PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("MLP", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.250
```



Results for RSC (original images)

```
pd.DataFrame(summary)
```

	area	image_type	method	strategy	ttest	pvalue
0	RSC	original	LogisticRegression	None	1.606990	0.057377
1	RSC	original	LogisticRegression	PCA	1.606990	0.057377
2	RSC	original	SVM	None	1.319371	0.096718
3	RSC	original	SVM	PCA	1.606990	0.057377
4	RSC	original	RFC	None	0.365928	0.358030
5	RSC	original	RFC	PCA	0.365928	0.358030
6	RSC	original	RFC	RandomSearch	1.885370	0.032785
7	RSC	original	MLP	None	1.606990	0.057377
8	RSC	original	MLP	PCA	1.319371	0.096718

Second Analysis: RSC and line drawings

```
seed = 0
user = 'S01'
area = 'RSC'
image_type = 'lineDrawings'

df = pd.read_csv(f'fMRI_Scenes/{user}_{area}.csv',sep=r',',skipinitialspace = True, ir
df = df.loc[image_type]
df = df.sample(frac=1)
n_classes = len(df.category.unique())
```

Data-splitting strategy

The goal was to select approximatelly 20% of the trials, while avoiding to split runs. To do so, I selected one run at random.

Spliting and treating data (same as done in first analysis)

```
np.random.seed(seed)
```

```
test_run = np.random.choice(df.run.unique(), 1).tolist()

df_train = df.loc[~df.run.isin(test_run)]
df_test = df.loc[df.run.isin(test_run)]

print('Train size:', len(df_train))
print('Test size:', len(df_test))

X_train, y_train = df_train.loc[:, [x for x in df_train.columns if x.startswith('vox')
X_test, y_test = df_test.loc[:, [x for x in df_test.columns if x.startswith('vox')]].t

encoder = LabelEncoder().fit(y_train)
y_train, y_test = encoder.transform(y_train), encoder.transform(y_test)

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

pca = PCA(n_components=60).fit(X_train)
X_train_pca, X_test_pca = pca.transform(X_train), pca.transform(X_test)
print('PCA explained variance ratio: %.3f' % pca.explained_variance_ratio_.sum())
```

```
Train size: 288
Test size: 48
PCA explained variance ratio: 0.793
```

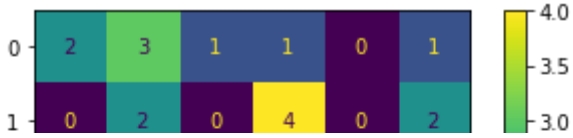
First model: Logistic Regression

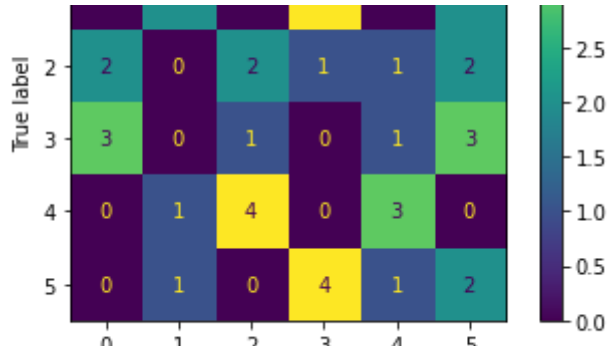
[ ] ↳ *2 cells hidden*

Second model: Support Vector Machine (SVM)

```
# without PCA
model = SVC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("SVM", None, make_ttest(model.predict(X_test) == y_test))
```

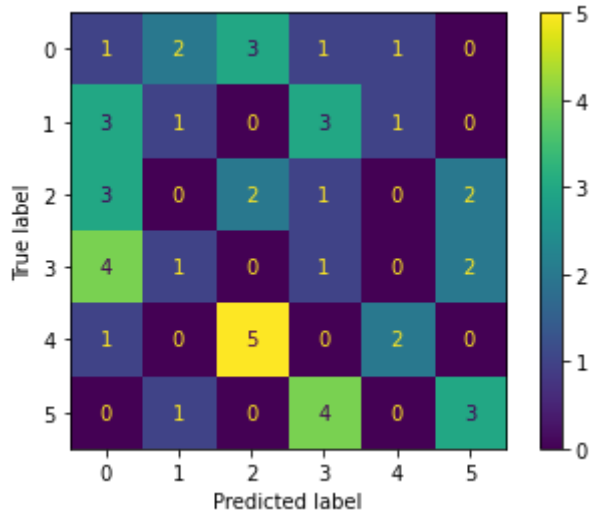
```
Train acc.: 0.997
Test acc.: 0.229
```






```
# with PCA
model = LogisticRegression().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("LogisticRegression", "PCA", make_ttest(model.predict(X_test_pca) == y_
```

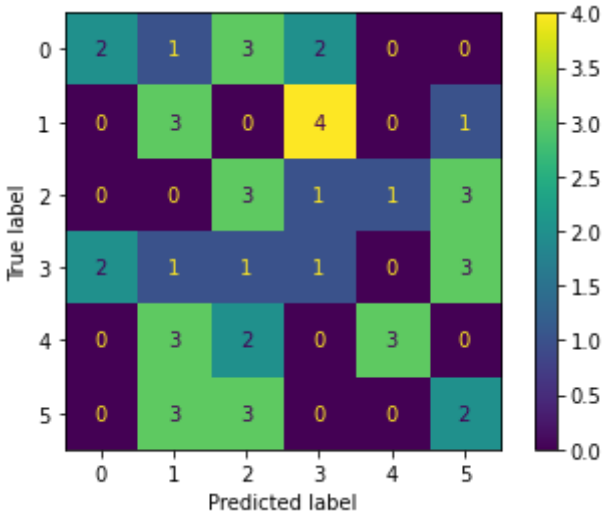
```
Train acc.: 0.753
Test acc.: 0.208
```



Third model: Random Forest Classifier (RFC)

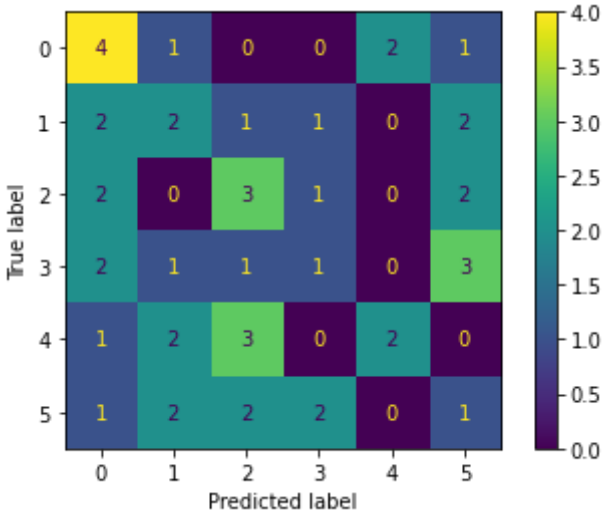
```
# without PCA
model = RFC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.292
```




```
# with PCA
model = RFC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("RFC", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.271
```

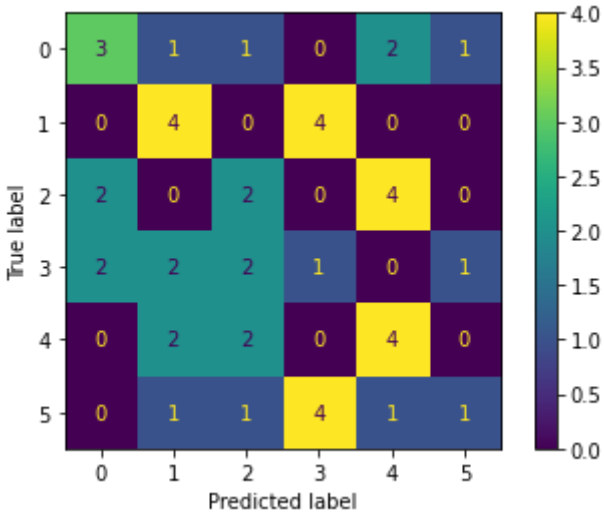


Random Search

```
# without PCA + RandomSearch
from sklearn.model_selection import RandomizedSearchCV

params = {'bootstrap': [True, False],
```

```
params = {'bootstrap': [True, False],
 'max_depth': [10, 30, 60, 90, None],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 800, 1000, 2000]}

model = RandomizedSearchCV(RFC(), params, n_iter=20, cv=3).fit(X_train, y_train).best_

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", "RandomSearch", make_ttest(model.predict(X_test) == y_test))
```

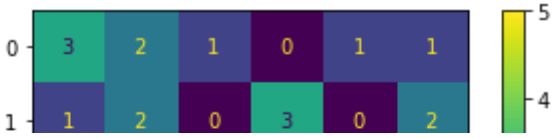
```
Train acc.: 1.000
Test acc.: 0.312
```

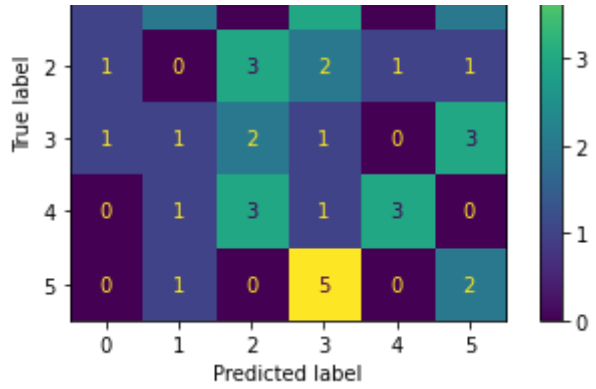


Fourth model: Neural Network - Multilayer Perceptron (MLP)

```
# without PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("MLP", None, make_ttest(model.predict(X_test) == y_test))
```

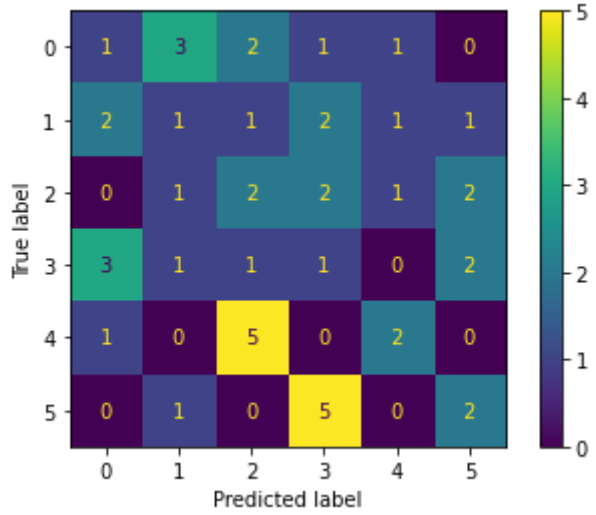
```
Train acc.: 1.000
Test acc.: 0.292
```






```
# with PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("MLP", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.188
```



Results for RSC (line drawings)

```
pd.DataFrame(summary)
```

	area	image_type	method	strategy	ttest	pvalue
0	RSC	original	LogisticRegression	None	1.606990	0.057377
1	RSC	original	LogisticRegression	PCA	1.606990	0.057377
2	RSC	original	SVM	None	1.319371	0.096718

3	RSC	original	SVM	PCA	1.606990	0.057377
4	RSC	original	RFC	None	0.365928	0.358030
5	RSC	original	RFC	PCA	0.365928	0.358030
6	RSC	original	RFC	RandomSearch	1.885370	0.032785
7	RSC	original	MLP	None	1.606990	0.057377
8	RSC	original	MLP	PCA	1.319371	0.096718
9	RSC	lineDrawings	LogisticRegression	None	0.365928	0.358030
10	RSC	lineDrawings	LogisticRegression	PCA	0.703375	0.242646
11	RSC	lineDrawings	SVM	None	1.019467	0.156600
12	RSC	lineDrawings	LogisticRegression	PCA	0.703375	0.242646
13	RSC	lineDrawings	RFC	None	1.885370	0.032785
14	RSC	lineDrawings	RFC	PCA	1.606990	0.057377
15	RSC	lineDrawings	RFC	RandomSearch	2.156971	0.018077
16	RSC	lineDrawings	MLP	None	1.885370	0.032785
17	RSC	lineDrawings	MLP	PCA	0.365928	0.358030

Third Analysis: V1 and original drawings

```
# opening right files
seed = 0
user = 'S01'
area = 'V1'
image_type = 'original'

df = pd.read_csv(f'fMRI_Scenes/{user}_{area}.csv',sep=r',',skipinitialspace = True, ir
df = df.loc[image_type]
df = df.sample(frac=1)
n_classes = len(df.category.unique())

# splitting dataset
np.random.seed(seed)

test_run = np.random.choice(df.run.unique(), 1).tolist()

df_train = df.loc[~df.run.isin(test_run)]
df_test = df.loc[df.run.isin(test_run)]

print('Train size:', len(df_train))
print('Test size:', len(df test))
```

```
# selecting only variables of interest and transforming data
X_train, y_train = df_train.loc[:, [x for x in df_train.columns if x.startswith('vox')
X_test, y_test = df_test.loc[:, [x for x in df_test.columns if x.startswith('vox')]].t

encoder = LabelEncoder().fit(y_train)
y_train, y_test = encoder.transform(y_train), encoder.transform(y_test)

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

pca = PCA(n_components=60).fit(X_train)
X_train_pca, X_test_pca = pca.transform(X_train), pca.transform(X_test)
print('PCA explained variance ratio: %.3f' % pca.explained_variance_ratio_.sum())
```

```
Train size: 288
Test size: 48
PCA explained variance ratio: 0.810
```

First model: Logistic Regression

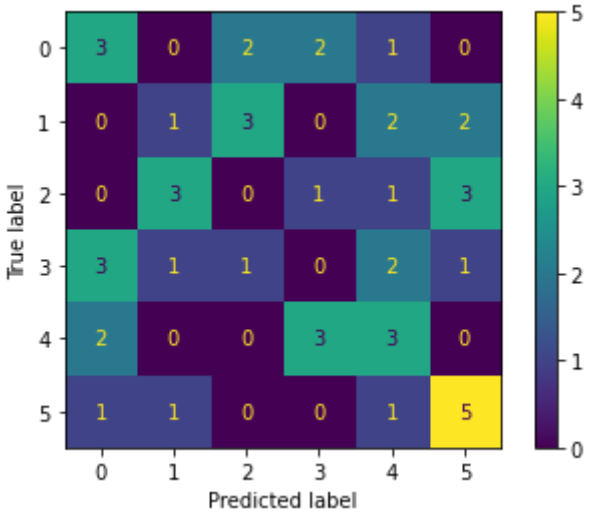
```
# without PCA
model = LogisticRegression().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("LogisticRegression", None, make_ttest(model.predict(X_test) == y_test)
```

```
Train acc.: 1.000
Test acc.: 0.125
```


```
# with PCA
model = LogisticRegression().fit(X_train_pca, y_train)
```

```
print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("LogisticRegression", "PCA", make_ttest(model.predict(X_test_pca) == y_
```

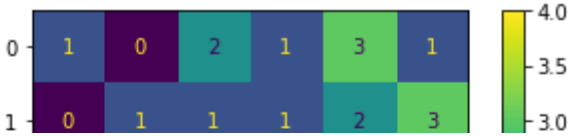
```
Train acc.: 0.639
Test acc.: 0.250
```

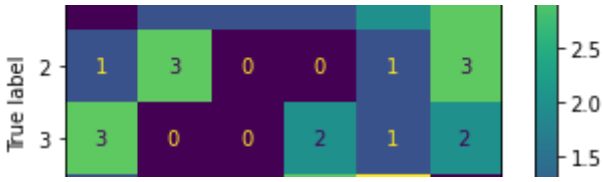


Second model: Support Vector Machine (SVM)

```
# without PCA
model = SVC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("SVM", None, make_ttest(model.predict(X_test) == y_test))
```

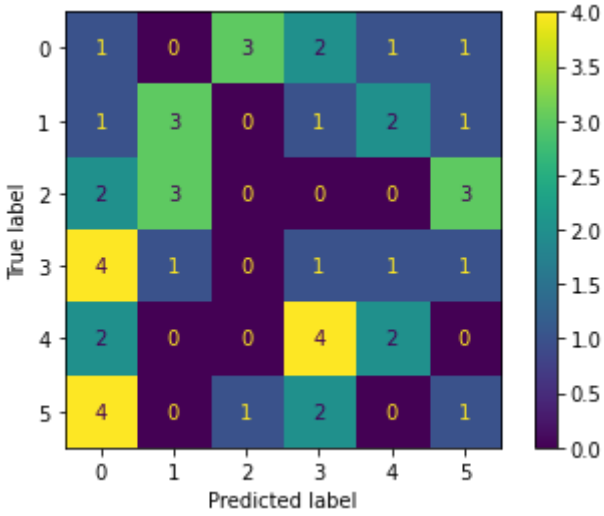
```
Train acc.: 0.986
Test acc.: 0.167
```






```
# with PCA
model = SVC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("SVM", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

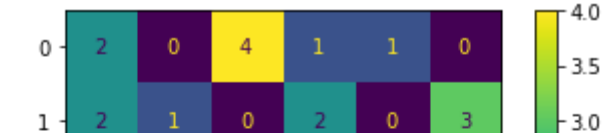
```
Train acc.: 0.958
Test acc.: 0.167
```

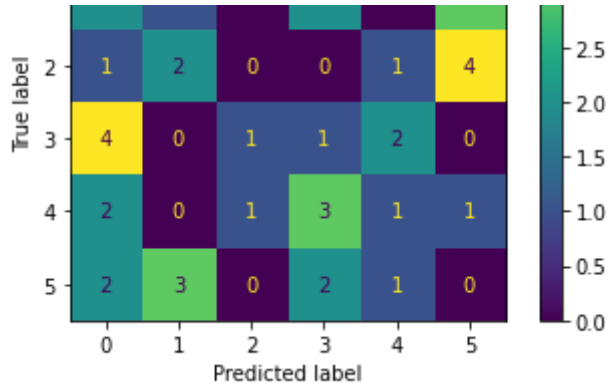


Third model: Random Forest Classifier (RFC)

```
# without PCA
model = RFC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", None, make_ttest(model.predict(X_test) == y_test))
```

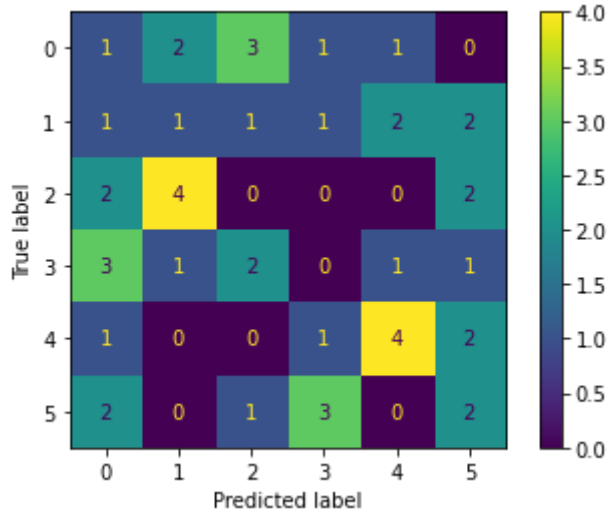
```
Train acc.: 1.000
Test acc.: 0.104
```






```
# with PCA
model = RFC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("RFC", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.167
```

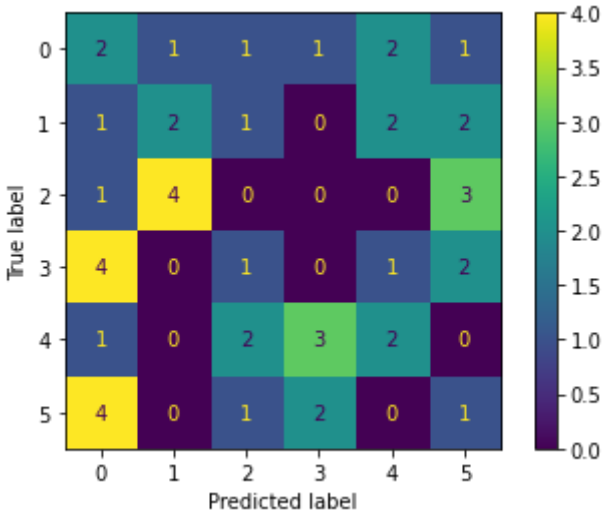


Random Search

```
# without PCA + RandomSearch
from sklearn.model_selection import RandomizedSearchCV

params = {'bootstrap': [True, False],
 'max_depth': [10, 30, 60, 90, None],
 'min_samples_split': [2, 5, 10],
```

```
    'n_estimators': [200, 400, 800, 1000, 2000]}

model = RandomizedSearchCV(RFC(), params, n_iter=20, cv=3).fit(X_train, y_train).best_

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", "RandomSearch", make_ttest(model.predict(X_test) == y_test))
```

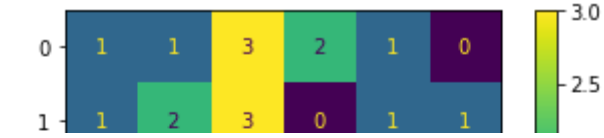
```
Train acc.: 1.000
Test acc.: 0.146
```

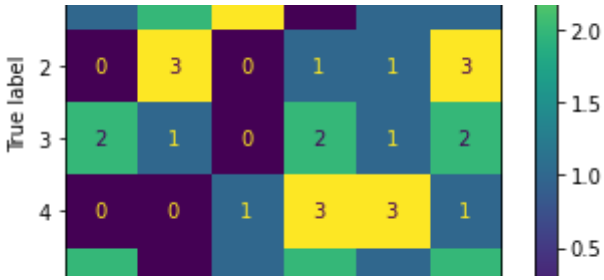


Fourth model: Neural Network - Multilayer Perceptron (MLP)

```
# without PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("MLP", None, make_ttest(model.predict(X_test) == y_test))
```

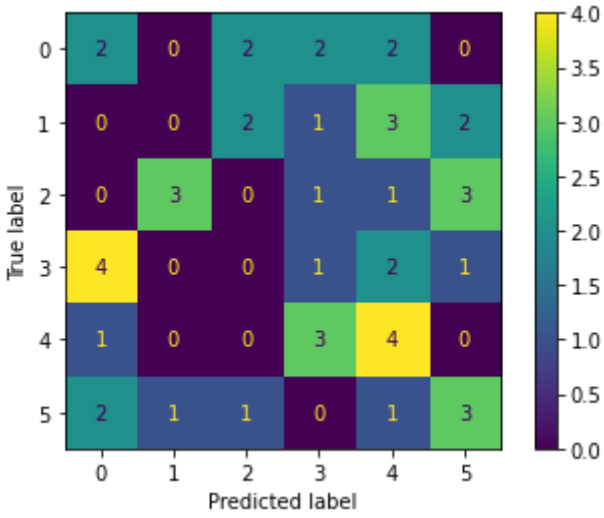
```
Train acc.: 1.000
Test acc.: 0.208
```






```
# with PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("MLP", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.208
```



Fourth Analysis: V1 and line drawings

```
# opening right files
seed = 0
user = 'S01'
area = 'V1'
image_type = 'lineDrawings'

df = pd.read_csv(f'fMRI_Scenes/{user}_{area}.csv',sep=r',',skipinitialspace = True, ir
df = df.loc[image_type]
df = df.sample(frac=1)
n_classes = len(df.category.unique())

# splitting dataset
```

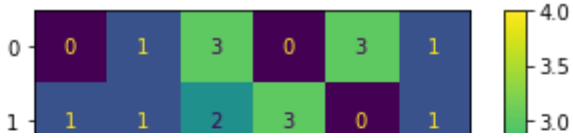
```
np.random.seed(seed)
test_run = np.random.choice(df.run.unique(), 1).tolist()

df_train = df.loc[~df.run.isin(test_run)]
df_test = df.loc[df.run.isin(test_run)]

print('Train size:', len(df_train))
print('Test size:', len(df_test))

# selecting only variables of interest and transforming data
X_train, y_train = df_train.loc[:, [x for x in df_train.columns if x.startswith('vox')
X_test, y_test = df_test.loc[:, [x for x in df_test.columns if x.startswith('vox')]].t

encoder = LabelEncoder().fit(y_train)
y_train, y_test = encoder.transform(y_train), encoder.transform(y_test)

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

pca = PCA(n_components=60).fit(X_train)
X_train_pca, X_test_pca = pca.transform(X_train), pca.transform(X_test)
print('PCA explained variance ratio: %.3f' % pca.explained_variance_ratio_.sum())
```

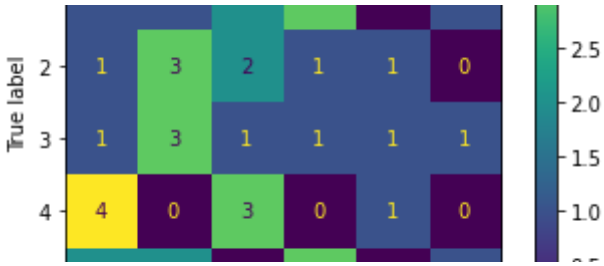
```
Train size: 288
Test size: 48
PCA explained variance ratio: 0.812
```

First model: Logistic Regression

```
# without PCA
model = LogisticRegression().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("LogisticRegression", None, make_ttest(model.predict(X_test) == y_test)
```

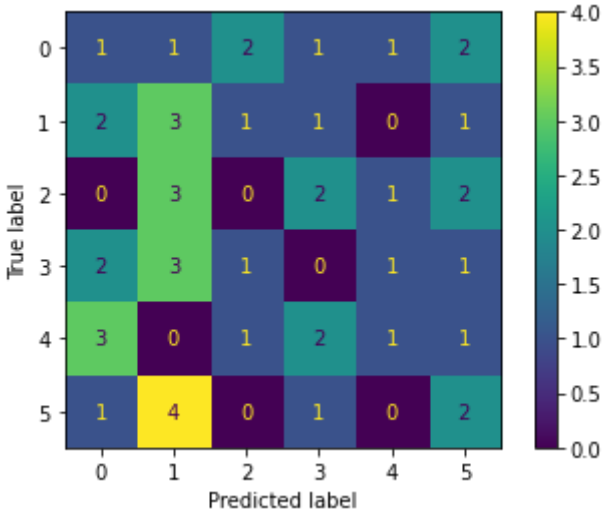
```
Train acc.: 1.000
Test acc.: 0.125
```






```
# with PCA
model = LogisticRegression().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("LogisticRegression", "PCA", make_ttest(model.predict(X_test_pca) == y_
```

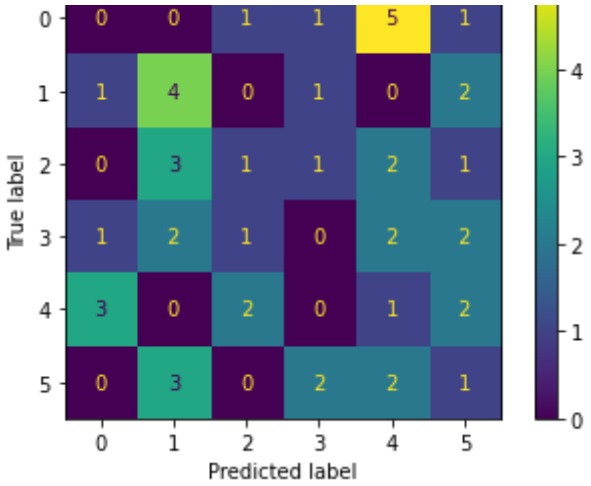
```
Train acc.: 0.667
Test acc.: 0.146
```



Second model: Support Vector Machine (SVM)

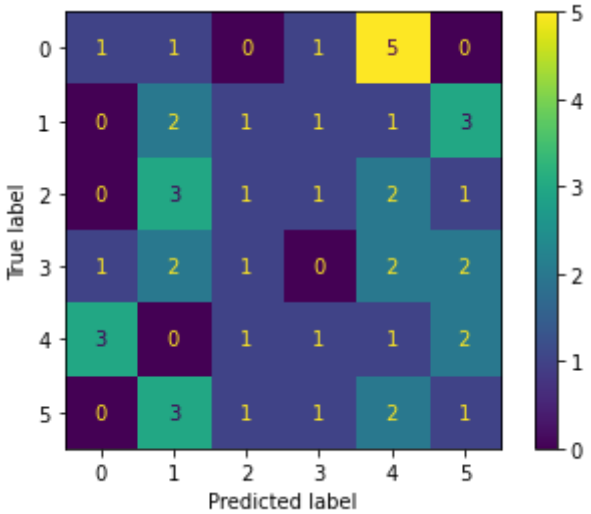
```
# without PCA
model = SVC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))


show_confusion_matrix(model, X_test, y_test)
append_summary("SVM", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 0.986
Test acc.: 0.146
```




```
# with PCA
model = SVC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("SVM", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 0.951
Test acc.: 0.125
```

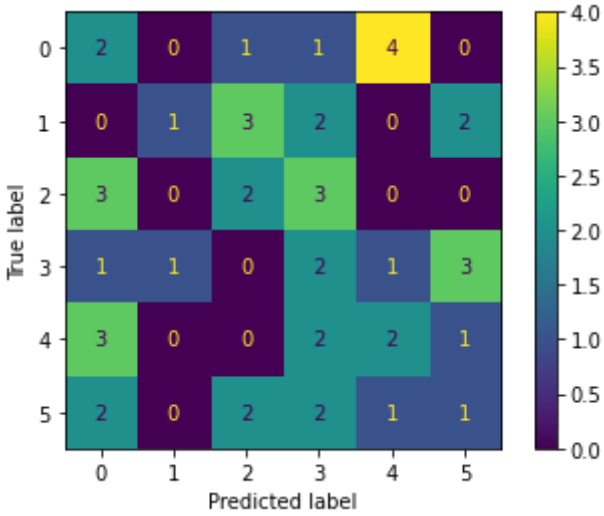


Third model: Random Forest Classifier (RFC)

```
# without PCA
model = RFC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))
```

```
show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.208
```



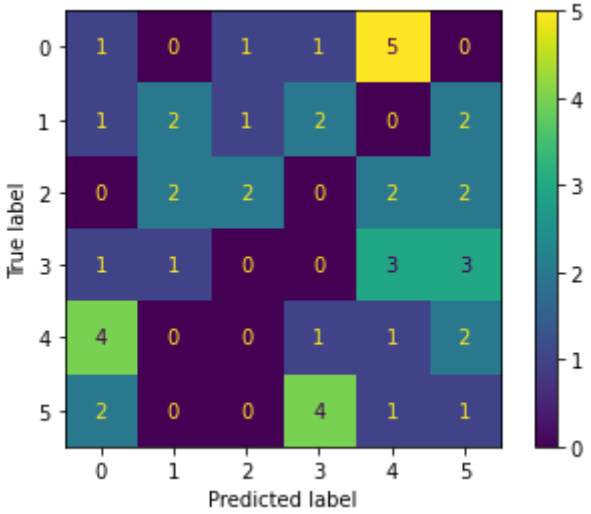

```
# with PCA
model = RFC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("RFC", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.250
```

Random Search

```
# without PCA + RandomSearch
from sklearn.model_selection import RandomizedSearchCV

params = {'bootstrap': [True, False],
 'max_depth': [10, 30, 60, 90, None],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 800, 1000, 2000]}

model = RandomizedSearchCV(RFC(), params, n_iter=20, cv=3).fit(X_train, y_train).best_

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", "RandomSearch", make_ttest(model.predict(X_test) == y_test))
```

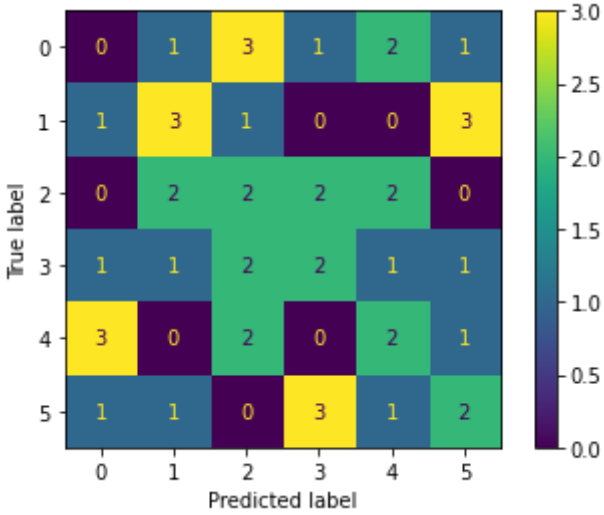
```
Train acc.: 1.000
Test acc.: 0.146
```



Fourth model: Neural Network - Multilayer Perceptron (MLP)

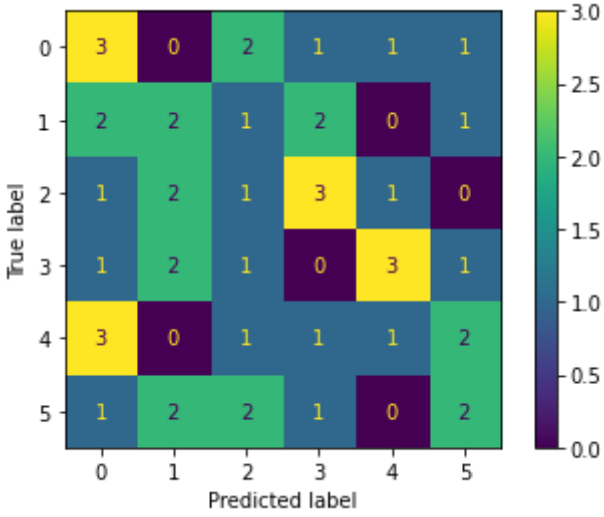
```
# without PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("MLP", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.229
```




```
# with PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("MLP", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.188
```



Fifth Analysis: V2 and original images

```
# opening right files
seed = 0
user = 'S01'
area = 'V2'
```

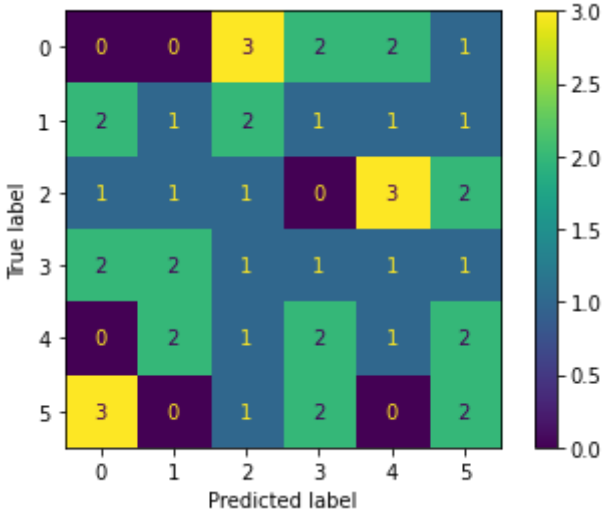
```
area = 'v2'
image_type = 'original'

df = pd.read_csv(f'fMRI_Scenes/{user}_{area}.csv',sep=r',',skipinitialspace = True, ir
df = df.loc[image_type]
df = df.sample(frac=1)
n_classes = len(df.category.unique())

# splitting dataset
np.random.seed(seed)
test_run = np.random.choice(df.run.unique(), 1).tolist()

df_train = df.loc[~df.run.isin(test_run)]
df_test = df.loc[df.run.isin(test_run)]

print('Train size:', len(df_train))
print('Test size:', len(df_test))

# selecting only variables of interest and transforming data
X_train, y_train = df_train.loc[:, [x for x in df_train.columns if x.startswith('vox')
X_test, y_test = df_test.loc[:, [x for x in df_test.columns if x.startswith('vox')]].t

encoder = LabelEncoder().fit(y_train)
y_train, y_test = encoder.transform(y_train), encoder.transform(y_test)

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

pca = PCA(n_components=60).fit(X_train)
X_train_pca, X_test_pca = pca.transform(X_train), pca.transform(X_test)
print('PCA explained variance ratio: %.3f' % pca.explained_variance_ratio_.sum())
```

```
Train size: 288
Test size: 48
PCA explained variance ratio: 0.606
```

First model: Logistic Regression

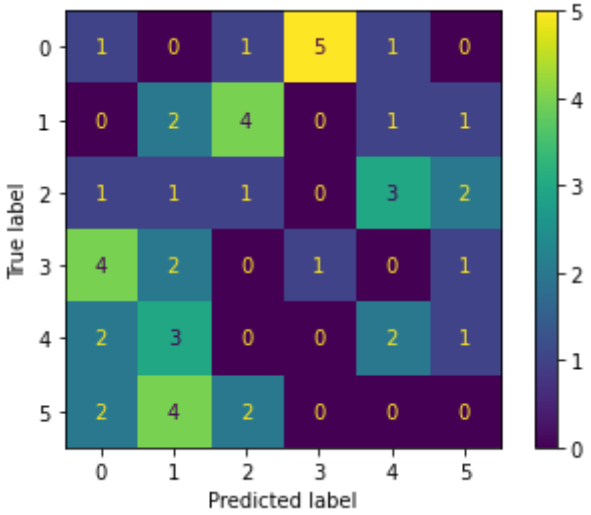
```
# without PCA
model = LogisticRegression().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))


show_confusion_matrix(model, X_test, y_test)
append_summary("LogisticRegression", None, make_ttest(model.predict(X_test) == y_test)
```

```
Train acc.: 1.000
Test acc.: 0.125
```




```
# with PCA
model = LogisticRegression().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("LogisticRegression", "PCA", make_ttest(model.predict(X_test_pca) == y_
```

```
Train acc.: 0.854
Test acc.: 0.146
```

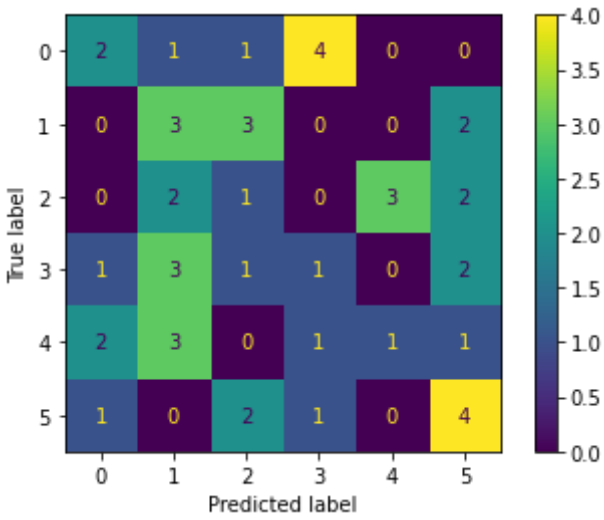


Second model: Support Vector Machine (SVM)

```
# without PCA
model = SVC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))
```

```
show_confusion_matrix(model, X_test, y_test)
append_summary("SVM", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.250
```



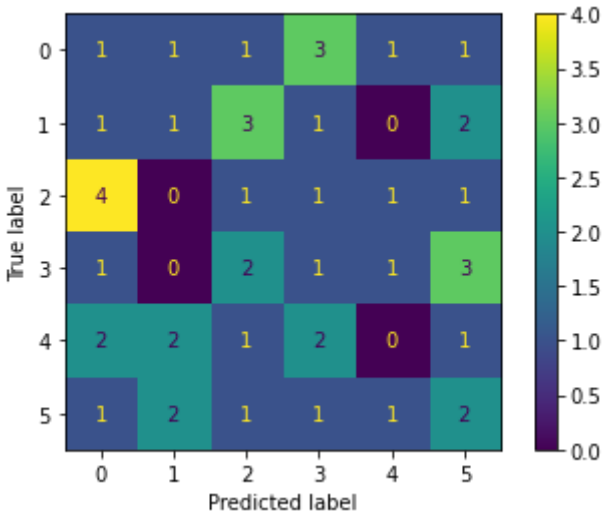

```
# with PCA
model = SVC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("SVM", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 0.986
Test acc.: 0.250
```

Third model: Random Forest Classifier (RFC)

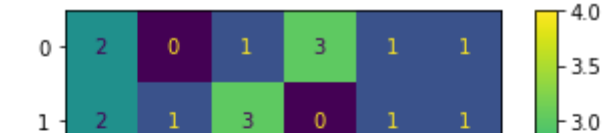
```
# without PCA
model = RFC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", None, make_ttest(model.predict(X_test) == y_test))
```

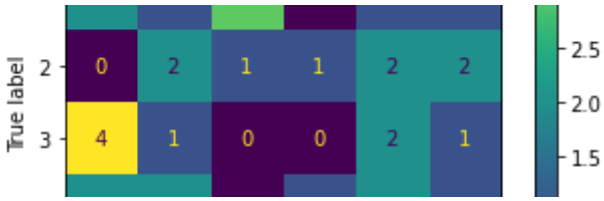
```
Train acc.: 1.000
Test acc.: 0.125
```




```
# with PCA
model = RFC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("RFC", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.125
```

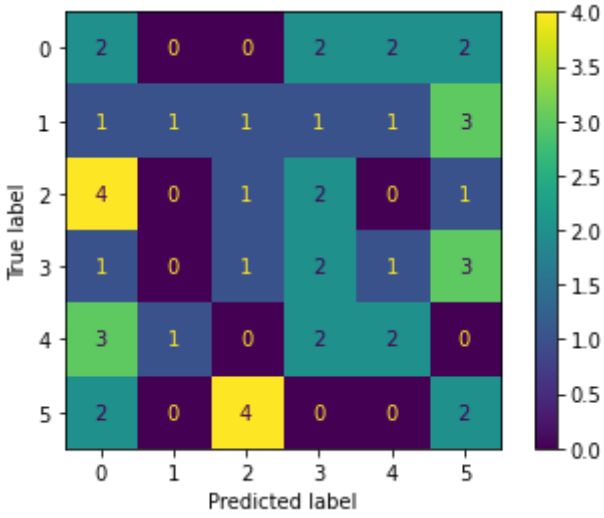




Random Search


```
# without PCA + RandomSearch
from sklearn.model_selection import RandomizedSearchCV

params = {'bootstrap': [True, False],
 'max_depth': [10, 30, 60, 90, None],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 800, 1000, 2000]}

model = RandomizedSearchCV(RFC(), params, n_iter=20, cv=3).fit(X_train, y_train).best_

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", "RandomSearch", make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.208
```

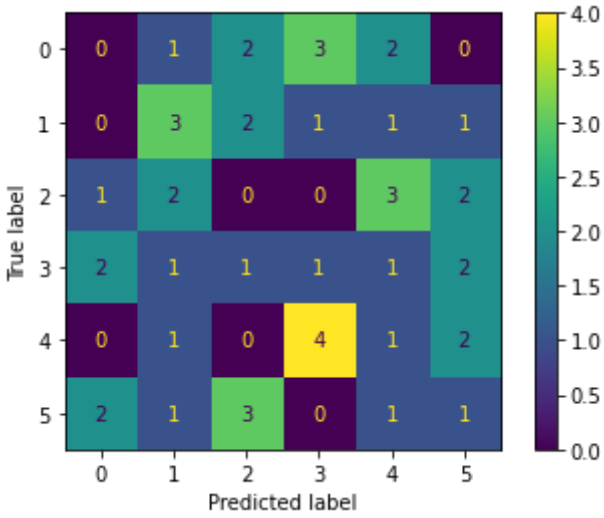


Fourth model: Neural Network - Multilayer Perceptron (MLP)

```
# without PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
```

```
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("MLP", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.125
```




```
# with PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("MLP", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.167
```

- Sixth Analysis: V2 and line drawings

Sixth Analysis: V2 and line drawings

```
# opening right files
seed = 0
user = 'S01'
area = 'V2'
image_type = 'lineDrawings'

df = pd.read_csv(f'fMRI_Scenes/{user}_{area}.csv',sep=r',',skipinitialspace = True, ir
df = df.loc[image_type]
df = df.sample(frac=1)
n_classes = len(df.category.unique())

# splitting dataset
np.random.seed(seed)
test_run = np.random.choice(df.run.unique(), 1).tolist()

df_train = df.loc[~df.run.isin(test_run)]
df_test = df.loc[df.run.isin(test_run)]

print('Train size:', len(df_train))
print('Test size:', len(df_test))

# selecting only variables of interest and transforming data
X_train, y_train = df_train.loc[:, [x for x in df_train.columns if x.startswith('vox')
X_test, y_test = df_test.loc[:, [x for x in df_test.columns if x.startswith('vox')]].t

encoder = LabelEncoder().fit(y_train)
y_train, y_test = encoder.transform(y_train), encoder.transform(y_test)

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

pca = PCA(n_components=60).fit(X_train)
X_train_pca, X_test_pca = pca.transform(X_train), pca.transform(X_test)
print('PCA explained variance ratio: %.3f' % pca.explained_variance_ratio_.sum())
```

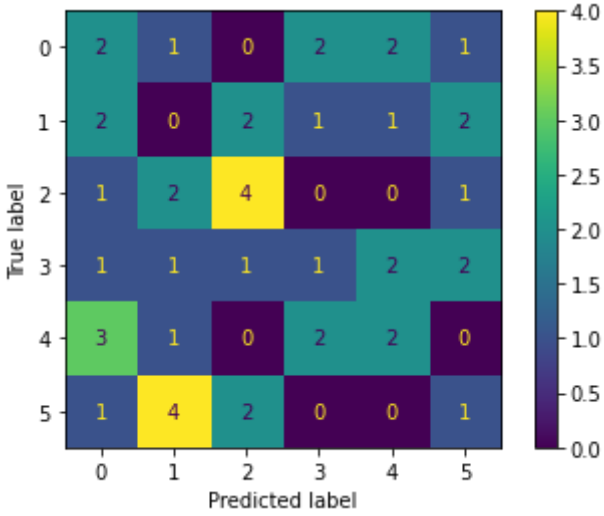
```
Train size: 288
Test size: 48
PCA explained variance ratio: 0.611
```

First model: Logistic Regression

```
# without PCA
model = LogisticRegression().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))
```

```
show_confusion_matrix(model, X_test, y_test)
append_summary("LogisticRegression", None, make_ttest(model.predict(X_test) == y_test)
```

```
Train acc.: 1.000
Test acc.: 0.208
```



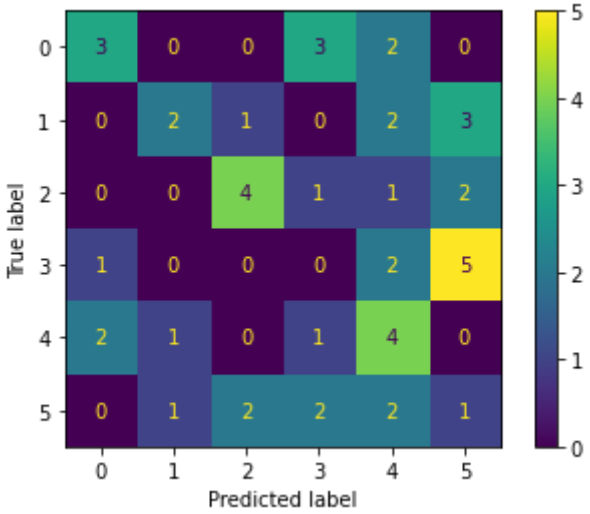

```
# with PCA
model = LogisticRegression().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("LogisticRegression", "PCA", make_ttest(model.predict(X_test_pca) == y_
```

```
Train acc.: 0.792
Test acc.: 0.188
```

Second model: Support Vector Machine (SVM)

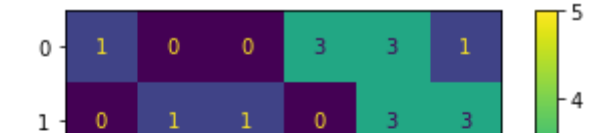
```
# without PCA
model = SVC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("SVM", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.292
```




```
# with PCA
model = SVC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("SVM", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

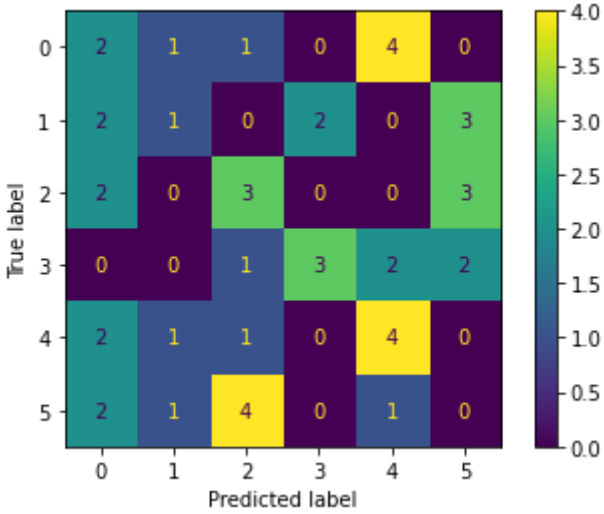
```
Train acc.: 0.972
Test acc.: 0.229
```



Third model: Random Forest Classifier (RFC)

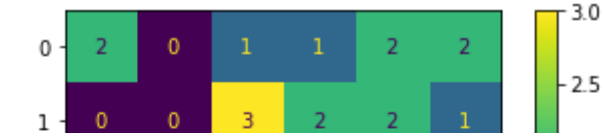
```
# without PCA
model = RFC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", None, make_ttest(model.predict(X_test) == y_test))
```

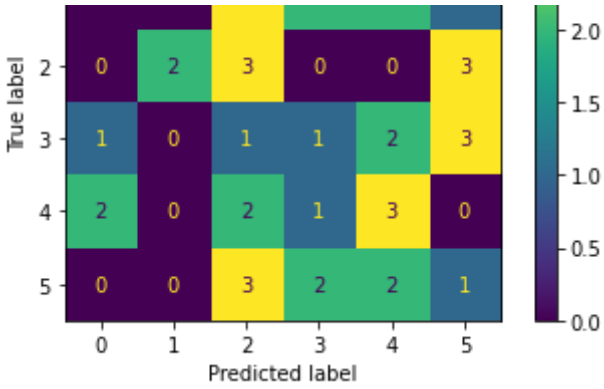
```
Train acc.: 1.000
Test acc.: 0.271
```




```
# with PCA
model = RFC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("RFC", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.208
```





Random Search

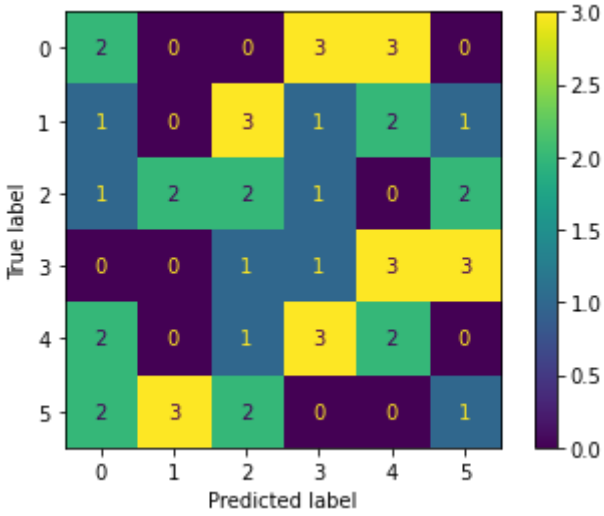
```
# without PCA + RandomSearch
from sklearn.model_selection import RandomizedSearchCV

params = {'bootstrap': [True, False],
 'max_depth': [10, 30, 60, 90, None],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 800, 1000, 2000]}

model = RandomizedSearchCV(RFC(), params, n_iter=20, cv=3).fit(X_train, y_train).best_

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", "RandomSearch", make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.292
```

Fourth model: Neural Network - Multilayer Perceptron (MLP)

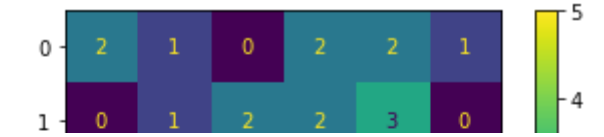
```
# without PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("MLP", None, make_ttest(model.predict(X_test) == y_test))
```

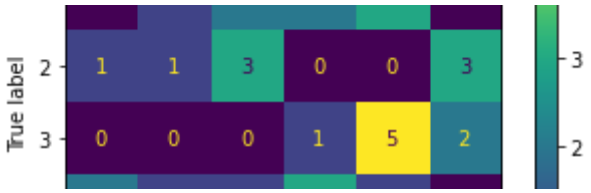
```
Train acc.: 1.000
Test acc.: 0.167
```




```
# with PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("MLP", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.188
```





Seventh Analysis: V4 and original images

```
          0    1    2    3    4    5
```

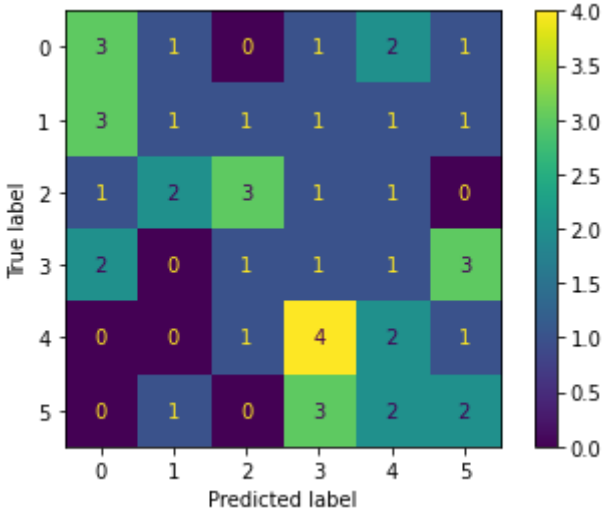
```
# opening right files
seed = 0
user = 'S01'
area = 'V4'
image_type = 'original'

df = pd.read_csv(f'fMRI_Scenes/{user}_{area}.csv',sep=r',',skipinitialspace = True, ir
df = df.loc[image_type]
df = df.sample(frac=1)
n_classes = len(df.category.unique())

# splitting dataset
np.random.seed(seed)
test_run = np.random.choice(df.run.unique(), 1).tolist()

df_train = df.loc[~df.run.isin(test_run)]
df_test = df.loc[df.run.isin(test_run)]

print('Train size:', len(df_train))
print('Test size:', len(df_test))

# selecting only variables of interest and transforming data
X_train, y_train = df_train.loc[:, [x for x in df_train.columns if x.startswith('vox')
X_test, y_test = df_test.loc[:, [x for x in df_test.columns if x.startswith('vox')]].t

encoder = LabelEncoder().fit(y_train)
y_train, y_test = encoder.transform(y_train), encoder.transform(y_test)

X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

pca = PCA(n_components=60).fit(X_train)
X_train_pca, X_test_pca = pca.transform(X_train), pca.transform(X_test)
print('PCA explained variance ratio: %.3f' % pca.explained_variance_ratio_.sum())
```

```
    Train size: 288
    Test size: 48
    PCA explained variance ratio: 0.671
```

First model: Logistic Regression

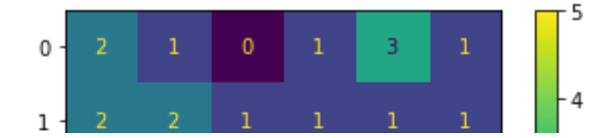
```
# without PCA
model = LogisticRegression().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("LogisticRegression", None, make_ttest(model.predict(X_test) == y_test)
```

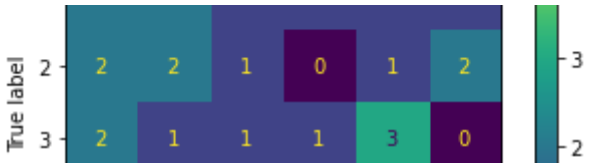
```
Train acc.: 1.000
Test acc.: 0.250
```




```
# with PCA
model = LogisticRegression().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("LogisticRegression", "PCA", make_ttest(model.predict(X_test_pca) == y_
```

```
Train acc.: 0.809
Test acc.: 0.208
```

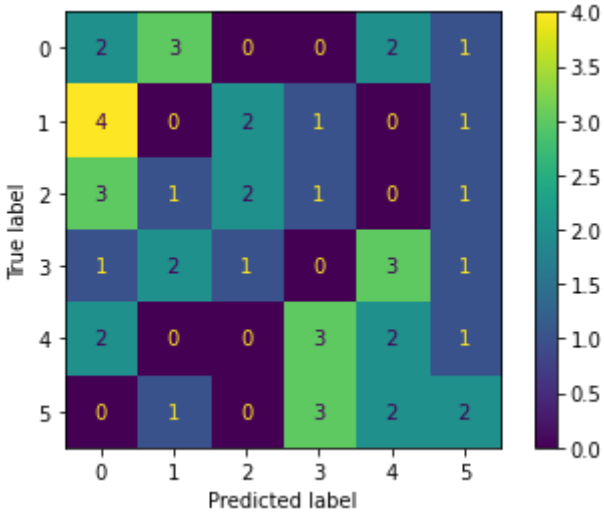




Second model: Support Vector Machine (SVM)

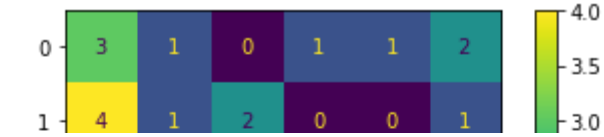

```
# without PCA
model = SVC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("SVM", None, make_ttest(model.predict(X_test) == y_test))
```

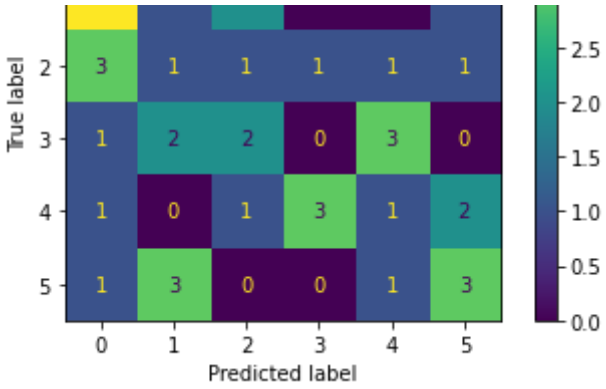
```
Train acc.: 1.000
Test acc.: 0.167
```




```
# with PCA
model = SVC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("SVM", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 0.993
Test acc.: 0.188
```

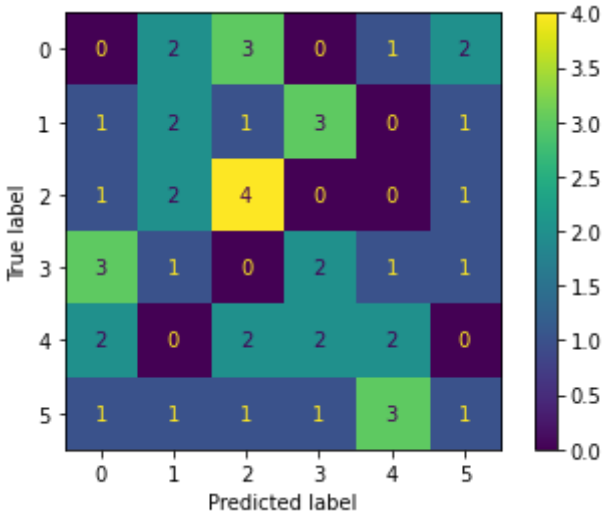




Third model: Random Forest Classifier (RFC)

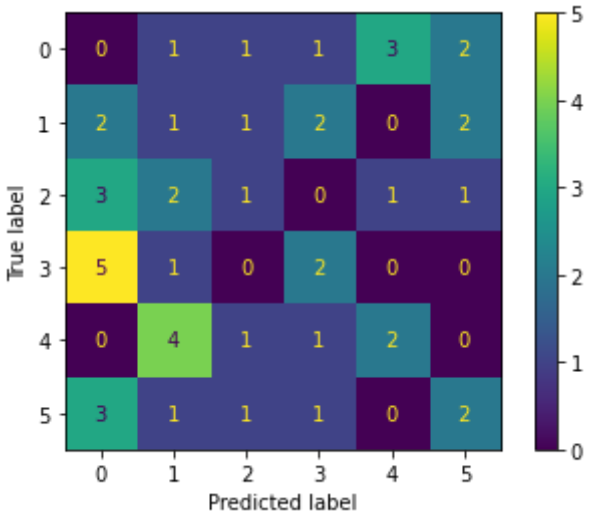
```
# without PCA
model = RFC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.229
```




```
# with PCA

model = RFC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("RFC", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

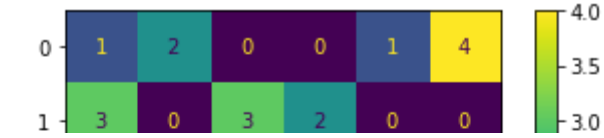
```
Train acc.: 1.000
Test acc.: 0.167
```

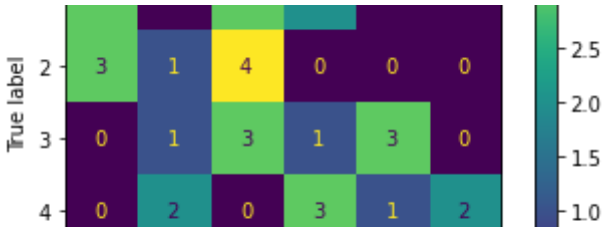


Random Search

```
# without PCA + RandomSearch
from sklearn.model_selection import RandomizedSearchCV

params = {'bootstrap': [True, False],
 'max_depth': [10, 30, 60, 90, None],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 800, 1000, 2000]}

model = RandomizedSearchCV(RFC(), params, n_iter=20, cv=3).fit(X_train, y_train).best_

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", "RandomSearch", make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.188
```



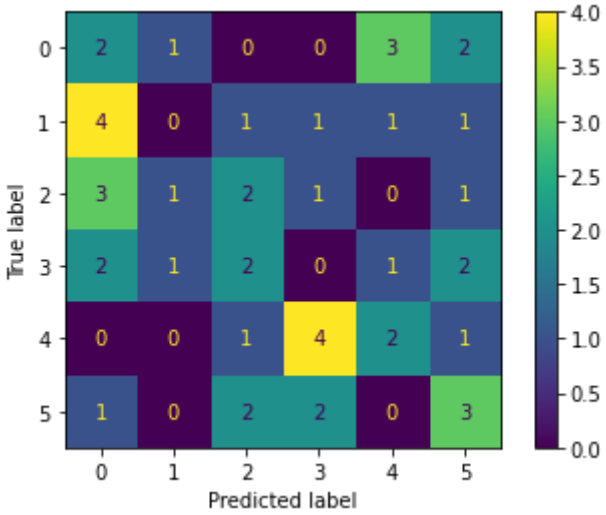


Fourth model: Neural Network - Multilayer Perceptron (MLP)

0 1 2 3 4 5

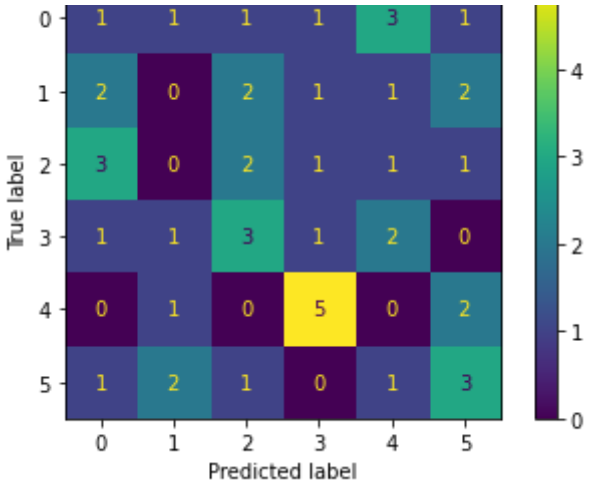
```
# without PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("MLP", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.188
```




```
# with PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))



show_confusion_matrix(model, X_test_pca, y_test)
append_summary("MLP", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.146
```



Results for V4 (original)

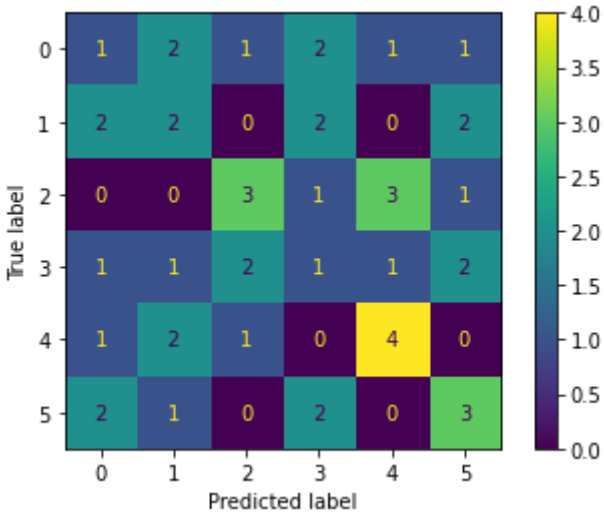
```
# opening right files
seed = 0
user = 'S01'
area = 'V4'
image_type = 'original'

df = pd.read_csv(f'fMRI_Scenes/{user}_{area}.csv',sep=r',',skipinitialspace = True, ir
df = df.loc[image_type]
df = df.sample(frac=1)
n_classes = len(df.category.unique())

# splitting dataset
np.random.seed(seed)
test_run = np.random.choice(df.run.unique(), 1).tolist()

df_train = df.loc[~df.run.isin(test_run)]
df_test = df.loc[df.run.isin(test_run)]

print('Train size:', len(df_train))
print('Test size:', len(df_test))

# selecting only variables of interest and transforming data
X_train, y_train = df_train.loc[:, [x for x in df_train.columns if x.startswith('vox')
X_test, y_test = df_test.loc[:, [x for x in df_test.columns if x.startswith('vox')]].t

encoder = LabelEncoder().fit(y_train)
y_train, y_test = encoder.transform(y_train), encoder.transform(y_test)


X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

pca = PCA(n_components=60).fit(X_train)
X_train_pca, X_test_pca = pca.transform(X_train), pca.transform(X_test)
print('PCA explained variance ratio: %.3f' % pca.explained_variance_ratio_.sum())
```

```
Train size: 288
Test size: 48
PCA explained variance ratio: 0.667
```

First model: Logistic Regression

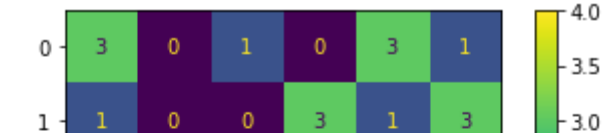
```
# without PCA
model = LogisticRegression().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("LogisticRegression", None, make_ttest(model.predict(X_test) == y_test)
```

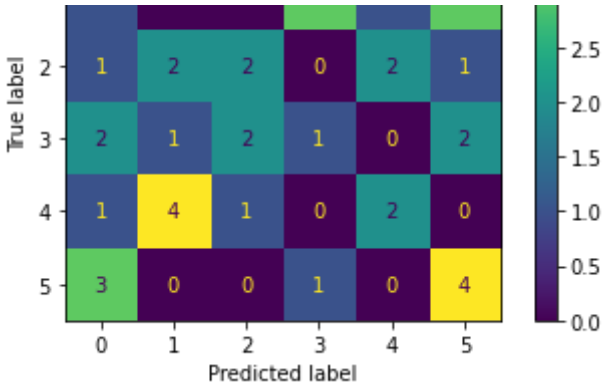
```
Train acc.: 1.000
Test acc.: 0.292
```




```
# with PCA
model = LogisticRegression().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("LogisticRegression", "PCA", make_ttest(model.predict(X_test_pca) == y_
```

```
Train acc.: 0.757
Test acc.: 0.250
```

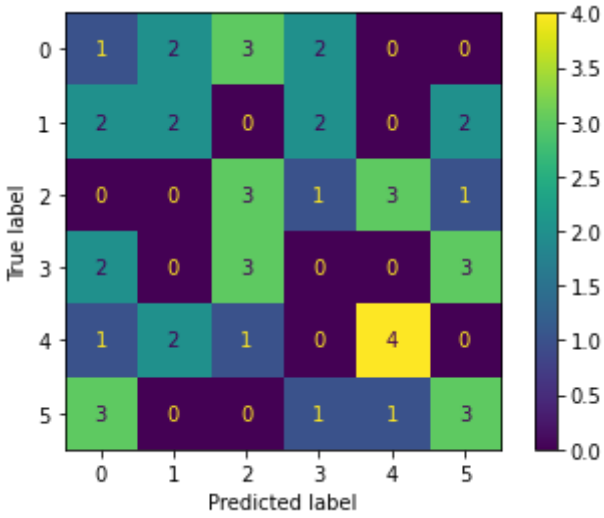




Second model: Support Vector Machine (SVM)

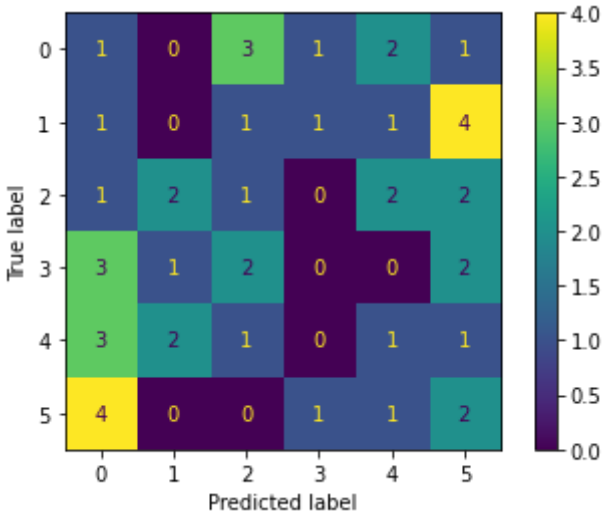
```
# without PCA
model = SVC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("SVM", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.271
```




```
# with PCA
model = SVC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("SVM", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.104
```



Third model: Random Forest Classifier (RFC)

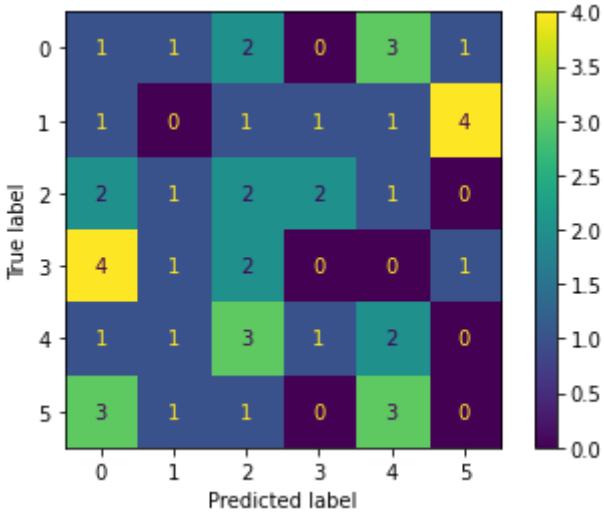
```
# without PCA
model = RFC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.271
```


```
# with PCA
model = RFC().fit(X_train_pca, y_train)
```

```
print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("RFC", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

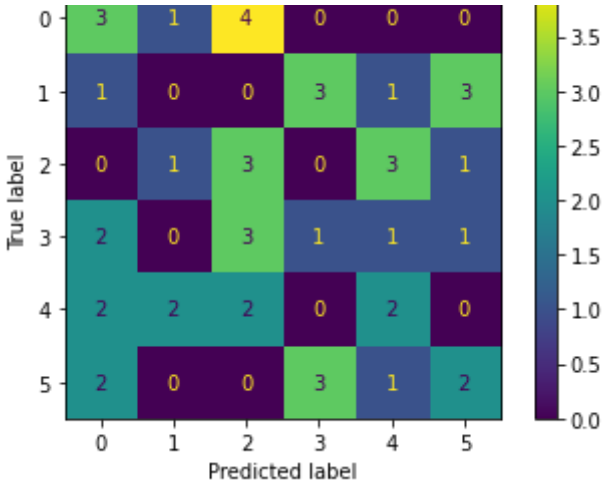
```
Train acc.: 1.000
Test acc.: 0.104
```



Random Search

```
# without PCA + RandomSearch
from sklearn.model_selection import RandomizedSearchCV

params = {'bootstrap': [True, False],
 'max_depth': [10, 30, 60, 90, None],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 800, 1000, 2000]}

model = RandomizedSearchCV(RFC(), params, n_iter=20, cv=3).fit(X_train, y_train).best_

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))



show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", "RandomSearch", make_ttest(model.predict(X_test) == y_test))
```

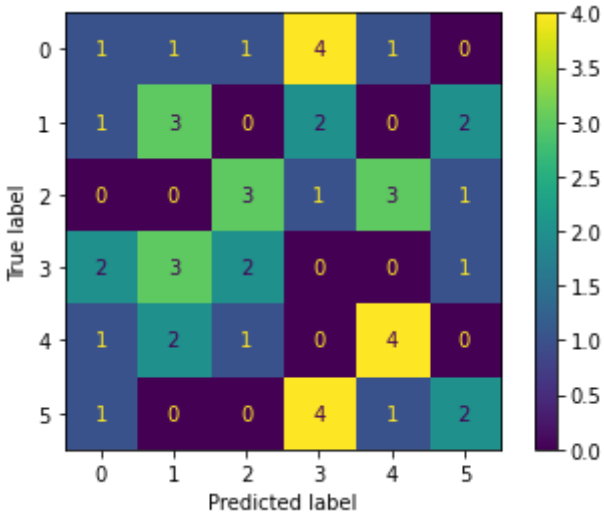
```
Train acc.: 1.000
Test acc.: 0.229
```



Fourth model: Neural Network - Multilayer Perceptron (MLP)

```
# without PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("MLP", None, make_ttest(model.predict(X_test) == y_test))
```

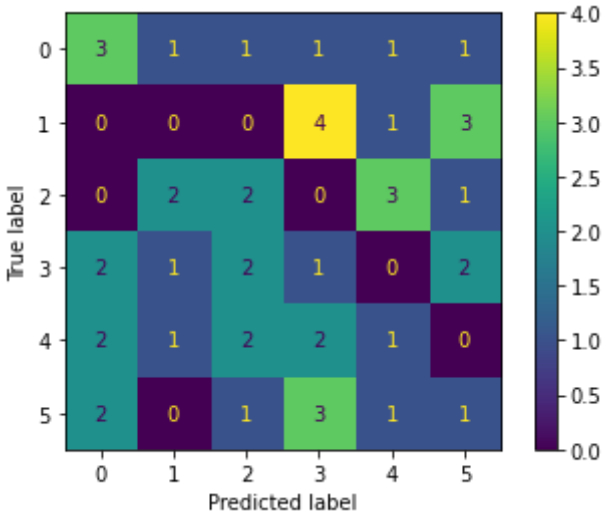
```
Train acc.: 1.000
Test acc.: 0.271
```




```
# with PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
```

```
append_summary("MLP", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.167
```



Eighth Analysis: V4 and line drawings

```
# opening right files
seed = 0
user = 'S01'
area = 'V4'
image_type = 'lineDrawings'

df = pd.read_csv(f'fMRI_Scenes/{user}_{area}.csv',sep=r',',skipinitialspace = True, ir
df = df.loc[image_type]
df = df.sample(frac=1)
n_classes = len(df.category.unique())

# splitting dataset
np.random.seed(seed)
test_run = np.random.choice(df.run.unique(), 1).tolist()

df_train = df.loc[~df.run.isin(test_run)]
df_test = df.loc[df.run.isin(test_run)]

print('Train size:', len(df_train))
print('Test size:', len(df_test))


# selecting only variables of interest and transforming data
X_train, y_train = df_train.loc[:, [x for x in df_train.columns if x.startswith('vox')
X_test, y_test = df_test.loc[:, [x for x in df_test.columns if x.startswith('vox')]].t

encoder = LabelEncoder().fit(y_train)
y train, y test = encoder.transform(y train), encoder.transform(y test)
```

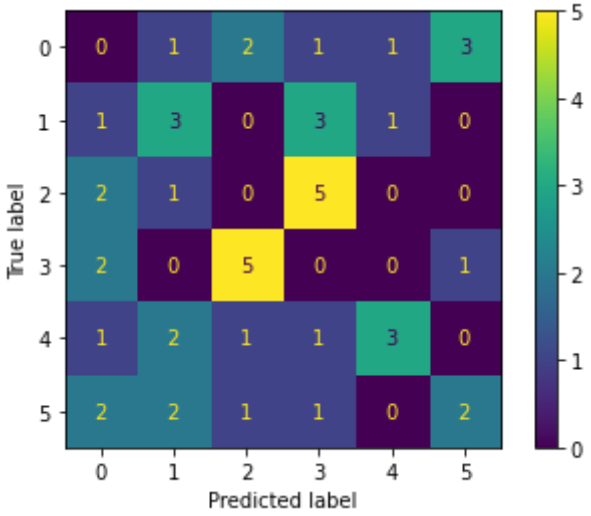
```
X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)

pca = PCA(n_components=60).fit(X_train)
X_train_pca, X_test_pca = pca.transform(X_train), pca.transform(X_test)
print('PCA explained variance ratio: %.3f' % pca.explained_variance_ratio_.sum())
```

```
Train size: 288
Test size: 48
PCA explained variance ratio: 0.673
```

First model: Logistic Regression

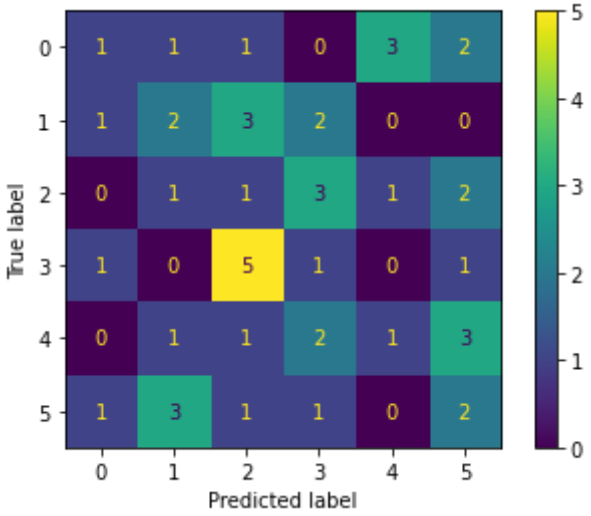
```
# without PCA
model = LogisticRegression().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("LogisticRegression", None, make_ttest(model.predict(X_test) == y_test)
```

```
Train acc.: 1.000
Test acc.: 0.167
```




```
# with PCA

model = LogisticRegression().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("LogisticRegression", "PCA", make_ttest(model.predict(X_test_pca) == y
```

```
Train acc.: 0.767
Test acc.: 0.167
```



Second model: Support Vector Machine (SVM)

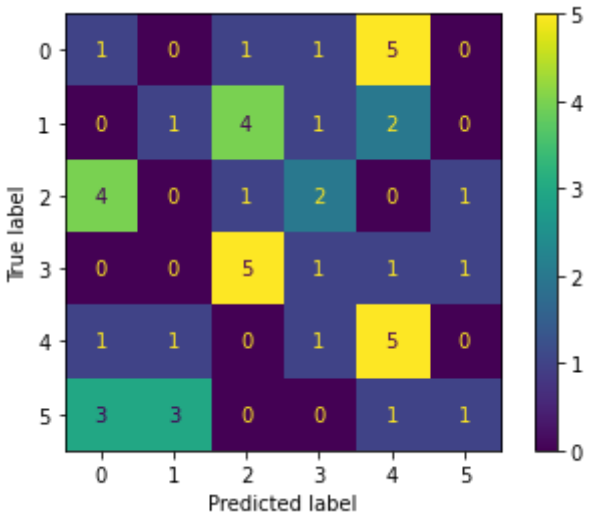
```
# without PCA
model = SVC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("SVM", None, make_ttest(model.predict(X_test) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.167
```

```
# with PCA
model = SVC().fit(X_train_pca, y_train)
```

```
print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("SVM", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

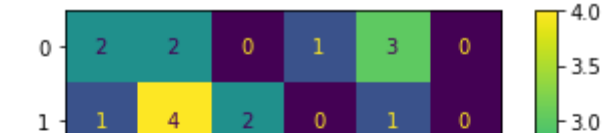
```
Train acc.: 0.983
Test acc.: 0.208
```

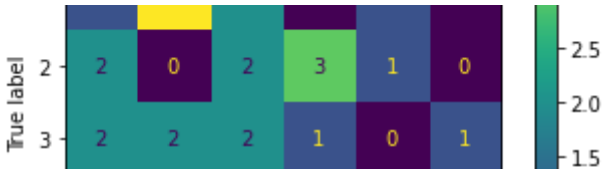


Third model: Random Forest Classifier (RFC)

```
# without PCA
model = RFC().fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("RFC", None, make_ttest(model.predict(X_test) == y_test))
```

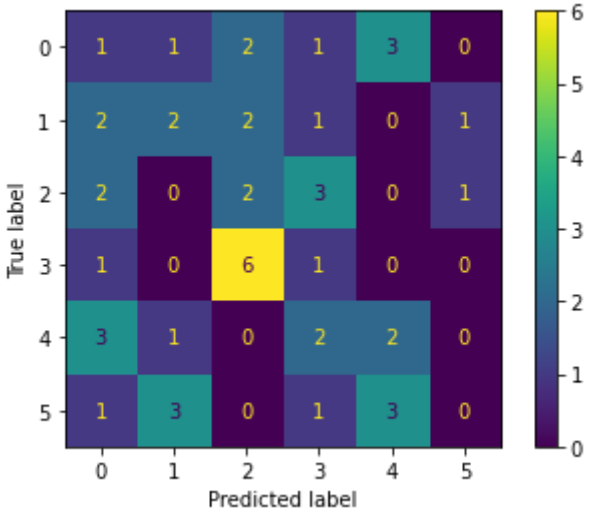
```
Train acc.: 1.000
Test acc.: 0.312
```






```
# with PCA
model = RFC().fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("RFC", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.167
```

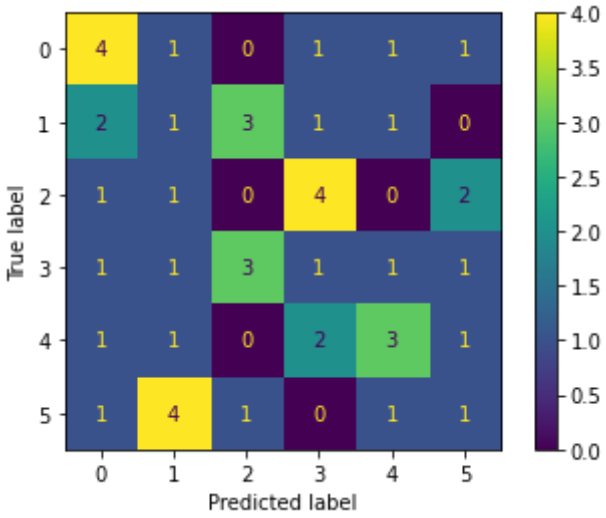


Random Search

```
# without PCA + RandomSearch
from sklearn.model_selection import RandomizedSearchCV

params = {'bootstrap': [True, False],
 'max_depth': [10, 30, 60, 90, None],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 800, 1000, 2000]}


model = RandomizedSearchCV(RFC(), params, n_iter=20, cv=3).fit(X_train, y_train).best_

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
```

```
append_summary("RFC", "RandomSearch", make_ttest(model.predict(X_test) == y_test))
```

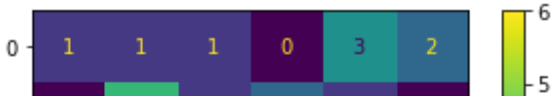
```
Train acc.: 1.000
Test acc.: 0.208
```



Fourth model: Neural Network - Multilayer Perceptron (MLP)

```
# without PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train, y_train)

print('Train acc.: %.3f'% model.score(X_train, y_train))
print('Test acc.: %.3f\n'%model.score(X_test, y_test))

show_confusion_matrix(model, X_test, y_test)
append_summary("MLP", None, make_ttest(model.predict(X_test) == y_test))
```

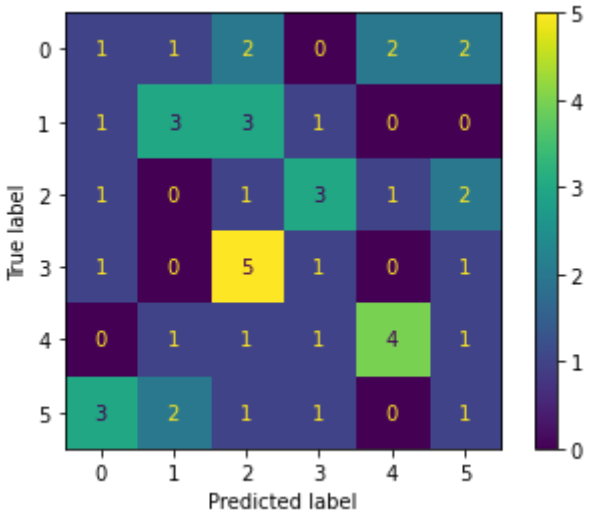
```
Train acc.: 1.000
Test acc.: 0.229
```




```
# with PCA
```

```
# With PCA
model = MLPClassifier(learning_rate_init=1e-2).fit(X_train_pca, y_train)

print('Train acc.: %.3f'% model.score(X_train_pca, y_train))
print('Test acc.: %.3f\n'%model.score(X_test_pca, y_test))

show_confusion_matrix(model, X_test_pca, y_test)
append_summary("MLP", "PCA", make_ttest(model.predict(X_test_pca) == y_test))
```

```
Train acc.: 1.000
Test acc.: 0.229
```



Results for V4 (line drawings)

For RSC, the best classifier was random forests with no data preprocessing. That was true for both original images and line drawings. For V1, the best classifier was SVM with PCA preprocessing, but only for original images. For V2, none of the methods performed significantly better than chance. For V4, none of the methods performed significantly better than chance.

```
pd.set_option("display.max_rows", 90)
pd.DataFrame(summary)
```

	area	image_type	method	strategy	ttest	pvalue
0	RSC	original	LogisticRegression	None	1.606990	0.057377
1	RSC	original	LogisticRegression	PCA	1.606990	0.057377
2	RSC	original	SVM	None	1.319371	0.096718

3	RSC	original	SVM	PCA	1.606990	0.057377
4	RSC	original	RFC	None	0.365928	0.358030
5	RSC	original	RFC	PCA	0.365928	0.358030
6	RSC	original	RFC	RandomSearch	1.885370	0.032785
7	RSC	original	MLP	None	1.606990	0.057377
8	RSC	original	MLP	PCA	1.319371	0.096718
9	RSC	lineDrawings	LogisticRegression	None	0.365928	0.358030
10	RSC	lineDrawings	LogisticRegression	PCA	0.703375	0.242646
11	RSC	lineDrawings	SVM	None	1.019467	0.156600
12	RSC	lineDrawings	LogisticRegression	PCA	0.703375	0.242646
13	RSC	lineDrawings	RFC	None	1.885370	0.032785
14	RSC	lineDrawings	RFC	PCA	1.606990	0.057377
15	RSC	lineDrawings	RFC	RandomSearch	2.156971	0.018077
16	RSC	lineDrawings	MLP	None	1.885370	0.032785
17	RSC	lineDrawings	MLP	PCA	0.365928	0.358030
18	V1	original	LogisticRegression	None	-0.863731	0.803939
19	V1	original	LogisticRegression	PCA	1.319371	0.096718
20	V1	original	SVM	None	0.000000	0.500000
21	V1	original	SVM	PCA	0.000000	0.500000
22	V1	original	RFC	None	-1.402655	0.916355
23	V1	original	RFC	PCA	0.000000	0.500000
24	V1	original	RFC	RandomSearch	-0.404676	0.656224
25	V1	original	MLP	None	0.703375	0.242646
26	V1	original	MLP	PCA	0.703375	0.242646
27	V1	lineDrawings	LogisticRegression	None	-0.863731	0.803939
28	V1	lineDrawings	LogisticRegression	PCA	-0.404676	0.656224
29	V1	lineDrawings	SVM	None	-0.404676	0.656224
30	V1	lineDrawings	SVM	PCA	-0.863731	0.803939
31	V1	lineDrawings	RFC	None	0.703375	0.242646
32	V1	lineDrawings	RFC	PCA	1.319371	0.096718
33	V1	lineDrawings	RFC	RandomSearch	-0.404676	0.656224
34	V1	lineDrawings	MLP	None	1.019467	0.156600

35	V1	lineDrawings	MLP	PCA	0.365928	0.358030
36	V2	original	LogisticRegression	None	-0.863731	0.803939
37	V2	original	LogisticRegression	PCA	-0.404676	0.656224
38	V2	original	SVM	None	1.319371	0.096718
39	V2	original	SVM	PCA	1.319371	0.096718
40	V2	original	RFC	None	-0.863731	0.803939
41	V2	original	RFC	PCA	-0.863731	0.803939
42	V2	original	RFC	RandomSearch	0.703375	0.242646
43	V2	original	MLP	None	-0.863731	0.803939
44	V2	original	MLP	PCA	0.000000	0.500000
45	V2	lineDrawings	LogisticRegression	None	0.703375	0.242646
46	V2	lineDrawings	LogisticRegression	PCA	0.365928	0.358030
47	V2	lineDrawings	SVM	None	1.885370	0.032785
48	V2	lineDrawings	SVM	PCA	1.019467	0.156600
49	V2	lineDrawings	RFC	None	1.606990	0.057377
50	V2	lineDrawings	RFC	PCA	0.703375	0.242646
51	V2	lineDrawings	RFC	RandomSearch	1.885370	0.032785
52	V2	lineDrawings	MLP	None	0.000000	0.500000
53	V2	lineDrawings	MLP	PCA	0.365928	0.358030
54	V4	original	LogisticRegression	None	1.319371	0.096718
55	V4	original	LogisticRegression	PCA	0.703375	0.242646
56	V4	original	SVM	None	0.000000	0.500000
57	V4	original	SVM	PCA	0.365928	0.358030
58	V4	original	RFC	None	1.019467	0.156600
59	V4	original	RFC	PCA	0.000000	0.500000
60	V4	original	RFC	RandomSearch	0.365928	0.358030
61	V4	original	MLP	None	0.365928	0.358030
62	V4	original	MLP	PCA	-0.404676	0.656224
63	V4	original	LogisticRegression	None	1.885370	0.032785
64	V4	original	LogisticRegression	PCA	1.319371	0.096718
65	V4	original	SVM	None	1.606990	0.057377

66	V4	original	SVM	PCA	-1.402655	0.916355
67	V4	original	RFC	None	1.606990	0.057377
68	V4	original	RFC	PCA	-1.402655	0.916355
69	V4	original	RFC	RandomSearch	1.019467	0.156600
70	V4	original	MLP	None	1.606990	0.057377
71	V4	original	MLP	PCA	0.000000	0.500000
72	V4	lineDrawings	LogisticRegression	None	0.000000	0.500000
73	V4	lineDrawings	LogisticRegression	PCA	0.000000	0.500000
74	V4	lineDrawings	SVM	None	0.000000	0.500000
75	V4	lineDrawings	SVM	PCA	0.703375	0.242646
76	V4	lineDrawings	RFC	None	2.156971	0.018077
77	V4	lineDrawings	RFC	PCA	0.000000	0.500000
78	V4	lineDrawings	RFC	RandomSearch	0.703375	0.242646
79	V4	lineDrawings	MLP	None	1.019467	0.156600
80	V4	lineDrawings	MLP	PCA	1.019467	0.156600

34m 46s completed at 1:50 PM