

transfer-3-2

September 21, 2021

```
[1]: base_transfer_set = ['01', '02', '04', '05', '08', '09', '12', '13', '16',  
    ↪ '17', '18', '20']  
target_transfer_set = ['03', '06', '07', '10', '11', '14', '15', '19']  
  
import random  
def random_combination(iterable, r):  
    "Random selection from itertools.combinations(iterable, r)"  
    pool = tuple(iterable)  
    n = len(pool)  
    indices = sorted(random.sample(range(n), r))  
    return tuple(pool[i] for i in indices)  
  
transfers_size_6 = []  
for i in range(4):  
    transfers_size_6.append(random_combination(target_transfer_set, 6))  
print(transfers_size_6)  
transfers_size_6 = [('03', '06', '07', '10', '11', '14'), ('03', '06', '07',  
    ↪ '10', '14', '15'), ('03', '06', '07', '10', '14', '15'), ('03', '07', '10',  
    ↪ '14', '15', '19')]  
for i, tmp in enumerate(transfers_size_6):  
    transfers_size_6[i] = list(transfers_size_6[i])  
print(transfers_size_6)  
  
transfers_size_4 = []  
for i in range(4):  
    transfers_size_4.append(random_combination(target_transfer_set, 4))  
print(transfers_size_4)  
transfers_size_4 = [('06', '10', '14', '15'), ('03', '10', '14', '19'), ('03',  
    ↪ '06', '10', '15'), ('03', '07', '10', '15')]  
for i, tmp in enumerate(transfers_size_4):  
    transfers_size_4[i] = list(transfers_size_4[i])  
print(transfers_size_4)  
  
transfers_size_3 = []  
for i in range(4):  
    transfers_size_3.append(random_combination(target_transfer_set, 3))  
print(transfers_size_3)
```

```

transfers_size_3 = [('07', '11', '14'), ('06', '07', '10'), ('03', '15', '19'),
↳('06', '14', '19')]
for i, tmp in enumerate(transfers_size_3):
    transfers_size_3[i] = list(transfers_size_3[i])
print(transfers_size_3)

transfers_size_2 = []
for i in range(4):
    transfers_size_2.append(random_combination(target_transfer_set, 2))
print(transfers_size_2)
transfers_size_2 = [('06', '10'), ('07', '11'), ('06', '15'), ('14', '15')]
for i, tmp in enumerate(transfers_size_2):
    transfers_size_2[i] = list(transfers_size_2[i])
print(transfers_size_2)

```

```

[('03', '07', '10', '11', '14', '15'), ('03', '06', '07', '10', '15', '19'),
('03', '06', '07', '10', '11', '19'), ('06', '07', '10', '11', '15', '19')]
[['03', '06', '07', '10', '11', '14'], ['03', '06', '07', '10', '14', '15'],
['03', '06', '07', '10', '14', '15'], ['03', '07', '10', '14', '15', '19']]
[('06', '11', '14', '19'), ('03', '06', '07', '19'), ('06', '11', '14', '19'),
('07', '10', '11', '14')]
[['06', '10', '14', '15'], ['03', '10', '14', '19'], ['03', '06', '10', '15'],
['03', '07', '10', '15']]
[('06', '10', '11'), ('07', '11', '14'), ('03', '11', '14'), ('03', '06', '15')]
[['07', '11', '14'], ['06', '07', '10'], ['03', '15', '19'], ['06', '14', '19']]
[('10', '15'), ('14', '15'), ('07', '19'), ('11', '15')]
[['06', '10'], ['07', '11'], ['06', '15'], ['14', '15']]

```

```

[2]: import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

```

```

def make_confusion_matrix(cf,
                           group_names=None,
                           categories='auto',
                           count=True,
                           percent=True,
                           cbar=True,
                           xyticks=True,
                           xyplotlabels=True,
                           sum_stats=True,
                           figsize=None,
                           cmap='Blues',
                           title=None):
    """

```

This function will make a pretty plot of an sklearn Confusion Matrix cm_
↳*using a Seaborn heatmap visualization.*

Arguments

```
-----  
cf:          confusion matrix to be passed in  
group_names: List of strings that represent the labels row by row to be  
→shown in each square.  
categories:  List of strings containing the categories to be displayed on  
→the x,y axis. Default is 'auto'  
count:       If True, show the raw number in the confusion matrix.  
→Default is True.  
normalize:   If True, show the proportions for each category. Default is  
→True.  
cbar:        If True, show the color bar. The cbar values are based off  
→the values in the confusion matrix.  
             Default is True.  
xyticks:     If True, show x and y ticks. Default is True.  
xyplotlabels: If True, show 'True Label' and 'Predicted Label' on the  
→figure. Default is True.  
sum_stats:   If True, display summary statistics below the figure.  
→Default is True.  
figsize:     Tuple representing the figure size. Default will be the  
→matplotlib rcParams value.  
cmap:        Colormap of the values displayed from matplotlib.pyplot.cm.  
→Default is 'Blues'  
             See http://matplotlib.org/examples/color/colormaps\_reference.html  
→html  
  
title:       Title for the heatmap. Default is None.  
'''  
  
# CODE TO GENERATE TEXT INSIDE EACH SQUARE  
blanks = ['' for i in range(cf.size)]  
  
if group_names and len(group_names)==cf.size:  
    group_labels = ["{}\n".format(value) for value in group_names]  
else:  
    group_labels = blanks  
  
if count:  
    group_counts = ["{0:0.0f}\n".format(value) for value in cf.flatten()]  
else:  
    group_counts = blanks  
  
if percent:  
    group_percentages = ["{0:.2%}".format(value) for value in cf.flatten()/  
→np.sum(cf)]
```

```

else:
    group_percentages = blanks

    box_labels = [f"{v1}-{v2}-{v3}".strip() for v1, v2, v3 in
→zip(group_labels,group_counts,group_percentages)]
    box_labels = np.asarray(box_labels).reshape(cf.shape[0],cf.shape[1])

# CODE TO GENERATE SUMMARY STATISTICS & TEXT FOR SUMMARY STATS
if sum_stats:
    #Accuracy is sum of diagonal divided by total observations
    accuracy = np.trace(cf) / float(np.sum(cf))

    #if it is a binary confusion matrix, show some more stats
    if len(cf)==2:
        #Metrics for Binary Confusion Matrices
        precision = cf[1,1] / sum(cf[:,1])
        recall    = cf[1,1] / sum(cf[1,:])
        f1_score  = 2*precision*recall / (precision + recall)
        stats_text = "\n\nAccuracy={:0.3f}\nPrecision={:0.3f}\nRecall={:0.
→3f}\nF1 Score={:0.3f}".format(
            accuracy,precision,recall,f1_score)
    else:
        stats_text = "\n\nAccuracy={:0.3f}".format(accuracy)
else:
    stats_text = ""

# SET FIGURE PARAMETERS ACCORDING TO OTHER ARGUMENTS
if figsize==None:
    #Get default figure size if not set
    figsize = plt.rcParams.get('figure.figsize')

if xyticks==False:
    #Do not show categories if xyticks is False
    categories=False

# MAKE THE HEATMAP VISUALIZATION
plt.figure(figsize=figsize)
sns.
→heatmap(cf,annot=box_labels,fmt="",cmap=cmap,cbar=cbar,xticklabels=categories,yticklabels=c

if xyplotlabels:
    plt.ylabel('True label')
    plt.xlabel('Predicted label' + stats_text)
else:

```

```
plt.xlabel(stats_text)

if title:
    plt.title(title)
plt.show()
```

```
-----
RuntimeError                                Traceback (most recent call last)
RuntimeError: module compiled against API version 0xe but this version of numpy
↳ is 0xd
```

```
-----
RuntimeError                                Traceback (most recent call last)
RuntimeError: module compiled against API version 0xe but this version of numpy
↳ is 0xd
```

```
[3]: import os
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

def create_transfer_models(baseset, gesture_subset):
    print("Processing tranfers models at 10%, 20%, 25%, 50% and 80% data for_
↳ gestures: ", gesture_subset)
    print("Baseset: ", baseset)
    print("Loadind Dataset: ", gesture_subset)
    path = 'gestures-dataset'
    dataset = None

    samples = 0
    for subject in os.listdir(path):
        if os.path.isfile(os.path.join(path, subject)):
            continue
        if subject in ('U01', 'U02', 'U03', 'U04', 'U05', 'U06', 'U07', 'U08'):
            for gesture in os.listdir(os.path.join(path, subject)):
                if os.path.isfile(os.path.join(path, subject, gesture)):
                    continue
                gesture = str(gesture)
                if gesture not in gesture_subset:
                    continue
                for samplefile in os.listdir(os.path.join(path, subject,
↳ gesture)):
```

```

        if os.path.isfile(os.path.join(path, subject, gesture,
↪samplefile)):
            df = pd.read_csv(os.path.join(path, subject, gesture,
↪samplefile), \
                            sep = ' ', \
                            names = ['System.currentTimeMillis()', \
                                    'System.nanoTime()', \
                                    'sample.timestamp', \
                                    'X', \
                                    'Y', \
                                    'Z' \
                                    ])
            df = df[["sample.timestamp", "X", "Y", "Z"]]

            start = df["sample.timestamp"][0]
            df["sample.timestamp"] -= start
            df["sample.timestamp"] /= 10000000
            df["subject"] = subject
            df["gesture"] = gesture
            df["sample"] = str(samplefile[:-4])
            samples += 1
            #print(df)
            if dataset is None:
                dataset = df.copy()
            else:
                dataset = pd.concat([dataset, df])

    dataset = dataset.sort_values(by=['gesture', 'subject', 'sample', 'sample.
↪timestamp'])
    data = dataset
    print(str(samples) + " samples loaded")

    print("Scaling Dataset: ", gesture_subset)
    from sklearn.preprocessing import StandardScaler

    scaler = StandardScaler()
    dataset_scaled = None

    samples = 0
    for i, gesture in enumerate(gesture_subset):
        df_gesture=data[data['gesture']==gesture]
        for j, subject in enumerate(df_gesture['subject'].unique()):
            df_subject=df_gesture[df_gesture['subject']==subject]
            for k, sample in enumerate(df_subject['sample'].unique()):
                df_sample=df_subject[df_subject['sample']==sample].copy()
                df_sample.sort_values(by=['sample.timestamp'])

```

```

        sc = scaler
        sc = sc.fit_transform(df_sample[["X", "Y", "Z"]])
        sc = pd.DataFrame(data=sc, columns=["X", "Y", "Z"])
        df_sample['X'] = sc['X']
        df_sample['Y'] = sc['Y']
        df_sample['Z'] = sc['Z']
        if dataset_scaled is None:
            dataset_scaled = df_sample.copy()
        else:
            dataset_scaled = pd.concat([dataset_scaled, df_sample])
        samples += 1
    print(str(samples) + " samples scaled")
    data = dataset_scaled

    print("Cleaning Dataset: ", gesture_subset)
    dataset_outliers = None
    dataset_cleaned = None

    samples = 0
    outliers = 0
    for i, gesture in enumerate(gesture_subset):
        df_gesture = data[data['gesture']==gesture]
        for j, subject in enumerate(df_gesture['subject'].unique()):
            df_subject = df_gesture[df_gesture['subject']==subject]

            time_mean = df_subject.groupby(["gesture", "subject", "sample"]).
→count().groupby(["gesture", "subject"]).agg({'sample.timestamp': ['mean']})
            time_std = df_subject.groupby(["gesture", "subject", "sample"]).
→count().groupby(["gesture", "subject"]).agg({'sample.timestamp': ['std']})
            time_max = time_mean['sample.timestamp'].iloc[0]['mean'] + 1.0 *
→time_std['sample.timestamp'].iloc[0]['std']
            time_min = time_mean['sample.timestamp'].iloc[0]['mean'] - 1.0 *
→time_std['sample.timestamp'].iloc[0]['std']
            for k, sample in enumerate(df_subject['sample'].unique()):
                df_sample=df_subject[df_subject['sample']==sample]
                df_sample_count = df_sample.count()['sample.timestamp']
                if df_sample_count < time_min or df_sample_count > time_max:
                    if dataset_outliers is None:
                        dataset_outliers = df_sample.copy()
                    else:
                        dataset_outliers = pd.concat([dataset_outliers,
→df_sample])

                outliers += 1
            else:
                if dataset_cleaned is None:
                    dataset_cleaned = df_sample.copy()
                else:

```

```

dataset_cleaned = pd.concat([dataset_cleaned,
↳df_sample])

        samples += 1
print(str(samples) + " samples cleaned")
print(str(outliers) + " samples outliers")
data = dataset_cleaned

print("Time slicing Cleaned Dataset: ", gesture_subset)
dataset_timecut = None
samples = 0
damaged = 0
for i, gesture in enumerate(data['gesture'].unique()):
    df_gesture = data[data['gesture']==gesture]
    for j, subject in enumerate(df_gesture['subject'].unique()):
        df_subject = df_gesture[df_gesture['subject']==subject]
        time_max = 19 # 18 * 11 = 198
        for i, sample in enumerate(df_subject['sample'].unique()):
            df_sample = df_subject[df_subject['sample']==sample]
            df_sample_count = df_sample.count()['sample.timestamp']
            #print(df_sample_count)
            if df_sample_count >= time_max:
                df_sample = df_sample[df_sample['sample.timestamp'] <= (11
↳* (time_max-1))]
                df_sample_count = df_sample.count()['sample.timestamp']
                #print(df_sample_count)
            elif df_sample_count < time_max:
                for tmp in range(df_sample_count * 11, (time_max) * 11, 11):
                    df = pd.DataFrame([[tmp, 0.0, 0.0, 0.0, gesture,
↳subject, sample]], columns=['sample.timestamp', 'X', 'Y', 'Z', 'gesture',
↳'subject', 'sample'])
                    df_sample = df_sample.append(df, ignore_index=True)
                    #print(df_sample)
                    df_sample_count = df_sample.count()['sample.timestamp']
                    #print(df_sample_count)
                    if df_sample_count != time_max:
                        damaged += 1
                        continue
                    if dataset_timecut is None:
                        dataset_timecut = df_sample.copy()
                    else:
                        dataset_timecut = pd.concat([dataset_timecut, df_sample])
                    samples += 1

dataset_cleaned = dataset_timecut
print(str(samples) + " cleaned samples sliced")
print(str(damaged) + " cleaned samples damaged")

```



```

data = dataset_outliers
print("Time slicing Outliers Dataset: ", gesture_subset)
dataset_timecut = None
samples = 0
damaged = 0
for i, gesture in enumerate(data['gesture'].unique()):
    df_gesture = data[data['gesture']==gesture]
    for j, subject in enumerate(df_gesture['subject'].unique()):
        df_subject = df_gesture[df_gesture['subject']==subject]
        time_max = 19 # 18 * 11 = 198
        for i, sample in enumerate(df_subject['sample'].unique()):
            df_sample = df_subject[df_subject['sample']==sample]
            df_sample_count = df_sample.count()['sample.timestamp']
            #print(df_sample_count)
            if df_sample_count >= time_max:
                df_sample = df_sample[df_sample['sample.timestamp'] <= (11_
↪* (time_max-1))]
                df_sample_count = df_sample.count()['sample.timestamp']
                #print(df_sample_count)
            elif df_sample_count < time_max:
                for tmp in range(df_sample_count * 11, (time_max) * 11, 11):
                    df = pd.DataFrame([[tmp, 0.0, 0.0, 0.0, gesture,
↪subject, sample]], columns=['sample.timestamp', 'X', 'Y', 'Z', 'gesture',
↪'subject', 'sample'])
                    df_sample = df_sample.append(df, ignore_index=True)
                    #print(df_sample)
                    df_sample_count = df_sample.count()['sample.timestamp']
                    #print(df_sample_count)
                    if df_sample_count != time_max:
                        damaged += 1
                        continue
                    if dataset_timecut is None:
                        dataset_timecut = df_sample.copy()
                    else:
                        dataset_timecut = pd.concat([dataset_timecut, df_sample])
                    samples += 1

dataset_outliers = dataset_timecut
print(str(samples) + " outliers samples sliced")
print(str(damaged) + " outliers samples damaged")

from keras import backend as K
data = dataset_cleaned
from keras.models import Sequential
from keras.layers import Bidirectional
from keras.layers import LSTM
from keras.layers import Dense

```

```

from keras.layers import Dropout
from keras.optimizers import adam_v2
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import StratifiedGroupKFold
from sklearn.model_selection import cross_validate
from sklearn.model_selection import GridSearchCV
from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
import numpy as np
import tensorflow as tf

# fix random seed for reproducibility
seed = 1000
np.random.seed(seed)
# create the dataset
def get_dataset(data, index=[]):
    X_train = []
    Y_train = []
    groups = []
    samples_idx=0
    for i, gesture in enumerate(data['gesture'].unique()):
        df_gesture = data[data['gesture']==gesture]
        for j, subject in enumerate(df_gesture['subject'].unique()):
            df_subject = df_gesture[df_gesture['subject']==subject]
            for k, sample in enumerate(df_subject['sample'].unique()):
                df_sample = df_subject[df_subject['sample']==sample]
                accel_vector = []
                for idx, row in df_sample.sort_values(by='sample.
→timestamp').iterrows():
                    accel_vector.append([row['X'],row['Y'],row['Z']])
                accel_vector = np.asarray(accel_vector)
                if len(index)==0:
                    X_train.append(accel_vector)
                    Y_train.append(gesture)
                    groups.append(subject)
                else:
                    if samples_idx in index:
                        X_train.append(accel_vector)
                        Y_train.append(gesture)
                        groups.append(subject)
                    samples_idx+=1
    X_train = np.asarray(X_train)
    Y_train = LabelEncoder().fit_transform(Y_train)
    #print(Y_train)
    return X_train, Y_train, groups

```

```

def build_model(baseset, gesture_subset):
    baseset.sort()
    basename = '-'.join(baseset)
    basemodel = tf.keras.models.load_model(basename + '_lstm')
    model = tf.keras.Sequential()
    for layer in basemodel.layers[:-1]: # go through until last layer
        layer.trainable= True
        model.add(layer)
    model.add(tf.keras.layers.Dense(len(gesture_subset),
    ↪activation='softmax', name="transfer_adjust"))
    model.build([None, 19, 3])
    #print(model.summary())
    model.compile(loss='sparse_categorical_crossentropy', optimizer=adam_v2.
    ↪Adam(learning_rate=0.001), metrics=['accuracy'])
    return model

# Function to create model, required for KerasClassifier
import pickle
def load_classifier(baseset, gesture_subset):
    gesture_subset.sort()
    name = '-'.join(gesture_subset)
    classifier = KerasClassifier(build_fn=build_model, baseset=baseset,
    ↪gesture_subset=gesture_subset, epochs=128, batch_size=19, verbose=0)
    classifier.classes_ = pickle.load(open(name + '_model_classes.
    ↪pkl','rb'))
    classifier.model = build_model(baseset,gesture_subset)
    return classifier

#print(model.model.summary())
#print(model.classes_)
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

for n_splits in [10, 5, 4, 2]:
    for epoch in [[8], [16], [32], [64], [128]]:
        cv = StratifiedGroupKFold(n_splits=n_splits, shuffle=True,
    ↪random_state=(1000+epoch[0]))
        X, y, g = get_dataset(dataset_cleaned)

        # Initialize the accuracy of the models to blank list. The accuracy
    ↪of each model will be appended to this list
        accuracy_model = []
        best_estimator = None
        # Initialize the array to zero which will store the confusion matrix

```

```

array = None
outliers = None

report_cleaned = None
report_outliers = None

print("Processing started for split estimator: " + str(n_splits) +
→", epochs: " + str(epoch))
# Iterate over each train-test split
fold = 1
for train_index, test_index in cv.split(X, y, g):
    #print(test_index)
    if len(test_index) == 0:
        continue
    print("Processing ", fold, "-fold")
    fold += 1

    classifier = load_classifier(basetest, gesture_subset)
    # Split train-test (Inverted)
    X_train, y_train, group_train = get_dataset(dataset_cleaned,
→test_index)
    X_test, y_test, group_test = get_dataset(dataset_cleaned,
→train_index)
    X_outliers, y_outliers, group_test =
→get_dataset(dataset_outliers)
    # Train the model
    History = classifier.fit(X_train, y_train, epochs=epoch[0])
    # Append to accuracy_model the accuracy of the model
    accuracy_model.append(accuracy_score(y_test, classifier.
→predict(X_test), normalize=True))
    if accuracy_model[-1] == max(accuracy_model):
        best_estimator = classifier
    # Calculate the confusion matrix
    c = confusion_matrix(y_test, classifier.predict(X_test))
    # Add the score to the previous confusion matrix of previous
→model

    if isinstance(array, np.ndarray) == False:
        array = c.copy()
    else:
        array = array + c

    # Calculate the confusion matrix
    c = confusion_matrix(y_outliers, classifier.predict(X_outliers))
    # Add the score to the previous confusion matrix of previous
→model

    if isinstance(outliers, np.ndarray) == False:

```

```

        outliers = c.copy()
    else:
        outliers = outliers + c

    #Accumulate for classification report
    if isinstance(report_cleaned, list) == False:
        report_cleaned = [y_test, classifier.predict(X_test)]
    else:
        report_cleaned[0] = np.append(report_cleaned[0],y_test)
        report_cleaned[1] = np.append(report_cleaned[1],classifier.
→predict(X_test))

    #Accumulate for classification report
    if isinstance(report_outliers, list) == False:
        report_outliers = [y_outliers, classifier.
→predict(X_outliers)]
    else:
        report_outliers[0] = np.
→append(report_outliers[0],y_outliers)
        report_outliers[1] = np.
→append(report_outliers[1],classifier.predict(X_outliers))

    # Print the accuracy
    print("At split estimator: " + str(n_splits) + ", epochs: " +
→str(epoch))
    print("Accurace mean(std): " + str(np.mean(accuracy_model)) + "(" +
→str(np.std(accuracy_model)) + ")")

    # To calculate the classification reports

    print("Classification report for all valid cross_validations
→against their tests sets")
    print(classification_report(report_cleaned[0], report_cleaned[1],
→target_names=gesture_subset))

    print("Classification report for all valid cross_validations
→against outliers")
    print(classification_report(report_outliers[0], report_outliers[1],
→target_names=gesture_subset))

    # To calculate the confusion matrix

    print("Confusion Matrix for all valid cross_validations against
→their tests sets")
    make_confusion_matrix(array, categories=gesture_subset)

```

```

        print("Confusion Matrix for all valid cross_validations against_
→outliers")
        make_confusion_matrix(outliers, categories=gesture_subset)

    for n_splits in [5]:
        for epoch in [[8], [16], [32], [64], [128]]:
            cv = StratifiedGroupKFold(n_splits=n_splits, shuffle=True,
→random_state=(1000+epoch[0]))
            X, y, g = get_dataset(dataset_cleaned)

            # Initialize the accuracy of the models to blank list. The accuracy_
→of each model will be appended to this list
            accuracy_model = []
            best_estimator = None
            # Initialize the array to zero which will store the confusion matrix
            array = None
            outliers = None

            report_cleaned = None
            report_outliers = None

            print("Processing started for real split: " + str(n_splits) + ",
→epochs: " + str(epoch))
            # Iterate over each train-test split
            fold = 1
            for train_index, test_index in cv.split(X, y, g):
                #print(test_index)
                if len(test_index) == 0:
                    continue
                print("Processing ", fold, "-fold")
                fold += 1

                classifier = load_classifier(baseset, gesture_subset)
                # Split train-test (Inverted)
                X_train, y_train, group_train = get_dataset(dataset_cleaned,
→train_index)
                X_test, y_test, group_test = get_dataset(dataset_cleaned,
→test_index)
                X_outliers, y_outliers, group_test =
→get_dataset(dataset_outliers)
                # Train the model
                History = classifier.fit(X_train, y_train, epochs=epoch[0])
                # Append to accuracy_model the accuracy of the model
                accuracy_model.append(accuracy_score(y_test, classifier.
→predict(X_test), normalize=True))
                if accuracy_model[-1] == max(accuracy_model):

```

```

        best_estimator = classifier
        # Calculate the confusion matrix
        c = confusion_matrix(y_test, classifier.predict(X_test))
        # Add the score to the previous confusion matrix of previous
→model

        if isinstance(array, np.ndarray) == False:
            array = c.copy()
        else:
            array = array + c

        # Calculate the confusion matrix
        c = confusion_matrix(y_outliers, classifier.predict(X_outliers))
        # Add the score to the previous confusion matrix of previous
→model

        if isinstance(outliers, np.ndarray) == False:
            outliers = c.copy()
        else:
            outliers = outliers + c

        #Accumulate for classification report
        if isinstance(report_cleaned, list) == False:
            report_cleaned = [y_test, classifier.predict(X_test)]
        else:
            report_cleaned[0] = np.append(report_cleaned[0],y_test)
            report_cleaned[1] = np.append(report_cleaned[1],classifier.
→predict(X_test))

        #Accumulate for classification report
        if isinstance(report_outliers, list) == False:
            report_outliers = [y_outliers, classifier.
→predict(X_outliers)]
        else:
            report_outliers[0] = np.
→append(report_outliers[0],y_outliers)
            report_outliers[1] = np.
→append(report_outliers[1],classifier.predict(X_outliers))

        # Print the accuracy
        print("At split estimator: " + str(n_splits) + ", epochs: " +
→str(epoch))
        print("Accurace mean(std): " + str(np.mean(accuracy_model)) + "(" +
→str(np.std(accuracy_model)) + ")")

        # To calculate the classification reports

        print("Classification report for all valid cross_validations
→against their tests sets")

```

```

        print(classification_report(report_cleaned[0], report_cleaned[1],
        ↪target_names=gesture_subset))

        print("Classification report for all valid cross_validations
        ↪against outliers")
        print(classification_report(report_outliers[0], report_outliers[1],
        ↪target_names=gesture_subset))

        # To calculate the confusion matrix

        print("Confusion Matrix for all valid cross_validations against
        ↪their tests sets")
        make_confusion_matrix(array, categories=gesture_subset)

        print("Confusion Matrix for all valid cross_validations against
        ↪outliers")
        make_confusion_matrix(outliers, categories=gesture_subset)

baseset = base_transfer_set
dataset = transfers_size_3[2]

model = create_transfer_models(baseset, dataset)

```

Processing tranfers models at 10%, 20%, 25%, 50% and 80% data for gestures:

['03', '15', '19']

Baseset: ['01', '02', '04', '05', '08', '09', '12', '13', '16', '17', '18', '20']

Loadind Dataset: ['03', '15', '19']

490 samples loaded

Scaling Dataset: ['03', '15', '19']

490 samples scaled

Cleaning Dataset: ['03', '15', '19']

365 samples cleaned

125 samples outliers

Time slicing Cleaned Dataset: ['03', '15', '19']

365 cleaned samples sliced

0 cleaned samples damaged

Time slicing Outliers Dataset: ['03', '15', '19']

125 outliers samples sliced

0 outliers samples damaged

2021-09-21 12:24:47.764095: W

tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dLError: libcudart.so.11.0: cannot open shared object file: No such file or directory

2021-09-21 12:24:47.764131: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]

Ignore above cudart dlerror if you do not have a GPU set up on your machine.

Processing started for split estimator: 10, epochs: [8]

Processing 1 -fold

2021-09-21 12:25:12.158003: W

tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object file: No such file or directory

2021-09-21 12:25:12.158038: W

tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit: UNKNOWN ERROR (303)

2021-09-21 12:25:12.158060: I

tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (mqx-public): /proc/driver/nvidia/version does not exist

2021-09-21 12:25:12.158610: I tensorflow/core/platform/cpu_feature_guard.cc:142]

This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2 AVX512F FMA

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

2021-09-21 12:29:41.006556: I

tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

Processing 6 -fold

Processing 7 -fold

Processing 8 -fold

At split estimator: 10, epochs: [8]

Accurace mean(std): 0.8257416370870847(0.09776895568454622)

Classification report for all valid cross_validations against their tests sets

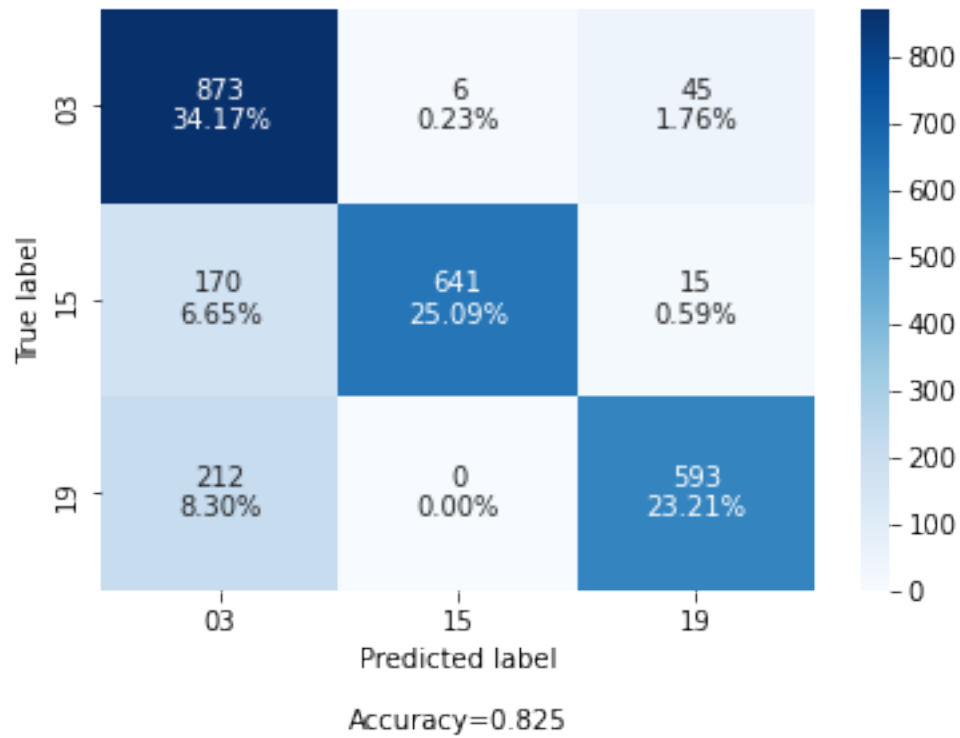
	precision	recall	f1-score	support
03	0.70	0.94	0.80	924
15	0.99	0.78	0.87	826
19	0.91	0.74	0.81	805
accuracy			0.82	2555
macro avg	0.86	0.82	0.83	2555
weighted avg	0.86	0.82	0.83	2555

Classification report for all valid cross_validations against outliers

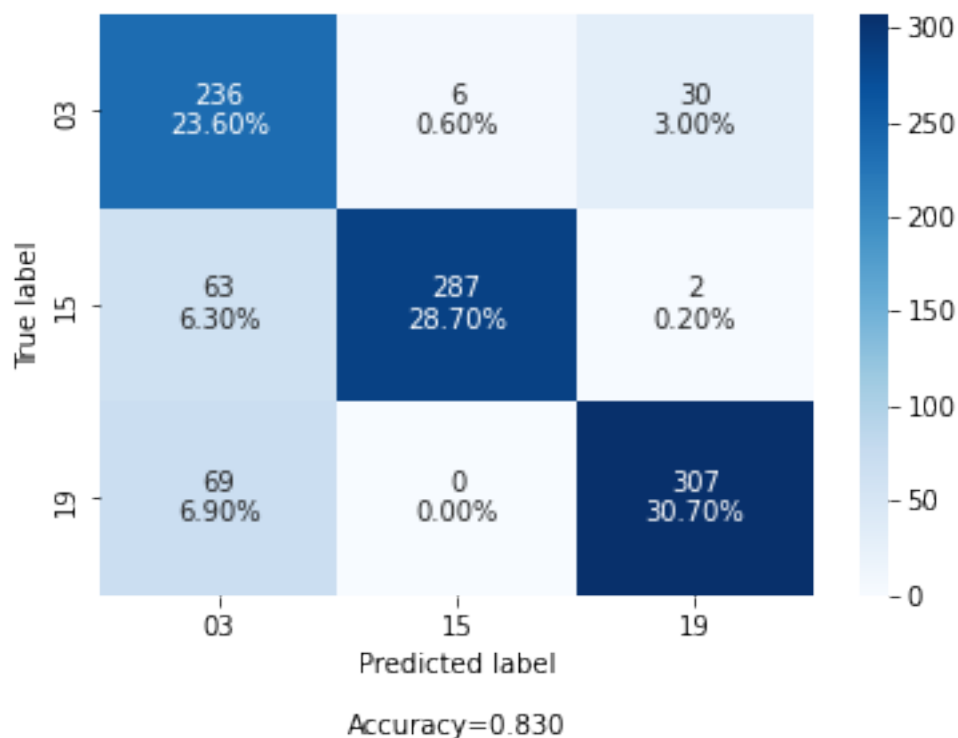
	precision	recall	f1-score	support
--	-----------	--------	----------	---------

03	0.64	0.87	0.74	272
15	0.98	0.82	0.89	352
19	0.91	0.82	0.86	376
accuracy			0.83	1000
macro avg	0.84	0.83	0.83	1000
weighted avg	0.86	0.83	0.84	1000

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 10, epochs: [16]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

Processing 6 -fold

Processing 7 -fold

Processing 8 -fold

At split estimator: 10, epochs: [16]

Accurace mean(std): 0.851810644058087(0.11023002741452818)

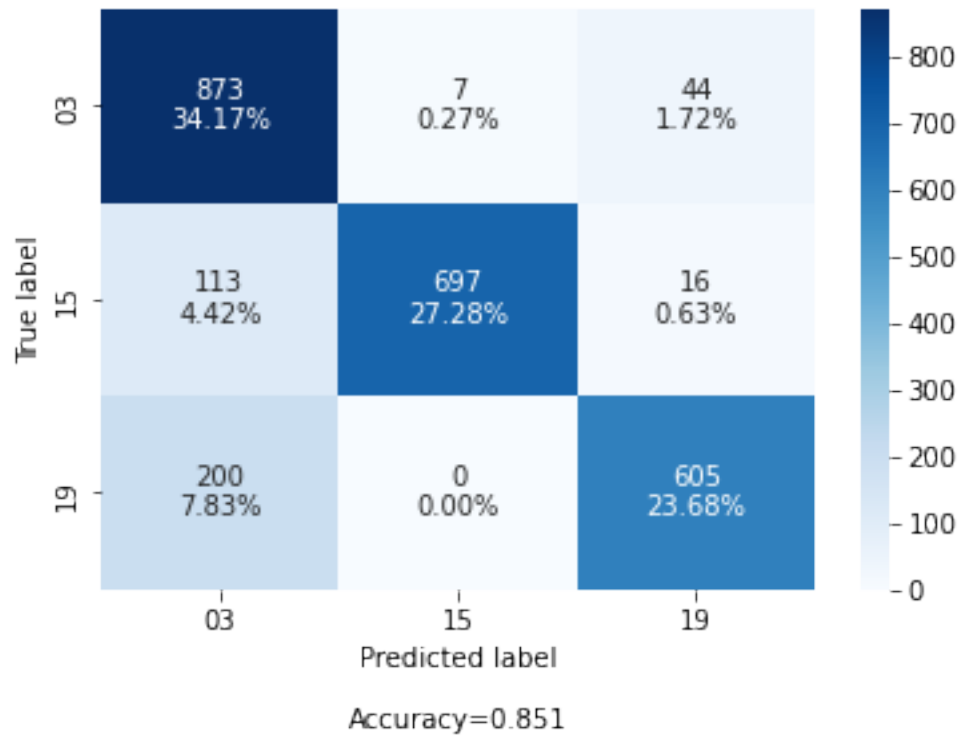
Classification report for all valid cross_validations against their tests sets

	precision	recall	f1-score	support
03	0.74	0.94	0.83	924
15	0.99	0.84	0.91	826
19	0.91	0.75	0.82	805
accuracy			0.85	2555
macro avg	0.88	0.85	0.85	2555
weighted avg	0.87	0.85	0.85	2555

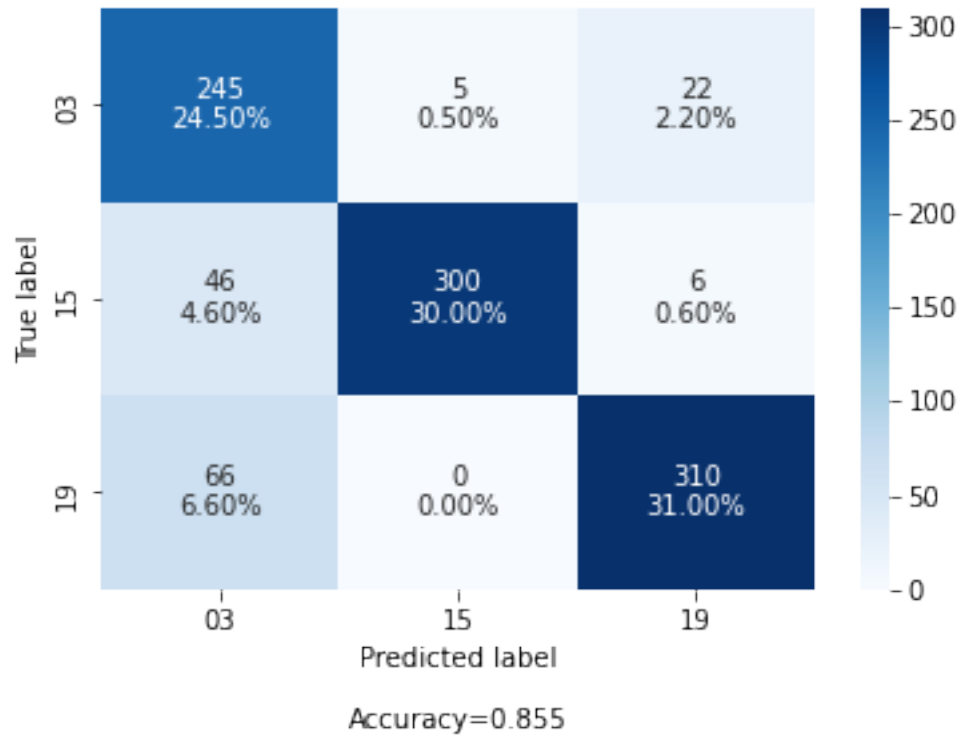
Classification report for all valid cross_validations against outliers

	precision	recall	f1-score	support
03	0.69	0.90	0.78	272
15	0.98	0.85	0.91	352
19	0.92	0.82	0.87	376
accuracy			0.85	1000
macro avg	0.86	0.86	0.85	1000
weighted avg	0.88	0.85	0.86	1000

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 10, epochs: [32]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

Processing 6 -fold

Processing 7 -fold

Processing 8 -fold

At split estimator: 10, epochs: [32]

Accurace mean(std): 0.844699819921805(0.09641333113860052)

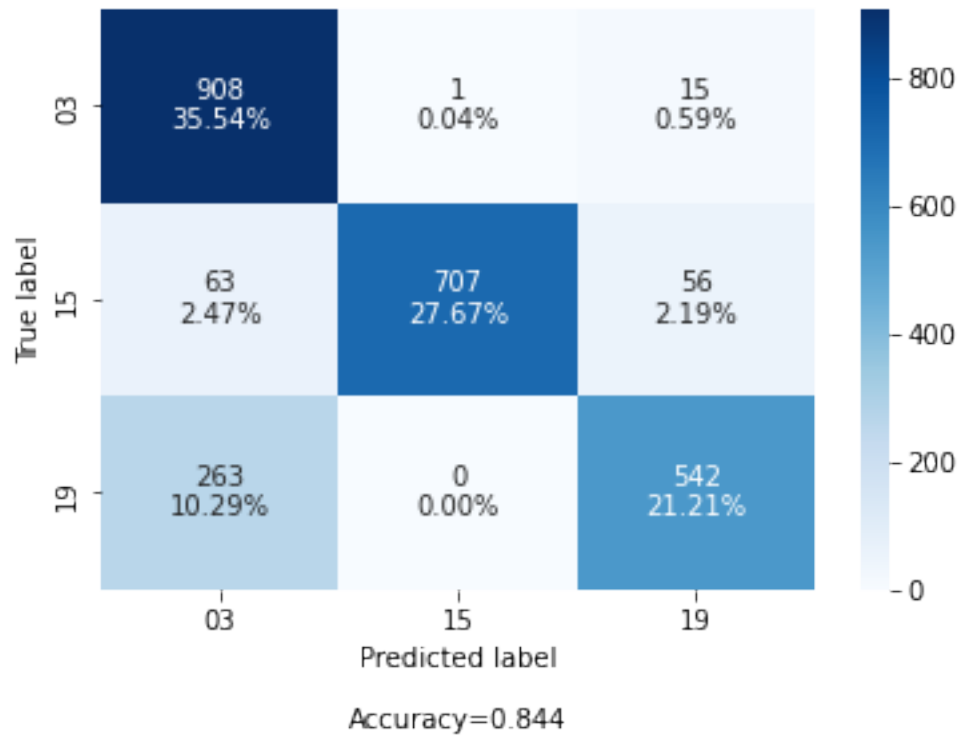
Classification report for all valid cross_validations against their tests sets

	precision	recall	f1-score	support
03	0.74	0.98	0.84	924
15	1.00	0.86	0.92	826
19	0.88	0.67	0.76	805
accuracy			0.84	2555
macro avg	0.87	0.84	0.84	2555
weighted avg	0.87	0.84	0.84	2555

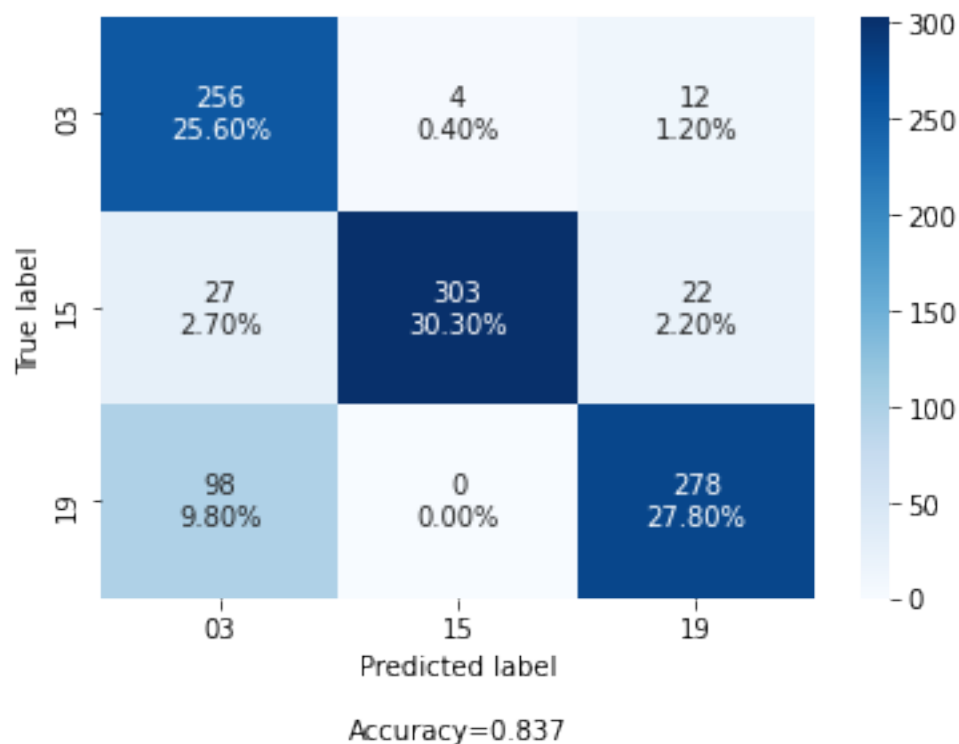
Classification report for all valid cross_validations against outliers

	precision	recall	f1-score	support
03	0.67	0.94	0.78	272
15	0.99	0.86	0.92	352
19	0.89	0.74	0.81	376
accuracy			0.84	1000
macro avg	0.85	0.85	0.84	1000
weighted avg	0.87	0.84	0.84	1000

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 10, epochs: [64]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

Processing 6 -fold

Processing 7 -fold

Processing 8 -fold

At split estimator: 10, epochs: [64]

Accurace mean(std): 0.9272699853875566(0.03230822757674885)

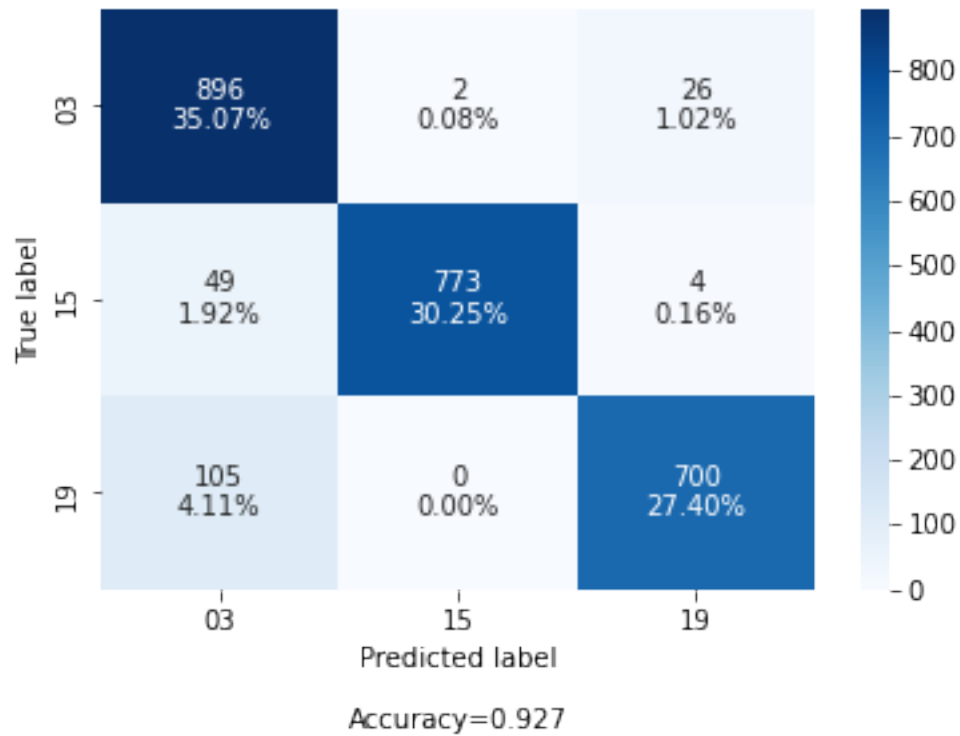
Classification report for all valid cross_validations against their tests sets

	precision	recall	f1-score	support
03	0.85	0.97	0.91	924
15	1.00	0.94	0.97	826
19	0.96	0.87	0.91	805
accuracy			0.93	2555
macro avg	0.94	0.93	0.93	2555
weighted avg	0.93	0.93	0.93	2555

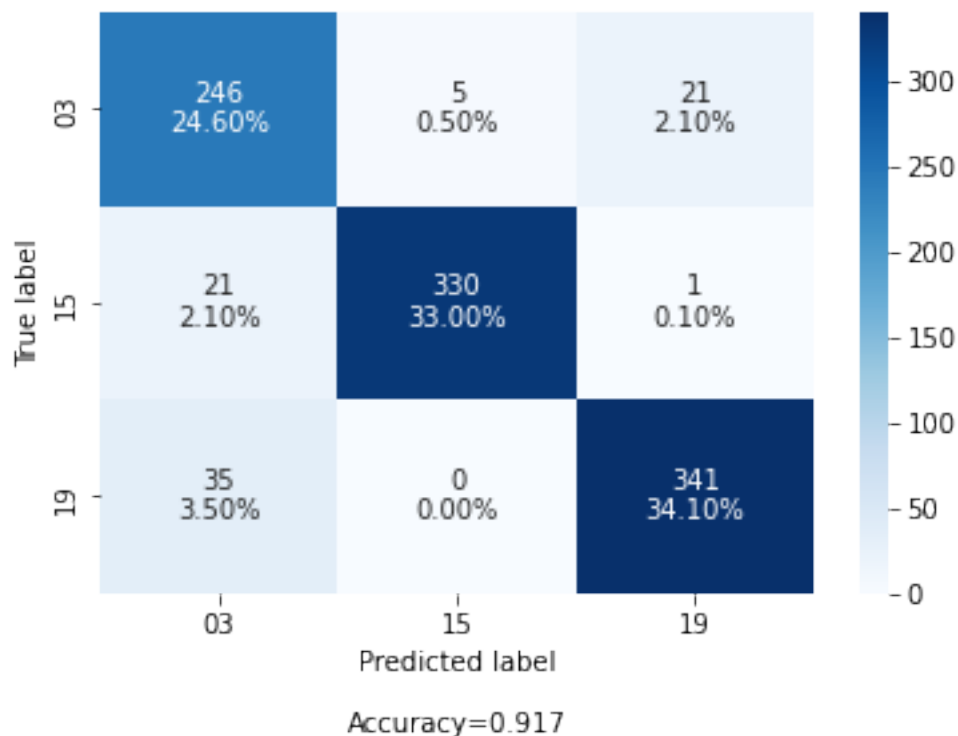
Classification report for all valid cross_validations against outliers

	precision	recall	f1-score	support
03	0.81	0.90	0.86	272
15	0.99	0.94	0.96	352
19	0.94	0.91	0.92	376
accuracy			0.92	1000
macro avg	0.91	0.92	0.91	1000
weighted avg	0.92	0.92	0.92	1000

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 10, epochs: [128]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

Processing 6 -fold

Processing 7 -fold

Processing 8 -fold

At split estimator: 10, epochs: [128]

Accurace mean(std): 0.8695858681609907(0.06650714087213956)

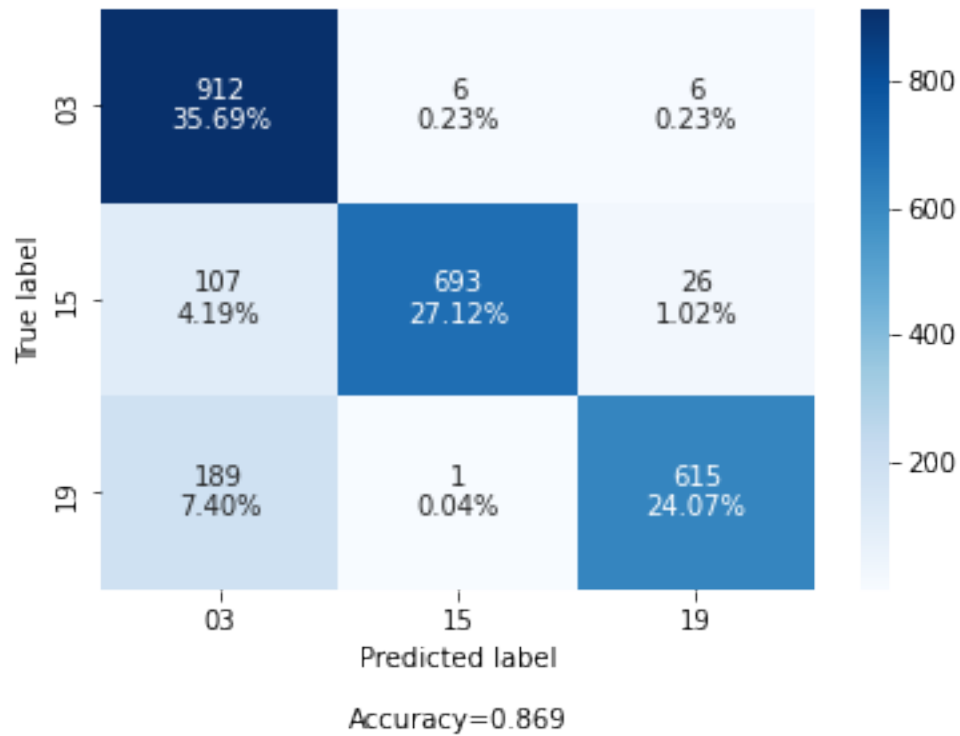
Classification report for all valid cross_validations against their tests sets

	precision	recall	f1-score	support
03	0.75	0.99	0.86	924
15	0.99	0.84	0.91	826
19	0.95	0.76	0.85	805
accuracy			0.87	2555
macro avg	0.90	0.86	0.87	2555
weighted avg	0.89	0.87	0.87	2555

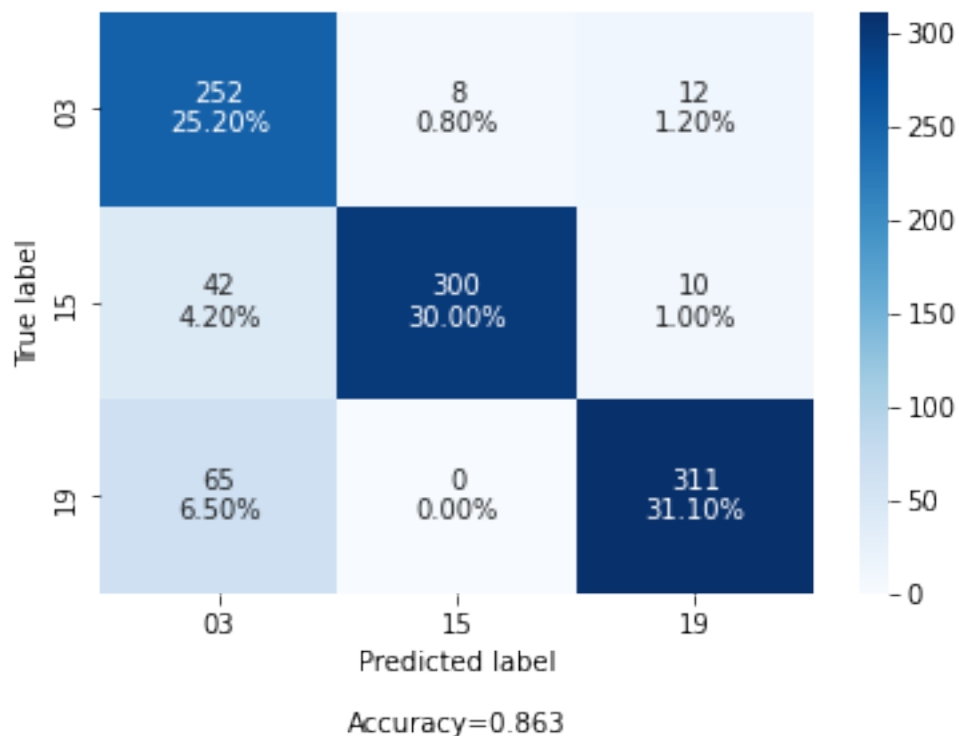
Classification report for all valid cross_validations against outliers

	precision	recall	f1-score	support
03	0.70	0.93	0.80	272
15	0.97	0.85	0.91	352
19	0.93	0.83	0.88	376
accuracy			0.86	1000
macro avg	0.87	0.87	0.86	1000
weighted avg	0.88	0.86	0.87	1000

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 5, epochs: [8]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [8]

Accurace mean(std): 0.9355414294855817(0.03267804927509206)

Classification report for all valid cross_validations against their tests sets

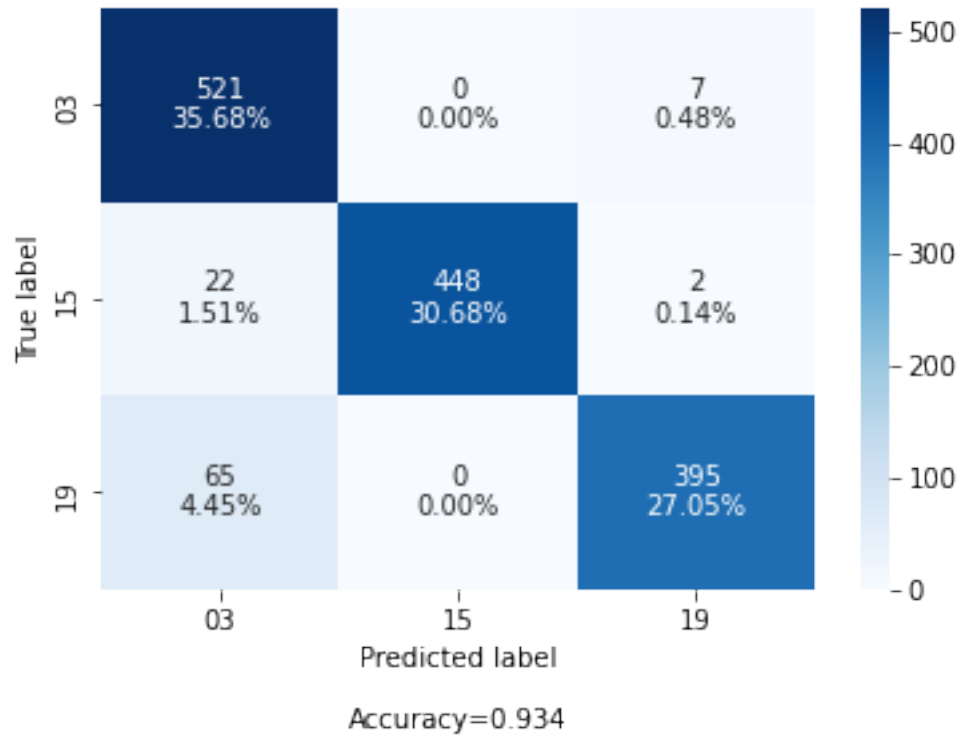
	precision	recall	f1-score	support
03	0.86	0.99	0.92	528
15	1.00	0.95	0.97	472
19	0.98	0.86	0.91	460
accuracy			0.93	1460
macro avg	0.94	0.93	0.94	1460
weighted avg	0.94	0.93	0.93	1460

Classification report for all valid cross_validations against outliers

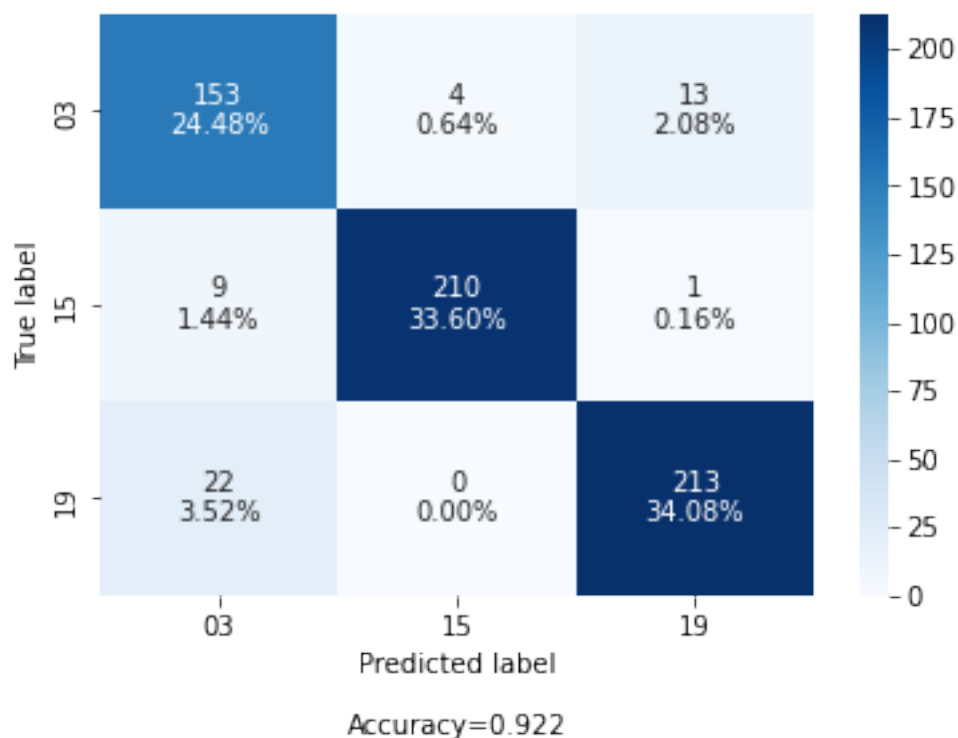
	precision	recall	f1-score	support
03	0.83	0.90	0.86	170

15	0.98	0.95	0.97	220
19	0.94	0.91	0.92	235
accuracy			0.92	625
macro avg	0.92	0.92	0.92	625
weighted avg	0.92	0.92	0.92	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 5, epochs: [16]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [16]

Accurace mean(std): 0.8978602136153988(0.07137031441233274)

Classification report for all valid cross_validations against their tests sets

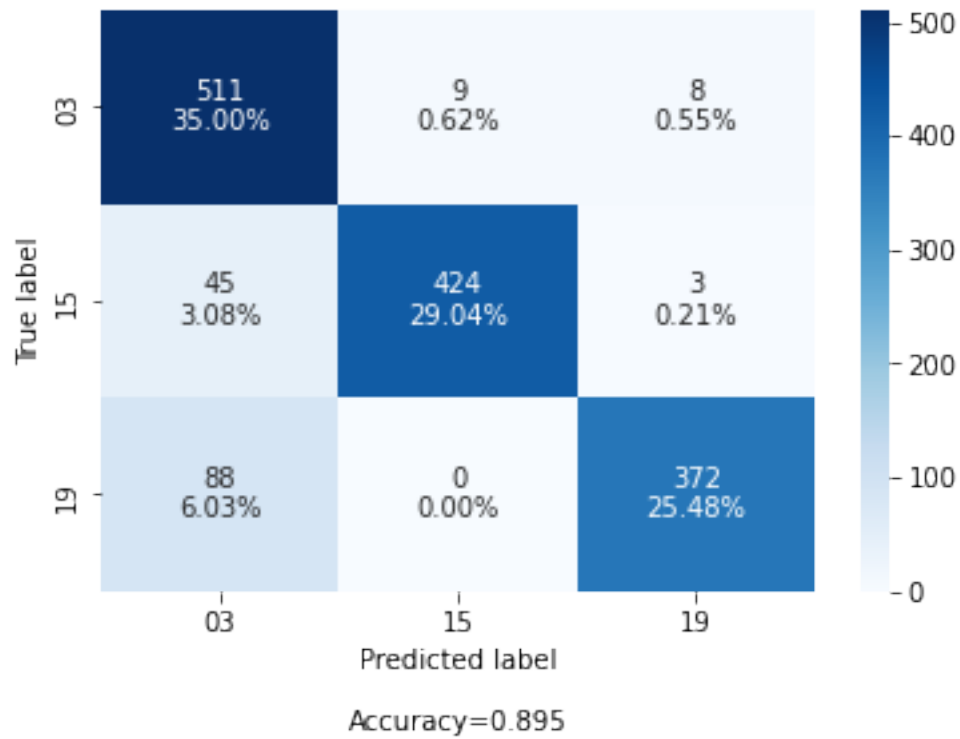
	precision	recall	f1-score	support
03	0.79	0.97	0.87	528
15	0.98	0.90	0.94	472
19	0.97	0.81	0.88	460
accuracy			0.90	1460
macro avg	0.91	0.89	0.90	1460
weighted avg	0.91	0.90	0.90	1460

Classification report for all valid cross_validations against outliers

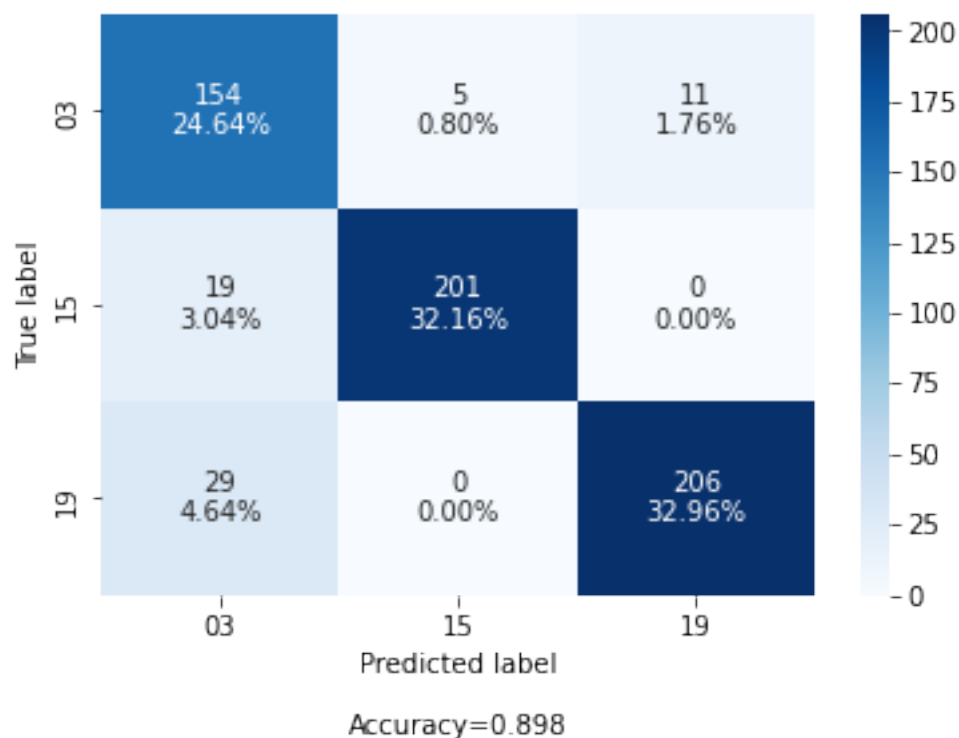
	precision	recall	f1-score	support
03	0.76	0.91	0.83	170

15	0.98	0.91	0.94	220
19	0.95	0.88	0.91	235
accuracy			0.90	625
macro avg	0.90	0.90	0.89	625
weighted avg	0.91	0.90	0.90	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 5, epochs: [32]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [32]

Accurace mean(std): 0.8939489602095605(0.13933779817516648)

Classification report for all valid cross_validations against their tests sets

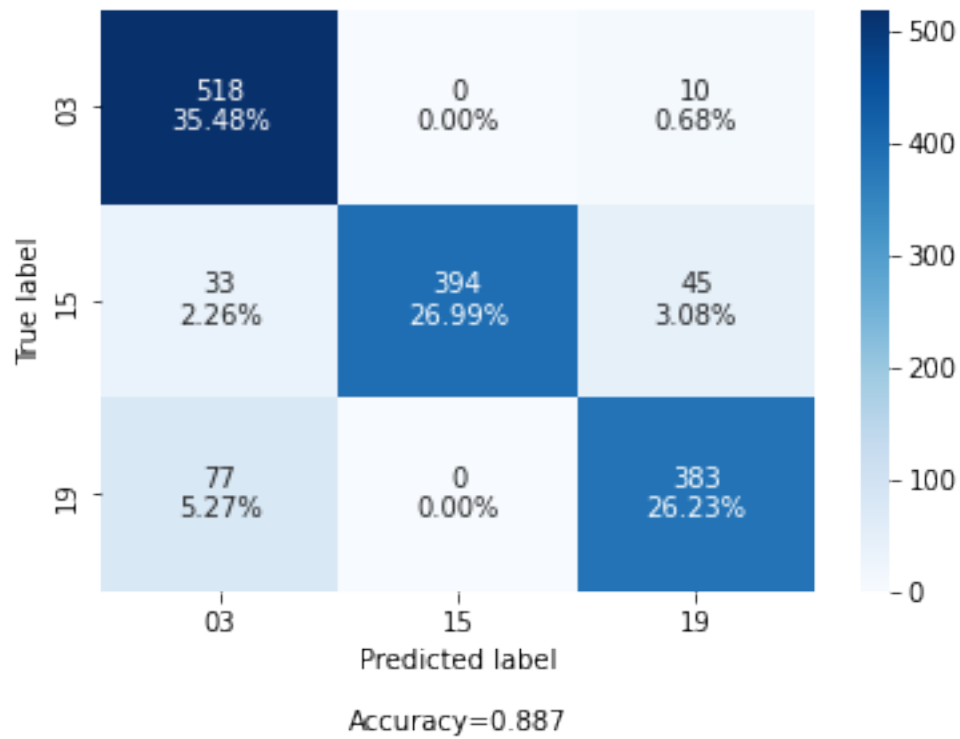
	precision	recall	f1-score	support
03	0.82	0.98	0.90	528
15	1.00	0.83	0.91	472
19	0.87	0.83	0.85	460
accuracy			0.89	1460
macro avg	0.90	0.88	0.89	1460
weighted avg	0.90	0.89	0.89	1460

Classification report for all valid cross_validations against outliers

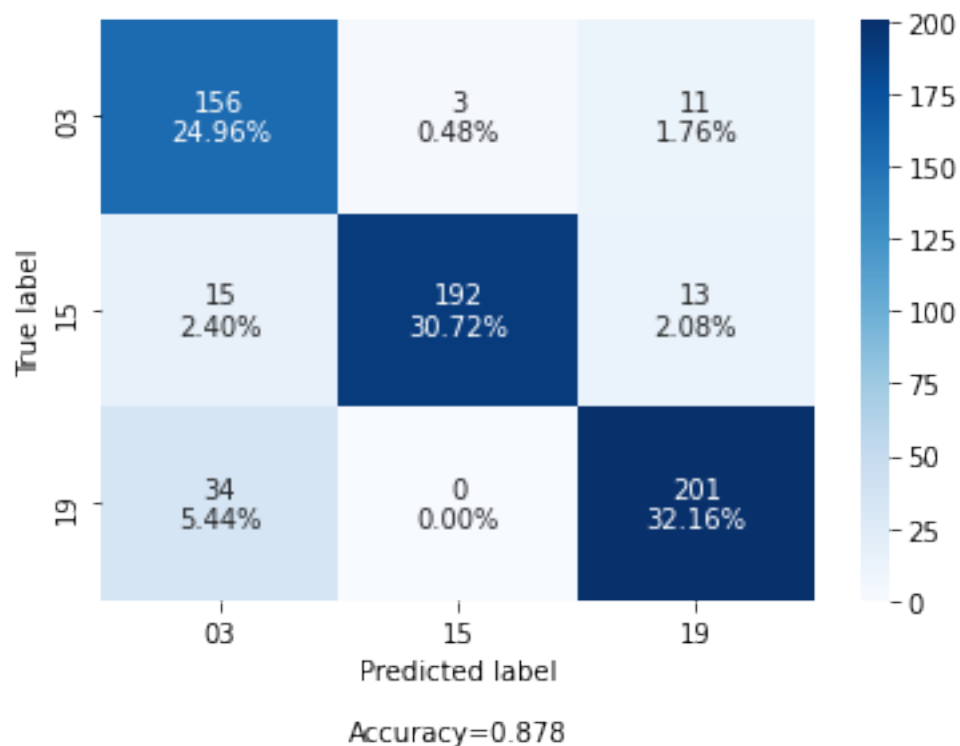
	precision	recall	f1-score	support
03	0.76	0.92	0.83	170

15	0.98	0.87	0.93	220
19	0.89	0.86	0.87	235
accuracy			0.88	625
macro avg	0.88	0.88	0.88	625
weighted avg	0.89	0.88	0.88	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 5, epochs: [64]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [64]

Accurace mean(std): 0.9227551996890014(0.037020555916630414)

Classification report for all valid cross_validations against their tests sets

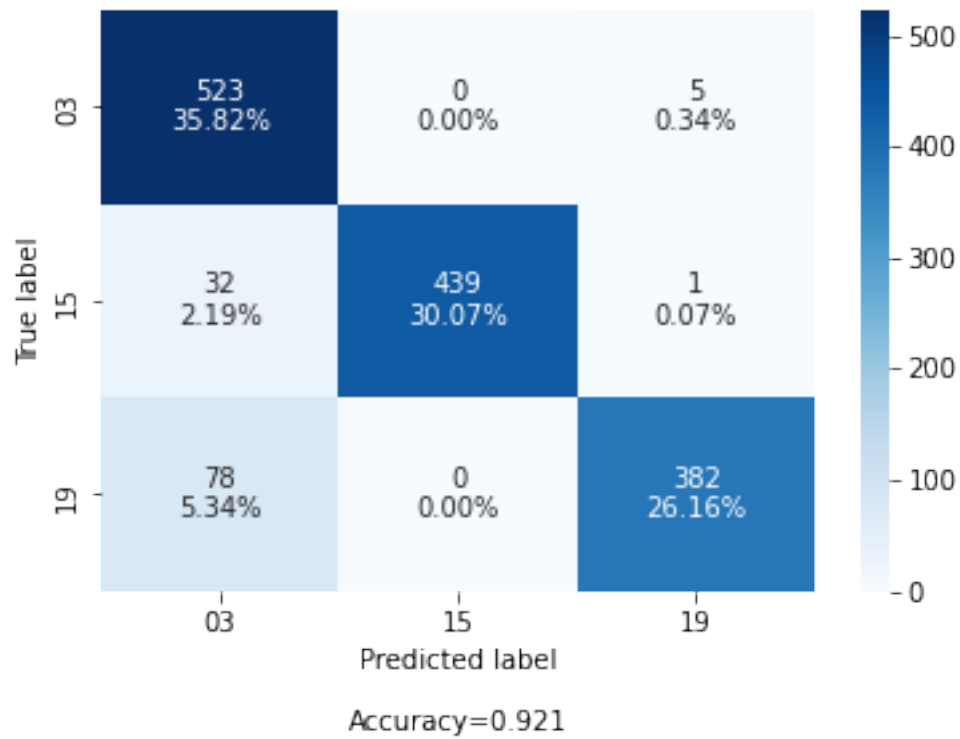
	precision	recall	f1-score	support
03	0.83	0.99	0.90	528
15	1.00	0.93	0.96	472
19	0.98	0.83	0.90	460
accuracy			0.92	1460
macro avg	0.94	0.92	0.92	1460
weighted avg	0.93	0.92	0.92	1460

Classification report for all valid cross_validations against outliers

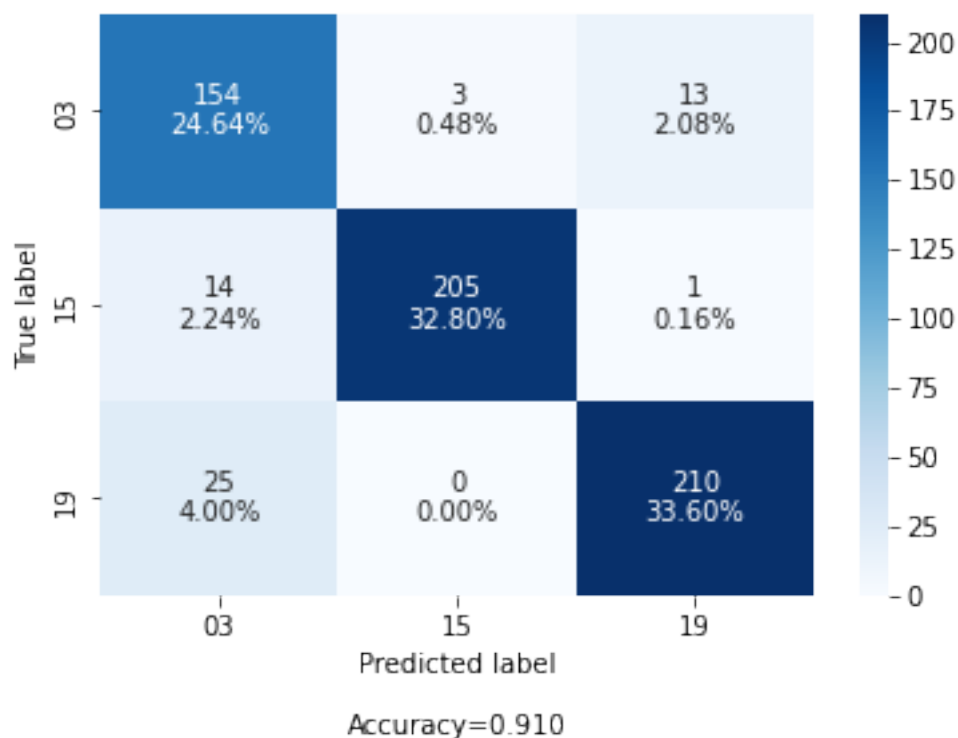
	precision	recall	f1-score	support
03	0.80	0.91	0.85	170

15	0.99	0.93	0.96	220
19	0.94	0.89	0.92	235
accuracy			0.91	625
macro avg	0.91	0.91	0.91	625
weighted avg	0.92	0.91	0.91	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 5, epochs: [128]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [128]

Accurace mean(std): 0.9296398993899203(0.02491871136707304)

Classification report for all valid cross_validations against their tests sets

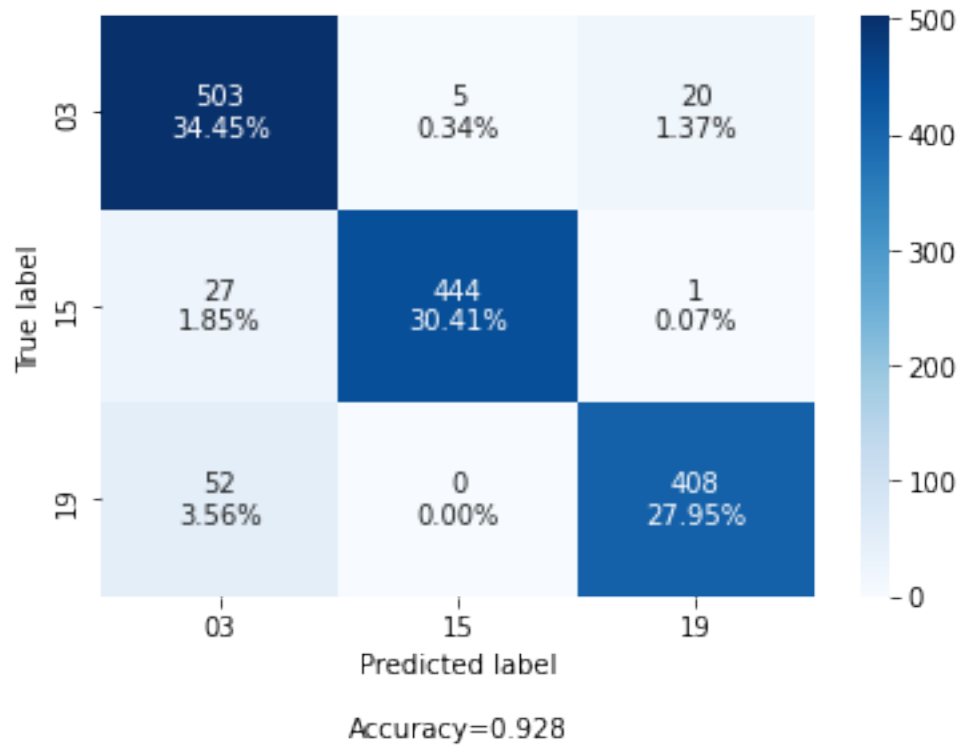
	precision	recall	f1-score	support
03	0.86	0.95	0.91	528
15	0.99	0.94	0.96	472
19	0.95	0.89	0.92	460
accuracy			0.93	1460
macro avg	0.93	0.93	0.93	1460
weighted avg	0.93	0.93	0.93	1460

Classification report for all valid cross_validations against outliers

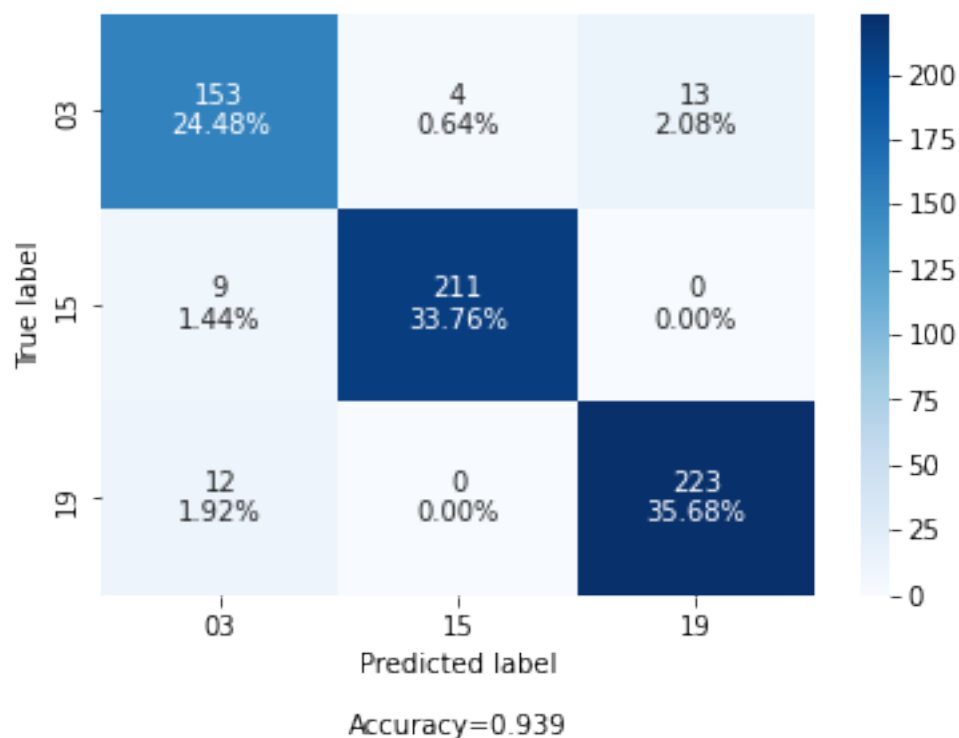
	precision	recall	f1-score	support
03	0.88	0.90	0.89	170

15	0.98	0.96	0.97	220
19	0.94	0.95	0.95	235
accuracy			0.94	625
macro avg	0.94	0.94	0.94	625
weighted avg	0.94	0.94	0.94	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 4, epochs: [8]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

At split estimator: 4, epochs: [8]

Accurace mean(std): 0.9591384446419331(0.031287127288540086)

Classification report for all valid cross_validations against their tests sets

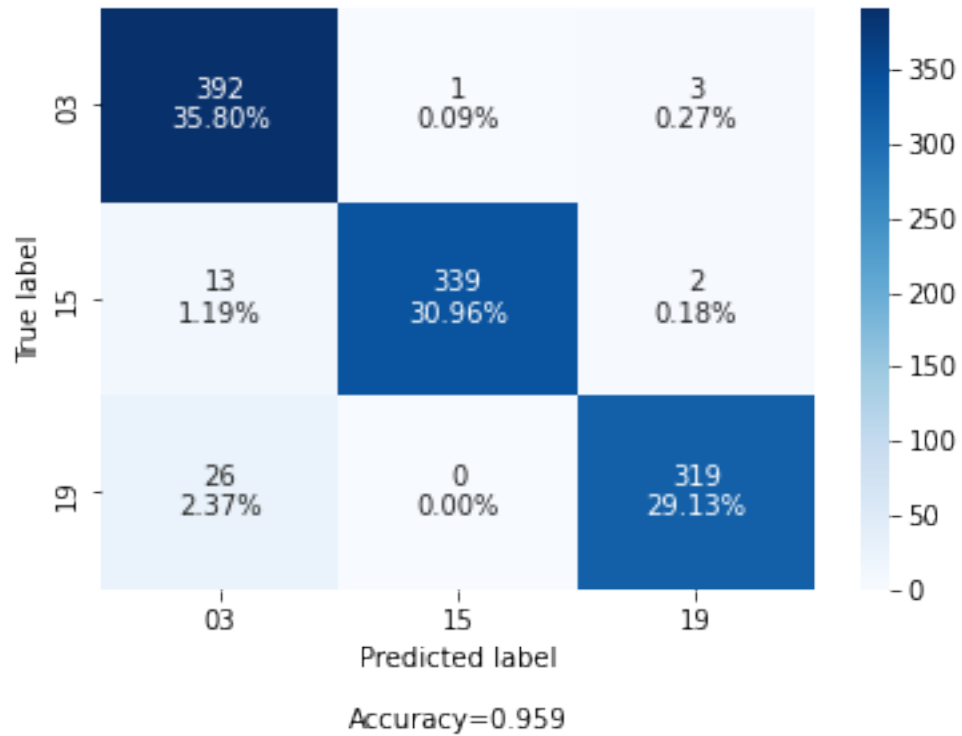
	precision	recall	f1-score	support
03	0.91	0.99	0.95	396
15	1.00	0.96	0.98	354
19	0.98	0.92	0.95	345
accuracy			0.96	1095
macro avg	0.96	0.96	0.96	1095
weighted avg	0.96	0.96	0.96	1095

Classification report for all valid cross_validations against outliers

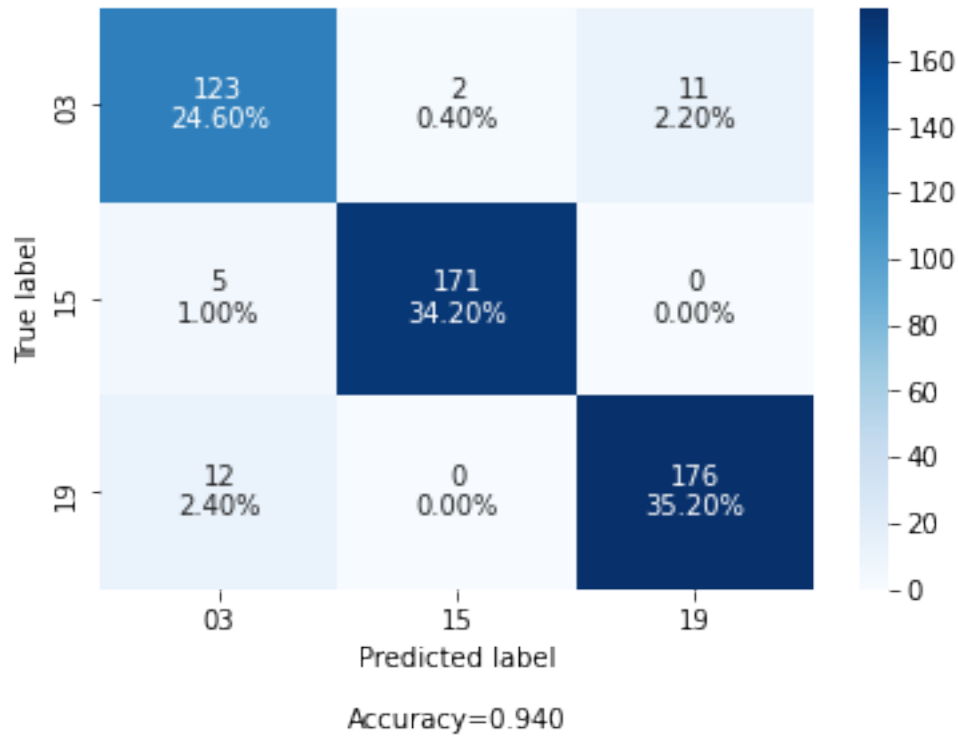
	precision	recall	f1-score	support
03	0.88	0.90	0.89	136
15	0.99	0.97	0.98	176

19	0.94	0.94	0.94	188
accuracy			0.94	500
macro avg	0.94	0.94	0.94	500
weighted avg	0.94	0.94	0.94	500

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 4, epochs: [16]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

At split estimator: 4, epochs: [16]

Accurace mean(std): 0.9674701509427636(0.030944729318466178)

Classification report for all valid cross_validations against their tests sets

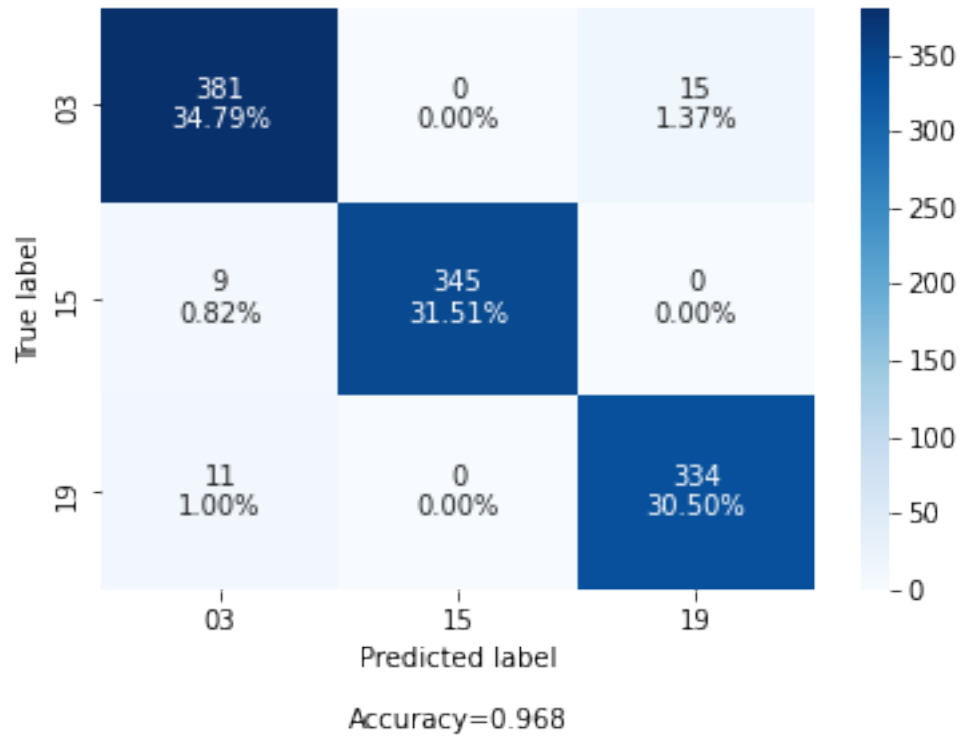
	precision	recall	f1-score	support
03	0.95	0.96	0.96	396
15	1.00	0.97	0.99	354
19	0.96	0.97	0.96	345
accuracy			0.97	1095
macro avg	0.97	0.97	0.97	1095
weighted avg	0.97	0.97	0.97	1095

Classification report for all valid cross_validations against outliers

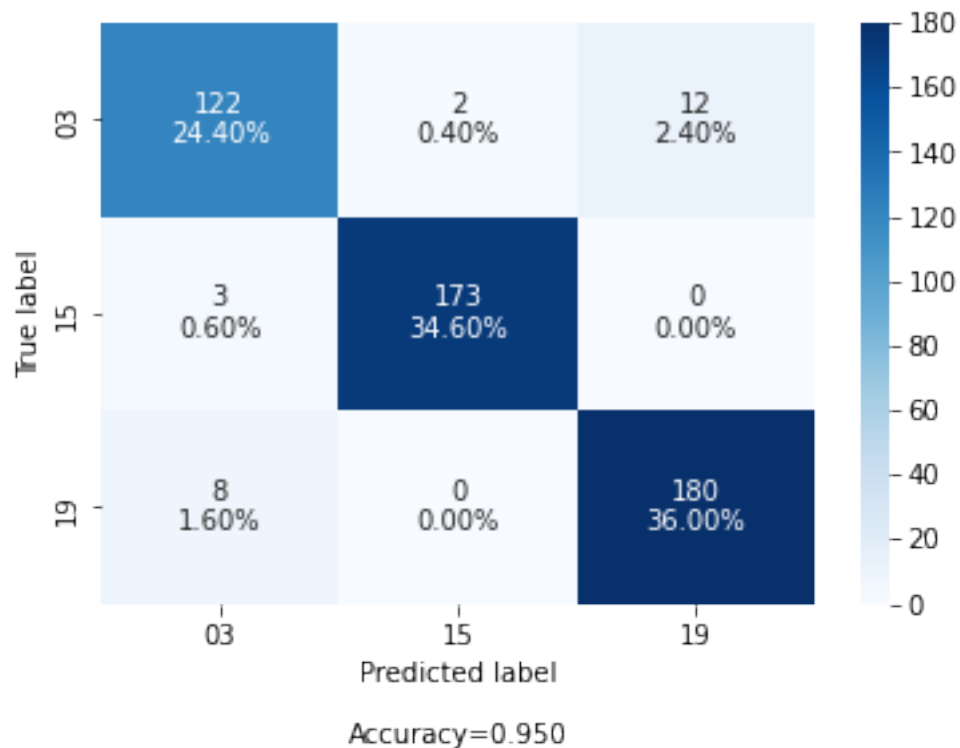
	precision	recall	f1-score	support
03	0.92	0.90	0.91	136
15	0.99	0.98	0.99	176

19	0.94	0.96	0.95	188
accuracy			0.95	500
macro avg	0.95	0.95	0.95	500
weighted avg	0.95	0.95	0.95	500

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 4, epochs: [32]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

At split estimator: 4, epochs: [32]

Accurace mean(std): 0.9605097218871572(0.018756219096305538)

Classification report for all valid cross_validations against their tests sets

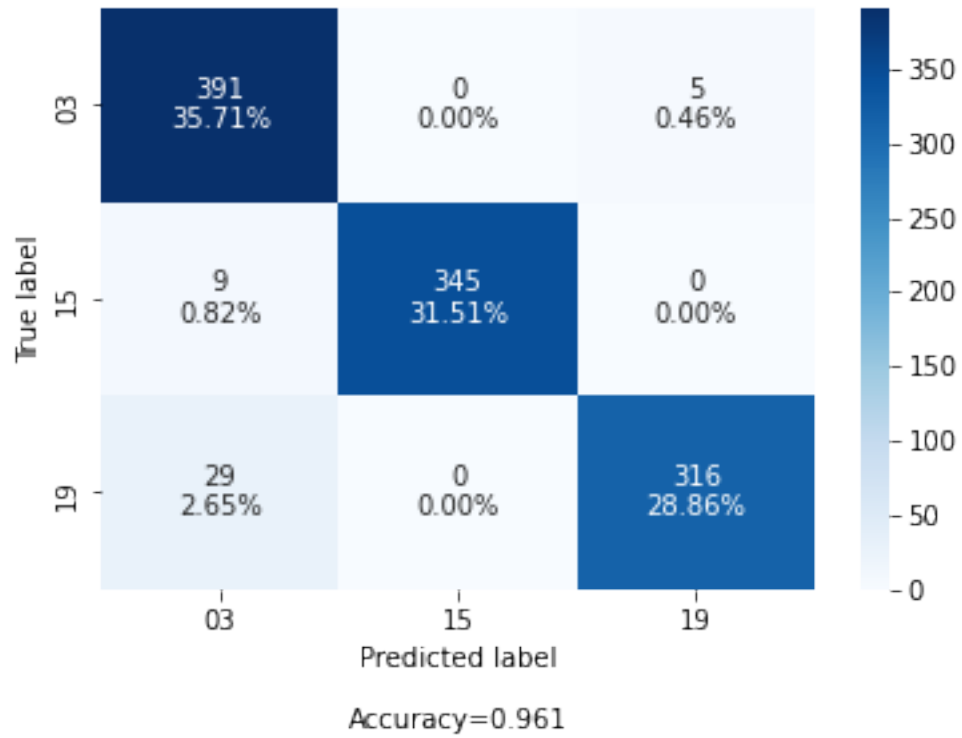
	precision	recall	f1-score	support
03	0.91	0.99	0.95	396
15	1.00	0.97	0.99	354
19	0.98	0.92	0.95	345
accuracy			0.96	1095
macro avg	0.97	0.96	0.96	1095
weighted avg	0.96	0.96	0.96	1095

Classification report for all valid cross_validations against outliers

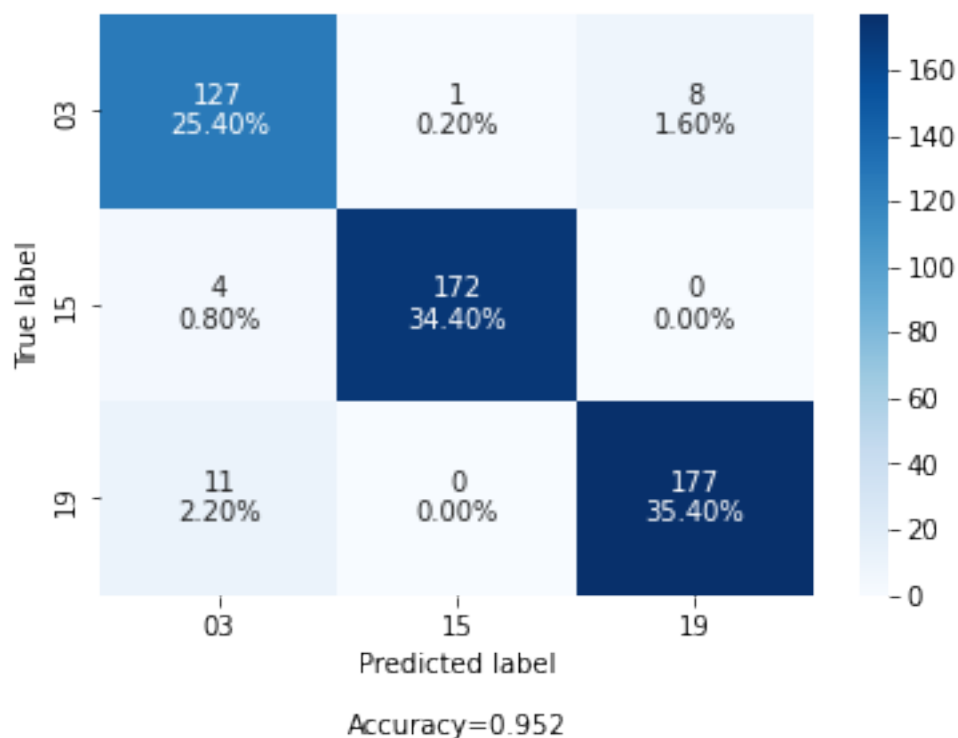
	precision	recall	f1-score	support
03	0.89	0.93	0.91	136
15	0.99	0.98	0.99	176

	19	0.96	0.94	0.95	188
accuracy				0.95	500
macro avg		0.95	0.95	0.95	500
weighted avg		0.95	0.95	0.95	500

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 4, epochs: [64]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

At split estimator: 4, epochs: [64]

Accurace mean(std): 0.9332867148461889(0.05059844987606554)

Classification report for all valid cross_validations against their tests sets

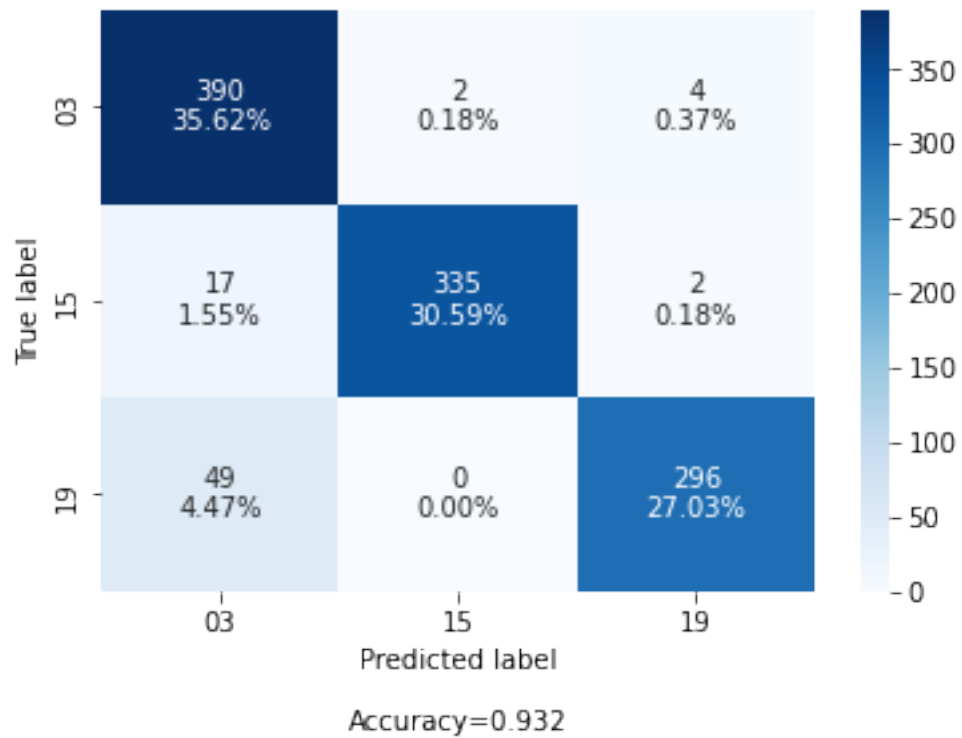
	precision	recall	f1-score	support
03	0.86	0.98	0.92	396
15	0.99	0.95	0.97	354
19	0.98	0.86	0.91	345
accuracy			0.93	1095
macro avg	0.94	0.93	0.93	1095
weighted avg	0.94	0.93	0.93	1095

Classification report for all valid cross_validations against outliers

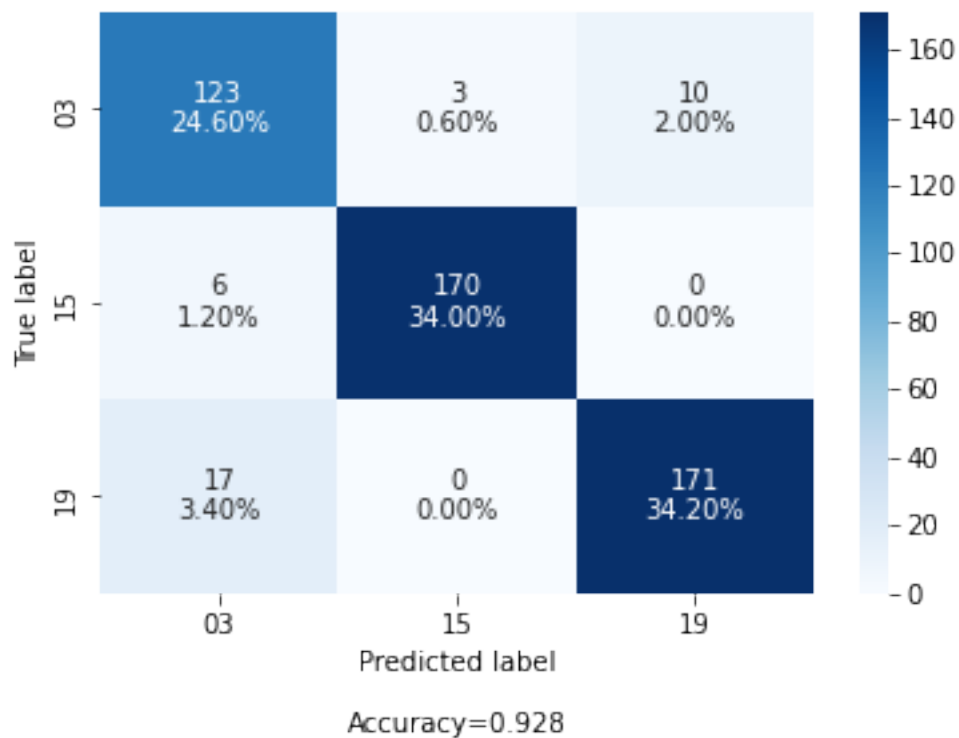
	precision	recall	f1-score	support
03	0.84	0.90	0.87	136
15	0.98	0.97	0.97	176

19	0.94	0.91	0.93	188
accuracy			0.93	500
macro avg	0.92	0.93	0.92	500
weighted avg	0.93	0.93	0.93	500

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 4, epochs: [128]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

At split estimator: 4, epochs: [128]

Accurace mean(std): 0.9476895048015327(0.03823325699133971)

Classification report for all valid cross_validations against their tests sets

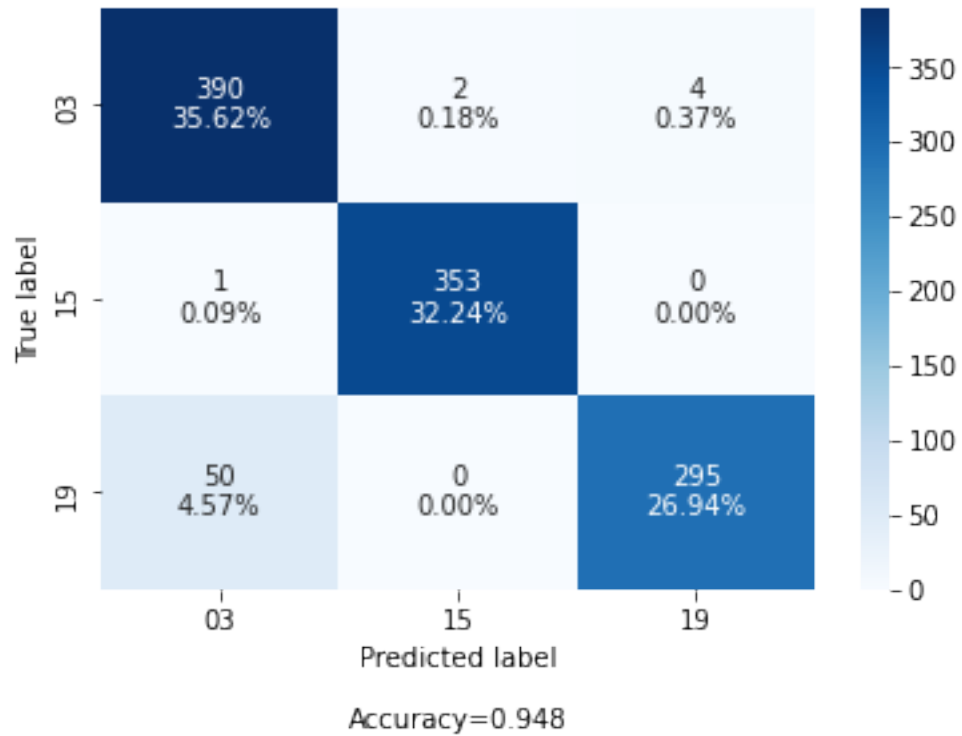
	precision	recall	f1-score	support
03	0.88	0.98	0.93	396
15	0.99	1.00	1.00	354
19	0.99	0.86	0.92	345
accuracy			0.95	1095
macro avg	0.96	0.95	0.95	1095
weighted avg	0.95	0.95	0.95	1095

Classification report for all valid cross_validations against outliers

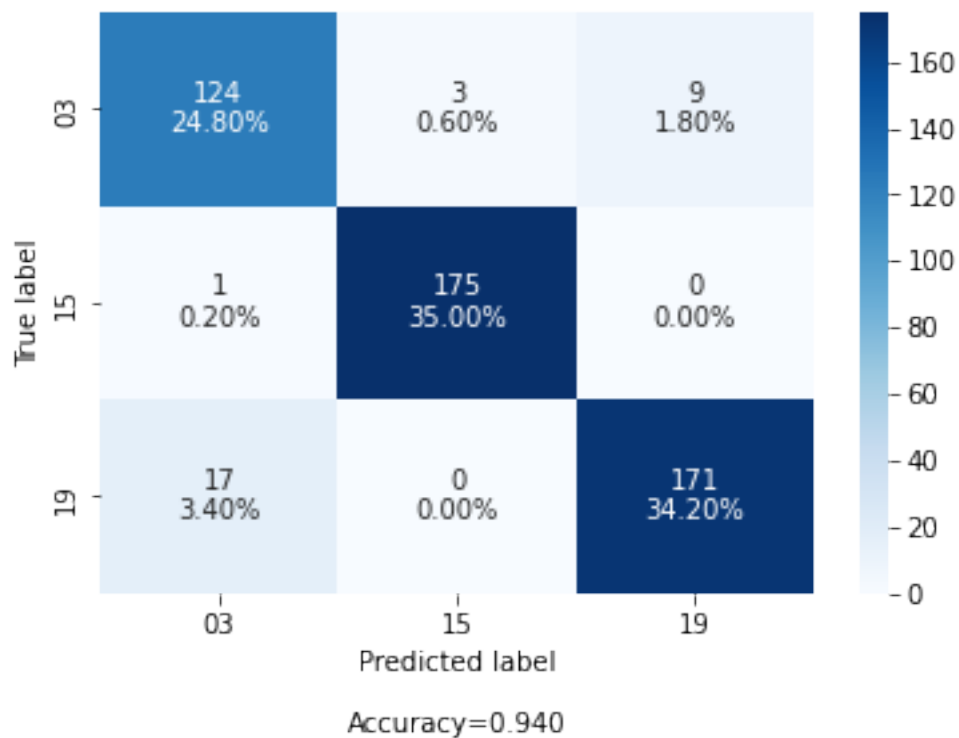
	precision	recall	f1-score	support
03	0.87	0.91	0.89	136
15	0.98	0.99	0.99	176

19	0.95	0.91	0.93	188
accuracy			0.94	500
macro avg	0.94	0.94	0.94	500
weighted avg	0.94	0.94	0.94	500

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 2, epochs: [8]

Processing 1 -fold

Processing 2 -fold

At split estimator: 2, epochs: [8]

Accurace mean(std): 0.9810130385146909(0.007751006429129348)

Classification report for all valid cross_validations against their tests sets

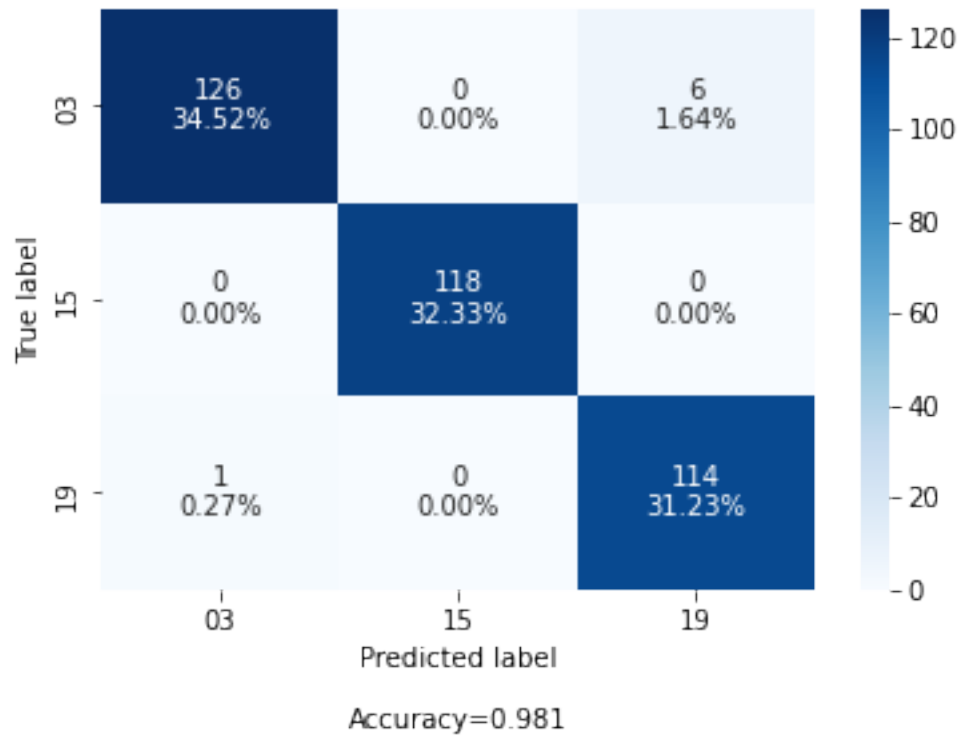
	precision	recall	f1-score	support
03	0.99	0.95	0.97	132
15	1.00	1.00	1.00	118
19	0.95	0.99	0.97	115
accuracy			0.98	365
macro avg	0.98	0.98	0.98	365
weighted avg	0.98	0.98	0.98	365

Classification report for all valid cross_validations against outliers

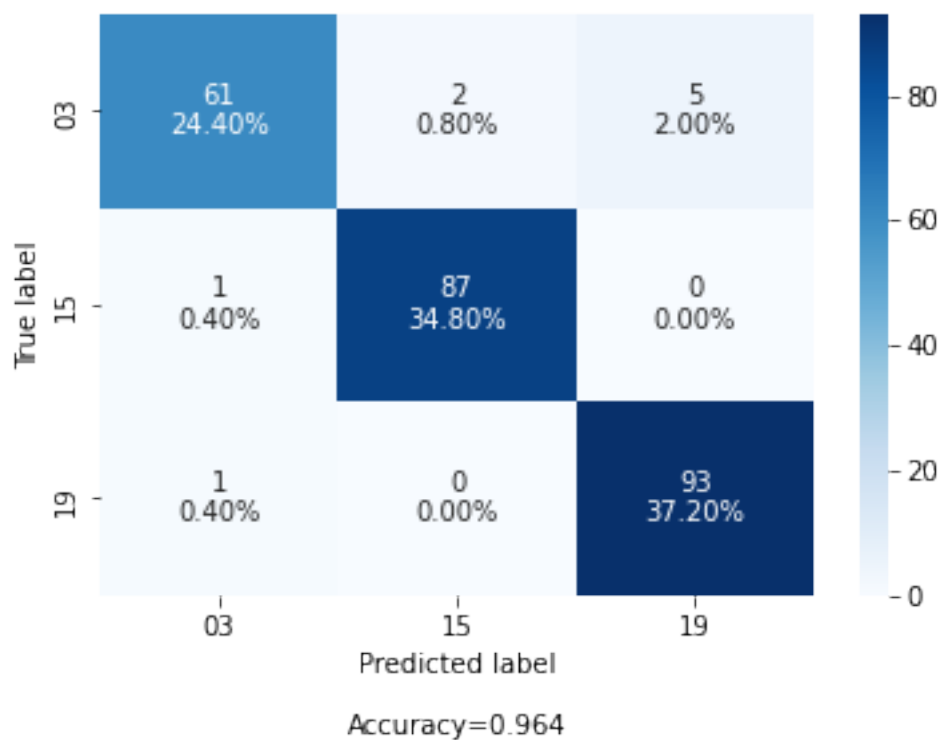
	precision	recall	f1-score	support
03	0.97	0.90	0.93	68
15	0.98	0.99	0.98	88
19	0.95	0.99	0.97	94

accuracy			0.96	250
macro avg	0.96	0.96	0.96	250
weighted avg	0.96	0.96	0.96	250

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 2, epochs: [16]

Processing 1 -fold

Processing 2 -fold

At split estimator: 2, epochs: [16]

Accurace mean(std): 0.986390977443609(0.0021804511278195604)

Classification report for all valid cross_validations against their tests sets

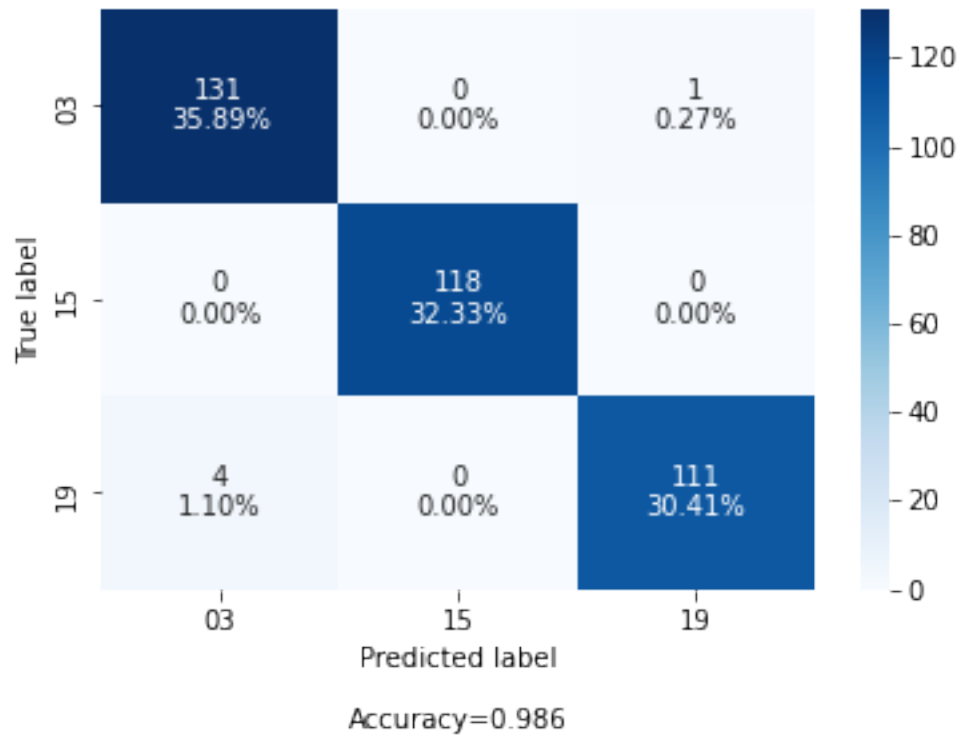
	precision	recall	f1-score	support
03	0.97	0.99	0.98	132
15	1.00	1.00	1.00	118
19	0.99	0.97	0.98	115
accuracy			0.99	365
macro avg	0.99	0.99	0.99	365
weighted avg	0.99	0.99	0.99	365

Classification report for all valid cross_validations against outliers

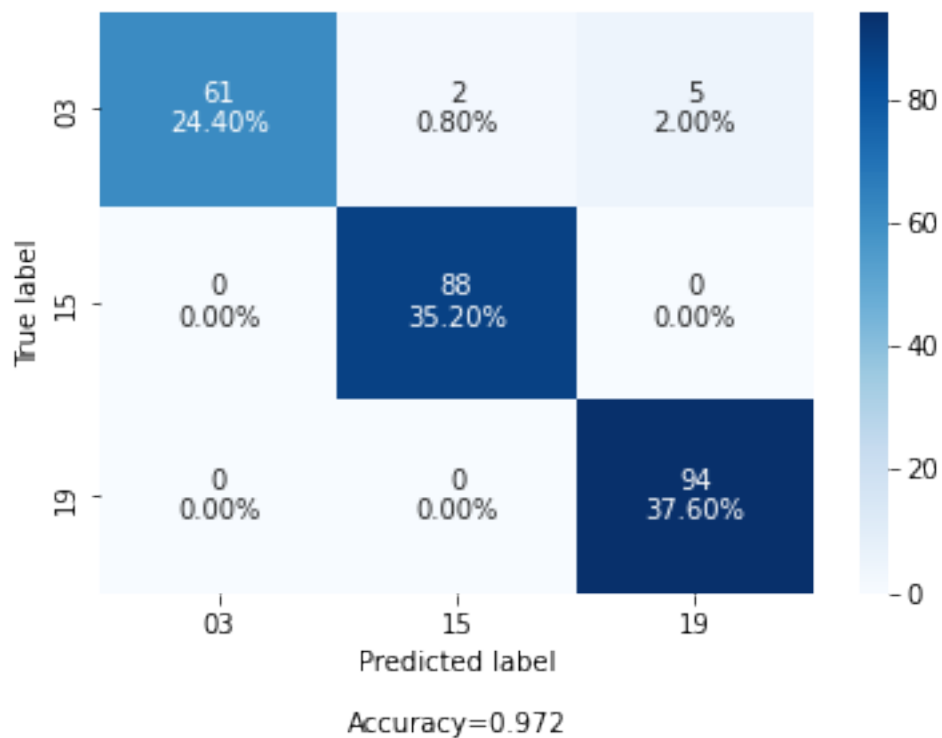
	precision	recall	f1-score	support
03	1.00	0.90	0.95	68
15	0.98	1.00	0.99	88
19	0.95	1.00	0.97	94

accuracy			0.97	250
macro avg	0.98	0.97	0.97	250
weighted avg	0.97	0.97	0.97	250

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 2, epochs: [32]

Processing 1 -fold

Processing 2 -fold

At split estimator: 2, epochs: [32]

Accurace mean(std): 0.9811827956989247(0.018817204301075252)

Classification report for all valid cross_validations against their tests sets

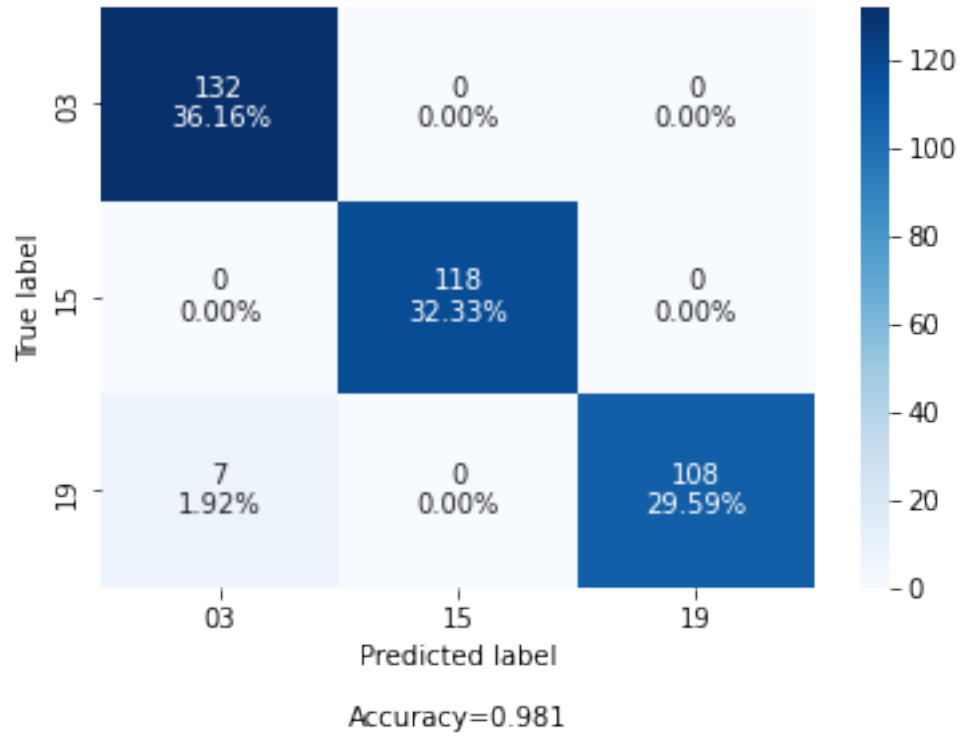
	precision	recall	f1-score	support
03	0.95	1.00	0.97	132
15	1.00	1.00	1.00	118
19	1.00	0.94	0.97	115
accuracy			0.98	365
macro avg	0.98	0.98	0.98	365
weighted avg	0.98	0.98	0.98	365

Classification report for all valid cross_validations against outliers

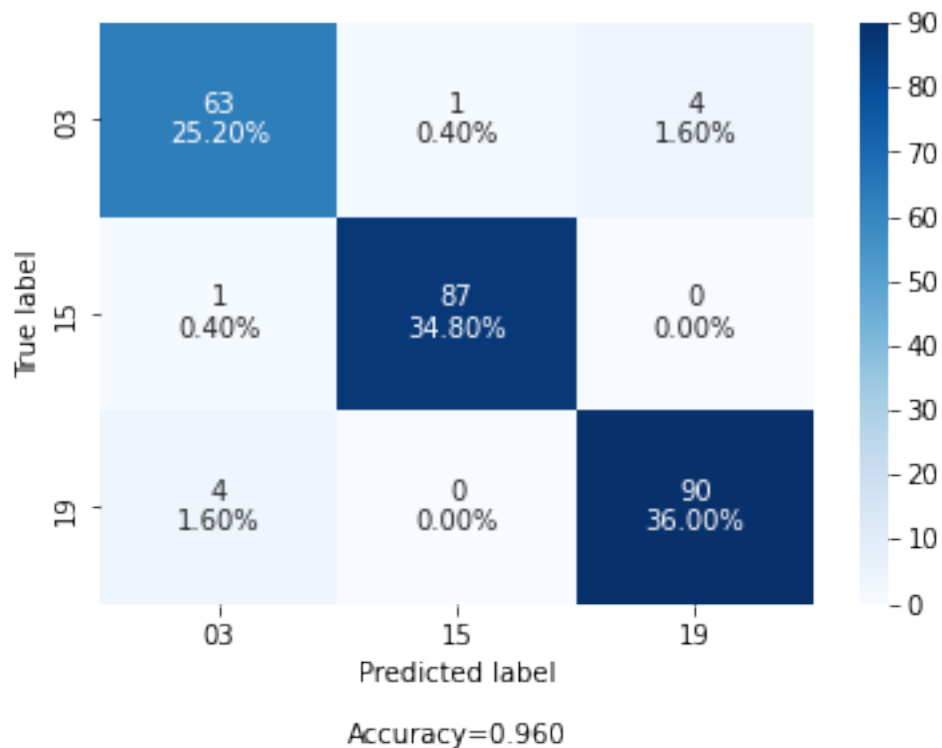
	precision	recall	f1-score	support
03	0.93	0.93	0.93	68
15	0.99	0.99	0.99	88
19	0.96	0.96	0.96	94

accuracy			0.96	250
macro avg	0.96	0.96	0.96	250
weighted avg	0.96	0.96	0.96	250

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 2, epochs: [64]

Processing 1 -fold

Processing 2 -fold

At split estimator: 2, epochs: [64]

Accurace mean(std): 0.9253397065897065(0.05761483886483887)

Classification report for all valid cross_validations against their tests sets

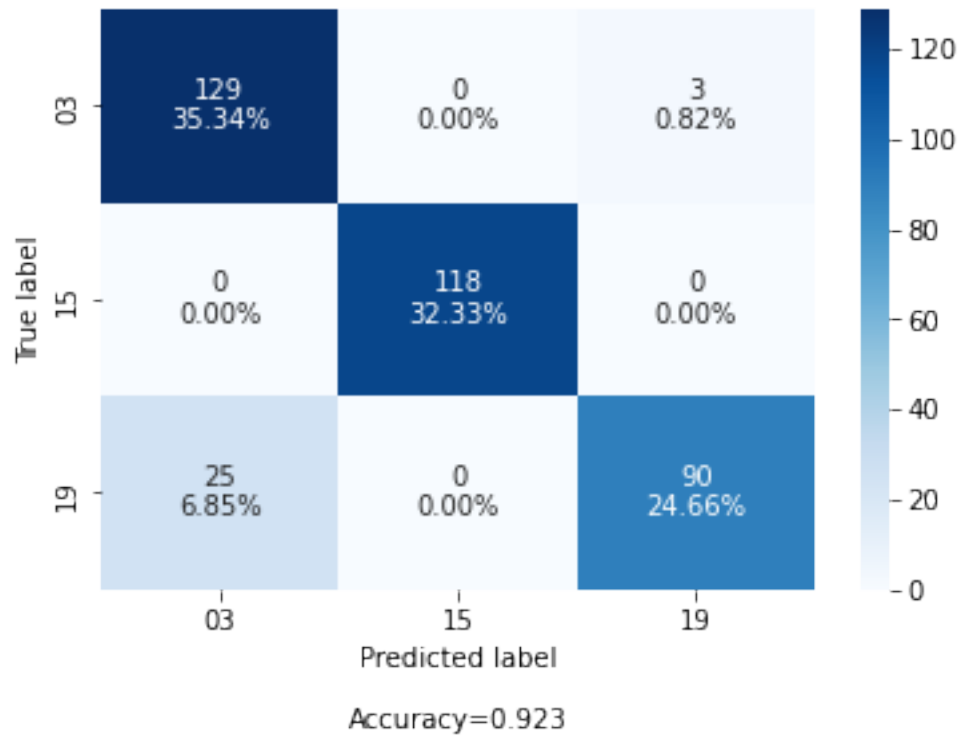
	precision	recall	f1-score	support
03	0.84	0.98	0.90	132
15	1.00	1.00	1.00	118
19	0.97	0.78	0.87	115
accuracy			0.92	365
macro avg	0.94	0.92	0.92	365
weighted avg	0.93	0.92	0.92	365

Classification report for all valid cross_validations against outliers

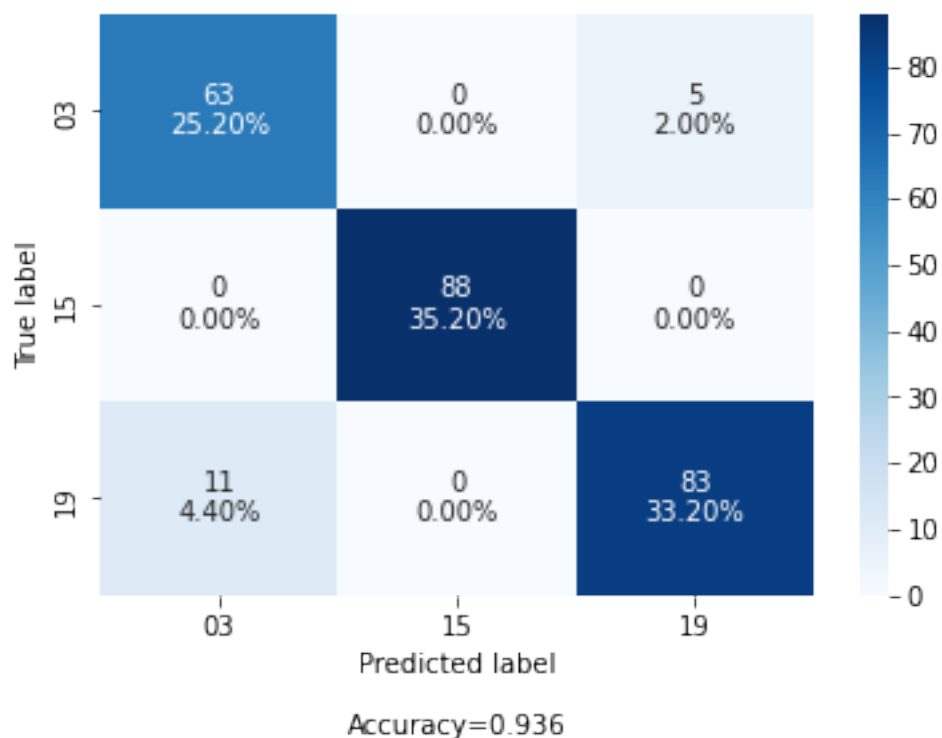
	precision	recall	f1-score	support
03	0.85	0.93	0.89	68
15	1.00	1.00	1.00	88
19	0.94	0.88	0.91	94

accuracy			0.94	250
macro avg	0.93	0.94	0.93	250
weighted avg	0.94	0.94	0.94	250

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for split estimator: 2, epochs: [128]

Processing 1 -fold

Processing 2 -fold

At split estimator: 2, epochs: [128]

Accurace mean(std): 0.9916541353383459(0.003082706766917298)

Classification report for all valid cross_validations against their tests sets

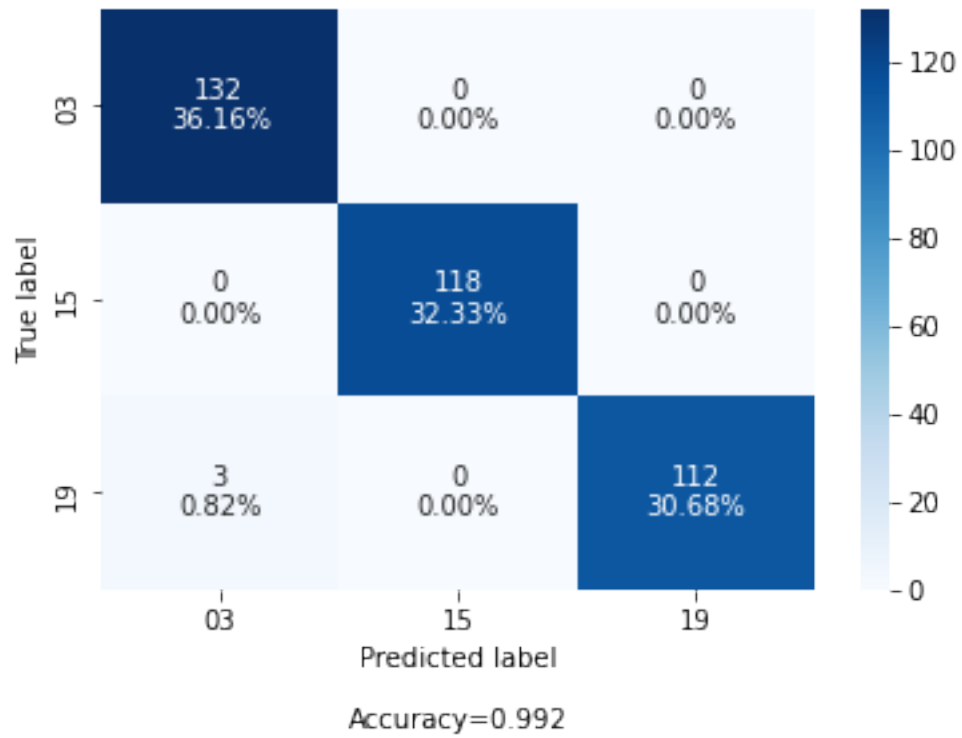
	precision	recall	f1-score	support
03	0.98	1.00	0.99	132
15	1.00	1.00	1.00	118
19	1.00	0.97	0.99	115
accuracy			0.99	365
macro avg	0.99	0.99	0.99	365
weighted avg	0.99	0.99	0.99	365

Classification report for all valid cross_validations against outliers

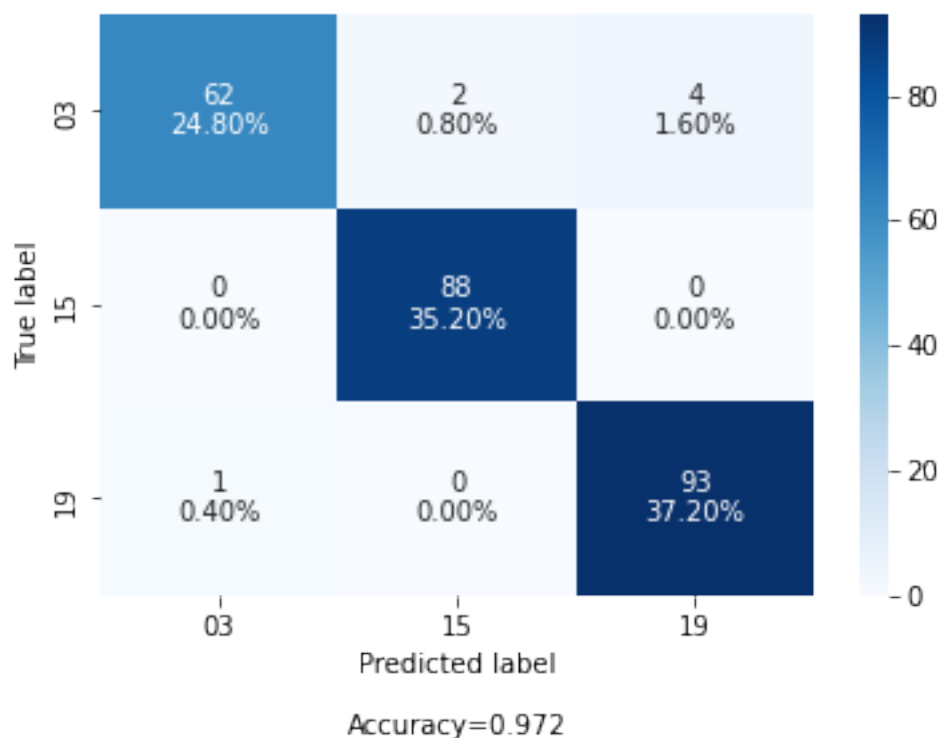
	precision	recall	f1-score	support
03	0.98	0.91	0.95	68
15	0.98	1.00	0.99	88
19	0.96	0.99	0.97	94

accuracy			0.97	250
macro avg	0.97	0.97	0.97	250
weighted avg	0.97	0.97	0.97	250

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for real split: 5, epochs: [8]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [8]

Accurace mean(std): 0.993639406982305(0.008135143443315144)

Classification report for all valid cross_validations against their tests sets

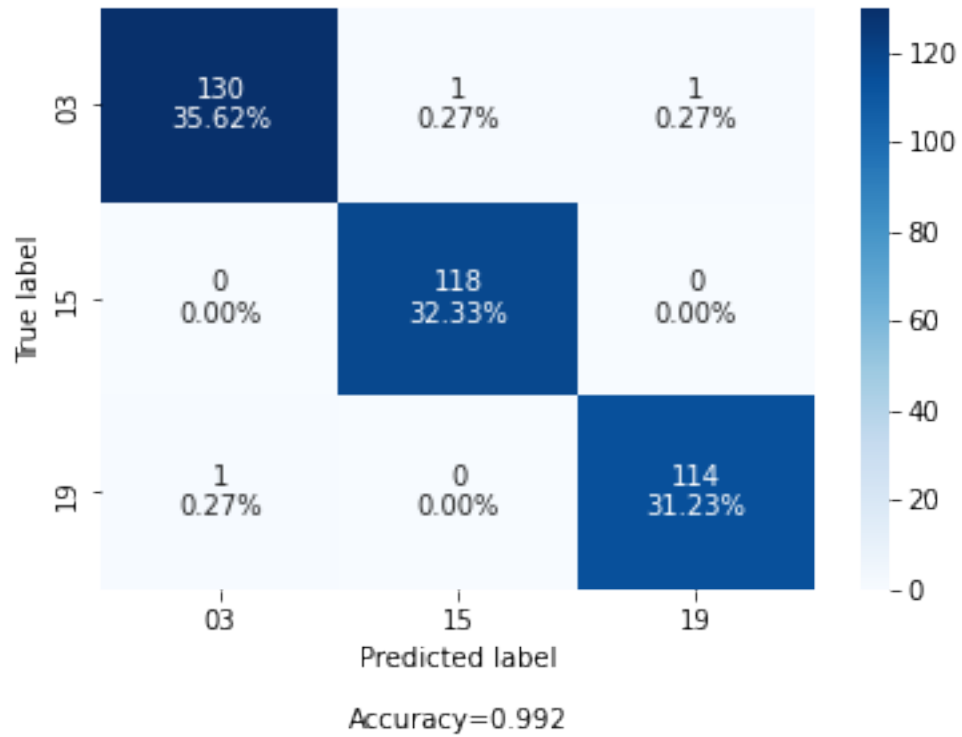
	precision	recall	f1-score	support
03	0.99	0.98	0.99	132
15	0.99	1.00	1.00	118
19	0.99	0.99	0.99	115
accuracy			0.99	365
macro avg	0.99	0.99	0.99	365
weighted avg	0.99	0.99	0.99	365

Classification report for all valid cross_validations against outliers

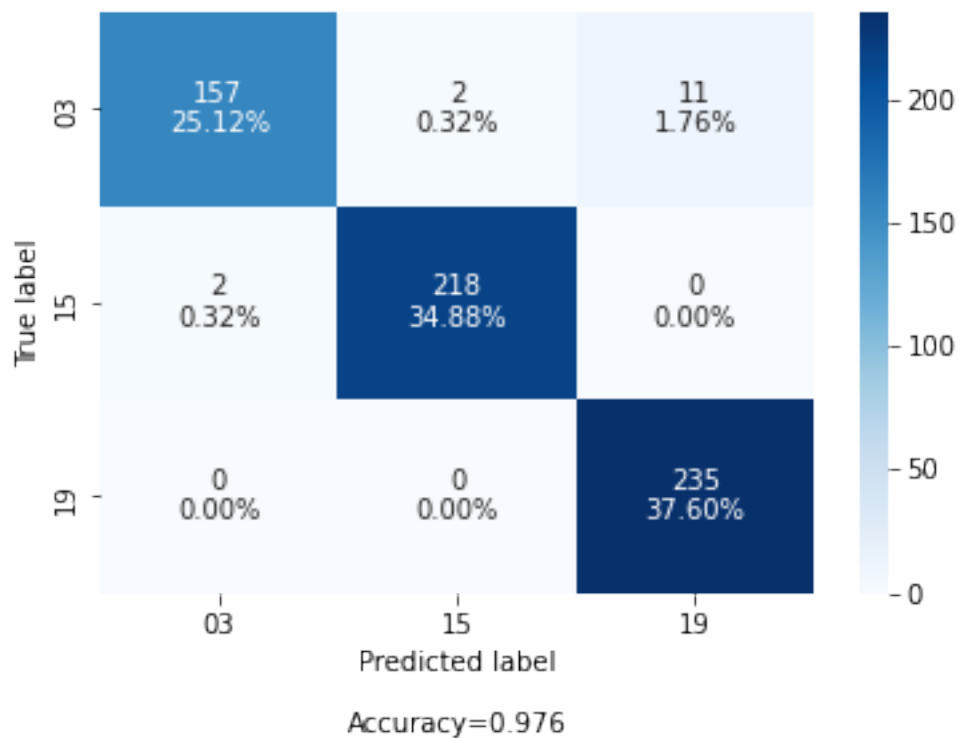
	precision	recall	f1-score	support
03	0.99	0.92	0.95	170

	15	0.99	0.99	0.99	220
	19	0.96	1.00	0.98	235
accuracy				0.98	625
macro avg		0.98	0.97	0.97	625
weighted avg		0.98	0.98	0.98	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for real split: 5, epochs: [16]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [16]

Accurace mean(std): 0.9915565509518476(0.012279610249307016)

Classification report for all valid cross_validations against their tests sets

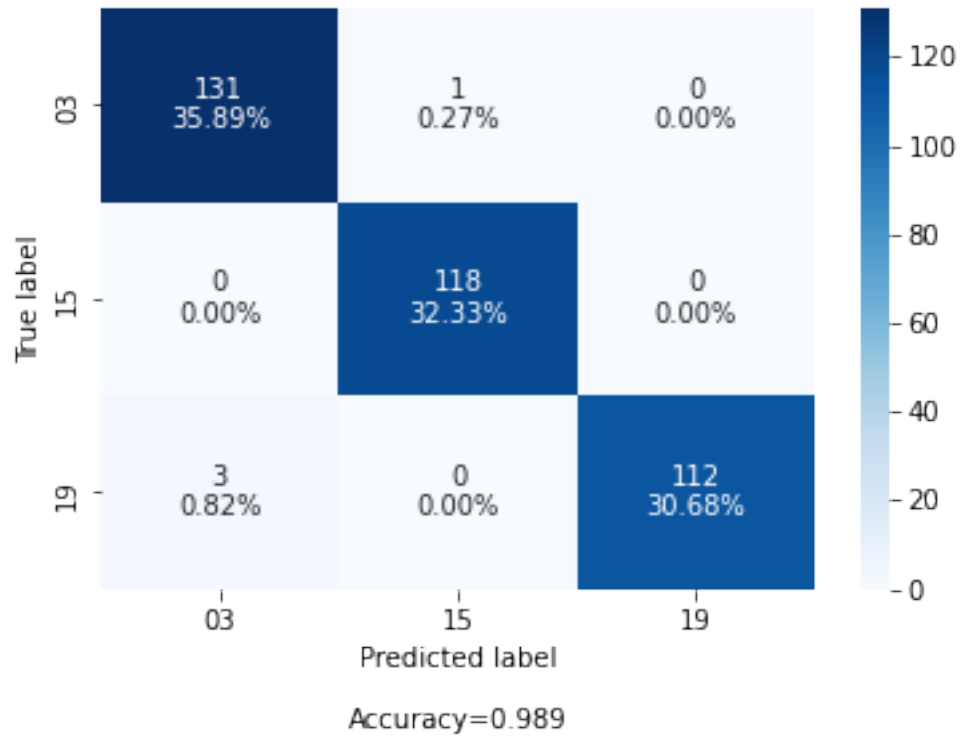
	precision	recall	f1-score	support
03	0.98	0.99	0.98	132
15	0.99	1.00	1.00	118
19	1.00	0.97	0.99	115
accuracy			0.99	365
macro avg	0.99	0.99	0.99	365
weighted avg	0.99	0.99	0.99	365

Classification report for all valid cross_validations against outliers

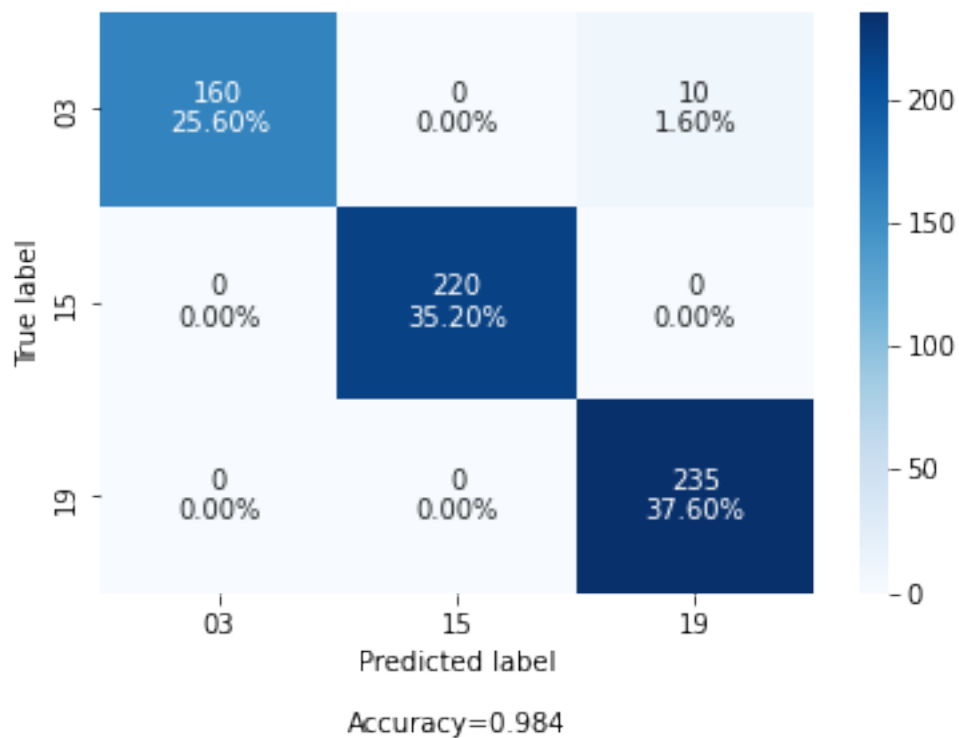
	precision	recall	f1-score	support
03	1.00	0.94	0.97	170

15	1.00	1.00	1.00	220
19	0.96	1.00	0.98	235
accuracy			0.98	625
macro avg	0.99	0.98	0.98	625
weighted avg	0.98	0.98	0.98	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for real split: 5, epochs: [32]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [32]

Accurace mean(std): 0.9953998678122934(0.005636468926569055)

Classification report for all valid cross_validations against their tests sets

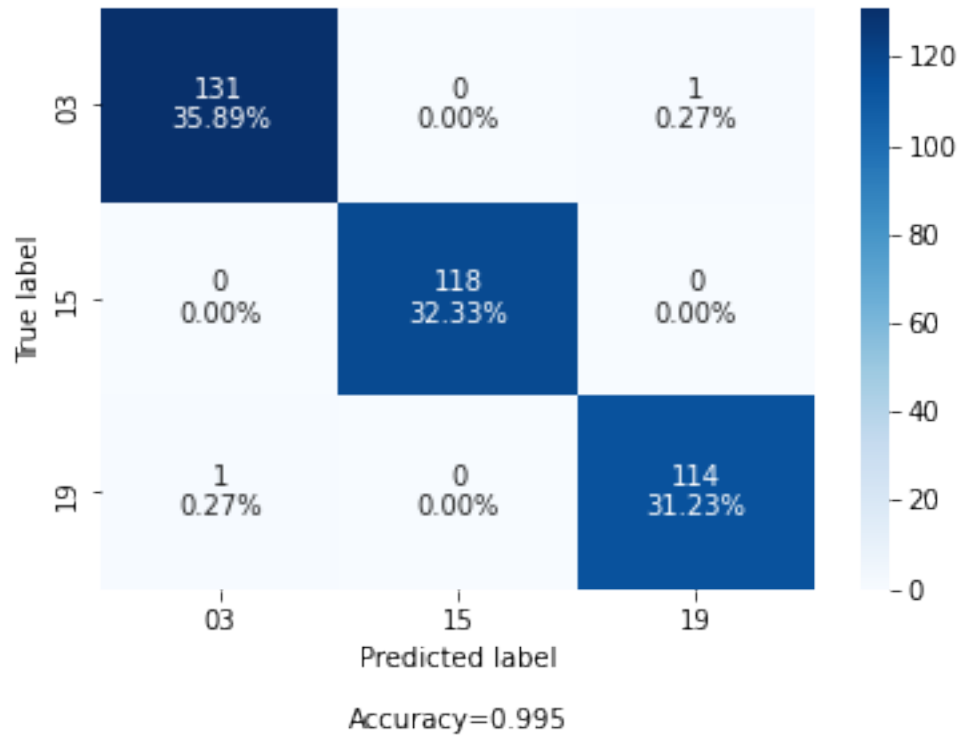
	precision	recall	f1-score	support
03	0.99	0.99	0.99	132
15	1.00	1.00	1.00	118
19	0.99	0.99	0.99	115
accuracy			0.99	365
macro avg	0.99	0.99	0.99	365
weighted avg	0.99	0.99	0.99	365

Classification report for all valid cross_validations against outliers

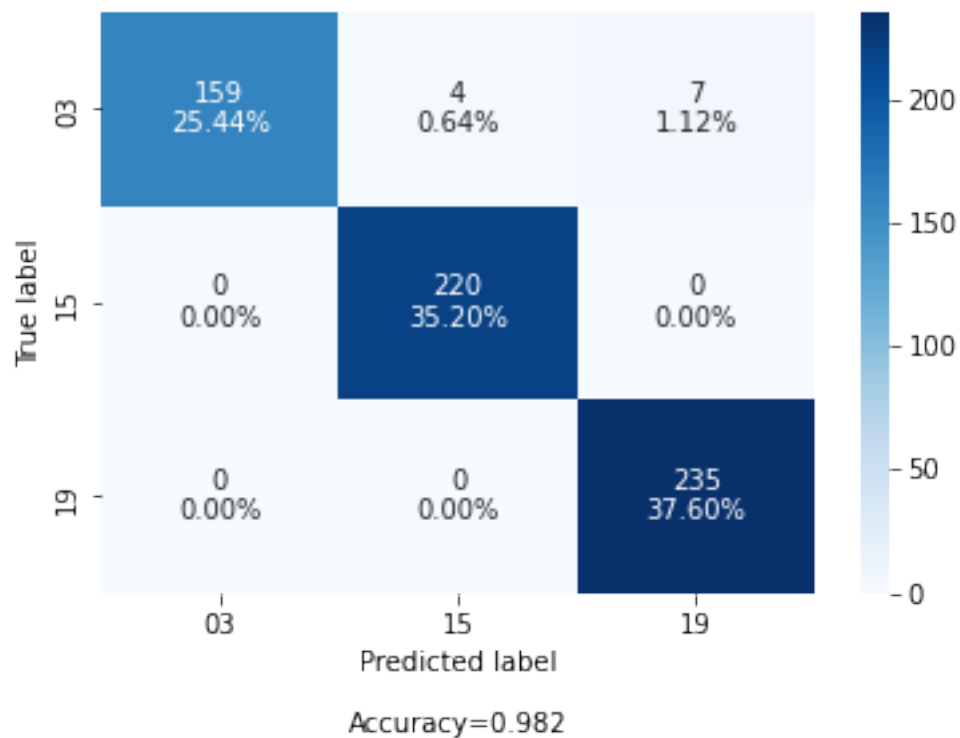
	precision	recall	f1-score	support
03	1.00	0.94	0.97	170

	15	0.98	1.00	0.99	220
	19	0.97	1.00	0.99	235
accuracy				0.98	625
macro avg		0.98	0.98	0.98	625
weighted avg		0.98	0.98	0.98	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for real split: 5, epochs: [64]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [64]

Accurace mean(std): 1.0(0.0)

Classification report for all valid cross_validations against their tests sets

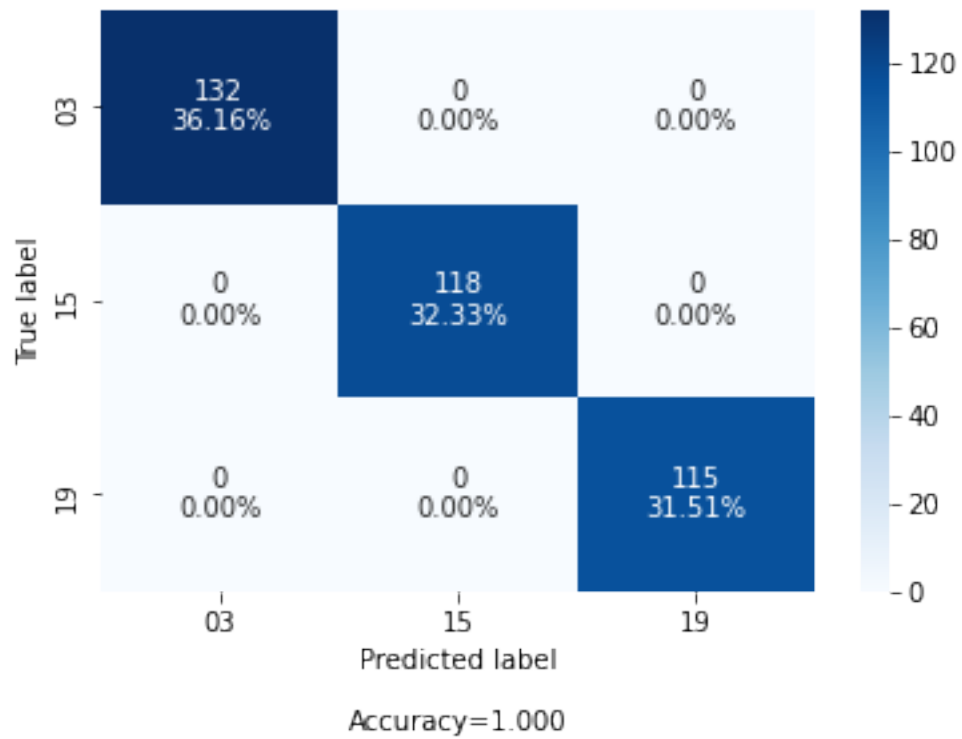
	precision	recall	f1-score	support
03	1.00	1.00	1.00	132
15	1.00	1.00	1.00	118
19	1.00	1.00	1.00	115
accuracy			1.00	365
macro avg	1.00	1.00	1.00	365
weighted avg	1.00	1.00	1.00	365

Classification report for all valid cross_validations against outliers

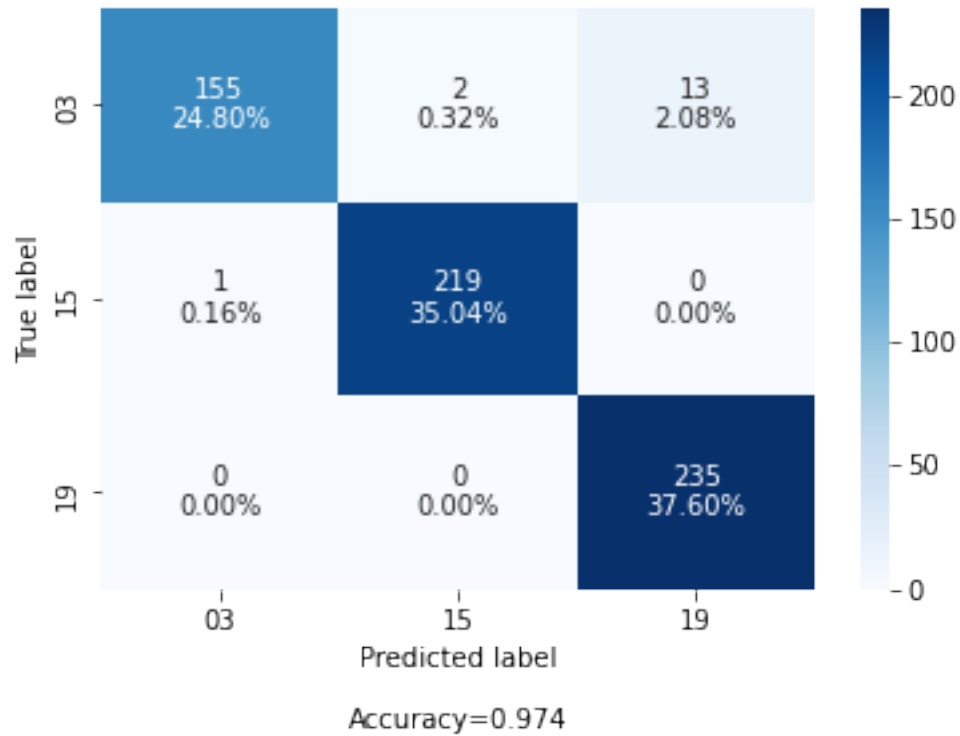
	precision	recall	f1-score	support
03	0.99	0.91	0.95	170

15	0.99	1.00	0.99	220
19	0.95	1.00	0.97	235
accuracy			0.97	625
macro avg	0.98	0.97	0.97	625
weighted avg	0.98	0.97	0.97	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



Processing started for real split: 5, epochs: [128]

Processing 1 -fold

Processing 2 -fold

Processing 3 -fold

Processing 4 -fold

Processing 5 -fold

At split estimator: 5, epochs: [128]

Accurace mean(std): 0.9952941176470589(0.009411764705882364)

Classification report for all valid cross_validations against their tests sets

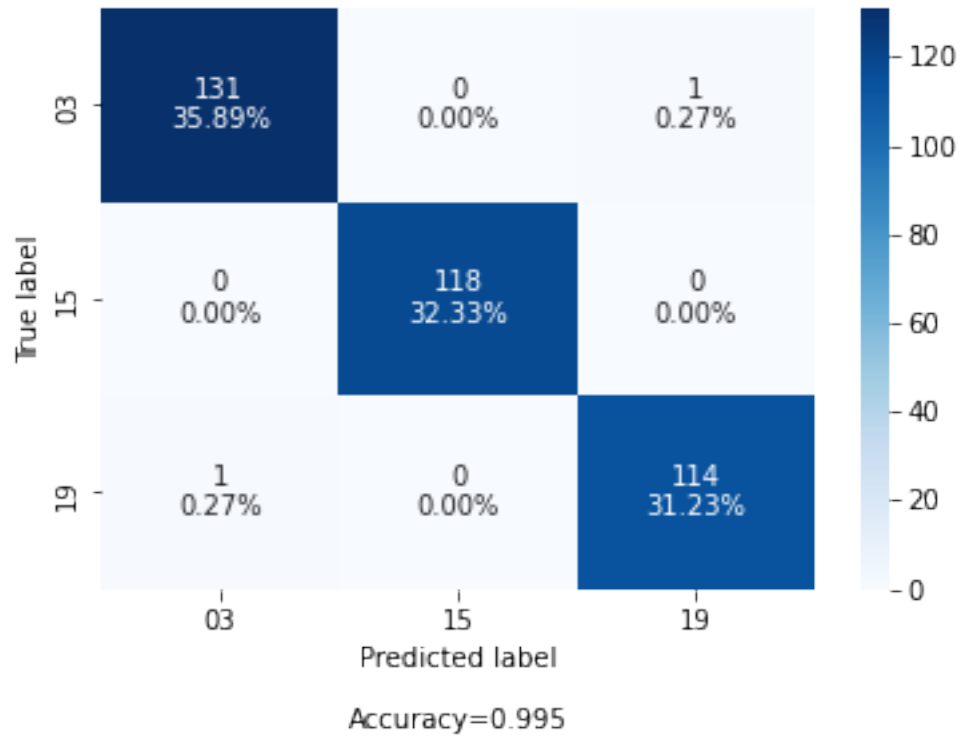
	precision	recall	f1-score	support
03	0.99	0.99	0.99	132
15	1.00	1.00	1.00	118
19	0.99	0.99	0.99	115
accuracy			0.99	365
macro avg	0.99	0.99	0.99	365
weighted avg	0.99	0.99	0.99	365

Classification report for all valid cross_validations against outliers

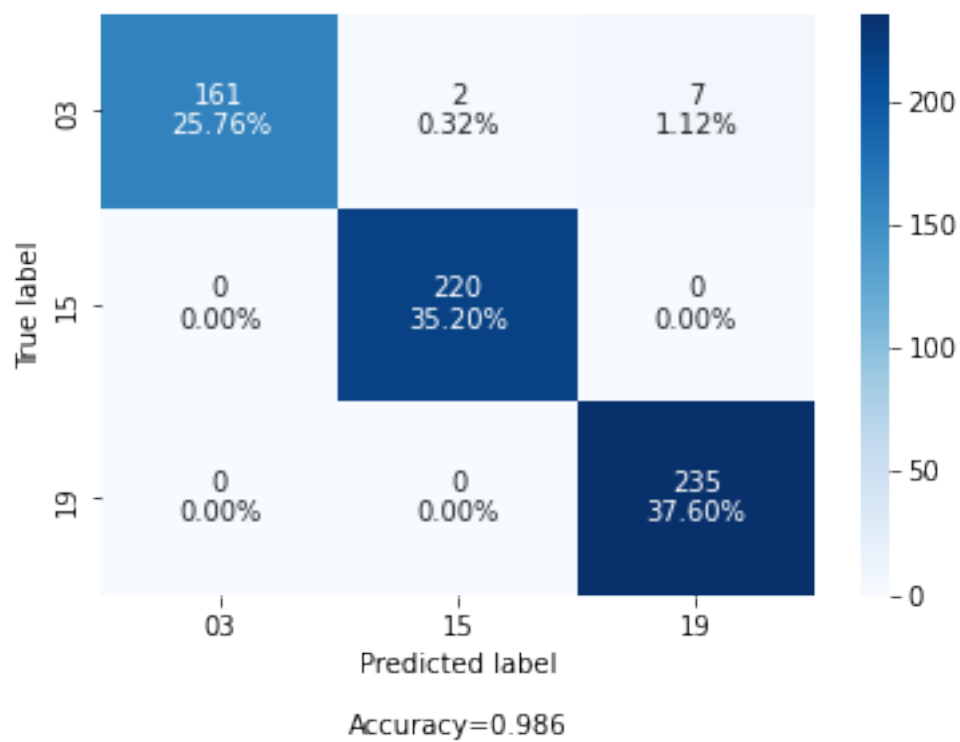
	precision	recall	f1-score	support
03	1.00	0.95	0.97	170

15	0.99	1.00	1.00	220
19	0.97	1.00	0.99	235
accuracy			0.99	625
macro avg	0.99	0.98	0.98	625
weighted avg	0.99	0.99	0.99	625

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers



[]: