transfer-3-3

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

    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(baseset, 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")
```

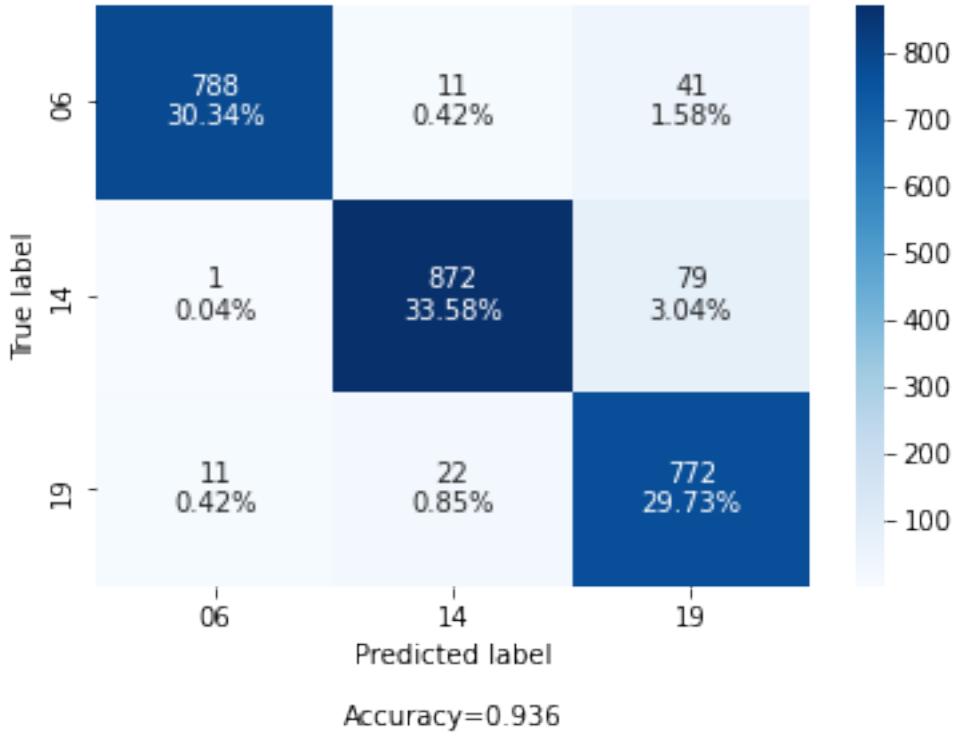
```
            print(classification_report(report_cleaned[0], report_cleaned[1],␣
↪target_names=gesture_subset))

            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[3]

model = create_transfer_models(baseset, dataset)
```

```
Processing tranfers models at 10%, 20%, 25%, 50% and 80% data for gestures:
['06', '14', '19']
Baseset:  ['01', '02', '04', '05', '08', '09', '12', '13', '16', '17', '18',
'20']
Loadind Dataset:  ['06', '14', '19']
494 samples loaded
Scaling Dataset:  ['06', '14', '19']
494 samples scaled
Cleaning Dataset:  ['06', '14', '19']
371 samples cleaned
123 samples outliers
Time slicing Cleaned Dataset:  ['06', '14', '19']
371 cleaned samples sliced
0 cleaned samples damaged
Time slicing Outliers Dataset:  ['06', '14', '19']
123 outliers samples sliced
0 outliers samples damaged

2021-09-21 12:25:03.969976: 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 12:25:03.970004: I tensorflow/stream_executor/cuda/cudart_stub.cc:29]
```

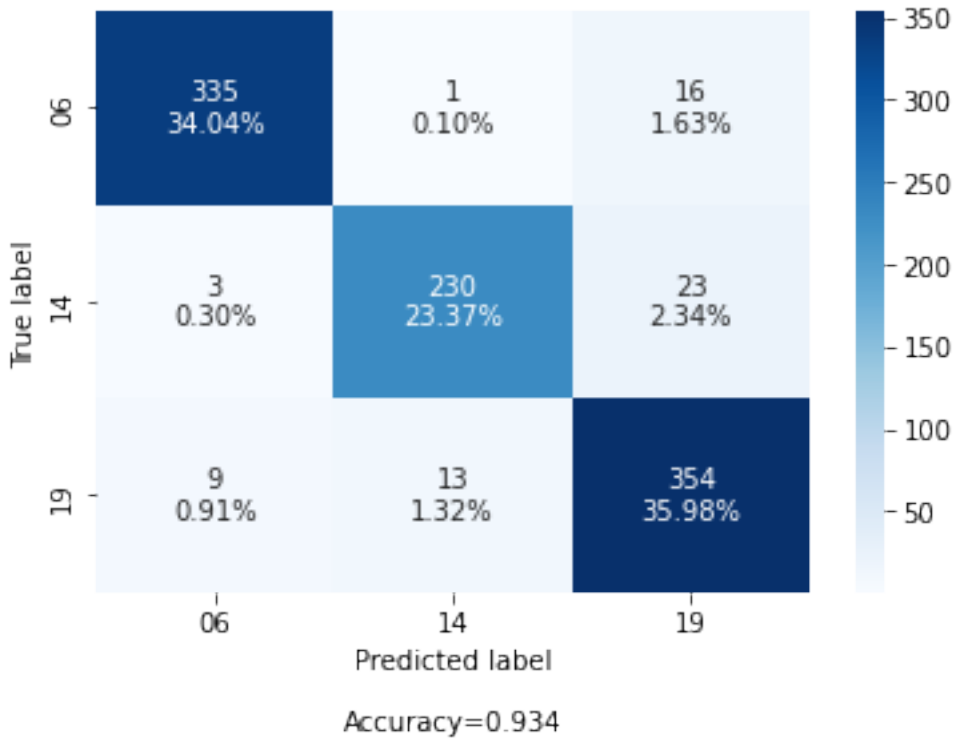
```
Ignore above cudart dlerror if you do not have a GPU set up on your machine.

Processing started for split estimator: 10, epochs: [8]
Processing  1 -fold

2021-09-21 12:25:30.670640: 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 12:25:30.670682: W
tensorflow/stream_executor/cuda/cuda_driver.cc:269] failed call to cuInit:
UNKNOWN ERROR (303)
2021-09-21 12:25:30.670704: 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 12:25:30.671279: 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 12:30:14.040513: 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): 0.9368659442101861(0.0460039084628664)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.98      0.94      0.96       840
          14       0.96      0.92      0.94       952
          19       0.87      0.96      0.91       805

    accuracy                           0.94      2597
   macro avg       0.94      0.94      0.94      2597
weighted avg       0.94      0.94      0.94      2597

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support
```

```
              06       0.97      0.95      0.96       352
              14       0.94      0.90      0.92       256
              19       0.90      0.94      0.92       376

        accuracy                           0.93       984
       macro avg       0.94      0.93      0.93       984
    weighted avg       0.93      0.93      0.93       984
```

Confusion Matrix for all valid cross_validations against their tests sets



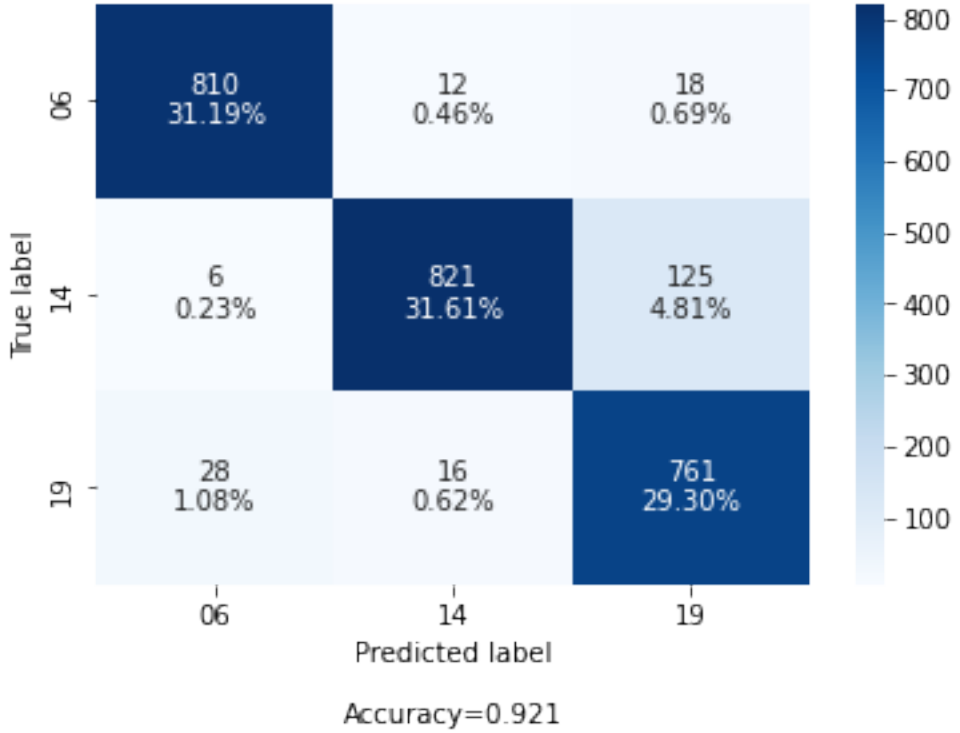
Confusion Matrix for all valid cross_validations against outliers



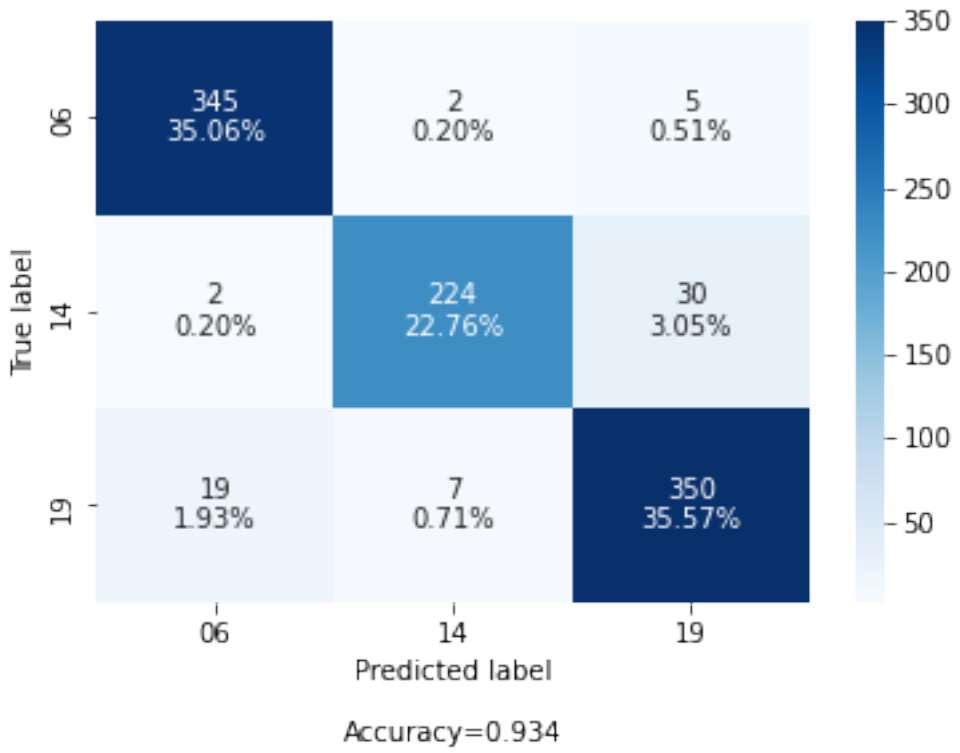

```
Processing started for split estimator: 10, epochs: [16]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
Processing  6 -fold
Processing  7 -fold
Processing  8 -fold
At split estimator: 10, epochs: [16]
Accurace mean(std): 0.9216955706624087(0.0864662730565137)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.96      0.96      0.96       840
          14       0.97      0.86      0.91       952
          19       0.84      0.95      0.89       805

    accuracy                           0.92      2597
   macro avg       0.92      0.92      0.92      2597
weighted avg       0.93      0.92      0.92      2597

Classification report for all valid cross_validations against outliers
```

	precision	recall	f1-score	support
06	0.94	0.98	0.96	352
14	0.96	0.88	0.92	256
19	0.91	0.93	0.92	376
accuracy			0.93	984
macro avg	0.94	0.93	0.93	984
weighted avg	0.93	0.93	0.93	984

Confusion Matrix for all valid cross_validations against their tests sets



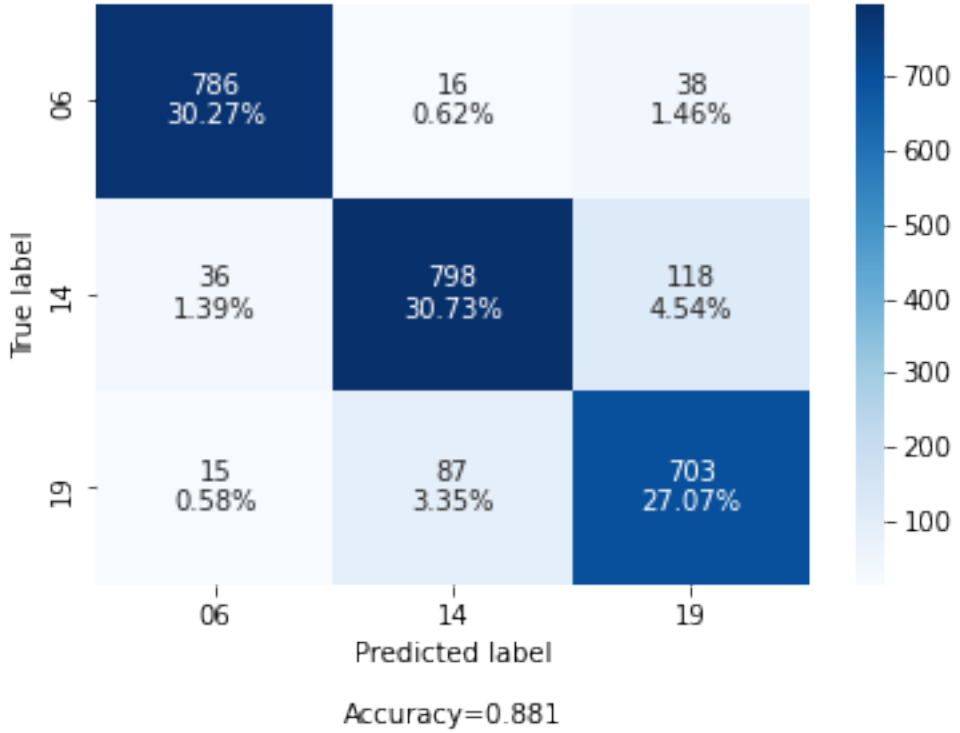
Confusion Matrix for all valid cross_validations against outliers



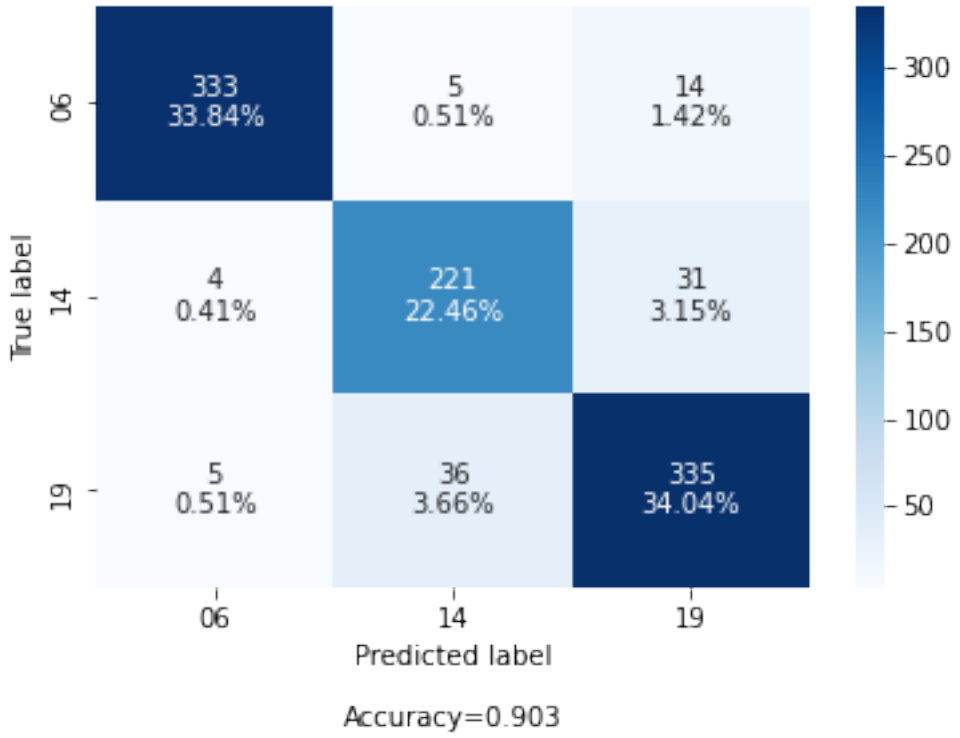

```
Processing started for split estimator: 10, epochs: [32]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
Processing  6 -fold
Processing  7 -fold
Processing  8 -fold
At split estimator: 10, epochs: [32]
Accurace mean(std): 0.8816825111032804(0.09734183138968397)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.94      0.94      0.94       840
          14       0.89      0.84      0.86       952
          19       0.82      0.87      0.84       805

    accuracy                           0.88      2597
   macro avg       0.88      0.88      0.88      2597
weighted avg       0.88      0.88      0.88      2597

Classification report for all valid cross_validations against outliers
```

	precision	recall	f1-score	support
06	0.97	0.95	0.96	352
14	0.84	0.86	0.85	256
19	0.88	0.89	0.89	376
accuracy			0.90	984
macro avg	0.90	0.90	0.90	984
weighted avg	0.90	0.90	0.90	984

Confusion Matrix for all valid cross_validations against their tests sets



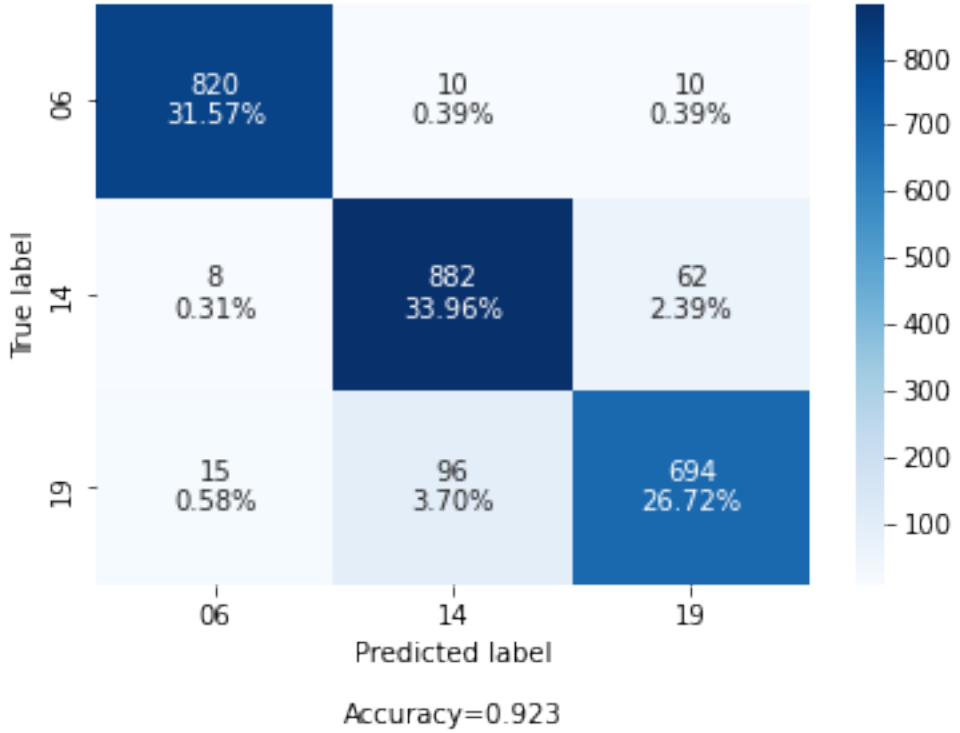
Confusion Matrix for all valid cross_validations against outliers



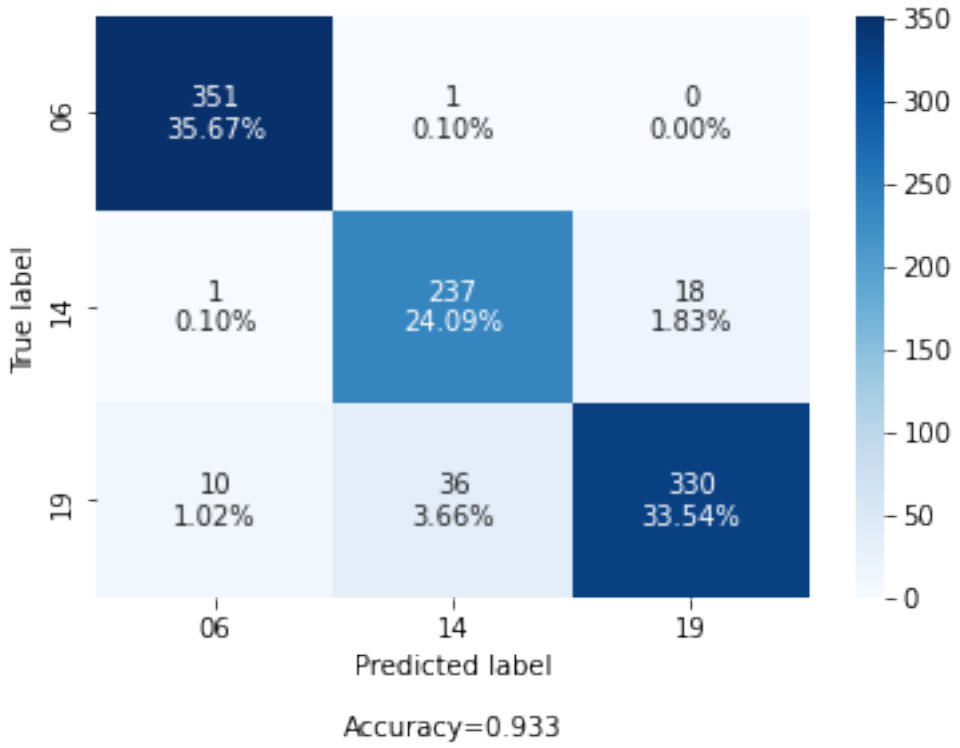

```
Processing started for split estimator: 10, epochs: [64]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
Processing  6 -fold
Processing  7 -fold
Processing  8 -fold
At split estimator: 10, epochs: [64]
Accurace mean(std): 0.9233167473025086(0.09162671032037997)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.97      0.98      0.97       840
          14       0.89      0.93      0.91       952
          19       0.91      0.86      0.88       805

    accuracy                           0.92      2597
   macro avg       0.92      0.92      0.92      2597
weighted avg       0.92      0.92      0.92      2597

Classification report for all valid cross_validations against outliers
```

	precision	recall	f1-score	support
06	0.97	1.00	0.98	352
14	0.86	0.93	0.89	256
19	0.95	0.88	0.91	376
accuracy			0.93	984
macro avg	0.93	0.93	0.93	984
weighted avg	0.93	0.93	0.93	984

Confusion Matrix for all valid cross_validations against their tests sets



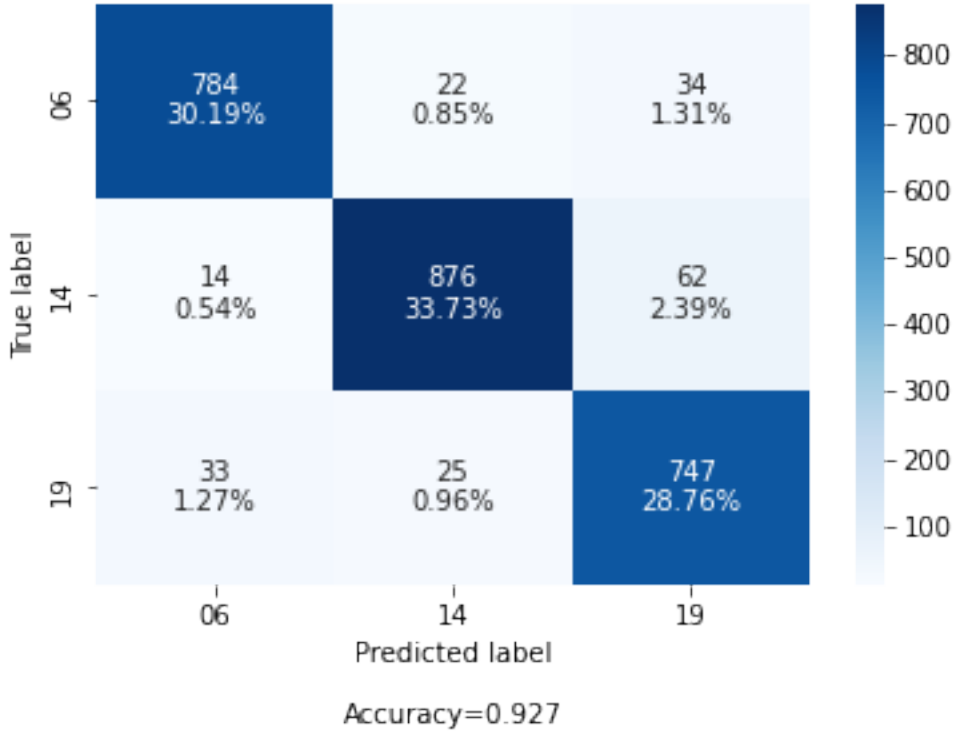
Confusion Matrix for all valid cross_validations against outliers



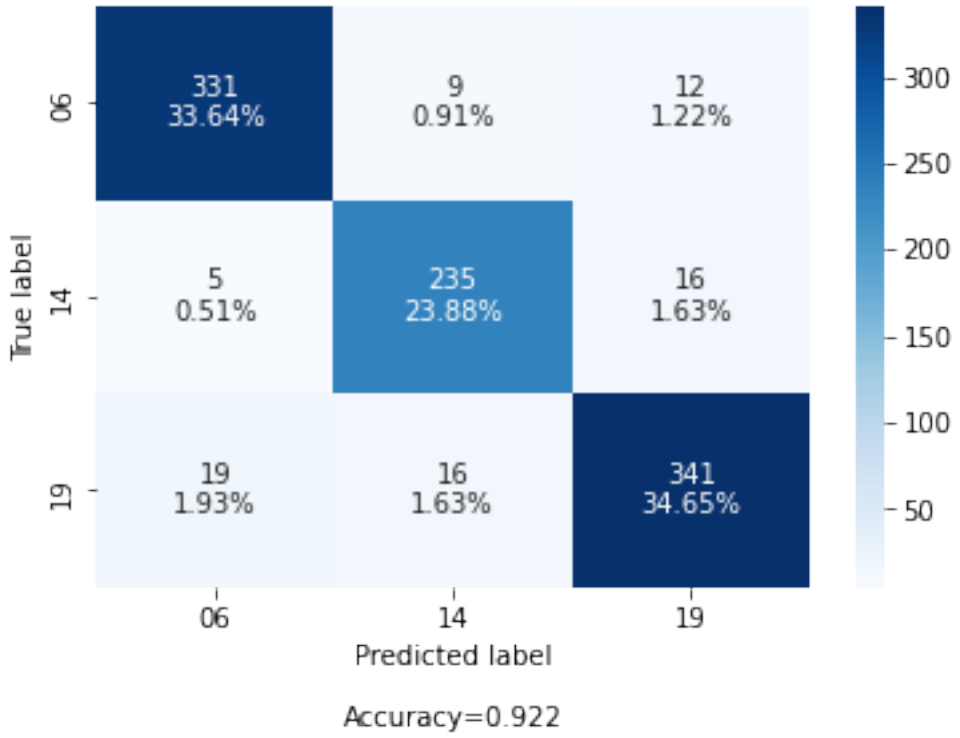

```
Processing started for split estimator: 10, epochs: [128]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
Processing  6 -fold
Processing  7 -fold
Processing  8 -fold
At split estimator: 10, epochs: [128]
Accurace mean(std): 0.9274113552862434(0.05137096953026564)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.94      0.93      0.94       840
          14       0.95      0.92      0.93       952
          19       0.89      0.93      0.91       805

    accuracy                           0.93      2597
   macro avg       0.93      0.93      0.93      2597
weighted avg       0.93      0.93      0.93      2597

Classification report for all valid cross_validations against outliers
```

	precision	recall	f1-score	support
06	0.93	0.94	0.94	352
14	0.90	0.92	0.91	256
19	0.92	0.91	0.92	376
accuracy			0.92	984
macro avg	0.92	0.92	0.92	984
weighted avg	0.92	0.92	0.92	984

Confusion Matrix for all valid cross_validations against their tests sets



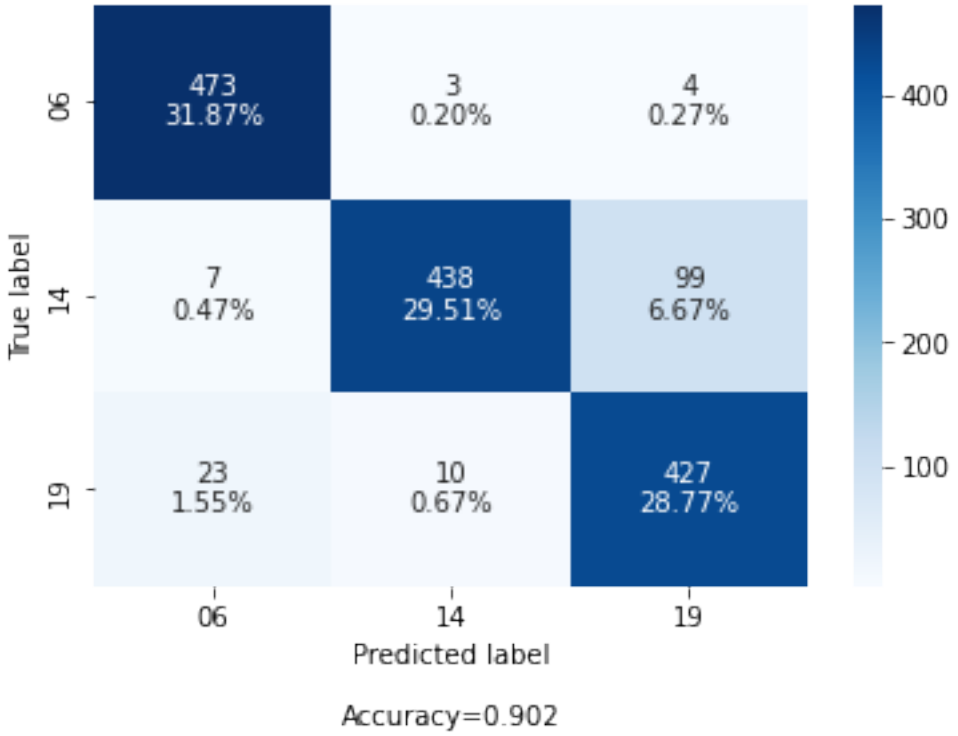
Confusion Matrix for all valid cross_validations against outliers



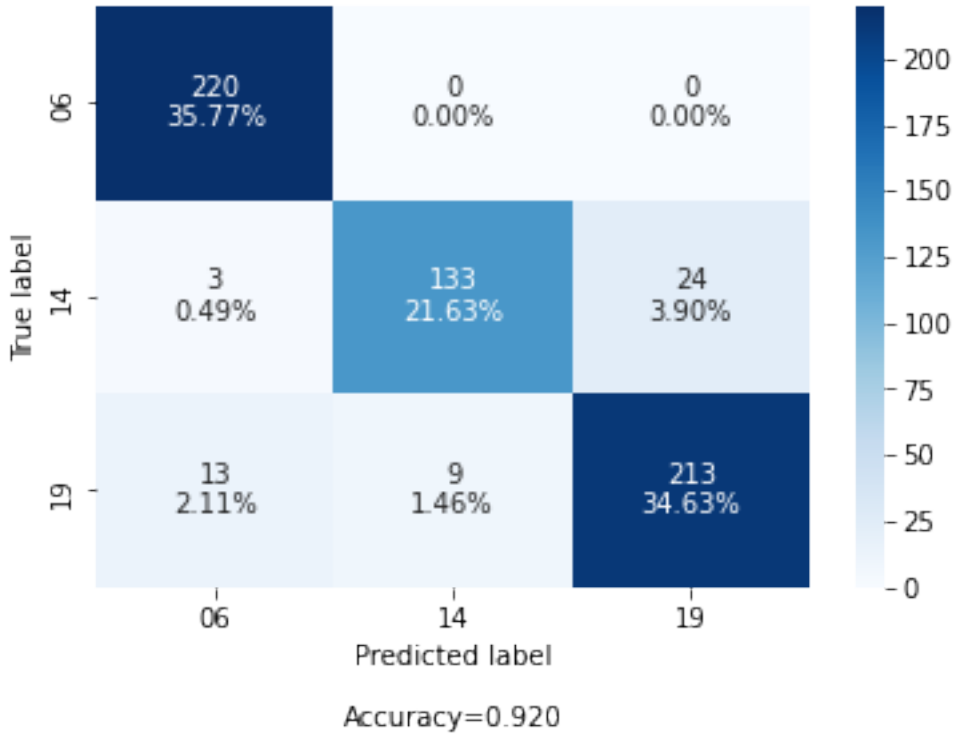

```
Processing started for split estimator: 5, epochs: [8]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [8]
Accurace mean(std): 0.9091230870018039(0.1091279249234166)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.94      0.99      0.96       480
          14       0.97      0.81      0.88       544
          19       0.81      0.93      0.86       460

    accuracy                           0.90      1484
   macro avg       0.91      0.91      0.90      1484
weighted avg       0.91      0.90      0.90      1484

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.93      1.00      0.96       220
```

14	0.94	0.83	0.88	160
19	0.90	0.91	0.90	235
accuracy			0.92	615
macro avg	0.92	0.91	0.92	615
weighted avg	0.92	0.92	0.92	615

Confusion Matrix for all valid cross_validations against their tests sets



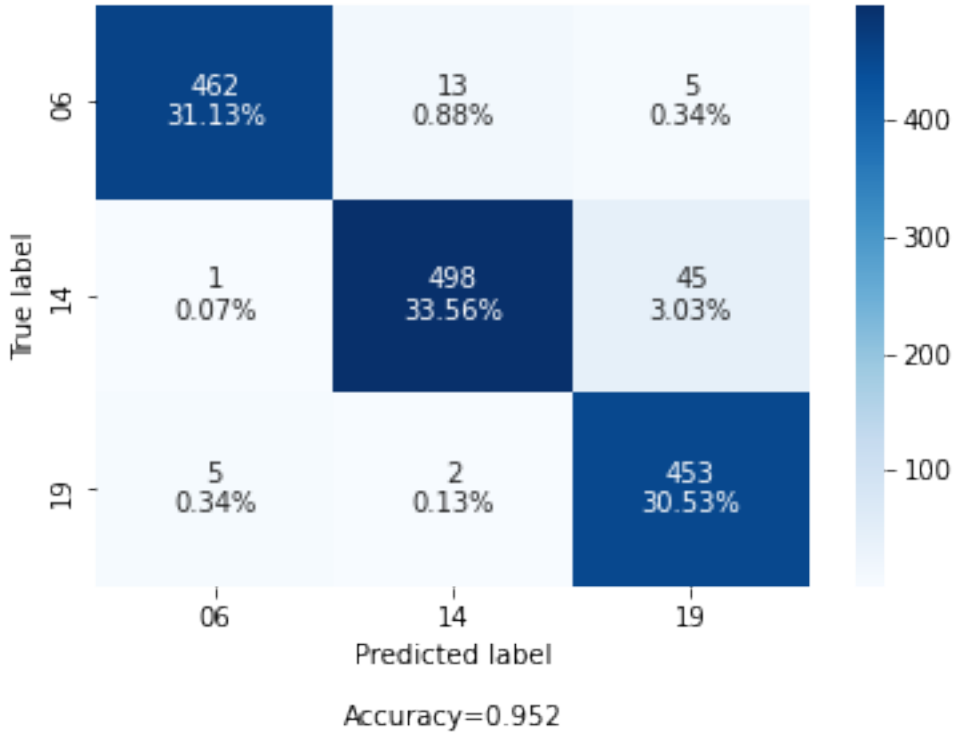
Confusion Matrix for all valid cross_validations against outliers



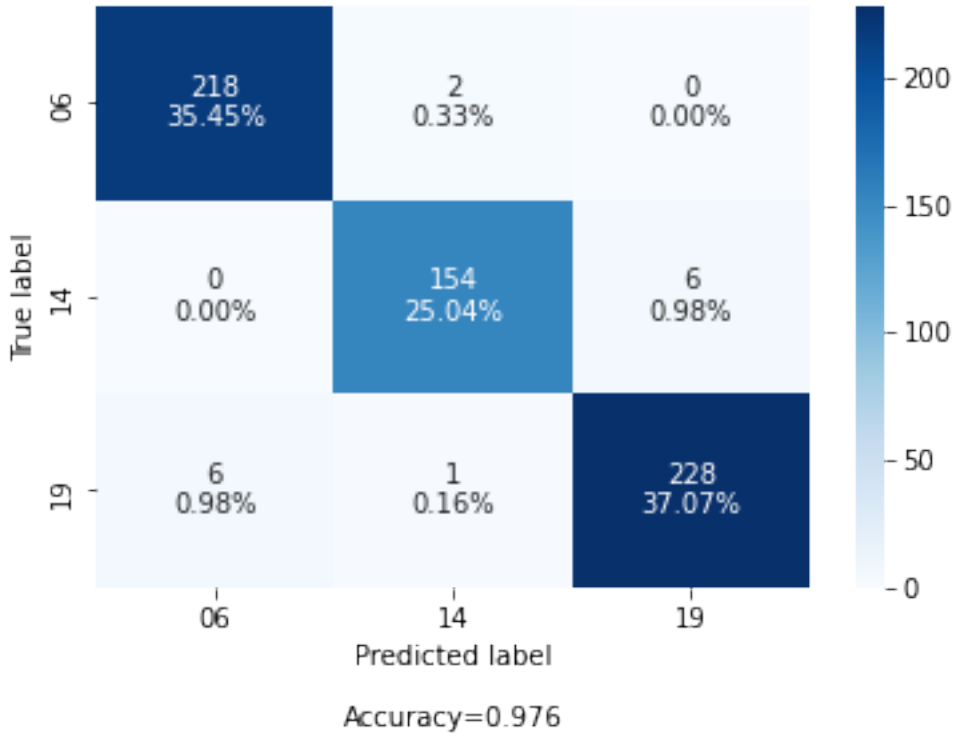

```
Processing started for split estimator: 5, epochs: [16]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [16]
Accurace mean(std): 0.9539334209545605(0.02210916426404485)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.99      0.96      0.97       480
          14       0.97      0.92      0.94       544
          19       0.90      0.98      0.94       460

    accuracy                           0.95      1484
   macro avg       0.95      0.95      0.95      1484
weighted avg       0.95      0.95      0.95      1484

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.97      0.99      0.98       220
```

14	0.98	0.96	0.97	160
19	0.97	0.97	0.97	235
accuracy			0.98	615
macro avg	0.98	0.97	0.98	615
weighted avg	0.98	0.98	0.98	615

Confusion Matrix for all valid cross_validations against their tests sets



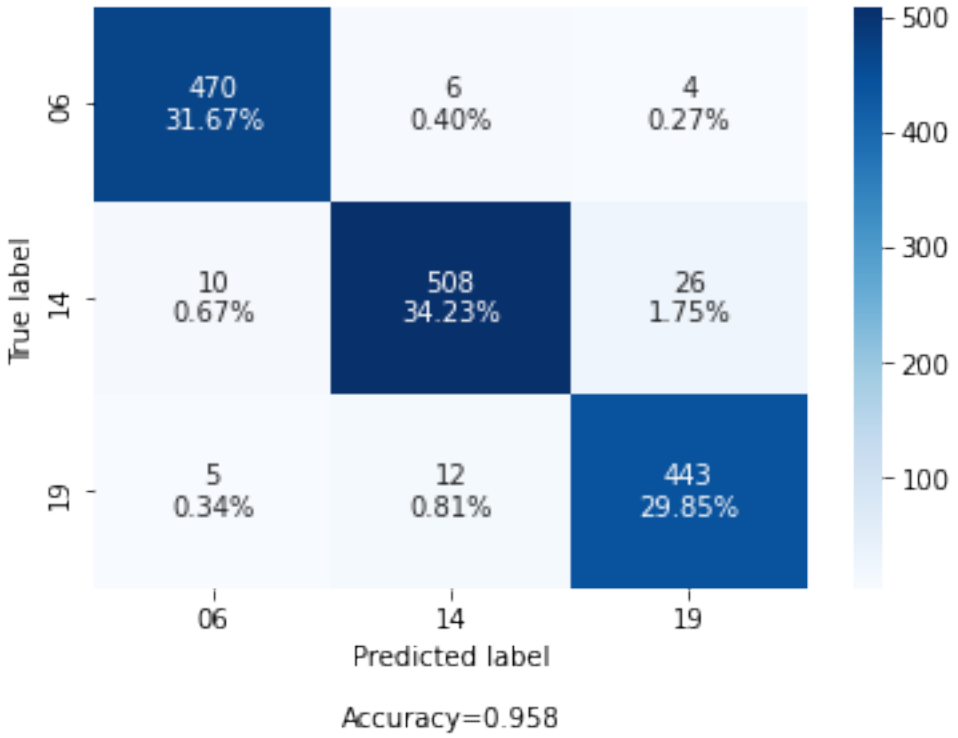
Confusion Matrix for all valid cross_validations against outliers



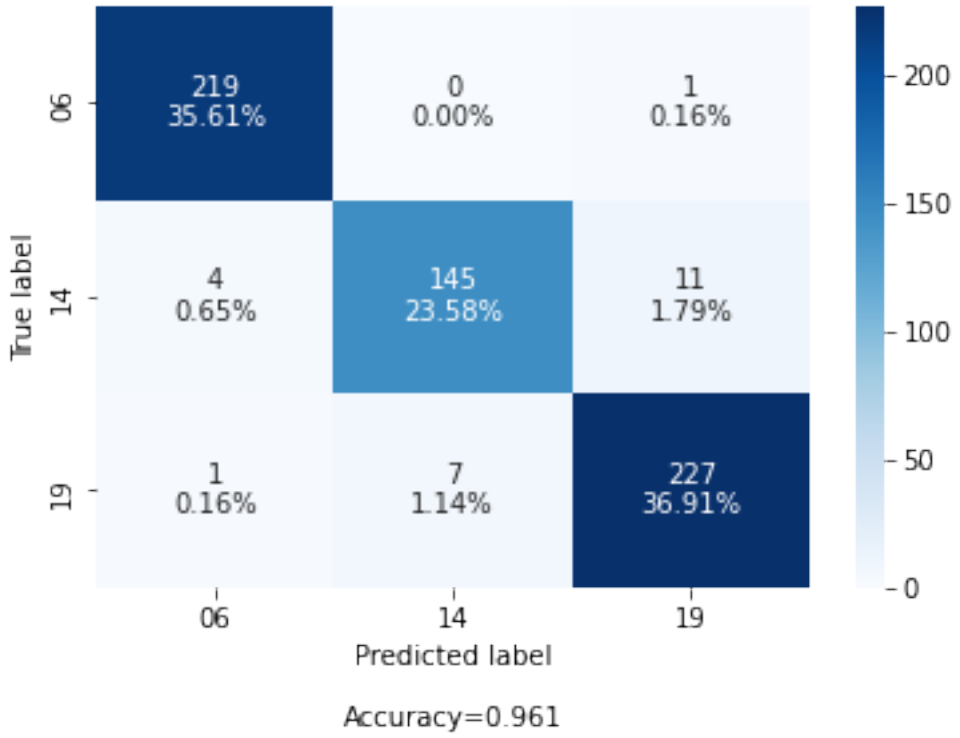

```
Processing started for split estimator: 5, epochs: [32]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [32]
Accurace mean(std): 0.9582473784896628(0.017254415636463245)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.97      0.98      0.97       480
          14       0.97      0.93      0.95       544
          19       0.94      0.96      0.95       460

    accuracy                           0.96      1484
   macro avg       0.96      0.96      0.96      1484
weighted avg       0.96      0.96      0.96      1484

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.98      1.00      0.99       220
```

14	0.95	0.91	0.93	160
19	0.95	0.97	0.96	235
accuracy			0.96	615
macro avg	0.96	0.96	0.96	615
weighted avg	0.96	0.96	0.96	615

Confusion Matrix for all valid cross_validations against their tests sets



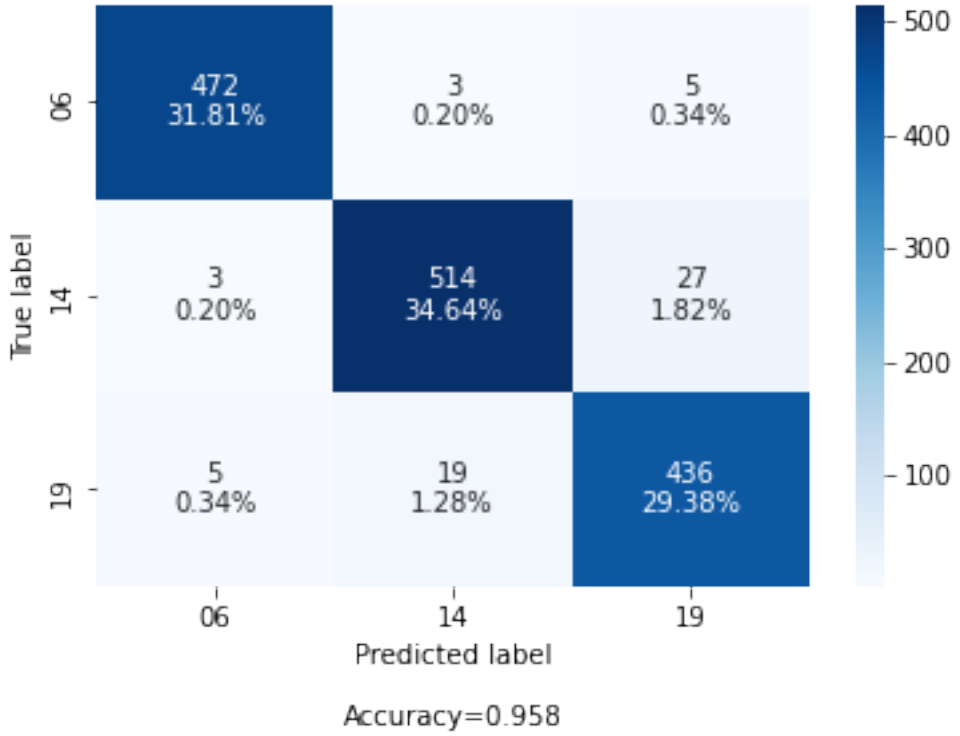
Confusion Matrix for all valid cross_validations against outliers



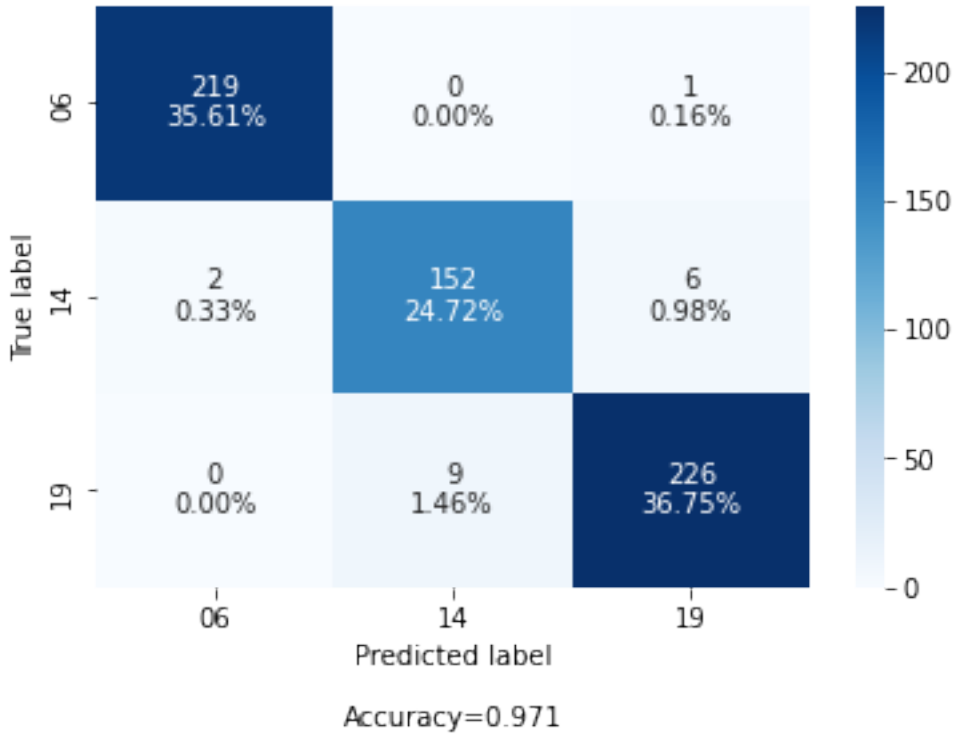

```
Processing started for split estimator: 5, epochs: [64]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [64]
Accurace mean(std): 0.9583337972560881(0.015851956792644385)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.98      0.98      0.98       480
          14       0.96      0.94      0.95       544
          19       0.93      0.95      0.94       460

    accuracy                           0.96      1484
   macro avg       0.96      0.96      0.96      1484
weighted avg       0.96      0.96      0.96      1484

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.99      1.00      0.99       220
```

14	0.94	0.95	0.95	160
19	0.97	0.96	0.97	235
accuracy			0.97	615
macro avg	0.97	0.97	0.97	615
weighted avg	0.97	0.97	0.97	615

Confusion Matrix for all valid cross_validations against their tests sets



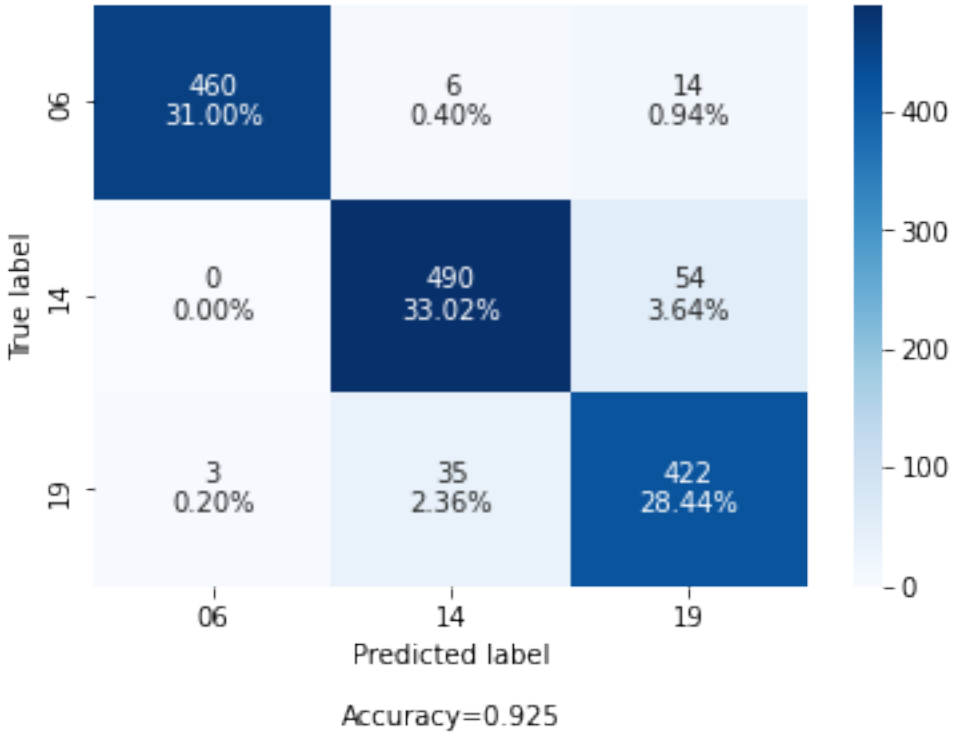
Confusion Matrix for all valid cross_validations against outliers



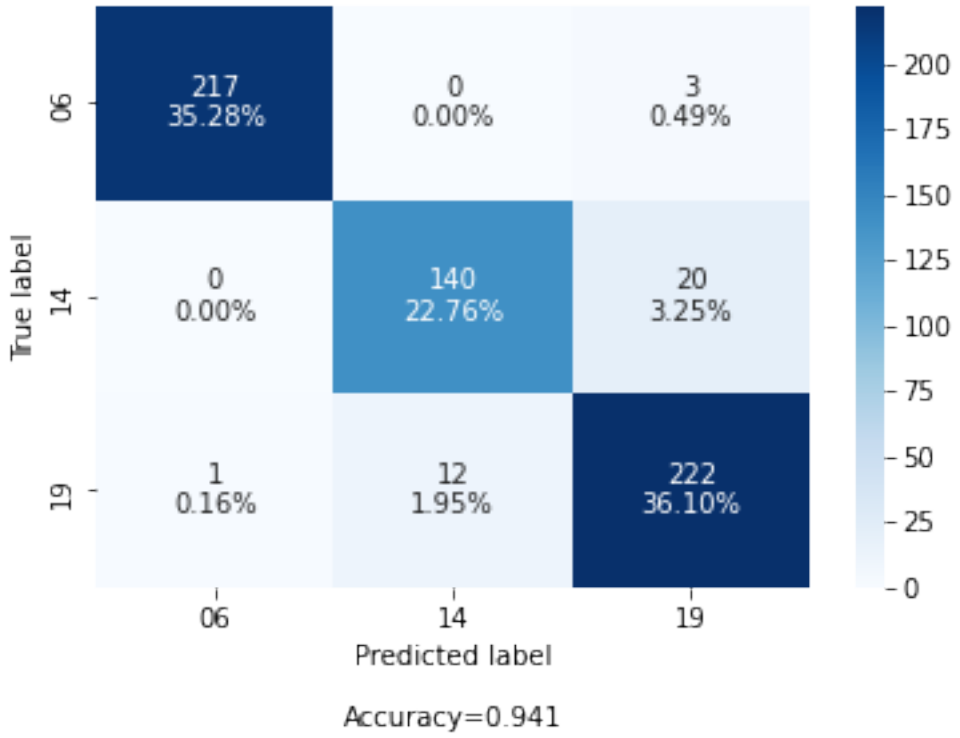

```
Processing started for split estimator: 5, epochs: [128]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [128]
Accurace mean(std): 0.9292149016939655(0.08538875066378908)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.99      0.96      0.98       480
          14       0.92      0.90      0.91       544
          19       0.86      0.92      0.89       460

    accuracy                           0.92      1484
   macro avg       0.93      0.93      0.93      1484
weighted avg       0.93      0.92      0.93      1484

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      0.99      0.99       220
```

```
              14       0.92      0.88      0.90       160
              19       0.91      0.94      0.92       235

        accuracy                           0.94       615
       macro avg       0.94      0.94      0.94       615
    weighted avg       0.94      0.94      0.94       615
```

Confusion Matrix for all valid cross_validations against their tests sets



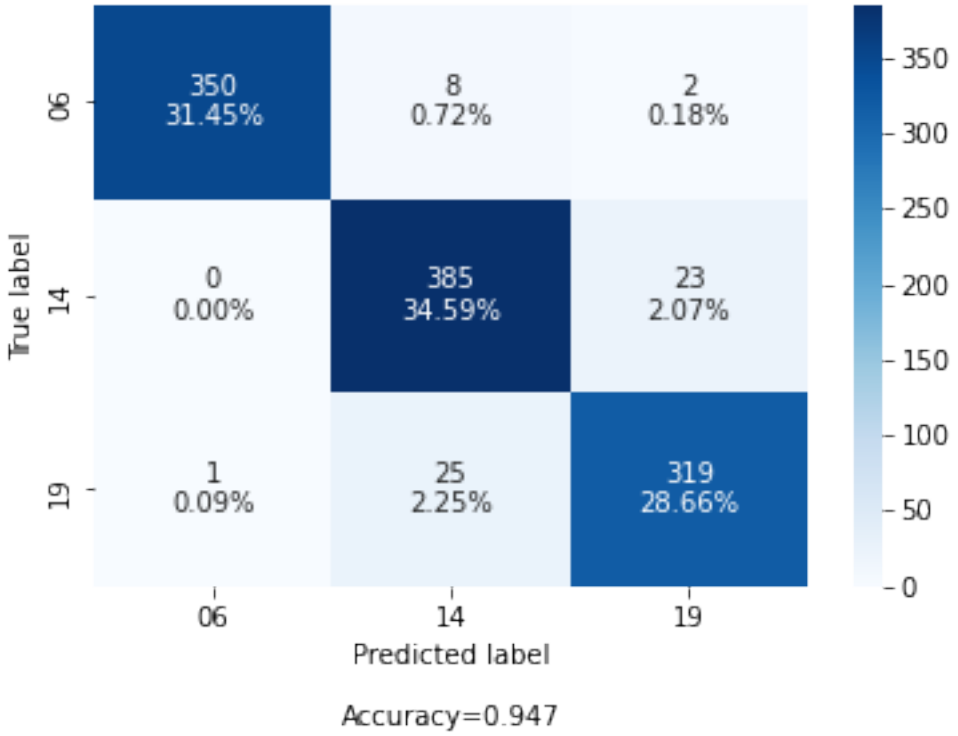
Confusion Matrix for all valid cross_validations against outliers



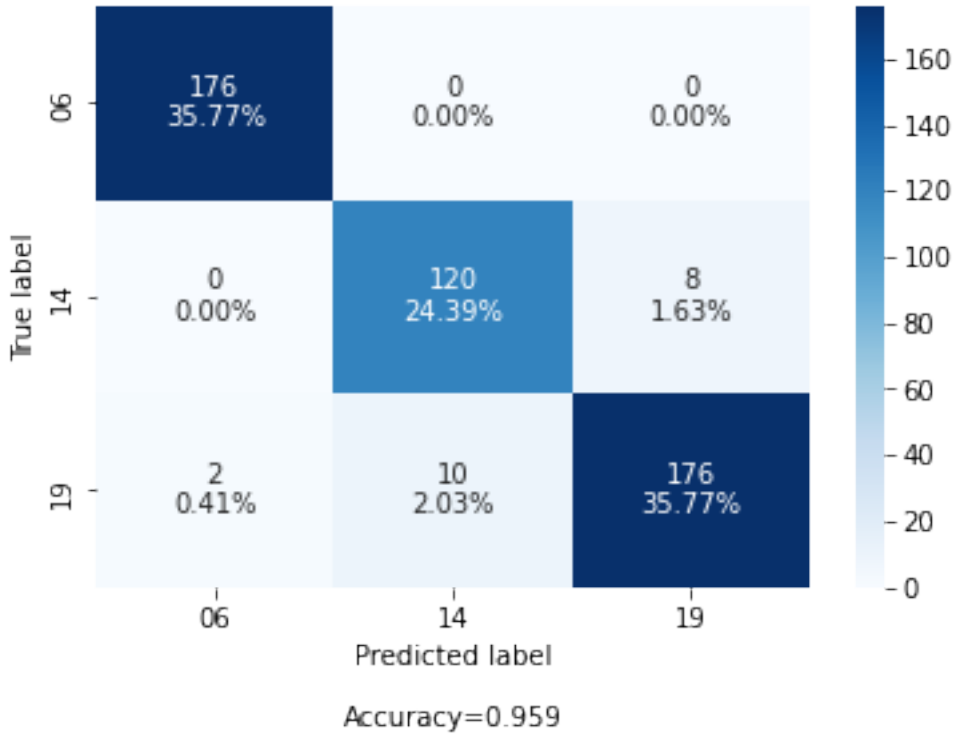

```
Processing started for split estimator: 4, epochs: [8]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
At split estimator: 4, epochs: [8]
Accurace mean(std): 0.9478742027301058(0.041876990352357306)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       1.00      0.97      0.98       360
          14       0.92      0.94      0.93       408
          19       0.93      0.92      0.93       345

    accuracy                           0.95      1113
   macro avg       0.95      0.95      0.95      1113
weighted avg       0.95      0.95      0.95      1113

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.99      1.00      0.99       176
          14       0.92      0.94      0.93       128
```

```
          19       0.96      0.94      0.95       188

    accuracy                           0.96       492
   macro avg       0.96      0.96      0.96       492
weighted avg       0.96      0.96      0.96       492
```

Confusion Matrix for all valid cross_validations against their tests sets



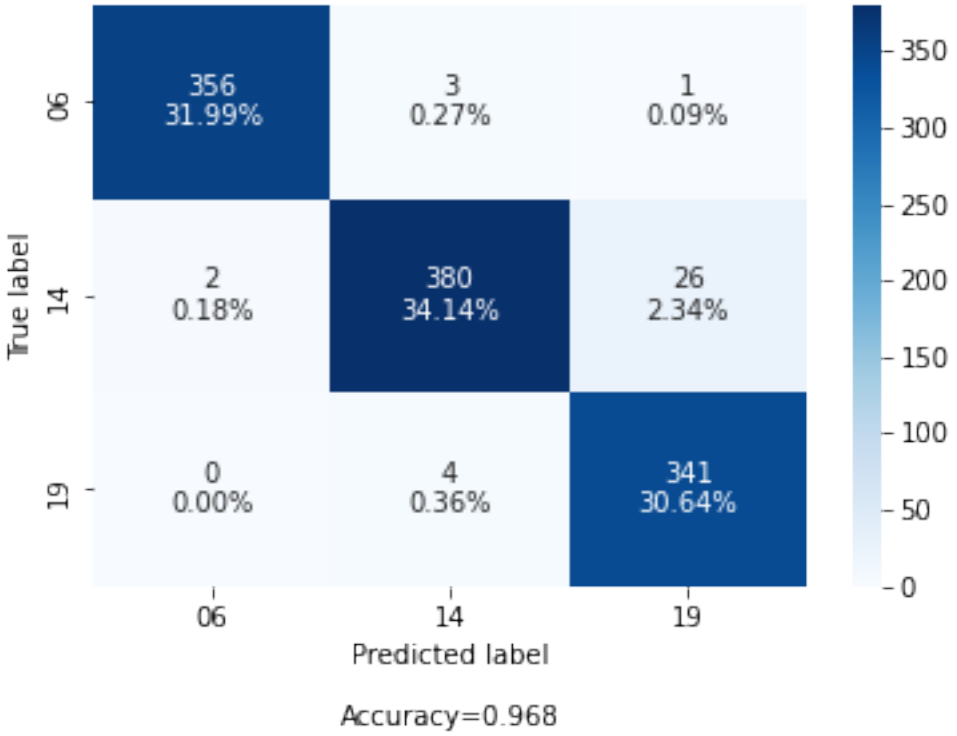
Confusion Matrix for all valid cross_validations against outliers



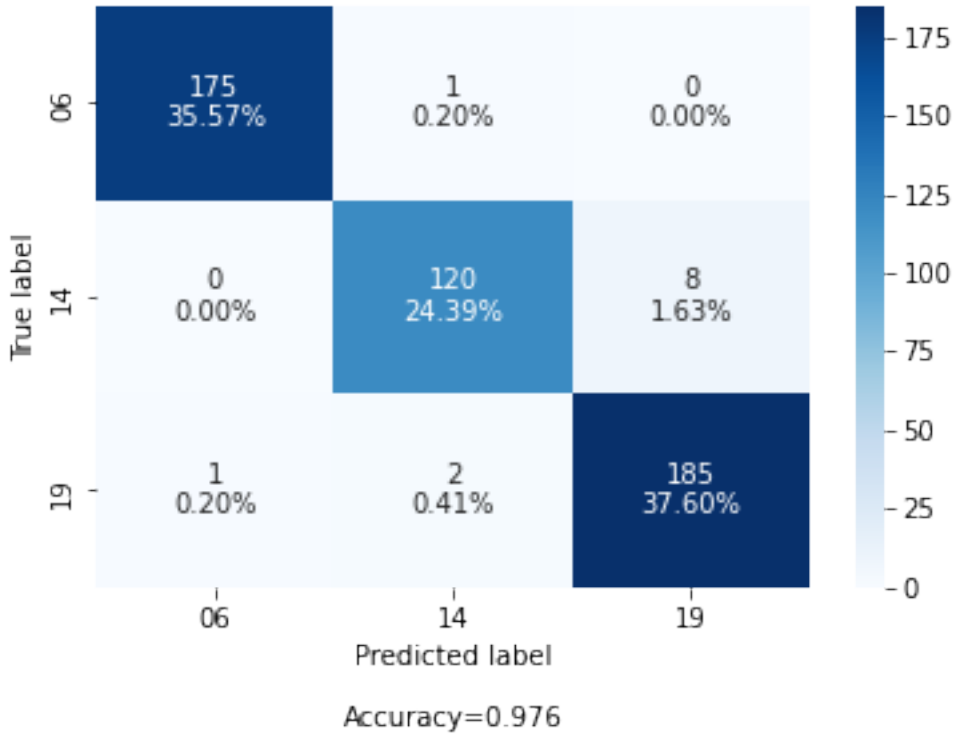

```
Processing started for split estimator: 4, epochs: [16]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
At split estimator: 4, epochs: [16]
Accurace mean(std): 0.9673377289296989(0.0087091306311048)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.99      0.99      0.99       360
          14       0.98      0.93      0.96       408
          19       0.93      0.99      0.96       345

    accuracy                           0.97      1113
   macro avg       0.97      0.97      0.97      1113
weighted avg       0.97      0.97      0.97      1113

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.99      0.99      0.99       176
          14       0.98      0.94      0.96       128
```

```
          19       0.96      0.98      0.97       188

    accuracy                           0.98       492
   macro avg       0.98      0.97      0.97       492
weighted avg       0.98      0.98      0.98       492
```

Confusion Matrix for all valid cross_validations against their tests sets



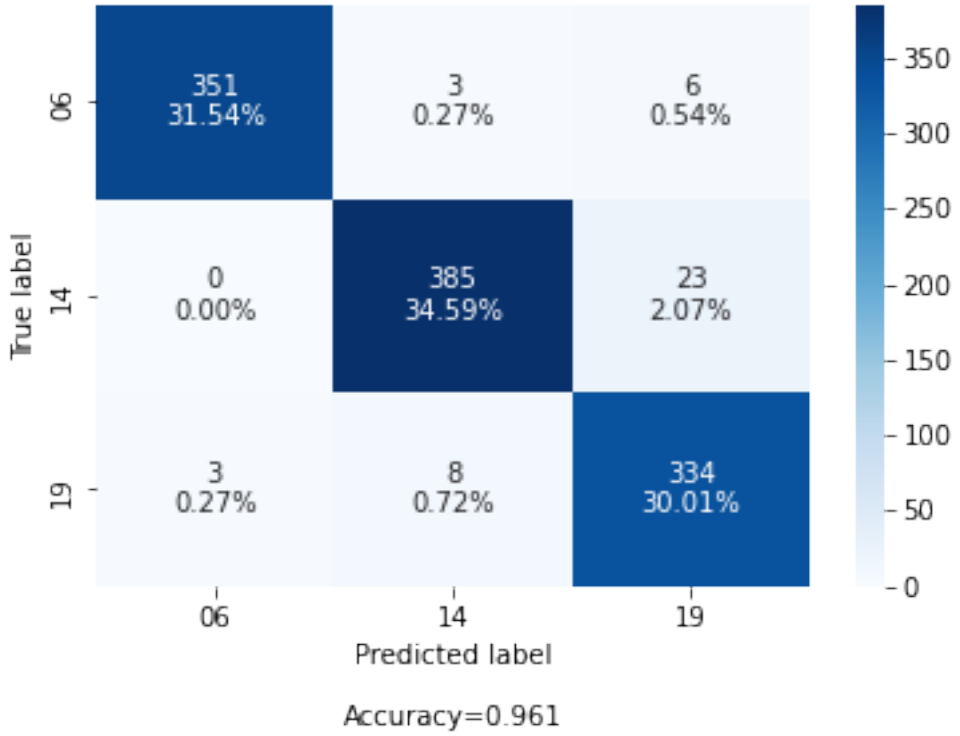
Confusion Matrix for all valid cross_validations against outliers



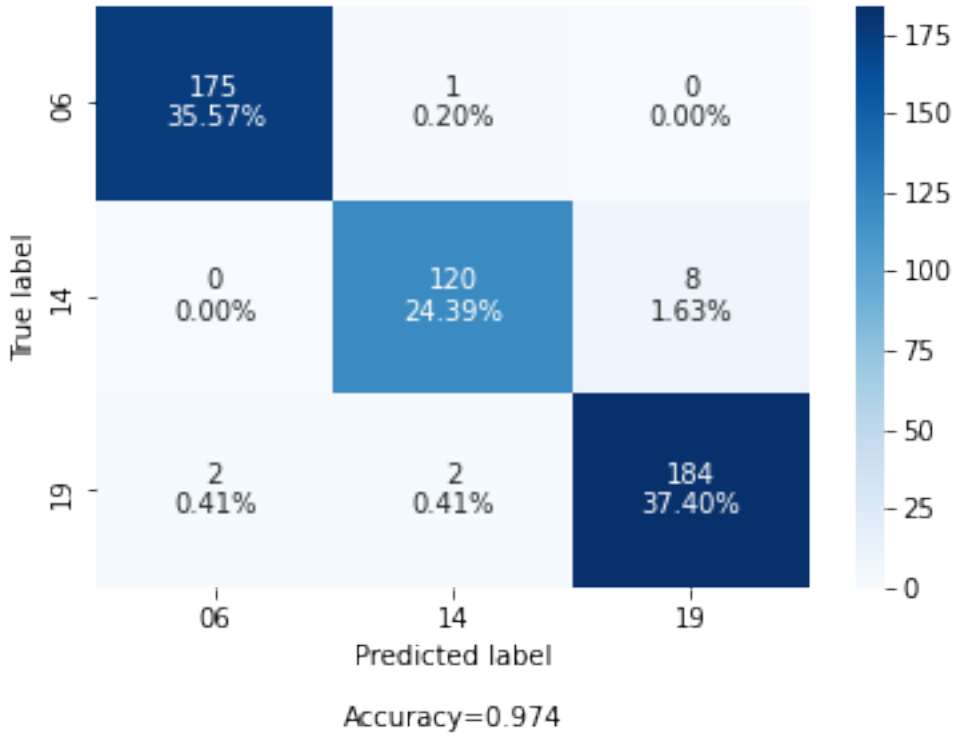

```
Processing started for split estimator: 4, epochs: [32]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
At split estimator: 4, epochs: [32]
Accurace mean(std): 0.9615259110234882(0.010334889982006008)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.99      0.97      0.98       360
          14       0.97      0.94      0.96       408
          19       0.92      0.97      0.94       345

    accuracy                           0.96      1113
   macro avg       0.96      0.96      0.96      1113
weighted avg       0.96      0.96      0.96      1113

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.99      0.99      0.99       176
          14       0.98      0.94      0.96       128
```

```
          19       0.96      0.98      0.97       188

    accuracy                           0.97       492
   macro avg       0.97      0.97      0.97       492
weighted avg       0.97      0.97      0.97       492
```

Confusion Matrix for all valid cross_validations against their tests sets



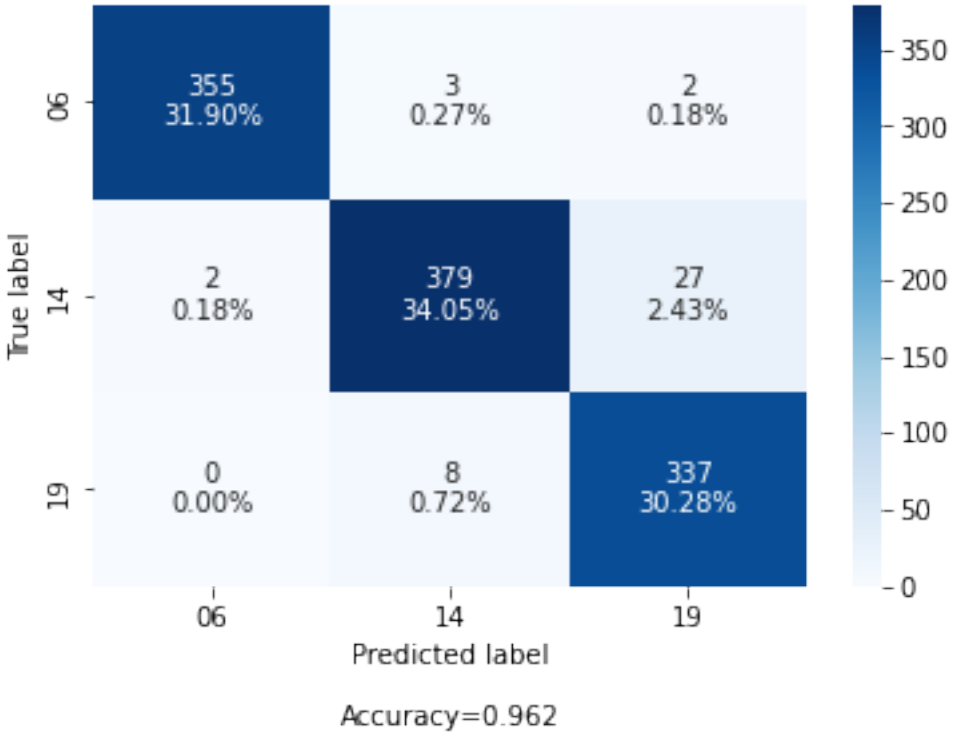
Confusion Matrix for all valid cross_validations against outliers



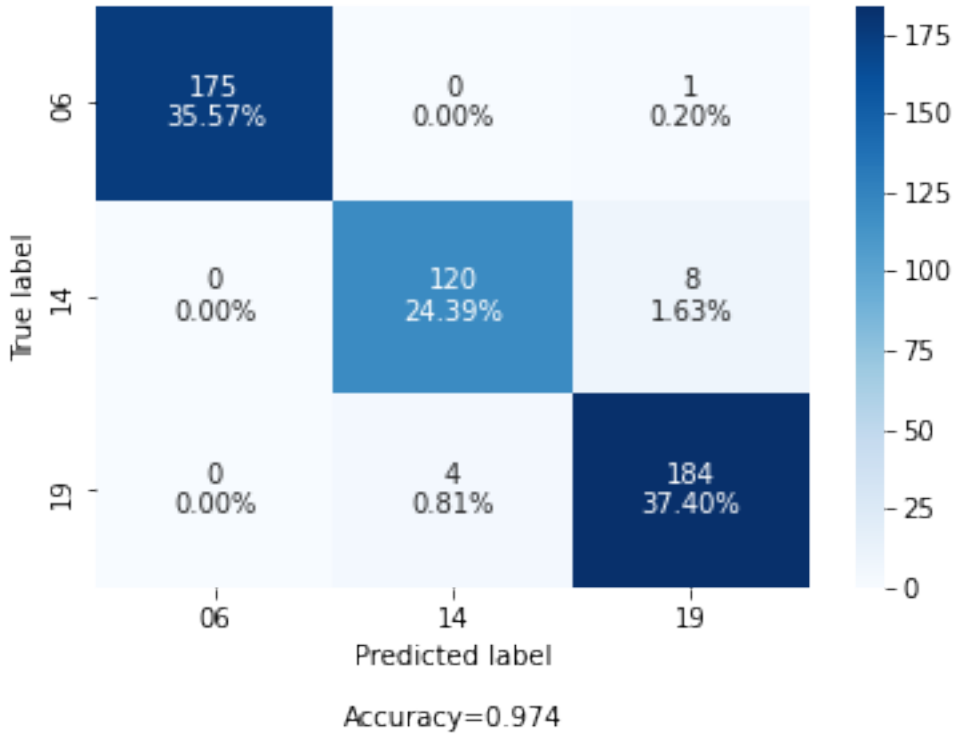

```
Processing started for split estimator: 4, epochs: [64]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
At split estimator: 4, epochs: [64]
Accurace mean(std): 0.962094298245614(0.023301991692305086)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.99      0.99      0.99       360
          14       0.97      0.93      0.95       408
          19       0.92      0.98      0.95       345

    accuracy                           0.96      1113
   macro avg       0.96      0.96      0.96      1113
weighted avg       0.96      0.96      0.96      1113

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      0.99      1.00       176
          14       0.97      0.94      0.95       128
```

```
          19       0.95      0.98      0.97       188

    accuracy                           0.97       492
   macro avg       0.97      0.97      0.97       492
weighted avg       0.97      0.97      0.97       492
```

Confusion Matrix for all valid cross_validations against their tests sets



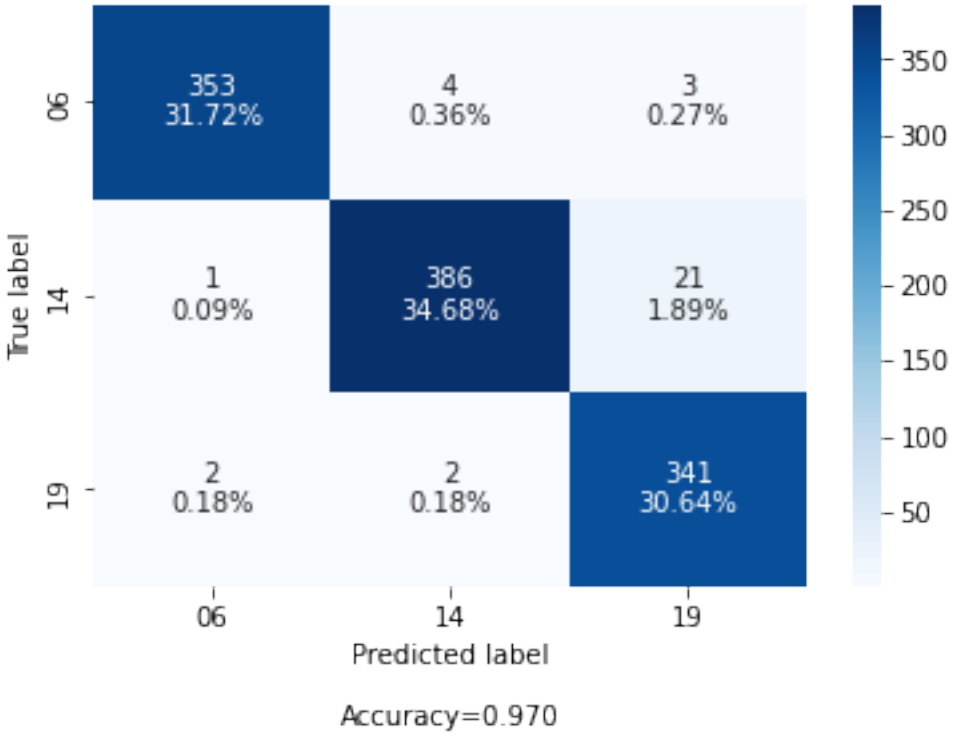
Confusion Matrix for all valid cross_validations against outliers



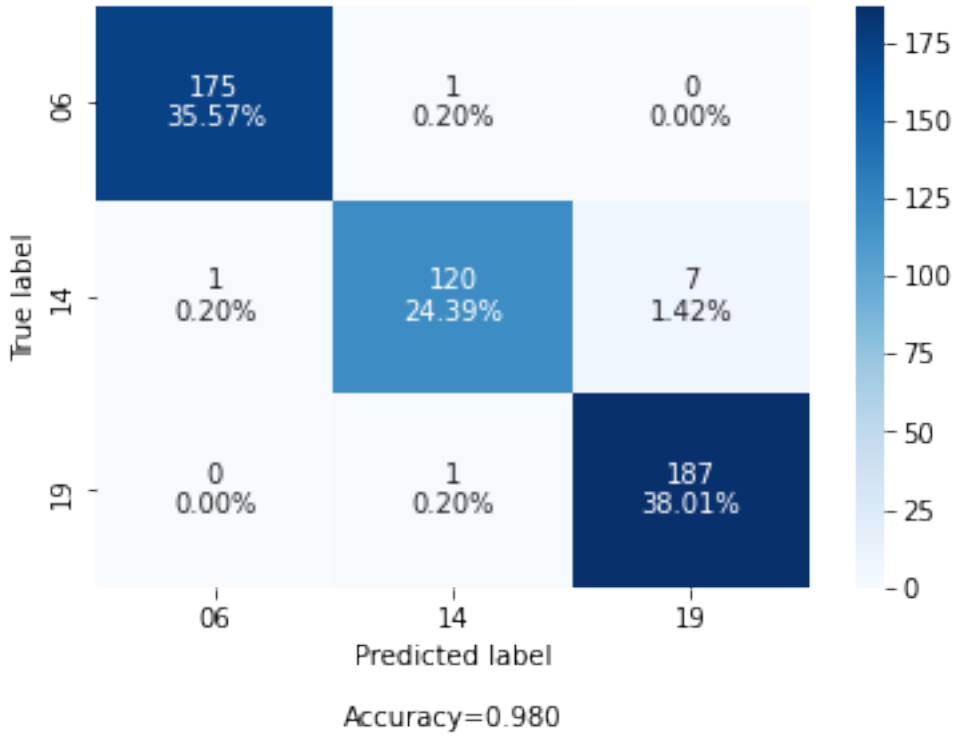

```
Processing started for split estimator: 4, epochs: [128]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
At split estimator: 4, epochs: [128]
Accurace mean(std): 0.9699854523255875(0.012794105184584402)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.99      0.98      0.99       360
          14       0.98      0.95      0.96       408
          19       0.93      0.99      0.96       345

    accuracy                           0.97      1113
   macro avg       0.97      0.97      0.97      1113
weighted avg       0.97      0.97      0.97      1113

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.99      0.99      0.99       176
          14       0.98      0.94      0.96       128
```

```
          19       0.96      0.99      0.98       188

    accuracy                           0.98       492
   macro avg       0.98      0.98      0.98       492
weighted avg       0.98      0.98      0.98       492
```

Confusion Matrix for all valid cross_validations against their tests sets



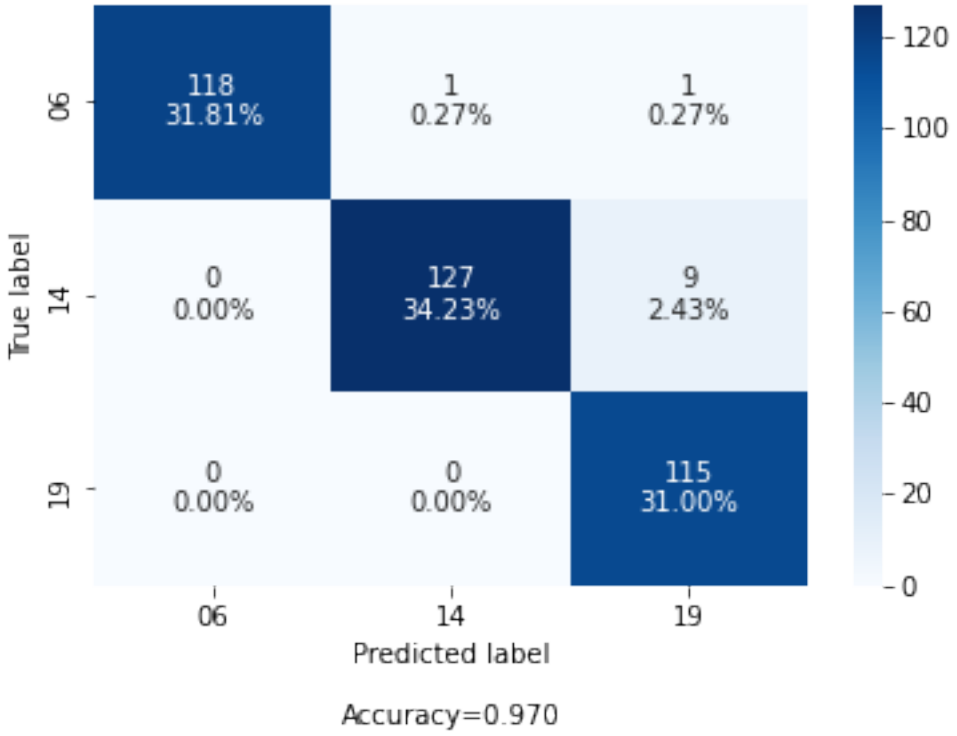
Confusion Matrix for all valid cross_validations against outliers



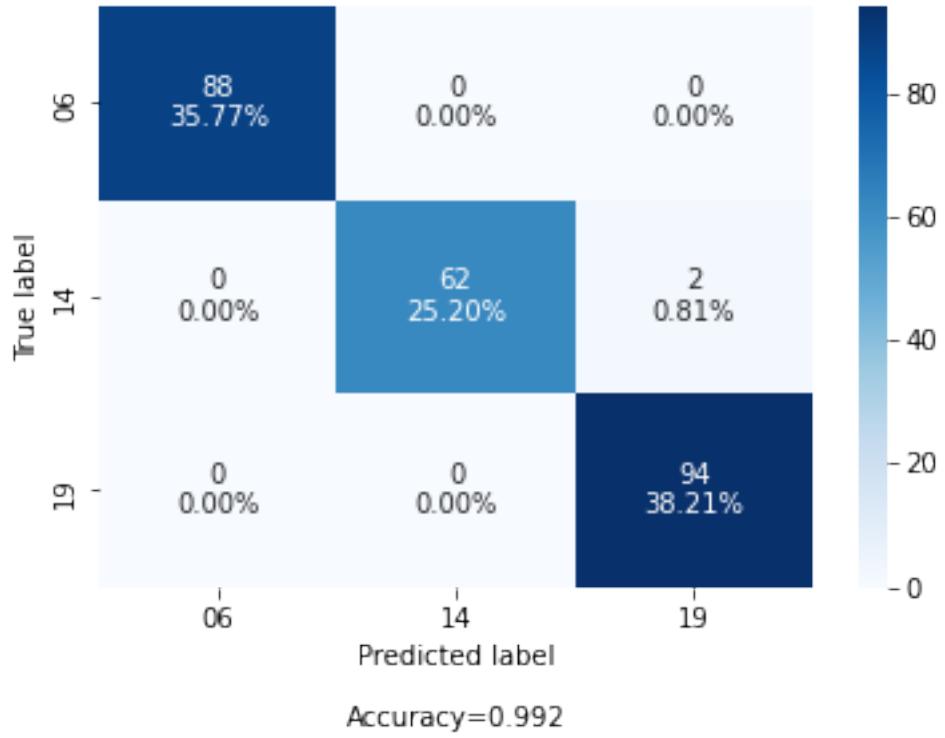

```
Processing started for split estimator: 2, epochs: [8]
Processing  1 -fold
Processing  2 -fold
At split estimator: 2, epochs: [8]
Accurace mean(std): 0.9696044211751018(0.025159976730657396)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       1.00      0.98      0.99       120
          14       0.99      0.93      0.96       136
          19       0.92      1.00      0.96       115

    accuracy                           0.97       371
   macro avg       0.97      0.97      0.97       371
weighted avg       0.97      0.97      0.97       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      1.00      1.00        88
          14       1.00      0.97      0.98        64
          19       0.98      1.00      0.99        94
```

```
    accuracy                           0.99       246
   macro avg       0.99      0.99      0.99       246
weighted avg       0.99      0.99      0.99       246
```

Confusion Matrix for all valid cross_validations against their tests sets



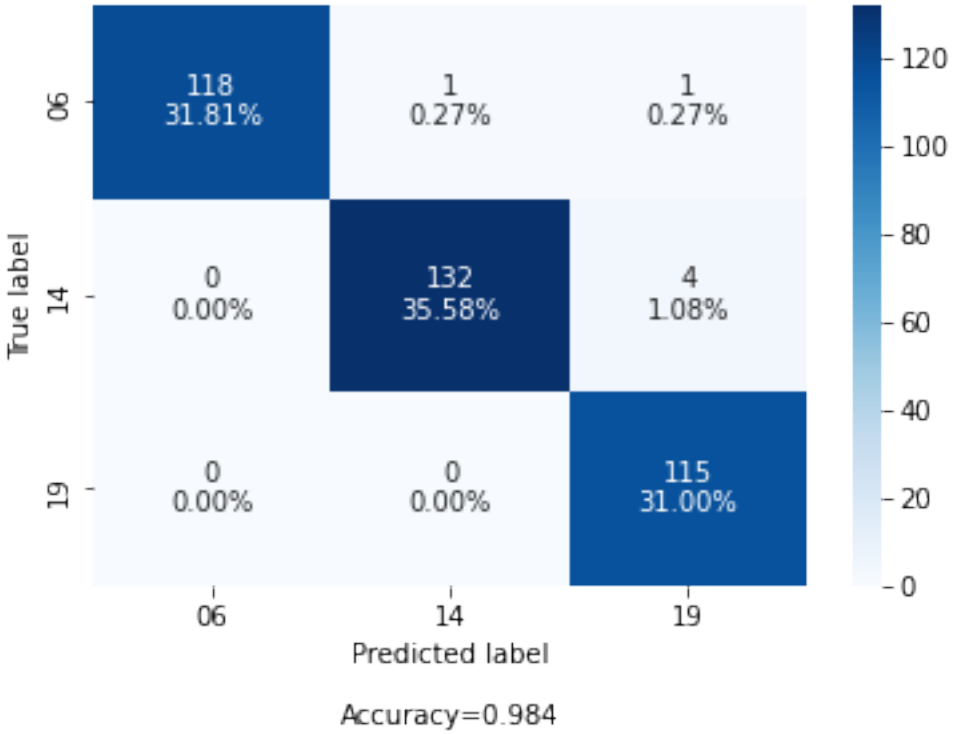
Confusion Matrix for all valid cross_validations against outliers



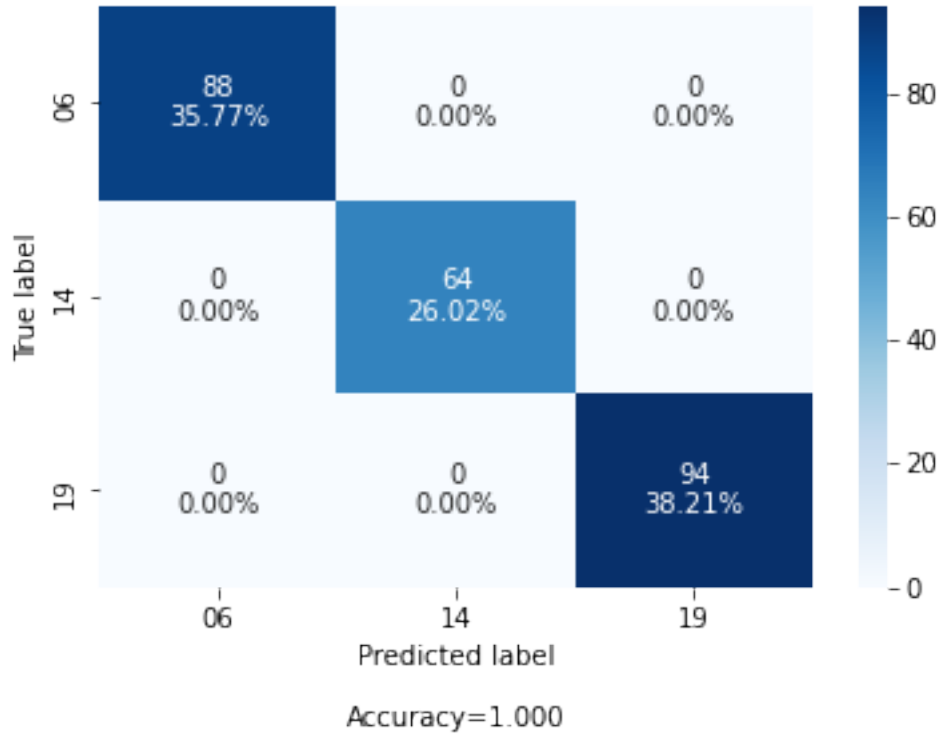

```
Processing started for split estimator: 2, epochs: [16]
Processing  1 -fold
Processing  2 -fold
At split estimator: 2, epochs: [16]
Accurace mean(std): 0.983469387755102(0.006326530612244918)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       1.00      0.98      0.99       120
          14       0.99      0.97      0.98       136
          19       0.96      1.00      0.98       115

    accuracy                           0.98       371
   macro avg       0.98      0.98      0.98       371
weighted avg       0.98      0.98      0.98       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      1.00      1.00        88
          14       1.00      1.00      1.00        64
          19       1.00      1.00      1.00        94
```

```
    accuracy                           1.00       246
   macro avg       1.00      1.00      1.00       246
weighted avg       1.00      1.00      1.00       246
```

Confusion Matrix for all valid cross_validations against their tests sets



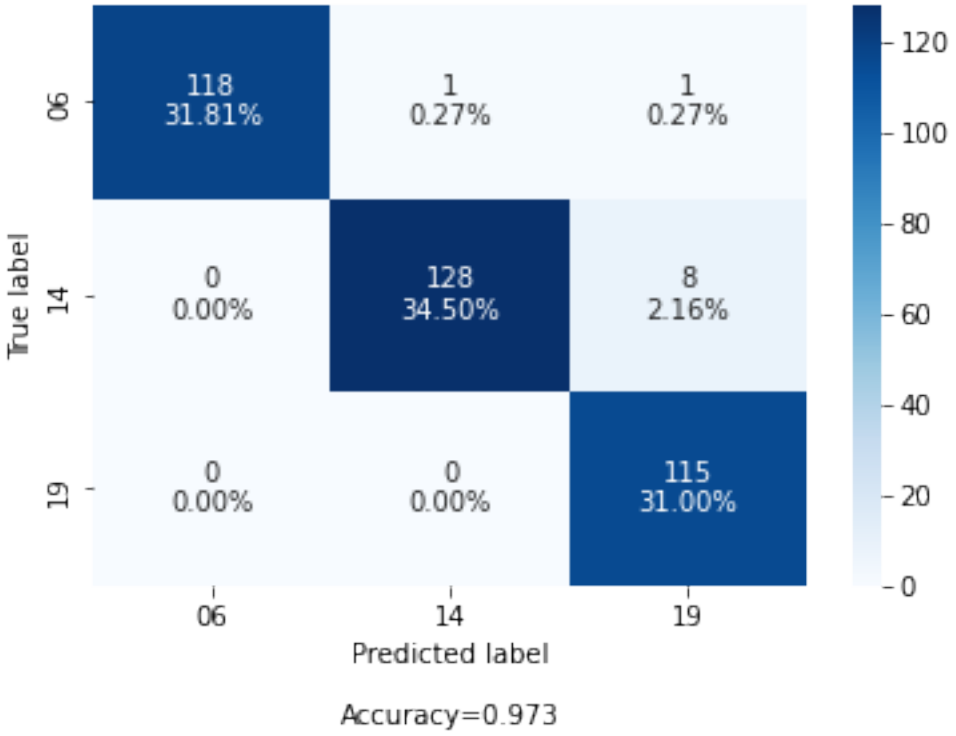
Confusion Matrix for all valid cross_validations against outliers



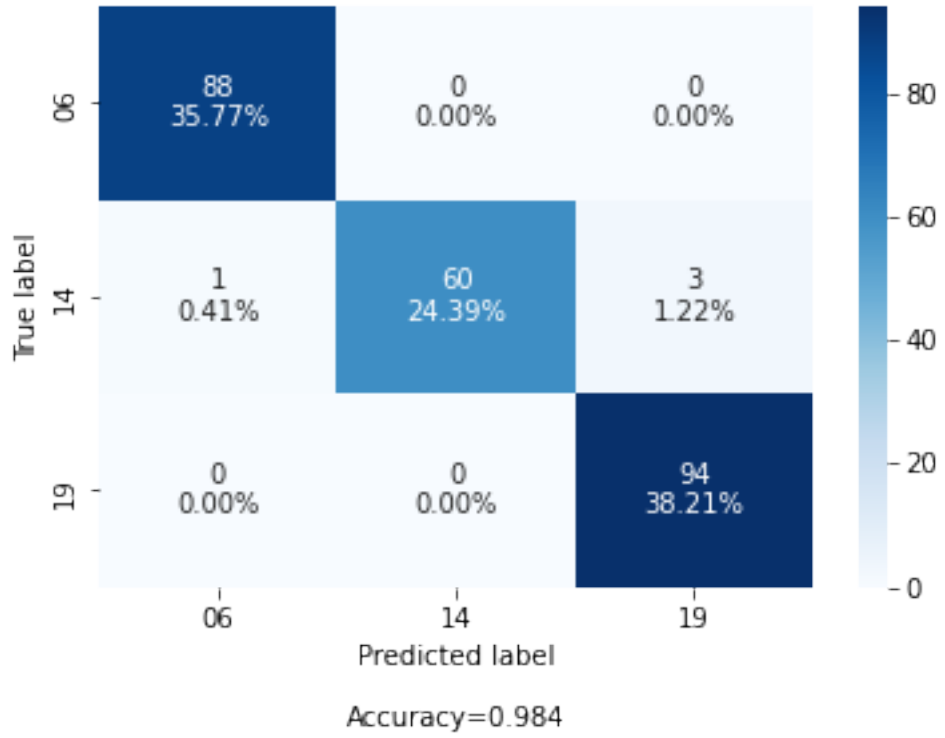

```
Processing started for split estimator: 2, epochs: [32]
Processing  1 -fold
Processing  2 -fold
At split estimator: 2, epochs: [32]
Accurace mean(std): 0.972972972972973(0.02702702702702703)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       1.00      0.98      0.99       120
          14       0.99      0.94      0.97       136
          19       0.93      1.00      0.96       115

    accuracy                           0.97       371
   macro avg       0.97      0.97      0.97       371
weighted avg       0.97      0.97      0.97       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.99      1.00      0.99        88
          14       1.00      0.94      0.97        64
          19       0.97      1.00      0.98        94
```

```
    accuracy                           0.98       246
   macro avg       0.99      0.98      0.98       246
weighted avg       0.98      0.98      0.98       246
```

Confusion Matrix for all valid cross_validations against their tests sets



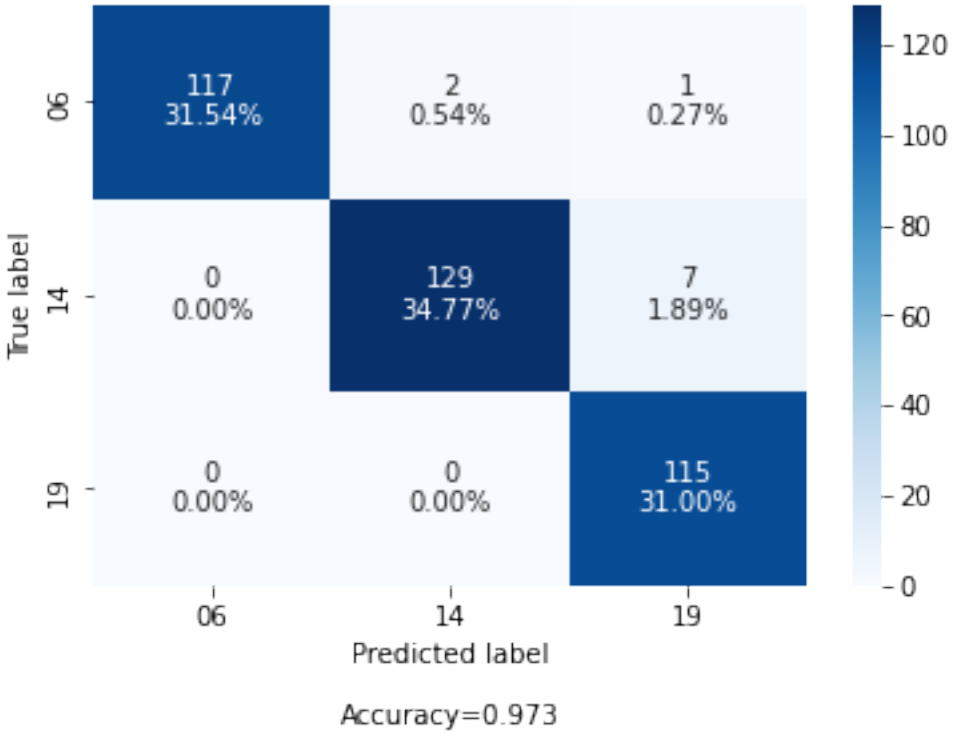
Confusion Matrix for all valid cross_validations against outliers



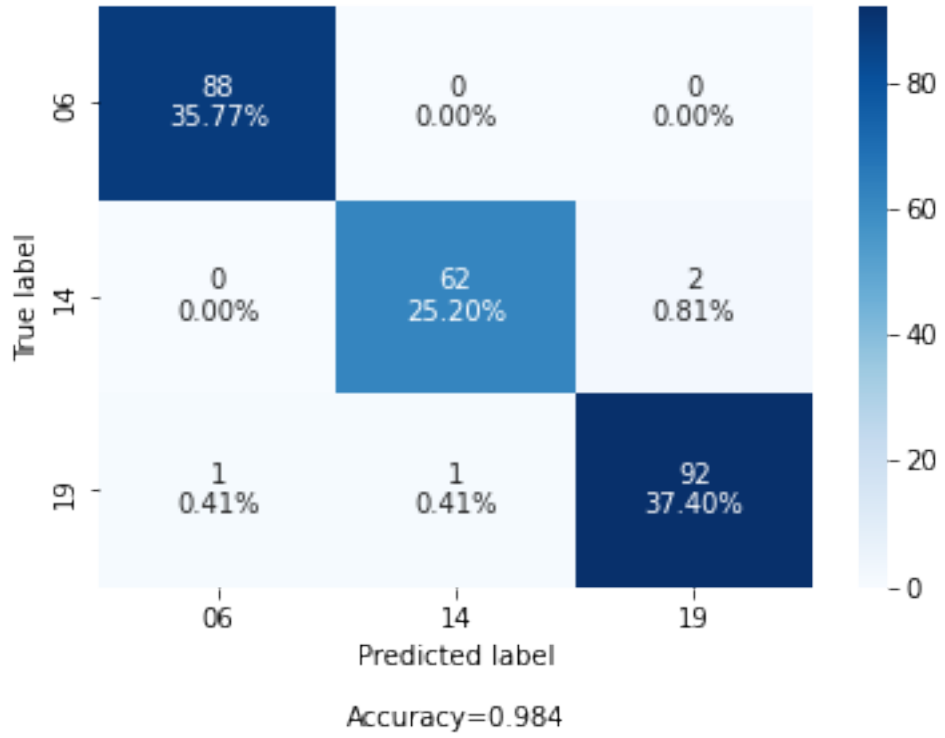

```
Processing started for split estimator: 2, epochs: [64]
Processing  1 -fold
Processing  2 -fold
At split estimator: 2, epochs: [64]
Accurace mean(std): 0.9723467427373813(0.017290562962100464)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       1.00      0.97      0.99       120
          14       0.98      0.95      0.97       136
          19       0.93      1.00      0.97       115

    accuracy                           0.97       371
   macro avg       0.97      0.97      0.97       371
weighted avg       0.97      0.97      0.97       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       0.99      1.00      0.99        88
          14       0.98      0.97      0.98        64
          19       0.98      0.98      0.98        94
```

```
              precision    recall  f1-score   support

    accuracy                         0.98       246
   macro avg       0.98      0.98      0.98       246
weighted avg       0.98      0.98      0.98       246
```

Confusion Matrix for all valid cross_validations against their tests sets



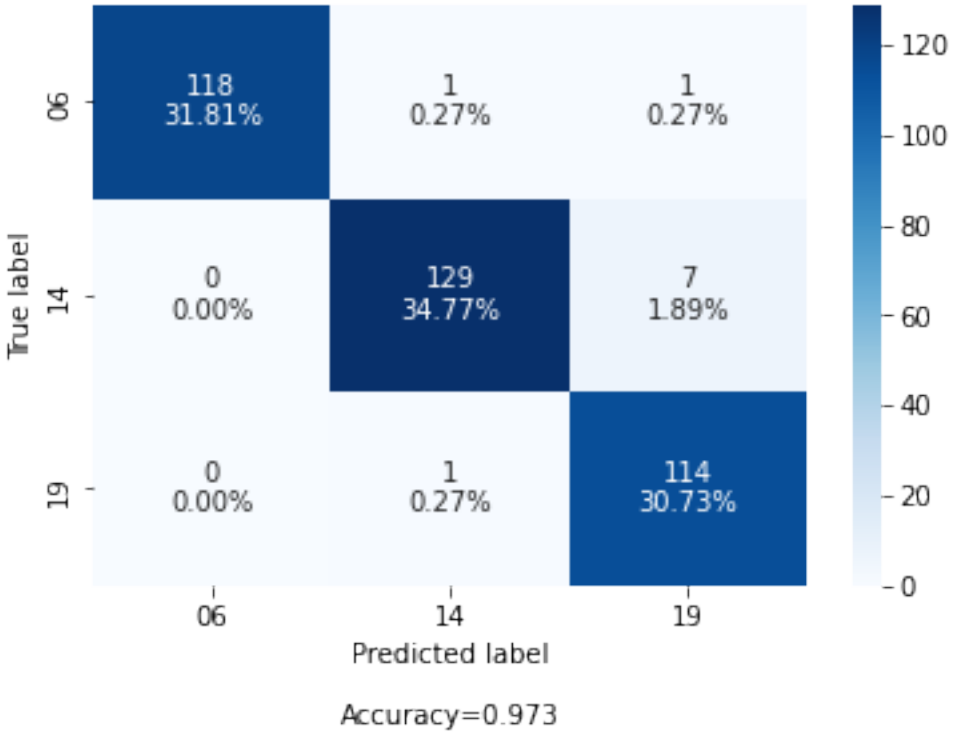
Confusion Matrix for all valid cross_validations against outliers



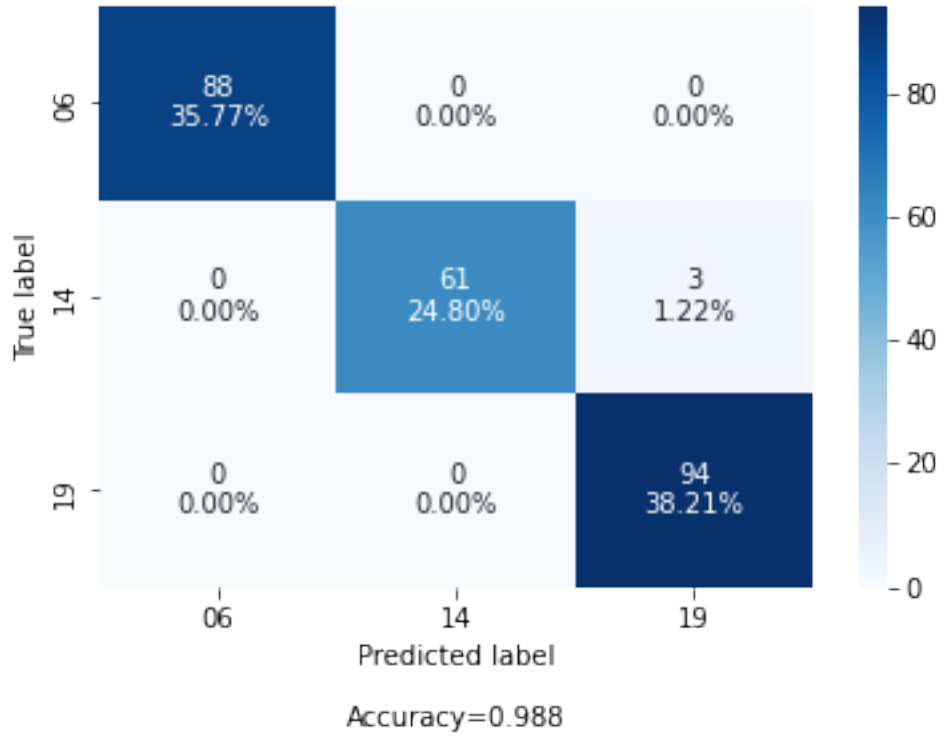

```
Processing started for split estimator: 2, epochs: [128]
Processing  1 -fold
Processing  2 -fold
At split estimator: 2, epochs: [128]
Accurace mean(std): 0.972987503632665(0.021636152281313548)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       1.00      0.98      0.99       120
          14       0.98      0.95      0.97       136
          19       0.93      0.99      0.96       115

    accuracy                           0.97       371
   macro avg       0.97      0.97      0.97       371
weighted avg       0.97      0.97      0.97       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      1.00      1.00        88
          14       1.00      0.95      0.98        64
          19       0.97      1.00      0.98        94
```

```
    accuracy                           0.99       246
   macro avg       0.99      0.98      0.99       246
weighted avg       0.99      0.99      0.99       246
```

Confusion Matrix for all valid cross_validations against their tests sets



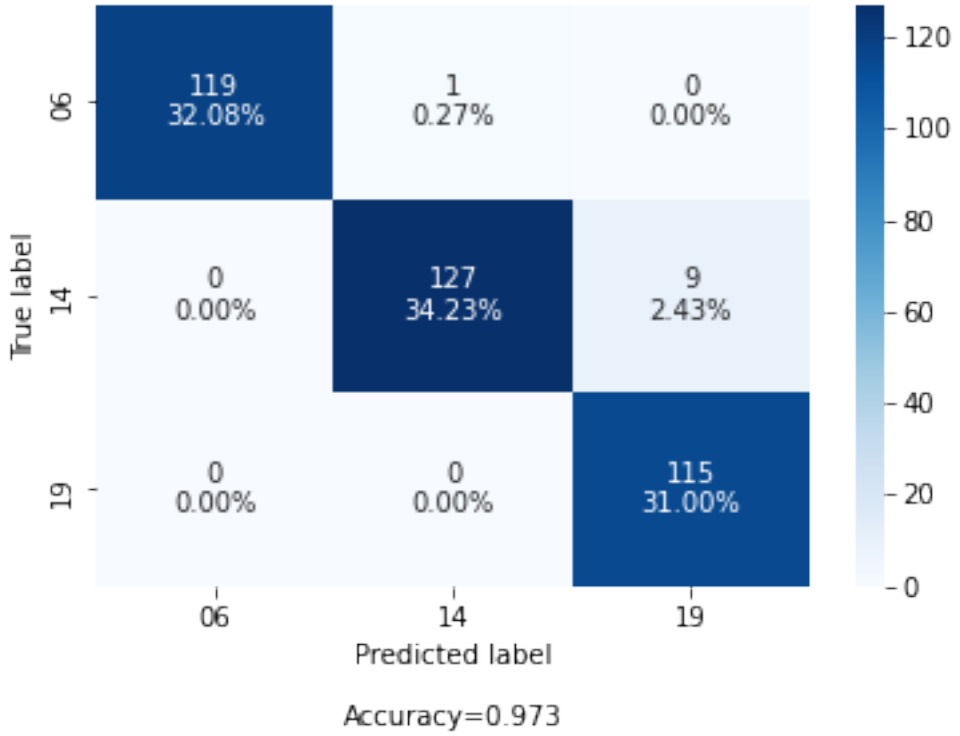
Confusion Matrix for all valid cross_validations against outliers



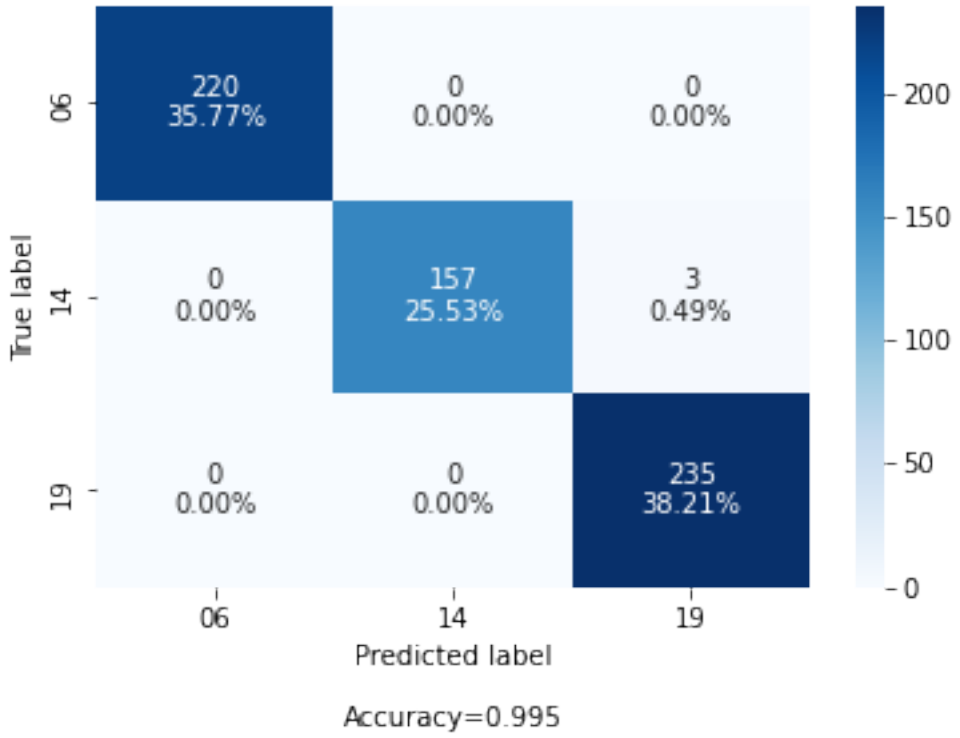

```
Processing started for real split: 5, epochs: [8]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [8]
Accurace mean(std): 0.9518461538461539(0.09138979524299314)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       1.00      0.99      1.00       120
          14       0.99      0.93      0.96       136
          19       0.93      1.00      0.96       115

    accuracy                           0.97       371
   macro avg       0.97      0.98      0.97       371
weighted avg       0.97      0.97      0.97       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      1.00      1.00       220
```

```
          14       1.00      0.98      0.99       160
          19       0.99      1.00      0.99       235

    accuracy                           1.00       615
   macro avg       1.00      0.99      0.99       615
weighted avg       1.00      1.00      1.00       615
```

Confusion Matrix for all valid cross_validations against their tests sets



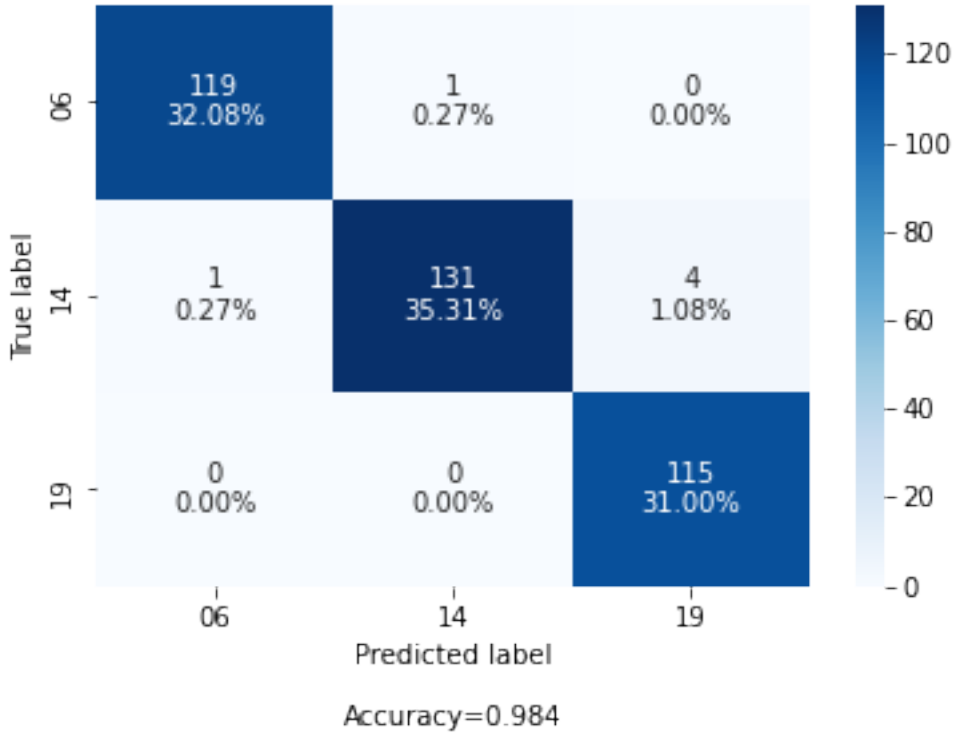
Confusion Matrix for all valid cross_validations against outliers



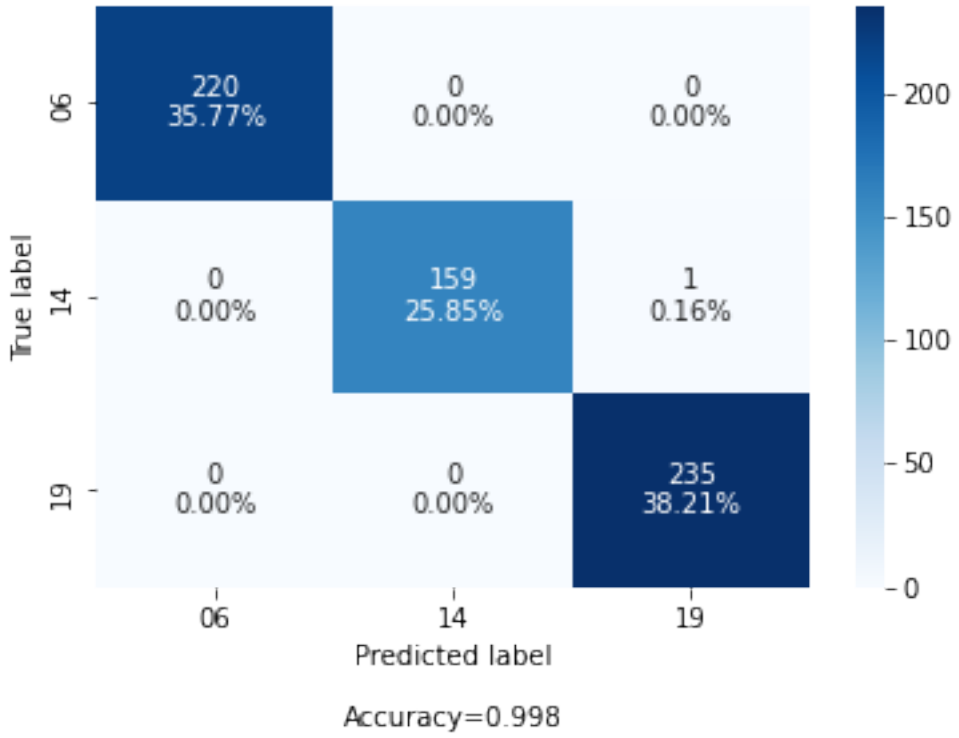

```
Processing started for real split: 5, epochs: [16]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [16]
Accurace mean(std): 0.972275641025641(0.050402105987433124)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.99      0.99      0.99       120
          14       0.99      0.96      0.98       136
          19       0.97      1.00      0.98       115

    accuracy                           0.98       371
   macro avg       0.98      0.98      0.98       371
weighted avg       0.98      0.98      0.98       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      1.00      1.00       220
```

```
          14       1.00      0.99      1.00       160
          19       1.00      1.00      1.00       235

    accuracy                           1.00       615
   macro avg       1.00      1.00      1.00       615
weighted avg       1.00      1.00      1.00       615
```

Confusion Matrix for all valid cross_validations against their tests sets



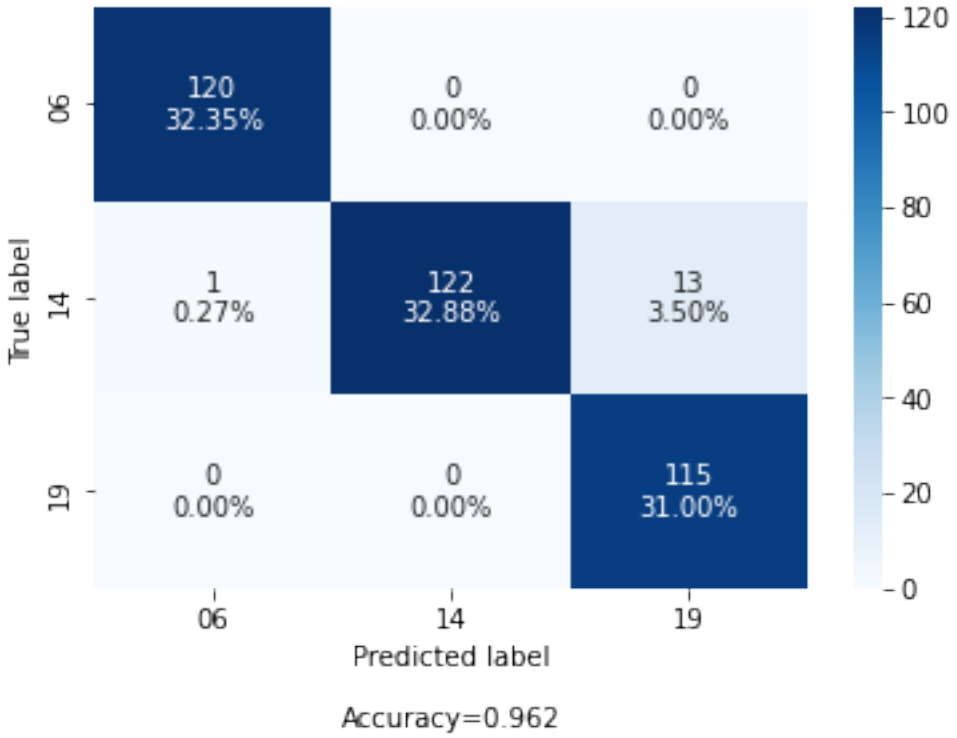
Confusion Matrix for all valid cross_validations against outliers



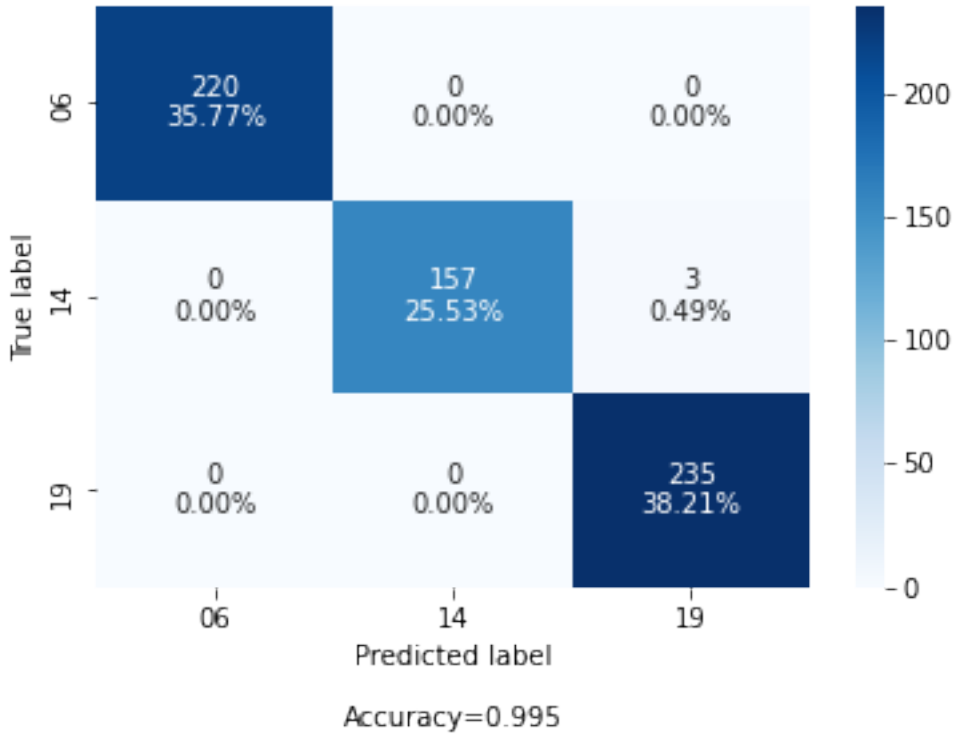

```
Processing started for real split: 5, epochs: [32]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [32]
Accurace mean(std): 0.9640373637778931(0.06183847735145037)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       0.99      1.00      1.00       120
          14       1.00      0.90      0.95       136
          19       0.90      1.00      0.95       115

    accuracy                           0.96       371
   macro avg       0.96      0.97      0.96       371
weighted avg       0.97      0.96      0.96       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      1.00      1.00       220
```

```
              14       1.00      0.98      0.99       160
              19       0.99      1.00      0.99       235

    accuracy                           1.00       615
   macro avg       1.00      0.99      0.99       615
weighted avg       1.00      1.00      1.00       615
```

Confusion Matrix for all valid cross_validations against their tests sets



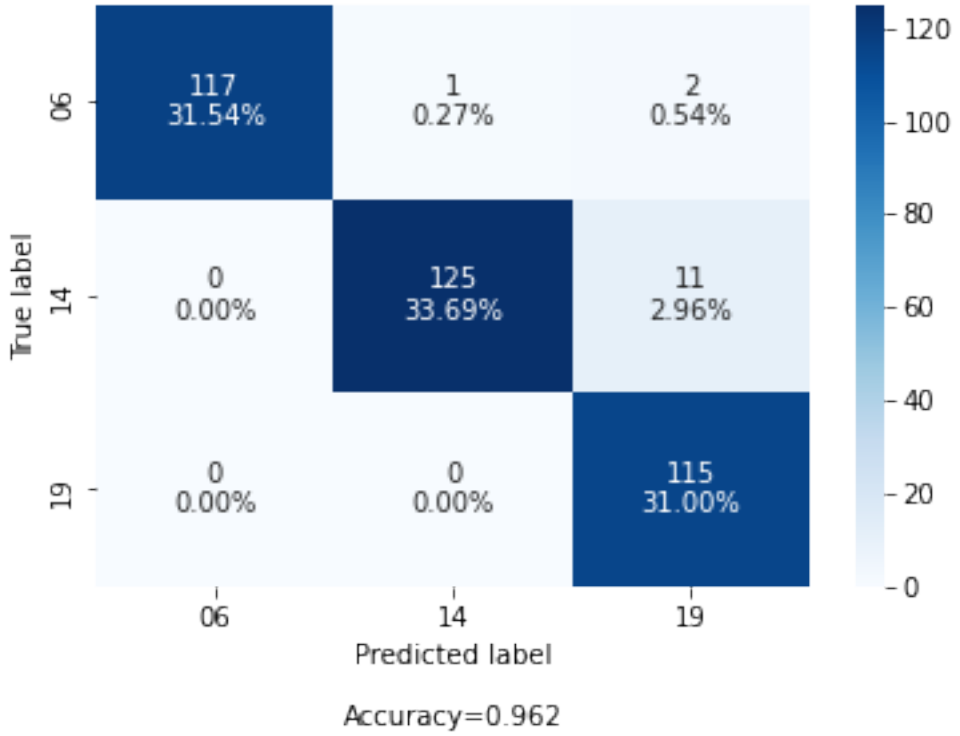
Confusion Matrix for all valid cross_validations against outliers



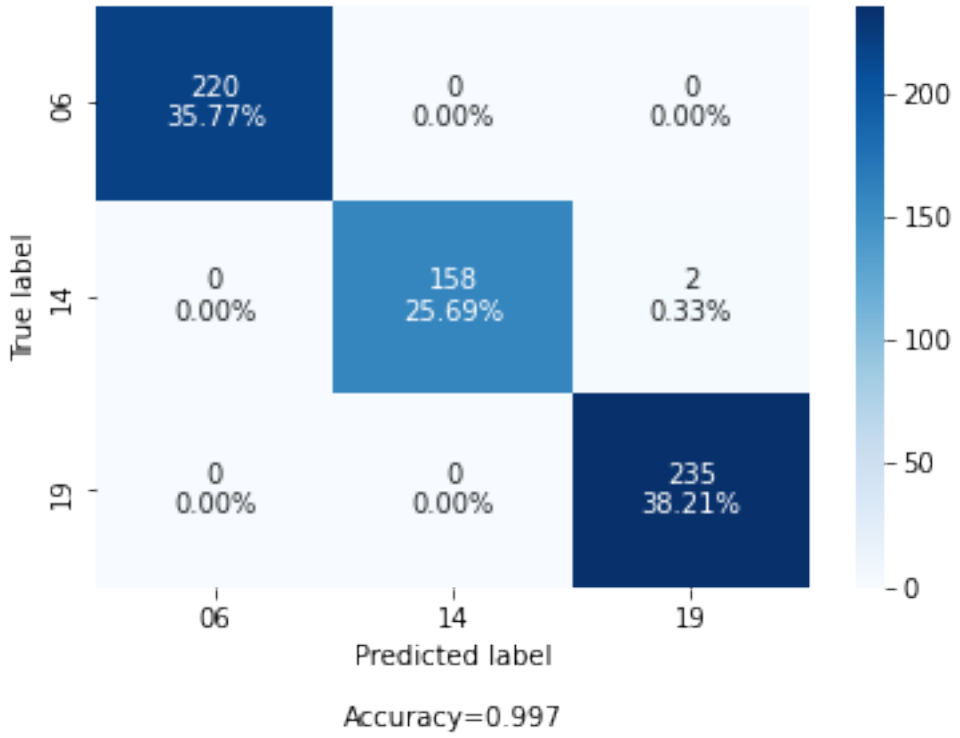

```
Processing started for real split: 5, epochs: [64]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [64]
Accurace mean(std): 0.9666913032766692(0.05705541016861657)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       1.00      0.97      0.99       120
          14       0.99      0.92      0.95       136
          19       0.90      1.00      0.95       115

    accuracy                           0.96       371
   macro avg       0.96      0.96      0.96       371
weighted avg       0.97      0.96      0.96       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      1.00      1.00       220
```

```
              14       1.00      0.99      0.99       160
              19       0.99      1.00      1.00       235

    accuracy                             1.00       615
   macro avg       1.00      1.00      1.00       615
weighted avg       1.00      1.00      1.00       615
```

Confusion Matrix for all valid cross_validations against their tests sets



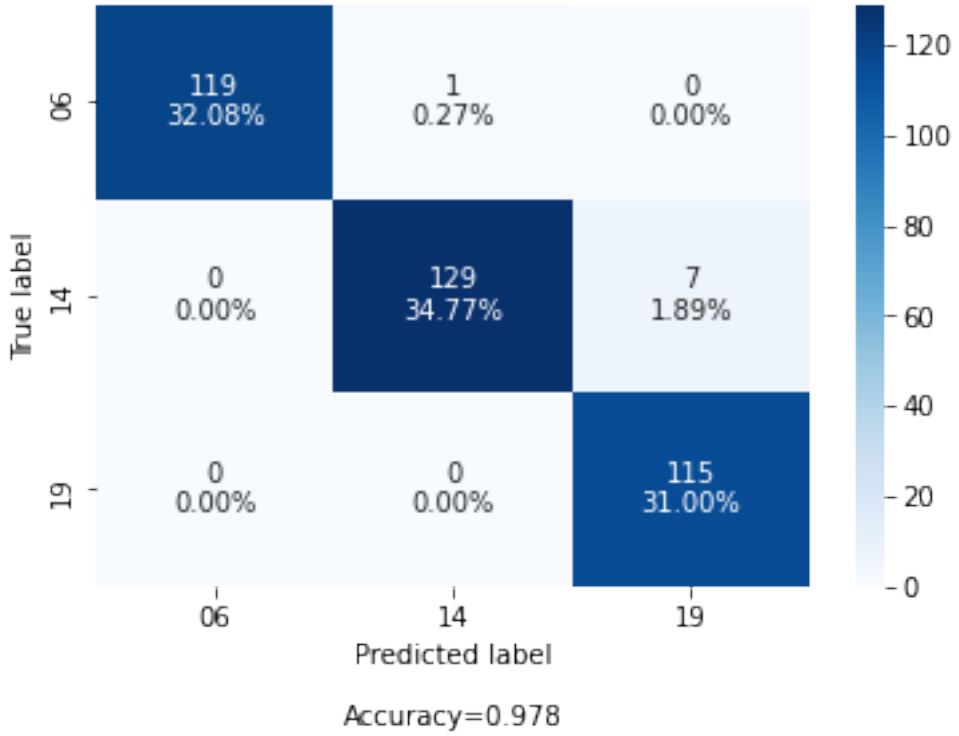
Confusion Matrix for all valid cross_validations against outliers



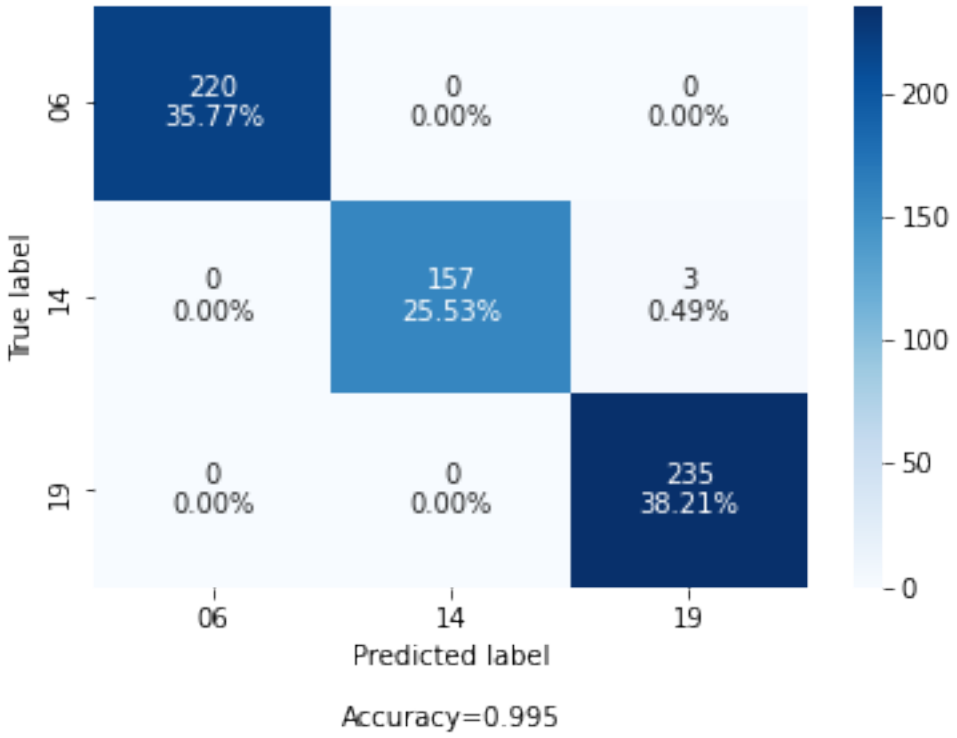

```
Processing started for real split: 5, epochs: [128]
Processing  1 -fold
Processing  2 -fold
Processing  3 -fold
Processing  4 -fold
Processing  5 -fold
At split estimator: 5, epochs: [128]
Accurace mean(std): 0.9620192307692308(0.07086813160780056)
Classification report for all valid cross_validations against their tests sets
              precision    recall  f1-score   support

          06       1.00      0.99      1.00       120
          14       0.99      0.95      0.97       136
          19       0.94      1.00      0.97       115

    accuracy                           0.98       371
   macro avg       0.98      0.98      0.98       371
weighted avg       0.98      0.98      0.98       371

Classification report for all valid cross_validations against outliers
              precision    recall  f1-score   support

          06       1.00      1.00      1.00       220
```

```
              14       1.00      0.98      0.99       160
              19       0.99      1.00      0.99       235

    accuracy                           1.00       615
   macro avg       1.00      0.99      0.99       615
weighted avg       1.00      1.00      1.00       615
```

Confusion Matrix for all valid cross_validations against their tests sets



Confusion Matrix for all valid cross_validations against outliers


220
35.77%
0
0.00%
0
0.00%
0
0.00%
157
25.53%
3
0.49%
0
0.00%
0
0.00%
235
38.21%
06
14
19
True label
06
14
19
Predicted label
Accuracy=0.995
200
150
100
50
0

[ ]: