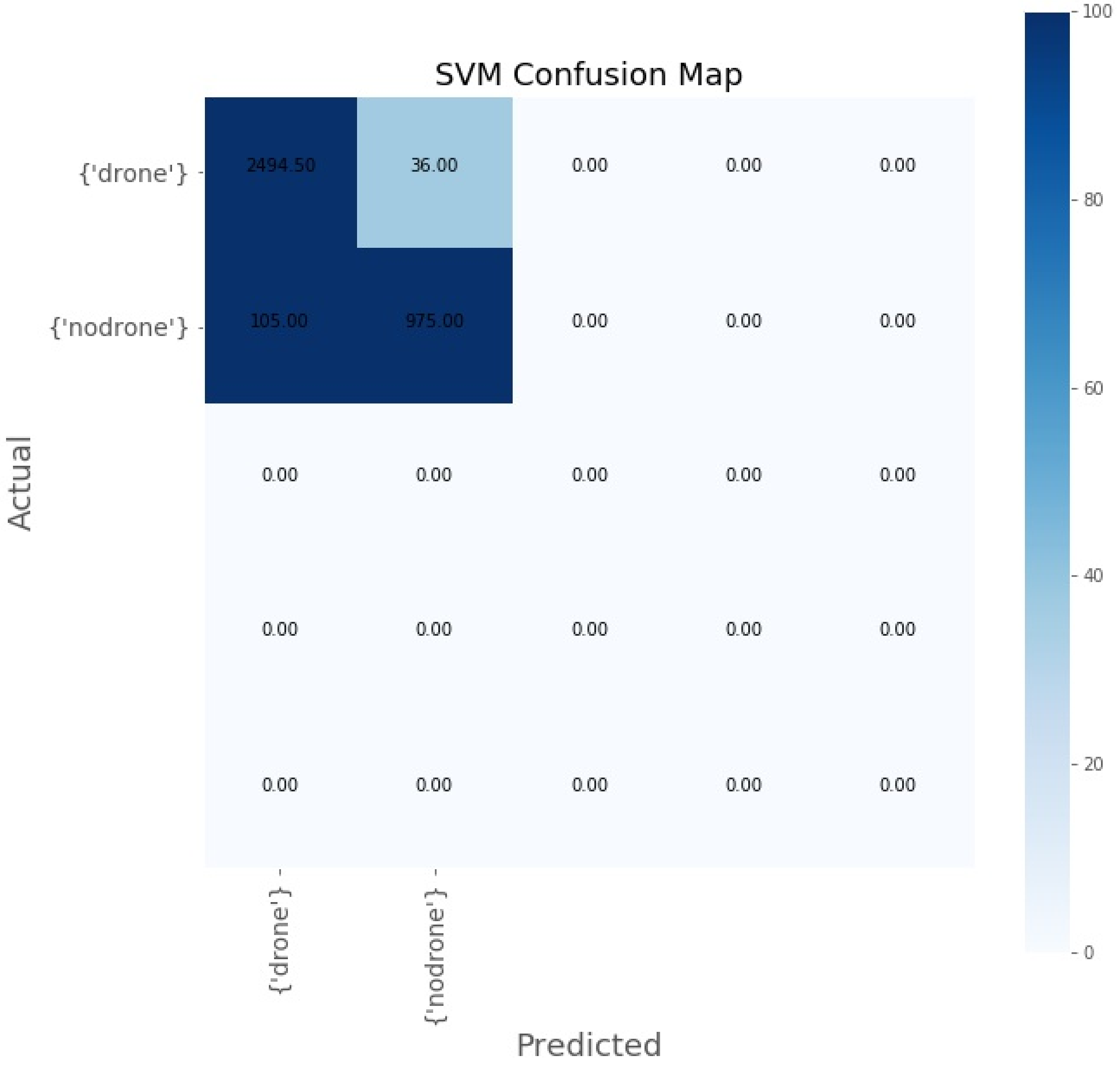
```
   ...: mfcc_data = []
   ...: lpc_data1 = []
   ...: with open(metafile, 'r',newline='') as f:
   ...:     reader = csv.DictReader(f, delimiter=',')
   ...:     i=0
   ...:     mfccl = []
   ...:     chromal = []
   ...:     mell = []
   ...:     spectl = []
   ...:     tonnetzl = []
   ...:     lpcl = []
   ...:     rlpcl= []
   ...:     psdl =[]
   ...:     for row in reader:
   ...:         fileid = float(row['ID'])
   ...:         lpc = [float(t) for t in row['LPCCOEFF'].strip("[]").split()]
   ...:         rlpc = [float(t) for t in row['RCOEFF'].strip("[]").split()]
   ...:         pfreq= [float(t) for t in row['Pfreq'].strip("[]").split()]
   ...:         h = row['Height']
   ...:         if h == 'n':
   ...:             label1 = "nodrone"
   ...:         else:
   ...:             label1 = "drone"
   ...:
   ...:         #features = np.empty((0,193))
   ...:         fpfeatures = np.empty((0,6))
   ...:         #ext_features = np.hstack([mfcc,chroma,mel,spect,tonnetz])
   ...:         fpext_features = np.hstack([pfreq])
   ...:         #features = np.vstack([features,ext_features])
   ...:         fpfeatures = np.vstack([fpfeatures,fpext_features])
   ...:         i+=1
   ...:         if not np.isnan(lpc[0]):
   ...:             #mfccl.append(mfcc)
   ...:             #chromal.append(chroma)
   ...:             #mell.append(mel)
   ...:             #spectl.append(spect)
   ...:             #tonnetzl.append(tonnetz)
   ...:             lpcl.append(lpc)
   ...:             psdl.append(psd)
   ...:             rlpcl.append(rlpc)
   ...:             #mfcc_data.append([features, features.shape, label])
   ...:             lpc_data1.append([fpfeatures, fpfeatures.shape, label1])
   ...: cols=["features", "shape","label"]
   ...: #mfcc_pd1 = pd.DataFrame(data = mfcc_data, columns=cols)
   ...: lpc_pd1 = pd.DataFrame(data = lpc_data1, columns=cols)

In [36]: lpc_pd = lpc_pd1

In [37]: le = LabelEncoder()
   ...: #label_num = le.fit_transform(mfcc_pd["label"])
   ...: label_num1 = le.fit_transform(lpc_pd["label"])
   ...: ohe = OneHotEncoder()
   ...: #onehot = ohe.fit_transform(label_num.reshape(-1, 1))
   ...: onehot1 = ohe.fit_transform(label_num1.reshape(-1, 1))
   ...: def labelling(pddata, ln):
   ...:     pddata.insert(loc=3, column='label_id', value=ln)
   ...:     labels = set(pddata['label'])
   ...:     print(labels)
   ...:     cnt = [[label,list(pddata['label']).count(label)] for label in
labels]
    ...:     dict_cnt = dict(cnt)
    ...:     print(dict_cnt)
    ...:     cnt_cols=["classes","occurence"]
    ...:     count_pd = pd.DataFrame(data = cnt, columns=cnt_cols)
    ...:     return dict_cnt
    ...: dictlpc = labelling(lpc_pd, label_num1)
    ...: it = int(len(data)*0.8)
    ...: iv = int(len(data)*0.2)
    ...: labels1 = set(lpc_pd['label_id'])
    ...: mapping = []
    ...: for label_id in labels1:
    ...:     label_name1 = set(lpc_pd.loc[lpc_pd['label_id'] == label_id]
['label'])
    ...:     mapping.append((label_id,label_name1))
    ...:
    ...: label_mapping1 = dict(mapping)
    ...: label_mapping1
    ...:
    ...: ll1 = [lpc_pd['features'][i].ravel() for i in range(lpc_pd.shape[0])]
    ...: lpc_pd['sample'] = pd.Series(ll1, index=lpc_pd.index)
    ...: del lpc_pd['features']
    ...:
    ...: lpc_train_data = np.array(list(lpc_pd[:-iv]['sample']))
    ...: lpc_train_label = np.array(list(lpc_pd[:-iv]['label_id']))
    ...: lpc_validation_data = np.array(list(lpc_pd[-iv:]['sample']))
    ...: lpc_validation_label = np.array(list(lpc_pd[-iv:]['label_id']))
    ...:
    ...: print(lpc_train_data.shape)
    ...: print(type(lpc_train_data))
    ...: def confusion(true, predicted):
    ...:     matrix = np.zeros([5,5])
    ...:     #d = 0
    ...:     for t, p in zip(true, predicted):
    ...:         matrix[t,p] += 1.5
    ...:     #    d += 1
    ...:     #print(d)
    ...:     return matrix
{'nodrone', 'drone'}
{'nodrone': 3461, 'drone': 8575}
(9629, 6)
<class 'numpy.ndarray'>

In [38]: lpc_pd['sample'] = lpc_pd['sample'].apply(np.sort,axis = 0)

In [39]: lpc_train_data = np.array(list(lpc_pd[:-iv]['sample']))
    ...: lpc_train_label = np.array(list(lpc_pd[:-iv]['label_id']))
    ...: lpc_validation_data = np.array(list(lpc_pd[-iv:]['sample']))
    ...: lpc_validation_label = np.array(list(lpc_pd[-iv:]['label_id']))
    ...:
    ...: print(lpc_train_data.shape)
    ...: print(type(lpc_train_data))
(9629, 6)
<class 'numpy.ndarray'>

In [40]: svmmodel_lpc = svm1.fit(lpc_train_data, lpc_train_label)

In [41]: svm1 = OneVsRestClassifier(NuSVC(nu=.2, kernel='linear',
decision_function_shape='ovr'))
```

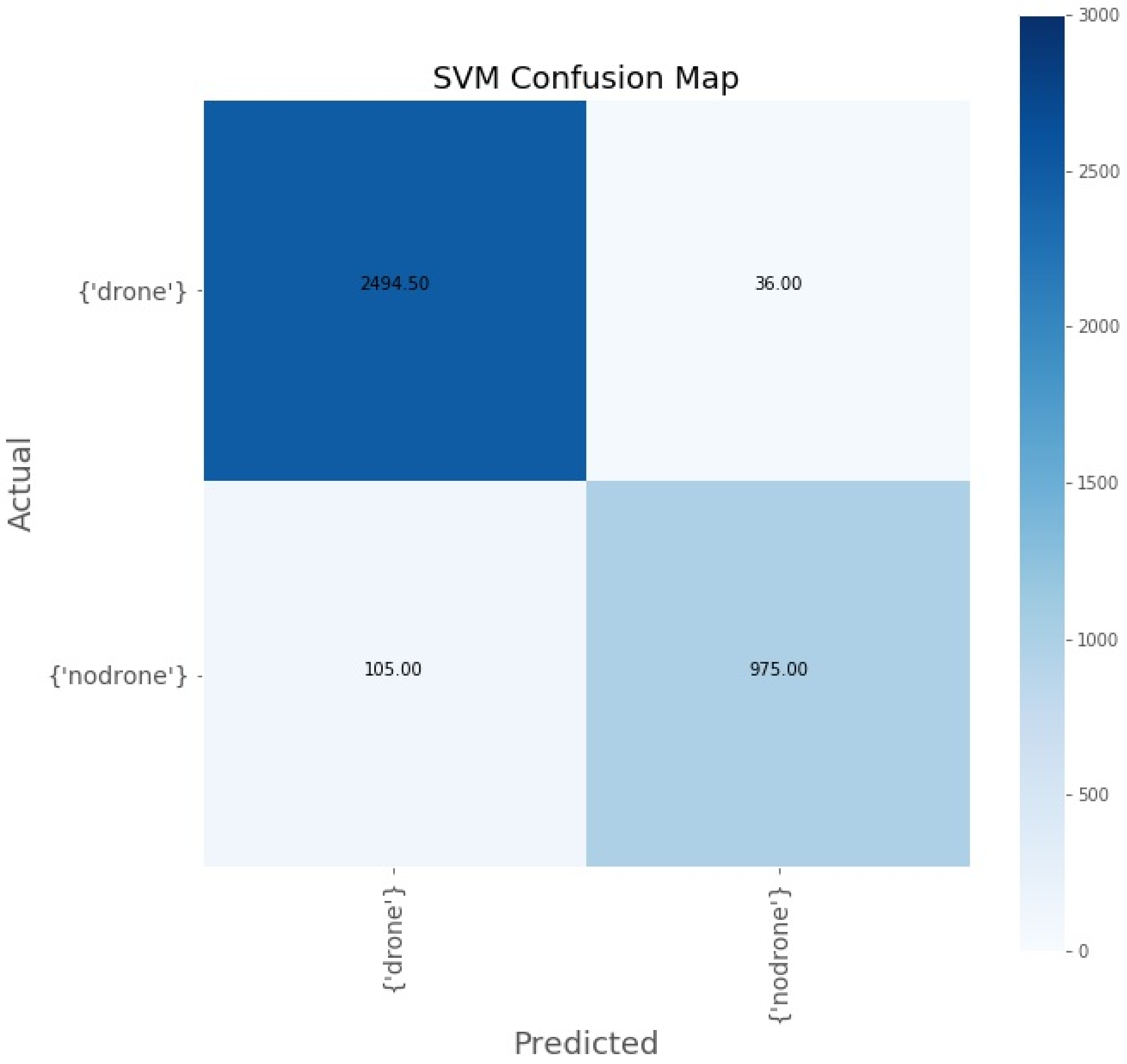
```
In [42]: svmmodel_lpc = svm1.fit(lpc_train_data, lpc_train_label)

In [43]: svc_prediction1 = svmmodel_lpc.predict(lpc_validation_data)
    ...: svc_accuracy1 = np.sum(svc_prediction1 == lpc_validation_label)/
lpc_validation_label.shape[0]
    ...: print(svc_accuracy1)
    ...: classe_names1 = label_mapping1.values()
    ...: matrix1 = confusion(lpc_validation_label, svc_prediction1)
0.4549231408392189

In [44]: #train_X = df.iloc[:-180, 10:].values
    ...: #train_y = df.iloc[:-150, 1:3].values
    ...: #test_y = df.iloc[-150:, 1:3].values
    ...: mfcc_data = []
    ...: lpc_data1 = []
    ...: with open(metafile, 'r',newline='') as f:
    ...:     reader = csv.DictReader(f, delimiter=',')
    ...:     i=0
    ...:     mfccl = []
    ...:     chromal = []
    ...:     mell = []
    ...:     spectl = []
    ...:     tonnetzl = []
    ...:     lpcl = []
    ...:     rlpcl= []
    ...:     psdl =[]
    ...:     for row in reader:
    ...:         fileid = float(row['ID'])
    ...:         lpc = [float(t) for t in row['LPCCOEFF'].strip("[]").split()]
    ...:         rlpc = [float(t) for t in row['RCOEFF'].strip("[]").split()]
    ...:         pfreq= [float(t) for t in row['Pfreq'].strip("[]").split()]
    ...:         h = row['Height']
    ...:         if h == 'n':
    ...:             label1 = "nodrone"
    ...:         else:
    ...:             label1 = "drone"
    ...:
    ...:         #features = np.empty((0,193))
    ...:         fpfeatures = np.empty((0,26))
    ...:         #ext_features = np.hstack([mfcc,chroma,mel,spect,tonnetz])
    ...:         fpext_features = np.hstack([lpc, rlpc, pfreq])
    ...:         #features = np.vstack([features,ext_features])
    ...:         fpfeatures = np.vstack([fpfeatures,fpext_features])
    ...:         i+=1
    ...:         if not np.isnan(lpc[0]):
    ...:             #mfccl.append(mfcc)
    ...:             #chromal.append(chroma)
    ...:             #mell.append(mel)
    ...:             #spectl.append(spect)
    ...:             #tonnetzl.append(tonnetz)
    ...:             lpcl.append(lpc)
    ...:             psdl.append(psd)
    ...:             rlpcl.append(rlpc)
    ...:             #mfcc_data.append([features, features.shape, label])
    ...:             lpc_data1.append([fpfeatures, fpfeatures.shape, label1])
    ...: cols=["features", "shape","label"]
    ...: #mfcc_pd1 = pd.DataFrame(data = mfcc_data, columns=cols)
    ...: lpc_pd1 = pd.DataFrame(data = lpc_data1, columns=cols)
```

```
In [45]: labels1 = set(lpc_pd['label_id'])
    ...: mapping = []
    ...: for label_id in labels1:
    ...:     label_name1 = set(lpc_pd.loc[lpc_pd['label_id'] == label_id]
['label'])
    ...:     mapping.append((label_id,label_name1))
    ...:
    ...: label_mapping1 = dict(mapping)
    ...: label_mapping1
    ...:
    ...: ll1 = [lpc_pd['features'][i].ravel() for i in range(lpc_pd.shape[0])]
    ...: lpc_pd['sample'] = pd.Series(ll1, index=lpc_pd.index)
    ...: del lpc_pd['features']
    ...:
    ...: lpc_train_data = np.array(list(lpc_pd[:-iv]['sample']))
    ...: lpc_train_label = np.array(list(lpc_pd[:-iv]['label_id']))
    ...: lpc_validation_data = np.array(list(lpc_pd[-iv:]['sample']))
    ...: lpc_validation_label = np.array(list(lpc_pd[-iv:]['label_id']))
    ...:
    ...: print(lpc_train_data.shape)
    ...: print(type(lpc_train_data))
    ...:
    ...:
    ...: lpc_train_data = np.array(list(lpc_pd[:-iv]['sample']))
    ...: lpc_train_label = np.array(list(lpc_pd[:-iv]['label_id']))
    ...: lpc_validation_data = np.array(list(lpc_pd[-iv:]['sample']))
    ...: lpc_validation_label = np.array(list(lpc_pd[-iv:]['label_id']))
    ...:
    ...: print(lpc_train_data.shape)
    ...: print(type(lpc_train_data))
    ...: svmmodel_lpc = svm1.fit(lpc_train_data, lpc_train_label)
    ...: svc_prediction1 = svmmodel_lpc.predict(lpc_validation_data)
    ...: svc_accuracy1 = np.sum(svc_prediction1 == lpc_validation_label)/
lpc_validation_label.shape[0]
    ...: print(svc_accuracy1)
    ...: classe_names1 = label_mapping1.values()
    ...: matrix1 = confusion(lpc_validation_label, svc_prediction1)
Traceback (most recent call last):

  File "<ipython-input-45-adebb89104ef>", line 10, in <module>
    ll1 = [lpc_pd['features'][i].ravel() for i in range(lpc_pd.shape[0])]

  File "<ipython-input-45-adebb89104ef>", line 10, in <listcomp>
    ll1 = [lpc_pd['features'][i].ravel() for i in range(lpc_pd.shape[0])]

  File "/home/sayan/anaconda3/lib/python3.6/site-packages/pandas/core/frame.py",
line 2139, in __getitem__
    return self._getitem_column(key)

  File "/home/sayan/anaconda3/lib/python3.6/site-packages/pandas/core/frame.py",
line 2146, in _getitem_column
    return self._get_item_cache(key)

  File "/home/sayan/anaconda3/lib/python3.6/site-packages/pandas/core/generic.py",
line 1842, in _get_item_cache
    values = self._data.get(item)

  File "/home/sayan/anaconda3/lib/python3.6/site-packages/pandas/core/
```

```
internals.py", line 3843, in get
    loc = self.items.get_loc(item)

  File "/home/sayan/anaconda3/lib/python3.6/site-packages/pandas/core/indexes/
base.py", line 2527, in get_loc
    return self._engine.get_loc(self._maybe_cast_indexer(key))

  File "pandas/_libs/index.pyx", line 117, in
pandas._libs.index.IndexEngine.get_loc

  File "pandas/_libs/index.pyx", line 139, in
pandas._libs.index.IndexEngine.get_loc

  File "pandas/_libs/hashtable_class_helper.pxi", line 1265, in
pandas._libs.hashtable.PyObjectHashTable.get_item

  File "pandas/_libs/hashtable_class_helper.pxi", line 1273, in
pandas._libs.hashtable.PyObjectHashTable.get_item

KeyError: 'features'


In [46]:

In [46]: le = LabelEncoder()
    ...: #label_num = le.fit_transform(mfcc_pd["label"])
    ...: label_num1 = le.fit_transform(lpc_pd["label"])
    ...: ohe = OneHotEncoder()
    ...: #onehot = ohe.fit_transform(label_num.reshape(-1, 1))
    ...: onehot1 = ohe.fit_transform(label_num1.reshape(-1, 1))
    ...: def labelling(pddata, ln):
    ...:     pddata.insert(loc=3, column='label_id', value=ln)
    ...:     labels = set(pddata['label'])
    ...:     print(labels)
    ...:     cnt = [[label,list(pddata['label']).count(label)] for label in
labels]
    ...:     dict_cnt = dict(cnt)
    ...:     print(dict_cnt)
    ...:     cnt_cols=["classes","occurence"]
    ...:     count_pd = pd.DataFrame(data = cnt, columns=cnt_cols)
    ...:     return dict_cnt
    ...: dictlpc = labelling(lpc_pd, label_num1)
    ...: it = int(len(data)*0.8)
    ...: iv = int(len(data)*0.2)
    ...: labels1 = set(lpc_pd['label_id'])
    ...: mapping = []
    ...: for label_id in labels1:
    ...:     label_name1 = set(lpc_pd.loc[lpc_pd['label_id'] == label_id]
['label'])
    ...:     mapping.append((label_id,label_name1))
    ...:
    ...: label_mapping1 = dict(mapping)
    ...: label_mapping1
    ...:
    ...: ll1 = [lpc_pd['features'][i].ravel() for i in range(lpc_pd.shape[0])]
    ...: lpc_pd['sample'] = pd.Series(ll1, index=lpc_pd.index)
    ...: del lpc_pd['features']
    ...:
    ...: lpc_train_data = np.array(list(lpc_pd[:-iv]['sample']))
    ...: lpc_train_label = np.array(list(lpc_pd[:-iv]['label_id']))
    ...: lpc_validation_data = np.array(list(lpc_pd[-iv:]['sample']))
    ...: lpc_validation_label = np.array(list(lpc_pd[-iv:]['label_id']))
    ...:
    ...: print(lpc_train_data.shape)
    ...: print(type(lpc_train_data))
    ...: def confusion(true, predicted):
    ...:     matrix = np.zeros([5,5])
    ...:     #d = 0
    ...:     for t, p in zip(true, predicted):
    ...:         matrix[t,p] += 1.5
    ...:     #    d += 1
    ...:     #print(d)
    ...:     return matrix
Traceback (most recent call last):

  File "<ipython-input-46-97747f819c45>", line 17, in <module>
    dictlpc = labelling(lpc_pd, label_num1)

  File "<ipython-input-46-97747f819c45>", line 8, in labelling
    pddata.insert(loc=3, column='label_id', value=ln)

  File "/home/sayan/anaconda3/lib/python3.6/site-packages/pandas/core/frame.py",
line 2613, in insert
    allow_duplicates=allow_duplicates)

  File "/home/sayan/anaconda3/lib/python3.6/site-packages/pandas/core/
internals.py", line 4063, in insert
    raise ValueError('cannot insert {}, already exists'.format(item))

ValueError: cannot insert label_id, already exists


In [47]:

In [47]: lpc_pd = lpc_pd1

In [48]: le = LabelEncoder()
    ...: #label_num = le.fit_transform(mfcc_pd["label"])
    ...: label_num1 = le.fit_transform(lpc_pd["label"])
    ...: ohe = OneHotEncoder()
    ...: #onehot = ohe.fit_transform(label_num.reshape(-1, 1))
    ...: onehot1 = ohe.fit_transform(label_num1.reshape(-1, 1))
    ...: def labelling(pddata, ln):
    ...:     pddata.insert(loc=3, column='label_id', value=ln)
    ...:     labels = set(pddata['label'])
    ...:     print(labels)
    ...:     cnt = [[label,list(pddata['label']).count(label)] for label in
labels]
    ...:     dict_cnt = dict(cnt)
    ...:     print(dict_cnt)
    ...:     cnt_cols=["classes","occurence"]
    ...:     count_pd = pd.DataFrame(data = cnt, columns=cnt_cols)
    ...:     return dict_cnt
    ...: dictlpc = labelling(lpc_pd, label_num1)
    ...: it = int(len(data)*0.8)
    ...: iv = int(len(data)*0.2)
    ...: labels1 = set(lpc_pd['label_id'])
    ...: mapping = []
    ...: for label_id in labels1:
    ...:     label_name1 = set(lpc_pd.loc[lpc_pd['label_id'] == label_id]
['label'])
    ...:     mapping.append((label_id,label_name1))
    ...:
    ...: label_mapping1 = dict(mapping)
    ...: label_mapping1
    ...:
    ...: ll1 = [lpc_pd['features'][i].ravel() for i in range(lpc_pd.shape[0])]
    ...: lpc_pd['sample'] = pd.Series(ll1, index=lpc_pd.index)
    ...: del lpc_pd['features']
    ...:
    ...: lpc_train_data = np.array(list(lpc_pd[:-iv]['sample']))
    ...: lpc_train_label = np.array(list(lpc_pd[:-iv]['label_id']))
    ...: lpc_validation_data = np.array(list(lpc_pd[-iv:]['sample']))
    ...: lpc_validation_label = np.array(list(lpc_pd[-iv:]['label_id']))
    ...:
    ...: print(lpc_train_data.shape)
    ...: print(type(lpc_train_data))
    ...: def confusion(true, predicted):
    ...:     matrix = np.zeros([5,5])
    ...:     #d = 0
    ...:     for t, p in zip(true, predicted):
    ...:         matrix[t,p] += 1.5
    ...:     #    d += 1
    ...:     #print(d)
    ...:     return matrix
{'nodrone', 'drone'}
{'nodrone': 3461, 'drone': 8575}
(9629, 26)
<class 'numpy.ndarray'>

In [49]: svm1 = OneVsRestClassifier(NuSVC(nu=.2, kernel='linear',
decision_function_shape='ovr'))

In [49]:

In [50]: svmmodel_lpc = svm1.fit(lpc_train_data, lpc_train_label)

In [51]: svc_prediction1 = svmmodel_lpc.predict(lpc_validation_data)
    ...:
    ...: svc_accuracy1 = np.sum(svc_prediction1 == lpc_validation_label)/
lpc_validation_label.shape[0]
    ...: print(svc_accuracy1)
    ...: classe_names1 = label_mapping1.values()
    ...: matrix1 = confusion(lpc_validation_label, svc_prediction1)
0.9609472372247612

In [52]: lpc_validation_data
Out[52]:
array([[ -3.4368625 ,   5.42598345,  -4.92921068, ..., 141.5       ,
         142.        , 140.75      ],
       [ -3.50103609,   6.52759846,  -9.12678347, ..., 145.        ,
         145.25      , 138.5       ],
       [ -3.01015804,   5.44509991,  -7.50789628, ..., 153.        ,
         136.5       , 136.25      ],
       ...,
       [ -3.27608498,   6.08097336,  -8.57974445, ..., 143.25      ,
         154.25      , 142.        ],
       [ -2.80631694,   4.14604324,  -4.48216975, ..., 139.25      ,
        139.        , 142.        ],
       [ -3.25702612,   5.97489515,  -8.2410552 , ..., 139.75      ,
        157.25      , 156.        ]])

In [53]: plt.figure(figsize=[10,10])
    ...: plt.imshow(matrix1, cmap=plt.cm.Blues, interpolation='nearest',  vmin=0,
vmax=100)
    ...: plt.colorbar()
    ...: plt.title('SVM Confusion Map', fontsize=18)
    ...: plt.ylabel('Actual', fontsize=18)
    ...: plt.xlabel('Predicted', fontsize=18)
    ...: plt.grid(b=False)
    ...: plt.yticks(range(2), classe_names1, fontsize=14)
    ...: plt.xticks(range(2), classe_names1, fontsize=14, rotation='vertical')
    ...: for i, j in itertools.product(range(matrix1.shape[0]),
range(matrix1.shape[1])):
    ...:     plt.text(j, i, format(matrix1[i, j], '.
2f'),horizontalalignment="center",color="black")
    ...: plt.show()
```



```
In [54]: def confusion(true, predicted):
    ...:     matrix = np.zeros([2,2])
    ...:     #d = 0
    ...:     for t, p in zip(true, predicted):
    ...:         matrix[t,p] += 1.5
    ...:     #    d += 1
    ...:     #print(d)
    ...:     return matrix
    ...: svc_accuracy1 = np.sum(svc_prediction1 == lpc_validation_label)/
lpc_validation_label.shape[0]
    ...: print(svc_accuracy1)
    ...: classe_names = label_mapping1.values()
    ...: matrix1 = confusion(validation_label, svc_prediction)
    ...: plt.figure(figsize=[10,10])
    ...: plt.imshow(matrix1, cmap=plt.cm.Blues, interpolation='nearest',  vmin=0,
vmax=3000)
    ...: plt.colorbar()
    ...: plt.title('SVM Confusion Map', fontsize=18)
    ...: plt.ylabel('Actual', fontsize=18)
    ...: plt.xlabel('Predicted', fontsize=18)
    ...: plt.grid(b=False)
    ...: plt.yticks(range(2), classe_names1, fontsize=14)
    ...: plt.xticks(range(2), classe_names1, fontsize=14, rotation='vertical')
    ...: for i, j in itertools.product(range(matrix1.shape[0]),
range(matrix1.shape[1])):
    ...:     plt.text(j, i, format(matrix1[i, j], '.
2f'),horizontalalignment="center",color="black")
    ...: plt.show()
0.9609472372247612
Traceback (most recent call last):

  File "<ipython-input-54-926f855d1d49>", line 12, in <module>
    matrix1 = confusion(validation_label, svc_prediction)

NameError: name 'validation_label' is not defined


In [55]:

In [55]: def confusion(true, predicted):
    ...:     matrix = np.zeros([2,2])
    ...:     #d = 0
    ...:     for t, p in zip(true, predicted):
    ...:         matrix[t,p] += 1.5
    ...:     #    d += 1
    ...:     #print(d)
    ...:     return matrix
    ...: svc_accuracy1 = np.sum(svc_prediction1 == lpc_validation_label)/
lpc_validation_label.shape[0]
    ...: print(svc_accuracy1)
    ...: classe_names = label_mapping1.values()
    ...: matrix1 = confusion(lpc_validation_label, svc_prediction1)
    ...: plt.figure(figsize=[10,10])
    ...: plt.imshow(matrix1, cmap=plt.cm.Blues, interpolation='nearest',  vmin=0,
vmax=3000)
    ...: plt.colorbar()
    ...: plt.title('SVM Confusion Map', fontsize=18)
    ...: plt.ylabel('Actual', fontsize=18)
    ...: plt.xlabel('Predicted', fontsize=18)
    ...: plt.grid(b=False)
    ...: plt.yticks(range(2), classe_names1, fontsize=14)
    ...: plt.xticks(range(2), classe_names1, fontsize=14, rotation='vertical')
    ...: for i, j in itertools.product(range(matrix1.shape[0]),
range(matrix1.shape[1])):
    ...:     plt.text(j, i, format(matrix1[i, j], '.
2f'),horizontalalignment="center",color="black")
    ...: plt.show()
0.9609472372247612
```



```
In [56]: joblib.dump(svm, 'input/dronedetectionfinal_new.pkl')
Traceback (most recent call last):

  File "<ipython-input-56-03ddbdc97cb1>", line 1, in <module>
    joblib.dump(svm, 'input/dronedetectionfinal_new.pkl')

  File "/home/sayan/anaconda3/lib/python3.6/site-packages/sklearn/externals/
joblib/numpy_pickle.py", line 484, in dump
    NumpyPickler(f, protocol=protocol).dump(value)

  File "/home/sayan/anaconda3/lib/python3.6/pickle.py", line 409, in dump
    self.save(obj)

  File "/home/sayan/anaconda3/lib/python3.6/site-packages/sklearn/externals/
joblib/numpy_pickle.py", line 281, in save
    return Pickler.save(self, obj)

  File "/home/sayan/anaconda3/lib/python3.6/pickle.py", line 496, in save
    rv = reduce(self.proto)
```

```
TypeError: can't pickle module objects


In [57]:

In [57]: joblib.dump(svm1, 'input/dronedetectionfinal_new.pkl')
Out[57]: ['input/dronedetectionfinal_new.pkl']

In [58]:
```