
agentMET4FOF Documentation

Bang Xiang Yong

Apr 22, 2020

Getting started:

1	Getting started	3
2	Download data	5
3	Data Preprocessing	7
4	Train model Bayesian Neural Network	11
5	Run Multi Agent System	67
6	Agent Dashboard User Interface	73
7	agentMET4FOF agents	81
8	agentMET4FOF streams	89
9	Indices and tables	91
10	References	93
	Bibliography	95
	Python Module Index	97
	Index	99

agentMET4FOF is a Python library developed at the [Institute for Manufacturing](#) of the University of Cambridge (UK) as part of the European joint Research Project [EMPIR 17IND12 Met4FoF](#).

For the *agentMET4FOF* homepage go to [GitHub](#).

agentMET4FOF is written in Python 3.

CHAPTER 1

Getting started

Here goes your text.

More info on how to create the contents [here](#)

Multi-Agent System for Metrology for Factory of the Future (Met4FoF) Code This is supported by European Metrology Programme for Innovation and Research (EMPIR) under the project Metrology for the Factory of the Future (Met4FoF), project number 17IND12. (<https://www.ptb.de/empir2018/met4fof/home/>)

About — - How can metrological input be incorporated into an agent-based system for addressing uncertainty of machine learning in future manufacturing? - Includes agent-based simulation and implementation

Updates — - Implemented Sensor Agent, Aggregator Agent, Predictor Agent, DecisionMaker Agent, Sensor Network Agent - Able to handle multiple Sensor Agents & Predictor Agents (each equipped with a different model) - Implemented with ZEMA condition monitoring of hydraulic system data set as use case [\[!DOI\]\(https://zenodo.org/badge/DOI/10.5281/zenodo.1323611.svg\)](https://zenodo.org/badge/DOI/10.5281/zenodo.1323611.svg) [}}\(https://doi.org/10.5281/zenodo.1323611\)](https://doi.org/10.5281/zenodo.1323611) - Implemented web visualization with user interface

To run — - Run Code 01-03 to prepare the ML models, and run Code 04 to start and run the agents. While Code 04 is running, run Code 05 in separate terminal to visualize them.

Screenshot of web visualization [\[Web Screenshot\]\(https://github.com/bangxiangyong/agentMet4FoF/blob/master/screenshot_met4fof.png\)](https://github.com/bangxiangyong/agentMet4FoF/blob/master/screenshot_met4fof.png)

Note — - In the event of agents not terminating cleanly, run `taskkill /f /im python.exe /t`` in Windows Command Prompt to terminate all background python processes.

CHAPTER 2

Download data

Firstly we download data for condition monitoring from zenodo at <https://doi.org/10.5281/zenodo.1323611>

```
[1]: data_url='https://zenodo.org/record/1323611/files/data.zip?download=1'
```

```
[2]: import os, requests, zipfile, io

def download_and_extract(url, destination, force=False):
    response = requests.get(url)
    zipDocument = zipfile.ZipFile(io.BytesIO(response.content))
    # Attempt to see if we are going to overwrite anything
    if not force:
        abort = False
        for file in zipDocument.filelist:
            if os.path.isfile(os.path.join(destination, file.filename)):
                print(file.filename, '↪')
                ↪ 'already exists. If you want to overwrite the file call the method with force=True'
                abort = True
        if abort:
            print('Zip file was not extracted')
            return

    zipDocument.extractall(destination)
```

```
[3]: download_and_extract(data_url, 'Dataset/ZEMA_Hydraulic/')

[ ]:
```


We preprocess and resample the raw data .txt files we downloaded earlier into numpy.

3.1 Resample 10Hz and 100Hz data to 1Hz

```
[1]: import numpy as np

data_path = "Dataset/ZEMA_Hydraulic/"

filenames_input_data_10Hz = ["fs1","fs2"]
filenames_input_data_10Hz = [file + ".txt" for file in filenames_input_data_10Hz]

filenames_input_data_100Hz = ["ps1","ps2","ps3","ps4","ps5","ps6","eps1"]
filenames_input_data_100Hz = [file + ".txt" for file in filenames_input_data_100Hz]

data_input_data_10Hz = np.zeros((2205,600,len(filenames_input_data_10Hz)))
data_input_data_100Hz = np.zeros((2205,6000,len(filenames_input_data_100Hz)))

for id_,file_name in enumerate(filenames_input_data_10Hz):
    input_data = np.loadtxt(data_path + file_name, delimiter = "\t")
    data_input_data_10Hz[:, :,id_] = input_data.copy()

for id_,file_name in enumerate(filenames_input_data_100Hz):
    input_data = np.loadtxt(data_path + file_name, delimiter = "\t")
    data_input_data_100Hz[:, :,id_] = input_data.copy()

filenames_input_data_10Hz_resampled = ["res_"+file for file in filenames_input_data_
↪10Hz]
filenames_input_data_100Hz_resampled = ["res_"+file for file in filenames_input_data_
↪100Hz]

#resample 10Hz
resample = np.linspace(0,600-1, num =60, dtype="int")
```

(continues on next page)

(continued from previous page)

```

data_resampled_10Hz=data_input_data_10Hz[:,resample,:]

#resample 100Hz
resample = np.linspace(0,5999, num =60,dtype="int")
data_resampled_100Hz=data_input_data_100Hz[:,resample,:]

#save file
for id_,file_name in enumerate(filenamees_input_data_10Hz_resampled):
    np.savetxt(data_path+file_name,data_resampled_10Hz[:, :,id_],delimiter='\t')
for id_,file_name in enumerate(filenamees_input_data_100Hz_resampled):
    np.savetxt(data_path+file_name,data_resampled_100Hz[:, :,id_],delimiter='\t')

```

3.2 Load all the 1Hz data

Load all data including the resampled sensors into numpy arrays

```

[2]: #save data
datarows = 2205
seq_length = 60

#deal with inputs data
filenamees_input_data_1Hz = ["ts1","ts2","ts3","ts4","vs1","se","res_fs1","res_fs2",
    ↪ "res_ps1","res_ps2","res_ps3","res_ps4","res_ps5","res_ps6","res_eps1","ce","cp"]
filenamees_input_data_1Hz = [file + ".txt" for file in filenamees_input_data_1Hz]
filename_target_data = "profile.txt"

data_input_data_1Hz = np.zeros((datarows,seq_length,len(filenamees_input_data_1Hz)))

for id_,file_name in enumerate(filenamees_input_data_1Hz):
    input_data = np.loadtxt(data_path + file_name, delimiter = "\t")
    data_input_data_1Hz[:, :,id_] = input_data.copy()

```

3.3 Load the target multi-target, multi-class output data

We load them and preprocess into one hot vector

```

[3]: #deal with output data now
targets_data = np.loadtxt(data_path+filename_target_data, delimiter = "\t")

#conversion of outputs to one hot
def makeOneHotVectorMap(length):
    map_toOneHot = {}
    for i in range(length):
        oneHot = np.zeros(length)
        oneHot[i] = 1
        map_toOneHot[i] = oneHot
    return map_toOneHot

```

(continues on next page)

(continued from previous page)

```

id2x_dictionaries = []
x2id_dictionaries = []
id2onehot_dictionaries = []

for label in range(targets_data.shape[1]):
    label_column = list(set(targets_data[:,label]))
    label_column.sort(reverse=True)
    id2x_dictionary = {}
    x2id_dictionary = {}
    id2onehot_dictionary = makeOneHotVectorMap(len(label_column))
    for i in range(len(label_column)):
        id2x_dictionary[i] = label_column[i]
        x2id_dictionary[label_column[i]] = i
    id2x_dictionaries+= [id2x_dictionary]
    x2id_dictionaries+= [x2id_dictionary]
    id2onehot_dictionaries+= [id2onehot_dictionary]

#convert a row into one-hot coded multi-class multi-label
onehot_tensor_output = []
id_output = []
for row in range(targets_data.shape[0]):
    row_output_data= targets_data[row]
    onehots_row = []
    id_row = []
    for label in range(row_output_data.shape[0]):
        id_ = x2id_dictionaries[label][row_output_data[label]]
        onehot= id2onehot_dictionaries[label][id_]
        onehots_row = np.append(onehots_row,onehot)
        id_row = np.append(id_row,id_)
    id_output+= [id_row]
    onehot_tensor_output += [onehots_row]
onehot_tensor_output = np.array(onehot_tensor_output)
id_tensor_output = np.array(id_output)

tensor_output = id_tensor_output
all_tensor_output = id_tensor_output

```

3.4 Pickle data

```

[5]: import os
import pickle

pickle_folder= "pickles"

if os.path.exists(pickle_folder) == False:
    os.mkdir(pickle_folder)

#Pickle them
pickle.dump(data_input_data_1Hz, open( pickle_folder+"/data_input_data_1Hz_full.p", "wb" ) )
pickle.dump(data_input_data_10Hz, open( pickle_folder+"/data_input_data_10Hz.p", "wb" ) )
pickle.dump(data_input_data_100Hz, open( pickle_folder+"/data_input_data_100Hz.p", "wb" ) )

```

(continues on next page)

(continued from previous page)

```
pickle.dump(id2onehot_dictionaries, open( pickle_folder+"/id2onehot_dictionaries.p",  
↪ "wb" ) )  
pickle.dump(all_tensor_output, open( pickle_folder+"/zema_outputs.p", "wb" ) )
```

```
[ ]:
```

Train model Bayesian Neural Network

We train and evaluate the BNN on the dataset which we loaded earlier.

Credits to Felix Laumann (https://github.com/felix-laumann/Bayesian_CNN) for the method of training BNN with Bayes by Backprop and the model code.

```
[1]: import torch
import pickle
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler
import random
from torch.autograd import Variable
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels
import matplotlib.pyplot as plt
import pandas as pd
from copy import copy
from ML_models.BBBlayers import BBBLinearFactorial
from ML_models.BBBlayers import GaussianVariationalInference
from ML_models.BNN_Wrapper import BNN_Wrapper
from ML_models.BNN_Wrapper import BNN_Full
import seaborn as sns

from scipy.stats import skew
from scipy.stats import kurtosis
from scipy.stats import sem
from scipy.fftpack import fft
from sklearn.metrics import f1_score
```

4.1 Load data and label outputs

```
[2]: pickle_path="pickles/"
data_input = pickle.load( open( pickle_path+"data_input_data_1Hz_full.p", "rb" ) )
data_output = pickle.load( open( pickle_path+"zema_outputs.p", "rb" ) )

output_labels = [{0: "Optimal", 1: "Reduced", 2: "Nearly Fail"},
                 {0: "Optimal", 1: "Small lag", 2: "Severe lag", 3: "Nearly Fail"},
                 {0: "No Leakage", 1: "Weak Leakage", 2: "Severe Leakage"},
                 {0: "Optimal", 1: "Slightly Reduced", 2: "Severely Reduced", 3: "Nearly Fail"},
                 {0: "Stable", 1: "Unstable"}]
output_sizes = [3,4,3,4,2]
```

4.2 Initialization

Setup parameters such as learning rate, number of Monte Carlo samples during predictions, training epoch, certainty threshold

```
[3]: learning_rate = 0.005
num_samples = 50
num_epochs = 300
certainty_threshold = 80

X_data = data_input
Y_data = data_output
```

4.3 Shuffle data

1. Shuffle data randomly to have a random distribution
2. Split the time series into segments if necessary (n=1 means no segmentation occurs)

```
[4]: randomShuffling= True

#randomShuffling
if(randomShuffling == True):
    index_list = np.arange(X_data.shape[0])
    random.shuffle(index_list)
    Y_data=Y_data[index_list,:]
    X_data=X_data[index_list,:,:]

def split_segment(X_data,split_n =1):
    X_data_split=np.split(X_data,split_n,axis=1)
    X_data_split=np.moveaxis(X_data_split, 0, -2)
    X_data_split=X_data_split.reshape((X_data_split.shape[0],X_data_split.shape[1],-
    ↪1))
    return X_data_split

X_data=split_segment(X_data)
```


4.4 Train and evaluate data in k-fold validation

- Setup empty arrays for keeping results from each k-fold validation iteration
- Training consists of two loops:
 1. Outer loop: Train a model for each prediction task (from valve, pump, accumulator, etc)
 2. Inner loop: k-fold iteration
- Results are stored in arrays

```
[5]: simulate_broken_sensor = False #experimental and not fully implemented yet

y_pred_kfold = []
y_actuals_kfold = []
mse_kfold = []
certainties_kfold = []
confidences_kfold = []
DL_model_kfold = []
final_results = []

for target in range(len(output_sizes)):
    selected_output = target
    output_size = output_sizes[selected_output]
    bnn_wrapper = BNN_Wrapper(BNN_Full, output_size=output_size)

    kfold_times = 0
    kfold_limit = 5
    #kfold validation training
    kf = KFold(n_splits=5)
    fl_score_model_kfold = []
    p_accurate_certain_kfold = []
    p_accurate_uncertain_kfold = []

    for train, test in kf.split(X_data, y=Y_data):

        if (kfold_times > kfold_limit):
            break
        print("%s %s" % (train, test))

        x_train = X_data[train]
        y_train = Y_data[train, selected_output]
        x_test = X_data[test]
        y_test = Y_data[test, selected_output]

        #simulate broken sensors on test set
        if simulate_broken_sensor == True:
            x_test_brokenSensor = Variable(torch.from_numpy(x_test).float()).repeat(x_
            ↪ train.shape[-1]+1, 1, 1, 1)
            for corrupted_sensor in range(x_train.shape[-1]):
                x_test_brokenSensor[corrupted_sensor, :, :, corrupted_sensor] = torch.
            ↪ randn_like(x_test_brokenSensor[corrupted_sensor, :, :, corrupted_sensor])
                x_test_brokenSensor = x_test_brokenSensor.view(-1, x_test_brokenSensor.
            ↪ shape[2], x_test_brokenSensor.shape[3])
                x_test_brokenSensor = x_test_brokenSensor.cpu().detach().numpy()
            x_test = copy(x_test_brokenSensor)
```

(continues on next page)

(continued from previous page)

```

y_test_tensor = Variable(torch.from_numpy(y_test).long())
if simulate_broken_sensor == True:
    y_test_tensor_repeated = y_test_tensor.repeat(x_train.shape[-1]+1,1)
    y_test_tensor_repeated = y_test_tensor_repeated.view(-1)
elif simulate_broken_sensor == False:
    y_test_tensor_repeated = y_test_tensor

#perform feature extraction on train and test
df_feats_train = bnn_wrapper.extract_features(x_train)
x_train = df_feats_train.values
df_feats_test = bnn_wrapper.extract_features(x_test)
x_test = df_feats_test.values

#train model
bnn_wrapper.train_model(x_train,y_train,learning_rate= learning_rate,num_
↳epochs = num_epochs)

#predict using model
y_pred_test,certainties,confidences = bnn_wrapper.predict_model_wUnc(x_test,
↳num_samples=num_samples)

#now form all the kfoldds
DL_model_kfold.append(bnn_wrapper.trained_model)
y_pred_kfold.append(y_pred_test)
y_actuals_kfold.append(y_test_tensor_repeated.cpu().detach().numpy())
certainties_kfold.append(np.array(certainties))
confidences_kfold.append(np.array(confidences))
kfold_times = kfold_times+1

#evaluate model
y_true_np=np.array(y_actuals_kfold).reshape(-1)
y_pred_np=np.array(y_pred_kfold).reshape(-1)
certainties_np = np.array(certainties_kfold).reshape(-1)

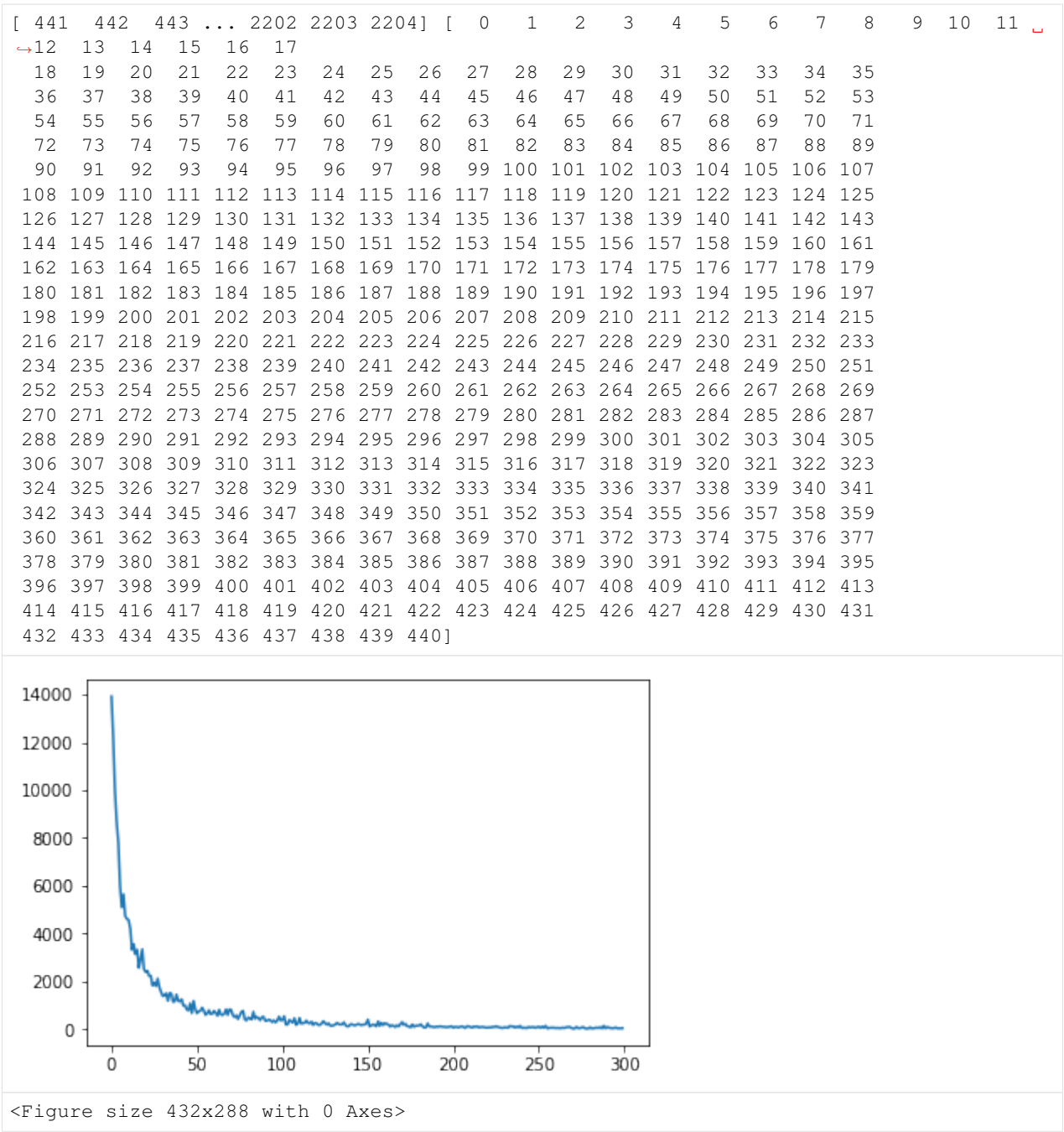
f1_score_model, p_accurate_certain, p_accurate_uncertain = bnn_wrapper.
↳evaluate_model(y_true_np,y_pred_np,certainties_np,certainty_threshold=certainy_
↳threshold)
f1_score_model_kfold.append(f1_score_model)
p_accurate_certain_kfold.append(p_accurate_certain)
p_accurate_uncertain_kfold.append(p_accurate_uncertain)

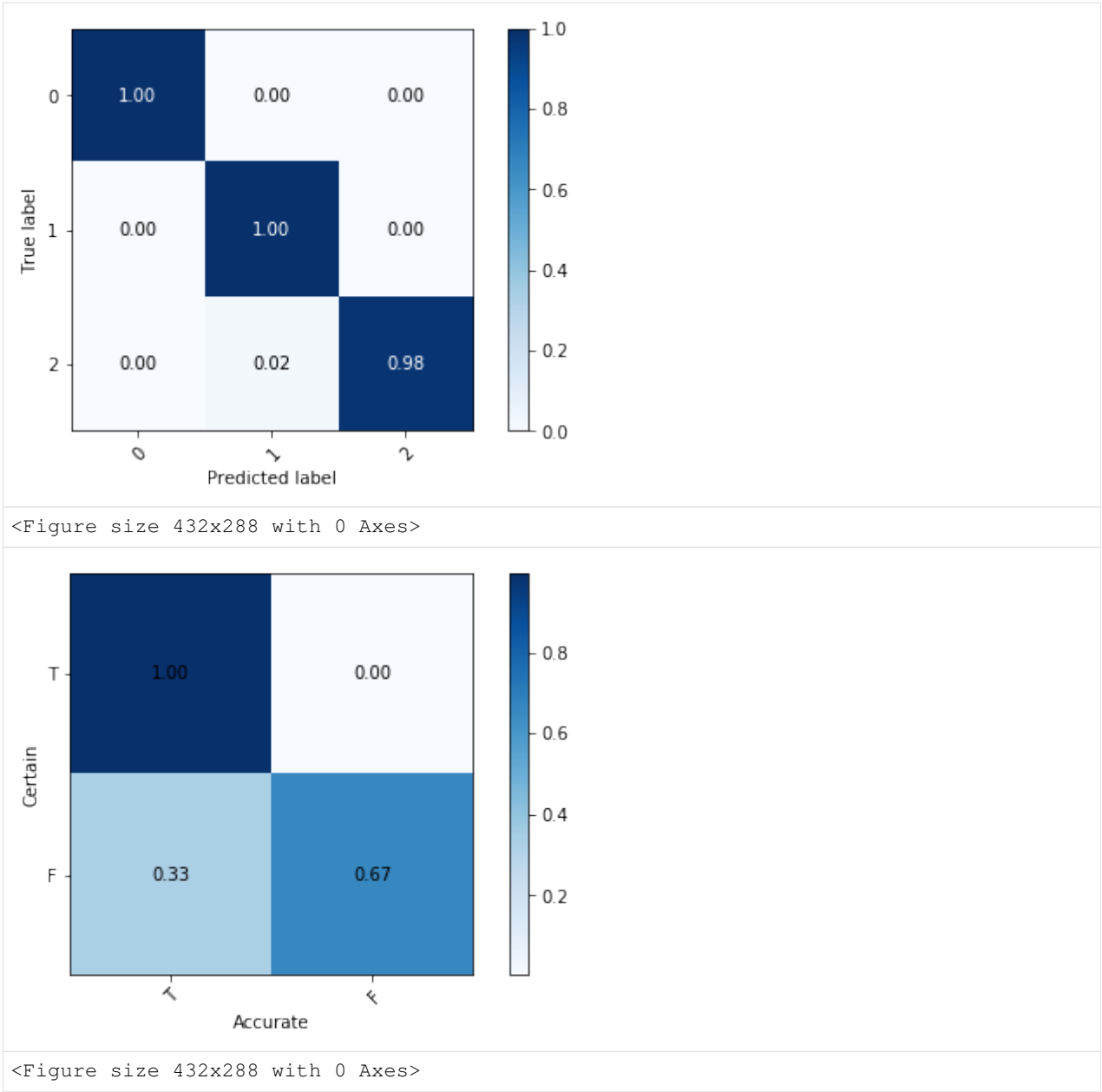
print("Target output: ", selected_output)
print("F1 SCORE: ",np.mean(f1_score_model_kfold),"+-",sem(f1_score_model_kfold),
↳f1_score_model_kfold)
print("P(Acc | Certain): ",np.mean(p_accurate_certain_kfold),"+-",sem(p_accurate_
↳certain_kfold),p_accurate_certain_kfold)
print("P(Acc | Uncertain): ",np.mean(p_accurate_uncertain_kfold),"+-",sem(p_
↳accurate_uncertain_kfold),p_accurate_uncertain_kfold)

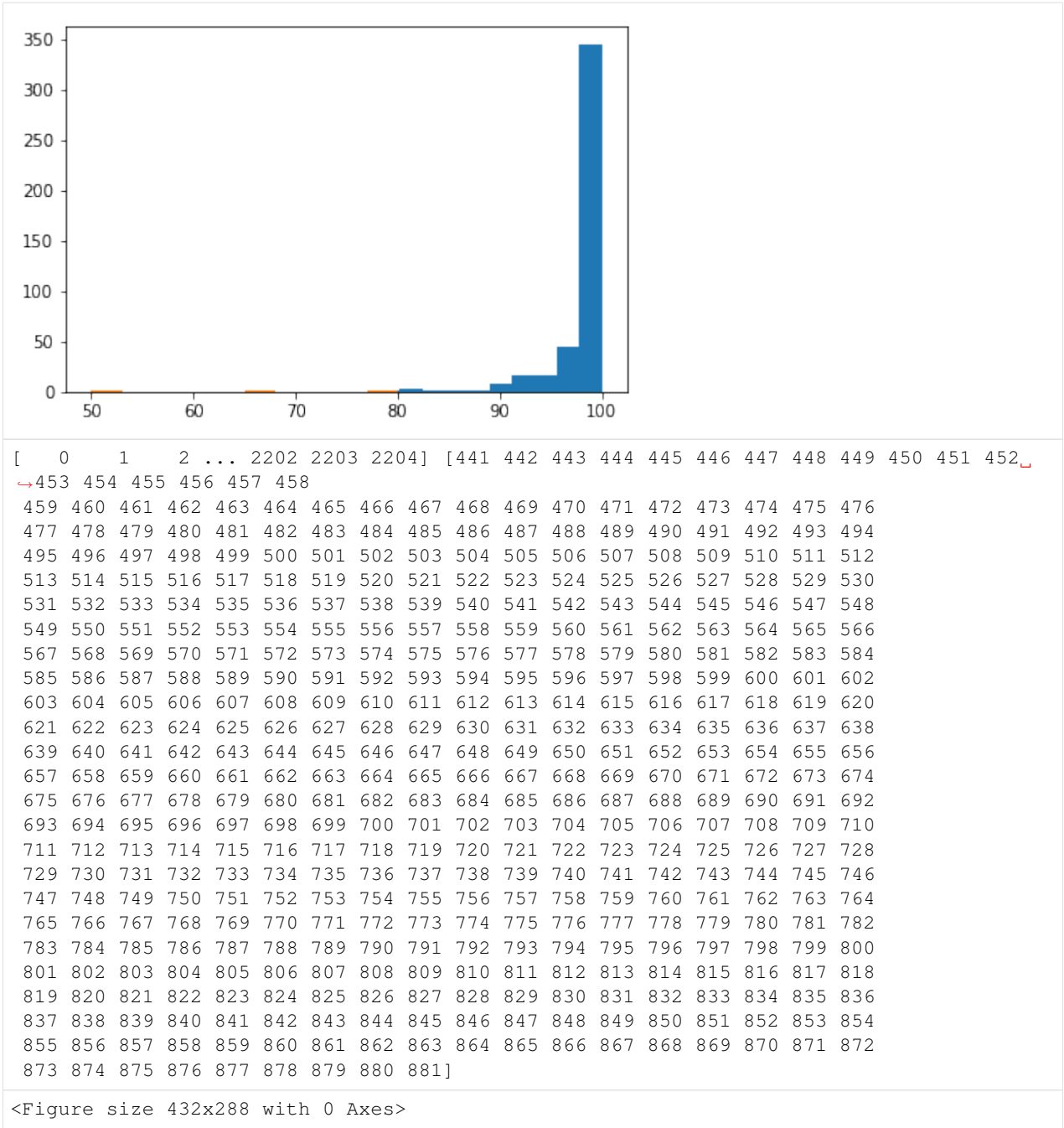
final_results.append([f1_score_model_kfold, p_accurate_certain_kfold, p_accurate_
↳uncertain_kfold])

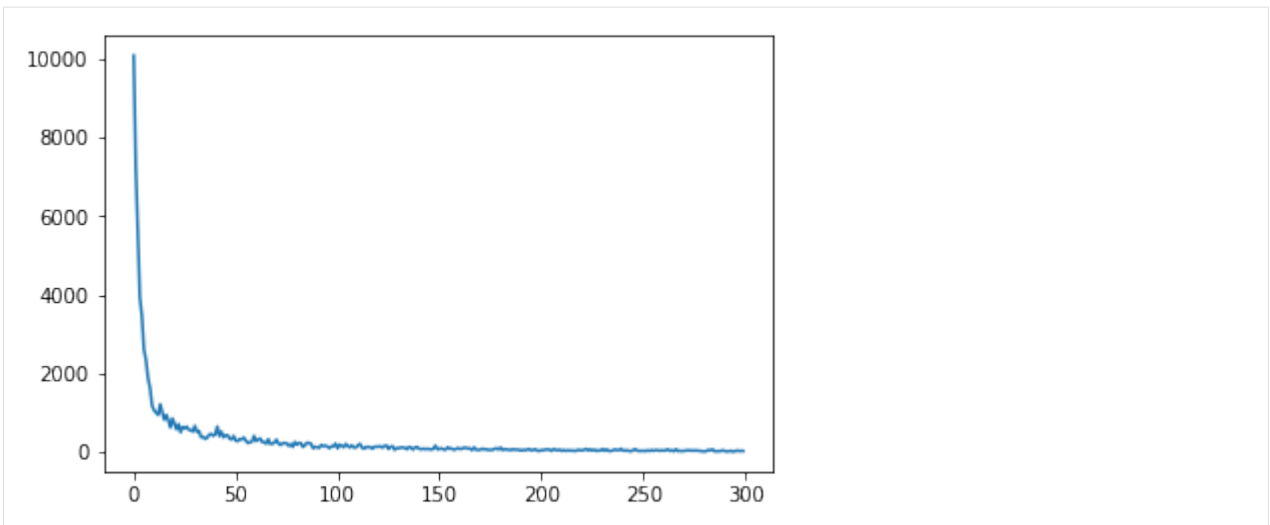
pickle.dump(bnn_wrapper, open( pickle_path+"bnn_wrapper_"+str(selected_
↳output)+".p", "wb" ) )

```

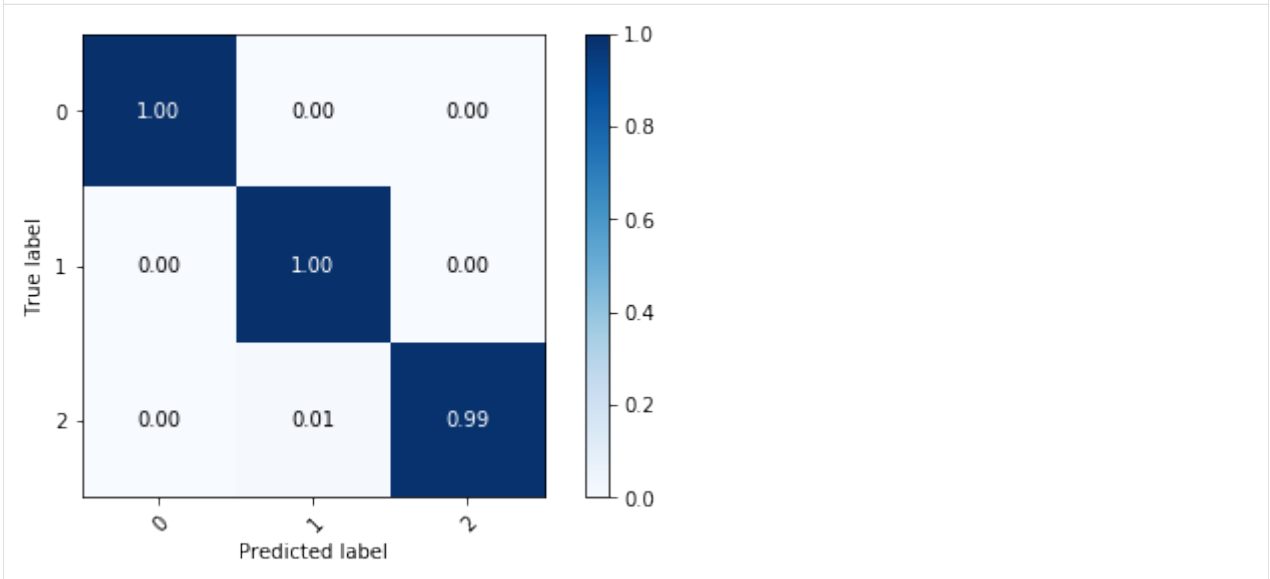




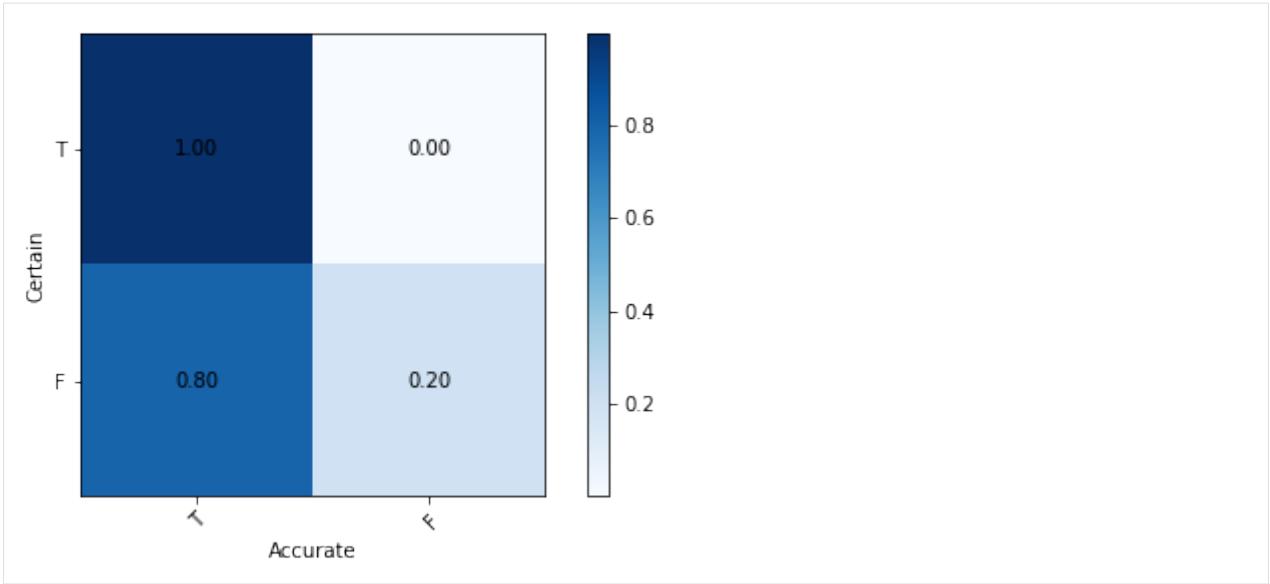




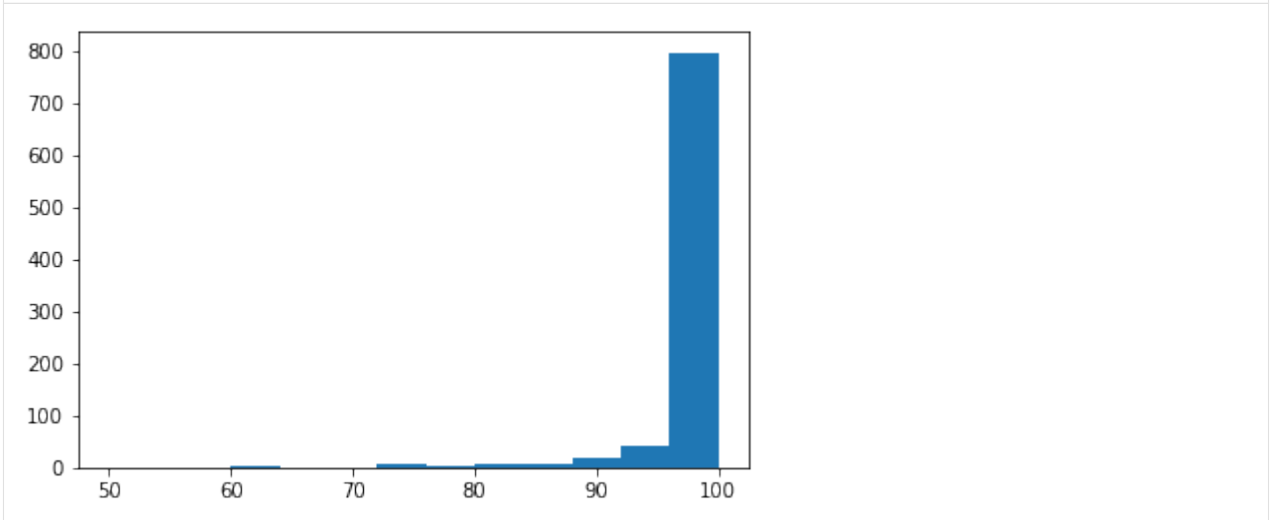
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



[0	1	2	...	2202	2203	2204]	[882	883	884	885	886	887	888	889	890	...
↪	891	892	893	894	895													
	896	897	898	899	900	901	902	903	904	905	906	907	908	909				
	910	911	912	913	914	915	916	917	918	919	920	921	922	923				
	924	925	926	927	928	929	930	931	932	933	934	935	936	937				
	938	939	940	941	942	943	944	945	946	947	948	949	950	951				
	952	953	954	955	956	957	958	959	960	961	962	963	964	965				
	966	967	968	969	970	971	972	973	974	975	976	977	978	979				
	980	981	982	983	984	985	986	987	988	989	990	991	992	993				
	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007				
	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021				
	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035				
	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049				
	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063				
	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077				
	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091				
	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105				
	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119				

(continues on next page)

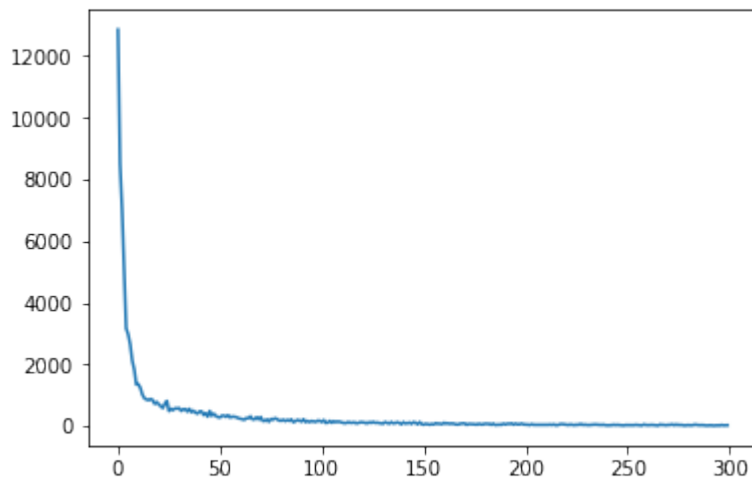
(continued from previous page)

```

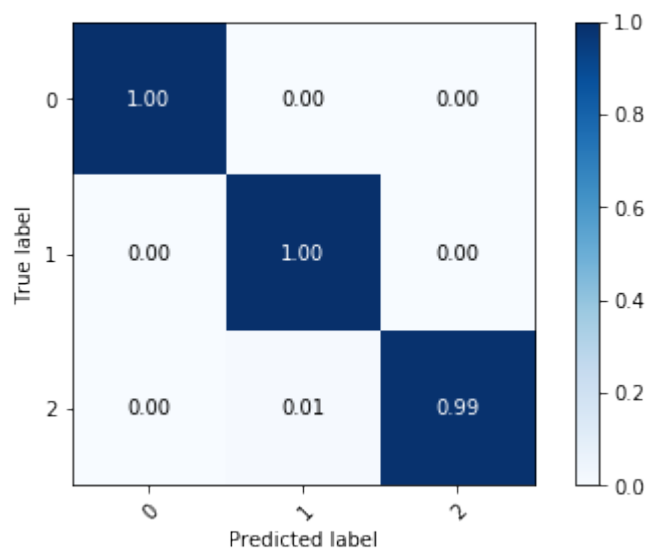
1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133
1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147
1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161
1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175
1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189
1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217
1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231
1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245
1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259
1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273
1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287
1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301
1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315
1316 1317 1318 1319 1320 1321 1322]

```

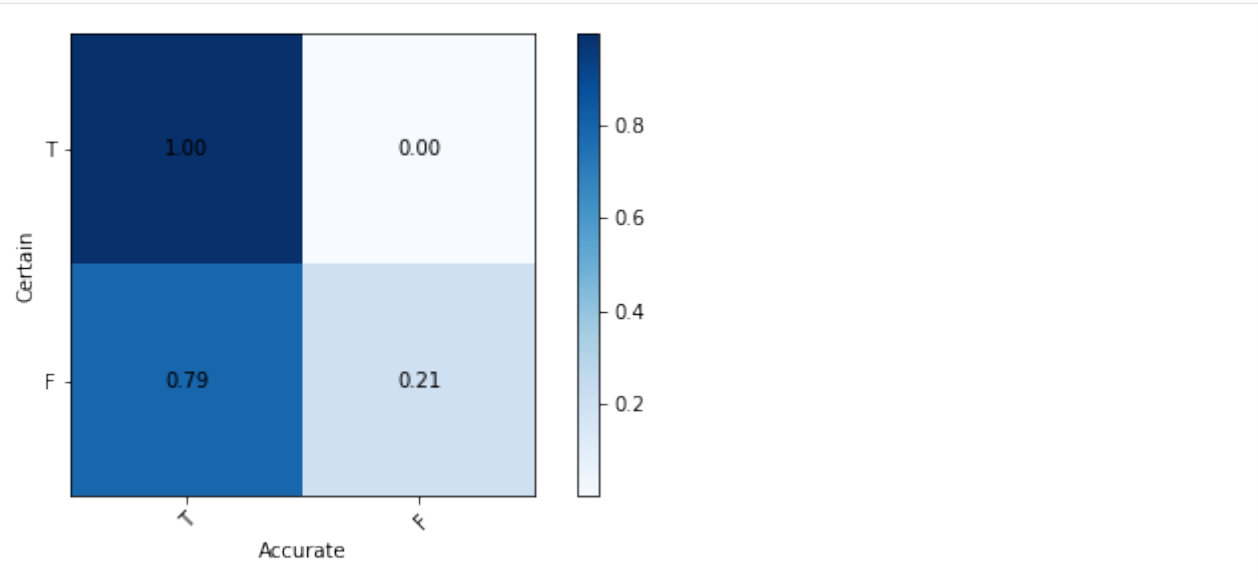
<Figure size 432x288 with 0 Axes>



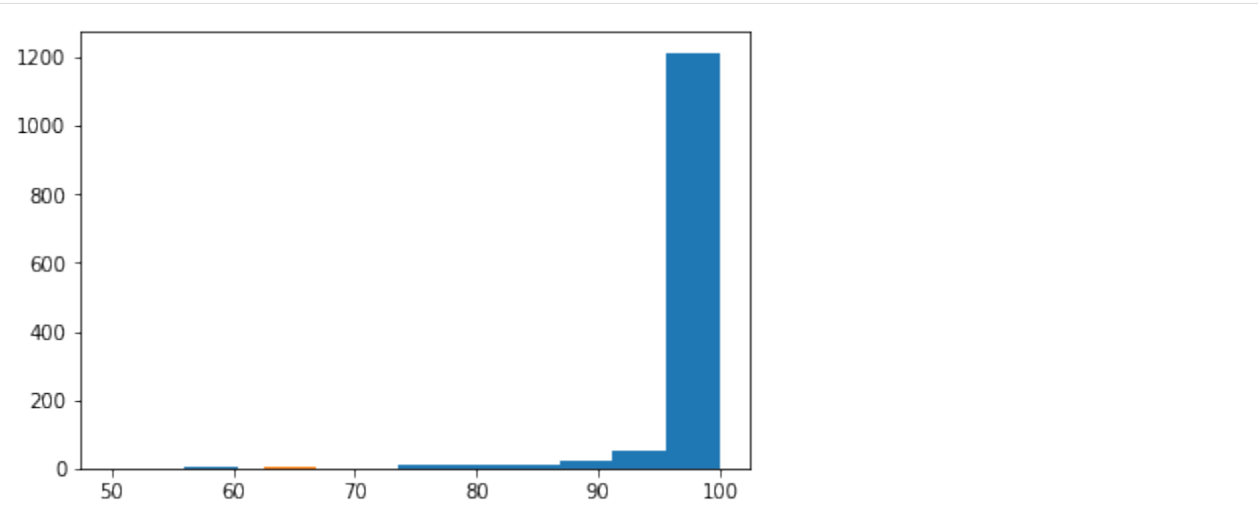
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



```
[ 0 1 2 ... 2202 2203 2204] [1323 1324 1325 1326 1327 1328 1329 1330 1331_
↪1332 1333 1334 1335 1336
1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364
1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378
1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406
1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420
1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434
1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448
1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462
1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476
1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490
1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504
1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518
1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532
1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546
```

(continues on next page)

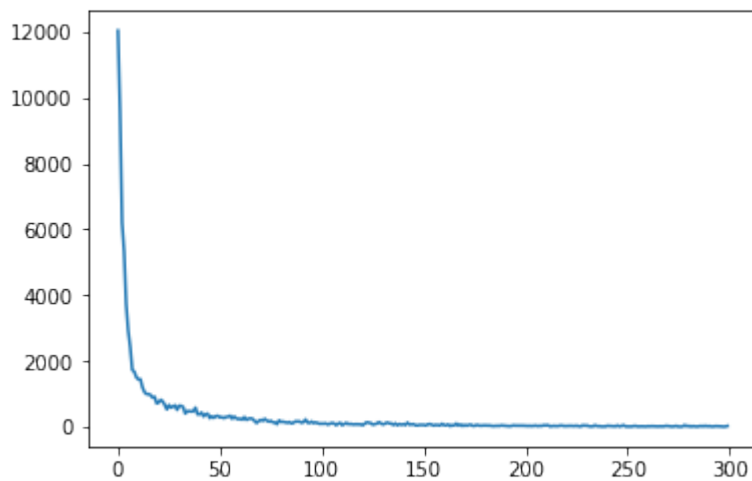
(continued from previous page)

```

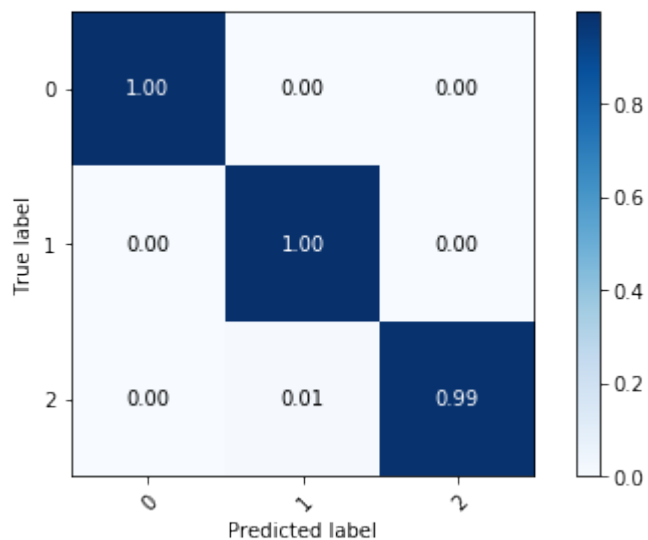
1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560
1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574
1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588
1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602
1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616
1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630
1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644
1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658
1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672
1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686
1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700
1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714
1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728
1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742
1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756
1757 1758 1759 1760 1761 1762 1763]

```

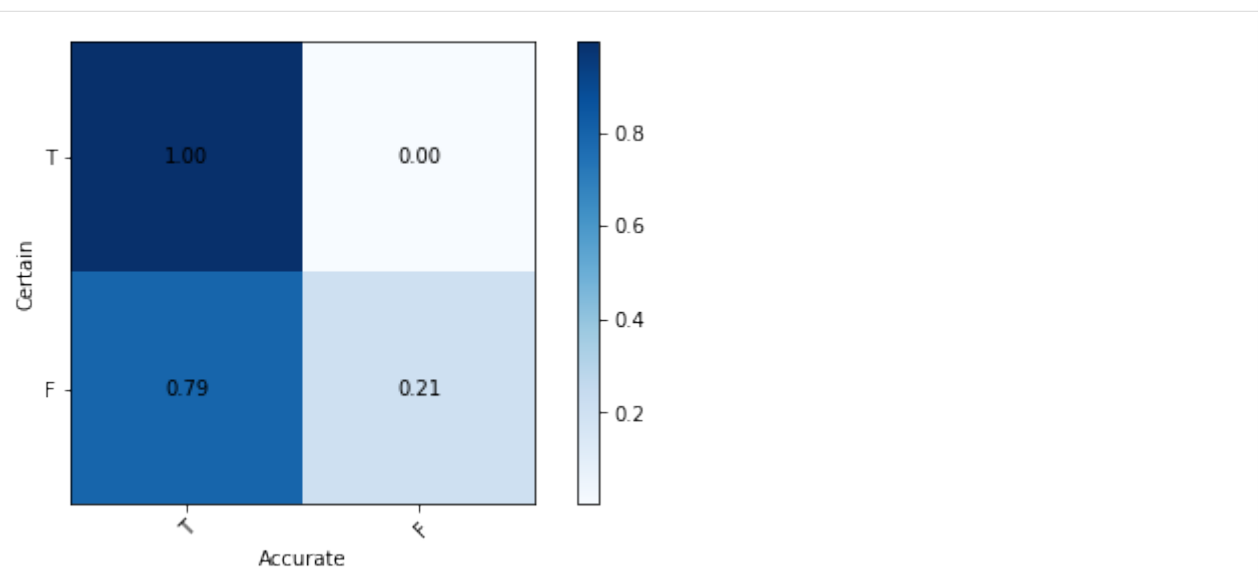
<Figure size 432x288 with 0 Axes>



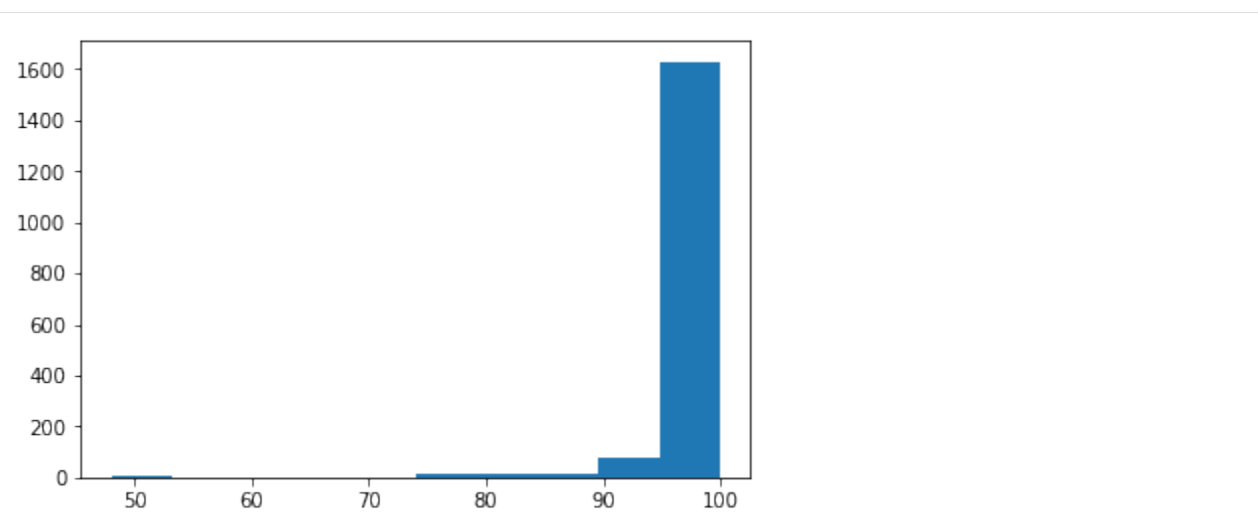
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



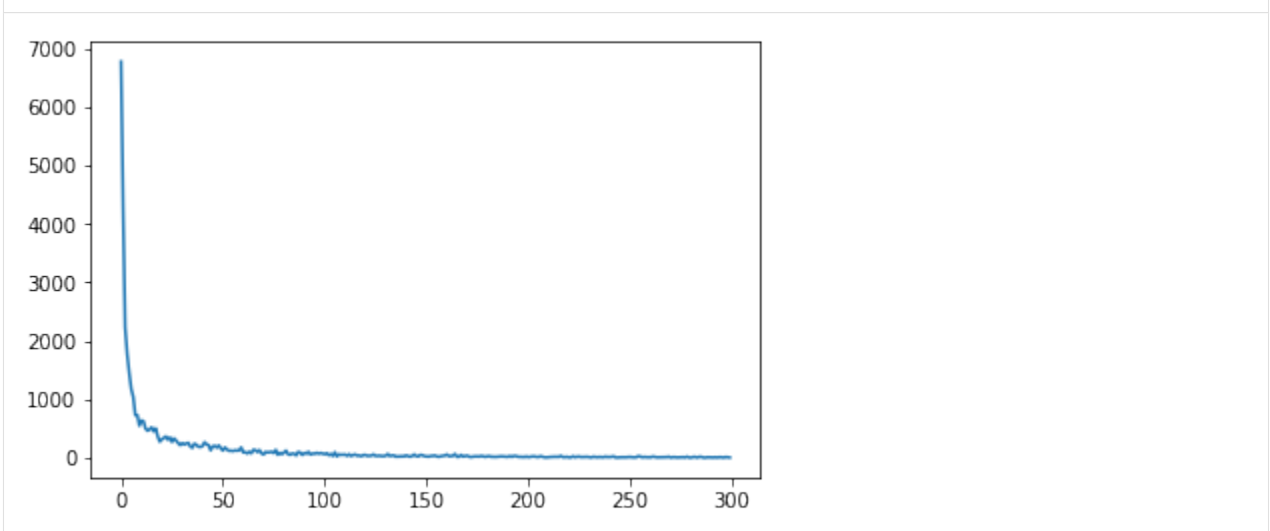
```
[ 0 1 2 ... 1761 1762 1763] [1764 1765 1766 1767 1768 1769 1770 1771 1772_
↪1773 1774 1775 1776 1777
1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791
1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805
1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819
1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833
1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847
1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861
1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875
1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889
1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903
1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917
1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931
1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945
1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959
1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973
1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987
```

(continues on next page)

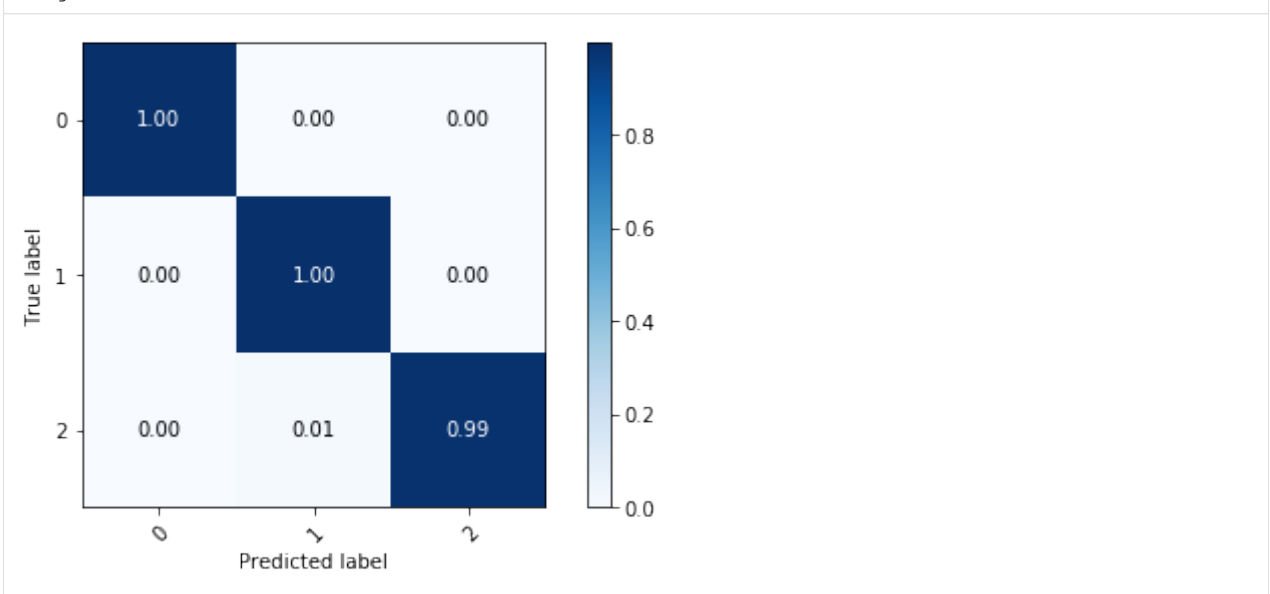
(continued from previous page)

1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029
2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043
2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057
2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071
2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085
2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099
2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113
2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141
2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155
2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169
2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183
2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197
2198	2199	2200	2201	2202	2203	2204							

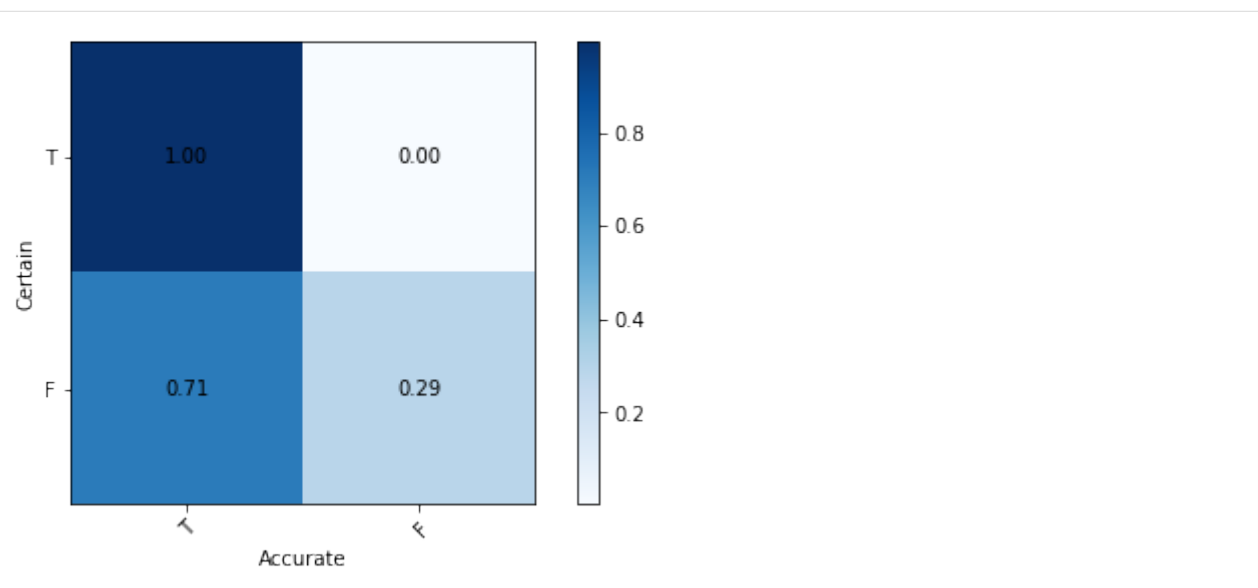
<Figure size 432x288 with 0 Axes>



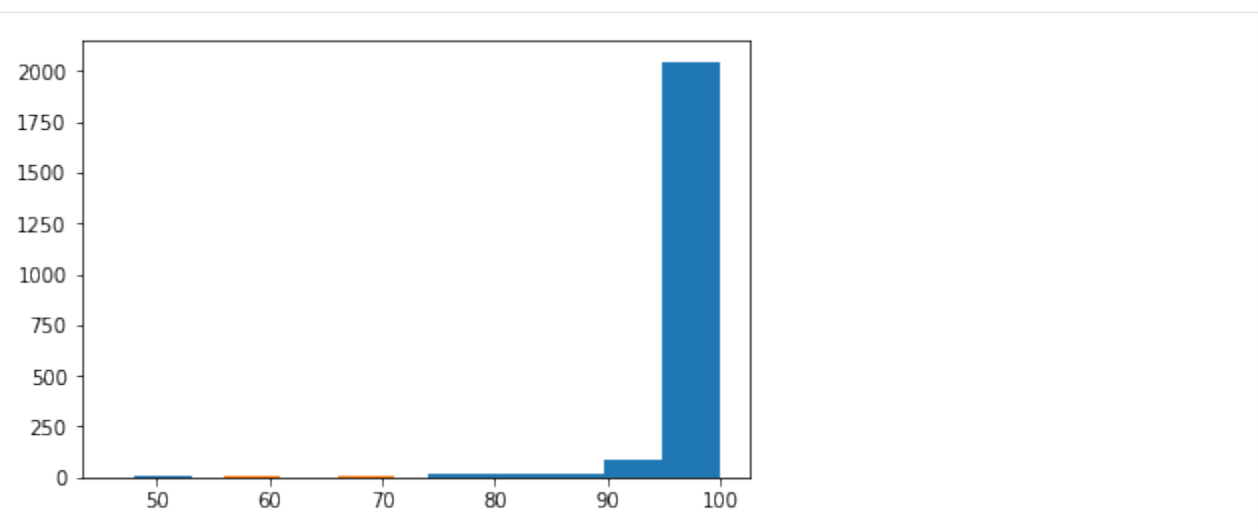
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



Target output: 0
F1 SCORE: 0.9945351473922903 +- 0.0004707605326990252 [0.9931972789115646, 0.9954648526077098, 0.9954648526077098, 0.9948979591836735, 0.9936507936507937]
P(Acc | Certain): 0.9980861983060005 +- 0.00024045557469217787 [0.997716894977169, 0.9988465974625144, 0.9984662576687117, 0.9977011494252873, 0.9977000919963201]
P(Acc | Uncertain): 0.6848302207130731 +- 0.08938562630975022 [0.3333333333333333, 0.8, 0.7894736842105263, 0.7916666666666666, 0.7096774193548387]
[441 442 443 ... 2202 2203 2204] [0 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107
108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161

(continues on next page)

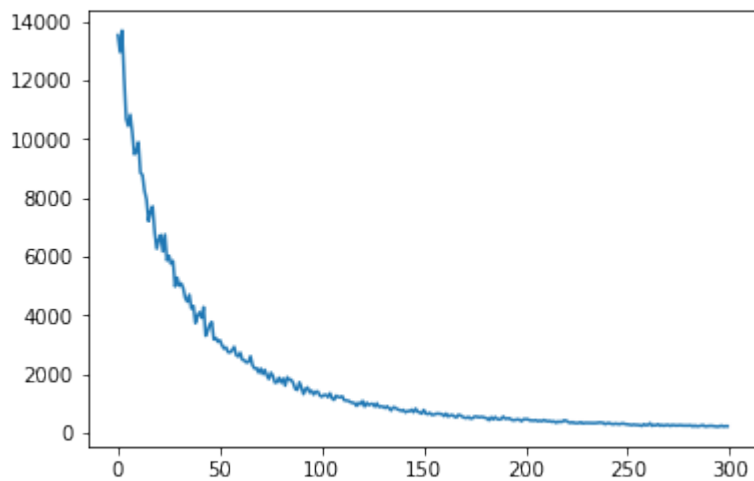
(continued from previous page)

```

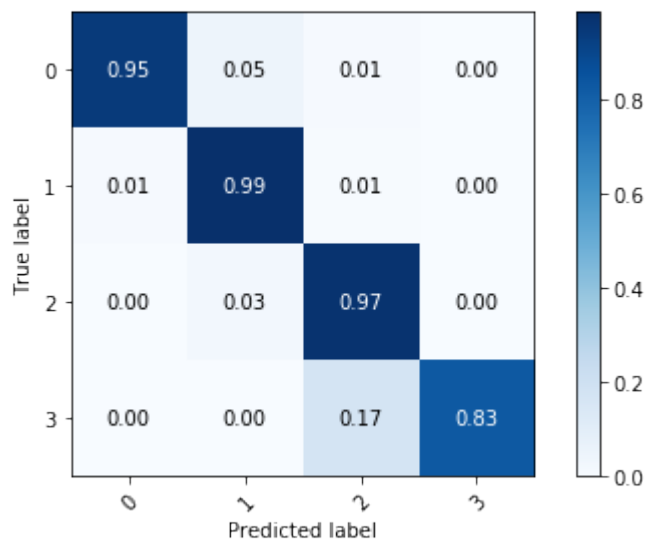
162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
432 433 434 435 436 437 438 439 440]

```

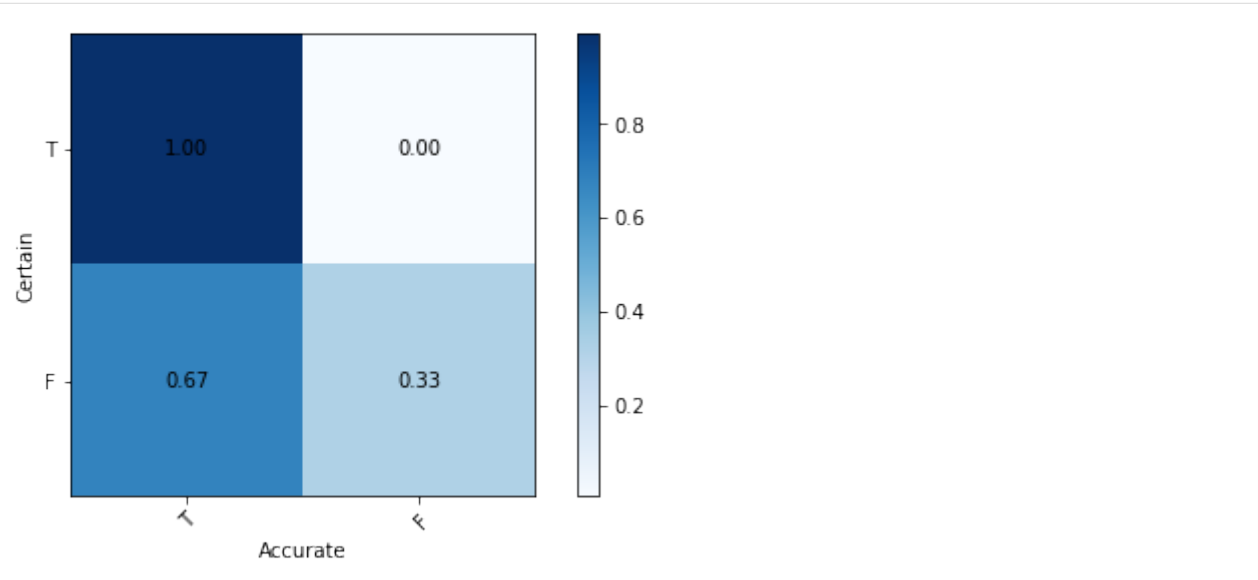
<Figure size 432x288 with 0 Axes>



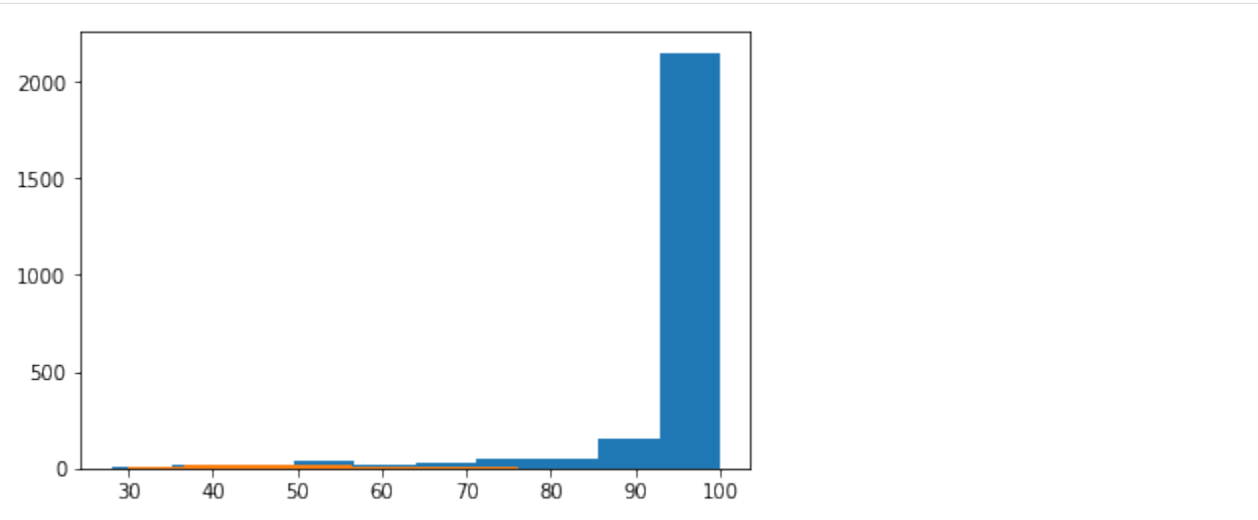
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



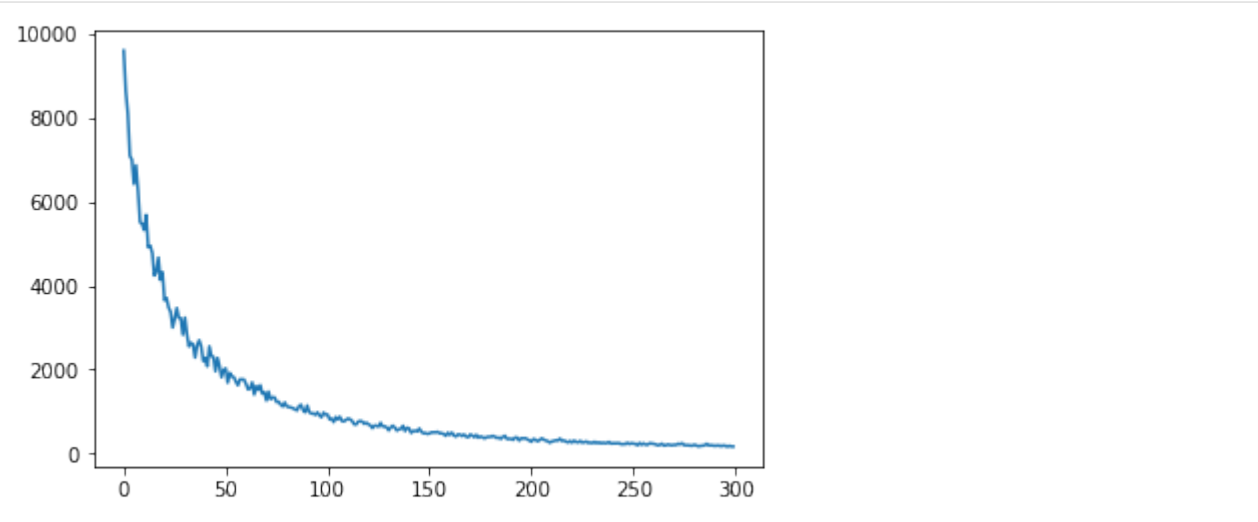
[0 1 2 ... 2202 2203 2204] [441 442 443 444 445 446 447 448 449 450 451 452...
→ 453 454 455 456 457 458
459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476
477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494
495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512
513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530
531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548
549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566
567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584
585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620
621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638
639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674
675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692
693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710
711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728

(continues on next page)

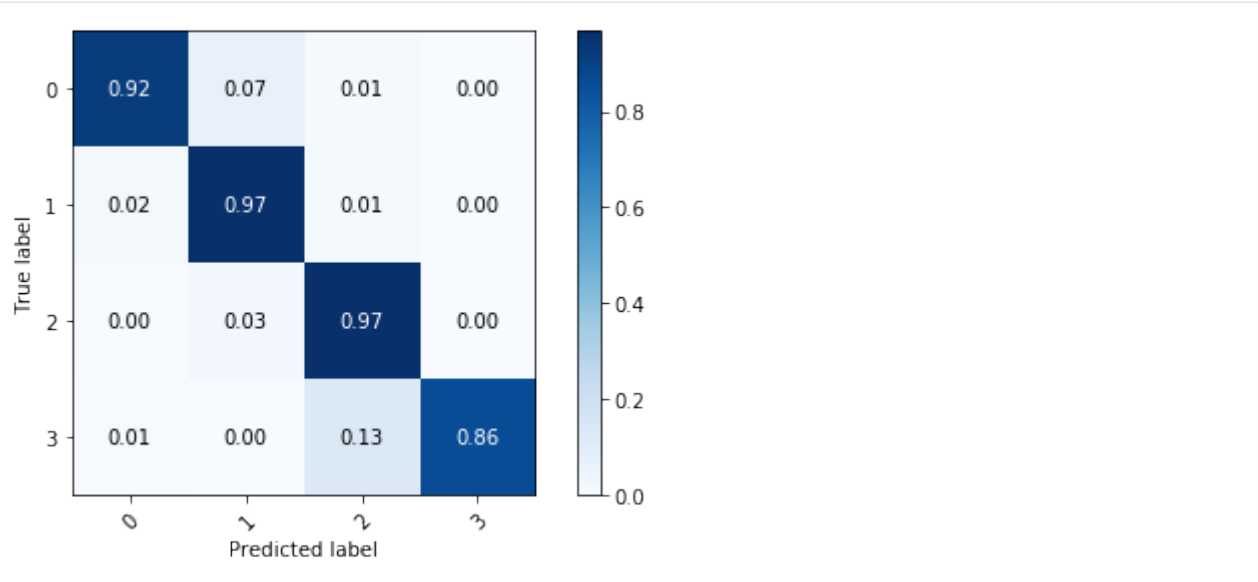
(continued from previous page)

```
729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746
747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764
765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782
783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800
801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818
819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836
837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854
855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872
873 874 875 876 877 878 879 880 881]
```

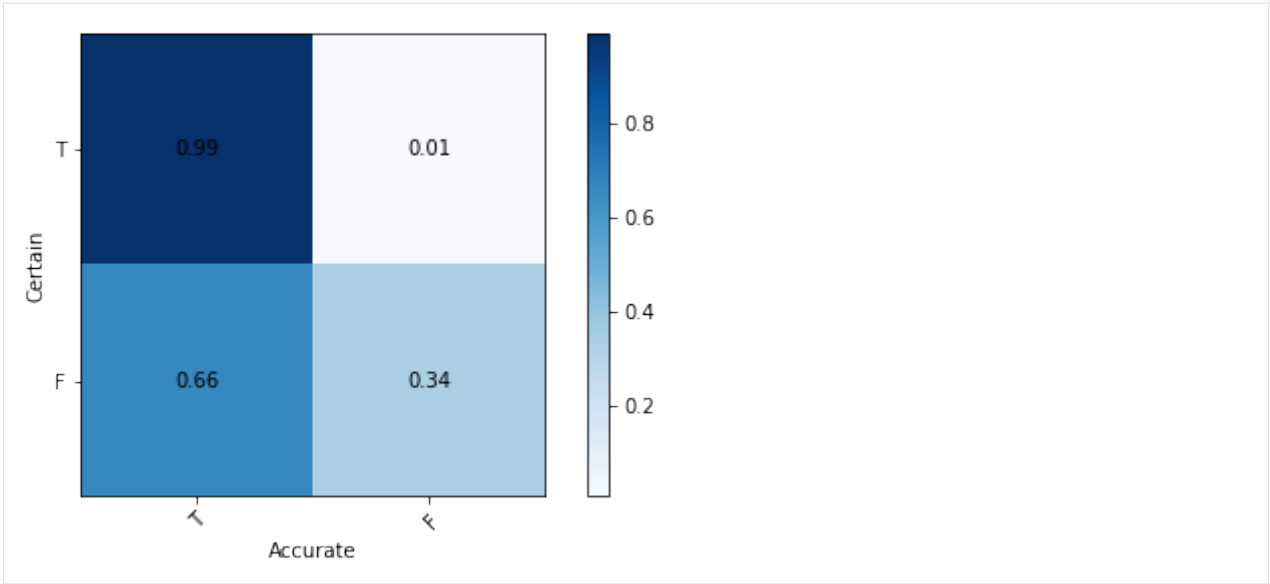
<Figure size 432x288 with 0 Axes>



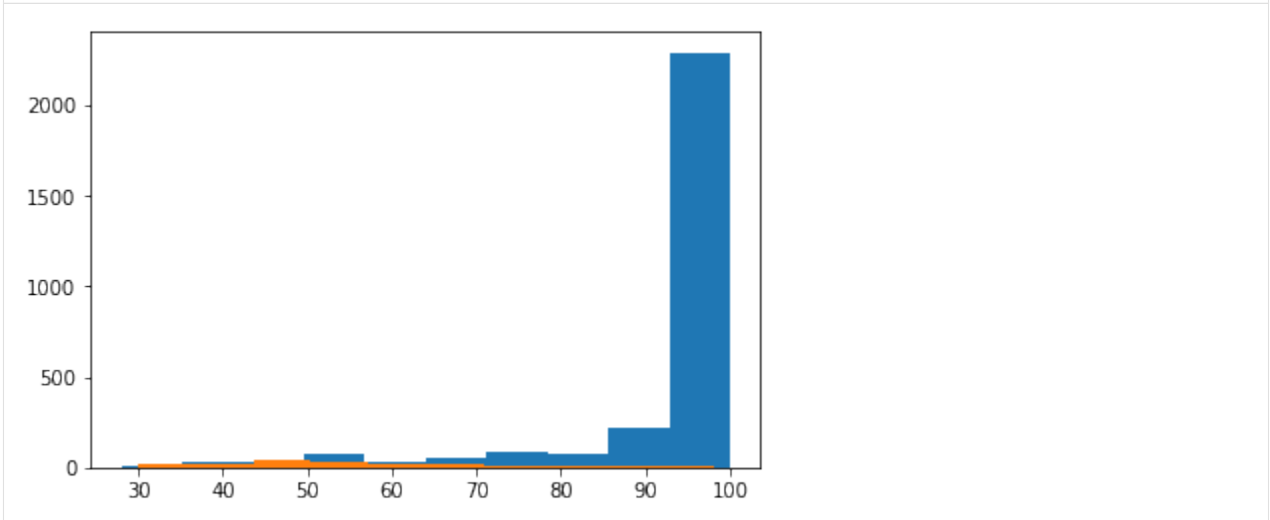
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



[0	1	2	...	2202	2203	2204]	[882	883	884	885	886	887	888	889	890	...
↪	891	892	893	894	895													
	896	897	898	899	900	901	902	903	904	905	906	907	908	909				
	910	911	912	913	914	915	916	917	918	919	920	921	922	923				
	924	925	926	927	928	929	930	931	932	933	934	935	936	937				
	938	939	940	941	942	943	944	945	946	947	948	949	950	951				
	952	953	954	955	956	957	958	959	960	961	962	963	964	965				
	966	967	968	969	970	971	972	973	974	975	976	977	978	979				
	980	981	982	983	984	985	986	987	988	989	990	991	992	993				
	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007				
	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021				
	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035				
	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049				
	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063				
	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077				
	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091				
	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105				
	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119				

(continues on next page)

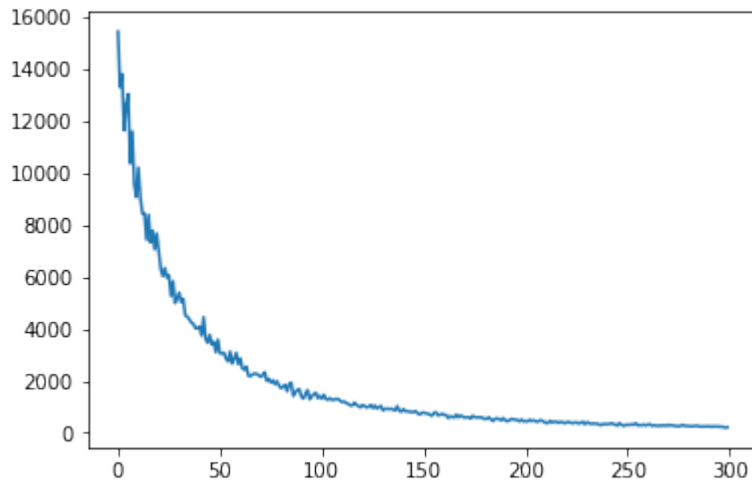
(continued from previous page)

```

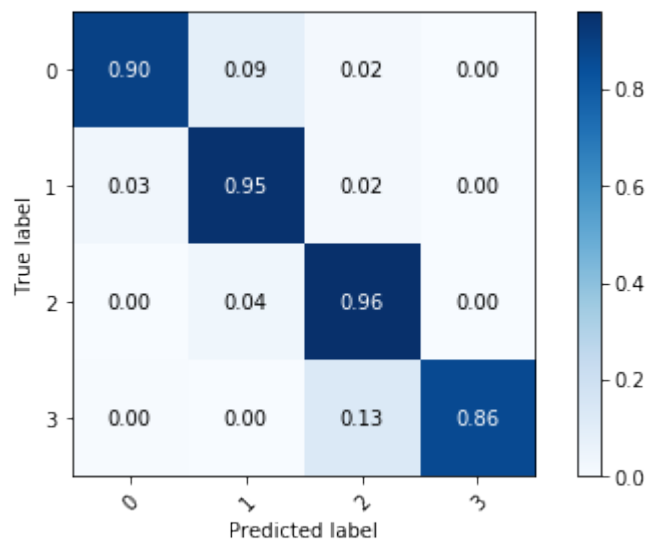
1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133
1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147
1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161
1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175
1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189
1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217
1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231
1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245
1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259
1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273
1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287
1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301
1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315
1316 1317 1318 1319 1320 1321 1322]

```

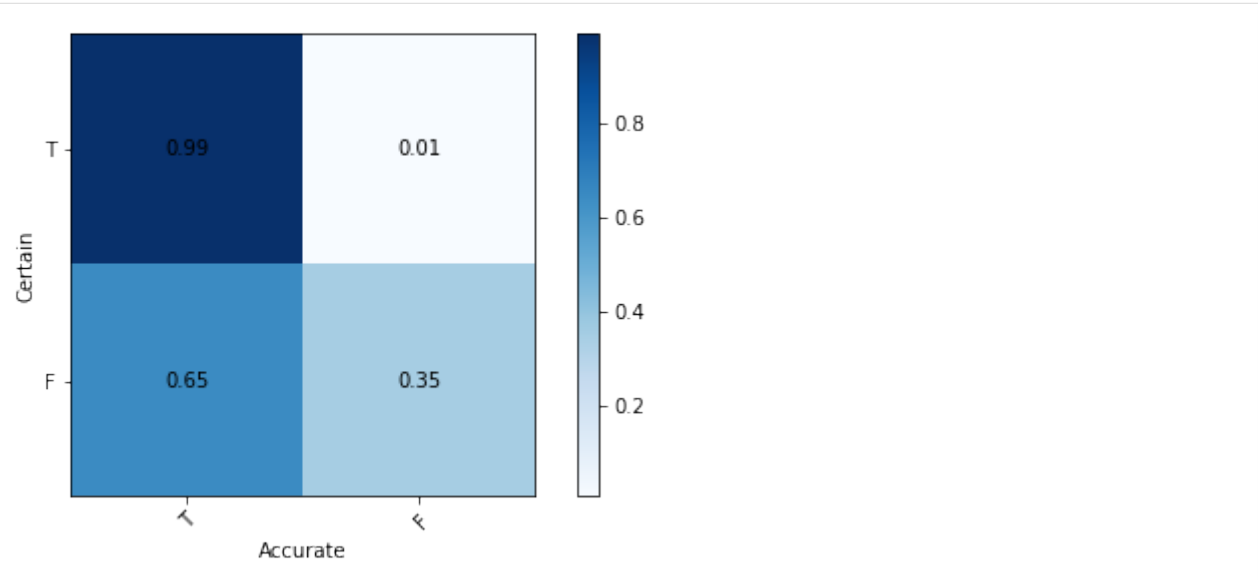
<Figure size 432x288 with 0 Axes>



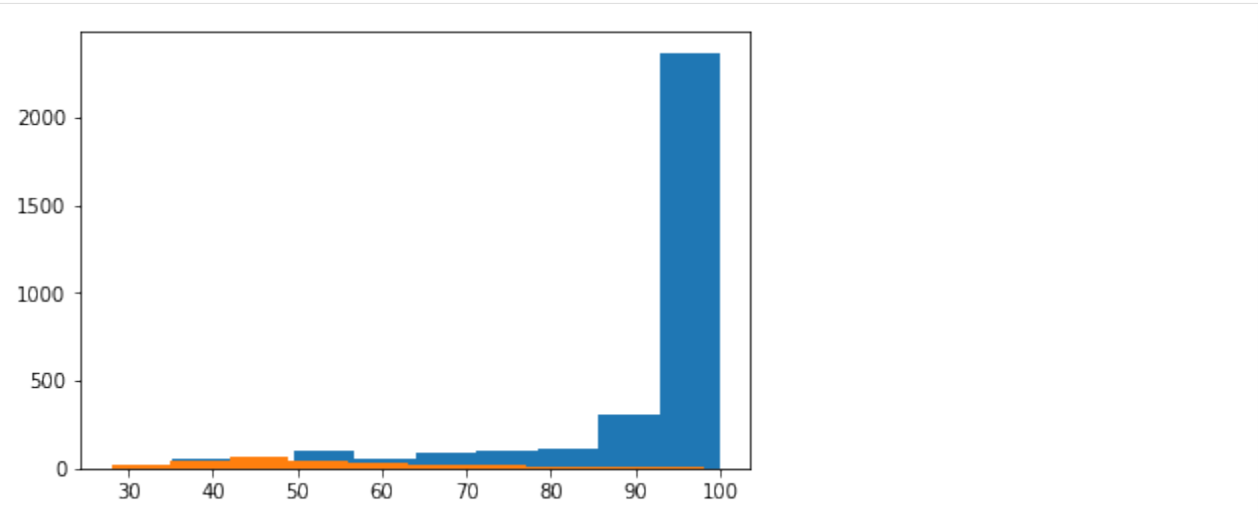
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



```
[ 0 1 2 ... 2202 2203 2204] [1323 1324 1325 1326 1327 1328 1329 1330 1331_
↪1332 1333 1334 1335 1336
1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364
1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378
1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406
1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420
1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434
1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448
1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462
1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476
1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490
1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504
1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518
1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532
1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546
```

(continues on next page)

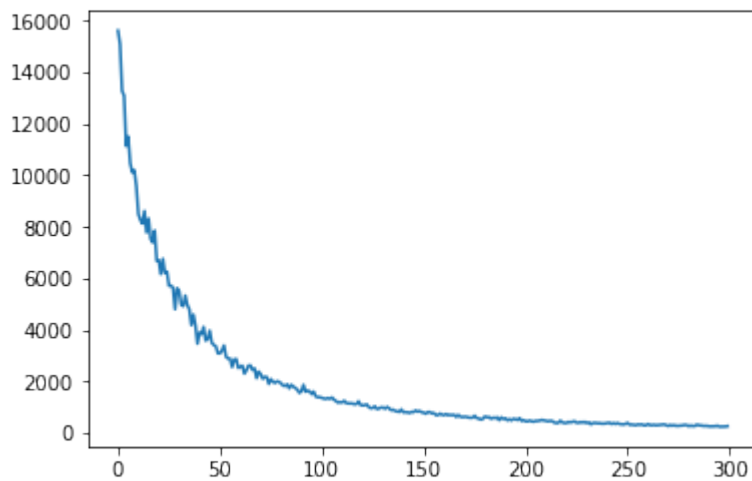
(continued from previous page)

```

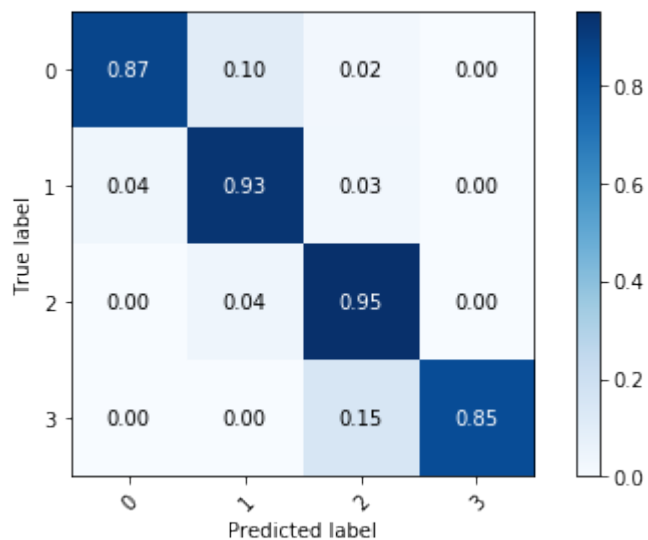
1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560
1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574
1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588
1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602
1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616
1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630
1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644
1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658
1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672
1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686
1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700
1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714
1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728
1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742
1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756
1757 1758 1759 1760 1761 1762 1763]

```

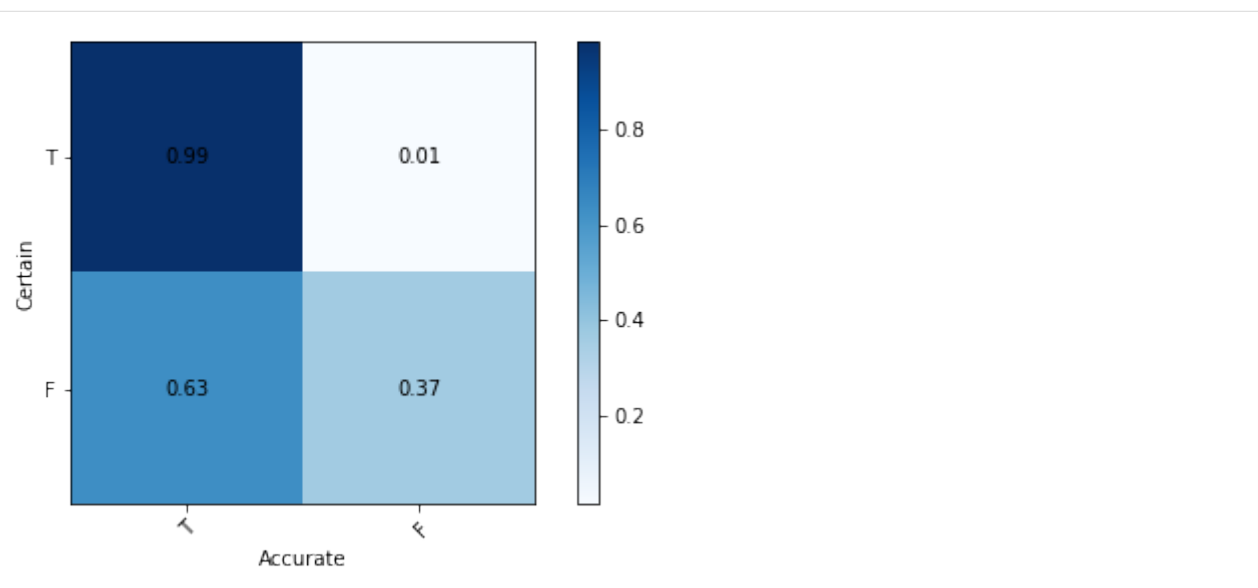
<Figure size 432x288 with 0 Axes>



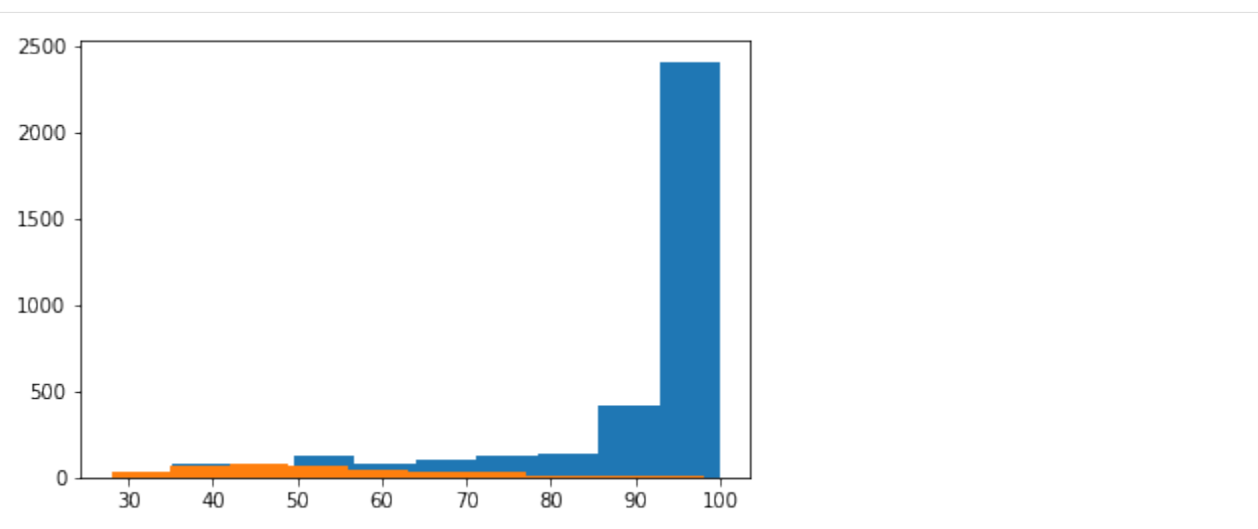
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



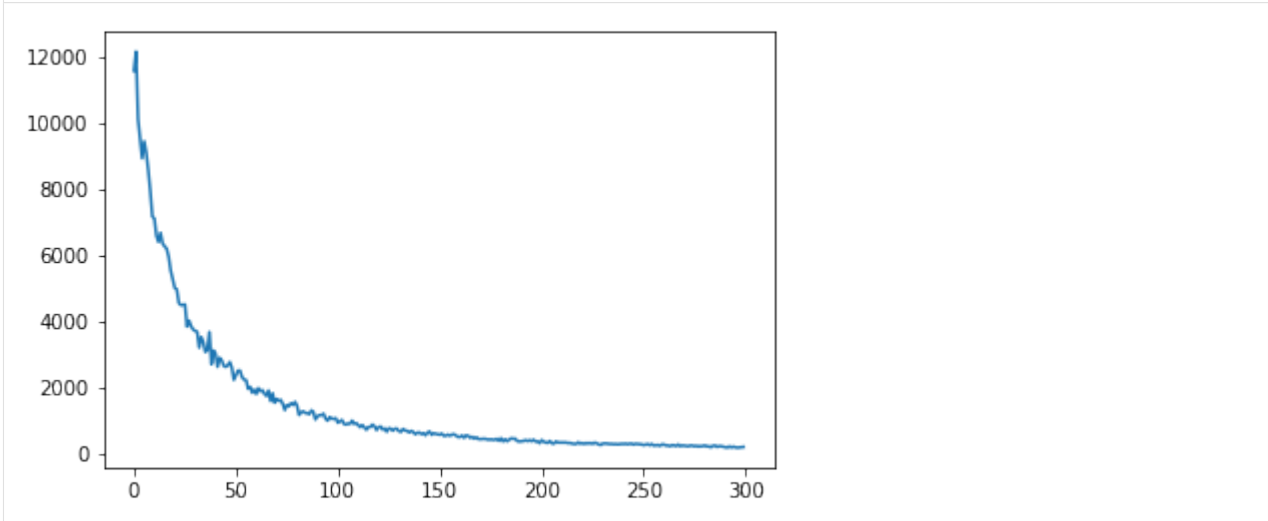
```
[ 0 1 2 ... 1761 1762 1763] [1764 1765 1766 1767 1768 1769 1770 1771 1772_
↪1773 1774 1775 1776 1777
1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791
1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805
1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819
1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833
1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847
1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861
1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875
1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889
1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903
1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917
1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931
1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945
1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959
1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973
1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987
```

(continues on next page)

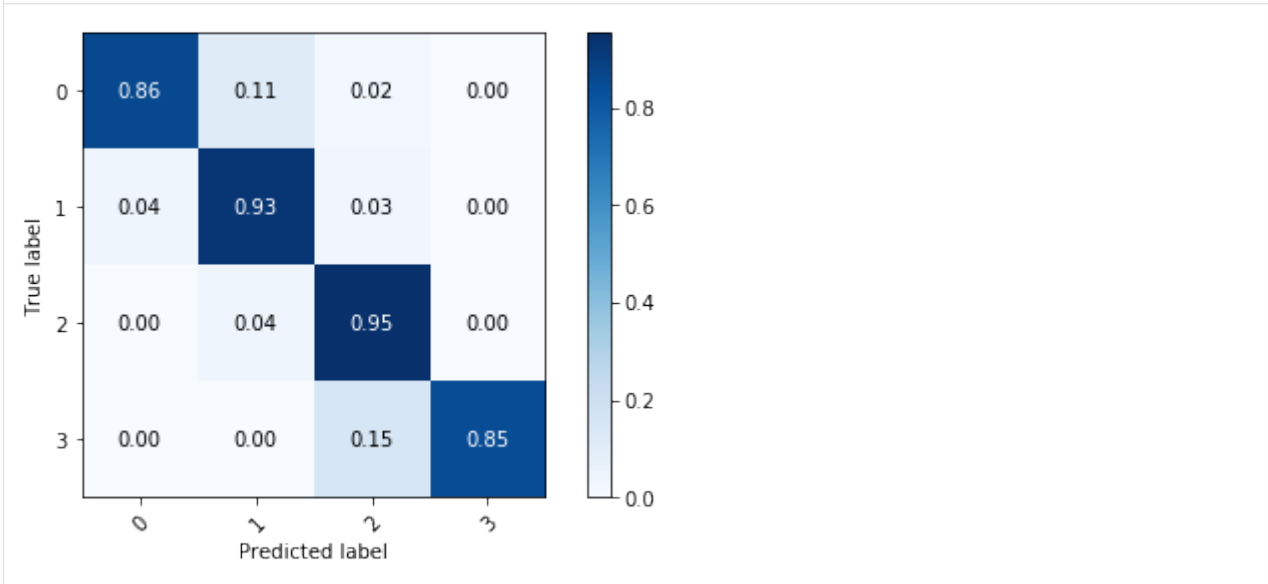
(continued from previous page)

1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029
2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043
2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057
2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071
2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085
2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099
2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113
2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141
2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155
2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169
2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183
2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197
2198	2199	2200	2201	2202	2203	2204							

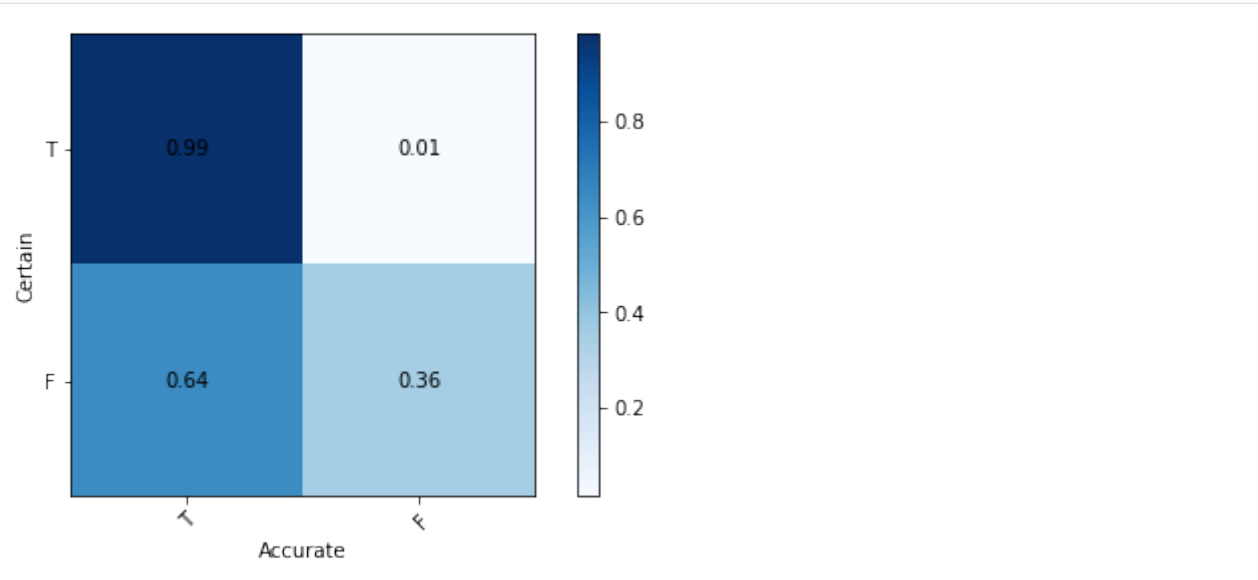
<Figure size 432x288 with 0 Axes>



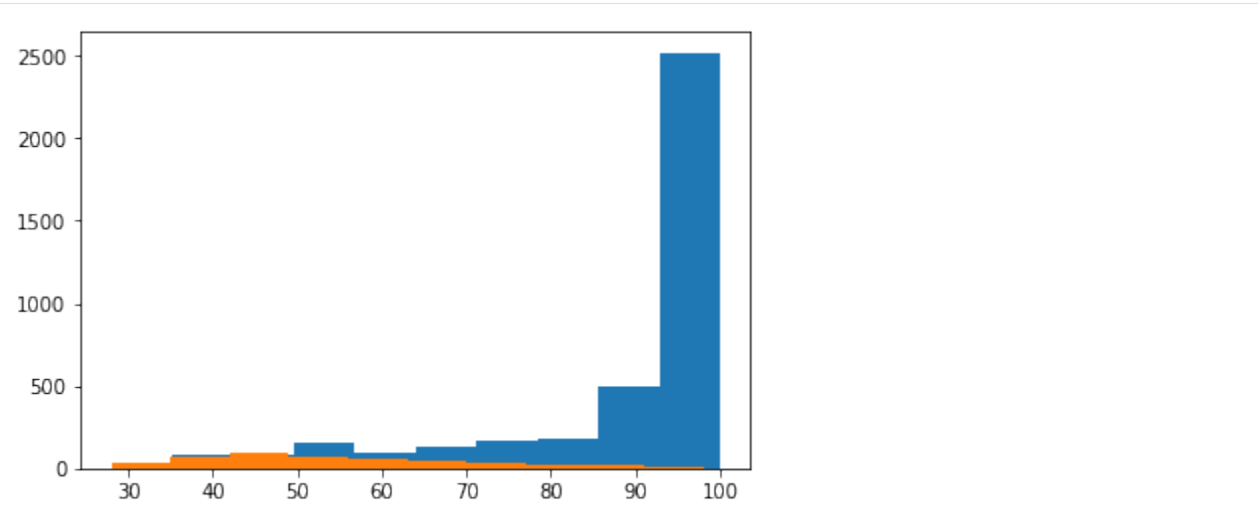
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



Target output: 1

F1 SCORE: 0.9244593816362524 +- 0.012535227234323546 [0.9622071050642479, 0.9416909620991255, 0.9220521541950113, 0.9022423784328546, 0.8941043083900226]

P(Acc | Certain): 0.9909499081179456 +- 0.001979188876666969 [0.9966144731273805, 0.9934665641813989, 0.9911253106141285, 0.9883488681757656, 0.9851943244910549]

P(Acc | Uncertain): 0.6525351521609853 +- 0.0074524186694632615 [0.6749116607773852, 0.6639175257731958, 0.6483825597749648, 0.6341968911917099, 0.6412671232876712]

[441 442 443 ... 2202 2203 2204] [0 1 2 3 4 5 6 7 8 9 10 11

12 13 14 15 16 17

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71

72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107

108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125

126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143

144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161

(continues on next page)

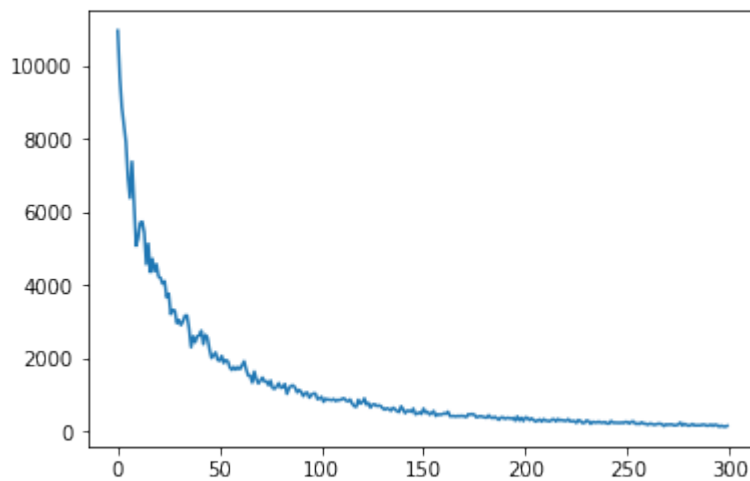
(continued from previous page)

```

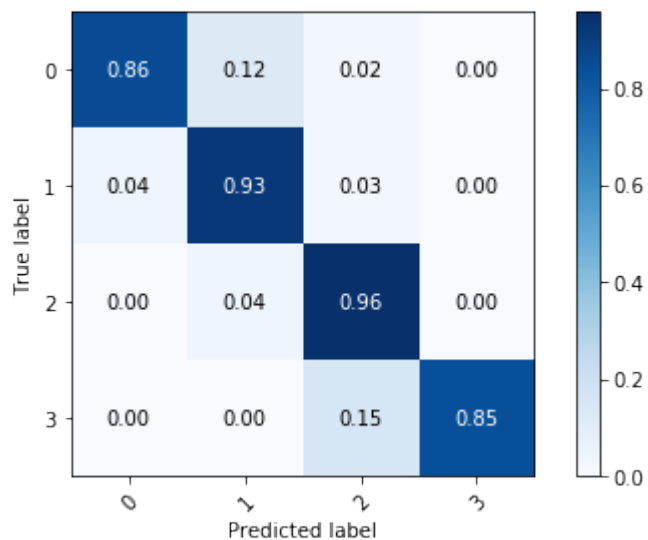
162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
432 433 434 435 436 437 438 439 440]

```

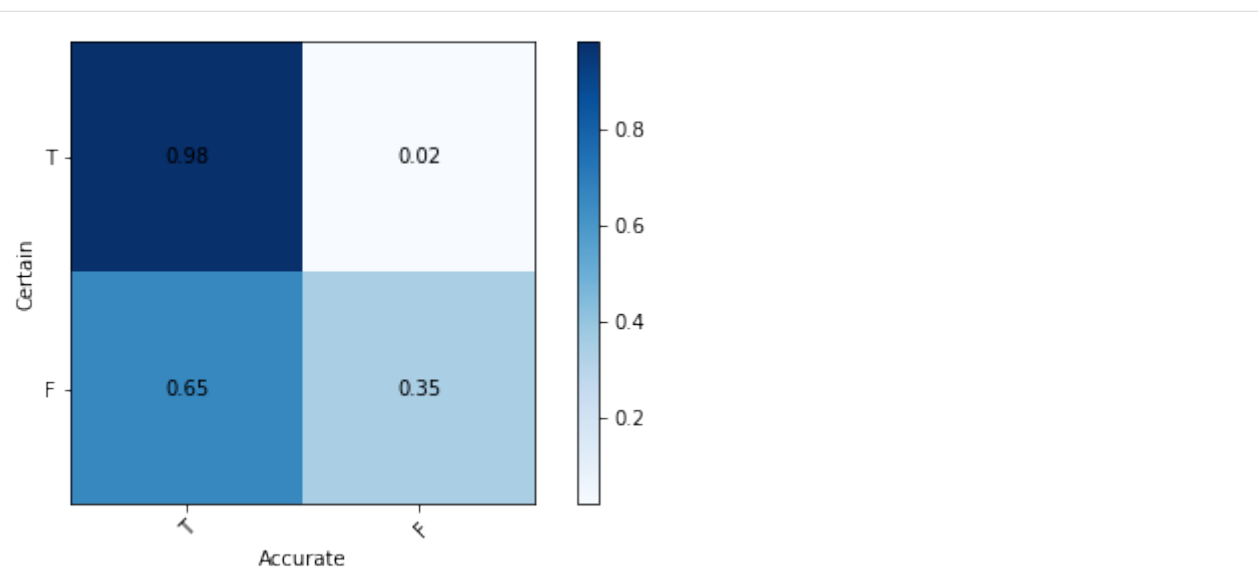
<Figure size 432x288 with 0 Axes>



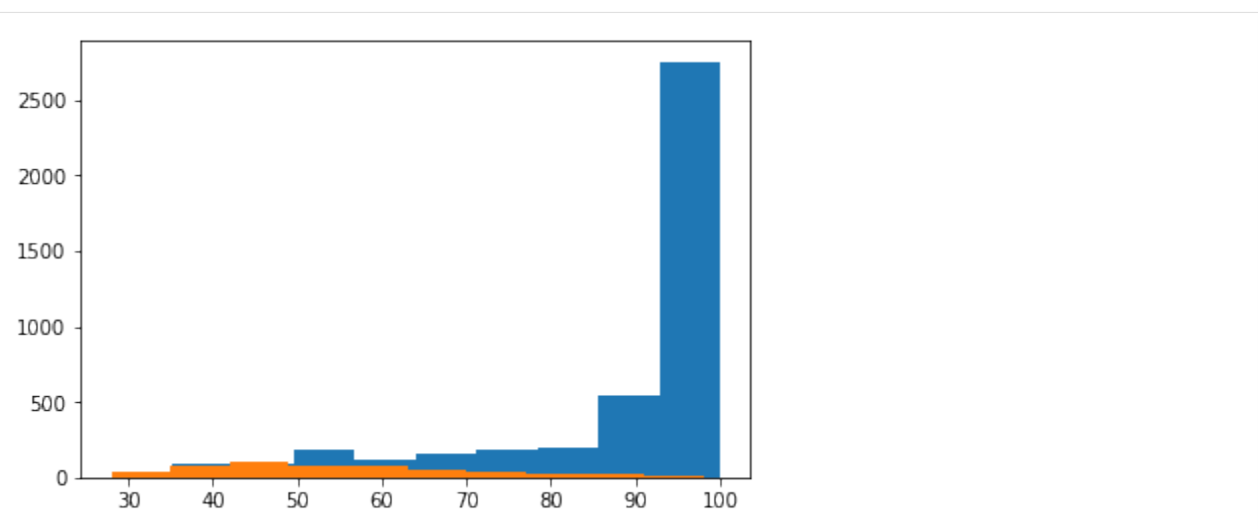
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



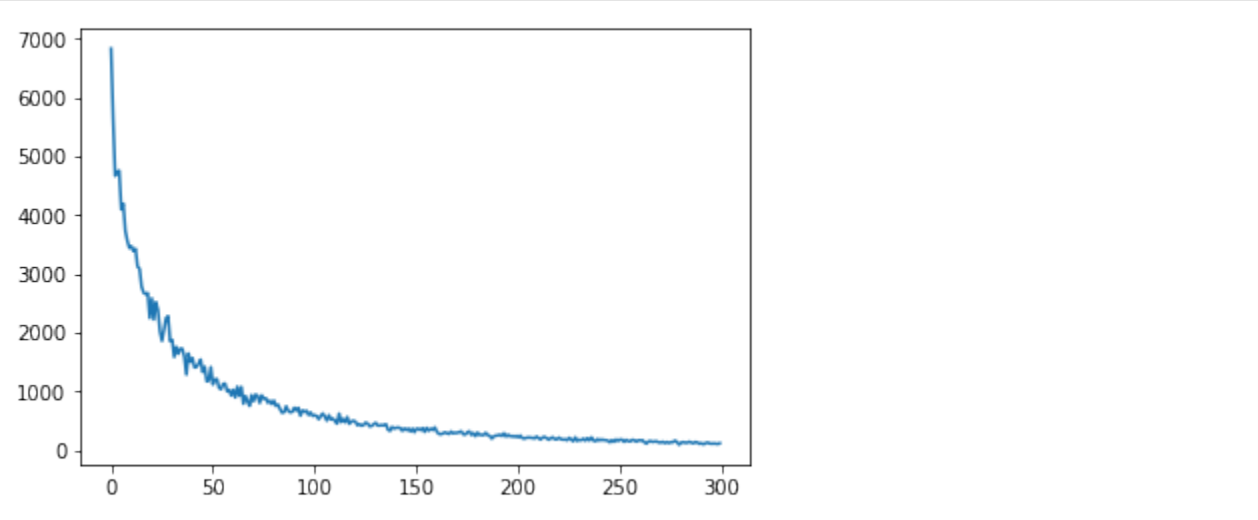
```
[ 0 1 2 ... 2202 2203 2204] [441 442 443 444 445 446 447 448 449 450 451 452]
↪453 454 455 456 457 458
459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476
477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494
495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512
513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530
531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548
549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566
567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584
585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620
621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638
639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674
675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692
693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710
711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728
```

(continues on next page)

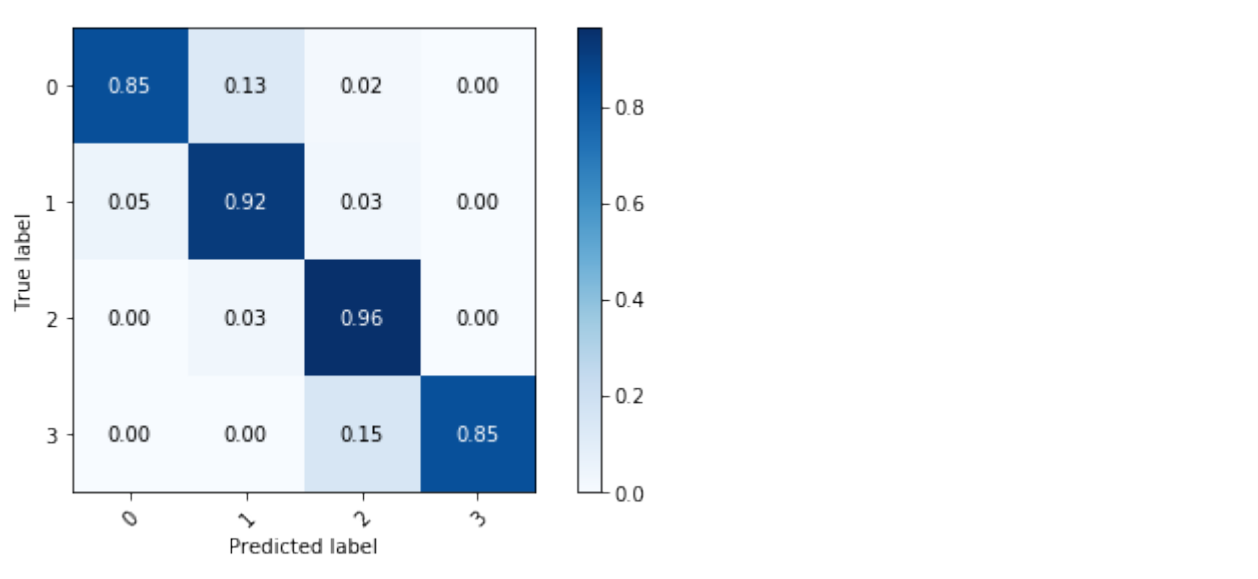
(continued from previous page)

```
729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746
747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764
765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782
783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800
801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818
819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836
837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854
855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872
873 874 875 876 877 878 879 880 881]
```

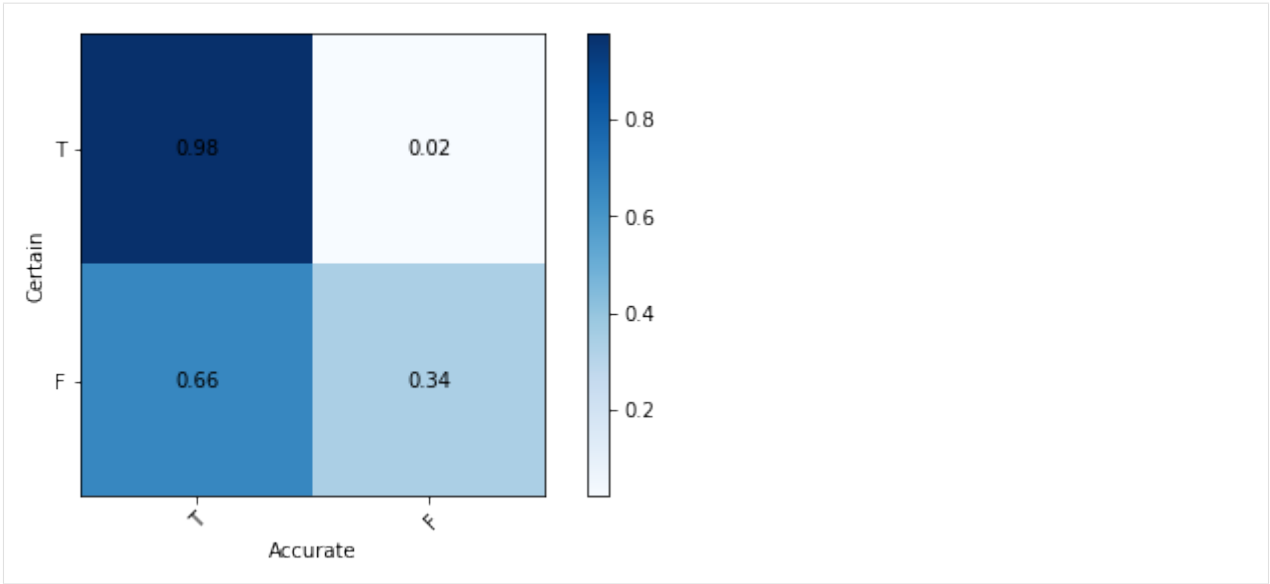
<Figure size 432x288 with 0 Axes>



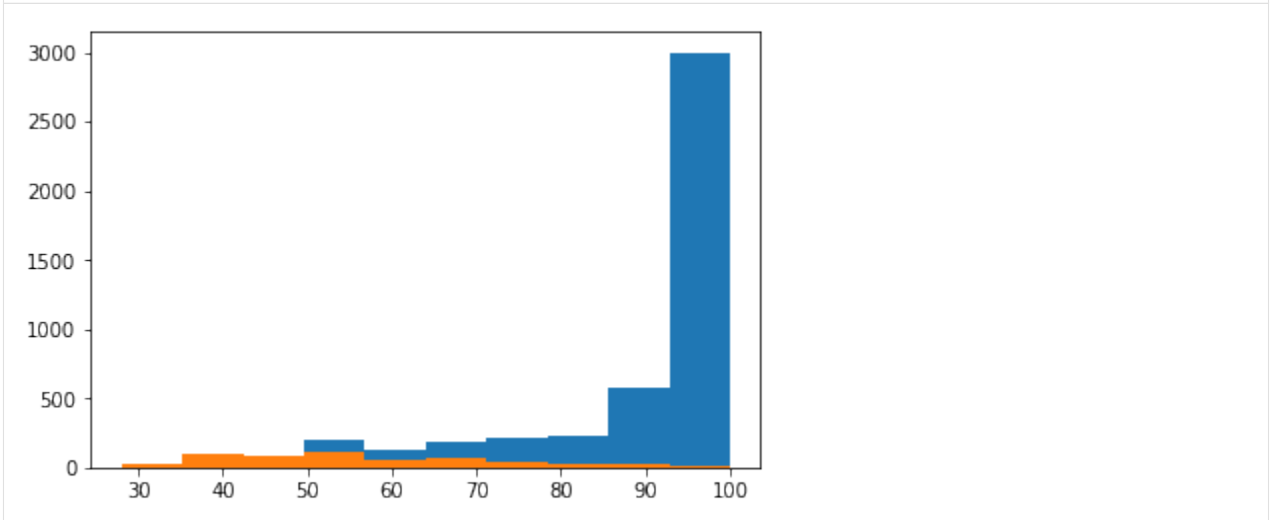
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



[0	1	2	...	2202	2203	2204]	[882	883	884	885	886	887	888	889	890	↵
↵	891	892	893	894	895													
	896	897	898	899	900	901	902	903	904	905	906	907	908	909				
	910	911	912	913	914	915	916	917	918	919	920	921	922	923				
	924	925	926	927	928	929	930	931	932	933	934	935	936	937				
	938	939	940	941	942	943	944	945	946	947	948	949	950	951				
	952	953	954	955	956	957	958	959	960	961	962	963	964	965				
	966	967	968	969	970	971	972	973	974	975	976	977	978	979				
	980	981	982	983	984	985	986	987	988	989	990	991	992	993				
	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007				
	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021				
	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035				
	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049				
	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063				
	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077				
	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091				
	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105				
	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119				

(continues on next page)

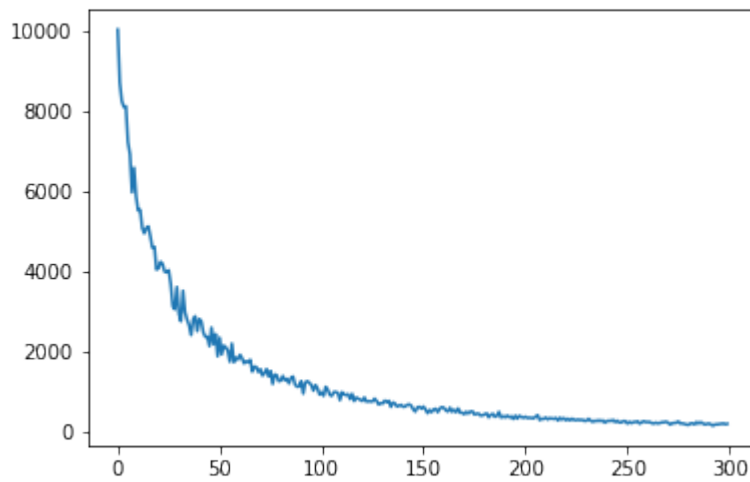
(continued from previous page)

```

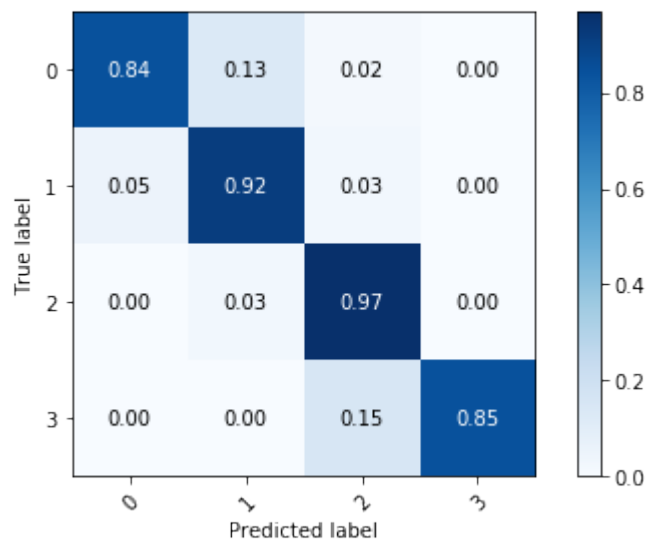
1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133
1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147
1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161
1162 1163 1164 1165 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175
1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189
1190 1191 1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203
1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216 1217
1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231
1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245
1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259
1260 1261 1262 1263 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273
1274 1275 1276 1277 1278 1279 1280 1281 1282 1283 1284 1285 1286 1287
1288 1289 1290 1291 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301
1302 1303 1304 1305 1306 1307 1308 1309 1310 1311 1312 1313 1314 1315
1316 1317 1318 1319 1320 1321 1322]

```

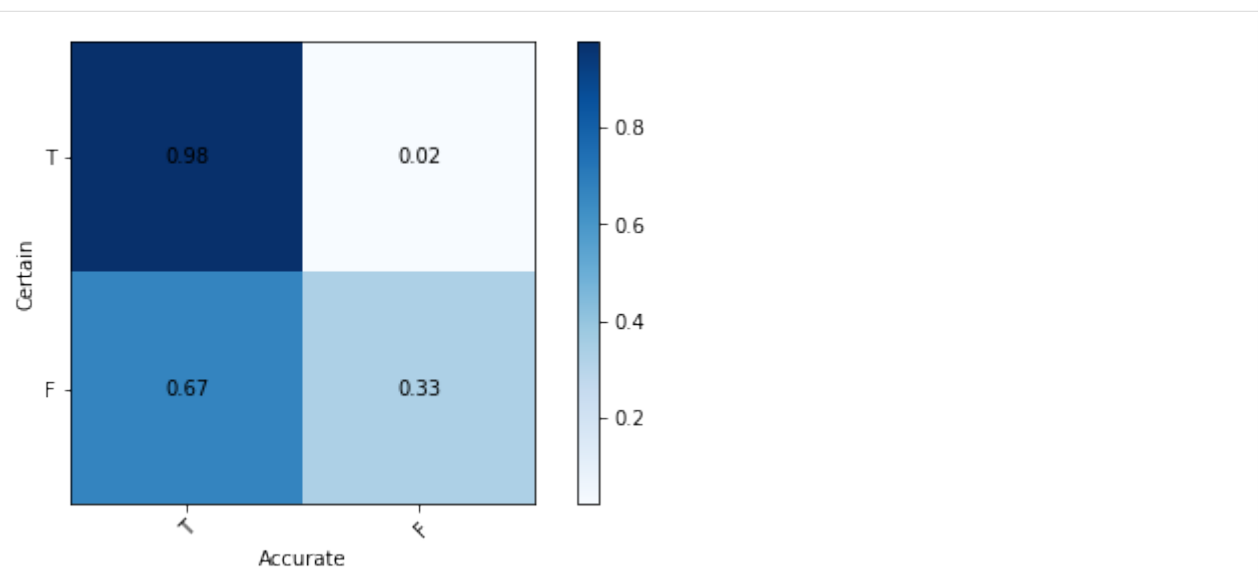
<Figure size 432x288 with 0 Axes>



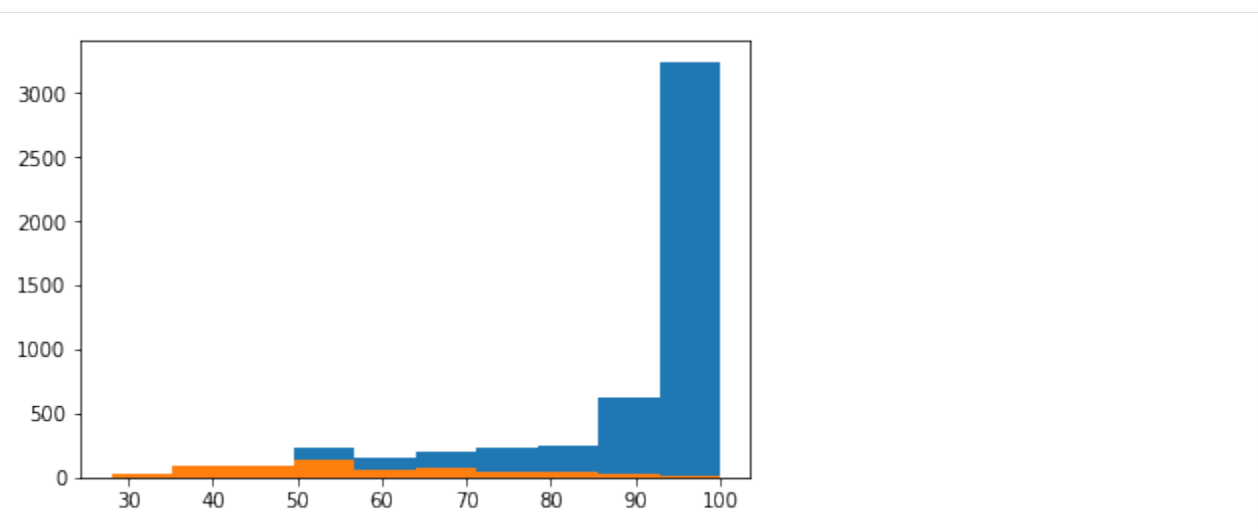
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



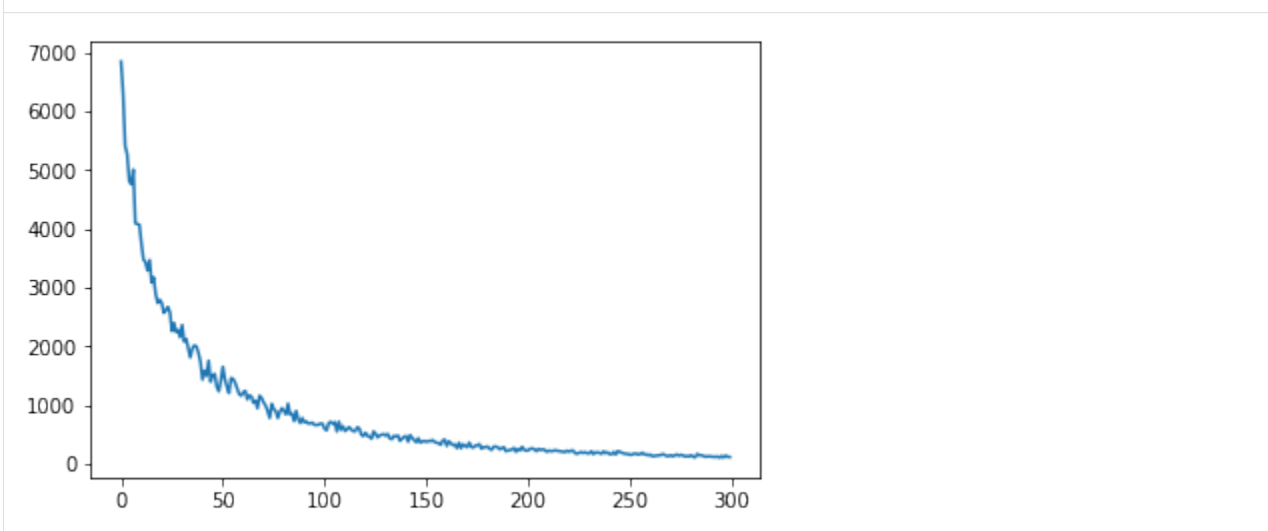
```
[ 0 1 2 ... 2202 2203 2204] [1323 1324 1325 1326 1327 1328 1329 1330 1331_
↪1332 1333 1334 1335 1336
1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364
1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378
1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406
1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420
1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434
1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448
1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462
1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476
1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490
1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504
1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518
1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532
1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546
```

(continues on next page)

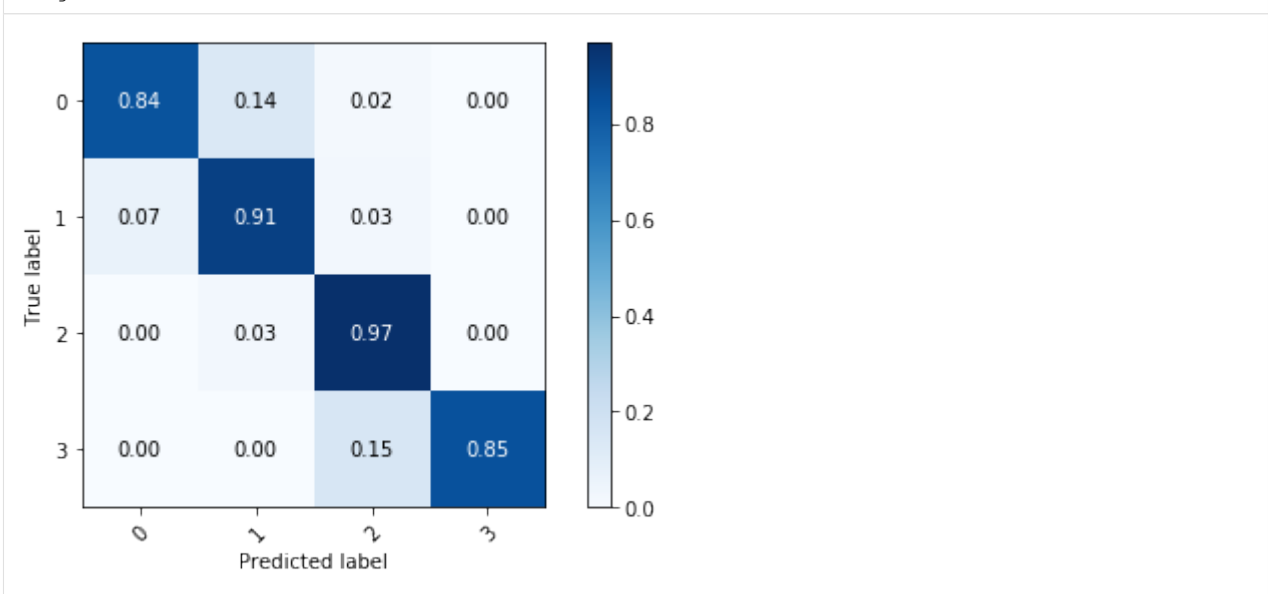
(continued from previous page)

1547	1548	1549	1550	1551	1552	1553	1554	1555	1556	1557	1558	1559	1560
1561	1562	1563	1564	1565	1566	1567	1568	1569	1570	1571	1572	1573	1574
1575	1576	1577	1578	1579	1580	1581	1582	1583	1584	1585	1586	1587	1588
1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599	1600	1601	1602
1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615	1616
1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630
1631	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644
1645	1646	1647	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658
1659	1660	1661	1662	1663	1664	1665	1666	1667	1668	1669	1670	1671	1672
1673	1674	1675	1676	1677	1678	1679	1680	1681	1682	1683	1684	1685	1686
1687	1688	1689	1690	1691	1692	1693	1694	1695	1696	1697	1698	1699	1700
1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711	1712	1713	1714
1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727	1728
1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742
1743	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756
1757	1758	1759	1760	1761	1762	1763							

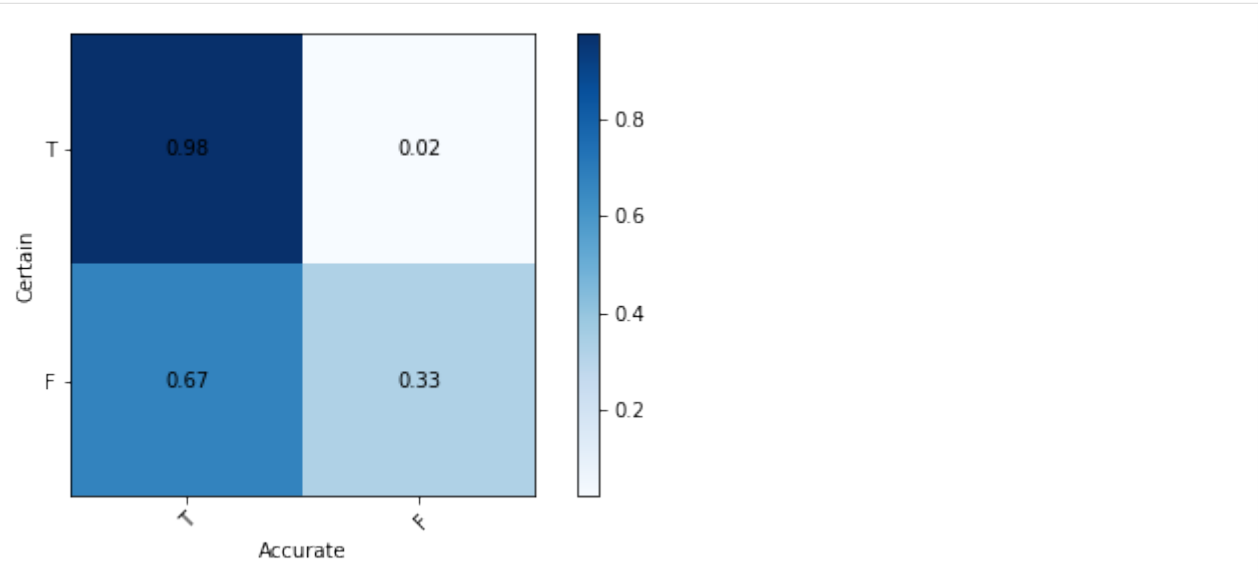
<Figure size 432x288 with 0 Axes>



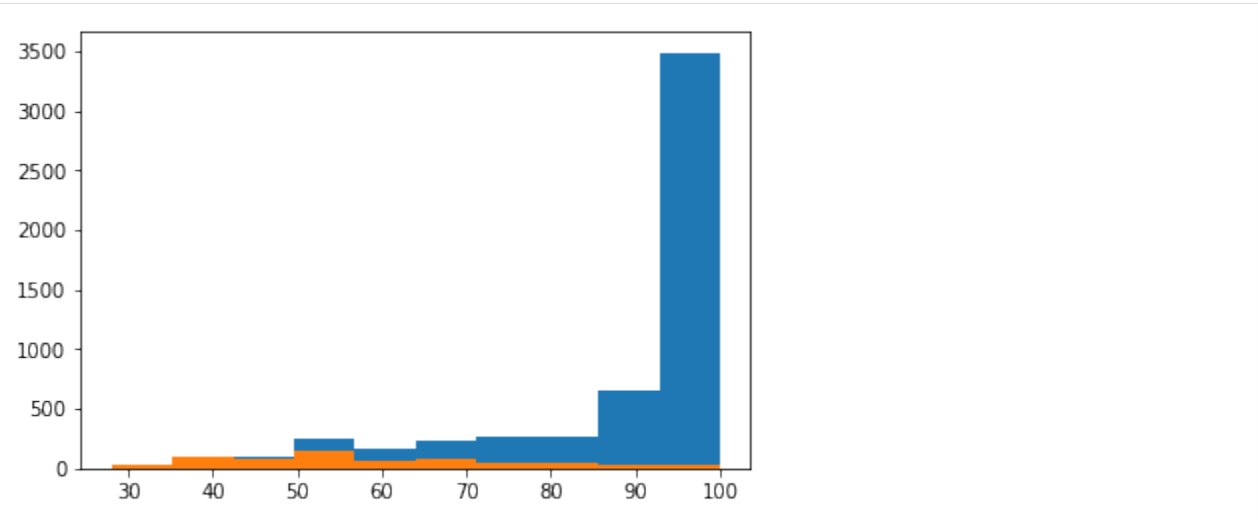
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



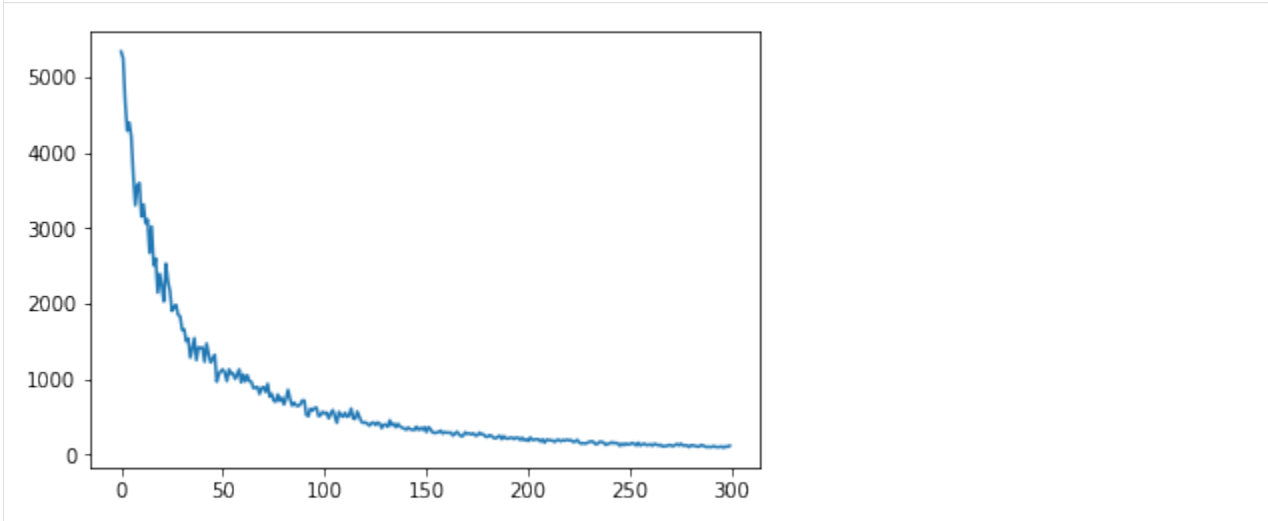
```
[ 0 1 2 ... 1761 1762 1763] [1764 1765 1766 1767 1768 1769 1770 1771 1772_
↪1773 1774 1775 1776 1777
1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791
1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805
1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819
1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833
1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847
1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861
1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875
1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889
1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903
1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917
1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931
1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945
1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959
1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973
1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987
```

(continues on next page)

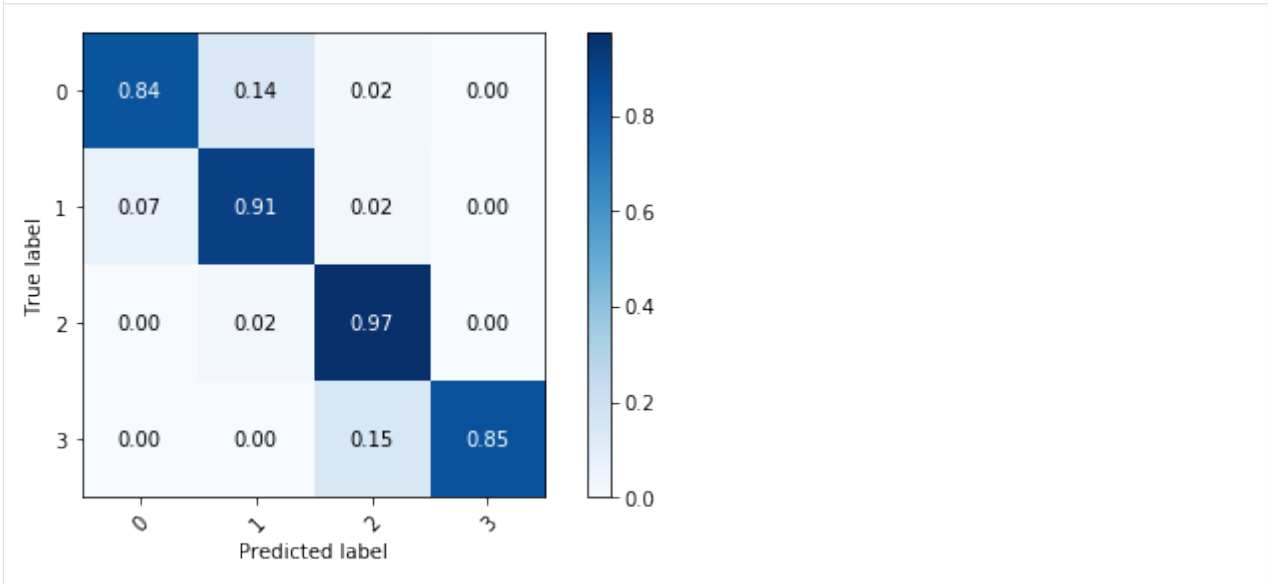
(continued from previous page)

1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029
2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043
2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057
2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071
2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085
2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099
2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113
2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141
2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155
2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169
2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183
2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197
2198	2199	2200	2201	2202	2203	2204							

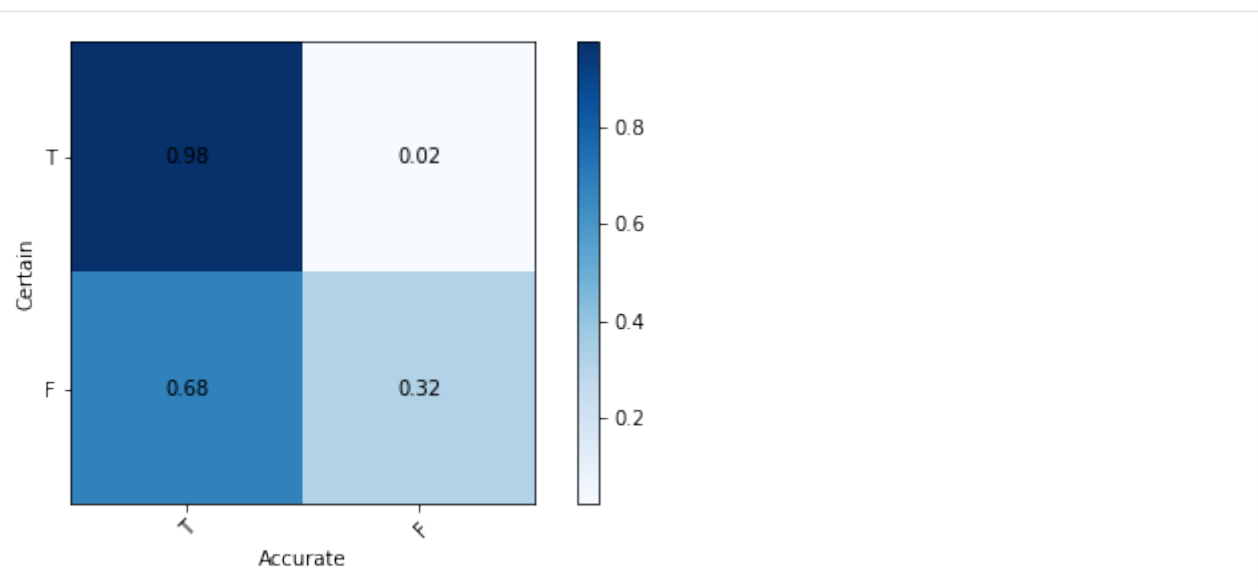
<Figure size 432x288 with 0 Axes>



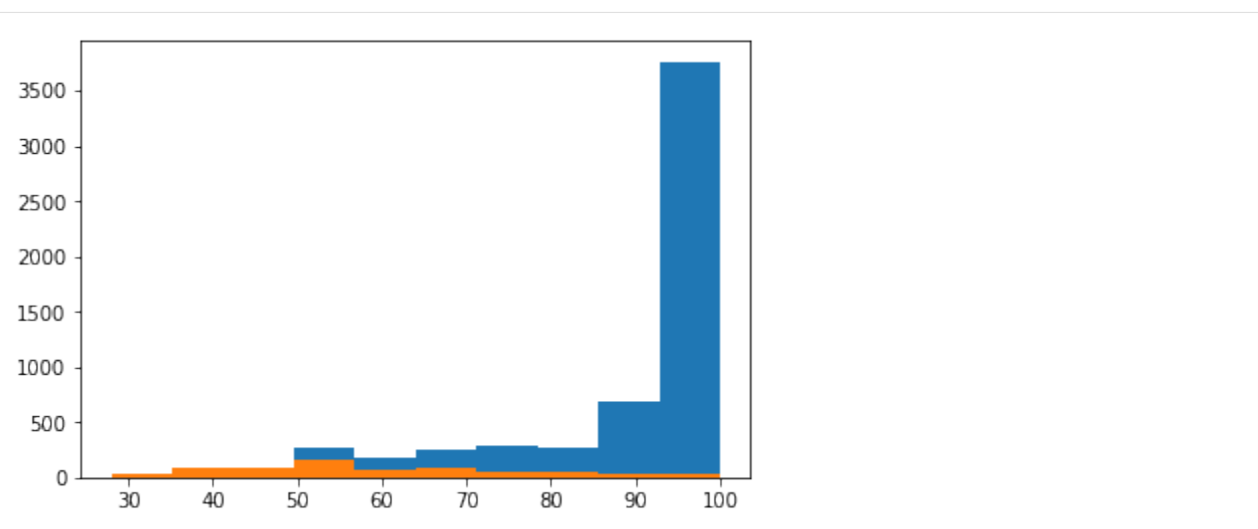
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



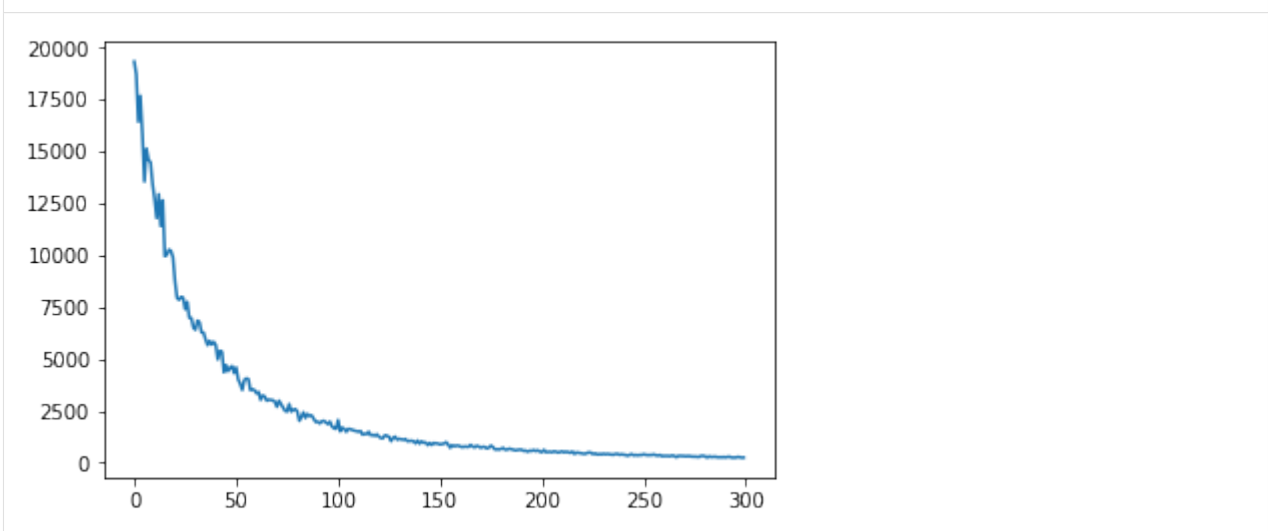
Target output: 2
F1 SCORE: 0.8956939054898239 +- 0.0004668869383694529 [0.8954854669140383, 0.8945578231292517, 0.8958660387231816, 0.8952057013281504, 0.8973544973544973]
P(Acc | Certain): 0.9802237531820188 +- 0.0008447896849073807 [0.983127109111361, 0.9806651198762568, 0.98018147086915, 0.9788841964881084, 0.9782608695652174]
P(Acc | Uncertain): 0.6658386804182905 +- 0.004254498716107378 [0.6548262548262548, 0.6581740976645435, 0.6673139158576051, 0.6704477611940298, 0.6784313725490196]
[441 442 443 ... 2202 2203 2204] [0 1 2 3 4 5 6 7 8 9 10 11
12 13 14 15 16 17
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107
108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161

(continues on next page)

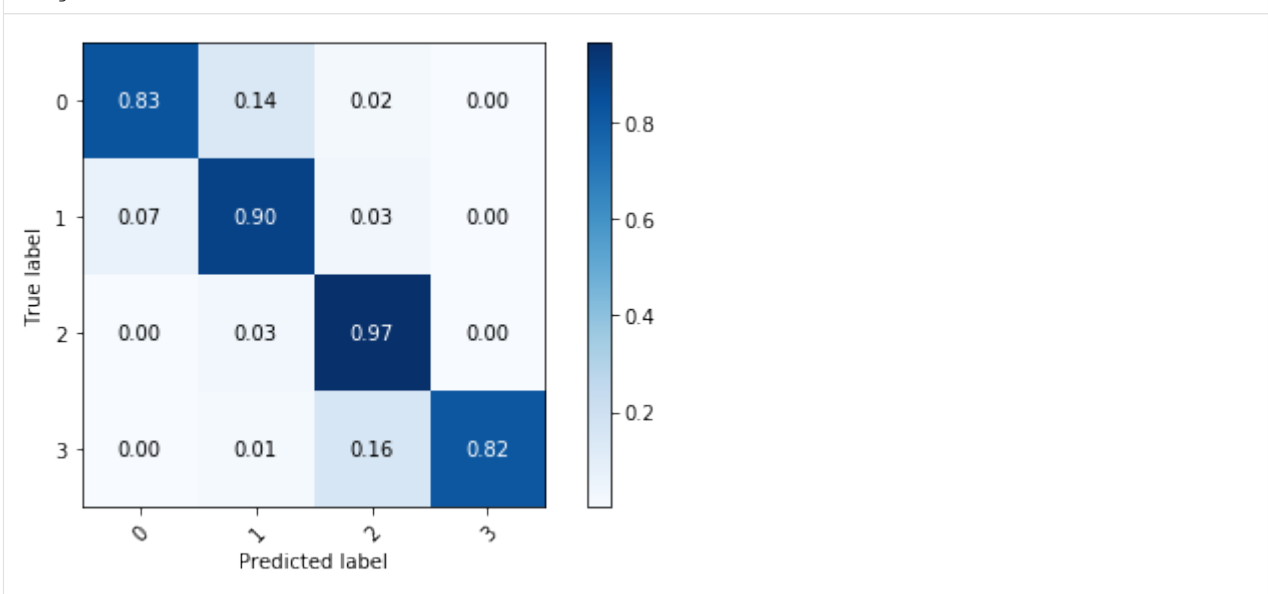
(continued from previous page)

162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197
198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215
216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233
234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251
252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269
270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287
288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305
306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323
324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341
342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359
360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377
378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395
396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413
414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431
432	433	434	435	436	437	438	439	440									

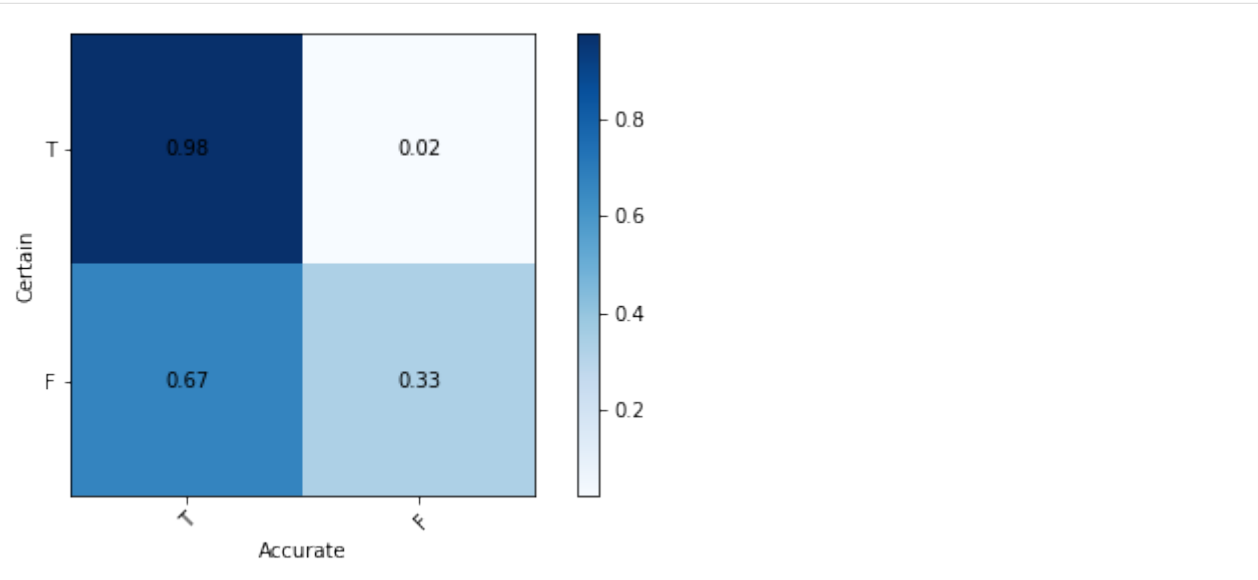
<Figure size 432x288 with 0 Axes>



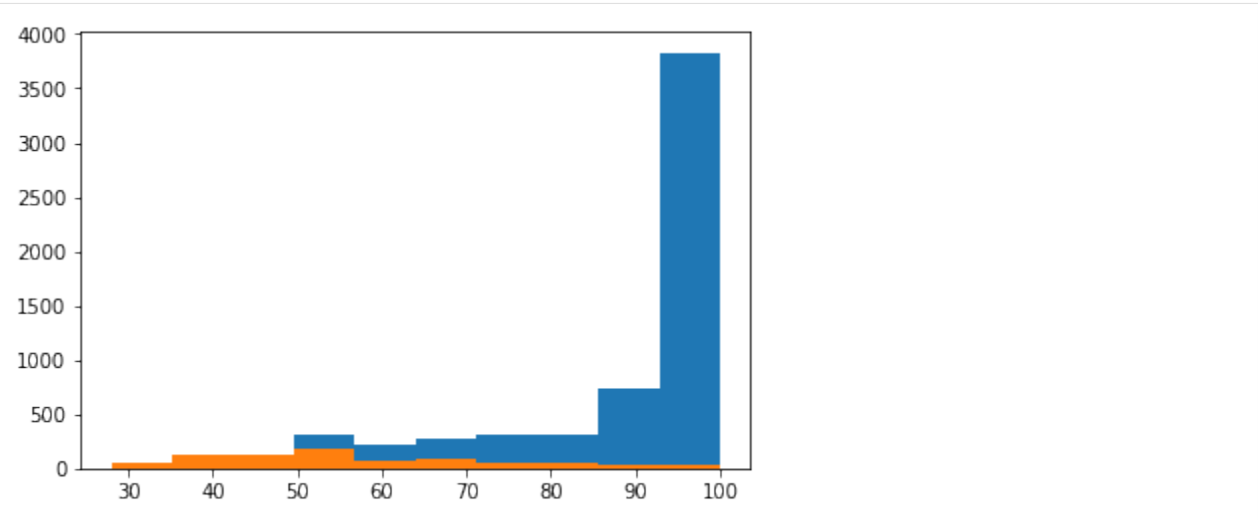
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



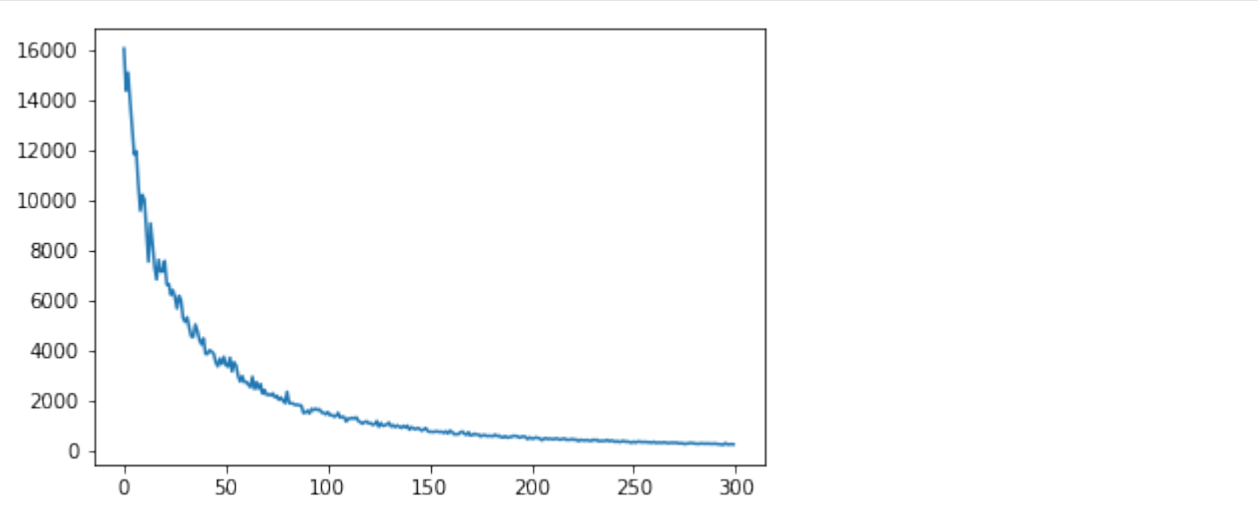
```
[ 0 1 2 ... 2202 2203 2204] [441 442 443 444 445 446 447 448 449 450 451 452]
↪ 453 454 455 456 457 458
459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476
477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494
495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512
513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530
531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548
549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566
567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584
585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620
621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638
639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674
675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692
693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710
711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728
```

(continues on next page)

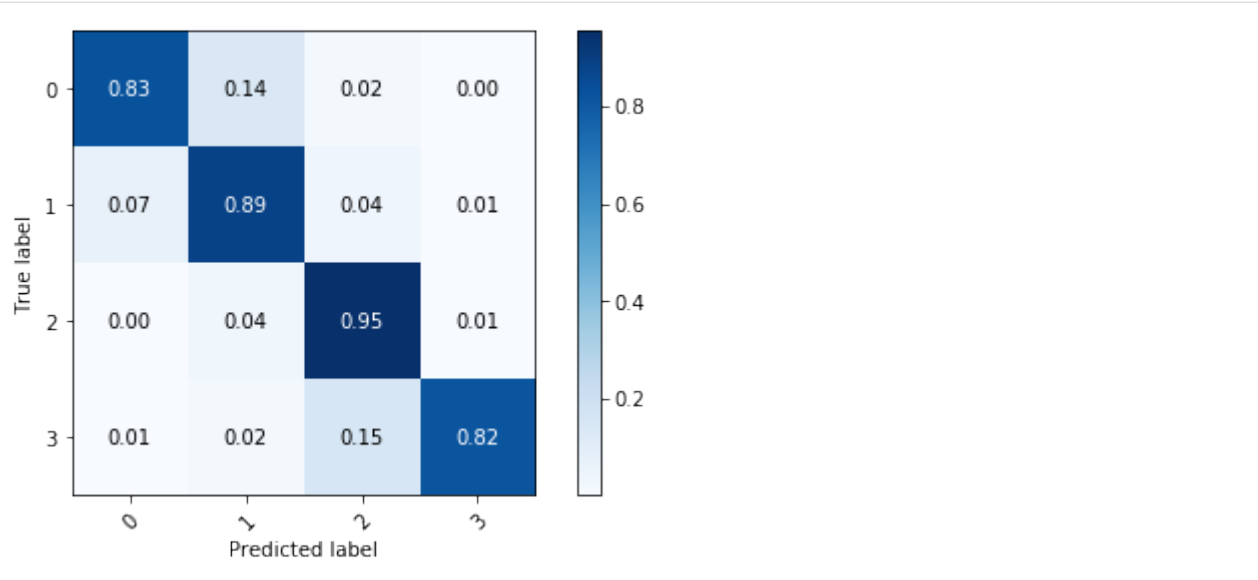
(continued from previous page)

```
729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746
747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764
765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782
783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800
801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818
819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836
837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854
855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872
873 874 875 876 877 878 879 880 881]
```

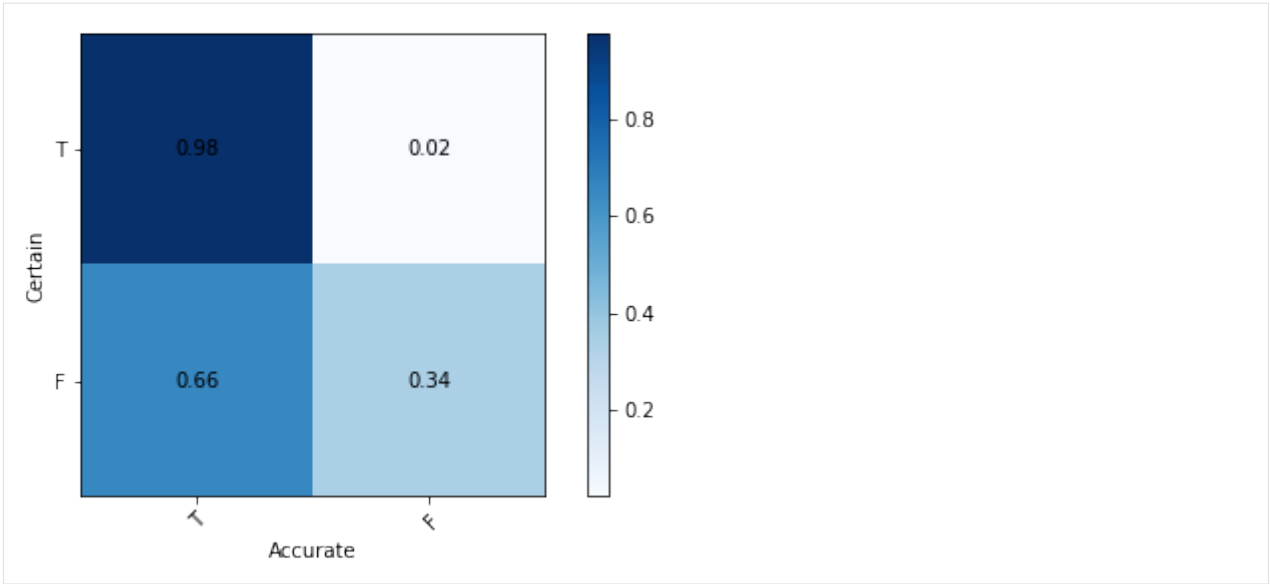
<Figure size 432x288 with 0 Axes>



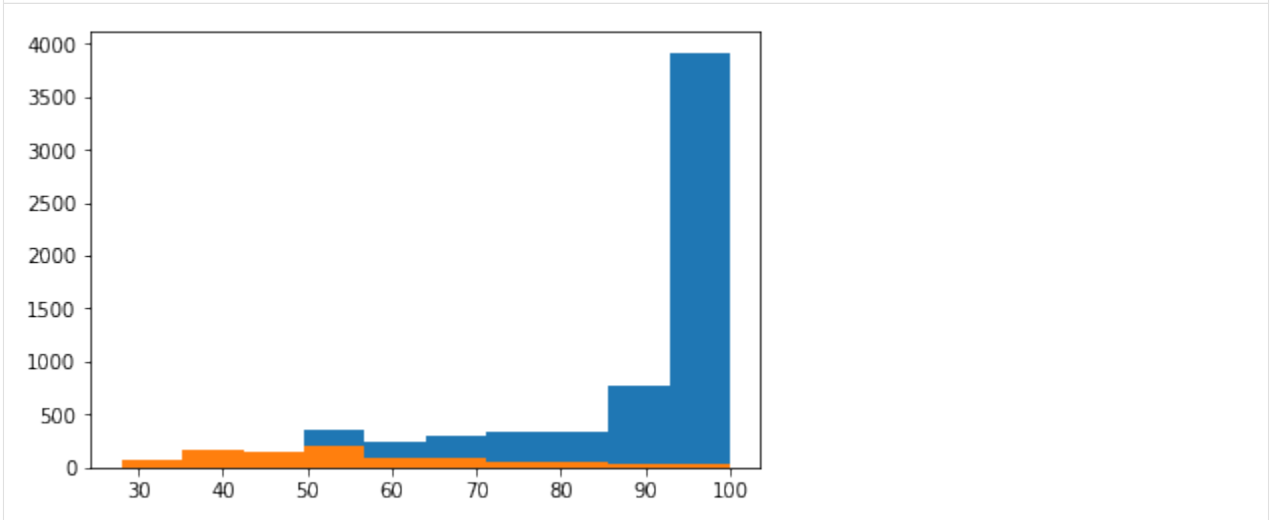
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



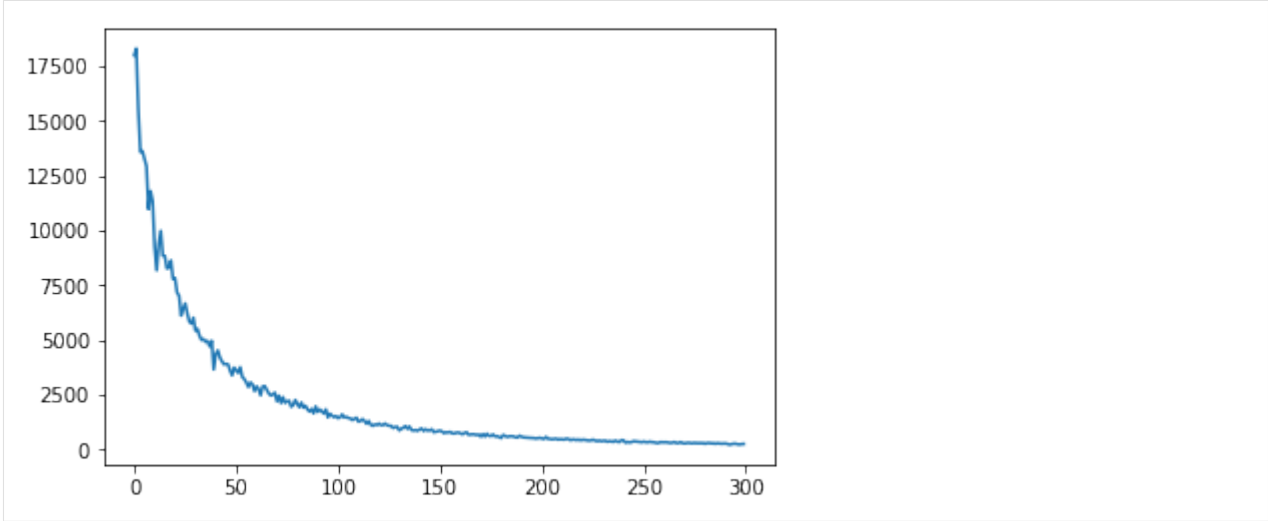
[0	1	2	...	2202	2203	2204]	[882	883	884	885	886	887	888	889	890	...
↪	891	892	893	894	895													
	896	897	898	899	900	901	902	903	904	905	906	907	908	909				
	910	911	912	913	914	915	916	917	918	919	920	921	922	923				
	924	925	926	927	928	929	930	931	932	933	934	935	936	937				
	938	939	940	941	942	943	944	945	946	947	948	949	950	951				
	952	953	954	955	956	957	958	959	960	961	962	963	964	965				
	966	967	968	969	970	971	972	973	974	975	976	977	978	979				
	980	981	982	983	984	985	986	987	988	989	990	991	992	993				
	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007				
	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021				
	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035				
	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049				
	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063				
	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077				
	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091				
	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105				
	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119				

(continues on next page)

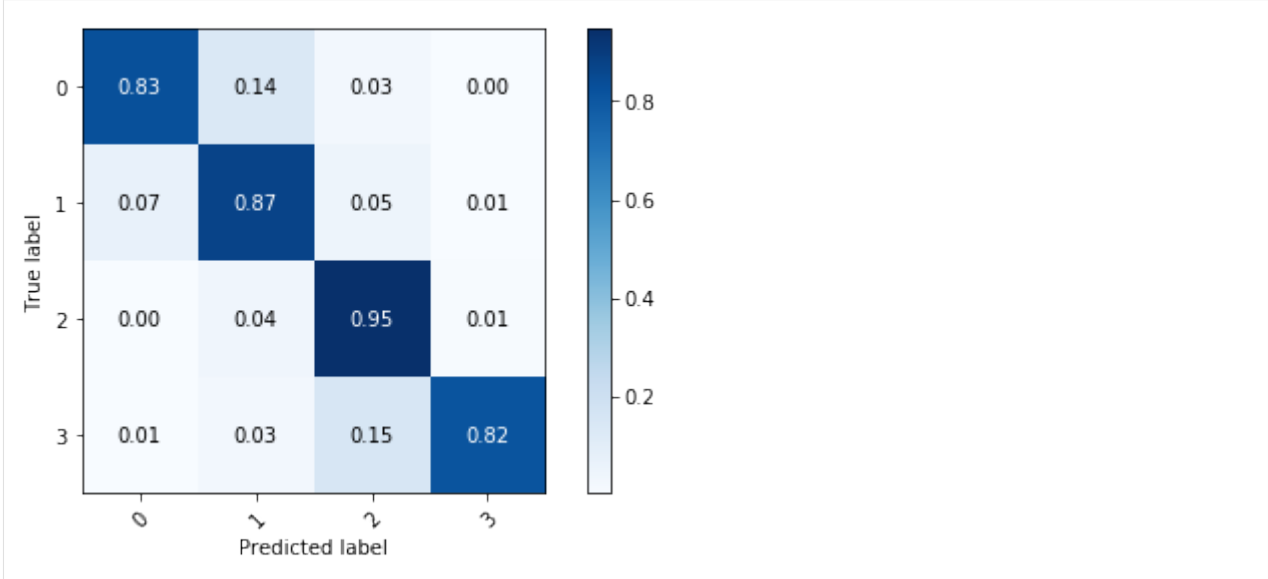
(continued from previous page)

1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133
1134	1135	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147
1148	1149	1150	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161
1162	1163	1164	1165	1166	1167	1168	1169	1170	1171	1172	1173	1174	1175
1176	1177	1178	1179	1180	1181	1182	1183	1184	1185	1186	1187	1188	1189
1190	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1202	1203
1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217
1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245
1246	1247	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259
1260	1261	1262	1263	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273
1274	1275	1276	1277	1278	1279	1280	1281	1282	1283	1284	1285	1286	1287
1288	1289	1290	1291	1292	1293	1294	1295	1296	1297	1298	1299	1300	1301
1302	1303	1304	1305	1306	1307	1308	1309	1310	1311	1312	1313	1314	1315
1316	1317	1318	1319	1320	1321	1322							

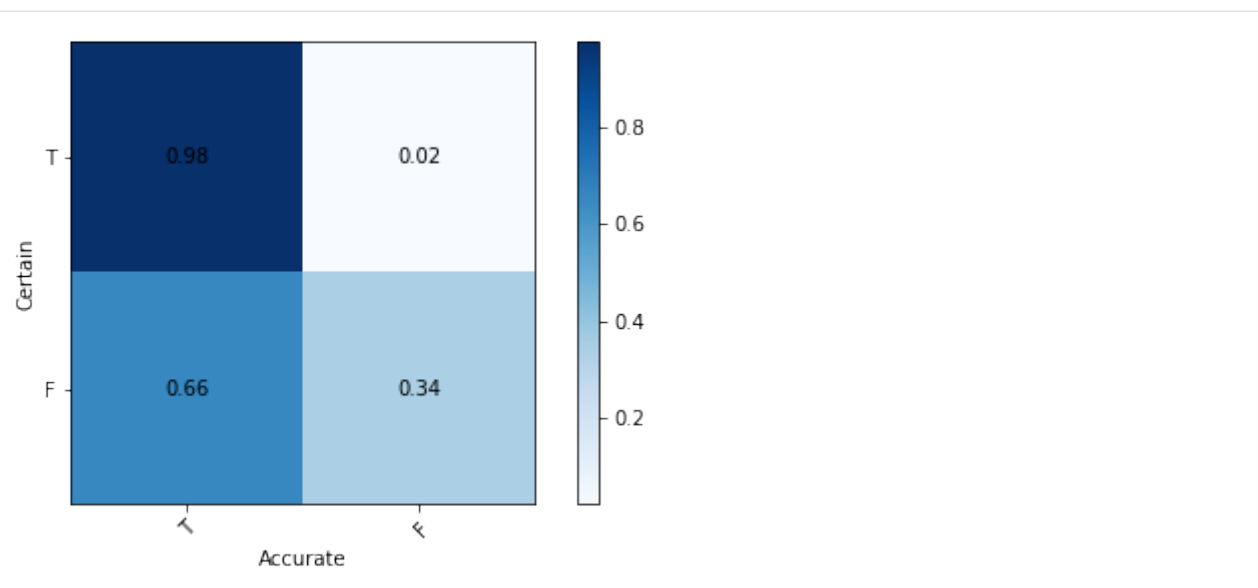
<Figure size 432x288 with 0 Axes>



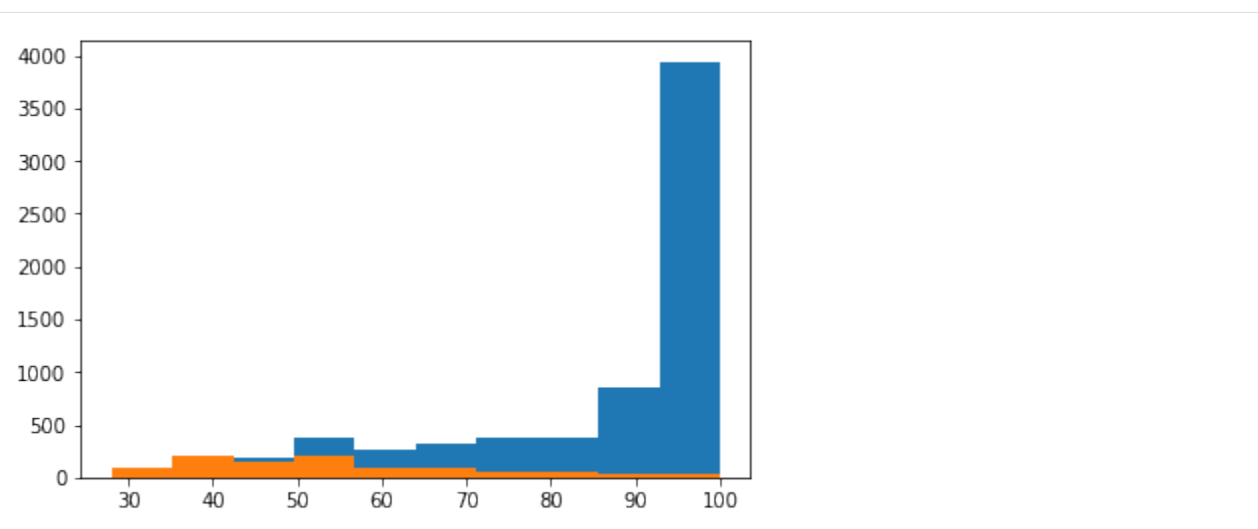
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



```
[ 0 1 2 ... 2202 2203 2204] [1323 1324 1325 1326 1327 1328 1329 1330 1331_
↪1332 1333 1334 1335 1336
1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364
1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378
1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406
1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420
1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434
1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448
1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462
1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476
1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490
1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504
1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518
1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532
1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546
```

(continues on next page)

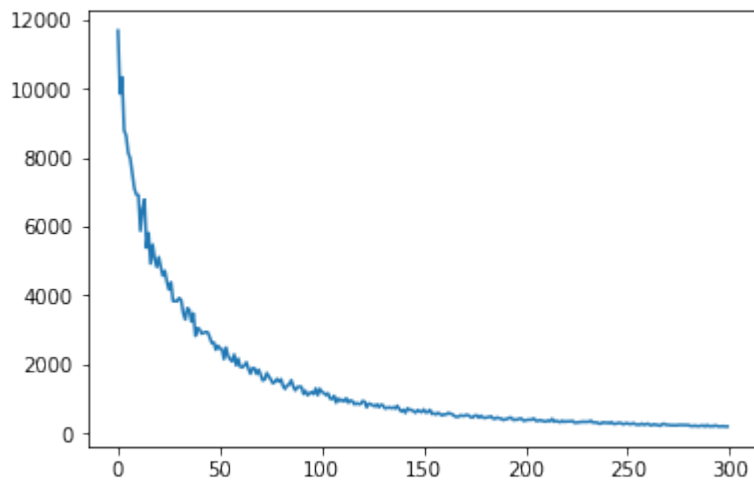
(continued from previous page)

```

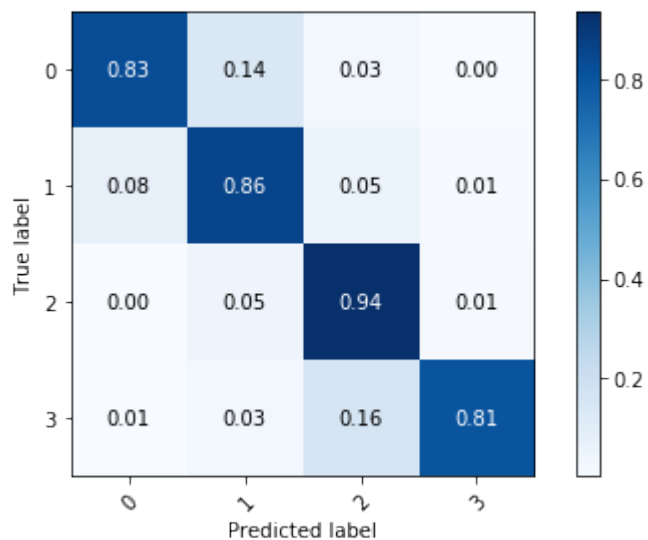
1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560
1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574
1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588
1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602
1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616
1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630
1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644
1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658
1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672
1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686
1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700
1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714
1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728
1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742
1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756
1757 1758 1759 1760 1761 1762 1763]

```

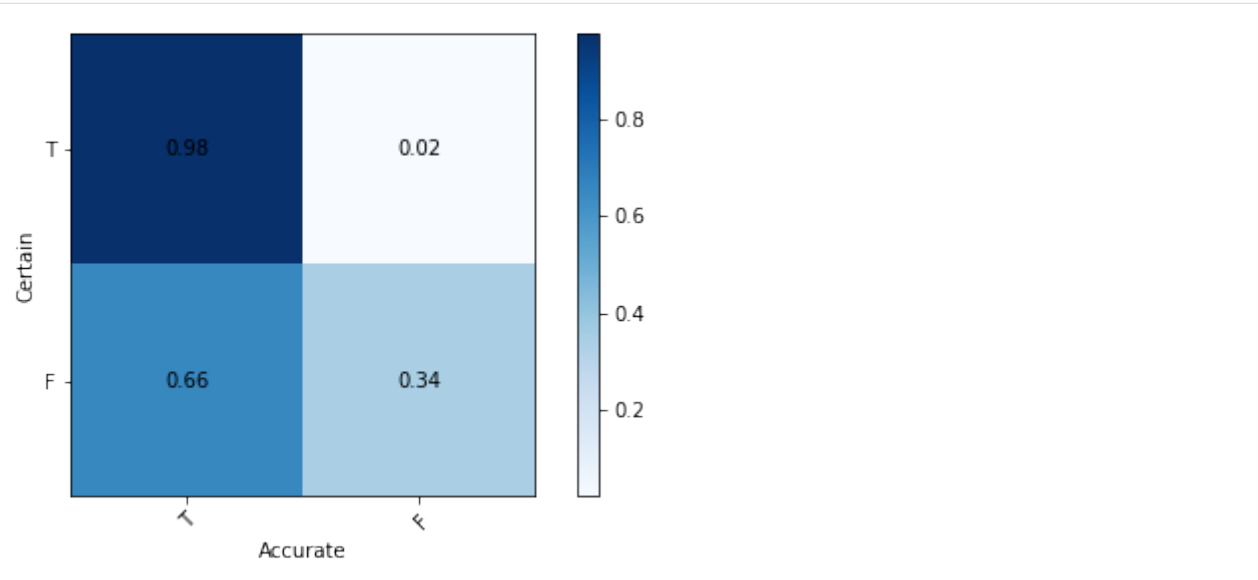
<Figure size 432x288 with 0 Axes>



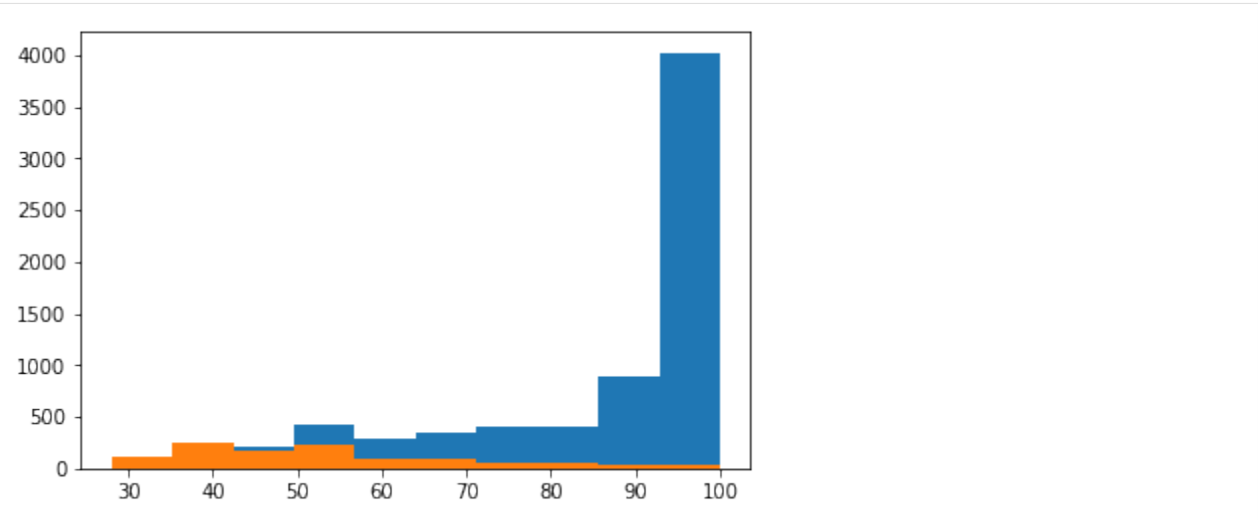
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



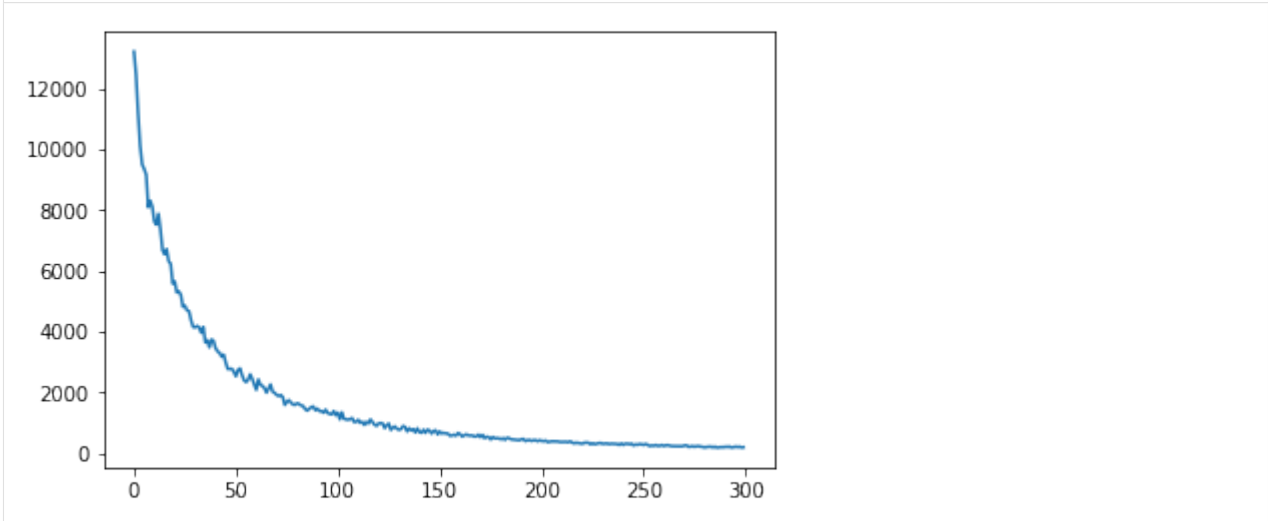
```
[ 0 1 2 ... 1761 1762 1763] [1764 1765 1766 1767 1768 1769 1770 1771 1772_
↪1773 1774 1775 1776 1777
1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791
1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805
1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819
1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833
1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847
1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861
1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875
1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889
1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903
1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917
1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931
1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945
1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959
1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973
1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987
```

(continues on next page)

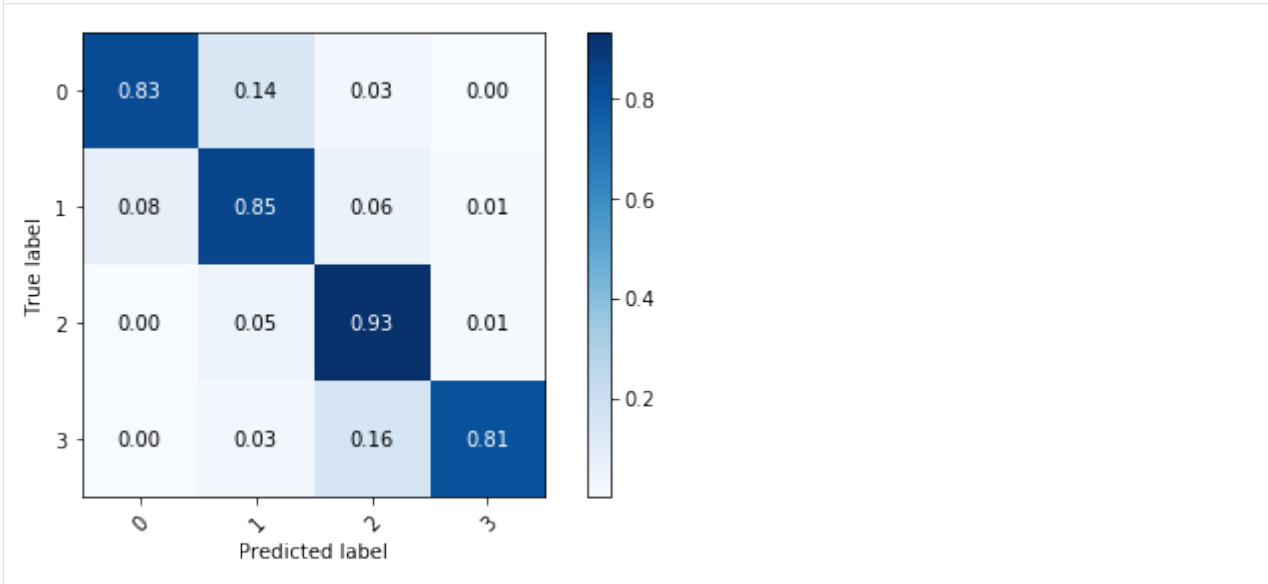
(continued from previous page)

1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029
2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043
2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057
2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071
2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085
2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099
2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113
2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141
2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155
2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169
2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183
2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197
2198	2199	2200	2201	2202	2203	2204							

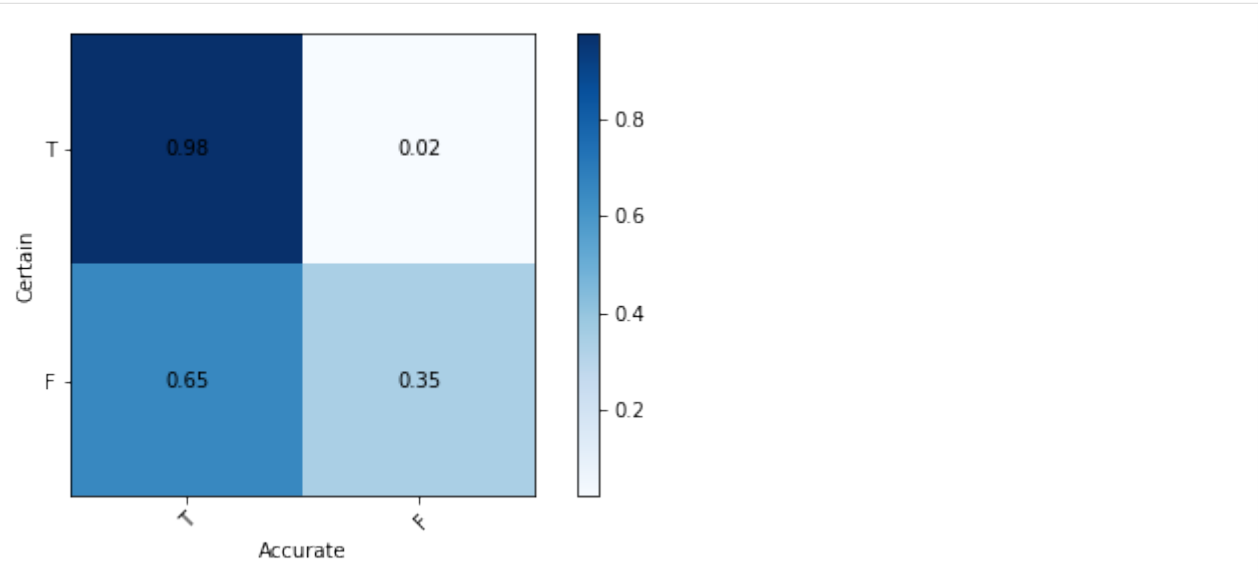
<Figure size 432x288 with 0 Axes>



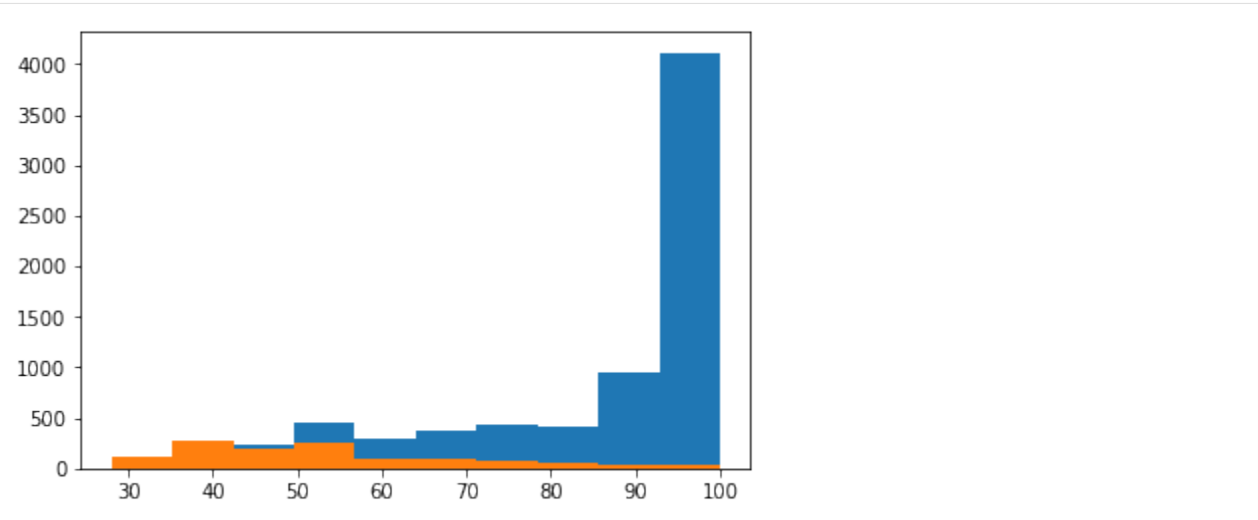
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



Target output: 3

F1 SCORE: 0.8721943923768338 +- 0.004728381107143069 [0.8864795918367347, 0.8782179538482061, 0.871504157218443, 0.8650196920873613, 0.859750566893424]

P(Acc | Certain): 0.9782688014829797 +- 8.581726290369855e-05 [0.9781036560867818, 0.9780241151302995, 0.9784090909090909, 0.9784490698102781, 0.9783580754784476]

P(Acc | Uncertain): 0.6594160576349053 +- 0.0021611149247641815 [0.6669874879692012, 0.6602972399150743, 0.6591422121896162, 0.656271186440678, 0.6543821616599567]

[441 442 443 ... 2202 2203 2204] [0 1 2 3 4 5 6 7 8 9 10 11

12 13 14 15 16 17

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35

36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53

54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71

72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89

90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107

108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125

126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143

144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161

(continues on next page)

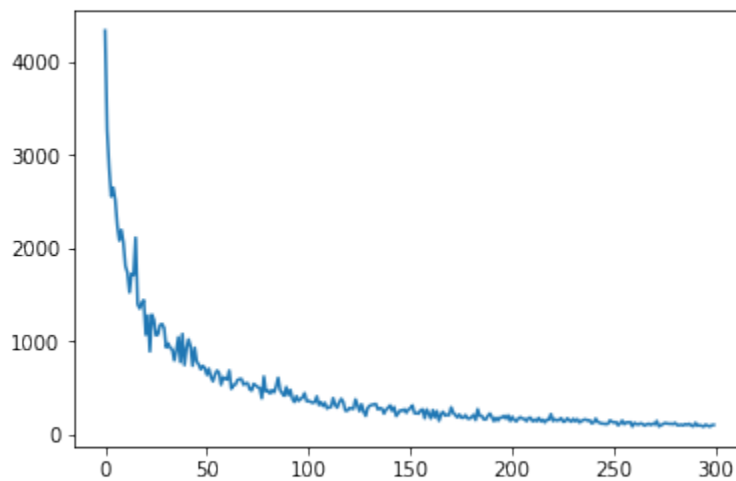
(continued from previous page)

```

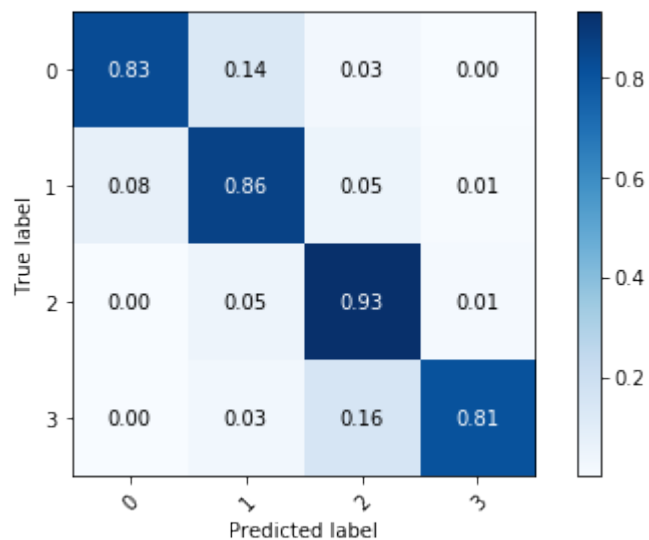
162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215
216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233
234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287
288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305
306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323
324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359
360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377
378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395
396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431
432 433 434 435 436 437 438 439 440]

```

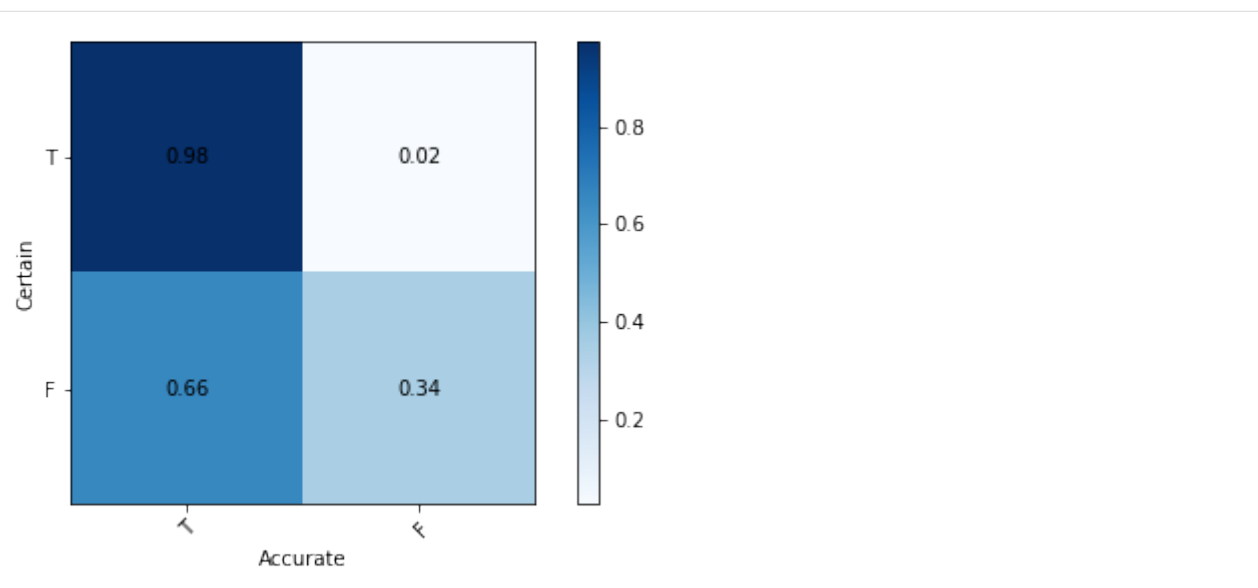
<Figure size 432x288 with 0 Axes>



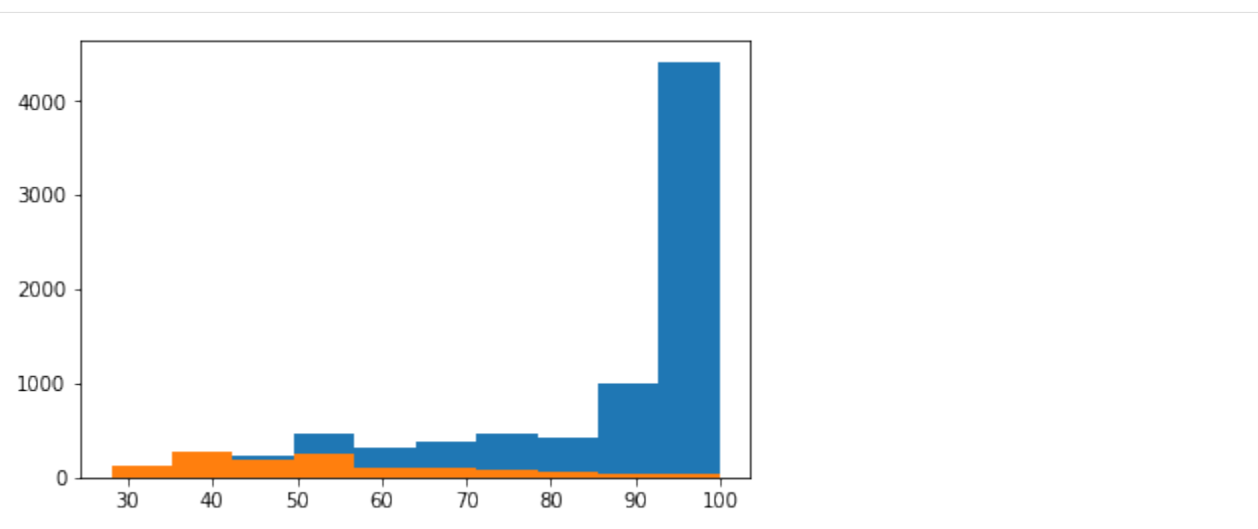
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



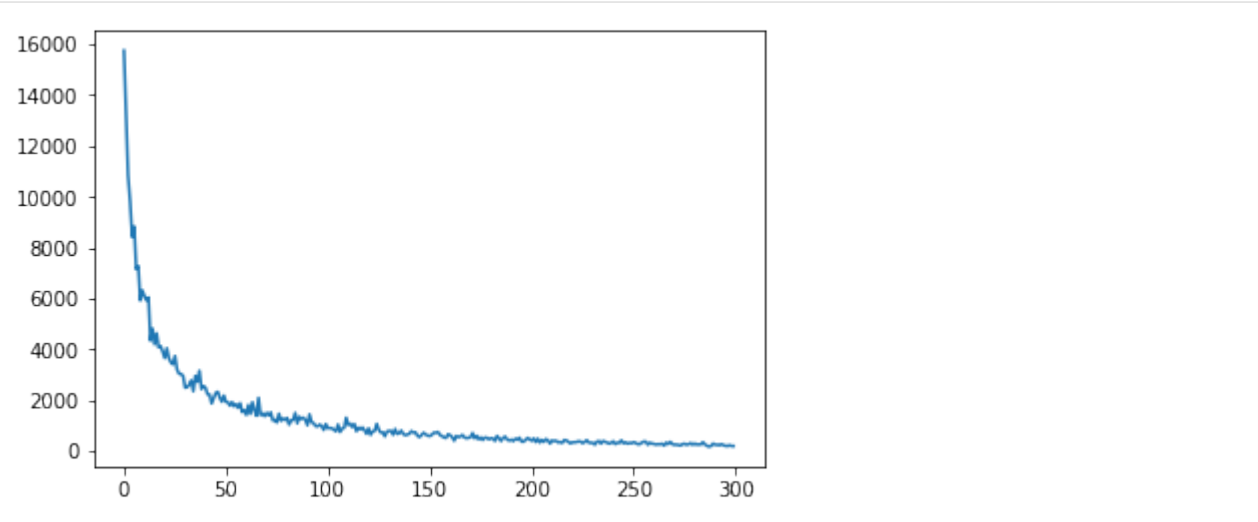
[0 1 2 ... 2202 2203 2204] [441 442 443 444 445 446 447 448 449 450 451 452]
↪453 454 455 456 457 458
459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476
477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494
495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512
513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530
531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548
549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566
567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584
585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602
603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620
621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638
639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656
657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674
675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692
693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710
711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728

(continues on next page)

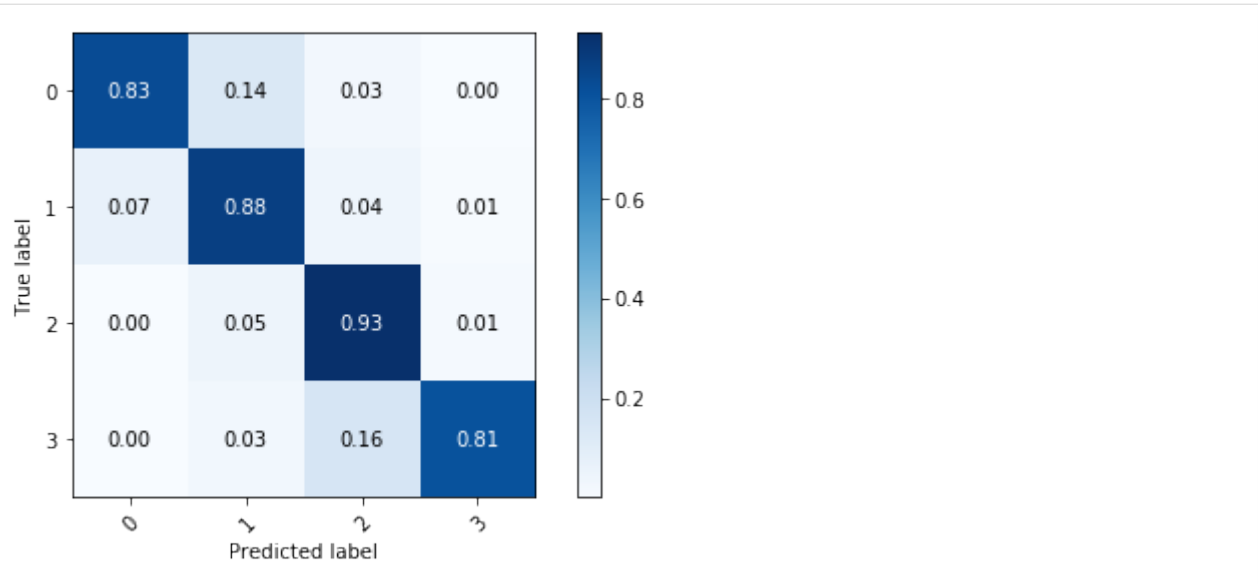
(continued from previous page)

```
729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746
747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764
765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782
783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800
801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818
819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836
837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854
855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872
873 874 875 876 877 878 879 880 881]
```

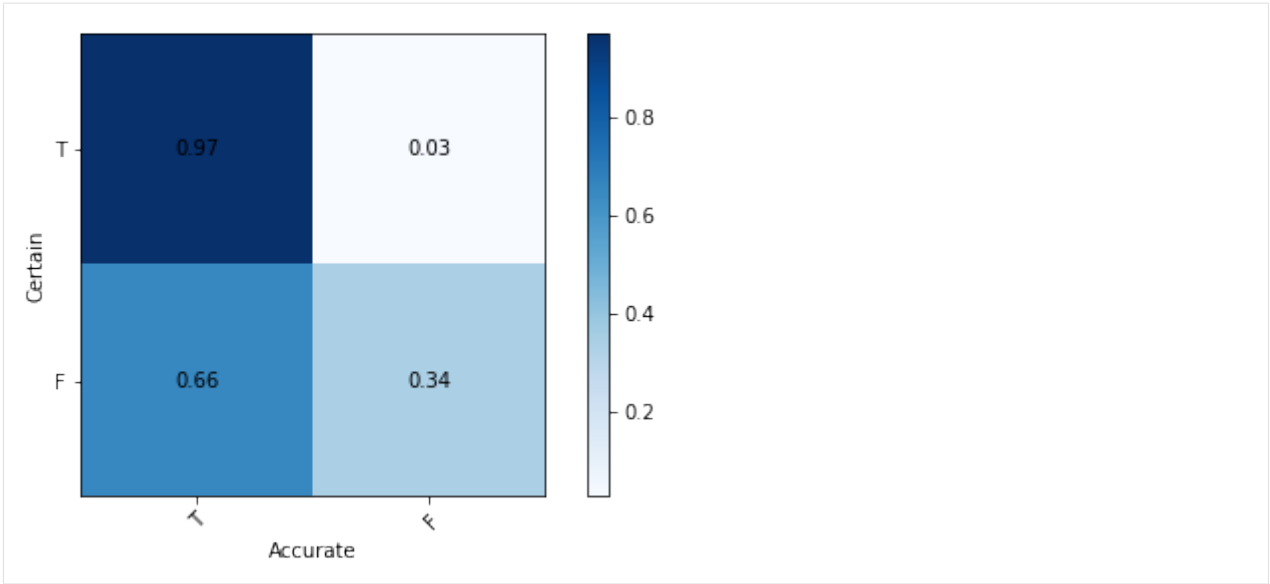
<Figure size 432x288 with 0 Axes>



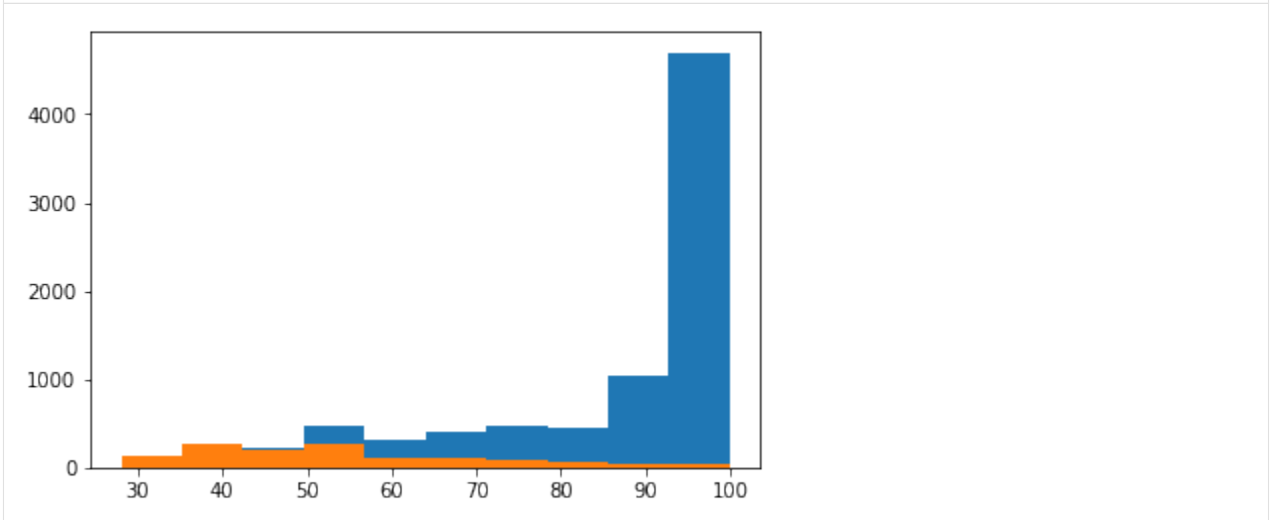
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



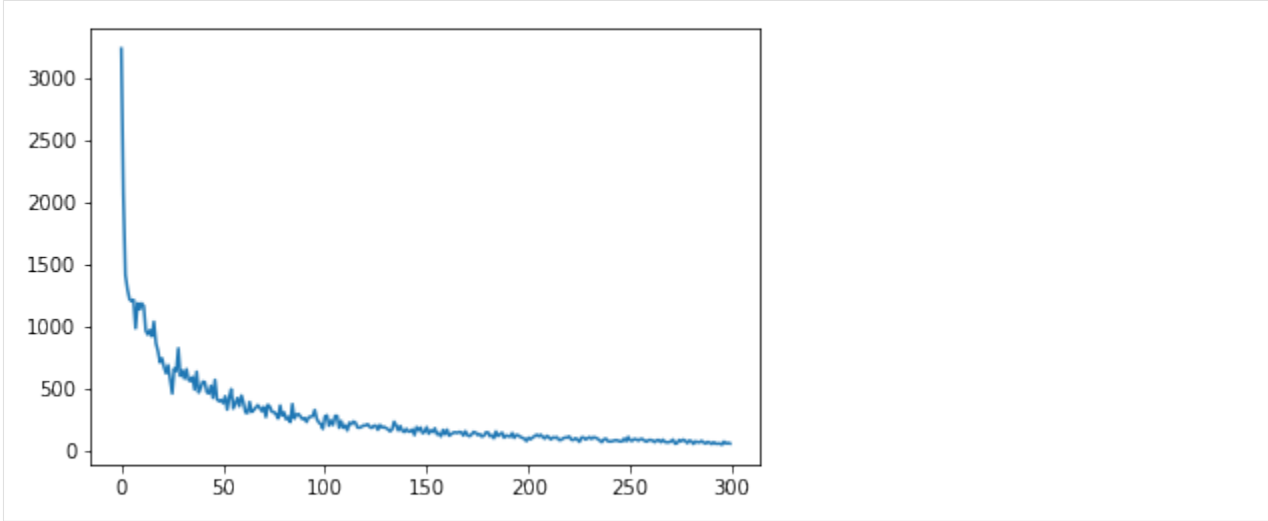
[0	1	2	...	2202	2203	2204]	[882	883	884	885	886	887	888	889	890	...
↪	891	892	893	894	895													
	896	897	898	899	900	901	902	903	904	905	906	907	908	909				
	910	911	912	913	914	915	916	917	918	919	920	921	922	923				
	924	925	926	927	928	929	930	931	932	933	934	935	936	937				
	938	939	940	941	942	943	944	945	946	947	948	949	950	951				
	952	953	954	955	956	957	958	959	960	961	962	963	964	965				
	966	967	968	969	970	971	972	973	974	975	976	977	978	979				
	980	981	982	983	984	985	986	987	988	989	990	991	992	993				
	994	995	996	997	998	999	1000	1001	1002	1003	1004	1005	1006	1007				
	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021				
	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035				
	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049				
	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063				
	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077				
	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091				
	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105				
	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119				

(continues on next page)

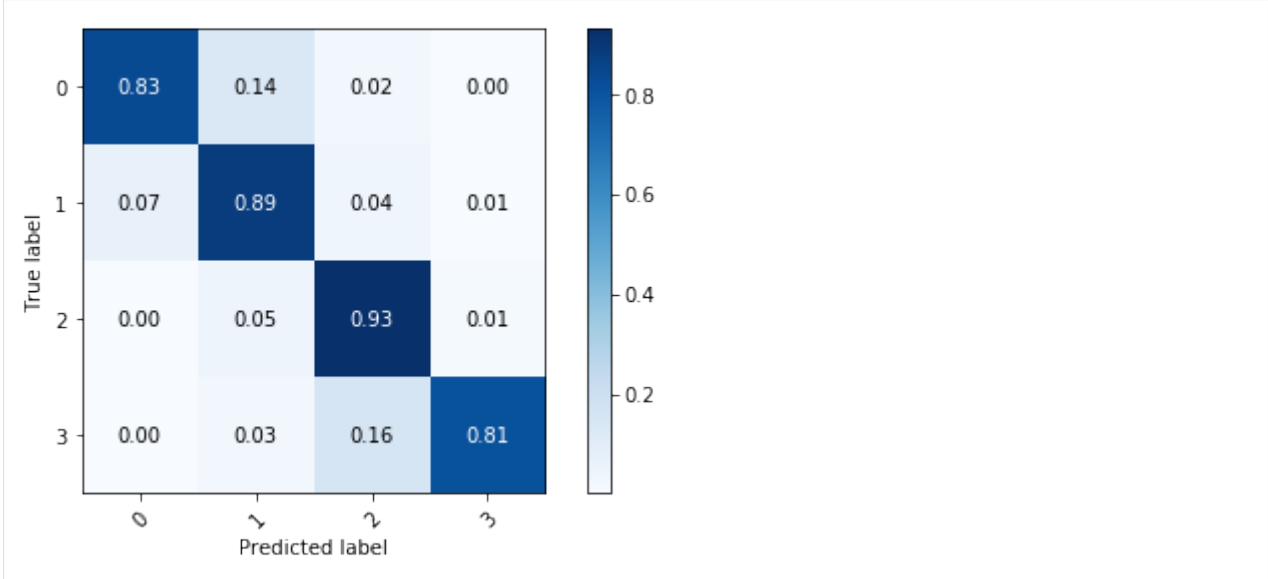
(continued from previous page)

1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133
1134	1135	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147
1148	1149	1150	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161
1162	1163	1164	1165	1166	1167	1168	1169	1170	1171	1172	1173	1174	1175
1176	1177	1178	1179	1180	1181	1182	1183	1184	1185	1186	1187	1188	1189
1190	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1202	1203
1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217
1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245
1246	1247	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259
1260	1261	1262	1263	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273
1274	1275	1276	1277	1278	1279	1280	1281	1282	1283	1284	1285	1286	1287
1288	1289	1290	1291	1292	1293	1294	1295	1296	1297	1298	1299	1300	1301
1302	1303	1304	1305	1306	1307	1308	1309	1310	1311	1312	1313	1314	1315
1316	1317	1318	1319	1320	1321	1322							

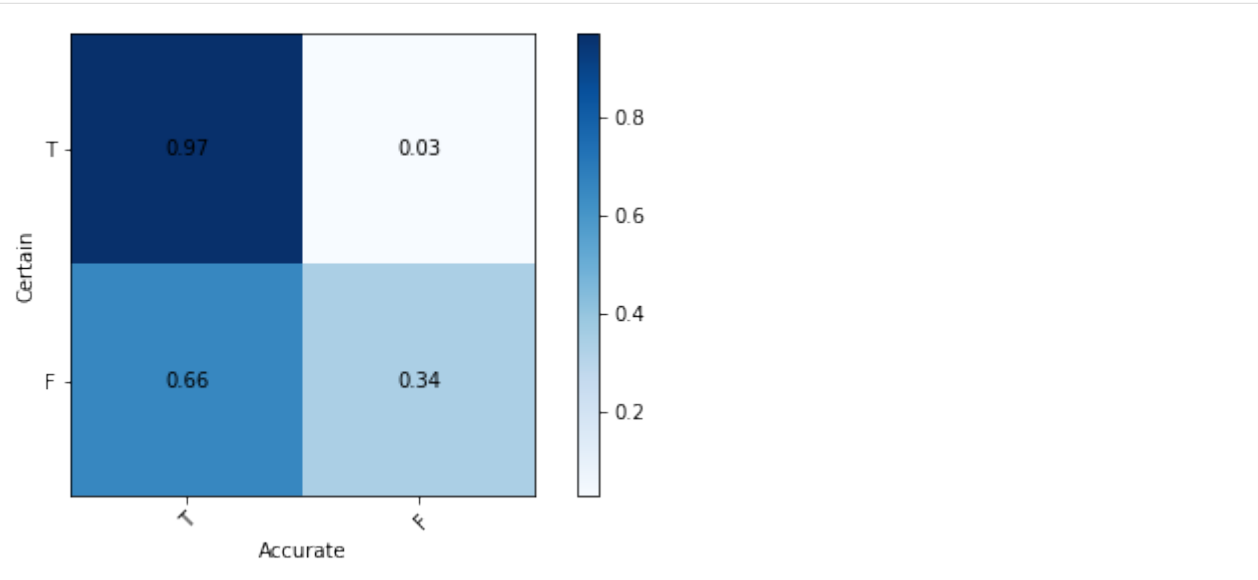
<Figure size 432x288 with 0 Axes>



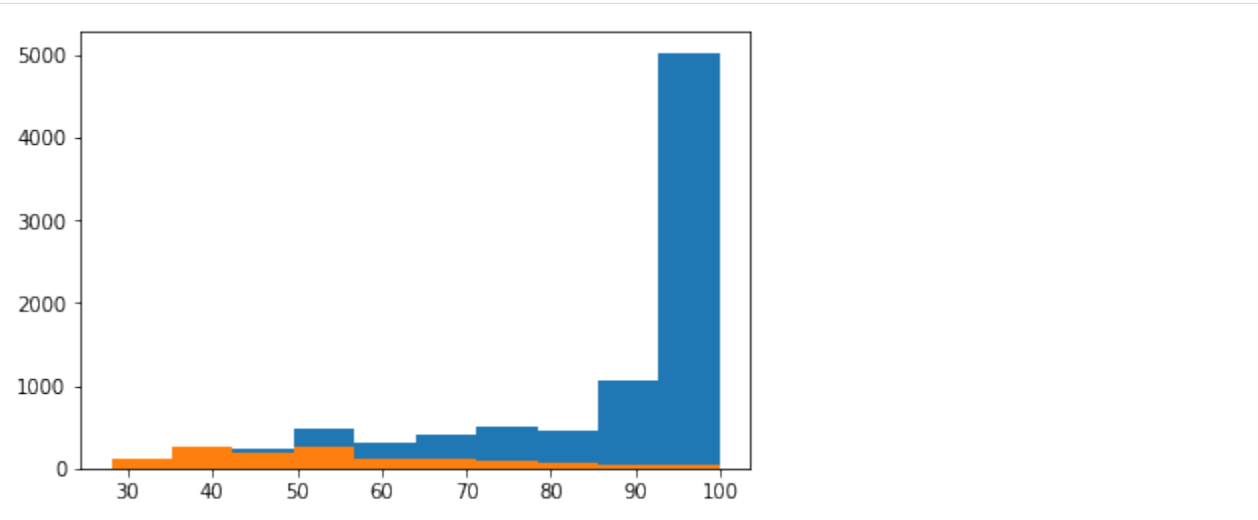
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



```
[ 0 1 2 ... 2202 2203 2204] [1323 1324 1325 1326 1327 1328 1329 1330 1331_
↪1332 1333 1334 1335 1336
1337 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
1351 1352 1353 1354 1355 1356 1357 1358 1359 1360 1361 1362 1363 1364
1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378
1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
1393 1394 1395 1396 1397 1398 1399 1400 1401 1402 1403 1404 1405 1406
1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420
1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434
1435 1436 1437 1438 1439 1440 1441 1442 1443 1444 1445 1446 1447 1448
1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460 1461 1462
1463 1464 1465 1466 1467 1468 1469 1470 1471 1472 1473 1474 1475 1476
1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490
1491 1492 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504
1505 1506 1507 1508 1509 1510 1511 1512 1513 1514 1515 1516 1517 1518
1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532
1533 1534 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546
```

(continues on next page)

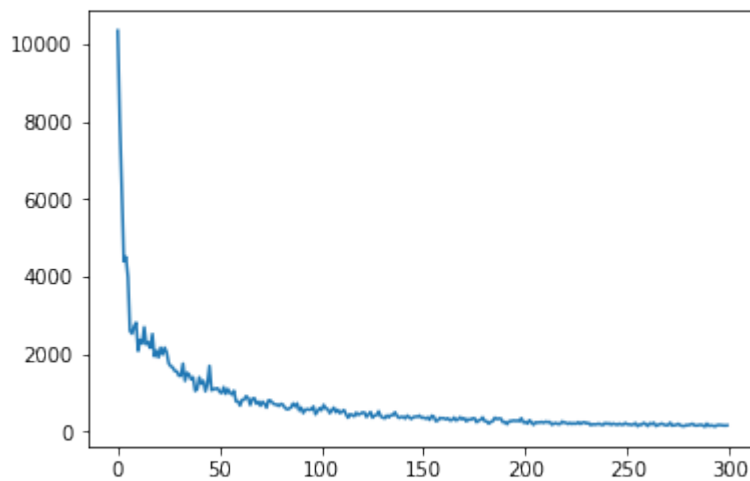
(continued from previous page)

```

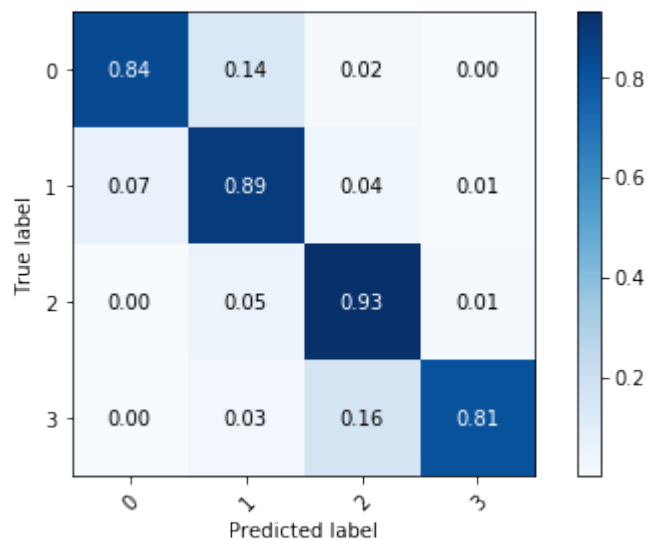
1547 1548 1549 1550 1551 1552 1553 1554 1555 1556 1557 1558 1559 1560
1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574
1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588
1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 1599 1600 1601 1602
1603 1604 1605 1606 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616
1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630
1631 1632 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644
1645 1646 1647 1648 1649 1650 1651 1652 1653 1654 1655 1656 1657 1658
1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672
1673 1674 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686
1687 1688 1689 1690 1691 1692 1693 1694 1695 1696 1697 1698 1699 1700
1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714
1715 1716 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728
1729 1730 1731 1732 1733 1734 1735 1736 1737 1738 1739 1740 1741 1742
1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753 1754 1755 1756
1757 1758 1759 1760 1761 1762 1763]

```

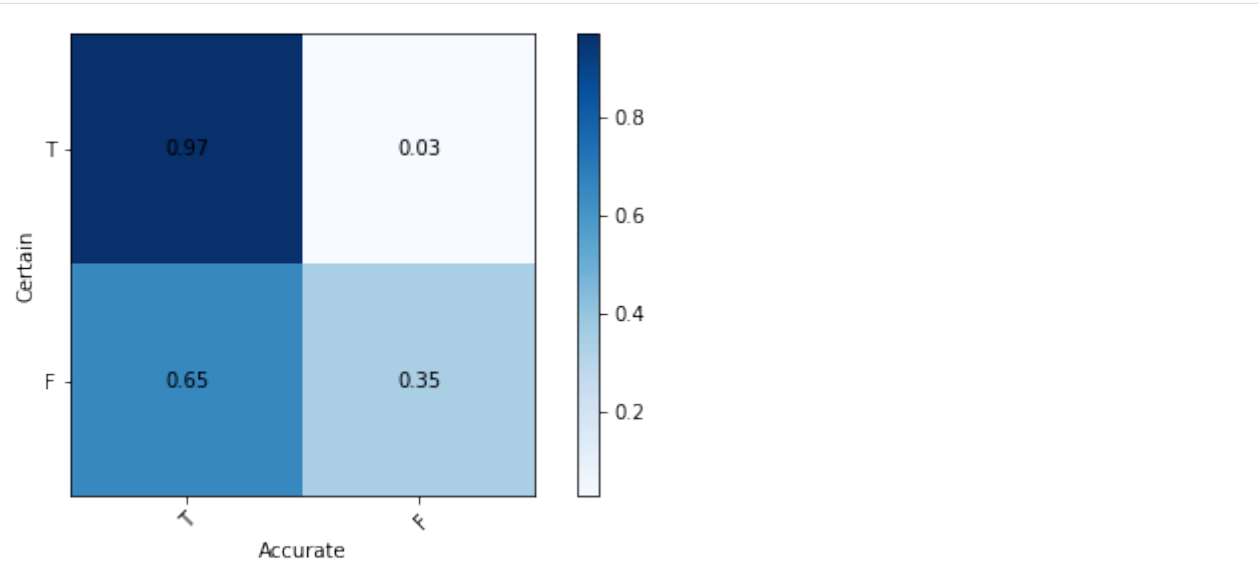
<Figure size 432x288 with 0 Axes>



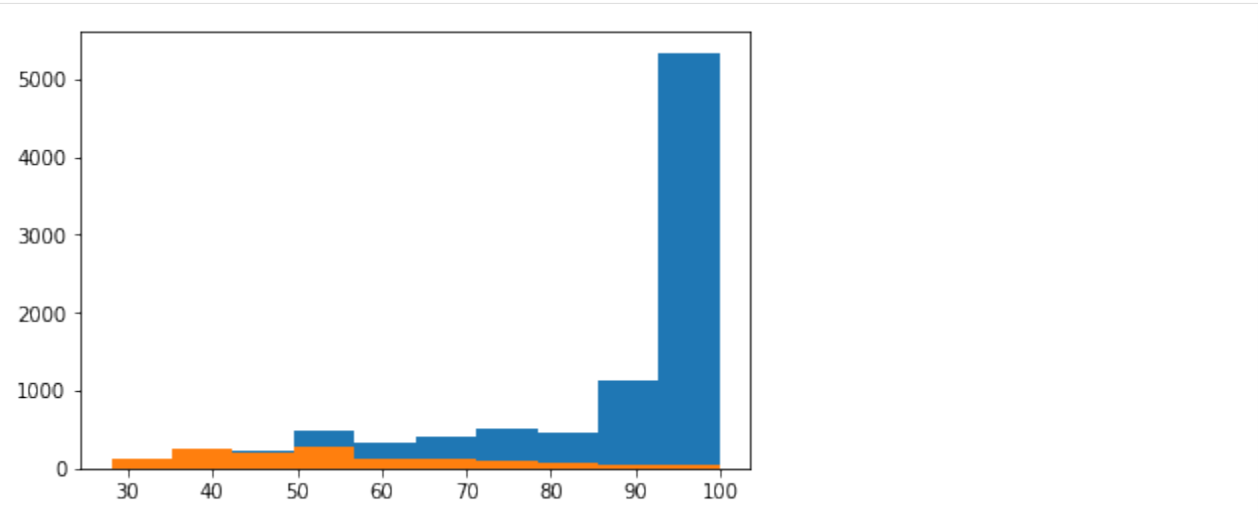
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



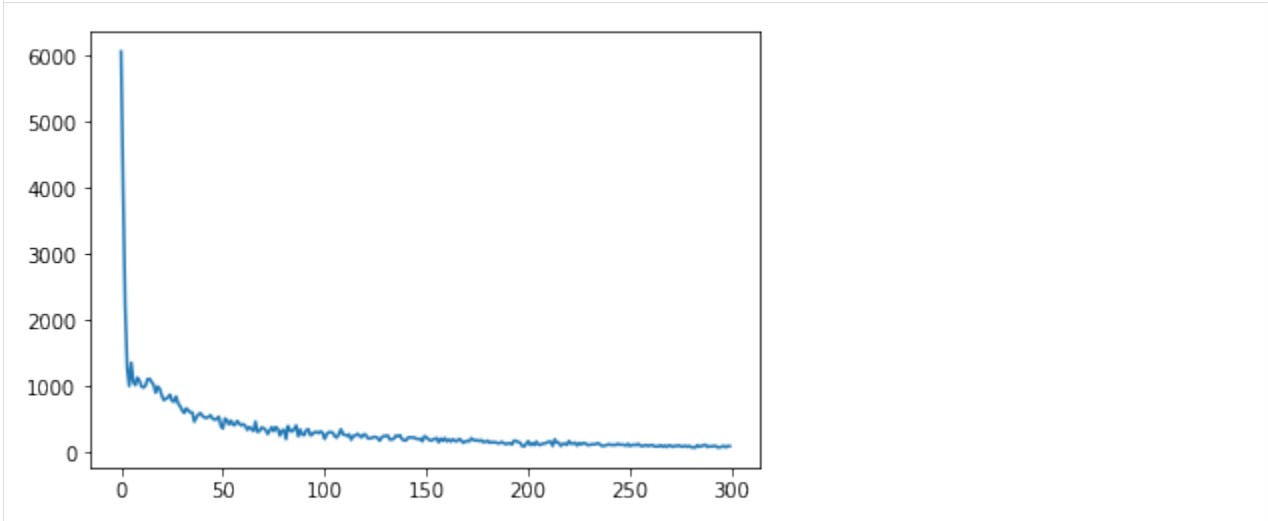
```
[ 0 1 2 ... 1761 1762 1763] [1764 1765 1766 1767 1768 1769 1770 1771 1772_
↪1773 1774 1775 1776 1777
1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791
1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805
1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819
1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833
1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847
1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861
1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875
1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889
1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903
1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917
1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931
1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945
1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959
1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973
1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987
```

(continues on next page)

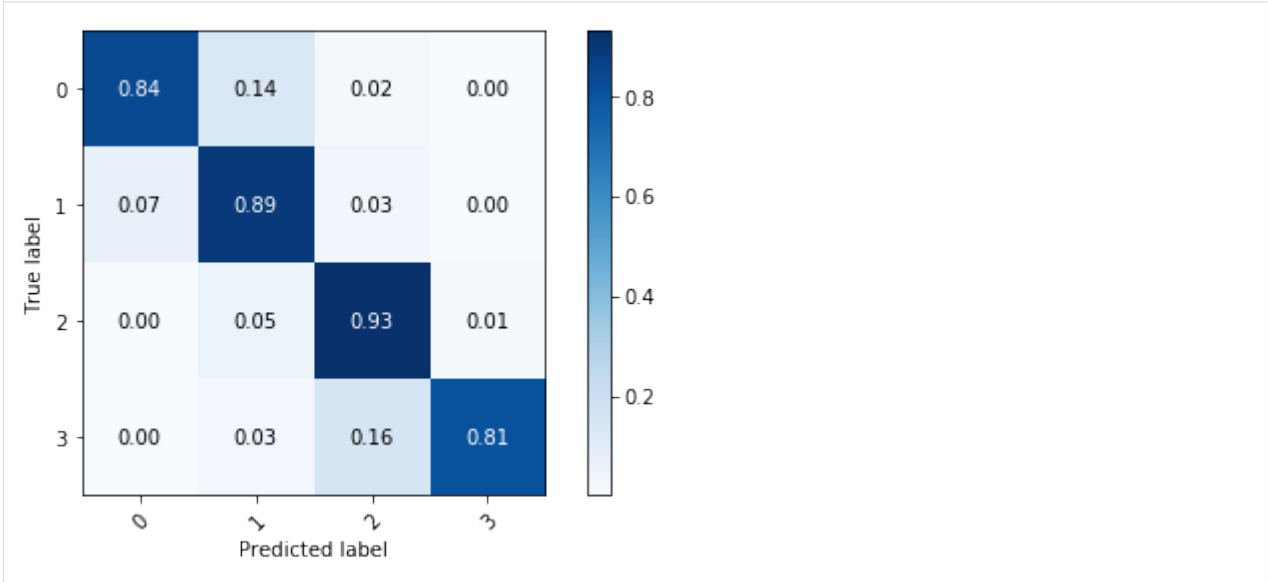
(continued from previous page)

1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001
2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029
2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043
2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057
2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071
2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085
2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099
2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113
2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141
2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155
2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169
2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183
2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197
2198	2199	2200	2201	2202	2203	2204							

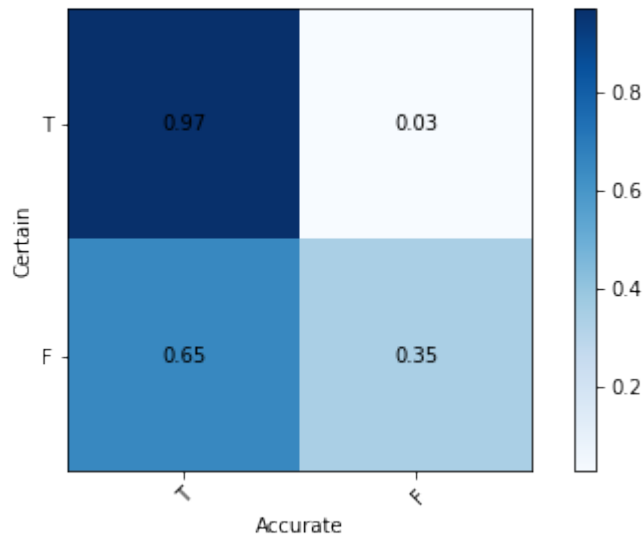
<Figure size 432x288 with 0 Axes>



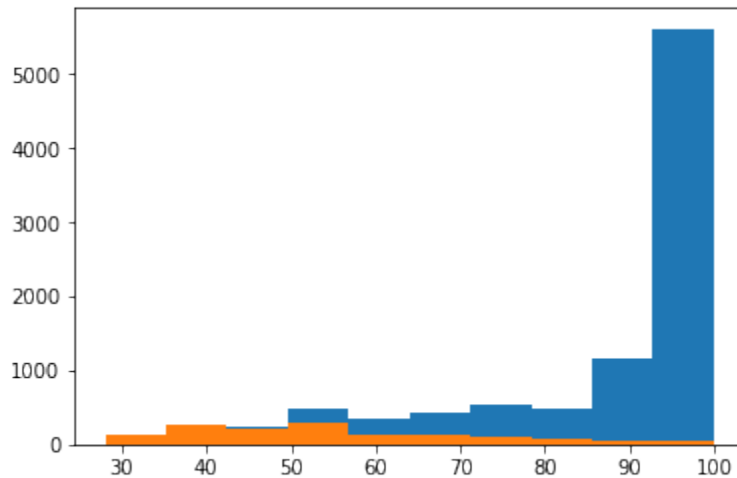
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



Target output: 4

F1 SCORE: 0.8668427457060999 +- 0.0014700490292759586 [0.8624338624338624, 0.8646670789527933, 0.8673962338558612, 0.8693310657596371, 0.8703854875283448]
 P(Acc | Certain): 0.974729743470298 +- 0.0005165493723274688 [0.9765494137353434, 0.9749290444654684, 0.9745838287752675, 0.9741173160782107, 0.9734691142971995]
 P(Acc | Uncertain): 0.6555131429772774 +- 0.00039401996474623446 [0.6554238833181404, 0.656547619047619, 0.6562225475841874, 0.6549640287769785, 0.654407636159461]

<Figure size 432x288 with 0 Axes>

```
[6]: # print results
task_names = ["Cooler Condition", "Valve Condition", "Internal Pump Leakage", "Accumulator", "Stable Flag"]

for i in range(len(final_results)):
    f1_score_model_kfold = final_results[i][0]
    p_accurate_certain_kfold = final_results[i][1]
```

(continues on next page)

(continued from previous page)

```

p_accurate_uncertain_kfold = final_results[i][2]
print("Target output: ", task_names[i])
print("F1 SCORE: ", np.mean(fl_score_model_kfold), "+-", sem(fl_score_model_kfold),
↪ fl_score_model_kfold)
print("P(Acc | Certain): ", np.mean(p_accurate_certain_kfold), "+-", sem(p_accurate_
↪ certain_kfold), p_accurate_certain_kfold)
print("P(Acc | Uncertain): ", np.mean(p_accurate_uncertain_kfold), "+-", sem(p_
↪ accurate_uncertain_kfold), p_accurate_uncertain_kfold)
print("\n")

```

Target output: Cooler Condition

F1 SCORE: 0.9945351473922903 +- 0.0004707605326990252 [0.9931972789115646, 0.
↪ 9954648526077098, 0.9954648526077098, 0.9948979591836735, 0.9936507936507937]
P(Acc | Certain): 0.9980861983060005 +- 0.00024045557469217787 [0.997716894977169, 0.
↪ 9988465974625144, 0.9984662576687117, 0.9977011494252873, 0.9977000919963201]
P(Acc | Uncertain): 0.6848302207130731 +- 0.08938562630975022 [0.3333333333333333, 0.
↪ 8, 0.7894736842105263, 0.7916666666666666, 0.7096774193548387]

Target output: Valve Condition

F1 SCORE: 0.9244593816362524 +- 0.012535227234323546 [0.9622071050642479, 0.
↪ 9416909620991255, 0.9220521541950113, 0.9022423784328546, 0.8941043083900226]
P(Acc | Certain): 0.9909499081179456 +- 0.001979188876666969 [0.9966144731273805, 0.
↪ 9934665641813989, 0.9911253106141285, 0.9883488681757656, 0.9851943244910549]
P(Acc | Uncertain): 0.6525351521609853 +- 0.0074524186694632615 [0.6749116607773852, ↪
↪ 0.6639175257731958, 0.6483825597749648, 0.6341968911917099, 0.6412671232876712]

Target output: Internal Pump Leakage

F1 SCORE: 0.8956939054898239 +- 0.0004668869383694529 [0.8954854669140383, 0.
↪ 8945578231292517, 0.8958660387231816, 0.8952057013281504, 0.8973544973544973]
P(Acc | Certain): 0.9802237531820188 +- 0.0008447896849073807 [0.983127109111361, 0.
↪ 9806651198762568, 0.98018147086915, 0.9788841964881084, 0.9782608695652174]
P(Acc | Uncertain): 0.6658386804182905 +- 0.004254498716107378 [0.6548262548262548, ↪
↪ 0.6581740976645435, 0.6673139158576051, 0.6704477611940298, 0.6784313725490196]

Target output: Accumulator

F1 SCORE: 0.8721943923768338 +- 0.004728381107143069 [0.8864795918367347, 0.
↪ 8782179538482061, 0.871504157218443, 0.8650196920873613, 0.859750566893424]
P(Acc | Certain): 0.9782688014829797 +- 8.581726290369855e-05 [0.9781036560867818, 0.
↪ 9780241151302995, 0.9784090909090909, 0.9784490698102781, 0.9783580754784476]
P(Acc | Uncertain): 0.6594160576349053 +- 0.0021611149247641815 [0.6669874879692012, ↪
↪ 0.6602972399150743, 0.6591422121896162, 0.656271186440678, 0.6543821616599567]

Target output: Stable Flag

F1 SCORE: 0.8668427457060999 +- 0.0014700490292759586 [0.8624338624338624, 0.
↪ 8646670789527933, 0.8673962338558612, 0.8693310657596371, 0.8703854875283448]
P(Acc | Certain): 0.974729743470298 +- 0.0005165493723274688 [0.9765494137353434, 0.
↪ 9749290444654684, 0.9745838287752675, 0.9741173160782107, 0.9734691142971995]
P(Acc | Uncertain): 0.6555131429772774 +- 0.00039401996474623446 [0.6554238833181404,
↪ 0.656547619047619, 0.6562225475841874, 0.6549640287769785, 0.654407636159461]

Run Multi Agent System

We initialize the data source and create the agent network, once the agents are fully up and running, run the dashboard code in a separate terminal to visualize the agents.

```
[1]: import osbrain
from osbrain.agent import run_agent
from osbrain.agent import Agent

import pandas as pd
from datetime import datetime

import time

import pickle
import numpy as np
import random
from copy import copy

from .Agent_models.agents import Sensor, Aggregator, Predictor, DecisionMaker, \
↳ SensorNetwork

# TYPES OF AGENT
# 0 - SENSOR NETWORK
# 1 - SENSOR
# 2 - AGGREGATOR
# 3 - PREDICTOR
# 4 - DECISIONMAKER

-----
ModuleNotFoundError                                Traceback (most recent call last)
<ipython-input-1-6053d4936c6f> in <module>
    13 from copy import copy
    14
--> 15 from .Agent_models.agents import Sensor, Aggregator, Predictor, DecisionMaker, \
↳ SensorNetwork
    16
```

(continues on next page)

(continued from previous page)

17 # TYPES OF AGENT

```
ModuleNotFoundError: No module named '__main__.Agent_models'; '__main__' is not a
↳package
```

```
[2]: DemoMode= True
pickle_path = "pickles/"
data_input = pickle.load(open(pickle_path + "data_input_data_1Hz_full.p", "rb"))
data_output = pickle.load(open(pickle_path + "zema_outputs.p", "rb"))

X_data = data_input
Y_data = data_output
randomShuffling = True

if (randomShuffling == True):
    index_list = np.arange(X_data.shape[0])
    random.shuffle(index_list)
    Y_data = Y_data[index_list, :]
    X_data = X_data[index_list, :, :]

-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-2-8d9935992e57> in <module>
      1 DemoMode= True
      2 pickle_path = "pickles/"
----> 3 data_input = pickle.load(open(pickle_path + "data_input_data_1Hz_full.p", "rb"
↳"))
      4 data_output = pickle.load(open(pickle_path + "zema_outputs.p", "rb"))
      5

FileNotFoundError: [Errno 2] No such file or directory: 'pickles/data_input_data_1Hz_
↳full.p'
```

5.1 Starting server

```
[3]: ns = osbrain.nameserver.run_nameserver(addr='127.0.0.1:14065')

Broadcast server running on 0.0.0.0:9091
NS running on 127.0.0.1:14065 (127.0.0.1)
URI = PYRO:Pyro.NameServer@127.0.0.1:14065
```

5.2 Creating Agent

We firstly create a SensorNetwork Agent which enable wrapper functions and manages agents

```
[4]: sensor_network = run_agent('sensor_network', base=SensorNetwork)

-----
NameError                                Traceback (most recent call last)
<ipython-input-4-75226932f9c3> in <module>
```

(continues on next page)

(continued from previous page)

```

----> 1 sensor_network = run_agent('sensor_network', base=SensorNetwork)

NameError: name 'SensorNetwork' is not defined

```

5.3 Sensor Agents

1. Next, we create a Sensor Agent by `sensor_network.addsimsensor(type,unit)`, and store into a list `sensors`.
2. We set the data source of the Sensor Agent: `sensor_new.set_generatorDataSet(dataSet)` where `dataSet` is a 3-dimensional numpy array with: [Row x Sequence Length x Sensor]

```

[5]: #add sensors
sensors=[]
sensorTypes = ["Temperature","Temperature","Temperature","Temperature","Vibration",
↳"EfficiencyFactor","VolumeFlow","VolumeFlow","Pressure","Pressure","Pressure",
↳"Pressure","Pressure","Pressure","MotorPower","CoolingEfficiency","CoolingPower"]
sensorUnits = ["C","C","C","C","mm/s","%", "l/min","l/min","bar","bar","bar","bar",
↳"bar","bar","W","%", "kW"]

for sensor_num in range(X_data.shape[2]):
    sensor_new = sensor_network.add_simsensor(type=sensorTypes[sensor_num], unit_
↳v=sensorUnits[sensor_num])
    sensor_new.set_generatorDataSet(X_data[:, :, sensor_num])
    sensors.append(sensor_new)

-----
NameError                                Traceback (most recent call last)
<ipython-input-5-ada498f69a90> in <module>
      4 sensorUnits = ["C","C","C","C","mm/s","%", "l/min","l/min","bar","bar","bar",
↳"bar","bar","bar","W","%", "kW"]
      5
----> 6 for sensor_num in range(X_data.shape[2]):
      7     sensor_new = sensor_network.add_simsensor(type=sensorTypes[sensor_num],
↳unit_v=sensorUnits[sensor_num])
      8     sensor_new.set_generatorDataSet(X_data[:, :, sensor_num])

NameError: name 'X_data' is not defined

```

1. We can access the Sensor Agents stored in array `sensors`.
2. Alternatively, the SensorNetwork Agent automatically keeps track of sensors added by it, we can access the list by calling `get_attr('sensor_list')`
3. Here, we demonstrate a function of Sensor Agent which is `read_generator` which returns a random data row from the loaded dataset

```

[6]: #the sensors are loaded into array sensors
sensor1 = sensors[0]
print(len(sensors))

#access sensors by either way
sensor_network.get_attr('sensor_list')[0].read_generator()
sensor1.read_generator()

```

```

-----
IndexError                                Traceback (most recent call last)
<ipython-input-6-fe35f12d5c0e> in <module>
      1 #the sensors are loaded into array sensors
----> 2 sensor1 = sensors[0]
      3 print(len(sensors))
      4
      5 #access sensors by either way

IndexError: list index out of range

```

5.4 Aggregator Agents

1. We add an Aggregator Agent to the sensor_network by calling the function `.add_aggregator(sensorList)` where `sensorList` is an optional list of Sensor Agents which automatically binds the aggregator to the Sensor Agents.
2. Aggregator Agent can bind to Sensor Agent in runtime by calling `.bind_sensors(sensorList)`.

```
[7]: #add aggregators and bind them to sensors
aggregator1 = sensor_network.add_aggregator(sensors)
```

```

-----
NameError                                Traceback (most recent call last)
<ipython-input-7-f1ff1ca81bf3> in <module>
      1 #add aggregators and bind them to sensors
----> 2 aggregator1 = sensor_network.add_aggregator(sensors)

NameError: name 'sensor_network' is not defined

```

5.5 Predictor Agents

1. Similarly, we can add Predictor Agent by `.add_predictor(aggregator)` with the optional aggregator to be binded to.
2. For each Predictor Agent, we load the prediction model by `.load_predictor_model(model)` where `model` is a trained `ML_Wrapper` with signature such as `.predict_model_wUnc(x_test, num_samples)` where `x_test` is the data input and `num_samples` is the number of samples for Monte Carlo sampling.
3. Here, we load the previously pickled prediction model.

```
[8]: #add predictor and bind to aggregator
predictor1 = sensor_network.add_predictor(aggregator=aggregator1)
predictor2 = sensor_network.add_predictor(aggregator=aggregator1)
predictor3 = sensor_network.add_predictor(aggregator=aggregator1)
predictor4 = sensor_network.add_predictor(aggregator=aggregator1)
predictor5 = sensor_network.add_predictor(aggregator=aggregator1)

#load predictor models
predictor1.load_predictor_model(pickle.load(open("pickles/" + "bnn_wrapper_0.p",
↪ "rb")))

```

(continues on next page)

(continued from previous page)

```

predictor2.load_predictor_model(pickle.load(open("pickles/" + "bnn_wrapper_1.p",
↵"rb"))))
predictor3.load_predictor_model(pickle.load(open("pickles/" + "bnn_wrapper_2.p",
↵"rb"))))
predictor4.load_predictor_model(pickle.load(open("pickles/" + "bnn_wrapper_3.p",
↵"rb"))))
predictor5.load_predictor_model(pickle.load(open("pickles/" + "bnn_wrapper_4.p",
↵"rb"))))

```

```

-----
NameError                                Traceback (most recent call last)
<ipython-input-8-cfeb5649323c> in <module>
      1 #add predictor and bind to aggregator
----> 2 predictor1 = sensor_network.add_predictor(agggregator=agggregator1)
      3 predictor2 = sensor_network.add_predictor(agggregator=agggregator1)
      4 predictor3 = sensor_network.add_predictor(agggregator=agggregator1)
      5 predictor4 = sensor_network.add_predictor(agggregator=agggregator1)

NameError: name 'sensor_network' is not defined

```

5.6 DecisionMaker Agent

1. We add Decision Maker Agent calling `.add_decisionMaker()` on SensorNetwork agent
2. The DM Agent is binded to every predictor by calling `.bind_predictor(predictor)` function

```

[9]: decisionMaker = sensor_network.add_decisionMaker()
decisionMaker.bind_predictor(predictor1)
decisionMaker.bind_predictor(predictor2)
decisionMaker.bind_predictor(predictor3)
decisionMaker.bind_predictor(predictor4)
decisionMaker.bind_predictor(predictor5)

```

```

-----
NameError                                Traceback (most recent call last)
<ipython-input-9-180d2cc1fe2f> in <module>
----> 1 decisionMaker = sensor_network.add_decisionMaker()
      2 decisionMaker.bind_predictor(predictor1)
      3 decisionMaker.bind_predictor(predictor2)
      4 decisionMaker.bind_predictor(predictor3)
      5 decisionMaker.bind_predictor(predictor4)

NameError: name 'sensor_network' is not defined

```

5.7 Demo

1. For demo, we run an infinite loop which continuously runs the `.request_sensors_data()`
2. Due to the bindings, the requested data will immediately be propagated to all binded Predictor Agents and to Decision Maker Agent
3. While this is running, run the dashboard code in a separate terminal to visualize the multi-agent testbed

```
[10]: #send request to aggregator agents for data from sensors
```

```
if DemoMode:
    for i in range(9999999999):
        aggregator1.request_sensors_data()
        time.sleep(3)
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-10-c6d4a27820dd> in <module>
      3 if DemoMode:
      4     for i in range(9999999999):
----> 5         aggregator1.request_sensors_data()
      6         time.sleep(3)

NameError: name 'aggregator1' is not defined
```

```
[ ]:
```

Agent Dashboard User Interface

1. Run this code after the Agents are setup and running
2. View the web visualization at port 8054 using Internet Browser

```
[1]: portNumber = 8054

[2]: import osbrain
from osbrain.agent import run_agent
from osbrain import NSProxy

# -*- coding: utf-8 -*-
import dash
import dash_core_components as dcc
import dash_html_components as html
import dash_cytoscape as cyto
import dash_daq as daq
import plotly.graph_objs as go
import networkx as nx
import numpy as np

import pickle
external_css = [
    ↪ "https://cdnjs.cloudflare.com/ajax/libs/skeleton/2.0.4/skeleton.min.css",
    ↪ "https://fonts.googleapis.com/css?family=Raleway:400,400i,700,700i",
    ↪ "https://fonts.googleapis.com/css?family=Product+Sans:400,400i,700,700i"]
app = dash.Dash(__name__, external_stylesheets=external_css)

#=====APP LAYOUT=====
agent_names = ['aggregator_1', 'sensor_0', 'sensor_1', 'sensor_2', 'sensor_3',
    ↪ 'sensor_4', 'sensor_5', 'sensor_6', 'sensor_7', 'sensor_8', 'sensor_9', 'sensor_10',
    ↪ 'sensor_11', 'sensor_12', 'sensor_13', 'sensor_14', 'predictor_0',
    ↪ 'decisionMaker_0']
```

(continues on next page)

(continued from previous page)

```

G = nx.Graph()
G.add_nodes_from(agent_names)
myEdges = []

for agent_x in agent_names:
    for agent_y in agent_names:
        include=False
        if 'sensor' in agent_x and 'aggregator' in agent_y:
            include = True
        elif 'aggregator' in agent_x and 'predictor' in agent_y:
            include = True
        elif 'predictor' in agent_x and 'decisionMaker' in agent_y:
            include = True
        if include:
            new_edge = (agent_x, agent_y)
            myEdges.append(new_edge)

G.add_edges_from(myEdges)
pos=nx.fruchterman_reingold_layout(G)

nodes_ct = [{'data': {'id': k, 'label': k}, 'position': {'x': pos[k][0], 'y': pos[k][1]}, 'classes': k.split('_')[0]} for k in agent_names]

edges_ct = [{'data': {'source': k[0], 'target': k[1]}} for k in myEdges]
elements = nodes_ct+edges_ct

tab_div_style = {
    "padding": "2",
    "marginLeft": "5",
    "marginRight": "5",
    "backgroundColor": "white",
    "border": "1px solid #C8D4E3",
    "borderRadius": "3px"
}
tab_title = { 'textAlign': 'center', }

output_labels = [{0: "Optimal", 1: "Reduced", 2: "Nearly Fail"},
                 {0: "Optimal", 1: "Small lag", 2: "Severe lag", 3: "Nearly Fail"},
                 {0: "No Leakage", 1: "Weak Leakage", 2: "Severe Leakage"},
                 {0: "Optimal", 1: "Slightly Reduced", 2: "Severely Reduced", 3: "Nearly Fail"},
                 {0: "Stable", 1: "Unstable"}]

output_category = ["Cooler Condition", "Valve Condition", "Internal Pump",
                  ↪ "Accumulator", "Stable Flag"]

def getConditionIndicator(condition_text="Optimal", certain=True):
    color = "#00cc96" if certain else "#ff0000"
    label = "Certain" if certain else "Uncertain"
    return [
        html.H5(condition_text, style=tab_title),
        daq.Indicator(
            value=True,
            color=color,
            label=label,
            style=tab_title
        )
    ]

```

(continues on next page)

(continued from previous page)

```

app.layout = html.Div(children=[
html.Div([
    html.H3("Multi Agents for Machine Learning under Uncertainty Testbed", style={
        'textAlign': 'center',

    }),
    ],
    html.Div([
        html.Div([
            html.H6("Cooler Condition", style= { 'textAlign': 'center', "border":_
↪ "1px solid #C8D4E3"}),
            html.Div(children=getConditionIndicator("Optimal", certain=True),
↪ id='cooler-indicator')
        ], className="two columns", style=tab_div_style),
        html.Div([
            html.H6("Valve Condition", style={'textAlign': 'center', "border":_
↪ "1px solid #C8D4E3"}),
            html.Div(children=getConditionIndicator("Optimal", certain=True),
↪ id='valve-indicator')
        ], className="two columns", style=tab_div_style),
        html.Div([
            html.H6("Internal Pump", style={'textAlign': 'center', "border":_
↪ "1px solid #C8D4E3"}),
            html.Div(children=getConditionIndicator("Optimal", certain=True),
↪ id='pump-indicator')
        ], className="two columns", style=tab_div_style),
        html.Div([
            html.H6("Accumulator", style={'textAlign': 'center', "border":_
↪ "1px solid #C8D4E3"}),
            html.Div(children=getConditionIndicator("Optimal", certain=True),
↪ id='accumulator-indicator')
        ], className="two columns", style=tab_div_style),
        html.Div([
            html.H6("Stable Flag", style={'textAlign': 'center', "border":_
↪ "1px solid #C8D4E3"}),
            html.Div(children=getConditionIndicator("Optimal", certain=True),
↪ id='stability-indicator')
        ], className="two columns", style=tab_div_style),
    ], className="row"),

    html.Div([
        html.Div([
            html.H5('Agent Network Dashboard', style=tab_title
        ),
        cyto.Cytoscape(
            id='agents-network',
            layout={'name': 'circle'},
            style={'width': '100%', 'height': '400px'},
            elements=elements,
            stylesheet= [
                {
                    'selector': 'node',
                    'style': {
                        'label': 'data(id)'
                    }
                },
            ],
        )
    ])
])

```

(continues on next page)

(continued from previous page)

```

        'selector': '.sensor',
        'style': {
            'background-color': 'green',
            'line-color': 'black'
        }
    },
    {
        'selector': '.aggregator',
        'style': {
            'background-color': 'blue',
            'line-color': 'black'
        }
    },
    {
        'selector': '.predictor',
        'style': {
            'background-color': 'red',
            'line-color': 'black'
        }
    },
    {
        'selector': '.decisionMaker',
        'style': {
            'background-color': 'yellow',
            'line-color': 'black'
        }
    }
],
], className="six columns", style=tab_div_style),
html.Div([
    html.H5(
        children='Select Predictor Agent',
        style={
            'text-align': 'left',
        }
    ),
    html.Div(
        id='predictor-dropdown-div',
        children=dcc.Dropdown(
            id='predictor-dropdown',
            options=[],
            value='predictor_0',
            style={'width': 250},
        ),
        html.Div([
            html.H5("Prediction Graph", style=tab_title)
        ]),
        dcc.Graph(id='prediction-graph'),
    ], className='six columns ', style=tab_div_style)
], className="row"),

html.Div([

    html.Div([
        html.Div([
            html.H5("Sensor Graph", style=tab_title)
        ]),

```

(continues on next page)

(continued from previous page)

```

        html.H5(
            children='Select Sensor Agent',
            style={
                'textAlign': 'left',
            }
        ),
        html.Div(
            id='sensor-dropdown-div',
            children=dcc.Dropdown(
                id='sensor-dropdown',
                options=[],
                value='sensor_number_0',
                style={'width': 250},
            ),
            dcc.Graph(id='sensor-graph'),
        ], className='six columns ', style=tab_div_style),
        html.Div([
            html.Div([
                html.H5("Uncertainty Graph", style=tab_title)
            ]),
            dcc.Graph(id='uncertainty-graph'),
        ], className='six columns ', style=tab_div_style),
    ], className='row'),

    dcc.Interval(
        id='interval-component',
        interval=3 * 1000, # in milliseconds
        n_intervals=0
    ),
    dcc.Interval(
        id='interval-component-network-graph',
        interval=1000 * 1000, # in milliseconds
        n_intervals=0
    )

], style={
    "padding": "8",
    "marginLeft": "45",
    "marginRight": "45",
    "backgroundColor": "white",
    "border": "1px solid #C8D4E3",
    "borderRadius": "3px"
})

@app.callback(dash.dependencies.Output('sensor-dropdown', 'value'),
              [dash.dependencies.Input('agents-network', 'tapNodeData')])
def displayTapNodeData(data):
    if 'sensor' in data['label'] and 'sensor_network' not in data['label']:
        return data['label']

@app.callback([dash.dependencies.Output('agents-network', 'elements'), dash.
    ↪ dependencies.Output('sensor-dropdown-div', 'children'), dash.dependencies.
    ↪ Output('predictor-dropdown-div', 'children')],
              [dash.dependencies.Input('interval-component-network-graph',
    ↪ 'n_intervals')])
def update_network_graph(n):

```

(continues on next page)

(continued from previous page)

```

ns_temp = NSProxy(nsaddr='127.0.0.1:14065')
agent_names = ns_temp.agents()
print(agent_names)
G = nx.Graph()
G.add_nodes_from(agent_names)
myEdges = []

for agent_x in agent_names:
    for agent_y in agent_names:
        include = False
        if 'sensor' in agent_x and 'aggregator' in agent_y and 'sensor_network'
↪not in agent_x:
            include = True
        elif 'aggregator' in agent_x and 'predictor' in agent_y:
            include = True
        elif 'predictor' in agent_x and 'decisionMaker' in agent_y:
            include = True
        if include:
            new_edge = (agent_x, agent_y)
            myEdges.append(new_edge)

G.add_edges_from(myEdges)
pos = nx.fruchterman_reingold_layout(G)

nodes_ct = [{'data': {'id': k, 'label': k}, 'position': {'x': pos[k][0], 'y':
↪pos[k][1]}, 'classes': k.split('_')[0]} for k in agent_names]
edges_ct = [{'data': {'source': k[0], 'target': k[1]}} for k in myEdges]
elements = nodes_ct + edges_ct

sensor_options = [{'label': name, 'value': name} for name in agent_names if
↪'sensor' in name and 'sensor_network' not in name]
predictor_options = [{'label': name, 'value': name} for name in agent_names if
↪'predictor' in name ]

sensor_dropdown_component = dcc.Dropdown(
    id='sensor-dropdown',
    options=sensor_options,
    value=sensor_options[0]['value'],
    style={'width': 250},
)
predictor_dropdown_component = dcc.Dropdown(
    id='predictor-dropdown',
    options=predictor_options,
    value=predictor_options[0]['value'],
    style={'width': 250},
)
print("HELLO")
return [elements, sensor_dropdown_component, predictor_dropdown_component]

@app.callback(dash.dependencies.Output('sensor-graph', 'figure'),
              [dash.dependencies.Input('interval-component', 'n_intervals'),dash.
↪dependencies.Input('sensor-dropdown', 'value')])
def update_sensor_graph(n, chosen_sensor_name):
    ns_temp = NSProxy(nsaddr='127.0.0.1:14065')
    sensor_type = ns_temp.proxy(chosen_sensor_name).get_attr('type')
    sensor_unit = ns_temp.proxy(chosen_sensor_name).get_attr('unit_v')

```

(continues on next page)

(continued from previous page)

```

final_data = ns_temp.proxy(chosen_sensor_name).get_attr('current_data')

final_data = np.array(final_data)

y_data = final_data
N_sequence = y_data.shape[0]
x_data = np.linspace(0, N_sequence - 1, N_sequence)

traces_sensor = go.Scatter(
    x=x_data,
    y=y_data,
    mode='lines',
    name='lines'
)
layout = {'title': 'Sensor '+sensor_type+" #" +chosen_sensor_name.split('_')[-1],
          'xaxis': {'title': 'Time (s)'},
          'yaxis': {'title': sensor_type+" (" +sensor_unit+" )"},
          }

return {
    'data': [traces_sensor],
    'layout': layout
}

predictions = []
uncertainties = []
probabilities_accurate = []

def getTimeSeriesGraph(y_data,title='Prediction Certainty vs Time',xaxis='Time (s)',
    ↪yaxis='Certainty (%) '):
    N_sequence = y_data.shape[0]
    x_data = np.linspace(0, N_sequence - 1, N_sequence)

    traces_sensor = go.Scatter(
        x=x_data,
        y=y_data,
        mode='lines',
        name='lines'
    )
    layout = {'title': title,
              'xaxis': {'title': xaxis},
              'yaxis': {'title': yaxis},
              }

    return {
        'data': [traces_sensor],
        'layout': layout
    }

@app.callback([dash.dependencies.Output('prediction-graph', 'figure'), dash.
    ↪dependencies.Output('uncertainty-graph', 'figure'), dash.dependencies.
    ↪Output('cooler-indicator', 'children'), dash.dependencies.Output('valve-indicator',
    ↪'children'), dash.dependencies.Output('pump-indicator', 'children'), dash.
    ↪dependencies.Output('accumulator-indicator', 'children'), dash.dependencies.
    ↪Output('stability-indicator', 'children')],
    [dash.dependencies.Input('interval-component', 'n_intervals'),dash.
    ↪dependencies.Input('predictor-dropdown', 'value')])

```

(continues on next page)

(continued from previous page)

```

def update_prediction_graph(n, selected_predictor):
    ns_temp = NSProxy(nsaddr='127.0.0.1:14065')
    overall_new_data = ns_temp.proxy('decisionMaker_0').get_attr('current_inference')
    column_numbers = [x.split('_')[-1] for x in overall_new_data.columns]
    overall_new_data.columns = column_numbers
    overall_new_data.sort_index(axis=1, inplace=True)
    #condition_indicators = [getConditionIndicator(output_labels[int(predictor['label'
    ↪'].split('_')[-1]))[overall_new_data[predictor['label']].pred], certain=overall_new_
    ↪data[predictor['label']].unc_state) for id_, predictor in enumerate(predictors) ]
    print(overall_new_data)
    print(overall_new_data.loc['pred'])

    new_prediction = overall_new_data.loc['pred']
    new_uncertainty = overall_new_data.loc['unc']
    new_uncertainty_state = overall_new_data.loc['unc_state']

    condition_indicators = [getConditionIndicator(output_labels[_id][new_prediction[_
    ↪id]], certain=unc_state) for _id, unc_state in enumerate(new_uncertainty_state)]

    predictions.append(new_prediction)
    uncertainties.append(new_uncertainty)
    selected_predictor_id=int(selected_predictor.split("_")[-1])
    # dim 0 = samples, dim 1 = predictor
    y_data_pred = np.array(predictions)[: ,selected_predictor_id]
    y_data_unc = np.array(uncertainties)[: , selected_predictor_id]

    #pred
    prediction_graph = getTimeSeriesGraph(y_data_pred, 'Model Prediction ('+output_
    ↪category[selected_predictor_id]+') vs Time', 'Time (s)', 'Prediction ')
    uncertainties_graph = getTimeSeriesGraph(y_data_unc,
    ↪'Prediction Certainty vs Time', 'Time (s)', 'Certainty (%) ')

    return [prediction_graph, uncertainties_graph] +condition_indicators

if __name__ == '__main__':

    app.run_server(debug=False, port=portNumber)

```

```

-----
ModuleNotFoundError                                Traceback (most recent call last)
<ipython-input-2-3d73c4fa70ad> in <module>()
      7 import dash_core_components as dcc
      8 import dash_html_components as html
----> 9 import dash_cytoscape as cyto
     10 import dash_daq as daq
     11 import plotly.graph_objs as go

ModuleNotFoundError: No module named 'dash_cytoscape'

```

agentMET4FOF agents

```
class agentMET4FOF.agents.AgentMET4FOF (name="", host=None, serializer=None, trans-  
port=None, attributes=None)
```

Base class for all agents with specific functions to be overridden/supplied by user.

Behavioural functions for users to provide are `init_parameters`, `agent_loop` and `on_received_message`. Communicative functions are `bind_output`, `unbind_output` and `send_output`.

agent_loop ()

User defined method for the agent to execute for *loop_wait* seconds specified either in *self.loop_wait* or explicitly via `'init_agent_loop(loop_wait)'`

To start a new loop, call `init_agent_loop(loop_wait)` on the agent Example of usage is to check the *current_state* of the agent and send data periodically

before_loop ()

This action is executed before initiating the loop

bind_output (*output_agent*)

Forms Output connection with another agent. Any call on `send_output` will reach this newly binded agent

Adds the agent to its list of Outputs.

Parameters `output_agent` (`AgentMET4FOF` or *list*) – Agent(s) to be binded to this agent's output channel

convert_to_plotly (*matplotlib_fig*)

Internal method to convert matplotlib figure to plotly figure

Parameters `matplotlib_fig` (*plt.Figure*) – Matplotlib figure to be converted

handle_process_data (*message*)

Internal method to handle incoming message before calling user-defined `on_received_message` method.

If *current_state* is either Stop or Reset, it will terminate early before entering `on_received_message`

init_agent_loop (*loop_wait=1.0*)

Initiates the agent loop, which iterates every `'loop_wait'` seconds

Stops every timers and initiate a new loop.

Parameters `loop_wait (int)` – The wait between each iteration of the loop

init_parameters ()

User provided function to initialize parameters of choice.

log_info (message)

Prints logs to be saved into logfile with Logger Agent

Parameters `message (str)` – Message to be logged to the internal Logger Agent

on_init ()

Internal initialization to setup the agent: mainly on setting the dictionary of Inputs, Outputs, PubAddr.

Calls user-defined `init_parameters()` upon finishing.

Inputs

Dictionary of Agents connected to its input channels. Messages will arrive from agents in this dictionary. Automatically updated when `bind_output()` function is called

Type dict

Outputs

Dictionary of Agents connected to its output channels. Messages will be sent to agents in this dictionary. Automatically updated when `bind_output()` function is called

Type dict

PubAddr_alias

Name of Publish address socket

Type str

PubAddr

Publish address socket handle

Type str

AgentType

Name of class

Type str

current_state

Current state of agent. Can be used to define different states of operation such as “Running”, “Idle”, “Stop”, etc.. Users will need to define their own flow of handling each type of `self.current_state` in the `agent_loop`

Type str

loop_wait

The interval to wait between loop. Call `init_agent_loop` to restart the timer or set the value of `loop_wait` in `init_parameters` when necessary.

Type int

memory_buffer_size

The total number of elements to be stored in the agent *memory* When total elements exceeds this number, the latest elements will be replaced with the incoming data elements

Type int

on_received_message (message)

User-defined method and is triggered to handle the message passed by Input.

Parameters `message (Dictionary)` – The message received is in form
{‘from’:agent_name, ‘data’: data, ‘senderType’: agent_class, ‘channel’:channel_name}
agent_name is the name of the Input agent which sent the message data is the actual content of the message

pack_data (*data*, *channel*='default')

Internal method to pack the data content into a dictionary before sending out.

Special case : if the *data* is already a *message*, then the *from* and *senderType* will be altered to this agent, without altering the *data* and *channel* within the message this is used for more succinct data processing and passing.

Parameters

- **data** (*argument*) – Data content to be packed before sending out to agents.
- **channel** (*str*) – Key of dictionary which stores data

Returns Packed message data

Return type dict of the form {'from':agent_name, 'data': data, 'senderType': agent_class, 'channel':channel_name}.

reset ()

This method will be called on all agents when the global *reset_agents* is called by the AgentNetwork and when the Reset button is clicked on the dashboard.

Method to reset the agent's states and parameters. User can override this method to reset the specific parameters.

send_output (*data*, *channel*='default')

Sends message data to all connected agents in self.Outputs.

Output connection can first be formed by calling *bind_output*. By default calls *pack_data*(data) before sending out. Can specify specific channel as opposed to 'default' channel.

Parameters

- **data** (*argument*) – Data content to be sent out
- **channel** (*str*) – Key of *message* dictionary which stores data

Returns message

Return type dict of the form {'from':agent_name, 'data': data, 'senderType': agent_class, 'channel':channel_name}.

send_plot (*fig*=<Figure size 640x480 with 0 Axes>)

Sends plot to agents connected to this agent's Output channel.

This method is different from *send_output* which will be sent to through the 'plot' channel to be handled.

Parameters **fig** (*Figure*) – Can be either matplotlib figure or plotly figure

Returns The message format is {'from'

Return type agent_name, 'plot': data, 'senderType': agent_class}.

stop_agent_loop ()

Stops agent_loop from running. Note that the agent will still be responding to messages

unbind_output (*output_agent*)

Remove existing output connection with another agent. This reverses the *bind_output* method

Parameters **output_agent** (*AgentMET4FOF*) – Agent binded to this agent's output channel

update_data_memory (*message*)

Updates data stored in *self.memory* with the received message

Checks if sender agent has sent any message before If it did,then append, otherwise create new entry for it

Parameters `message` (*dict*) – Standard message format specified by AgentMET4FOF class

```
class agentMET4FOF.agents.AgentNetwork (ip_addr='127.0.0.1', port=3333, connect=False, log_filename='log_file.csv', dashboard_modules=True, dashboard_update_interval=3, dashboard_max_monitors=10, dashboard_port=8050)
```

Object for starting a new Agent Network or connect to an existing Agent Network specified by ip & port

Provides function to add agents, (un)bind agents, query agent network state, set global agent states Interfaces with an internal `_AgentController` which is hidden from user

```
add_agent (name=' ', agentType=<class 'agentMET4FOF.agents.AgentMET4FOF'>, log_mode=True, memory_buffer_size=1000000, ip_addr=None)
```

Instantiates a new agent in the network.

Parameters

- **str** (*name*) – with the same name. Defaults to the agent’s class name.
- **AgentMET4FOF** (*agentType*) – network. Defaults to `AgentMET4FOF`
- **bool** (*log_mode*) – Logger Agent. Defaults to `True`.

Returns `AgentMET4FOF`

Return type Newly instantiated agent

```
agents (filter_agent=None)
```

Returns all agent names connected to Agent Network.

Returns `list`

Return type names of all agents

```
bind_agents (source, target)
```

Binds two agents communication channel in a unidirectional manner from *source* Agent to *target* Agent

Any subsequent calls of *source.send_output()* will reach *target* Agent’s message queue.

Parameters

- **source** (`AgentMET4FOF`) – Source agent whose Output channel will be binded to *target*
- **target** (`AgentMET4FOF`) – Target agent whose Input channel will be binded to *source*

```
connect (ip_addr='127.0.0.1', port=3333, verbose=True)
```

Parameters

- **ip_addr** (*str*) – IP Address of server to connect to
- **port** (*int*) – Port of server to connect to

```
get_agent (agent_name)
```

Returns a particular agent connected to Agent Network.

Parameters `agent_name` (*str*) – Name of agent to search for in the network

```
set_agents_state (filter_agent=None, state='Idle')
```

Blanket operation on all agents to set their *current_state* attribute to given state

Can be used to define different states of operation such as “Running”, “Idle”, “Stop”, etc.. Users will need to define their own flow of handling each type of *self.current_state* in the *agent_loop*

Parameters

- **filter_agent** (*str*) – (Optional) Filter name of agents to set the states
- **state** (*str*) – State of agents to set

set_running_state (*filter_agent=None*)

Blanket operation on all agents to set their *current_state* attribute to “Running”

Users will need to define their own flow of handling each type of *self.current_state* in the *agent_loop*

Parameters **filter_agent** (*str*) – (Optional) Filter name of agents to set the states

set_stop_state (*filter_agent=None*)

Blanket operation on all agents to set their *current_state* attribute to “Stop”

Users will need to define their own flow of handling each type of *self.current_state* in the *agent_loop*

Parameters **filter_agent** (*str*) – (Optional) Filter name of agents to set the states

shutdown ()

Shutowns the entire agent network and all agents

start_server (*ip_addr='127.0.0.1', port=3333*)

Parameters

- **ip_addr** (*str*) – IP Address of server to start
- **port** (*int*) – Port of server to start

unbind_agents (*source, target*)

Unbinds two agents communication channel in a unidirectional manner from *source* Agent to *target* Agent

This is the reverse of *bind_agents()*

Parameters

- **source** (*AgentMET4FOF*) – Source agent whose Output channel will be unbinded from *target*
- **target** (*AgentMET4FOF*) – Target agent whose Input channel will be unbinded from *source*

class agentMET4FOF.agents.**DataStreamAgent** (*name="", host=None, serializer=None, transport=None, attributes=None*)

Able to simulate generation of datastream by loading a given DataStreamMET4FOF object.

Can be used in incremental training or batch training mode. To simulate batch training mode, set *pretrain_size=-1*, otherwise, set *pretrain_size* and *batch_size* for the respective See *DataStreamMET4FOF* on loading your own data set as a data stream.

agent_loop ()

User defined method for the agent to execute for *loop_wait* seconds specified either in *self.loop_wait* or explicitly via ‘*init_agent_loop(loop_wait)*’

To start a new loop, call *init_agent_loop(loop_wait)* on the agent Example of usage is to check the *current_state* of the agent and send data periodically

init_parameters (*stream=<agentMET4FOF.streams.DataStreamMET4FOF object>, pretrain_size=None, batch_size=1, loop_wait=1, randomize=False*)

Parameters

- **stream** (*DataStreamMET4FOF*) – A DataStreamMET4FOF object which provides the sample data
- **pretrain_size** (*int*) – The number of sample data to send through in the first loop cycle, and subsequently, the *batch_size* will be used

- **batch_size** (*int*) – The number of sample data to send in every loop cycle
- **loop_wait** (*int*) – The duration to wait (seconds) at the end of each loop cycle before going into the next cycle
- **randomize** (*bool*) – Determines if the dataset should be shuffled before streaming

reset ()

This method will be called on all agents when the global *reset_agents* is called by the AgentNetwork and when the Reset button is clicked on the dashboard.

Method to reset the agent's states and parameters. User can override this method to reset the specific parameters.

```
class agentMET4FOF.agents.MonitorAgent (name="", host=None, serializer=None, trans-  
port=None, attributes=None)
```

Unique Agent for storing plots and data from messages received from input agents.

The dashboard searches for Monitor Agents' *memory* and *plots* to draw the graphs "plot" channel is used to receive base64 images from agents to plot on dashboard

memory

Dictionary of format *{agent1_name : agent1_data, agent2_name : agent2_data}*

Type dict

plots

Dictionary of format *{agent1_name : agent1_plot, agent2_name : agent2_plot}*

Type dict

plot_filter

List of keys to filter the 'data' upon receiving message to be saved into memory Used to specifically select only a few keys to be plotted

Type list of str

```
init_parameters (plot_filter=[], custom_plot_function=-1, **kwargs)
```

User provided function to initialize parameters of choice.

```
on_received_message (message)
```

Handles incoming data from 'default' and 'plot' channels.

Stores 'default' data into *self.memory* and 'plot' data into *self.plots*

Parameters *message* (*dict*) – Acceptable channel values are 'default' or 'plot'

reset ()

This method will be called on all agents when the global *reset_agents* is called by the AgentNetwork and when the Reset button is clicked on the dashboard.

Method to reset the agent's states and parameters. User can override this method to reset the specific parameters.

```
update_plot_memory (message)
```

Updates plot figures stored in *self.plots* with the received message

Parameters *message* (*dict*) – Standard message format specified by AgentMET4FOF class
Message['data'] needs to be base64 image string and can be nested in dictionary for multiple plots Only the latest plot will be shown kept and does not keep a history of the plots.

```
class agentMET4FOF.agents.TransformerAgent (name="", host=None, serializer=None, trans-  
port=None, attributes=None)
```

init_parameters (*method=None, **kwargs*)

User provided function to initialize parameters of choice.

on_received_message (*message*)

User-defined method and is triggered to handle the message passed by Input.

Parameters message (*Dictionary*) – The message received is in form
{‘from’:agent_name, ‘data’: data, ‘senderType’: agent_class, ‘channel’:channel_name}
agent_name is the name of the Input agent which sent the message data is the actual content
of the message

agentMET4FOF streams

```
class agentMET4FOF.streams.CosineGenerator (num_cycles=1000)
```

```
class agentMET4FOF.streams.DataStreamMET4FOF
```

Class for creating finite datastream for ML with x as inputs and y as target Data can be fetched sequentially using *next_sample()* or all at once *all_samples()*

For sensors data: The format shape for 2D data stream (num_samples, n_sensors) The format shape for 3D data stream (num_samples, sample_length , n_sensors)

```
all_samples ()
```

Returns all the samples in the data stream

Returns samples

Return type dict of the form $\{ 'x': \text{current_sample_x}, 'y': \text{current_sample_y} \}$

```
next_sample (batch_size=1)
```

Fetches the samples from the data stream and advances the internal pointer *current_idx*

Parameters batch_size (*int*) – number of batches to get from data stream

Returns samples

Return type dict of the form $\{ 'x': \text{current_sample_x}, 'y': \text{current_sample_y} \}$

```
class agentMET4FOF.streams.SineGenerator (num_cycles=1000)
```

```
agentMET4FOF.streams.extract_x_y (message)
```

Extracts features & target from *message['data']* with expected structure such as :

1. tuple - (x,y)
2. dict - $\{ 'x': \text{x_data}, 'y': \text{y_data} \}$

Handle data structures of dictionary to extract features & target

CHAPTER 9

Indices and tables

- `genindex`
- `modindex`
- `search`

CHAPTER 10

References

Bibliography

- [Bang2019] Bang X. Yong, A. Brintrup Multi Agent System for Machine Learning Under Uncertainty in Cyber Physical Manufacturing System, 9th Workshop on Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future

a

`agentMET4FOF.agents`, [81](#)
`agentMET4FOF.streams`, [89](#)

A

`add_agent()` (*agentMET4FOF.agents.AgentNetwork* method), 84
`agent_loop()` (*agentMET4FOF.agents.AgentMET4FOF* method), 81
`agent_loop()` (*agentMET4FOF.agents.DataStreamAgent* method), 85
`AgentMET4FOF` (*class in agentMET4FOF.agents*), 81
`agentMET4FOF.agents` (*module*), 81
`agentMET4FOF.streams` (*module*), 89
`AgentNetwork` (*class in agentMET4FOF.agents*), 84
`agents()` (*agentMET4FOF.agents.AgentNetwork* method), 84
`AgentType` (*agentMET4FOF.agents.AgentMET4FOF* attribute), 82
`all_samples()` (*agentMET4FOF.streams.DataStreamMET4FOF* method), 89

B

`before_loop()` (*agentMET4FOF.agents.AgentMET4FOF* method), 81
`bind_agents()` (*agentMET4FOF.agents.AgentNetwork* method), 84
`bind_output()` (*agentMET4FOF.agents.AgentMET4FOF* method), 81

C

`connect()` (*agentMET4FOF.agents.AgentNetwork* method), 84
`convert_to_plotly()` (*agentMET4FOF.agents.AgentMET4FOF* method), 81
`CosineGenerator` (*class in agentMET4FOF.streams*), 89

`current_state` (*agentMET4FOF.agents.AgentMET4FOF* attribute), 82

D

`DataStreamAgent` (*class in agentMET4FOF.agents*), 85
`DataStreamMET4FOF` (*class in agentMET4FOF.streams*), 89

E

`extract_x_y()` (*in module agentMET4FOF.streams*), 89

G

`get_agent()` (*agentMET4FOF.agents.AgentNetwork* method), 84

H

`handle_process_data()` (*agentMET4FOF.agents.AgentMET4FOF* method), 81

I

`init_agent_loop()` (*agentMET4FOF.agents.AgentMET4FOF* method), 81
`init_parameters()` (*agentMET4FOF.agents.AgentMET4FOF* method), 82
`init_parameters()` (*agentMET4FOF.agents.DataStreamAgent* method), 85
`init_parameters()` (*agentMET4FOF.agents.MonitorAgent* method), 86
`init_parameters()` (*agentMET4FOF.agents.TransformerAgent* method), 86

Inputs (*agentMET4FOF.agents.AgentMET4FOF attribute*), 82

L

`log_info()` (*agentMET4FOF.agents.AgentMET4FOF method*), 82

`loop_wait` (*agentMET4FOF.agents.AgentMET4FOF attribute*), 82

M

`memory` (*agentMET4FOF.agents.MonitorAgent attribute*), 86

`memory_buffer_size` (*agentMET4FOF.agents.AgentMET4FOF attribute*), 82

`MonitorAgent` (*class in agentMET4FOF.agents*), 86

N

`next_sample()` (*agentMET4FOF.streams.DataStreamMET4FOF method*), 89

O

`on_init()` (*agentMET4FOF.agents.AgentMET4FOF method*), 82

`on_received_message()` (*agentMET4FOF.agents.AgentMET4FOF method*), 82

`on_received_message()` (*agentMET4FOF.agents.MonitorAgent method*), 86

`on_received_message()` (*agentMET4FOF.agents.TransformerAgent method*), 87

Outputs (*agentMET4FOF.agents.AgentMET4FOF attribute*), 82

P

`pack_data()` (*agentMET4FOF.agents.AgentMET4FOF method*), 82

`plot_filter` (*agentMET4FOF.agents.MonitorAgent attribute*), 86

`plots` (*agentMET4FOF.agents.MonitorAgent attribute*), 86

`PubAddr` (*agentMET4FOF.agents.AgentMET4FOF attribute*), 82

`PubAddr_alias` (*agentMET4FOF.agents.AgentMET4FOF attribute*), 82

R

`reset()` (*agentMET4FOF.agents.AgentMET4FOF method*), 83

`reset()` (*agentMET4FOF.agents.DataStreamAgent method*), 86

`reset()` (*agentMET4FOF.agents.MonitorAgent method*), 86

S

`send_output()` (*agentMET4FOF.agents.AgentMET4FOF method*), 83

`send_plot()` (*agentMET4FOF.agents.AgentMET4FOF method*), 83

`set_agents_state()` (*agentMET4FOF.agents.AgentNetwork method*), 84

`set_running_state()` (*agentMET4FOF.agents.AgentNetwork method*), 85

`set_stop_state()` (*agentMET4FOF.agents.AgentNetwork method*), 85

`shutdown()` (*agentMET4FOF.agents.AgentNetwork method*), 85

`SineGenerator` (*class in agentMET4FOF.streams*), 89

`start_server()` (*agentMET4FOF.agents.AgentNetwork method*), 85

`stop_agent_loop()` (*agentMET4FOF.agents.AgentMET4FOF method*), 83

T

`TransformerAgent` (*class in agentMET4FOF.agents*), 86

U

`unbind_agents()` (*agentMET4FOF.agents.AgentNetwork method*), 85

`unbind_output()` (*agentMET4FOF.agents.AgentMET4FOF method*), 83

`update_data_memory()` (*agentMET4FOF.agents.AgentMET4FOF method*), 83

`update_plot_memory()` (*agentMET4FOF.agents.MonitorAgent method*), 86