Phase_1

May 14, 2020

[9]:

```
import pandas as pd
import pylab as pl
import numpy as np
import scipy.optimize as opt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
%matplotlib inline
import matplotlib.pyplot as plt
```

[6]:

```
#!pip install ibm_watson wget
!wget -O italy_normal2.csv https://drive.google.com/open?
 ↪id=1CX6txmOBM4flZITxNCa_YVDdb76nFrGu
```

```
--2020-05-05 18:12:19--
https://drive.google.com/open?id=1CX6txmOBM4flZITxNCa_YVDdb76nFrGu
Resolving drive.google.com (drive.google.com)… 172.217.1.174,
2607:f8b0:400b:809::200e
Connecting to drive.google.com (drive.google.com)|172.217.1.174|:443…
connected.
HTTP request sent, awaiting response… 307 Temporary Redirect
Location: https://drive.google.com/file/d/1CX6txmOBM4flZITxNCa_YVDdb76nFrGu/view
?usp=drive_open [following]
--2020-05-05 18:12:20--  https://drive.google.com/file/d/1CX6txmOBM4flZITxNCa_YV
Ddb76nFrGu/view?usp=drive_open
Reusing existing connection to drive.google.com:443.
HTTP request sent, awaiting response… 200 OK
Length: unspecified [text/html]
Saving to: ‘italy_normal2.csv’

italy_normal2.csv       [ <=>                ]  68.79K  --.-KB/s    in 0.02s

2020-05-05 18:12:21 (4.20 MB/s) - ‘italy_normal2.csv’ saved [70445]
```

[10]:

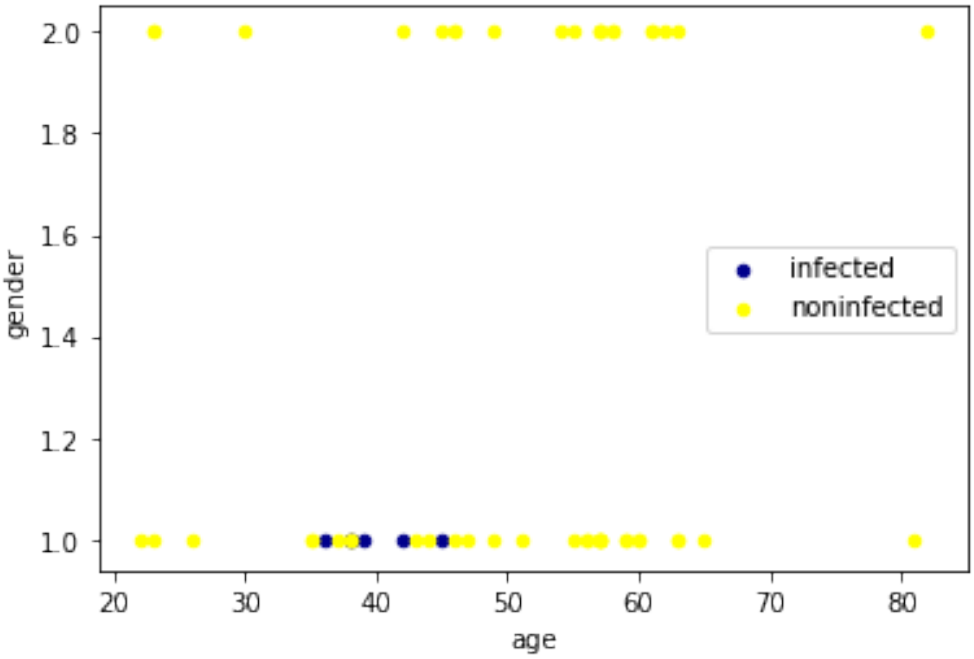
```
covid19 = pd.read_csv("italy_normal2.csv")
covid19.head()
```

[10]:

```
   age  gender  employmentstatus  tested  Construction  Deliveringtohomes  \
0   57       2                 1     4.0             0                  0
1   63       2                 1     4.0             0                  0
2   57       2                 1     4.0             0                  0
3   23       2                 1     4.0             0                  0
4   60       1                 1     4.0             0                  0

   Foodretail  Healthcare  Logisticsothertransportation  Manufacturing  …  \
0           0           0                             0              0  …
1           0           0                             0              0  …
2           0           0                             0              0  …
3           0           0                             0              0  …
4           0           0                             0              0  …

   Chronicobstructivepulmonary disease(COPD)  Diabetes  Epilepsy  \
0                                        0.0       0.0       0.0
1                                        0.0       0.0       0.0
2                                        0.0       0.0       0.0
3                                        0.0       0.0       0.0
4                                        0.0       0.0       0.0

   Heartdisease  Highbloodpressure  Highcholesterol  HIVAids  \
0           0.0                0.0              0.0      0.0
1           0.0                0.0              0.0      0.0
2           0.0                0.0              0.0      0.0
3           0.0                0.0              0.0      0.0
4           0.0                0.0              1.0      0.0

   Mentalhealthcondition  MultipleSclerosis  another
0                    0.0                0.0      1.0
1                    0.0                0.0      0.0
2                    0.0                0.0      1.0
3                    0.0                0.0      1.0
4                    0.0                0.0      0.0

[5 rows x 29 columns]
```

[11]:

```
ax = covid19[covid19['tested'] == 1][0:50].plot(kind='scatter', x='age',␣
 ↪y='gender', color='DarkBlue', label='infected');
covid19[covid19['tested'] == 4][0:50].plot(kind='scatter', x='age', y='gender',␣
 ↪color='Yellow', label='noninfected', ax=ax);
plt.show()
```



[13]:
```
#to fix the data from any nan or infinte value (very important pre process lone)
covid = covid19.replace(r'^\s*$', np.NaN, regex=True)
covid = covid.fillna(0)
np.any(np.isnan(covid))
np.all(np.isfinite(covid))
covid.dtypes
```

[13]:
```
age                                                  int64
gender                                               int64
employmentstatus                                     int64
tested                                             float64
Construction                                         int64
Deliveringtohomes                                    int64
Foodretail                                           int64
Healthcare                                           int64
Logisticsothertransportation                         int64
Manufacturing                                        int64
Policingorprisons                                    int64
Publictransport                                      int64
School                                               int64
Socialcare                                           int64
notworking                                           int64
Arthritis                                          float64
Asthma                                             float64
Cancer                                             float64
```

```
Cysticfibrosis                                 float64
Chronicobstructivepulmonary disease(COPD)     float64
Diabetes                                       float64
Epilepsy                                       float64
Heartdisease                                   float64
Highbloodpressure                              float64
Highcholesterol                                float64
HIVAids                                        float64
Mentalhealthcondition                          float64
MultipleSclerosis                              float64
another                                        float64
dtype: object
```

```
[127]: feature_df = covid[['age', 'gender', 'employmentstatus', 'tested',␣
        ↪'Construction',
              'Deliveringtohomes', 'Foodretail', 'Healthcare',
              'Logisticsothertransportation', 'Manufacturing', 'Policingorprisons',
              'Publictransport', 'School', 'Socialcare', 'notworking', 'Arthritis',
              'Asthma', 'Cancer', 'Cysticfibrosis',
              'Chronicobstructivepulmonary disease(COPD)', 'Diabetes', 'Epilepsy',
              'Heartdisease', 'Highbloodpressure', 'Highcholesterol', 'HIVAids',
              'Mentalhealthcondition', 'MultipleSclerosis', 'another']]
       X = np.asarray(feature_df)
       X[0:5]
```

```
[127]: array([[57.,  2.,  1.,  4.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                0.,  0.,  1.],
              [63.,  2.,  1.,  4.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                0.,  0.,  0.],
              [57.,  2.,  1.,  4.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                0.,  0.,  1.],
              [23.,  2.,  1.,  4.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                0.,  0.,  1.],
              [60.,  1.,  1.,  4.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,
                0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,
                0.,  0.,  0.]])
```

```
[128]: y = np.asarray(covid['tested'])
       y [0:100]
```

```
[128]: array([4, 4, 4, 4, 4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
              4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
              4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 2, 4, 4, 4, 4, 4, 4, 4])
```

[129]:

```
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2,␣
 ↪random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (1672, 29) (1672,)
Test set: (419, 29) (419,)
```

[130]:

```
from sklearn import svm
clf = svm.SVC(kernel='rbf')
clf.fit(X_train, y_train)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/sklearn/svm/base.py:196: FutureWarning: The default value of gamma will
change from 'auto' to 'scale' in version 0.22 to account better for unscaled
features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

[130]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

[131]:

```
yhat = clf.predict(X_test)
yhat [0:5]
```

[131]:

```
array([4, 4, 4, 4, 4])
```

[132]:

```
from sklearn.metrics import classification_report, confusion_matrix
import itertools
```

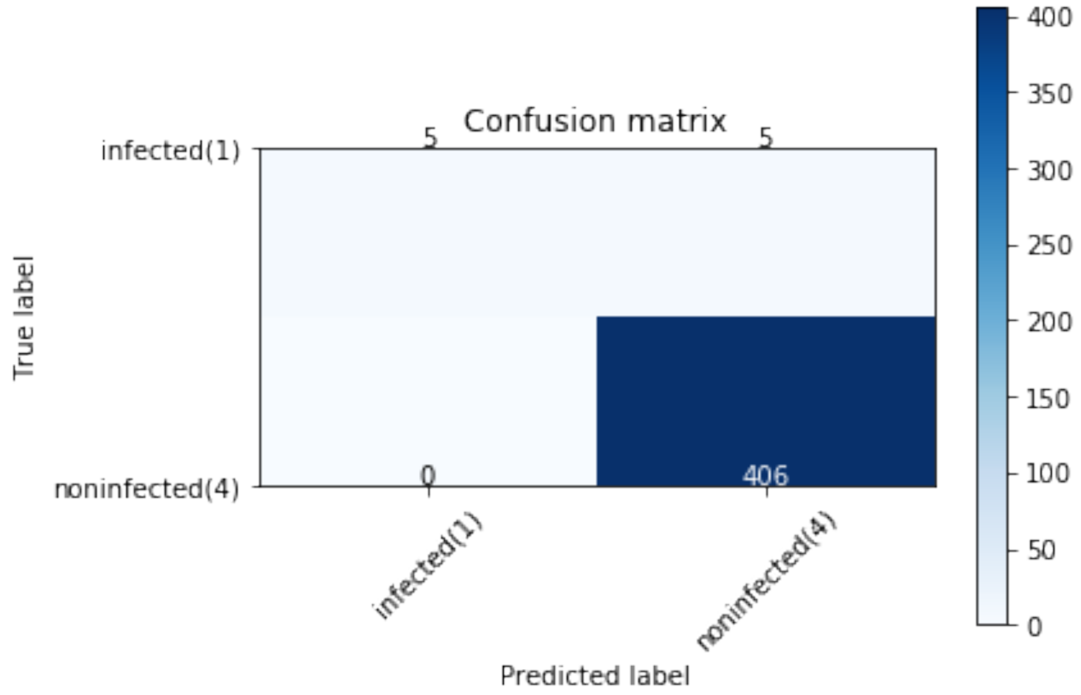
[133]:

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

[135]:

```
# Compute confusion matrix
cnf_matrix = confusion_matrix(y_test, yhat, labels=[2,4])
np.set_printoptions(precision=2)

print (classification_report(y_test, yhat))

# Plot non-normalized confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix,␣
 ↪classes=['infected(1)','noninfected(4)'],normalize= False,  title='Confusion␣
 ↪matrix')
```

```
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         1
           2       0.83      0.50      0.62        10
           3       0.00      0.00      0.00         2
           4       0.98      1.00      0.99       406

   micro avg       0.98      0.98      0.98       419
   macro avg       0.45      0.38      0.40       419
weighted avg       0.97      0.98      0.98       419

Confusion matrix, without normalization
[[  5   5]
 [  0 406]]
```



```
[136]: #f1_score from sklearn library
       from sklearn.metrics import f1_score
       f1_score(y_test, yhat, average='weighted')
```

```
/home/jupyterlab/conda/envs/python/lib/python3.6/site-
packages/sklearn/metrics/classification.py:1143: UndefinedMetricWarning: F-score
is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

```
[136]: 0.9756083878995574
```

```
[137]: #jaccard index for accuracy
       from sklearn.metrics import jaccard_similarity_score
       jaccard_similarity_score(y_test, yhat)
```

```
[137]: 0.9809069212410502
```

```
[ ]:
```

```
[ ]:
```