ML Part 1: Assignment N°:2

Alexei Marcilio, GBC 100988494

December 7, 2020

Contents

ML__Part__I

Prepared by : Moe Fadae & Najem Bouazza

Week of Dec 5th, 2020

GBC - Toronto

Assignment N°:2

- Please write the appropiate lines of code to get the outputs below.
- Answer the questions on the ML models part.
- You will be using a dataset titled: "Female_labor_force.csv" shared along with this document.

.

.

.

I - Numpy arrays:

Create an array of 20 elements with a value of 7 :

[3]:
```python
import numpy as np
a = np.linspace(7, 7, 20)
a
```

[3]:
```
array([7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7.,
       7., 7., 7.])
```

[3]:
```python

```

[3]:
```
array([7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7., 7.,
       7., 7., 7.])
```

Create an array of integers from 2 to 36 :

[7]:
```python
b = np.arange(2, 37, 1)
b
```

[7]:
```
array([ 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])
```

[4]:
```python

```

[4]:
```
array([ 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])
```

Create an array of odd integers from 13, to 73:

[8]:
```
c = np.arange(13, 74, 2)
c
```

[8]:
```
array([13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45,
       47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73])
```

[5]:
```

```

[5]:
```
array([13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45,
       47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73])
```

Create a 6X6 matrix with even numbers ranging from 0 to 36 :

[10]:
```
d = np.arange(0,72,2).reshape(6,6)
d
```

[10]:
```
array([[ 0,  2,  4,  6,  8, 10],
       [12, 14, 16, 18, 20, 22],
       [24, 26, 28, 30, 32, 34],
       [36, 38, 40, 42, 44, 46],
       [48, 50, 52, 54, 56, 58],
       [60, 62, 64, 66, 68, 70]])
```

[12]:
```
np.arange(0,72,2).reshape(6,6)
```

[12]:
```
array([[ 0,  2,  4,  6,  8, 10],
       [12, 14, 16, 18, 20, 22],
       [24, 26, 28, 30, 32, 34],
       [36, 38, 40, 42, 44, 46],
       [48, 50, 52, 54, 56, 58],
       [60, 62, 64, 66, 68, 70]])
```

Get the slice from the array as below :

[22]:
```
d[1:-1,1:-1]
```

[22]:
```
array([[14, 16, 18, 20],
       [26, 28, 30, 32],
       [38, 40, 42, 44],
       [50, 52, 54, 56]])
```

[14]:
```

```

[14]:
```
array([[14, 16, 18, 20],
       [26, 28, 30, 32],
       [38, 40, 42, 44],
       [50, 52, 54, 56]])
```

Generate 3 random numbers between 0 and 1 :

[24]:
```
e = np.random.rand(3)
e
```

[24]:
```
array([0.80470029, 0.03316439, 0.69640727])
```

[18]:
```

```

[18]:
```
array([0.66195835, 0.73284915, 0.12385177])
```

Generate the following matrix :

[32]:
```
f = np.arange(0.02,0.51,0.01).reshape(7,7)
np.fill_diagonal(f, f.diagonal() + 1)
f
```

[32]:
```
array([[1.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08],
       [0.09, 1.1 , 0.11, 0.12, 0.13, 0.14, 0.15],
       [0.16, 0.17, 1.18, 0.19, 0.2 , 0.21, 0.22],
       [0.23, 0.24, 0.25, 1.26, 0.27, 0.28, 0.29],
       [0.3 , 0.31, 0.32, 0.33, 1.34, 0.35, 0.36],
       [0.37, 0.38, 0.39, 0.4 , 0.41, 1.42, 0.43],
       [0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 1.5 ]])
```

[36]:
```

```

[36]:
```
array([[1.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08],
       [0.09, 1.1 , 0.11, 0.12, 0.13, 0.14, 0.15],
       [0.16, 0.17, 1.18, 0.19, 0.2 , 0.21, 0.22],
       [0.23, 0.24, 0.25, 1.26, 0.27, 0.28, 0.29],
       [0.3 , 0.31, 0.32, 0.33, 1.34, 0.35, 0.36],
       [0.37, 0.38, 0.39, 0.4 , 0.41, 1.42, 0.43],
       [0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 1.5 ]])
```

Write the code to get the following outputs from the previous matrix :

[35]:
```
f[:,5]
```

[35]:
```
array([0.07, 0.14, 0.21, 0.28, 0.35, 1.42, 0.49])
```

[37]:
```

```

[37]: array([0.07, 0.14, 0.21, 0.28, 0.35, 1.42, 0.49])

[49]:
```python
g = np.array([[f[4,4]]])
g
```

[49]: array([[1.34]])

[38]:
```python

```

[38]: array([[1.34]])

[56]:
```python
f[0:5,0:4]
```

[56]:
```
array([[1.02, 0.03, 0.04, 0.05],
       [0.09, 1.1 , 0.11, 0.12],
       [0.16, 0.17, 1.18, 0.19],
       [0.23, 0.24, 0.25, 1.26],
       [0.3 , 0.31, 0.32, 0.33]])
```

[40]:
```python

```

[40]:
```
array([[1.02, 0.03, 0.04, 0.05],
       [0.09, 1.1 , 0.11, 0.12],
       [0.16, 0.17, 1.18, 0.19],
       [0.23, 0.24, 0.25, 1.26],
       [0.3 , 0.31, 0.32, 0.33]])
```

II - Pandas dataframes:

Import the dataset 'Female_labor_force.csv', Check the first 3 samples from the dataset :

[57]:
```python
import pandas as pd
flf = pd.read_csv("Female_labor_force.csv")
flf.head(3)
```

[57]:
```
   No  Country Level of development European Union Membership Currency  \
0   4  Austria             Developed                    Member     Euro
1   6  Belgium             Developed                    Member     Euro
2  17  Estonia             Developed                    Member     Euro

   Women Entrepreneurship Index  Entrepreneurship Index  Inflation rate  \
0                          54.9                    64.9            0.90
1                          63.6                    65.5            0.60
2                          55.4                    60.2           -0.88

   Female Labor Force Participation Rate
0                                   67.1
```

```
1                                   58.0
2                                   68.5
```

[9]:

```

```

[9]:

```
   No  Country Level of development European Union Membership Currency  \
0   4  Austria            Developed                    Member     Euro
1   6  Belgium            Developed                    Member     Euro
2  17  Estonia            Developed                    Member     Euro

   Women Entrepreneurship Index  Entrepreneurship Index  Inflation rate  \
0                          54.9                    64.9            0.90
1                          63.6                    65.5            0.60
2                          55.4                    60.2           -0.88

   Female Labor Force Participation Rate
0                                   67.1
1                                   58.0
2                                   68.5
```

How many countries we have in this dataset, put them in a list and print the list :

[64]:

```
countUnique = flf['Country'].nunique()
```

[65]:

```
print(f"There are {countUnique} unique countries in the dataset.")
```

```
There are 51 unique countries in the dataset.
```

[72]:

```
# Put unique countries in a list
listUniqueCountry = flf['Country'].unique().tolist()
type(listUniqueCountry)
print(listUniqueCountry)
```

```
['Austria', 'Belgium', 'Estonia', 'Finland', 'France', 'Germany', 'Greece',
'Ireland', 'Italy', 'Latvia', 'Lithuania', 'Netherlands', 'Slovakia',
'Slovenia', 'Spain', 'Croatia', 'Denmark', 'Hungary', 'Poland', 'Sweden',
'Australia', 'Iceland', 'Japan', 'Norway', 'Singapore', 'Switzerland', 'Taiwan',
'Algeria', 'Argentina', 'Bolivia', 'Bosnia and Herzegovina', 'Brazil', 'China',
'Costa Rica', 'Ecuador', 'Egypt', 'El Salvador', 'Ghana', 'India', 'Jamaica',
'Macedonia', 'Malaysia', 'Mexico', 'Panama', 'Peru', 'Russia', 'Saudi Arabia',
'Thailand', 'Tunisia', 'Turkey', 'Uruguay']
```

Write the codes that show he following outputs :

[73]:

```
flf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 9 columns):
```

```
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   No                                  51 non-null     int64
 1   Country                             51 non-null     object
 2   Level of development                51 non-null     object
 3   European Union Membership           51 non-null     object
 4   Currency                            51 non-null     object
 5   Women Entrepreneurship Index        51 non-null     float64
 6   Entrepreneurship Index              51 non-null     float64
 7   Inflation rate                      51 non-null     float64
 8   Female Labor Force Participation Rate  51 non-null     float64
dtypes: float64(4), int64(1), object(4)
memory usage: 3.7+ KB
```

[12]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 9 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   No                                  51 non-null     int64
 1   Country                             51 non-null     object
 2   Level of development                51 non-null     object
 3   European Union Membership           51 non-null     object
 4   Currency                            51 non-null     object
 5   Women Entrepreneurship Index        51 non-null     float64
 6   Entrepreneurship Index              51 non-null     float64
 7   Inflation rate                      51 non-null     float64
 8   Female Labor Force Participation Rate  51 non-null     float64
dtypes: float64(4), int64(1), object(4)
memory usage: 3.7+ KB
```

[74]:

```
flf.describe()
```

[74]:

	No	Women Entrepreneurship Index	Entrepreneurship Index	\
count	51.000000	51.000000	51.000000	
mean	29.980392	47.835294	47.241176	
std	18.017203	14.268480	16.193149	
min	1.000000	25.300000	24.800000	
25%	14.500000	36.350000	31.900000	
50%	30.000000	44.500000	42.700000	
75%	45.500000	59.150000	65.400000	
max	60.000000	74.800000	77.600000	

	Inflation rate	Female Labor Force Participation Rate
count	51.000000	51.000000

```
mean         2.587647                                58.481765
std          5.380639                                13.864567
min         -2.250000                                13.000000
25%         -0.500000                                55.800000
50%          0.600000                                61.000000
75%          3.600000                                67.400000
max         26.500000                                82.300000
```

[13]:

[13]:
```
              No  Women Entrepreneurship Index  Entrepreneurship Index  \
count  51.000000                     51.000000               51.000000
mean   29.980392                     47.835294               47.241176
std    18.017203                     14.268480               16.193149
min     1.000000                     25.300000               24.800000
25%    14.500000                     36.350000               31.900000
50%    30.000000                     44.500000               42.700000
75%    45.500000                     59.150000               65.400000
max    60.000000                     74.800000               77.600000

       Inflation rate  Female Labor Force Participation Rate
count       51.000000                              51.000000
mean         2.587647                              58.481765
std          5.380639                              13.864567
min         -2.250000                              13.000000
25%         -0.500000                              55.800000
50%          0.600000                              61.000000
75%          3.600000                              67.400000
max         26.500000                              82.300000
```

Look for NaN and Duplicated values in the dataframe :

[103]:
```
# Answer
flf.isnull().sum()
```

[103]:
```
No                                       0
Country                                  0
Level of development                     0
European Union Membership                0
Currency                                 0
Women Entrepreneurship Index             0
Entrepreneurship Index                   0
Inflation rate                           0
Female Labor Force Participation Rate    0
dtype: int64
```

[14]:

[14]:
```
No                                     0
Country                                0
Level of development                   0
European Union Membership              0
Currency                               0
Women Entrepreneurship Index           0
Entrepreneurship Index                 0
Inflation rate                         0
Female Labor Force Participation Rate    0
dtype: int64
```

[102]:
```python
# Answer
flf.duplicated().unique()[0]
```

[102]:
```
False
```

[16]:
```python

```

[16]:
```
False
```

[77]:
```python
# Answer
flf['Currency'].unique()
```

[77]:
```
array(['Euro', 'National Currency'], dtype=object)
```

[24]:
```python

```

[24]:
```
array(['Euro', 'National Currency'], dtype=object)
```

[79]:
```python
# Answer
flf['Level of development'].unique()
```

[79]:
```
array(['Developed', 'Developing'], dtype=object)
```

[25]:
```python

```

[25]:
```
array(['Developed', 'Developing'], dtype=object)
```

[ ]:
```python

```

[ ]:
```python

```

III - Linear Regression :

1 - Encode the categorical variables:

[110]:
```
# As there are over 50 countries I decided not to encode this categorical␣
↪variable.
# Also it really does not make sense to encode the country as in the future we␣
↪won't have
# the country available if we are trying to predict Female Labour Participation␣
↪Rate
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
flf['Level of development'] = le.fit_transform(flf['Level of development'])
flf['European Union Membership'] = le.fit_transform(flf['European Union␣
↪Membership'])
flf['Currency'] = le.fit_transform(flf['Currency'])
```

[112]:
```
flf.columns
```

[112]:
```
Index(['No', 'Country', 'Level of development', 'European Union Membership',
       'Currency', 'Women Entrepreneurship Index', 'Entrepreneurship Index',
       'Inflation rate', 'Female Labor Force Participation Rate'],
      dtype='object')
```

2 -Define X and y:

[125]:
```
X = flf[['Level of development', 'European Union Membership', 'Currency',␣
↪'Women Entrepreneurship Index', \
        'Entrepreneurship Index','Inflation rate']]
y = flf['Female Labor Force Participation Rate']
```

3 -Split the data into Train and Test:

[127]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
↪random_state = 0)
```

4 -Create and fit a Linear regression model :

[129]:
```
from sklearn.linear_model import LinearRegression

#Create the model :
regressor = LinearRegression(normalize=True)

#Train the model :
regressor.fit(X_train, y_train)
```

[129]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=True)
```

5 -Predict X_test and evaluate the model:

[130]:
```
y_pred = regressor.predict(X_test)
from sklearn.metrics import r2_score
#R_squared :
R_squared = r2_score(y_test, y_pred)
print("R^2 Value in %:              {:.2f} % ".format((R_squared*100)))`b
```

```
R^2 Value in %:              -18.44 %
```

The R^2 is not that good at -18.44

6 -Print the model's equation (intercept and coeff):

[137]:
```
L = regressor.coef_

print("Female Part. rate = {:.2f} + {:.2f}*Level of development + {:.
↪2f}*European Union Membership + {:.2f}*Currency + {:.2f}*Women␣
↪Entrepreneurship Index + {:.2f}*Entrepreneurship Index + {:.2f}*Inflation␣
↪Rate".format(regressor.intercept_,L[0], L[1], L[2] ,L[3],L[4],L[5]));
```

```
Female Part. rate = 22.63 + -4.82*Level of development + 6.09*European Union
Membership + 1.13*Currency + 1.17*Women Entrepreneurship Index +
-0.54*Entrepreneurship Index + 0.63*Inflation Rate
```

IV - Logistic Regression:

1 -Define y as the column "Level of development" and X as the rest of the features :

[139]:
```
X = flf[['European Union Membership', 'Currency', 'Women Entrepreneurship␣
↪Index', \
     'Entrepreneurship Index','Inflation rate', 'Female Labor Force␣
↪Participation Rate']]
y = flf['Level of development']
```

2 -Split the data into Train set and Test set :

[141]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
↪random_state = 0)
```

3 -Create and train a logistic regression Model:

[142]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.linear_model import LogisticRegression
```

```python
log_reg = LogisticRegression()

log_reg.fit(X_train, y_train)
```

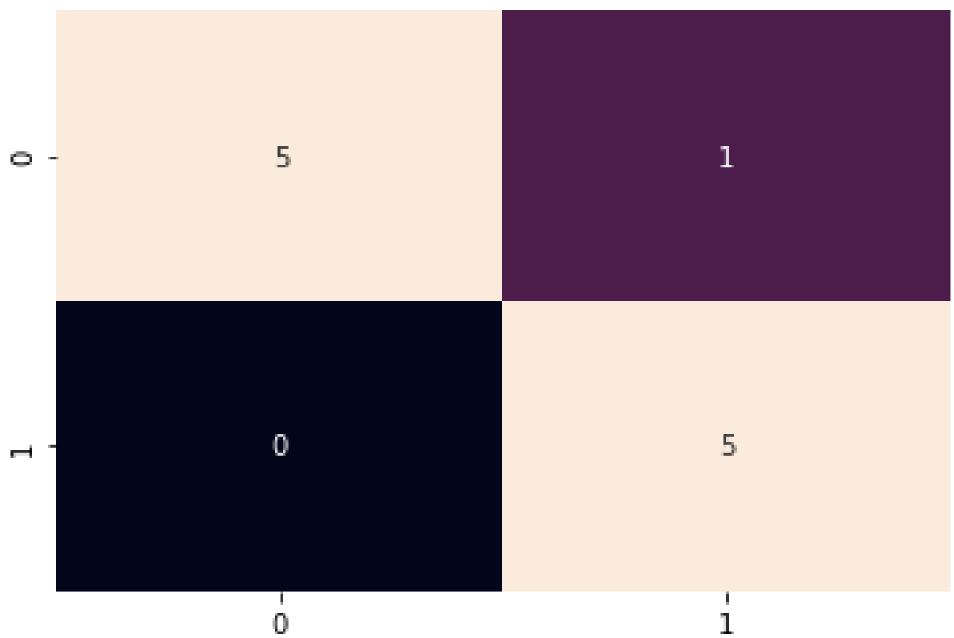
[142]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

4 -Predict X_test and print the results : confusion matrix and classification report

[145]:
```python
y_pred = log_reg.predict(X_test)
y_pred_proba = log_reg.predict_proba(X_test)

from sklearn.metrics import confusion_matrix
import seaborn as sns
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cbar = False)
```

[145]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa364b37a60>
```



[148]:
```python
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

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

```
             0       1.00      0.83      0.91         6
             1       0.83      1.00      0.91         5

      accuracy                           0.91        11
     macro avg       0.92      0.92      0.91        11
  weighted avg       0.92      0.91      0.91        11
```

5 -Use model.predict_proba() to get the predictions for X_test:

Use 70% as your threshold instead of 50% (default threshold).

[150]:
```
# Changing the threshold to 70%
def my_filter(x):
    if x > 0.7:
        return 1
    else:
        return 0
```

[151]:
```
y_pred_new = np.array([my_filter(x) for x in y_pred_proba[:,1]])
```

6 -Compare the confusion matrices for both predictions :

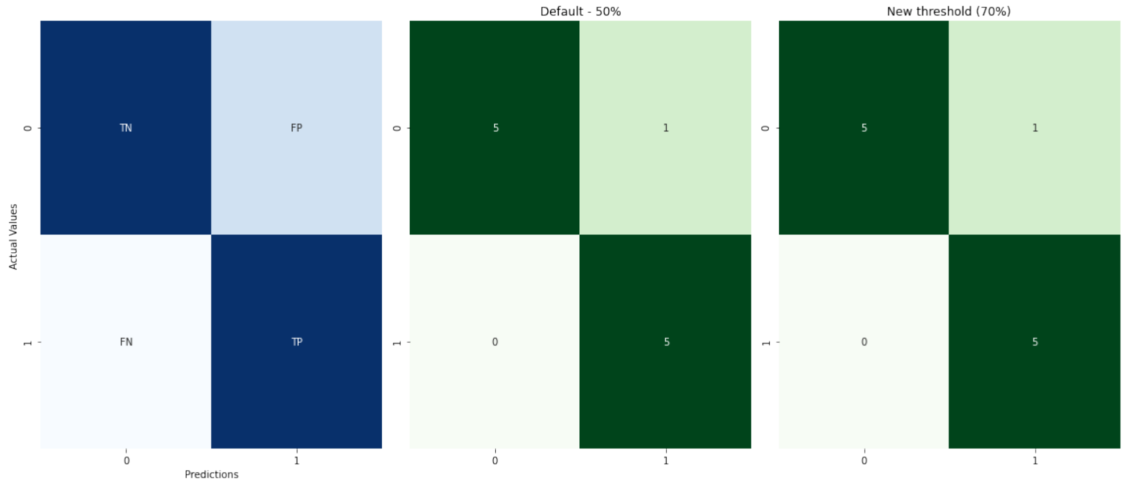
[155]:
```
import matplotlib.pyplot as plt
# Confustion Matrix for new threshold
plt.figure(figsize = (16,7))

#Confusion matrix with labels
plt.subplot(1,3,1)
labels =np.array([['TN','FP'],['FN','TP']])
sns.heatmap(confusion_matrix(y_test, y_pred), annot=labels,fmt='',␣
 ↪cmap='Blues', cbar = False)
plt.xlabel('Predictions')
plt.ylabel('Actual Values')

#Confusion matrix (0.5)
plt.subplot(1,3,2)
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Greens', cbar =␣
 ↪False)
plt.title("Default - 50%")

#Confusion matrix (new threshold)
plt.subplot(1,3,3)
sns.heatmap(confusion_matrix(y_test, y_pred_new), annot=True,cmap='Greens',␣
 ↪cbar = False)
plt.title("New threshold (70%)")
```

```
plt.tight_layout()
```



We can see that the new threshold did not change the results. Each confusion matrix shows only one false positive.

End !