alessio.reitano@studio.unibo.it

February 6, 2022

1 Alessio Reitano - Matricola 0001005384

[114]:
```
#Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier

#Variables
url = 'exam2022_01_13(1).csv'
sep =','
random_state = 42
target = 'language'

#Directives
%matplotlib inline
np.random.seed(random_state)
```

1.0.1 1. Load the data and explore them, showing size, structure and histograms of numeric data; show the histogram of the frequencies of the class labels, contained in the “language” column

[113]:
```
#Load the data
df = pd.read_csv(url, sep = sep)
#Show the size
print ("The size is: {}".format(df.shape))
```

```
The size is: (329, 13)
```

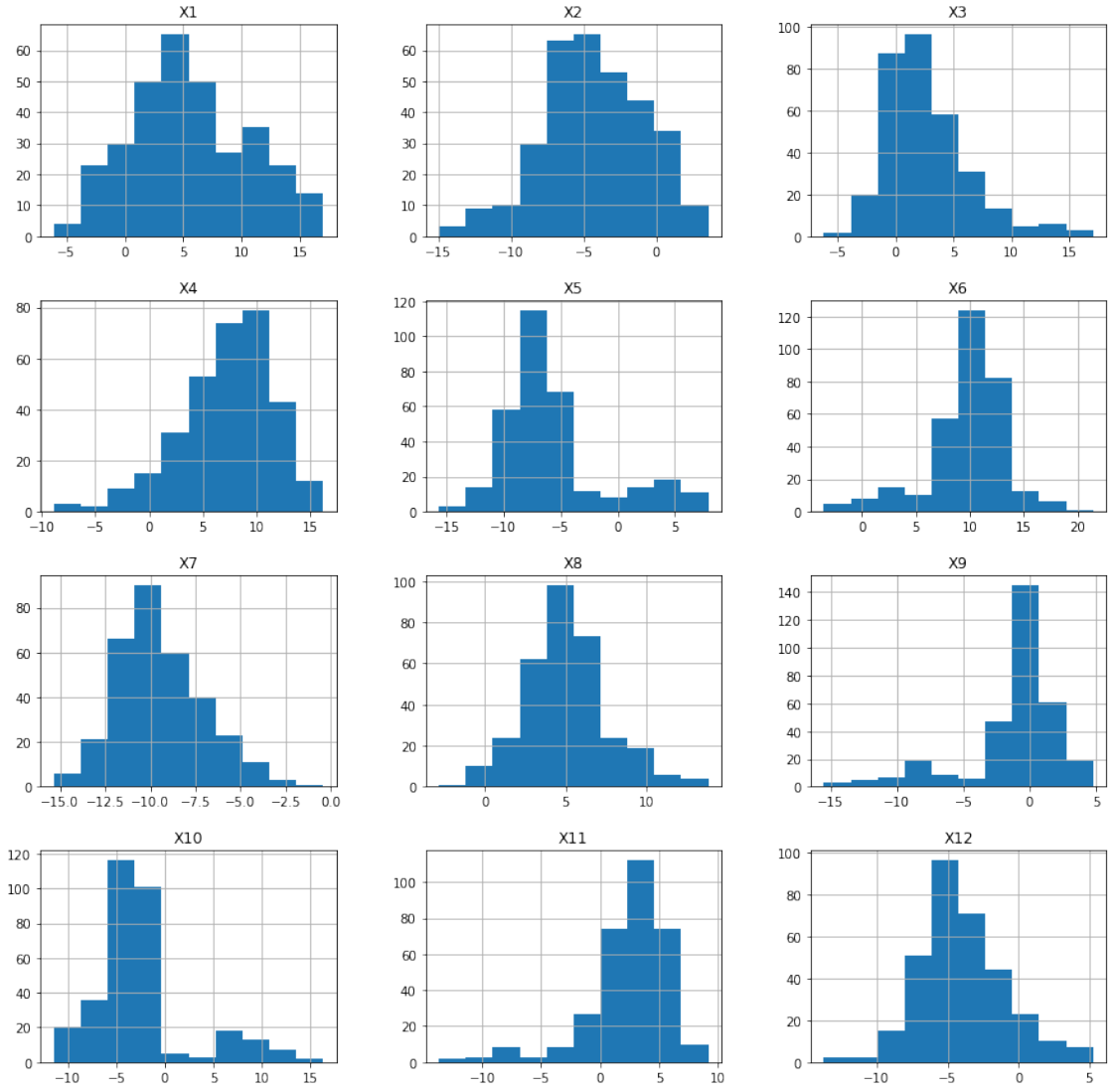
[112]:
```
#Show the structure of DataFrame
df.describe()
```

[112]:

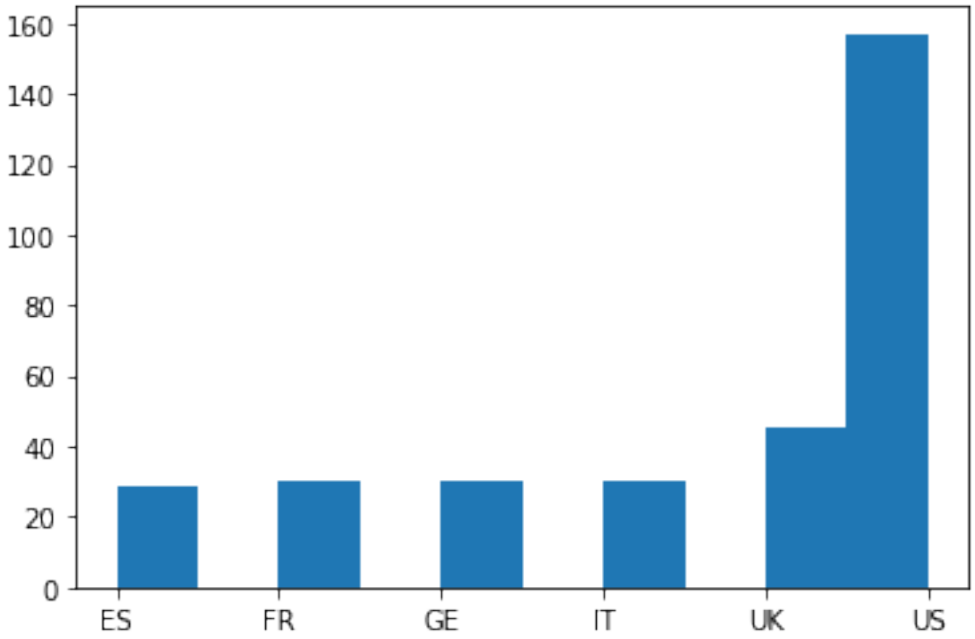
	X1	X2	X3	X4	X5	X6	\
count	321.000000	321.000000	321.000000	321.000000	321.000000	321.000000	
mean	5.557383	-4.287615	2.659787	7.191238	-5.666852	9.771914	
std	5.021384	3.534740	3.651345	4.305794	4.549815	3.563509	
min	-6.067831	-14.972962	-6.186583	-8.844231	-15.656596	-3.528964	
25%	1.945427	-6.515764	0.127094	4.700874	-8.417684	8.645897	
50%	4.915523	-4.323565	2.087431	7.842187	-6.786670	10.366871	
75%	9.530158	-1.560250	4.273387	10.069418	-4.532381	11.710766	
max	16.971161	3.570765	17.066487	16.178942	7.912809	21.445837	

	X7	X8	X9	X10	X11	X12
count	321.000000	321.000000	321.000000	321.000000	321.000000	321.000000
mean	-9.418684	5.137920	-1.228532	-2.411215	2.426878	-3.975382
std	2.429508	2.612064	3.588327	5.027840	3.473527	2.971385
min	-15.341538	-2.873862	-15.510974	-11.429178	-13.664104	-13.724103
25%	-11.114653	3.491943	-1.749082	-5.090180	1.268999	-5.900776
50%	-9.710399	4.843103	-0.412717	-3.327718	3.169703	-4.374334
75%	-7.998089	6.588931	0.721021	-1.512083	4.636732	-2.230275
max	-0.424033	13.846083	4.789989	16.326455	9.166066	5.259430

[111]:
```
#Show the histogram
pd.DataFrame.hist(df, figsize= [15,15]);
```



[110]:
```
#Show the histogram
plt.hist(df.language)
plt.show()
```



1.0.2 2. Drop the rows with NaN values, if any, show the shape of the dataset after this cleaning

[109]:
```
#Drop rows with null values
df = df.dropna()
print ("The rows after cleaning are: {}".format(df.shape[0]))
```

```
The rows after cleaning are: 321
```

1.0.3 3. Tune the hyper–parameters of Model1 with Cross Validation on the training set, optimize for recall_macro

[108]:
```
#Decide as Model1 DecisionTreeClassifier
X = df.drop(columns = [target])
y = df[target]
```

[107]:
```
#Split X and y in train and test
Xtrain, Xtest, ytrain, ytest = train_test_split(X,y,random_state=random_state)
print ("There are {} samples in the training dataset".format(Xtrain.shape[0]))
print ("There are {} samples in the testing dataset".format(Xtest.shape[0]))
print ("Each samples has {} features".format(Xtrain.shape[1]))
```

```
There are 240 samples in the training dataset
There are 81 samples in the testing dataset
Each samples has 12 features
```

[105]:
```
#Create the model1
model1 = DecisionTreeClassifier(criterion='entropy',random_state = random_state)
#Fit the classifier
model1.fit(Xtrain, ytrain)
param = range(1, model1.tree_.max_depth+1)
```

[106]:
```
scores = []
for par in param :
    model1 = DecisionTreeClassifier(criterion = 'entropy', max_depth = par,␣
 ↪random_state = random_state)
    score = cross_val_score(model1, Xtrain, ytrain, scoring = 'recall_macro',␣
 ↪cv=5)
    scores.append(np.mean(score))
print (scores)
```

```
[0.16666666666666666, 0.34074879227053134, 0.43878881987577645,
0.5343650793650794, 0.5680607315389923, 0.5140527950310558, 0.533616287094548,
0.5462750172532781, 0.5514354727398205, 0.5552035886818495]
```

1.0.4 4. produce a classification report for Model1 on the test set

[83]:
```
too_par = param[np.argmax(scores)]
model1 = DecisionTreeClassifier(max_depth = too_par, criterion =␣
 ↪'entropy',random_state=random_state)
model1.fit(Xtest,ytest)
ytest_model = model1.predict(Xtest)
accuracy = accuracy_score(ytest,ytest_model)*100
print("The accuracy on test set tuned with cross_validation with Model1 is {:.
 ↪2f}% with depth {}".format(accuracy,too_par))
```

```
The accuracy on test set tuned with cross_validation with Model1 is 92.59% with
depth 5
```

[84]:
```
print (classification_report(ytest,ytest_model))
```

```
              precision    recall  f1-score   support

          ES       1.00      1.00      1.00         8
          FR       0.88      1.00      0.93         7
          GE       1.00      1.00      1.00         9
          IT       0.73      1.00      0.85        11
          UK       0.80      0.67      0.73         6
          US       1.00      0.90      0.95        40

    accuracy                           0.93        81
   macro avg       0.90      0.93      0.91        81
weighted avg       0.94      0.93      0.93        81
```

1.0.5 5. produce the confusion matrix for Model1 on the test set

[67]:
```
confusion_matrix(ytest,ytest_model)
```

[67]:
```
array([[ 8,  0,  0,  0,  0,  0],
       [ 0,  7,  0,  0,  0,  0],
       [ 0,  0,  9,  0,  0,  0],
       [ 0,  0,  0, 11,  0,  0],
       [ 0,  0,  0,  2,  4,  0],
       [ 0,  1,  0,  2,  1, 36]])
```

1.0.6 6. tune the hyper–parameters of Model2 with Cross Validation on the training set, optimize for recall_macro

[89]:
```
#Decide as Model2
model2 = KNeighborsClassifier()
param2 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

[115]:
```
scores2 = []
for i in param2 :
    model2 = KNeighborsClassifier(n_neighbors = i, metric = 'euclidean')
    score2 = cross_val_score(model2, Xtrain, ytrain, scoring = 'recall_macro',␣
 ↪cv=5)
    scores2.append(np.mean(score2))
print (scores2)
```

```
[0.7903260869565217, 0.7532367149758454, 0.7618926846100759, 0.7510990338164251,
0.7236197377501725, 0.7337405106970324, 0.673719806763285, 0.6430641821946169,
0.6301293995859213, 0.6206452726017944]
```

1.0.7 7. produce a classification report for Model2 on the test set

[116]:
```
too_par = param[np.argmax(scores2)]
model2 = KNeighborsClassifier(n_neighbors = too_par, metric = 'euclidean')
model2.fit(Xtest,ytest)
ytest_model = model2.predict(Xtest)
accuracy = accuracy_score(ytest,ytest_model)*100
print("The accuracy on test set tuned with cross_validation with Model2 is {:.
 ↪2f}% with depth {}".format(accuracy,too_par))
```

```
The accuracy on test set tuned with cross_validation with Model2 is 100.00% with
depth 1
```

[117]:
```
print (classification_report(ytest,ytest_model))
```

```
              precision    recall  f1-score   support

          ES       1.00      1.00      1.00         8
          FR       1.00      1.00      1.00         7
```

```
           GE       1.00      1.00      1.00         9
           IT       1.00      1.00      1.00        11
           UK       1.00      1.00      1.00         6
           US       1.00      1.00      1.00        40

     accuracy                           1.00        81
    macro avg       1.00      1.00      1.00        81
 weighted avg       1.00      1.00      1.00        81
```

1.0.8 8. produce the confusion matrix for Model2 on the test set

[99]:
```
confusion_matrix(ytest,ytest_model)
```

[99]:
```
array([[ 8,  0,  0,  0,  0,  0],
       [ 0,  7,  0,  0,  0,  0],
       [ 0,  0,  9,  0,  0,  0],
       [ 0,  0,  0, 11,  0,  0],
       [ 0,  0,  0,  0,  6,  0],
       [ 0,  0,  0,  0,  0, 40]])
```

[ ]:
```

```