REGex

Task: 11

ID: SIRSS1196

Nitesh Marchande

In [80]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [81]:

```
df=pd.read_csv("train.csv")
td=pd.read_csv("test.csv")
```

In [82]:

```
df.head(10)
```

Out[82]:

	id	species	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	margin9	margi
0	1	Acer_Opalus	0.007812	0.023438	0.023438	0.003906	0.011719	0.009766	0.027344	0.0	0.001953	0.033
1	2	Pterocarya_Stenoptera	0.005859	0.000000	0.031250	0.015625	0.025391	0.001953	0.019531	0.0	0.000000	0.007
2	3	Quercus_Hartwissiana	0.005859	0.009766	0.019531	0.007812	0.003906	0.005859	0.068359	0.0	0.000000	0.044
3	5	Tilia_Tomentosa	0.000000	0.003906	0.023438	0.005859	0.021484	0.019531	0.023438	0.0	0.013672	0.017
4	6	Quercus_Variabilis	0.005859	0.003906	0.048828	0.009766	0.013672	0.015625	0.005859	0.0	0.000000	0.005
5	8	Magnolia_Salicifolia	0.070312	0.093750	0.033203	0.001953	0.000000	0.152340	0.007812	0.0	0.003906	0.027
6	10	Quercus_Canariensis	0.021484	0.031250	0.017578	0.009766	0.001953	0.042969	0.039062	0.0	0.003906	0.019
7	11	Quercus_Rubra	0.000000	0.000000	0.037109	0.050781	0.003906	0.000000	0.003906	0.0	0.048828	0.003
8	14	Quercus_Brantii	0.005859	0.001953	0.033203	0.015625	0.001953	0.000000	0.023438	0.0	0.000000	0.021
9	15	Salix_Fragilis	0.000000	0.000000	0.009766	0.037109	0.072266	0.000000	0.000000	0.0	0.007812	0.001

10 rows × 194 columns

In [83]:

```
td.head(10)
```

Out[83]:

	id	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	margin9	margin10	margin11	margin1
0	4	0.019531	0.009766	0.078125	0.011719	0.003906	0.015625	0.005859	0.0	0.005859	0.023438	0.005859	0.02148
1	7	0.007812	0.005859	0.064453	0.009766	0.003906	0.013672	0.007812	0.0	0.033203	0.023438	0.009766	0.01953
2	9	0.000000	0.000000	0.001953	0.021484	0.041016	0.000000	0.023438	0.0	0.011719	0.005859	0.001953	0.02148
3	12	0.000000	0.000000	0.009766	0.011719	0.017578	0.000000	0.003906	0.0	0.003906	0.001953	0.000000	0.02929
4	13	0.001953	0.000000	0.015625	0.009766	0.039062	0.000000	0.009766	0.0	0.005859	0.000000	0.001953	0.03320
5	16	0.021484	0.033203	0.021484	0.009766	0.015625	0.035156	0.039062	0.0	0.003906	0.029297	0.013672	0.01953
6	19	0.015625	0.025391	0.046875	0.009766	0.005859	0.027344	0.042969	0.0	0.000000	0.027344	0.015625	0.00976
7	23	0.007812	0.031250	0.011719	0.050781	0.000000	0.117190	0.003906	0.0	0.011719	0.017578	0.056641	0.00195

	id	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	margin9	margin10	margin11	margin1
8	24	0.003906	0.007812	0.074219	0.017578	0.015625	0.003906	0.011719	0.0	0.009766	0.011719	0.023438	0.03710
9	28	0.000000	0.000000	0.005859	0.021484	0.054688	0.000000	0.015625	0.0	0.011719	0.005859	0.000000	0.03320

10 rows × 193 columns

In [84]:

```
df.info()
td.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 990 entries, 0 to 989
Columns: 194 entries, id to texture64
dtypes: float64(192), int64(1), object(1)
memory usage: 1.5+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 594 entries, 0 to 593
Columns: 193 entries, id to texture64
dtypes: float64(192), int64(1)
memory usage: 895.8 KB
```

In [85]:

```
print('train shape ',df.shape)
print('test shape ',td.shape)
```

```
train shape  (990, 194)
test shape  (594, 193)
```

In [86]:

```
df.columns
```

Out[86]:

```
Index(['id', 'species', 'margin1', 'margin2', 'margin3', 'margin4', 'margin5',
       'margin6', 'margin7', 'margin8',
       ...
       'texture55', 'texture56', 'texture57', 'texture58', 'texture59',
       'texture60', 'texture61', 'texture62', 'texture63', 'texture64'],
      dtype='object', length=194)
```

In [52]:

```
td.columns
```

Out[52]:

```
Index(['id', 'margin1', 'margin2', 'margin3', 'margin4', 'margin5', 'margin6',
       'margin7', 'margin8', 'margin9',
       ...
       'texture55', 'texture56', 'texture57', 'texture58', 'texture59',
       'texture60', 'texture61', 'texture62', 'texture63', 'texture64'],
      dtype='object', length=193)
```

In [53]:

```
df.nunique()
td.nunique()
```

Out[53]:

```
id            594
margin1        42
margin2        77
margin3        59
margin4        63
             ...
texture60      61
texture61      34
```

```
texture62     103
texture63      56
texture64      87
Length: 193, dtype: int64
```

In [54]:

```
print(df.isnull().sum())
print(td.isnull().sum())
```

```
id            0
species       0
margin1       0
margin2       0
margin3       0
             ..
texture60     0
texture61     0
texture62     0
texture63     0
texture64     0
Length: 194, dtype: int64
id            0
margin1       0
margin2       0
margin3       0
margin4       0
             ..
texture60     0
texture61     0
texture62     0
texture63     0
texture64     0
Length: 193, dtype: int64
```

In [55]:

```
from sklearn.preprocessing import LabelEncoder
enc=LabelEncoder()
df['species']=enc.fit_transform(df['species'])
```

In [56]:

```
X=df.drop(['id','species'],axis=1).values
Y=df[['species']].values
print(X.shape,Y.shape)
```

```
(990, 192) (990, 1)
```

In [57]:

```
td.head()
```

Out[57]:

	id	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	margin9	margin10	margin11	margin1
0	4	0.019531	0.009766	0.078125	0.011719	0.003906	0.015625	0.005859	0.0	0.005859	0.023438	0.005859	0.02148
1	7	0.007812	0.005859	0.064453	0.009766	0.003906	0.013672	0.007812	0.0	0.033203	0.023438	0.009766	0.01953
2	9	0.000000	0.000000	0.001953	0.021484	0.041016	0.000000	0.023438	0.0	0.011719	0.005859	0.001953	0.02148
3	12	0.000000	0.000000	0.009766	0.011719	0.017578	0.000000	0.003906	0.0	0.003906	0.001953	0.000000	0.02929
4	13	0.001953	0.000000	0.015625	0.009766	0.039062	0.000000	0.009766	0.0	0.005859	0.000000	0.001953	0.03320

5 rows × 193 columns

Decision Tree Classifier

In [58]:

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion='entropy',random_state=2)
dtc.fit(x_train,y_train)
```

Out[58]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=2, splitter='best')
```

In [92]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X)
X = scaler.transform(X)
```

In [93]:

```
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=5)
```

Random Forest

In [94]:

```
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators = 20,criterion = 'entropy', max_dept
h = 20, random_state = 5)
rf_classifier.fit(x_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to
(n_samples,), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[94]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=20, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=20,
                       n_jobs=None, oob_score=False, random_state=5, verbose=0,
                       warm_start=False)
```

In [95]:

```
pred_train = rf_classifier.predict(x_train)
pred_test = rf_classifier.predict(x_test)
```

In [96]:

```
from sklearn.metrics import accuracy_score
print('Training Accuracy: ', accuracy_score(y_train, pred_train))
print('Testing Accuracy: ', accuracy_score(y_test, pred_test))
```

```
Training Accuracy:  1.0
Testing Accuracy:  0.8720538720538721
```

In [97]:

```
y_pred=rf_classifier.predict(x_test)
print(y_test,y_pred)
```

```
[[45]
 [28]
```

```
[25]
[82]
[29]
[71]
[71]
[96]
[82]
[94]
[29]
[15]
[22]
[46]
[82]
[53]
[94]
[ 5]
[84]
[83]
[77]
[72]
[66]
[62]
[59]
[88]
[48]
[30]
[86]
[14]
[84]
[48]
[95]
[69]
[37]
[70]
[55]
[10]
[16]
[80]
[24]
[42]
[14]
[13]
[56]
[88]
[61]
[68]
[66]
[17]
[18]
[89]
[63]
[49]
[10]
[97]
[31]
[57]
[51]
[30]
[78]
[47]
[16]
[42]
[73]
[71]
[39]
[ 0]
[76]
[32]
[48]
[72]
[ 0]
[80]
```

```
    [74]
    [57]
    [29]
    [11]
    [40]
    [84]
    [74]
    [ 0]
    [28]
    [42]
    [93]
    [30]
    [30]
    [12]
    [31]
    [34]
    [92]
    [17]
    [23]
    [54]
    [50]
    [33]
    [ 4]
    [31]
    [23]
    [51]
    [28]
    [25]
    [44]
    [24]
    [12]
    [ 3]
    [88]
    [45]
    [97]
    [67]
    [71]
    [20]
    [44]
    [28]
    [88]
    [ 0]
    [98]
    [74]
    [ 2]
    [64]
    [67]
    [86]
    [80]
    [89]
    [34]
    [41]
    [76]
    [55]
    [79]
    [12]
    [46]
    [63]
    [97]
    [77]
    [41]
    [11]
    [39]
    [13]
    [73]
    [29]
    [27]
    [ 6]
    [55]
    [46]
    [14]
    [46]
```

```
 [ 0]
 [68]
 [55]
 [48]
 [21]
 [31]
 [80]
 [80]
 [62]
 [ 2]
 [32]
 [40]
 [98]
 [97]
 [80]
 [90]
 [56]
 [94]
 [70]
 [75]
 [94]
 [19]
 [79]
 [30]
 [22]
 [23]
 [45]
 [98]
 [95]
 [ 4]
 [44]
 [94]
 [88]
 [36]
 [51]
 [51]
 [44]
 [64]
 [65]
 [54]
 [33]
 [36]
 [28]
 [84]
 [ 3]
 [89]
 [18]
 [26]
 [19]
 [53]
 [98]
 [35]
 [57]
 [47]
 [80]
 [21]
 [72]
 [91]
 [64]
 [25]
 [28]
 [67]
 [53]
 [ 8]
 [85]
 [72]
 [56]
 [35]
 [76]
 [48]
 [27]
 [17]
 [ 0]
 [92]
 [15]
 [93]
 [38]
 [25]
 [40]
 [53]
 [85]
 [66]
 [70]
 [46]
 [60]
 [83]
 [11]
 [36]
 [12]
 [42]
 [54]
 [42]
 [61]
 [92]
 [69]
 [21]
 [70]
 [ 9]
 [14]
 [81]
 [19]
 [15]
 [72]
 [73]
 [ 7]
 [35]
 [10]
 [33]
 [63]
 [66]
 [78]
 [32]
 [71]
 [52]
 [41]
 [42]
 [59]
 [37]
 [27]
 [88]
 [ 6]
 [82]
 [62]
 [51]
 [69]
 [78]
 [45]
 [ 5]
 [54]
 [68]
 [77]
 [56]
 [64]
 [49]
 [ 8]
 [ 9]
 [91]
 [40]
 [38]
 [49]
 [ 5]
 [49]
 [79]
 [44]
 [55]
 [ 4]
 [ 1]
 [56]
 [63]
 [ 7]
 [33]] [45 15 25 82 29 71 63 96 82 94 29 15 22 46 82 53 94  5 84 83 77 72 66 62
 59 88 48 30 86 14 84 17 95 69 37 70 55 10 16 11 24 42 14 13 56 88 61 55
 66 17 18 89 63  7 10 97 31 57 51 30 78 47 16 42 17 71 39  0 76 32 49 72
  0 80 74 57 29 11 40 84 74  0 29 42 93 30 30 12 31 34 92 17 23 54 50 33
  4 21 23 31 28 25 44 24 12  3 88 45 97 67 71 20 44 28 88  0 98 74  2 64
 67 86 35 89 34 41 76 50 79 12 46 82 97 77 41 11 39 13 18 29 27  6 55 46
 14 46  0 68 55 48 21 31 49 80 62  2 32 40 33 97 93 90 56 94 70 10 94 19
 29 30 22 23 45 33 95  4 44  3 88 36 51 31  5 53 65 54 33 36 29 84  3 89
 18 26 19 53  3 35 57 47 35 21 72 91 64 25 29 67 10  8 85 72 56 35 76 48
 27 17  0 92 15 93 38 22 39 53 85 66 70 46 60 83 11 36 12 42 54 42 61 92
 54 21 70  9 14 81 19 15 72 73  7 35 10 33 63 66 78 32 71 52 41 42 59 37
 27 88  6 82 62 31 85 78 45  5 54 68 77 56 64 14  1  9 91 40 38 49  5 13
 79 44 55  4  1 56 63  7 33]
```

In [98]:

```
print(accuracy_score(y_test,y_pred))
print(y_test,y_pred)
```

```
0.8720538720538721
[[45]
 [28]
 [25]
 [82]
 [29]
 [71]
 [71]
 [96]
 [82]
 [94]
 [29]
 [15]
 [22]
 [46]
 [82]
 [53]
 [94]
 [ 5]
 [84]
 [83]
 [77]
 [72]
 [66]
 [62]
 [59]
 [88]
 [48]
 [30]
 [86]
 [14]
 [84]
 [48]
 [95]
 [69]
 [37]
 [70]
 [55]
 [10]
 [16]
 [80]
 [24]
 [42]
 [14]
 [13]
 [56]
 [88]
```

```
[61]
[68]
[66]
[17]
[18]
[89]
[63]
[49]
[10]
[97]
[31]
[57]
[51]
[30]
[78]
[47]
[16]
[42]
[73]
[71]
[39]
[ 0]
[76]
[32]
[48]
[72]
[ 0]
[80]
[74]
[57]
[29]
[11]
[40]
[84]
[74]
[ 0]
[28]
[42]
[93]
[30]
[30]
[12]
[31]
[34]
[92]
[17]
[23]
[54]
[50]
[33]
[ 4]
[31]
[23]
[51]
[28]
[25]
[44]
[24]
[12]
[ 3]
[88]
[45]
[97]
[67]
[71]
[20]
[44]
[28]
[88]
[ 0]
[98]
[74]
```

```
 [ 2]
 [64]
 [67]
 [86]
 [80]
 [89]
 [34]
 [41]
 [76]
 [55]
 [79]
 [12]
 [46]
 [63]
 [97]
 [77]
 [41]
 [11]
 [39]
 [13]
 [73]
 [29]
 [27]
 [ 6]
 [55]
 [46]
 [14]
 [46]
 [ 0]
 [68]
 [55]
 [48]
 [21]
 [31]
 [80]
 [80]
 [62]
 [ 2]
 [32]
 [40]
 [98]
 [97]
 [80]
 [90]
 [56]
 [94]
 [70]
 [75]
 [94]
 [19]
 [79]
 [30]
 [22]
 [23]
 [45]
 [98]
 [95]
 [ 4]
 [44]
 [94]
 [88]
 [36]
 [51]
 [51]
 [44]
 [64]
 [65]
 [54]
 [33]
 [36]
 [28]
 [84]
 [ 3]
 [89]
 [18]
 [26]
 [19]
 [53]
 [98]
 [35]
 [57]
 [47]
 [80]
 [21]
 [72]
 [91]
 [64]
 [25]
 [28]
 [67]
 [53]
 [ 8]
 [85]
 [72]
 [56]
 [35]
 [76]
 [48]
 [27]
 [17]
 [ 0]
 [92]
 [15]
 [93]
 [38]
 [25]
 [40]
 [53]
 [85]
 [66]
 [70]
 [46]
 [60]
 [83]
 [11]
 [36]
 [12]
 [42]
 [54]
 [42]
 [61]
 [92]
 [69]
 [21]
 [70]
 [ 9]
 [14]
 [81]
 [19]
 [15]
 [72]
 [73]
 [ 7]
 [35]
 [10]
 [33]
 [63]
 [66]
 [78]
 [32]
 [71]
 [52]
 [41]
 [42]
 [59]
 [37]
 [27]
 [88]
 [ 6]
 [82]
 [62]
 [51]
 [69]
 [78]
 [45]
 [ 5]
 [54]
 [68]
 [77]
 [56]
 [64]
 [49]
 [ 8]
 [ 9]
 [91]
 [40]
 [38]
 [49]
 [ 5]
 [49]
 [79]
 [44]
 [55]
 [ 4]
 [ 1]
 [56]
 [63]
 [ 7]
 [33]] [45 15 25 82 29 71 63 96 82 94 29 15 22 46 82 53 94  5 84 83 77 72 66 62
 59 88 48 30 86 14 84 17 95 69 37 70 55 10 16 11 24 42 14 13 56 88 61 55
 66 17 18 89 63  7 10 97 31 57 51 30 78 47 16 42 17 71 39  0 76 32 49 72
  0 80 74 57 29 11 40 84 74  0 29 42 93 30 30 12 31 34 92 17 23 54 50 33
  4 21 23 31 28 25 44 24 12  3 88 45 97 67 71 20 44 28 88  0 98 74  2 64
 67 86 35 89 34 41 76 50 79 12 46 82 97 77 41 11 39 13 18 29 27  6 55 46
 14 46  0 68 55 48 21 31 49 80 62  2 32 40 33 97 93 90 56 94 70 10 94 19
 29 30 22 23 45 33 95  4 44  3 88 36 51 31  5 53 65 54 33 36 29 84  3 89
 18 26 19 53  3 35 57 47 35 21 72 91 64 25 29 67 10  8 85 72 56 35 76 48
 27 17  0 92 15 93 38 22 39 53 85 66 70 46 60 83 11 36 12 42 54 42 61 92
 54 21 70  9 14 81 19 15 72 73  7 35 10 33 63 66 78 32 71 52 41 42 59 37
 27 88  6 82 62 31 85 78 45  5 54 68 77 56 64 14  1  9 91 40 38 49  5 13
 79 44 55  4  1 56 63  7 33]
```

In [99]:

```
test_ids = df.pop('id')
x_test = df.values
```

In [100]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X)
X = scaler.transform(X)
```

In [110]:

```
submission = pd.DataFrame(y_test)
```

In [106]:

```
submission.head(5)
```

Out[106]:

	Acer_Capillipes	Acer_Circinatum	Acer_Mono	Acer_Opalus	Acer_Palmatum	Acer_Pictum	Acer_Platanoids	Acer_Rubrum

	Acer_Capillipes	Acer_Circinatum	Acer_Mono	Acer_Opalus	Acer_Palmatum	Acer_Pictum	Acer_Platanoids	Acer_Rubrum
0	0.0	0.0	0.05	0.05	0.0	0.0	0.00	0.00
1	0.0	0.0	0.00	0.00	0.0	0.0	0.00	0.05
2	0.0	0.0	0.00	0.00	0.0	0.0	0.00	0.00
3	0.0	0.0	0.00	0.00	0.0	0.0	0.00	0.00
4	0.0	0.0	0.00	0.00	0.0	0.0	0.05	0.00

5 rows × 99 columns

In [71]:

```
submission.to_csv('submission_leaf_classification.csv')
```