**#Task 11

Leela Satya Kartheek Raja

Registration Id : SIRSS1038

In [23]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
import warnings
warnings.filterwarnings("ignore")
```

```
/kaggle/input/leaf-classification/train.csv.zip
/kaggle/input/leaf-classification/sample_submission.csv.zip
/kaggle/input/leaf-classification/images.zip
/kaggle/input/leaf-classification/test.csv.zip
```

In [24]:

```
# load data
train_data = pd.read_csv('../input/leaf-classification/train.csv.zip', index_col ='id')
test_data = pd.read_csv('../input/leaf-classification/test.csv.zip')
```

In [25]:

```
test_ids = test_data.id
test_data = test_data.drop(['id'], axis =1)
```

Part 1: Data Preprocessing

In [26]:

```
train_data.head()
```

Out[26]:

id	species	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	margin9	...	textu
1	Acer_Opalus	0.007812	0.023438	0.023438	0.003906	0.011719	0.009766	0.027344	0.0	0.001953	...	0.007
2	Pterocarya_Stenoptera	0.005859	0.000000	0.031250	0.015625	0.025391	0.001953	0.019531	0.0	0.000000	...	0.000
3	Quercus_Hartwissiana	0.005859	0.009766	0.019531	0.007812	0.003906	0.005859	0.068359	0.0	0.000000	...	0.154
5	Tilia_Tomentosa	0.000000	0.003906	0.023438	0.005859	0.021484	0.019531	0.023438	0.0	0.013672	...	0.000
6	Quercus_Variabilis	0.005859	0.003906	0.048828	0.009766	0.013672	0.015625	0.005859	0.0	0.000000	...	0.096

5 rows × 193 columns

In [27]:

```
# taking care of missing values
train_data.isnull().any().sum()
```

Out[27]:

```
0
```

In [28]:

```
test_data.head()
```

Out[28]:

	margin1	margin2	margin3	margin4	margin5	margin6	margin7	margin8	margin9	margin10	...	texture55	texture5
0	0.019531	0.009766	0.078125	0.011719	0.003906	0.015625	0.005859	0.0	0.005859	0.023438	...	0.006836	0.00000
1	0.007812	0.005859	0.064453	0.009766	0.003906	0.013672	0.007812	0.0	0.033203	0.023438	...	0.000000	0.00000
2	0.000000	0.000000	0.001953	0.021484	0.041016	0.000000	0.023438	0.0	0.011719	0.005859	...	0.128910	0.00000
3	0.000000	0.000000	0.009766	0.011719	0.017578	0.000000	0.003906	0.0	0.003906	0.001953	...	0.012695	0.01562
4	0.001953	0.000000	0.015625	0.009766	0.039062	0.000000	0.009766	0.0	0.005859	0.000000	...	0.000000	0.04296

5 rows × 192 columns

In [29]:

```
test_data.isnull().any().sum()
```

Out[29]:

```
0
```

In [30]:

```
# encoding catagorical
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 990 entries, 1 to 1584
Columns: 193 entries, species to texture64
dtypes: float64(192), object(1)
memory usage: 1.5+ MB
```

In [31]:

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 594 entries, 0 to 593
Columns: 192 entries, margin1 to texture64
dtypes: float64(192)
memory usage: 891.1 KB
```

In [32]:

```
train_data.shape
```

Out[32]:

```
(990, 193)
```

In [33]:

```
test_data.shape
```

Out[33]:

```
(594, 192)
```

In [34]:

```
train_data.describe().T
```

Out[34]:

	count	mean	std	min	25%	50%	75%	max
margin1	990.0	0.017412	0.019739	0.0	0.001953	0.009766	0.025391	0.087891

	count	mean	std	min	25%	50%	75%	max
margin2	990.0	0.028539	0.038855	0.0	0.001953	0.011719	0.041016	0.205080
margin3	990.0	0.031988	0.025847	0.0	0.013672	0.025391	0.044922	0.156250
margin4	990.0	0.023280	0.028411	0.0	0.005859	0.013672	0.029297	0.169920
margin5	990.0	0.014264	0.018390	0.0	0.001953	0.007812	0.017578	0.111330
...	...	...	...	...	...	...	...	...
texture60	990.0	0.014017	0.060151	0.0	0.000000	0.000000	0.000000	0.578130
texture61	990.0	0.002688	0.011415	0.0	0.000000	0.000000	0.000000	0.151370
texture62	990.0	0.020291	0.039040	0.0	0.000000	0.003906	0.023438	0.375980
texture63	990.0	0.008989	0.013791	0.0	0.000000	0.002930	0.012695	0.086914
texture64	990.0	0.019420	0.022768	0.0	0.000977	0.011719	0.029297	0.141600

192 rows × 8 columns

In [35]:

```
test_data.describe().T
```

Out[35]:

	count	mean	std	min	25%	50%	75%	max
margin1	594.0	0.017562	0.019585	0.0	0.001953	0.009766	0.028809	0.085938
margin2	594.0	0.028425	0.038351	0.0	0.001953	0.010743	0.041016	0.189450
margin3	594.0	0.031858	0.025719	0.0	0.013672	0.023438	0.042969	0.167970
margin4	594.0	0.022556	0.028797	0.0	0.005859	0.013672	0.027344	0.164060
margin5	594.0	0.014527	0.018029	0.0	0.001953	0.007812	0.019531	0.093750
...	...	...	...	...	...	...	...	...
texture60	594.0	0.011217	0.052530	0.0	0.000000	0.000000	0.000000	0.606450
texture61	594.0	0.002617	0.011204	0.0	0.000000	0.000000	0.000000	0.123050
texture62	594.0	0.019975	0.034704	0.0	0.000000	0.003418	0.022461	0.247070
texture63	594.0	0.009389	0.013457	0.0	0.000000	0.002930	0.014648	0.086914
texture64	594.0	0.020970	0.023407	0.0	0.000977	0.013184	0.032227	0.149410

192 rows × 8 columns

In [36]:

```
train_data['species'].nunique()
```

Out[36]:

```
99
```

The target y is the only catagorical column

In [37]:

```
# IV and DV
x = train_data.drop('species',axis=1)
y = train_data['species']
```

In [38]:

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
y_fit = encoder.fit(train_data['species'])
y_label = y_fit.transform(train_data['species'])
classes = list(y_fit.classes_)
```

```
classes
```

Out[38]:

```
['Acer_Capillipes',
 'Acer_Circinatum',
 'Acer_Mono',
 'Acer_Opalus',
 'Acer_Palmatum',
 'Acer_Pictum',
 'Acer_Platanoids',
 'Acer_Rubrum',
 'Acer_Rufinerve',
 'Acer_Saccharinum',
 'Alnus_Cordata',
 'Alnus_Maximowiczii',
 'Alnus_Rubra',
 'Alnus_Sieboldiana',
 'Alnus_Viridis',
 'Arundinaria_Simonii',
 'Betula_Austrosinensis',
 'Betula_Pendula',
 'Callicarpa_Bodinieri',
 'Castanea_Sativa',
 'Celtis_Koraiensis',
 'Cercis_Siliquastrum',
 'Cornus_Chinensis',
 'Cornus_Controversa',
 'Cornus_Macrophylla',
 'Cotinus_Coggygria',
 'Crataegus_Monogyna',
 'Cytisus_Battandieri',
 'Eucalyptus_Glaucescens',
 'Eucalyptus_Neglecta',
 'Eucalyptus_Urnigera',
 'Fagus_Sylvatica',
 'Ginkgo_Biloba',
 'Ilex_Aquifolium',
 'Ilex_Cornuta',
 'Liquidambar_Styraciflua',
 'Liriodendron_Tulipifera',
 'Lithocarpus_Cleistocarpus',
 'Lithocarpus_Edulis',
 'Magnolia_Heptapeta',
 'Magnolia_Salicifolia',
 'Morus_Nigra',
 'Olea_Europaea',
 'Phildelphus',
 'Populus_Adenopoda',
 'Populus_Grandidentata',
 'Populus_Nigra',
 'Prunus_Avium',
 'Prunus_X_Shmittii',
 'Pterocarya_Stenoptera',
 'Quercus_Afares',
 'Quercus_Agrifolia',
 'Quercus_Alnifolia',
 'Quercus_Brantii',
 'Quercus_Canariensis',
 'Quercus_Castaneifolia',
 'Quercus_Cerris',
 'Quercus_Chrysolepis',
 'Quercus_Coccifera',
 'Quercus_Coccinea',
 'Quercus_Crassifolia',
 'Quercus_Crassipes',
 'Quercus_Dolicholepis',
 'Quercus_Ellipsoidalis',
 'Quercus_Greggii',
 'Quercus_Hartwissiana',
 'Quercus_Ilex',
 'Quercus_Imbricaria',
 'Quercus_Infectoria_sub'
```

```
 'Quercus_Kewensis',
 'Quercus_Nigra',
 'Quercus_Palustris',
 'Quercus_Phellos',
 'Quercus_Phillyraeoides',
 'Quercus_Pontica',
 'Quercus_Pubescens',
 'Quercus_Pyrenaica',
 'Quercus_Rhysophylla',
 'Quercus_Rubra',
 'Quercus_Semecarpifolia',
 'Quercus_Shumardii',
 'Quercus_Suber',
 'Quercus_Texana',
 'Quercus_Trojana',
 'Quercus_Variabilis',
 'Quercus_Vulcanica',
 'Quercus_x_Hispanica',
 'Quercus_x_Turneri',
 'Rhododendron_x_Russellianum',
 'Salix_Fragilis',
 'Salix_Intergra',
 'Sorbus_Aria',
 'Tilia_Oliveri',
 'Tilia_Platyphyllos',
 'Tilia_Tomentosa',
 'Ulmus_Bergmanniana',
 'Viburnum_Tinus',
 'Viburnum_x_Rhytidophylloides',
 'Zelkova_Serrata']
```

In [39]:

```
# splitting
from sklearn.model_selection import  train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y_label, test_size = 0.2, random_s
tate =1)
```

Part 2: Building model

In [40]:

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 40)
classifier.fit(x_train, y_train)
```

Out[40]:

```
RandomForestClassifier(n_estimators=40)
```

In [41]:

```
from sklearn.metrics import classification_report
predictions = classifier.predict(x_test)
print (classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         2
           3       1.00      1.00      1.00         3
           4       1.00      1.00      1.00         3
           5       1.00      1.00      1.00         2
           6       1.00      1.00      1.00         3
           7       1.00      1.00      1.00         1
           8       1.00      1.00      1.00         1
           9       1.00      1.00      1.00         3
          10       1.00      1.00      1.00         1
          11       1.00      1.00      1.00         3
```

12	1.00	1.00	1.00	3
13	1.00	1.00	1.00	2
14	1.00	1.00	1.00	2
15	1.00	1.00	1.00	2
16	1.00	1.00	1.00	2
17	1.00	1.00	1.00	1
18	1.00	1.00	1.00	3
19	1.00	1.00	1.00	2
20	1.00	0.80	0.89	5
21	1.00	1.00	1.00	3
22	1.00	1.00	1.00	3
23	1.00	0.67	0.80	3
24	0.50	1.00	0.67	1
25	1.00	1.00	1.00	1
26	1.00	1.00	1.00	3
27	0.00	0.00	0.00	1
28	1.00	0.67	0.80	3
29	0.00	0.00	0.00	0
30	1.00	1.00	1.00	2
31	1.00	1.00	1.00	1
32	0.67	1.00	0.80	2
33	1.00	0.80	0.89	5
34	1.00	1.00	1.00	1
36	1.00	1.00	1.00	2
37	1.00	1.00	1.00	2
39	1.00	1.00	1.00	3
40	1.00	1.00	1.00	2
41	1.00	1.00	1.00	4
42	1.00	1.00	1.00	2
43	1.00	1.00	1.00	1
44	0.00	0.00	0.00	1
45	1.00	1.00	1.00	2
46	1.00	1.00	1.00	2
47	1.00	1.00	1.00	2
48	1.00	1.00	1.00	3
49	1.00	1.00	1.00	2
50	1.00	1.00	1.00	2
51	0.00	0.00	0.00	0
52	1.00	1.00	1.00	1
53	1.00	1.00	1.00	3
54	0.80	0.67	0.73	6
55	0.33	1.00	0.50	1
56	1.00	1.00	1.00	1
58	1.00	1.00	1.00	1
59	1.00	1.00	1.00	3
60	0.67	1.00	0.80	2
61	1.00	1.00	1.00	5
62	1.00	1.00	1.00	2
63	1.00	1.00	1.00	3
64	1.00	0.50	0.67	2
65	1.00	1.00	1.00	4
66	0.67	1.00	0.80	2
68	0.00	0.00	0.00	1
69	1.00	0.50	0.67	2
70	1.00	1.00	1.00	2
71	1.00	1.00	1.00	2
72	1.00	1.00	1.00	1
73	1.00	1.00	1.00	1
74	1.00	1.00	1.00	1
75	1.00	1.00	1.00	1
76	1.00	1.00	1.00	1
77	1.00	1.00	1.00	2
78	1.00	1.00	1.00	2
79	1.00	1.00	1.00	1
81	0.00	0.00	0.00	1
82	1.00	1.00	1.00	2
83	1.00	1.00	1.00	2
84	0.50	1.00	0.67	1
85	1.00	1.00	1.00	1
86	0.67	1.00	0.80	2
87	1.00	1.00	1.00	2
88	1.00	1.00	1.00	1

```
          89       1.00      1.00      1.00         3
          90       1.00      1.00      1.00         2
          91       1.00      1.00      1.00         1
          92       1.00      1.00      1.00         2
          93       0.75      1.00      0.86         3
          94       1.00      1.00      1.00         3
          95       1.00      1.00      1.00         4
          96       1.00      1.00      1.00         3
          97       0.50      0.50      0.50         2
          98       1.00      1.00      1.00         2

    accuracy                           0.93       198
   macro avg       0.89      0.91      0.89       198
weighted avg       0.94      0.93      0.93       198
```

In [42]:

```
final_predictions = classifier.predict_proba(test_data)
```

In [43]:

```
submission = pd.DataFrame(final_predictions, columns=classes)
submission.insert(0, 'id', test_ids)
submission.reset_index()
```

Out[43]:

	index	id	Acer_Capillipes	Acer_Circinatum	Acer_Mono	Acer_Opalus	Acer_Palmatum	Acer_Pictum	Acer_Platanoids
0	0	4	0.000	0.000	0.000	0.000	0.000	0.0	0.000
1	1	7	0.000	0.000	0.100	0.050	0.000	0.0	0.000
2	2	9	0.000	0.525	0.025	0.000	0.075	0.0	0.000
3	3	12	0.000	0.000	0.000	0.025	0.000	0.0	0.025
4	4	13	0.025	0.000	0.000	0.000	0.000	0.0	0.000
...	...	...	...	...	...	...	...	...	...
589	589	1576	0.000	0.675	0.000	0.000	0.050	0.0	0.000
590	590	1577	0.000	0.025	0.000	0.000	0.000	0.0	0.000
591	591	1579	0.000	0.075	0.000	0.000	0.050	0.0	0.000
592	592	1580	0.000	0.000	0.025	0.000	0.000	0.0	0.000
593	593	1583	0.000	0.000	0.000	0.000	0.000	0.0	0.000

594 rows × 101 columns

In [44]:

```
submission.to_csv('result.csv', index = False)
```

In [ ]:

```
```