

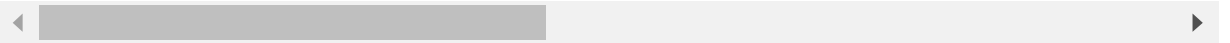
```
In [1]: from sklearn.svm import LinearSVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

```
In [2]: #Reading Training data
train_data = pd.read_csv("train.csv")
train_data.head()
```

```
Out[2]:
```

|   | id | species               | margin1  | margin2  | margin3  | margin4  | margin5  | margin6  | margin  |
|---|----|-----------------------|----------|----------|----------|----------|----------|----------|---------|
| 0 | 1  | Acer_Opalus           | 0.007812 | 0.023438 | 0.023438 | 0.003906 | 0.011719 | 0.009766 | 0.02734 |
| 1 | 2  | Pterocarya_Stenoptera | 0.005859 | 0.000000 | 0.031250 | 0.015625 | 0.025391 | 0.001953 | 0.01953 |
| 2 | 3  | Quercus_Hartwissiana  | 0.005859 | 0.009766 | 0.019531 | 0.007812 | 0.003906 | 0.005859 | 0.06835 |
| 3 | 5  | Tilia_Tomentosa       | 0.000000 | 0.003906 | 0.023438 | 0.005859 | 0.021484 | 0.019531 | 0.02343 |
| 4 | 6  | Quercus_Variabilis    | 0.005859 | 0.003906 | 0.048828 | 0.009766 | 0.013672 | 0.015625 | 0.00585 |

5 rows × 194 columns

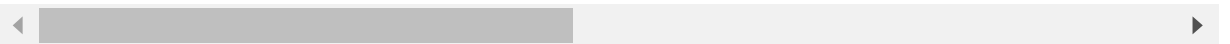


```
In [3]: #Reading testing data
test_data = pd.read_csv("test.csv")
test_data.head()
```

```
Out[3]:
```

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

5 rows × 193 columns



```
In [4]: #Encoding Data
labelencoder = LabelEncoder()
train_data["species"] = labelencoder.fit_transform(train_data["species"])
train_data.head()
```

```
Out[4]:
```

|   | id | species | margin1  | margin2  | margin3  | margin4  | margin5  | margin6  | margin7  | margin8 |
|---|----|---------|----------|----------|----------|----------|----------|----------|----------|---------|
| 0 | 1  | 3       | 0.007812 | 0.023438 | 0.023438 | 0.003906 | 0.011719 | 0.009766 | 0.027344 | 0.0     |
| 1 | 2  | 49      | 0.005859 | 0.000000 | 0.031250 | 0.015625 | 0.025391 | 0.001953 | 0.019531 | 0.0     |
| 2 | 3  | 65      | 0.005859 | 0.009766 | 0.019531 | 0.007812 | 0.003906 | 0.005859 | 0.068359 | 0.0     |
| 3 | 5  | 94      | 0.000000 | 0.003906 | 0.023438 | 0.005859 | 0.021484 | 0.019531 | 0.023438 | 0.0     |
| 4 | 6  | 84      | 0.005859 | 0.003906 | 0.048828 | 0.009766 | 0.013672 | 0.015625 | 0.005859 | 0.0     |

5 rows × 194 columns



```
In [5]: #Defining dependent and independent data
X = train_data.iloc[:, 1:]
Y = train_data["species"]
```

```
In [6]: #Train-test data
x_train, x_test, y_train, y_test = train_test_split(X, Y, random_state=5, test_size=20)
```

```
In [7]: # 1.Random Forest Classifier
randm_class = RandomForestClassifier(n_estimators=100)
randm_class.fit(x_train, y_train)
```

```
Out[7]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                               criterion='gini', max_depth=None, 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=100,
                               n_jobs=None, oob_score=False, random_state=None,
                               verbose=0, warm_start=False)
```

```
In [8]: predict_rf = randm_class.predict(x_test)
print("Random Classifier Accuracy Score: ", metrics.accuracy_score(predict_rf,
y_test))
```

Random Classifier Accuracy Score: 0.95

```
In [9]: # 2. Decision Tree Classifier
dec_class = DecisionTreeClassifier()
dec_class.fit(x_train, y_train)
```

```
Out[9]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                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=None, splitter='best')
```

```
In [10]: predict_dt = dec_class.predict((x_test))
print("Decision Tree Classifier Accuracy Score: ", metrics.accuracy_score(predict_dt, y_test))
```

Decision Tree Classifier Accuracy Score: 0.8

```
In [11]: # 3. Naive Bayes Classifier
gn_class = GaussianNB()
gn_class.fit(x_train, y_train)
```

```
Out[11]: GaussianNB(priors=None, var_smoothing=1e-09)
```

```
In [12]: predict_gn = gn_class.predict(x_test)
print("Naive Bayes Accuracy Score: ", metrics.accuracy_score(predict_gn, y_test))
```

Naive Bayes Accuracy Score: 1.0

```
In [13]: # 4. SVM Classifier
svm_class = LinearSVC()
svm_class.fit(x_train, y_train)
```

C:\Users\hp\anaconda3\lib\site-packages\sklearn\svm\\_base.py:947: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.  
"the number of iterations.", ConvergenceWarning)

```
Out[13]: LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
                    intercept_scaling=1, loss='squared_hinge', max_iter=1000,
                    multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
                    verbose=0)
```

```
In [14]: predict_svm = svm_class.predict(x_test)
print("SVM Accuracy Score: ", metrics.accuracy_score(predict_svm, y_test))
```

SVM Accuracy Score: 0.35

```
In [15]: # Best Classifier - Naive Bayes

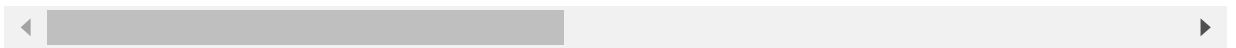
predict_test = gn_class.predict(test_data)

test_data["species"] = labelencoder.inverse_transform(predict_test) #Inverse
the encoding
test_data.head()
```

```
Out[15]:
```

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

5 rows × 194 columns



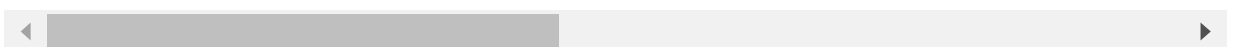
```
In [16]: #save predicted data
test_data.to_csv("Output.csv", index=True)
```

```
In [17]: data = pd.read_csv("Output.csv")
data.head()
```

```
Out[17]:
```

|   | Unnamed: 0 | id | margin1  | margin2  | margin3  | margin4  | margin5  | margin6  | margin7  | margin8 | margin9  |
|---|------------|----|----------|----------|----------|----------|----------|----------|----------|---------|----------|
| 0 | 0          | 4  | 0.019531 | 0.009766 | 0.078125 | 0.011719 | 0.003906 | 0.015625 | 0.005859 | 0.0     | 0.005859 |
| 1 | 1          | 7  | 0.007812 | 0.005859 | 0.064453 | 0.009766 | 0.003906 | 0.013672 | 0.007812 | 0.0     | 0.033203 |
| 2 | 2          | 9  | 0.000000 | 0.000000 | 0.001953 | 0.021484 | 0.041016 | 0.000000 | 0.023438 | 0.0     | 0.011719 |
| 3 | 3          | 12 | 0.000000 | 0.000000 | 0.009766 | 0.011719 | 0.017578 | 0.000000 | 0.003906 | 0.0     | 0.003906 |
| 4 | 4          | 13 | 0.001953 | 0.000000 | 0.015625 | 0.009766 | 0.039062 | 0.000000 | 0.009766 | 0.0     | 0.005859 |

5 rows × 195 columns



```
In [ ]:
```