Analysis of the records used by a leading scientific research organization, to classify the plants and leaves into different categories. Taking a sample file of 1.5 MB size and apply the different supervised machine learning algorithms (NVB,Decision trees,Random forest,SVM). Once the model is generated based on all approaches, find the accuracy of the model and compare them.

## Importing all the necessary libraries

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

## Getting all details about the file

In [ ]:
```
data = pd.read_csv("train.csv",header=0)

print(data.head(3))# as u can see our data have imported and having 33 columns

print(data.info())
```

## Data analysis - identifying the unique values of the dependent variable, using label encoder to convert the string categorical field to numerical field.

In [ ]:
```
op_cols=data["species"].nunique
print(op_cols)
```

In [ ]:
```
from sklearn.preprocessing import LabelEncoder

lbl = LabelEncoder()
data["species_new"]=lbl.fit_transform(data["species"])
resrpd.DataFrame(lbl.fit_transform(data["species"]))
matched_df=data[["species_new","species"]]
print(matched_df.head(10))
```

## Identifiying and splitting the dependent and independent variables.

In [ ]:
```
list_cols=data.columns
X1=[]
for i in list_cols:
    if i!="id" and i!="species" and i!="species_new":
        X1.append(i)
print(X1)
X=data[X1]
print(X.shape)
Y=data[["species_new"]]
print(Y.shape)
```

## Splitting the data into train and test .

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=111)
```

## Applying the decision tree algorithm on the identified data. Fit the model and predict the data as well.

In [ ]:
```
################## DECISION TREE ####################
from sklearn.tree import DecisionTreeClassifier
classifier_dt = DecisionTreeClassifier()
classifier_dt.fit(X_train,Y_train)
# Predicting on test data
y_pred_dt = classifier_dt.predict(X_test)
print(y_pred_dt)
```

## Build the confusion matrix and generate the classification report to identify the efficiency of the model. The confusion matrix is loaded into a file as the number of features is really high. The classification report tells how efficient the classification was using the parameters like precision,recall,f1-score.

In [ ]:
```
# Classification error metrics
from sklearn.metrics import confusion_matrix
print(Y_test.size)
print(y_pred_dt.size)
conf_res=confusion_matrix(Y_test,y_pred_dt)
conf_df=pd.DataFrame(conf_res)
conf_df.to_csv("confusion_data.csv")
```

In [ ]:
```
from sklearn.metrics import classification_report
from sklearn import tree

print(classification_report(Y_test,y_pred_dt))

fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)

tree.plot_tree(classifier_dt)

fig.savefig('imagename.png')
```

## Identify the accuracy of the model using the metrics library

In [ ]:
```
from sklearn import metrics

print(metrics.accuracy_score(Y_test,y_pred_dt)*100)

#The accuracy percentage is just 65
```

## Applying the same data on the Random Forest Classifier algorithm.

In [ ]:
```
################ RANDOM FOREST ###################
from sklearn.ensemble import RandomForestClassifier
classifier_rf = RandomForestClassifier(n_estimators=200,max_features=100)
classifier_rf.fit(X_train,Y_train)
# Predicting on test data
y_pred_rf = classifier_rf.predict(X_test)
```

## Applying the confusion matrix and the classification report on the predicted and actual data in order to check the efficiency of the model.

In [ ]:
```
# Classification error metrics
from sklearn.metrics import confusion_matrix
print(Y_test.size)
print(y_pred_rf.size)
conf_res=confusion_matrix(Y_test,y_pred_dt)
conf_df=pd.DataFrame(conf_res)
conf_df.to_csv("confusion_data_random.csv")
```

In [ ]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_test,y_pred_rf))
```

## Identifying the accuracy of the model using the metrics library

In [ ]:
```
from sklearn import metrics

print(metrics.accuracy_score(Y_test,y_pred_rf)*100)

#The accuracy here is 98
```

## Identifying the features which are having high importance in building this model. Arranging them in the order to identify the features that really matters.

In [ ]:
```
print(classifier_rf.feature_importances_.size)
print(classifier_rf.feature_importances_)
```

In [ ]:
```
imp_df = pd.DataFrame({
    "varname": X_train.columns,
    "Imp": classifier_rf.feature_importances_
})
```

In [ ]:
```
imp_df.sort_values(by="Imp", ascending=False)
```

## Applying the test.csv file on the random forest model created and storing the final result with the predicted species for every row.

In [ ]:
```
test_file=pd.read_csv("test.csv")

X=test_file.drop("id",axis=1)

y_fnl=classifier_rf.predict(X)

test_file["species_new"]=y_fnl

print(test_file.head(3))
print(len(y_fnl))

print(type(y_fnl))


int_op=matched_df[matched_df["species_new"].isin(y_fnl)]
print(int_op.head(3))

final_df=test_file.join(int_op.set_index('species_new'),on="species_new")
final_df=final_df.drop('species_new',axis=1)
final_df.to_csv("Test_File_op1.csv",index=False)
```

## Applying the same data in the naive byes algorithm.

In [ ]:
```
# Naive bayes classifier
from sklearn.naive_bayes import GaussianNB
classifier_nb = GaussianNB()
classifier_nb.fit(X_train,Y_train)
Y_pred_nb = classifier_nb.predict(X_test)
```

## The accuracy of this model can be predicted by using confusion matrix and the classification report(precision,recall,F1 score)

In [ ]:
```
# Classification error metrics
from sklearn.metrics import confusion_matrix
print(Y_test.size)
print(Y_pred_nb.size)
conf_nai=confusion_matrix(Y_test,Y_pred_nb)
print(conf_nai)
conf_df=pd.DataFrame(conf_nai)
conf_df.to_csv("confusion_data_naive.csv")
```

In [ ]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_test,Y_pred_nb))
```

In [ ]:
```
from sklearn import metrics
print(metrics.accuracy_score(Y_pred_nb,Y_test))
```

So as you observe the accuracy of the naive byes algorithm in this model is very less.

In [ ]:
```
Y_test_ser=Y_test.squeeze()
(Y_test_ser == Y_pred_nb).sum()
```

In [ ]:
```
(Y_test_ser=Y_pred_nb).sum()*1.0/len(Y_test)
```

The accuracy can also be calculated using the above method. The reason why naive byes gives low accuracy is because when the independent variable is categorical the values will be predicted better and accuracy goes high, but when it is continuous the accuracy becomes too low.

## Applying the test.csv file on the naive byes model created and storing the final result with the predicted species for every row.

In [ ]:
```
test_file=pd.read_csv("test.csv")

X=test_file.drop("id",axis=1)

y_fnl=classifier_nb.predict(X)

test_file["species_new"]=y_fnl

print(test_file.head(3))
print(len(y_fnl))

print(type(y_fnl))


int_op=matched_df[matched_df["species_new"].isin(y_fnl)]
print(int_op.head(3))

final_df=test_file.join(int_op.set_index('species_new'),on="species_new")
final_df=final_df.drop('species_new',axis=1)
final_df.to_csv("Test_File_op1_nb.csv",index=False)
```

## Applying the same data on the SVC model

In [ ]:
```
############## SUPPORT VECTOR MACHINE
from sklearn.svm import SVC
classifier_SVC = SVC()
classifier_SVC.fit(X_train,Y_train)
Y_pred_svc = classifier_SVC.predict(X_test)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,Y_pred_svc)
```

## Measuring the accuracy of the SVC model

In [ ]:
```
from sklearn import metrics
print(metrics.accuracy_score(Y_pred_svc,Y_test))
```

In [ ]:
```
Y_test_ser=Y_test.squeeze()
print("The accuracy is",(Y_test_ser==Y_pred_svc).sum()/len(Y_test))
```

## Applying the test.csv file on the SVC model created and storing the final result with the predicted species for every row.

In [ ]:
```
test_file=pd.read_csv("test.csv")

X=test_file.drop("id",axis=1)

y_fnl=classifier_SVC.predict(X)

test_file["species_new"]=y_fnl


int_op=matched_df[matched_df["species_new"].isin(y_fnl)]


final_df=test_file.join(int_op.set_index('species_new'),on="species_new")
final_df=final_df.drop('species_new',axis=1)


print(final_df.groupby(["species"]).size())
```

In [ ]:
```
plt.figure(figsize=(20,20))
plt.xticks(rotation=90)
sns.countplot(final_df["species"],label="count",orient="v")
```

So the observation of accuracy is Decision tree - 65 %, Random forest - 94 % , Naive byes - 43% and SVC - 88%

In [ ]: