# Noakhali Science & Technology University

Information & Communication Engineering

Advance Machine Learning, ICE-5121

***Assignment on:*** Create a Prediction based ML Model and prepares report of result on it where model can classify our input feature on category of `Variety` based on training dataset.

*Submit To*

Amzad Hossain
Assistant Professor
Dept. of
Information & Communication Engineering

*Submit By*

Abir Hosen
Year-1 Term-1
ASH 2111 MSc 114M
Session: 2020-21

Code: https://github.com/Abir-Hosen/ml-assignment

## Open-source machine learning packages

Some of top listed machine learning packages are

- ***Apache Mahout:*** It provides way to host machine learning project. It's mainly work with Apache project name spark.
- ***Core ML Tools:*** This is a ML framework makes ML available for application programming interface.
- ***Cortex:*** Cortex makes way to use Python, TensorFlow, PyTorch, Scikit-learn and others building machine learning model. It offers to use Docker for hosting environment using cortex.yml file.
- ***GoLearn:*** It's a ML library based on Go language by Google. Data loading & handling is more simpler than other in GoLearn.
- ***Gradio:*** It givesthe feature of customized UI for model training and prediction based service. Gradio support real-time interaction with models.
- ***H2O:*** In memory machine learning platform prepared for business purpose. Java, Python, R and Scala are allowed to interact with H2O.
- ***Oryx:*** Oryx use Hahoop distribution these are Apache Kafka, Apache Spark for machine learning model provides real-time feature.
- ***PyTorch Lightning:*** Wrapper of pyTorch that make it easy to use is a third party library.
- ***Scikit-learn:*** Makes math and science work more easier leveraging existing python packages like Numpy, SciPy and Matplotlib.
- ***Shogun:*** I is written in C++, but Java, Python, C#, Ruby, R & Matlab can also use it.
- ***Spark MLlib:*** Java is the primary language to work with MLLib is a library forApache Hadoop and Spark. It is faster and scalable.
- ***Weka:*** It has GUI feature for creating and training models without writing code.

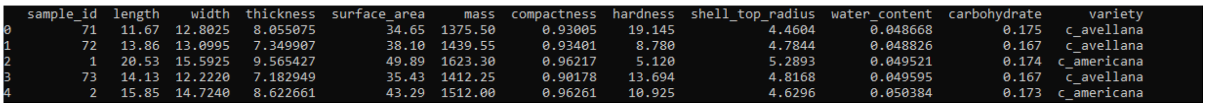
Here I select to use `**Scikit-Learn**` for this classification task. Scikit-Learn is powerful and robust library for machine learning in python. It is simple and has efficient tools for math and science work.

## Prepare datasets

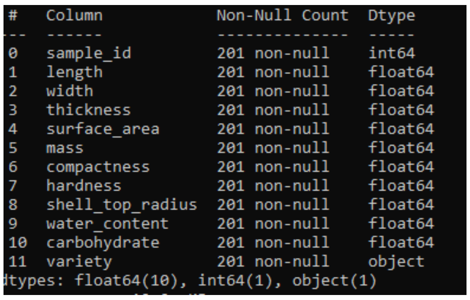
I found dataset and analyzing this data set I will prepare this for fitting in model. I'll show step of code for analyzing and preparing this dataset for our model.

I. First load our dataset using panda and view sample of our dataset, its column information and numerical summery of numeric column like standard deviation, mean, min & max value.

```
import pandas as pd
data = pd.read_csv('dataset.csv')
print(data.head())
print(data.info())
print(data.describe())
```

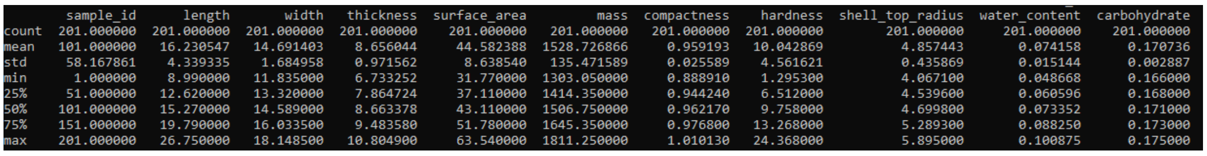


```
   sample_id  length    width  thickness  surface_area     mass  compactness  hardness  shell_top_radius  water_content  carbohydrate      variety
0         71   11.67  12.8025   8.055075         34.65  1375.50      0.93005    19.145            4.4604       0.048668         0.175   c_avellana
1         72   13.86  13.0995   7.349907         38.10  1439.55      0.93401     8.780            4.7844       0.048826         0.167   c_avellana
2          1   20.53  15.5925   9.565427         49.89  1623.30      0.96217     5.120            5.2893       0.049521         0.174  c_americana
3         73   14.13  12.2220   7.182949         35.43  1412.25      0.90178    13.694            4.8168       0.049595         0.167   c_avellana
4          2   15.85  14.7240   8.622661         43.29  1512.00      0.96261    10.925            4.6296       0.050384         0.173  c_americana
```



```
#   Column            Non-Null Count  Dtype
---  ------            --------------  -----
0   sample_id         201 non-null    int64
1   length            201 non-null    float64
2   width             201 non-null    float64
3   thickness         201 non-null    float64
4   surface_area      201 non-null    float64
5   mass              201 non-null    float64
6   compactness       201 non-null    float64
7   hardness          201 non-null    float64
8   shell_top_radius  201 non-null    float64
9   water_content     201 non-null    float64
10  carbohydrate      201 non-null    float64
11  variety           201 non-null    object
dtypes: float64(10), int64(1), object(1)
```

Code: https://github.com/Abir-Hosen/ml-assignment



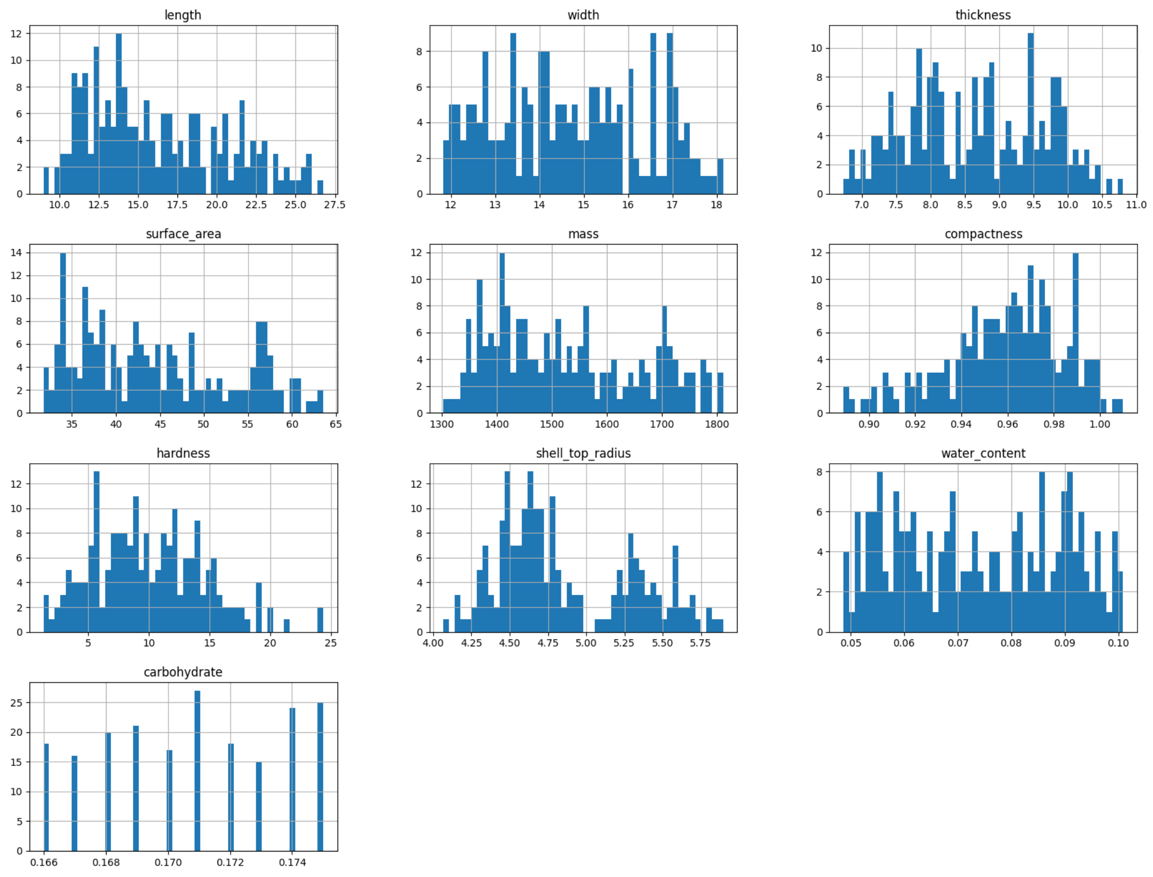
```
       sample_id      length       width   thickness  surface_area         mass  compactness    hardness  shell_top_radius  water_content  carbohydrate
count  201.000000  201.000000  201.000000  201.000000    201.000000   201.000000   201.000000  201.000000        201.000000     201.000000    201.000000
mean   101.000000   16.230547   14.691403    8.656044     44.582388  1528.726866     0.959193   10.042869          4.857443       0.074158      0.170736
std     58.167861    4.339335    1.684958    0.971562      8.638540   135.471589     0.025589    4.561621          0.435869       0.015144      0.002887
min      1.000000    8.990000   11.835000    6.733252     31.770000  1303.050000     0.888910    1.295300          4.067100       0.048668      0.166000
25%     51.000000   12.620000   13.320000    7.864724     37.110000  1414.350000     0.944240    6.512000          4.539600       0.060596      0.168000
50%    101.000000   15.270000   14.589000    8.663378     43.110000  1506.750000     0.962170    9.758000          4.699800       0.073352      0.171000
75%    151.000000   19.790000   16.033500    9.483580     51.780000  1645.350000     0.976800   13.268000          5.289300       0.088250      0.173000
max    201.000000   26.750000   18.148500   10.804900     63.540000  1811.250000     1.010130   24.368000          5.895000       0.100875      0.175000
```

II. Here `sample_id` is not the part of our data, first remove this.

```
data = data.drop(axis=1, columns=["sample_id"])
```

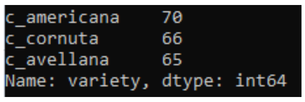
III. To get feel of the type of dataset lets plot a histogram for numerical values. Here I will use `matplotlib`.

```
import matplotlib.pyplot as plt
data.hist(bins=50, figsize=(20,15))
plt.show()
```



IV. `Variety` is the label of our dataset. It's is an object type data. Let's see how many category belongs to these label. For in depth

```
print(data['variety'].value_counts())
```



```
c_americana    70
c_cornuta      66
c_avellana     65
Name: variety, dtype: int64
```

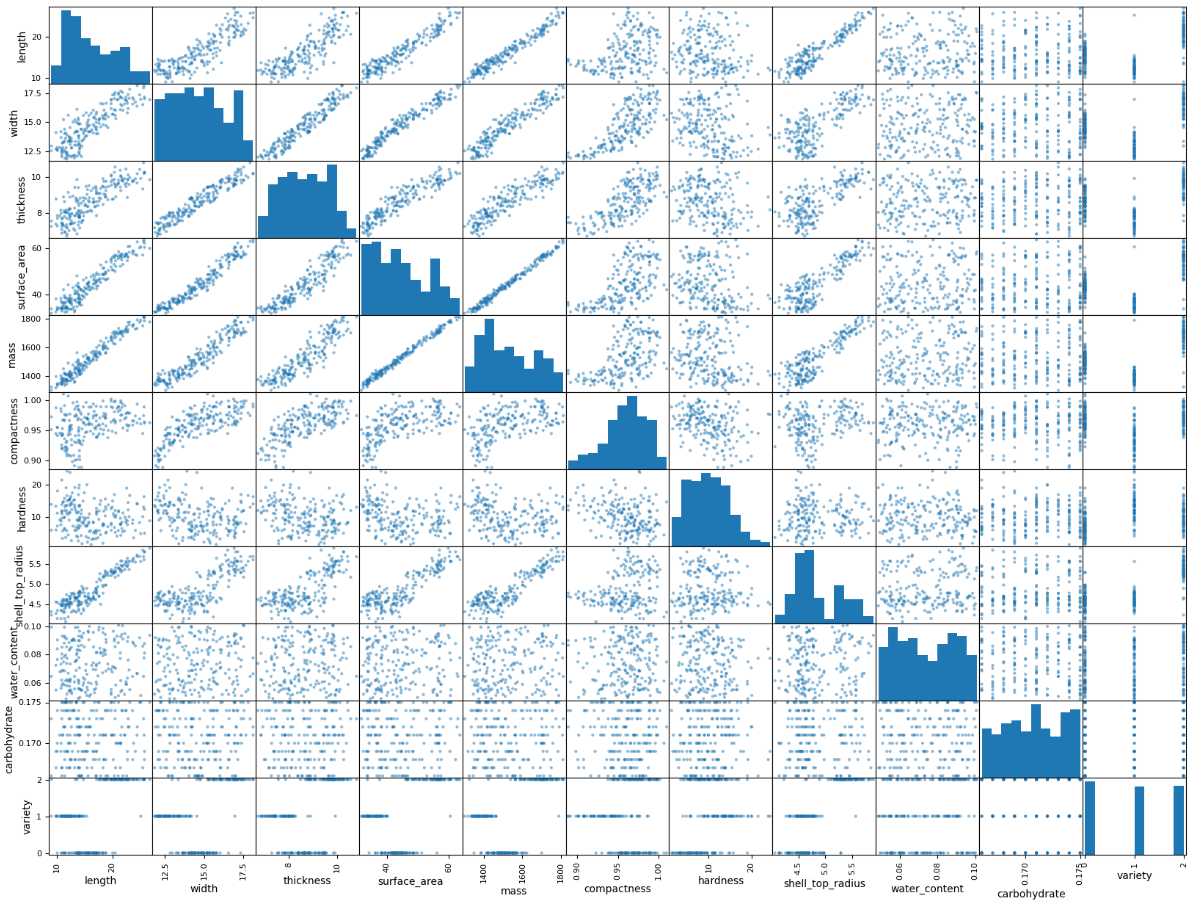
V. Let's have a sight on the correlation of the attributes of the datasets. As we are to find standard correlation coefficient then all data should be numerical. Then let's convert the target label object of `variety` to numerical attributes.

Code: https://github.com/Abir-Hosen/ml-assignment

```
from sklearn.preprocessing import LabelEncoder
encoded = LabelEncoder().fit_transform(data["variety"])
data["variety"] = encoded
```

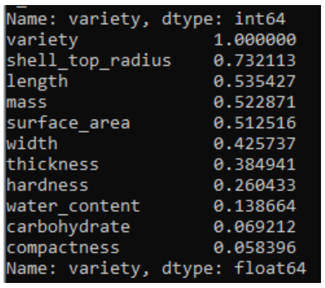
Then create the correlation matrix. We may also have a sight on graph.

```
from pandas.plotting import scatter_matrix
attributes = ["length", "width", "thickness", "surface_area", "mass", "com-
pactness", "hardness", "shell_top_radius", "water_content", "carbohydrate",
"variety"]
scatter_matrix(data[attributes], figsize=(20,15))
plt.show()
```



And have sight on numerical correlation of `variety`.

```
print(data.corr())
```



```
Name: variety, dtype: int64
variety             1.000000
shell_top_radius    0.732113
length              0.535427
mass                0.522871
surface_area        0.512516
width               0.425737
thickness           0.384941
hardness            0.260433
water_content       0.138664
carbohydrate        0.069212
compactness         0.058396
Name: variety, dtype: float64
```

Code: [https://github.com/Abir-Hosen/ml-assignment](https://github.com/Abir-Hosen/ml-assignment)

VI. Let’s split our dataset into training set and test set. First separate our feature and prediction on X Y.

```
X = data.drop(columns=["variety"]).values
y = data["variety"].values
```

VII. In few cases machine learning algorithms don’t perform well when input numerical attributes have very different scales. So let range these feature X.

```
from sklearn.preprocessing import StandardScaler
X = StandardScaler().fit_transform(X)
```

VIII. Then obtain train set and test set from them

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran-
dom_state=1, stratify=y)
```

Now our dataset is ready to feed on our model.

## Classification models

**Let’s apply our dataset in `KNN` (K Nearest Neighbor) model first.**

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train,y_train)
```

*Now understand how KNN model work.* KNN is used to solve both classification and regression problem. Here is the step KNN follow to solve problems.

1. Load the data.
2. Initialize k the number we chose our neighbor to compare our test value (query example) in particular.
    a. Calculate the distance between the query example and the current example
    b. Add the distance and the index in an order list
3. Sort (ascending) the ordered list by distance.
4. Select the k entries from top of list
5. Return the Mode of the k labels as it is classification. Otherwise mean in regression term.

**Let’s apply our dataset in another mode. I select `Random Forest` for now.**

```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(criterion='gini', n_estimators=50,
random_state=5, n_jobs=2)
forest.fit(X_train, y_train)
```

*Now understand how ` Random Forest` model work.* Random forest is based on `Decision Tree` model, where more than one tree is created using Bagging (Bootstrapping+ Aggregations). Let’s see the steps it follows.

1. From dataset of K records model takes n number of records randomly for T times.
2. T number of times decision tree is created with their selected records.
3. Each decision tree generates an output.
4. Mode of all output is the required labels as it is classification. Otherwise mean of output, in regression term.

Code: https://github.com/Abir-Hosen/ml-assignment

## Estimate future performance

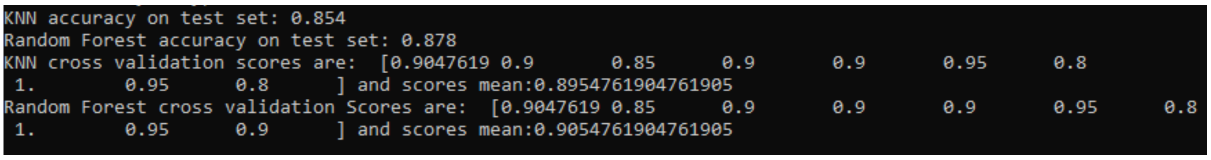
Let’s test our model with test set, then we will see 10 fold cross validation.

```
from sklearn.metrics import accuracy_score
y_pred_knn = knn.predict(X_test)
print("KNN accuracy on test set: %.3f" % accuracy_score(y_test, y_pred_knn))

y_pred_forest = forest.predict(X_test)
print("Random Forest accuracy on test set: %.3f" % accuracy_score(y_test,
y_pred_forest))
```

Now see the result of 10 fold cross validation for KNN and Random Forest model.

```
from sklearn.model_selection import cross_val_score
import numpy as np
knn_cv_scores = cross_val_score(knn, X, y, cv=10)
print("KNN cross validation scores are: ", knn_cv_scores, "and scores
mean:{}".format(np.mean(knn_cv_scores)))
forest_cv_scores = cross_val_score(forest, X, y, cv=10)
print("Random Forest cross validation Scores are: ", forest_cv_scores, "and
scores mean:{}".format(np.mean(forest_cv_scores)))
```



```
KNN accuracy on test set: 0.854
Random Forest accuracy on test set: 0.878
KNN cross validation scores are:  [0.9047619 0.9       0.85      0.9       0.9       0.95      0.8
 1.        0.95      0.8       ] and scores mean:0.8954761904761905
Random Forest cross validation Scores are:  [0.9047619 0.85      0.9       0.9       0.9       0.95      0.8
 1.        0.95      0.9       ] and scores mean:0.9054761904761905
```

## Report analysis

From our test set and mostly from 10 fold cross validation; result is not same but very similar. For KNN cross validation score is 0.895 and random forest score is 0.905. Random Forest gives better result than KNN in validation score. In our model of Random Forest n_estimator=50, means it create 50 Decision Tree model in time of bagging and use Gini Impurity for measurement of split quality.

On the other hand our model KNN uses 5 neighbors for estimating mode of related feature.

KNN depends on the quality of data. It is sensitive to the scale of data and irrelevant feature. On the other hand Random Forest did not train model with all data at a time. It does not create decision tree model with all data set at a time. It picks data randomly from main data set and gives mode of these results. Thus accuracy is little bit higher in Random Forest.

Though accuracy mostly depend on correct feature selection and quality of dataset, Random forest is much better in so many times than KNN in many motives.

Code: https://github.com/Abir-Hosen/ml-assignment