In [1]:

```
## Import library
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
# load data
dataset = pd.read_csv("C:/Users/s2759/Downloads/bill_authentication/bill_authentication.csv")
print(dataset.head())
# split data
from sklearn.model_selection import train_test_split
X = dataset.iloc[:, 0:4].values
y = dataset.iloc[:, 4].values
# split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
   Variance  Skewness  Curtosis  Entropy  Class
0   3.62160    8.6661   -2.8073 -0.44699      0
1   4.54590    8.1674   -2.4586 -1.46210      0
2   3.86600   -2.6383    1.9242  0.10645      0
3   3.45660    9.5228   -4.0112 -3.59440      0
4   0.32924   -4.4552    4.5718 -0.98880      0
```

LogisticRegression

In [2]:

```
# fit model
from sklearn import linear_model
Classifier = linear_model.LogisticRegression()
Classifier.fit(X_train, y_train)
# make predictions for test data
y_pred = Classifier.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
cm = confusion_matrix(y_test, predictions)
print(cm)
accuracy=accuracy_score(y_test,predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[225   7]
 [  0 180]]
Accuracy: 98.30%
```

DecisionTree Classifier

In [3]:

```
# fit model
from sklearn.tree import DecisionTreeClassifier
Classifier = DecisionTreeClassifier()
Classifier.fit(X_train, y_train)
# make predictions for test data
y_pred = Classifier.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
cm = confusion_matrix(y_test, predictions)
print(cm)
accuracy=accuracy_score(y_test,predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[226   6]
 [  4 176]]
Accuracy: 97.57%
```

GradientBoosting Classifier

In [4]:

```
# fit model
from sklearn.ensemble import GradientBoostingClassifier
Classifier = GradientBoostingClassifier(n_estimators=100,max_depth=5)
Classifier.fit(X_train, y_train)
# make predictions for test data
y_pred = Classifier.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
cm = confusion_matrix(y_test, predictions)
print(cm)
accuracy=accuracy_score(y_test,predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
D:\anaconda3\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning:
numpy.core.umath_tests is an internal NumPy module and should not be imported. It will be removed
in a future NumPy release.
  from numpy.core.umath_tests import inner1d
```

```
[[228   4]
 [  2 178]]
Accuracy: 98.54%
```

KNeighbors Classifier

In [5]:

```
# fit model
from sklearn.neighbors import KNeighborsClassifier
Classifier = KNeighborsClassifier(n_neighbors=3)
Classifier.fit(X_train, y_train)
# make predictions for test data
y_pred = Classifier.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
cm = confusion_matrix(y_test, predictions)
print(cm)
accuracy=accuracy_score(y_test,predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[231   1]
 [  0 180]]
Accuracy: 99.76%
```

svm Classifier

In [6]:

```
# fit model
from sklearn import svm
Classifier = svm.LinearSVC(random_state=20)
Classifier.fit(X_train, y_train)
# make predictions for test data
y_pred = Classifier.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
cm = confusion_matrix(y_test, predictions)
print(cm)
```

```
accuracy=accuracy_score(y_test,predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[227   5]
 [  0 180]]
Accuracy: 98.79%
```

Naive_bayes

In [7]:

```
# fit model
from sklearn.naive_bayes import GaussianNB
Classifier = GaussianNB()
Classifier.fit(X_train, y_train)
# make predictions for test data
y_pred = Classifier.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
cm = confusion_matrix(y_test, predictions)
print(cm)
accuracy=accuracy_score(y_test,predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[199  33]
 [ 36 144]]
Accuracy: 83.25%
```

RandomForest Classifier

In [8]:

```
# fit model
from sklearn.ensemble import RandomForestRegressor
Classifier = RandomForestRegressor(n_estimators=20, random_state=0)
Classifier.fit(X_train, y_train)
# make predictions for test data
y_pred = Classifier.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
cm = confusion_matrix(y_test, predictions)
print(cm)
accuracy=accuracy_score(y_test,predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[229   3]
 [  0 180]]
Accuracy: 99.27%
```

XGB Classifier

In [9]:

```
# fit model
from xgboost import XGBClassifier
Classifier = XGBClassifier()
Classifier.fit(X_train, y_train)
# make predictions for test data
y_pred = Classifier.predict(X_test)
predictions = [round(value) for value in y_pred]
# evaluate predictions
accuracy=accuracy_score(y_test,predictions)
cm = confusion_matrix(y_test, predictions)
print(cm)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
[[231   1]
 [  1 179]]
Accuracy: 99.51%
```

```
D:\anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: The truth v
alue of an empty array is ambiguous. Returning False, but in future this will result in an error.
Use `array.size > 0` to check that an array is not empty.
  if diff:
```

Neural Network using Keras for Classification

In [10]:

```
from keras import Sequential
from keras.layers import Dense
classifier = Sequential()
#First Hidden Layer
classifier.add(Dense(4, activation='relu', kernel_initializer='random_normal', input_dim=4))
#Second  Hidden Layer
classifier.add(Dense(4, activation='relu', kernel_initializer='random_normal'))
#Output Layer
classifier.add(Dense(1, activation='sigmoid', kernel_initializer='random_normal'))
#Compiling the neural network
classifier.compile(optimizer ='adam',loss='binary_crossentropy', metrics =['accuracy'])
classifier.fit(X_train,y_train, batch_size=10,epochs=5)
# eval_model=classifier.evaluate(X_train, y_train)
y_pred=classifier.predict(X_test)
y_pred =(y_pred>0.5)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy=accuracy_score(y_test,y_pred)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
D:\anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of the second
argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated
as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
WARNING:tensorflow:From D:\anaconda3\lib\site-
packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_
int32 (from tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/5
960/960 [==============================] - 1s 663us/step - loss: 0.6909 - acc: 0.5510
Epoch 2/5
960/960 [==============================] - 0s 154us/step - loss: 0.6497 - acc: 0.8437
Epoch 3/5
960/960 [==============================] - 0s 193us/step - loss: 0.4937 - acc: 0.9479
Epoch 4/5
960/960 [==============================] - 0s 216us/step - loss: 0.2960 - acc: 0.9604
Epoch 5/5
960/960 [==============================] - 0s 184us/step - loss: 0.1715 - acc: 0.9687
[[222  10]
 [  1 179]]
Accuracy: 97.33%
```