Simple Machine Learning Classification Demo

In [52]:
```
import sklearn
import csv
import random
```

The data is from UCI Data repository. The data contains bank-note image data that has been transfromed into 5 features.
https://archive.ics.uci.edu/ml/datasets/banknote+authentication (https://archive.ics.uci.edu/ml/datasets/banknote+authentication)

The goal is to try to classify the images into forged and genuine banknote images. The images were taken with a digital camera and wavelet tranformation was used to extract the features. All the data are contious data except for the classification data which is an integer.

Attribute Information:

1. variance of Wavelet Transformed image
2. skewness of Wavelet Transformed image
3. curtosis of Wavelet Transformed image
4. entropy of image
5. class

In [9]:
```
# create an empty list to append the data. Read data which comes in as a string and convert to float.
#Also convert classification column - last column - to TRUE/FALSE

data = []
with open('data_banknote_authentication.txt') as f:
    all_line = csv.reader(f, delimiter = ",")
    for line in all_line:
        li = [1] + [float(d) for d in line]
        li[-1] = li[-1] > 0
        data.append(li)


len(data)
clean_data = []
for d in data:
    if 'NA' in d or None in d:
        continue
    else:
        clean_data.append(d)


#check to see if data is clean
len(data) == len(clean_data)
```

Out[9]: True

In [10]:
```
data[0] #view a sample
```

Out[10]: [1, 3.6216, 8.6661, -2.8073, -0.44699, False]

In [17]:
```
total_samples = len(data)
```

In [18]:
```
posi = sum([d[-1] for d in data] )
```

In [19]:
```
percent_pos = posi/total_samples
```

In [50]:
```
print(f'Total samples: {total_samples}, Positive samples: {posi}, Percent positive: {percent_pos:.2f}
%')
```

```
Total samples: 1372, Positive samples: 610, Percent positive: 0.44%
```

Run through analysis without splitting into train and test for illustration

In [39]:
```
X = [d[:-1] for d in data]
y = [d[-1] for d in data]
```

In [40]:
```
y[100]
```

Out[40]:
```
False
```

In [25]:
```
from sklearn import linear_model
```

In [41]:
```
model = linear_model.LogisticRegression()
model.fit(X,y)
```

Out[41]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [42]:
```
predictions = model.predict(X)
```

In [51]:
```
accuracy = sum(predictions == y)/len(y) #note this is using full sample and not advisable
print(f'Accuracy : {accuracy:.4f}%')
```

```
Accuracy : 0.9898%
```

Analysis splitting data into train and test set using a 50% ration.

In [54]:
```
random.shuffle(data)
Xs = [d[:-1] for d in data]
ys = [d[-1] for d in data]
```

In [55]:
```
X_train = Xs[:total_samples//2]
X_test = Xs[total_samples//2:]

y_train = ys[:total_samples//2]
y_test = ys[total_samples//2:]
```

In [57]:
```
total_samples, len(X_train), len(X_test)
```

Out[57]:
```
(1372, 686, 686)
```

In [58]:
```
models = linear_model.LogisticRegression()
models.fit(X_train,y_train)
```

Out[58]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [60]:
```
preds = model.predict(X_train)
preds_test = model.predict(X_test)

accus = sum(preds == y_train)/len(y_train)
accutest = sum(preds_test == y_test)/len(y_test)

print(f'Accuracy training set : {accus:.4f}%')
print(f'Accuracy test set : {accutest:.4f}%')
```

```
Accuracy training set : 0.9942%
Accuracy test set : 0.9854%
```

Accuracy for the test set was almost as high as for the analysis without any test data, however the result is a better indicator of performance

In [ ]:
```

```