Name: Dushyant Singh ¶

Sec: A

Reg: 201700066

Importing necessary libraries

In [6]:

```
import numpy as np
import pandas as pd
```

In [7]:

```
df = pd.read_csv("Bill.csv")
df.shape
```

Out[7]:

```
(1372, 5)
```

Splitting the continuous dataset using stratify attribute so that we haveeven distribution of classes in both the training as well as testing set

In [8]:

```
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

In [9]:

```
#Splitting data into train & test sets; 30% for testing and 70% for training
X = df.loc[:, df.columns != 'Class']
y = df.loc[:, df.columns == 'Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, stratify = y, ra
```

Applying SVM for binary classification

In [14]:

```
from sklearn import svm
```

In [15]:

```
classify = svm.SVC(kernel='linear') # Applying Linear Kernel
classify.fit(X_train, y_train)
y_pred = classify.predict(X_test)
```

```
c:\users\sdjsr\appdata\local\programs\python\python37\lib\site-packages\skle
arn\utils\validation.py:760: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [16]:

```
metrics.accuracy_score(y_test, y_pred)
```

Out[16]:

```
0.9878640776699029
```

Linear classification gives 98.78% accuracy which is pretty good

The hyperparameters that may require tuning are kernel, regularization parameter & gamma

We may use polynomial, and radial basis function (RBF) for kernels or sigmoid function. Polynomial and RBF are useful for non-linear hyperplane. Polynomial and RBF kernels compute the separation line in the higher dimension.

A lower value of Gamma will loosely fit the training dataset, whereas a higher value of gamma will exactly fit the training dataset, which causes over-fitting.

Applying non linear SVM and evaluating

In [23]:

```
clf = svm.NuSVC(gamma='auto') # Applying non linear SVM
```

In [24]:

```
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
c:\users\sdjsr\appdata\local\programs\python\python37\lib\site-packages\skle
arn\utils\validation.py:760: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [25]:

```
metrics.accuracy_score(y_test, y_pred)
```

Out[25]:

```
1.0
```

We get a 100 percent accuracy

In [27]:

```
from sklearn.svm import SVC
clf = SVC(kernel='rbf') # RBF kernel
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
c:\users\sdjsr\appdata\local\programs\python\python37\lib\site-packages\skle
arn\utils\validation.py:760: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [28]:

```
metrics.accuracy_score(y_test, y_pred)
```

Out[28]:

```
0.9975728155339806
```

In [29]:

```
clf = SVC(kernel='poly') # polynomial kernel
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
c:\users\sdjsr\appdata\local\programs\python\python37\lib\site-packages\skle
arn\utils\validation.py:760: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [30]:

```
metrics.accuracy_score(y_test, y_pred)
```

Out[30]:

```
0.9805825242718447
```

In [31]:

```
clf = SVC(kernel='sigmoid') # sigmoid kernel
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
c:\users\sdjsr\appdata\local\programs\python\python37\lib\site-packages\skle
arn\utils\validation.py:760: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_sample
s, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [32]:

```
metrics.accuracy_score(y_test, y_pred)
```

Out[32]:

```
0.662621359223301
```

We can see that sigmoid kernel had the lowest accuracy

In [ ]:

```
```