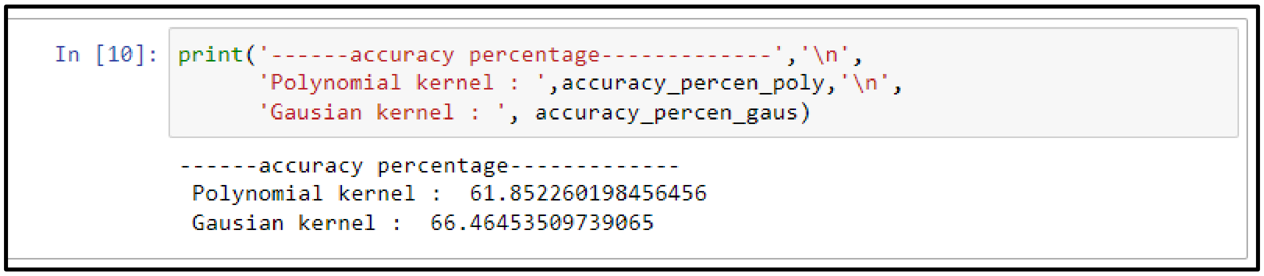
PROBLEM 1:



```
In [10]: print('------accuracy percentage------------','\n',
               'Polynomial kernel : ',accuracy_percen_poly,'\n',
               'Gausian kernel : ', accuracy_percen_gaus)

         ------accuracy percentage------------
          Polynomial kernel :  61.852260198456456
          Gausian kernel :  66.46453509739065
```

SVM (support vector machine) works on the concept of maximum margin. SVM creates a decision boundary using a subset from the training data which are called as the support vectors. Polynomial and gaussian kernel are the most used SVMs.

Trained SVM for 2 classification models:

1. Polynomial kernel with parameter 2 :
   a. Represents similarity of training data samples in the feature space over polynomials of original variables

      $$K(x, y) = (x^{\mathsf{T}} y + c)^d$$

2. Gaussian kernel with parameter 2
   a. The value depends on the distance from the origin or from some point

      $$K(X_1, X_2) = exponent(-\gamma \|X_1 - X_2\|^2)$$

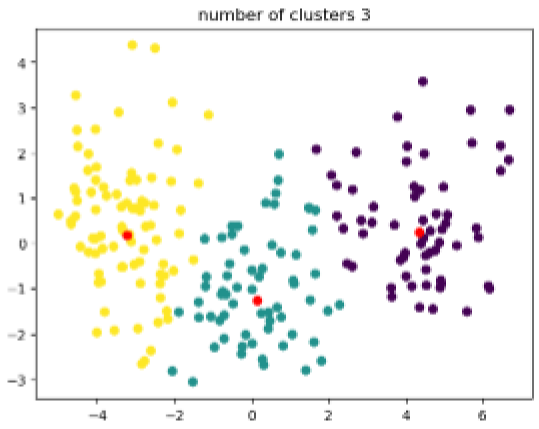
      ||X1 — X2 || = Euclidean distance between X1 & X2

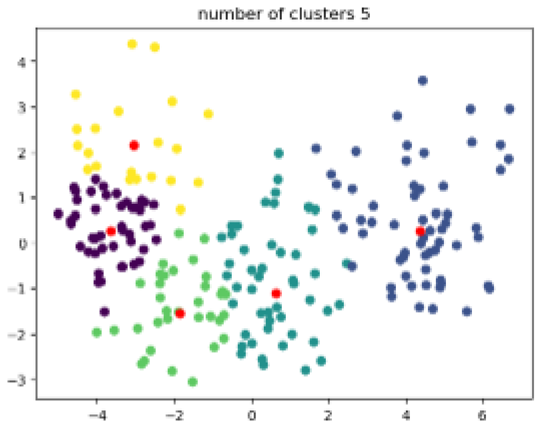
Using the formula mentioned in the problem

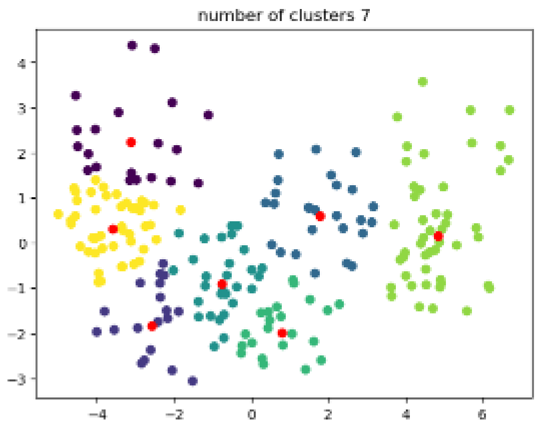
A = | T intersection P | / | T union P |

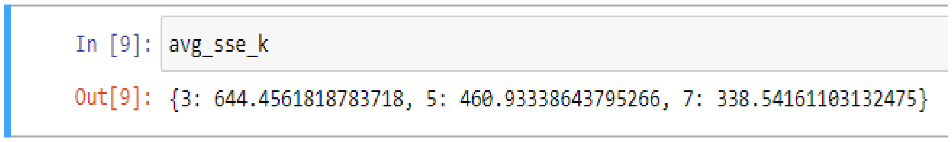
We can see from the results that the Gaussian kernel performs better than the polynomial kernel. This is because the kernel is more generalized than polynomial kernel and is not restricted to the shape of the boundary and can take any shape.

PROBLEM 2:









```
In [9]: avg_sse_k
Out[9]: {3: 644.4561818783718, 5: 460.93338643795266, 7: 338.54161103132475}
```

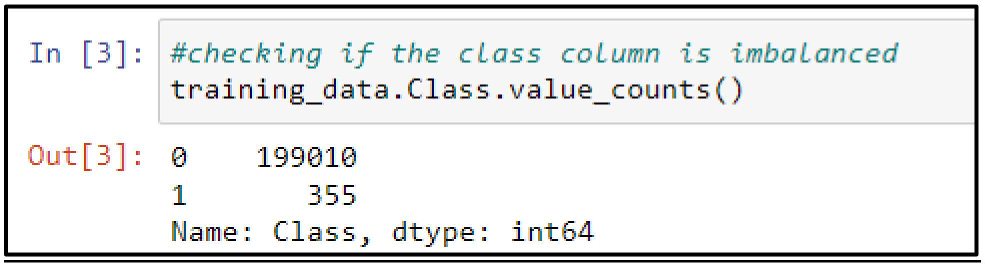
Average SSE vlaues for clusters 3,5,7 averaged over 10 initialisations and max 100 iterations and change in SSE >=0.001 are:

Number of clusters	SSE
3	644.45
5	460.93
7	338.54

We can see that the value of average SSE is decreasing with increase in the number of clusters.

PROBLEM 3:

Class imbalance in the training data:



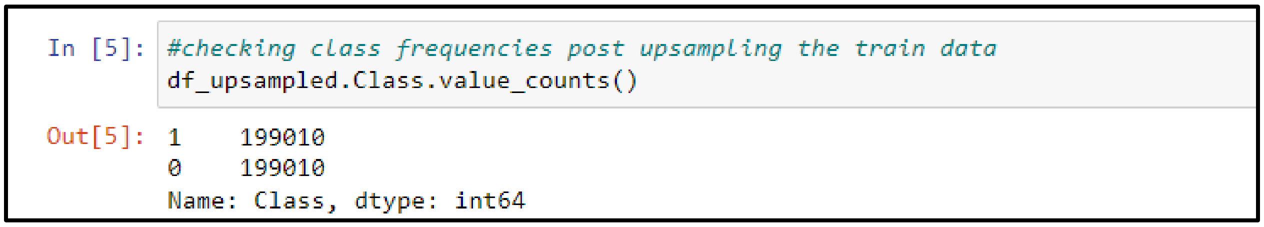
```
In [3]:  #checking if the class column is imbalanced
         training_data.Class.value_counts()

Out[3]:  0    199010
         1       355
         Name: Class, dtype: int64
```

We can see that the class 0 is having very high count than the class 1.

To address this issue I have tried to balance the data using the following techniques and then tested the result using the fscore metric.

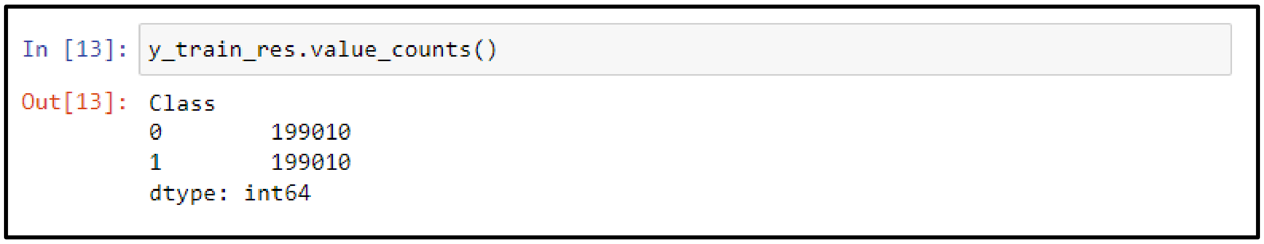
1. Upsampling :
    a. Adds copies to the minority class
    b. Good when we don’t have a large amount of data
    c. We have upsampled the minonrity class 1 to match the count of majority class 0



```
In [5]:  #checking class frequencies post upsampling the train data
         df_upsampled.Class.value_counts()

Out[5]:  1    199010
         0    199010
         Name: Class, dtype: int64
```

2. SMOTE :
    a. Synthetic minority oversampling technique
    b. This algorithm randomly picks a point from the minority class and computes the k-nearest neighbours for the point
    c. Dummy points are then added between this point and the neighbours
    d. Step is repeated until the data is balanced

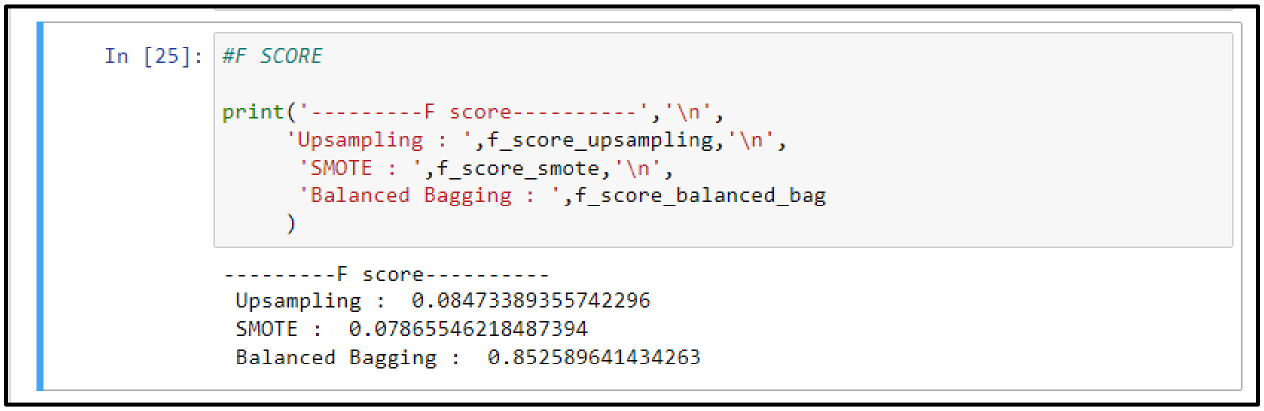


```
In [13]:  y_train_res.value_counts()

Out[13]:  Class
          0        199010
          1        199010
          dtype: int64
```

3. Balanced bagging
    a. Stands for bootstrap aggregating
    b. Involves generating n different bootstrap training samples with replacement
    c. Model is trained on each bootstrapped algorithm separately
    d. Predictions are aggregated at the end
    e. Bagging reduces overfitting

<u>F score:</u>



```
In [25]: #F SCORE

         print('---------F score----------','\n',
              'Upsampling : ',f_score_upsampling,'\n',
               'SMOTE : ',f_score_smote,'\n',
               'Balanced Bagging : ',f_score_balanced_bag
              )
```

```
---------F score----------
 Upsampling :  0.08473389355742296
 SMOTE :  0.07865546218487394
 Balanced Bagging :  0.852589641434263
```

We can see that Balanced bagging classifier gives a better fscore than the other 2 techniques.

<u>References:</u>

https://ieeexplore.ieee.org/document/9362471

https://www.analyticsvidhya.com/blog/2022/05/handling-imbalanced-data-with-imbalance-learn-in-python/

https://ieeexplore.ieee.org/document/9775866