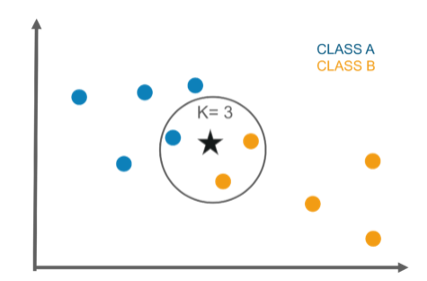
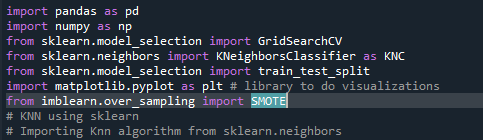
**k nearest neighbor(KNN)**

**KNN**can be used for **Classification**or**Regression**problem, but in general.The KNN algorithm works to **classify**new data based on its proximity to **K-neighbors** (training data). So if the new data is surrounded by training data that has Class 1, it can be concluded that the new data is included in Class 1. To make it easier to understand, see the **illustration below**



**Python code:**

Loading several libraries that will be used to do the analysis in this documentation



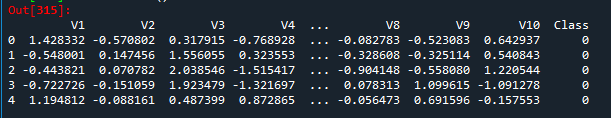
Load the dataset to be used, dataset contains historical data from credit cards information, to use various predictive models to see how accurate they are in detecting whether a transaction is a **normal payment or a fraud**



Let’s see some general information from the dataset



**Output:**



Drop Time and Amount columns as they are not relevant for prediction purpose



checking the target features



Looks like the target feature is balanced

Value 0 for normal transaction

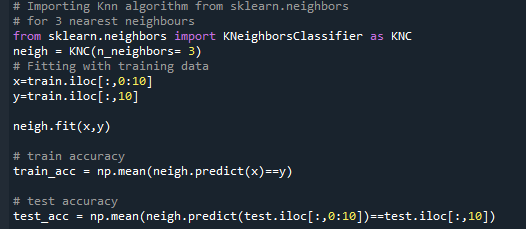
Value 1 for fraud transaction



Split the data in train and test

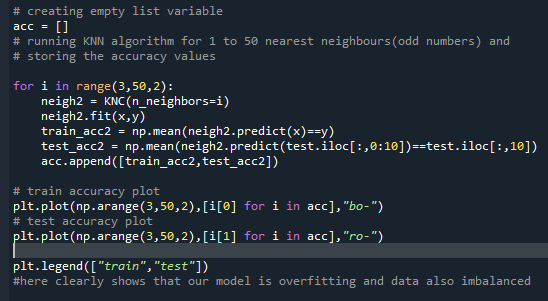


To make a model that will be used to predict transaction. In this case, we will use KNN algorithm

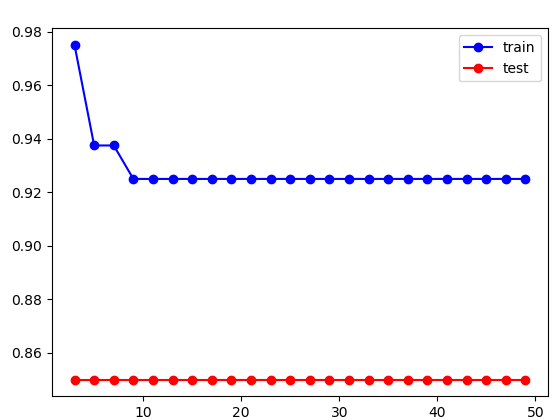


Here train accuracy is high and test accuracy is low. It proves that the model is more biased towards majority class. So, it proves that this is not the best model.

Running KNN algorithm for 1 to 50 nearest neighbours and storing the accuracy values

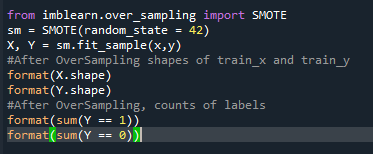


Here, the plot clearly shows that our model is overfitting, so use **imbalanced data handling techniques**to improve test accuracy

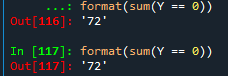


**SMOTE algorithm:**

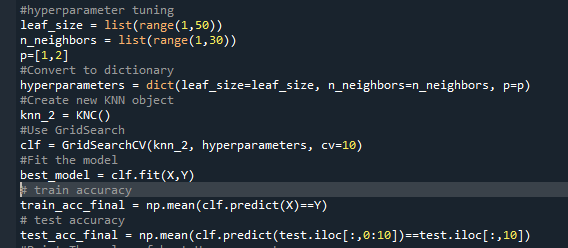
SMOTE Algorithm has oversampled the minority instances and made it equal to majority class. Both categories have equal amount of records



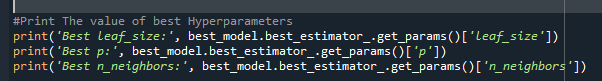
Now our data is balanced



Let's try to use Hyperparameter Tuning to Improve Model Performance.



Now test accuracy is high compare to train accuracy, so our model is best fit model



From GridSearch, it can be seen that the best number of leaf\_size is 1 while the optimal distance method is Manhattan or p = 1 and the most optimal number of K is 2

