# Machine Learning and Deep Learning - HW1 Report

**Hadi Nejabat s246601**

## 1. Introduction

Our task will be implementing a simple, yet powerful classification algorithm called K-Nearest-Neighbors (KNN). The KNN algorithm is a robust and versatile classifier that is often used as a benchmark for more complex classifiers such as Artificial Neural Networks (ANN) and Support Vector Machines (SVM). Despite its simplicity, KNN can outperform more powerful classifiers and is used in a variety of applications such as economic forecasting, data compression and genetics.

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given "unseen" observation. Similarity is defined according to a distance metric between two data points. A popular choice is the Euclidean distance given by

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

but other measures can be more suitable for a given setting and include the Manhattan, Chebyshev and Hamming distance. More formally, given a positive integer K, an unseen observation x and a similarity metric **d**, KNN classifier performs the following two steps:

- It runs through the whole dataset computing **d** between x and each training observation. We'll call the K points in the training data that are closest to x the set A. Note that K is usually odd to prevent tie situations.
- It then estimates the conditional probability for each class, that is, the fraction of points in A with that given class label.

$$P(y = j|X = x) = \frac{1}{K}\sum_{i \in \mathcal{A}} I(y^{(i)} = j)$$

Finally, our input x gets assigned to the class with the largest probability.

## 1.1. Support Vector Machine

in machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

## 1.2. Radial Basis Function Kernel

Radial Basis Function (RBF) is a commonly used kernel in SVC:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$$

where ||x−x'||^2 is the squared Euclidean distance between two data points x and x'. it is only important to know that an SVC classifier using an RBF kernel has two parameters: gamma and C.

### 1.2.1. Gamma:

gamma is a parameter of the RBF kernel and can be thought of as the 'spread' of the kernel and therefore the decision region. When gamma is low, the 'curve' of the decision boundary is very low and thus the decision region is very broad. When gamma is high, the 'curve' of the decision boundary is high, which creates islands of decision-boundaries around data points. We will see this very clearly below.

#### 1.2.2. C:

C is a parameter of the SVC learner and is the penalty for misclassifying a data point. When C is small, the classifier is okay with misclassified data points (high bias, low variance). When C is large, the classifier is heavily penalized for misclassified data and therefore bends over backwards avoid any misclassified data points (low bias, high variance).

# 2. Matrial and Methods

## 2.1. Dataset

### 2.1.1. Data Sources

We will be working on Wine dataset from Sklearn repository. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The attributes are Alcohol, Malic acid, Ash, Alkalinity of ash, Magnesium, Total phenols, Flavonoids, Nonflavonoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines, Proline. All attributes are continuous. In a classification context, this is a well posed problem with "well behaved" class structures.

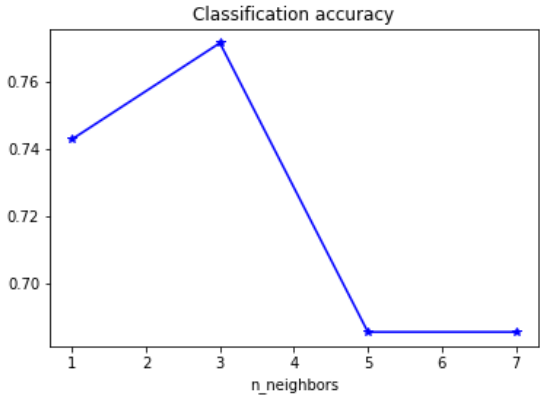
### 2.1.2. Data Preparation

The data set is loaded by `sklearn.datasets.load_wine(*, return_X_y=False, as_frame=False)`. For our task it is only needed to select the first two attributes for a 2D representation of images. Following that we Randomly split the data into train, validation and test sets using `(sklearn.model_selection import train_test_split)` for two separate times to be able to split the dataset into 3 datasets with determined proportions.

## 2.2. Approach

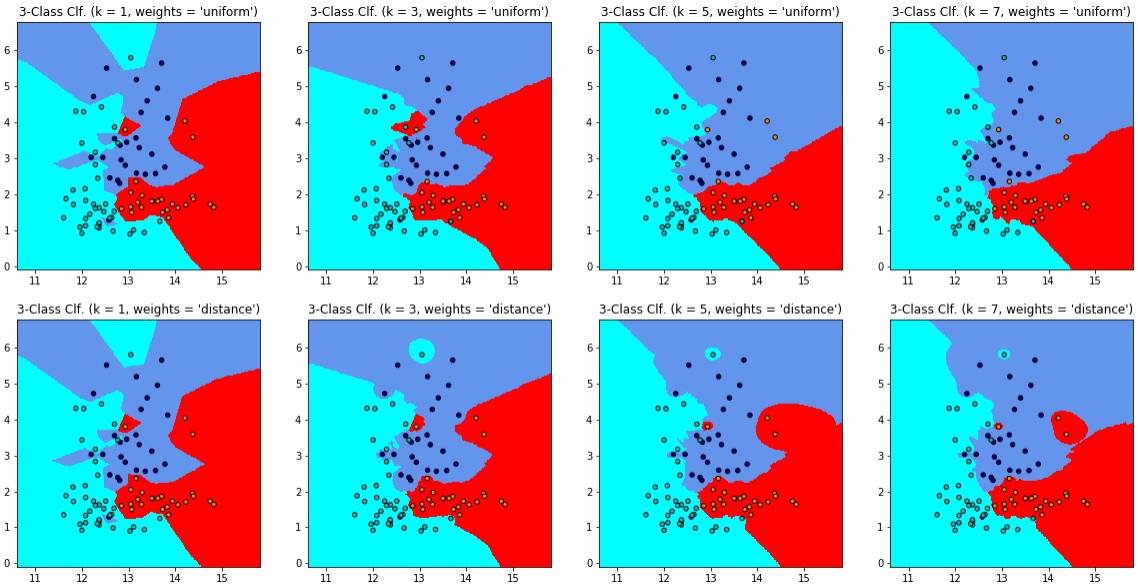
### 2.3.1. Implementing KNN and exploring results

Now we go ahead with our classifier. We'll be using scikit-learn to train a KNN classifier and evaluate its performance on the dataset. We must train our model with different values of K (K= [1,3,5,7]). So, we perform an iteration and predict the results for every value of initialization parameter. Our metric for this task is the accuracy of the classier on the evaluation set.



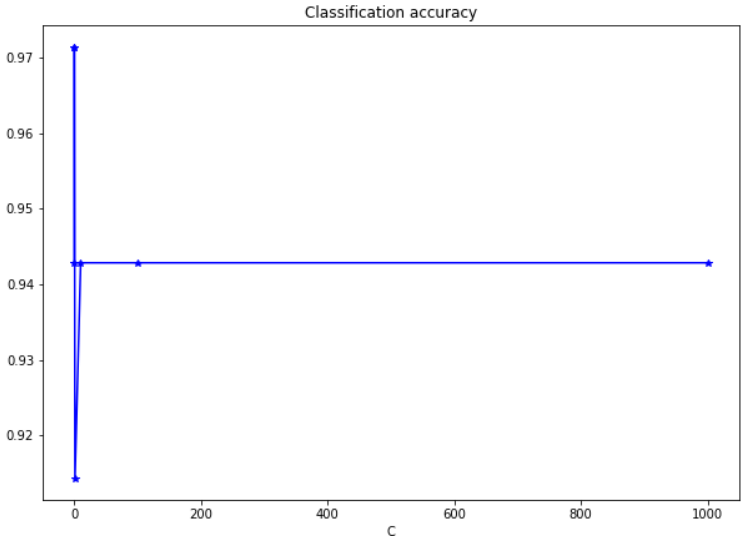

The basic Nearest Neighbors classification uses uniform weights: that is, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. Under some circumstances, it is better to weight the neighbors such that nearer neighbors contribute more to the fit. This can be accomplished through the **weights** keyword. The default value, **weights = 'uniform'**, assigns uniform weights to each neighbor.

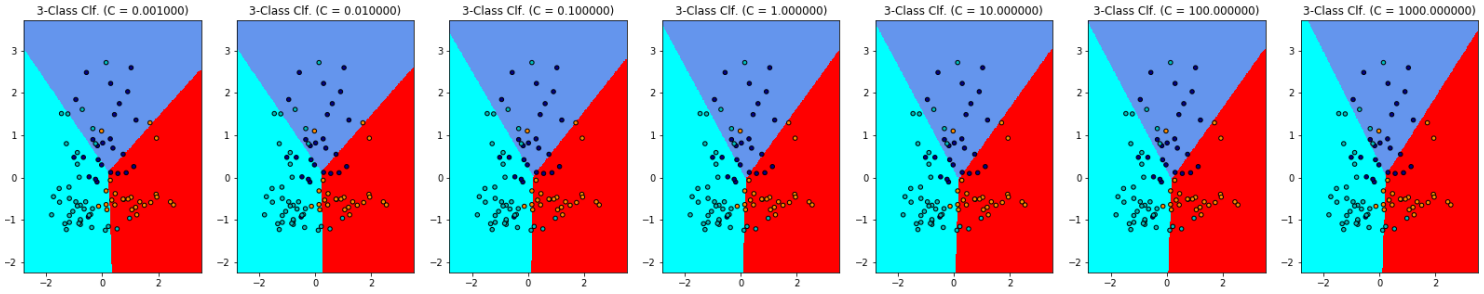
**weights = 'distance'** assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied to compute the weights.




### 2.3.2. Implementing SVM

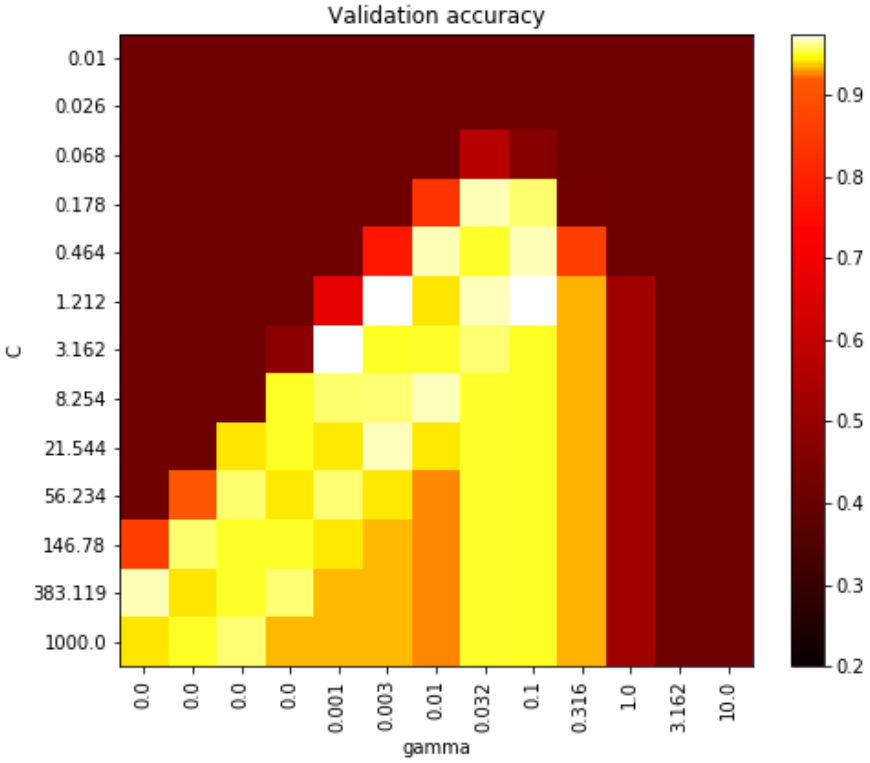
We'll be using scikit-learn to train a Linear SVC and evaluate its performance on the dataset. We must train our model with different values of C (C= [0.001,0.01,0.1,1,10,100,1000]). The procedure is like before. The results are as follows:



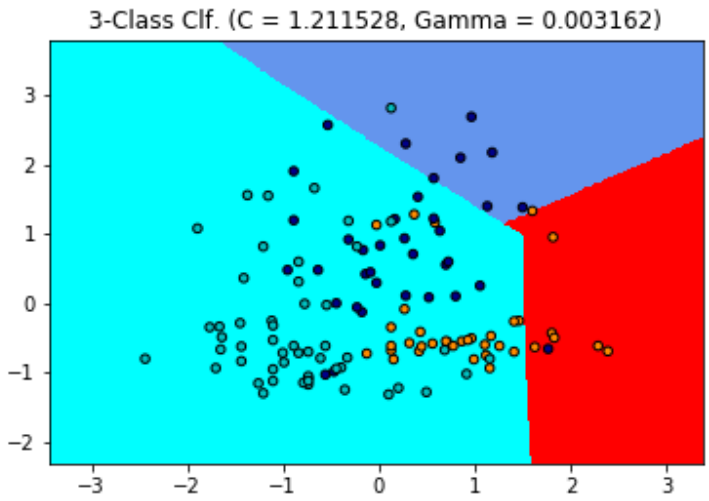



### 2.3.3. Grid Search & Cross-Validation for RBF Kernel

Following the given explanations about RBF Kernel. Best approach to achieve satisfactory results is to perform a grid search where multiple values of both C and Gamma are taken into account simultaneously and the best set of hyperparameters are selected among them. To implement this, we use “GridSearchCV” from sklearn library. Moreover, To visualize this method we use a heat-map as shown below:



The best parameters are **C = 1.211**, **gamma = 0.0038** with a **Validation Accuracy of 97.5%.**




## 3. Conclusion And Comparison between KNN & SVM:

one of the most valuable features of the K-nearest neighbor algorithm is that is simple to understand and easy to implement. With considerably little training time, it can be a useful tool for analysis of some data set you are planning to run more complex algorithms on. Furthermore, KNN works just as easily with multiclass data sets whereas other algorithms are hardcoded for the binary setting.

Yet, KNN can suffer from skewed class distributions. For example, if a certain class is very frequent in the training set, it will tend to dominate the majority voting of the new example (large number = more common). Finally, the accuracy of KNN can be severely degraded with high-dimension data because there is little difference between the nearest and farthest neighbor.

On the other hand, SVM can be very efficient, because it uses only a subset of the training data, only the support vectors; while Working very well on smaller data sets, on non-linear data sets and high dimensional spaces. Also, it is very effective in cases where number of dimensions is greater than the number of samples. And it is not very sensitive to overfitting. Nevertheless, Training time is high when we have large data sets. Moreover, When the data set has more noise (i.e. target classes are overlapping) SVM doesn't perform well.

The said above can also be spotted in our model and results of Accuracy of the validations set. Where in our case using the SVC with RBF kernel after hyperparameter tuning boosted the performance and resulted in better results.

## References:


1- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html
2- https://scikit-learn.org/stable/modules/neighbors.html
3- https://datascience.stackexchange.com/questions/4943/intuition-for-the-regularization-parameter-in-svm
4- https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200
5- https://chrisalbon.com/machine_learning/support_vector_machines/svc_parameters_using_rbf_kernel/
6- Apostolidis-Afentoulis, Vasileios, and Konstantina-Ina Lioufi. 2015. “SVM classification with linear and RBF kernels”. *http://www. academia. edu/13811676*