Report of Classification of unlabeled LoL match records with “Win/Loss”

**Introduction:** A detailed description of the objectives and requirements of the project, and a brief description of the methodology.

Objectives: The task is to create one or more classifiers that take as inputs from any fields from above (except “winner) such match record, and labels this record as a "1" or a "2". The test set comprises of ~2 Million of such records.

Requirement: The minimum requirement of the project may involve implementing at least one classification method we learn and successfully evaluate the test dataset using the test.py python program. The evaluation result should at least be higher than 50% (i.e. random guess)

Advanced requirement: use more than one classification methods and try to compare their advantages or disadvantages in terms of this data set. Examples: compare using the Accuracy or Training time.

Methodology: use Python or the libraries (Sklearn, Pytorch) to realise the implementation.

**Algorithms:** It contains the introduction of the classification methods. The parameters of the algorithm you use should also be introduction.

(1) kNN: The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice.

Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

scikit-learn implements two different nearest neighbors classifiers: KNeighborsClassifier implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. The optimal choice of the value k is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.

fit(X, y) -> Fit the model using X as training data and y as target values

(2) Naive Bayes: Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes’ theorem with the “naive” assumption of conditional independence between every pair of features given the value of the class variable. Bayes’ theorem states the following relationship, given class variable y and dependent feature vector x1 through xn

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters.

fit(X, y[, sample\_weight]) -> Fit Gaussian Naive Bayes according to X, y

(3) Random Forest: The sklearn.ensemble module includes two averaging algorithms based on randomized decision trees: the Random Forest algorithm and the Extra-Trees method. Both algorithms are perturb-and-combine techniques [B1998] specifically designed for trees. This means a diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers.

As other classifiers, forest classifiers have to be fitted with two arrays: a sparse or dense array X of size [n\_samples, n\_features] holding the training samples, and an array Y of size [n\_samples] holding the target values (class labels) for the training samples

(4) SVM：Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.The advantages of support vector machines are: 1. Effective in high dimensional spaces. 2. Still effective in cases where number of dimensions is greater than the number of samples. 3. Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. 4. Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels. The disadvantages of support vector machines include: 1. If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial. 2. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

fit(X, y[, sample\_weight]) -> Fit the model according to the given training data.

(5) ANN: Artificial neural networks (ANNs), usually simply called neural networks (NNs), are computing systems vaguely inspired by the biological neural networks that constitute animal brains.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

PyTorch is a deep learning framework gaining tremendous popularity among the researchers, mainly because of its very pythonic and understandable syntax.

a typical model training follows the following 5 steps: 1. Imports and Dataset 2. Train/Test Split 3. Defining a Neural Network Model 4. Model Training 5. Model Evaluation

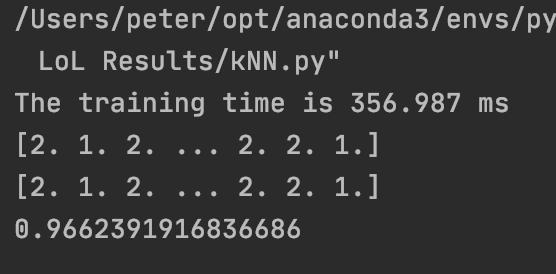
**Requirements:** The prerequisite packages you should use for your code

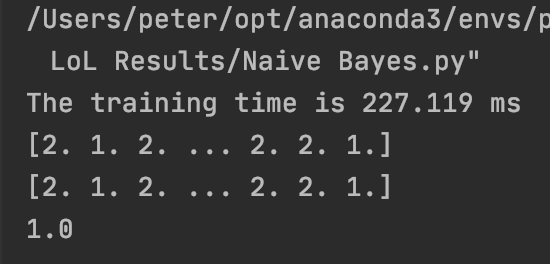
Basic: Torch, pandas, numpy

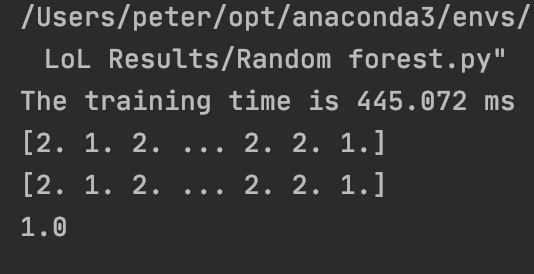
Sklearn: sklearn.metrics, sklearn.neighbors, sklearn.naive\_bayes, sklearn.ensemble, sklearn.svm

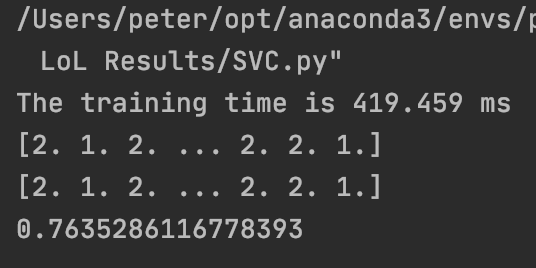
**Results:** A table showing your result and training time, as well as an inclusion of the results of executing your training program captured from the screen.

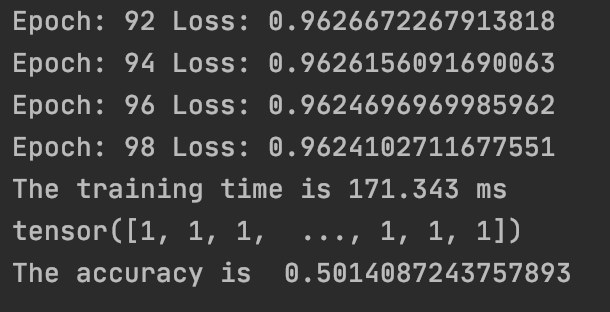
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | kNN | Naive Bayes | Random Forest | SVM | ANN |
| accuracy\_score | 0.966 | 1.0 | 1.0 | 0.76 | 0.50 |
| training time | 357ms | 227ms | 445ms | 419ms | 171ms |











**Comparison and discussion:** Summarize the experience gained in the project. Indicate how your program can be extended and improved if more time is allowed. The documentation for your project is a very important part. The ability of doing a good analysis will also be an important factor to the final assessment of your project, as well as your ability to do proper scientific work.

In this project, I used five models to fit the data, namely kNN, Naive Bayes, Random Forest, SVM, and ANN, and compared their differences, and it turns out that there is a big difference between the results from the different models, firstly in terms of accuracy (I have artificially removed gameId and' creationTime' the two invalid labels), Naive Bayes, with the highest Random Forest accuracy and an accuracy score of 1.0, and Naive Bayes showed a clear advantage in training speed, followed by kNN, although the model has a relatively simple features, but its accuracy score of 0.996 with invalid labels removed is still very impressive, SVM and ANN do not perform very well in relative terms, with considerably less accuracy than the previous three classifiers, and SVM also exhibits a relatively long training time, and on balance, Naive Bayes performs best. ANN performed the worst.

If I had more time, I would further expand the program by adding more available analytical models, and then analyze the similarities and differences between their accuracy score and run time, after which I would analyze the most important factors affecting the results and the proportion of the impact of each factor on the final results, i.e. the weights.