Machine Learning Lab 5

Implementing your kNN classifiers in WEKA

1) 1-Nearest Neighbour Classifier

a) Create a classifier called OneNN that extends AbstractClassifier. Implement `buildClassifier(Instances)` which is to simply store the train data (1NN is a **lazy classifier** https://en.wikipedia.org/wiki/Lazy_learning**).** A reference to the training data should be stored in a variable of the classifier so it can be accessed by other methods.

b) Implement the method `classifyInstance(Instance)` so that it can iterate over the training data to find the closest instance to the one being classified, and returns the class label of that closest instance.

   To measure the closeness of two instances, write a method

   `double distance(Instance, Instance)`
   that returns the Euclidian Distance between two instances. Remember not to include the class value in the calculation. Call this method in `classifyInstance`

c) Implement `distributionForInstance(Instance)` so that it sets the probability of the class predicted by `classifyInstance` to 1 and all other probabilities to 0.

d) Test your classifier on the toy data used in lab 1.

2) FootballPlayer problem

Sorry for the football theme, you will be glad to hear this problem is not about Arsenal. A few years ago an MSc student working for Norwich City Football Club addressed the following problem. Given the data about a player at the start of the season (Opta, etc), can we predict whether they will be a good purchase? The idea is that NCFC could use machine learning to spot good purchases and give themselves an advantage in the transfer market. He gathered data on all Championship players bought at the start of the season. At the end of the season, he asked a panel of experts (football scouts and others associated with the club) to assess whether the purchase was a success or not (1==successful purchase). This data is captured in the file FootballPlayers.arff. I want you to do an exploratory analysis of different classifiers on this data. I don't know the answer, I'm looking for you to explore to help familiarise you with WEKA.

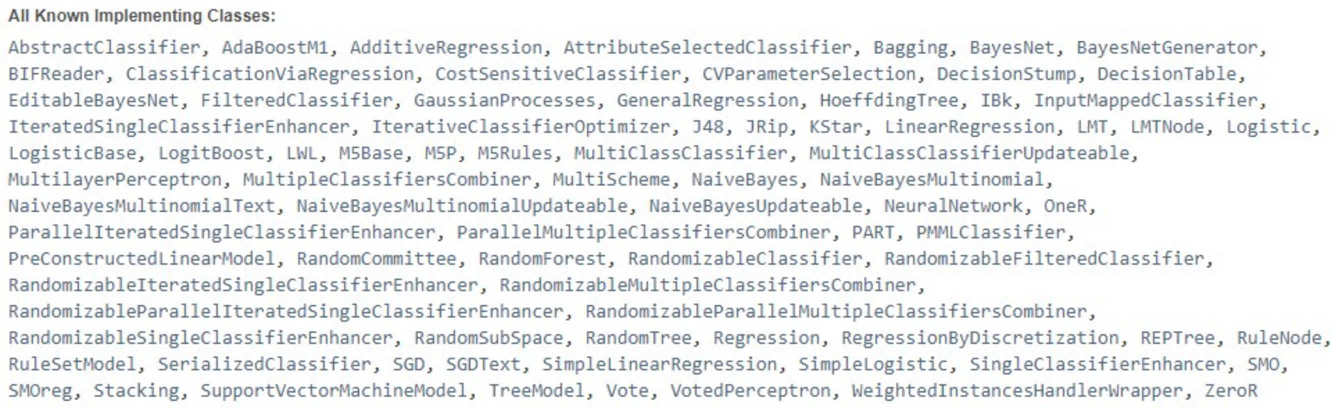
a) Load the data and print out the basic information: number of attributes, number instances, number of classes and class distribution.

b) Open the arff in an editor to get a feeling for the types of attributes used.

c) Split the data into 70% train and 30% test.

d) Build your 1-NN classifier on the data and assess the accuracy on the test data.

e) Look at the API for a list of classification algorithms available in WEKA. IB1 is the nearest thing to your 1-NN classifier. Assess the accuracy of the classifiers most related to the ones we teach: IB1 and IBk (nearest neighbour classifiers in lecture 5); LinearRegression (two class linear discriminant analysis, lecture 6); MultiLayerPerceptron (neural networks, lecture 7); SMO (support vector machines, lecture 8). Use them with the default settings for now.



All Known Implementing Classes:

AbstractClassifier, AdaBoostM1, AdditiveRegression, AttributeSelectedClassifier, Bagging, BayesNet, BayesNetGenerator, BIFReader, ClassificationViaRegression, CostSensitiveClassifier, CVParameterSelection, DecisionStump, DecisionTable, EditableBayesNet, FilteredClassifier, GaussianProcesses, GeneralRegression, HoeffdingTree, IBk, InputMappedClassifier, IteratedSingleClassifierEnhancer, IterativeClassifierOptimizer, J48, JRip, KStar, LinearRegression, LMT, LMTNode, Logistic, LogisticBase, LogitBoost, LWL, M5Base, M5P, M5Rules, MultiClassClassifier, MultiClassClassifierUpdateable, MultilayerPerceptron, MultipleClassifiersCombiner, MultiScheme, NaiveBayes, NaiveBayesMultinomial, NaiveBayesMultinomialText, NaiveBayesMultinomialUpdateable, NaiveBayesUpdateable, NeuralNetwork, OneR, ParallelIteratedSingleClassifierEnhancer, ParallelMultipleClassifiersCombiner, PART, PMMLClassifier, PreConstructedLinearModel, RandomCommittee, RandomForest, RandomizableClassifier, RandomizableFilteredClassifier, RandomizableIteratedSingleClassifierEnhancer, RandomizableMultipleClassifiersCombiner, RandomizableParallelIteratedSingleClassifierEnhancer, RandomizableParallelMultipleClassifiersCombiner, RandomizableSingleClassifierEnhancer, RandomSubSpace, RandomTree, Regression, RegressionByDiscretization, REPTree, RuleNode, RuleSetModel, SerializedClassifier, SGD, SGDText, SimpleLinearRegression, SimpleLogistic, SingleClassifierEnhancer, SMO, SMOreg, Stacking, SupportVectorMachineModel, TreeModel, Vote, VotedPerceptron, WeightedInstancesHandlerWrapper, ZeroR

f) Look again at the attributes for this data. How do you think 1-NN handles categorical attributes such as Position? Look in the API for a mechanism for testing whether an attribute is categorical or not. Remove all categorical attributes and repeat the classification experiment. Does it make a difference? Can we tell if the difference is down to just chance or caused by how we have changed the data?

g) Open the source for 1BK and see how it differs

3) K Nearest Neighbours

a) Implement a classifier that finds the k nearest neighbours, then chooses as a prediction the most frequent class in the k nearest neighbours.

b) Implement distributionForInstance so that it returns the class distribution of the k nearest neighbours.

c) Evaluate the test accuracy on the FootballPlayer problem for a range of k values, and plot them in excel. Can you make any inferences from this? Compare the best accuracy for a range of k values to the classifiers you used in 3). Is this a fair comparison?

d) Compare kNN for this problem to another classifier you have used on this module