Netaji Subhas Institute of Technology

Practical Training Report

on

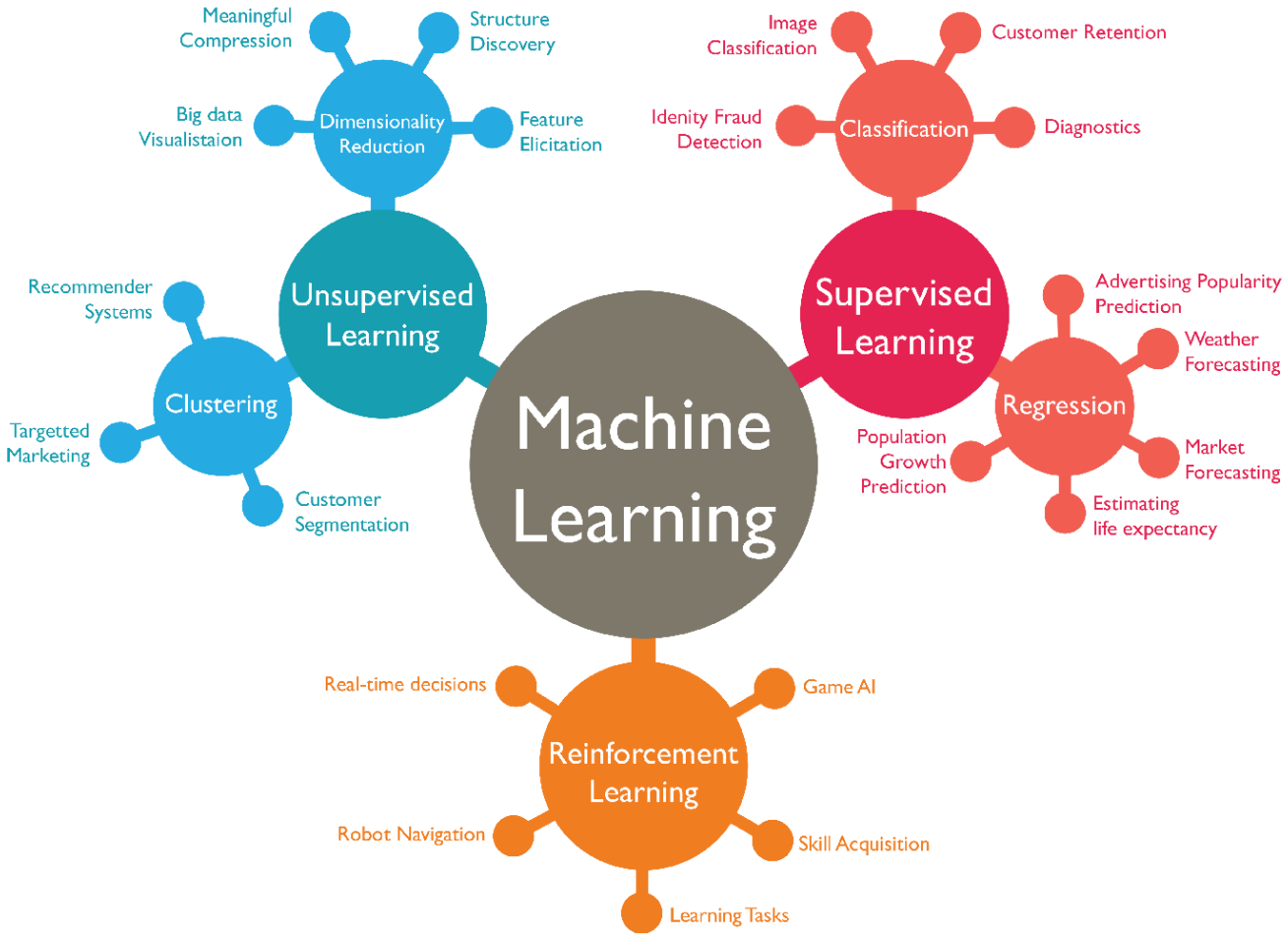
Machine Learning Algorithms : Analysis and Comparison

By

SHASHANK KANWAR
775/IT/14

May,2017

INTRODUCTION



Machine learning is a branch of science that deals with programming the systems in such a way that they automatically learn and improve with experience. Here, learning means recognizing and understanding the input data and making wise decisions based on the supplied data.

It is very difficult to cater to all the decisions based on all possible inputs. To tackle this problem, algorithms are developed. These algorithms build knowledge from specific data and past experience with the principles of statistics, probability theory, logic, combinatorial optimization, search, reinforcement learning, and control theory. Every algorithm is different in terms of computation and their performance may vary under different conditions. So it is necessary to choose appropriate algorithm(or classifier) for solving a problem.

OBJECTIVE

The objective of this project is to compare the following Machine Learning Algorithms (Classifiers) across different dimensions :

- Linear Regression
- Logistic Regression
- Naive Bayes
- SVM(Linear and RBF)
- KNN
- Decision Trees

Following are the aspects which were under consideration for comparison :

- Problem type
- Size of dataset
- Average predictive accuracy
- Training speed
- Predictive speed
- Amount of parameter tuning needed (excluding feature selection)
- Handles lots of irrelevant features well (separates signal from noise)?
- Automatically learns feature interactions?
- Features might need scaling?

CONTENTS

WHAT IS MACHINE LEARNING?

Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

Tom Mitchell provides a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

TYPES OF MACHINE LEARNING

• Supervised machine learning:

The program is “trained” on a predefined set of “training examples”, which then facilitate its ability to reach an accurate conclusion when given new data. Supervised learning algorithms make predictions based on a set of examples. For instance, historical stock prices can be used to hazard guesses at future prices. Each example used for training is labeled with the value of interest—in this case the stock price. A supervised learning algorithm looks for patterns in those value labels. It can use any information that might be relevant—the day of the week, the season, the company's financial data, the type of industry, the presence of disruptive geopolitical events—and each algorithm looks for different types of patterns. After the algorithm has found the best pattern it can, it uses that pattern to make predictions for unlabeled testing data—tomorrow's prices.

There are several specific types of supervised learning :

Classification

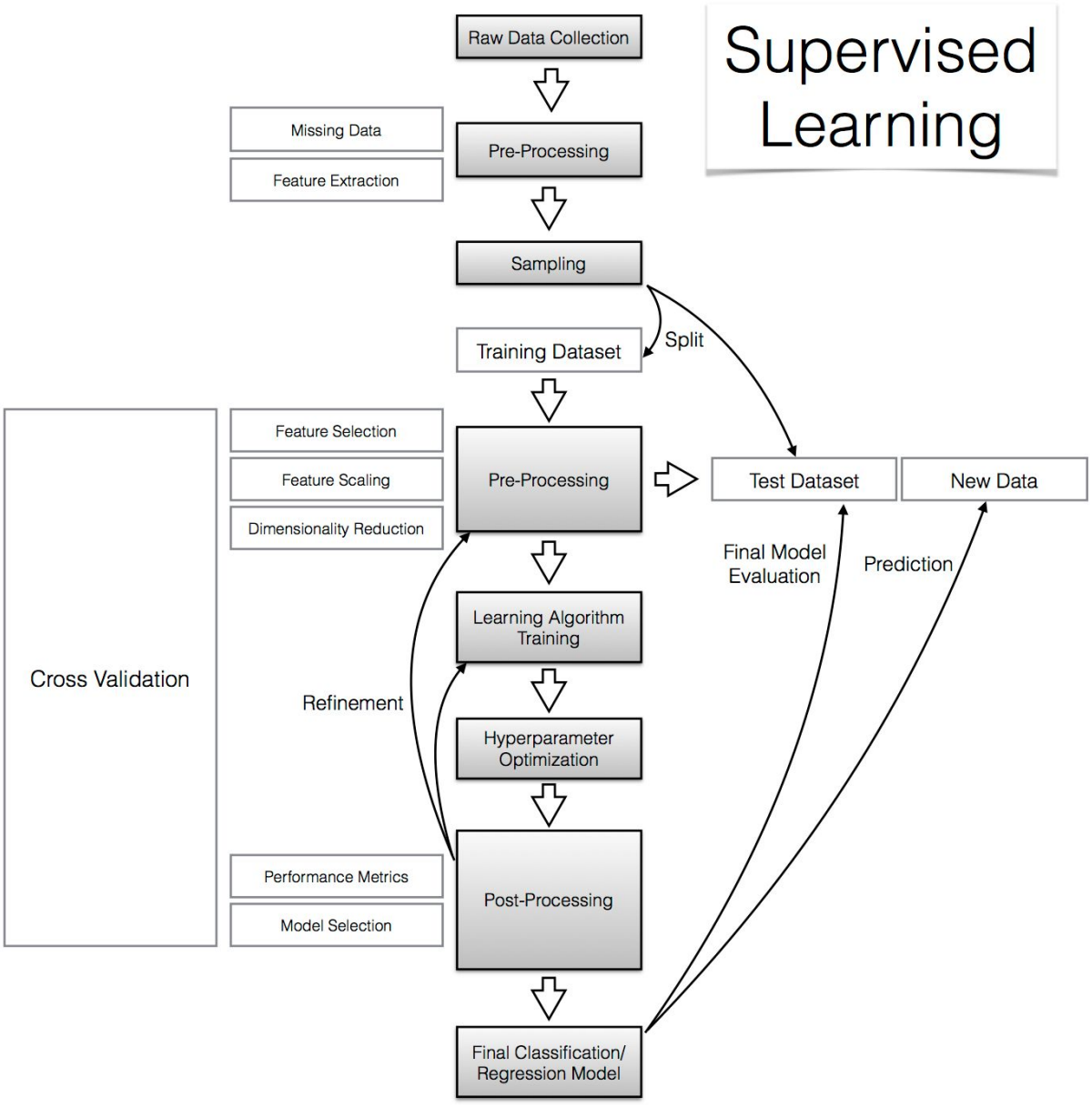
When the data are being used to predict a category, supervised learning is also called classification. This is the case when assigning an image as a picture of either a 'cat' or a 'dog'. When there are only two choices, it's called two-class or binomial classification. When there are more categories, as when predicting the winner of the NCAA March Madness tournament, this problem is known as multi-class classification.

Regression

When a value is being predicted, as with stock prices, supervised learning is called regression.Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questions of “How much?” or “How many?”.

Anomaly detection

Sometimes the goal is to identify data points that are simply unusual. In fraud detection, for example, any highly unusual credit card spending patterns are suspect. The possible variations are so numerous and the training examples so few, that it's not feasible to learn what fraudulent activity looks like. The approach that anomaly detection takes is to simply learn what normal activity looks like (using a history non-fraudulent transactions) and identify anything that is significantly different.



- Unsupervised machine learning:

The program is given a bunch of data and must find patterns and relationships therein.In unsupervised learning, data points have no labels associated with them. Instead, the goal of an unsupervised learning algorithm is to organize the data in some way or to describe its structure. This can mean grouping it into clusters or finding different ways of looking at complex data so that it appears simpler or more organized.

Clustering

Clustering is used to form groups or clusters of similar data based on common characteristics. Clustering is a form of unsupervised learning. The clustering engine goes through the input data completely and based on the characteristics of the data, it will decide under which cluster it should be grouped. Take a look at the following example.

- Reinforcement learning

In reinforcement learning, the algorithm gets to choose an action in response to each data point. The learning algorithm also receives a reward signal a short time later, indicating how good the decision was. Based on this, the algorithm modifies its strategy in order to achieve the highest reward. Currently there are no reinforcement learning algorithm modules in Azure Machine Learning. Reinforcement learning is common in robotics, where the set of sensor readings at one point in time is a data point, and the algorithm must choose the robot's next action. It is also a natural fit for Internet of Things applications.

M.L. CLASSIFIERS UNDER CONSIDERATION

- ## Linear regression

Linear regression fits a line (or plane, or hyperplane) to the data set. It's a workhorse, simple and fast, but it may be overly simplistic for some problems.

- ## Logistic regression

Although it confusingly includes 'regression' in the name, logistic regression is actually a powerful tool for two-class and multiclass classification. It's fast and simple. The fact that it uses an 'S'-shaped curve instead of a straight line makes it a natural fit for dividing data into groups. Logistic regression gives linear class boundaries, so when you use it, make sure a linear approximation is something you can live with.

- ## Decision Trees

There are many variants of decision trees, but they all do the same thing—subdivide the feature space into regions with mostly the same label. These can be regions of consistent category or of constant value, depending on whether you are doing classification or regression.

- ## Random forests

They are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

- ## Naive bayes

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

- ## SVM

Support vector machines (SVMs) find the boundary that separates classes by as wide a margin as possible. When the two classes can't be clearly separated, the algorithms find the best boundary they can. In these cases SVMs are able to separate classes more quickly and with less overfitting than most other algorithms, in addition to requiring only a modest amount of memory.

Aspects under consideration while choosing an algorithm

Accuracy

Getting the most accurate answer possible isn't always necessary. Sometimes an approximation is adequate, depending on what you want to use it for. If that's the case, you may be able to cut your processing time dramatically by sticking with more approximate methods. Another advantage of more approximate methods is that they naturally tend to avoid overfitting.

Training time

The number of minutes or hours necessary to train a model varies a great deal between algorithms. Training time is often closely tied to accuracy—one typically accompanies the other. In addition, some algorithms are more sensitive to the number of data points than others. When time is limited it can drive the choice of algorithm, especially when the data set is large.

Linearity

Lots of machine learning algorithms make use of linearity. Linear classification algorithms assume that classes can be separated by a straight line (or its higher-dimensional analog). These include logistic regression and support vector machines (as implemented in Azure Machine Learning). Linear regression algorithms assume that data trends follow a straight line. These assumptions aren't bad for some problems, but on others they bring accuracy down.

Number of parameters

Parameters are the knobs a data scientist gets to turn when setting up an algorithm. They are numbers that affect the algorithm's behavior, such as error tolerance or number of iterations, or options between variants of how the algorithm behaves. The training time and accuracy of the algorithm can

sometimes be quite sensitive to getting just the right settings. Typically, algorithms with large numbers parameters require the most trial and error to find a good combination.

Alternatively, there is a parameter sweeping module block in Azure Machine Learning that automatically tries all parameter combinations at whatever granularity you choose. While this is a great way to make sure you've spanned the parameter space, the time required to train a model increases exponentially with the number of parameters.

The upside is that having many parameters typically indicates that an algorithm has greater flexibility. It can often achieve very good accuracy. Provided you can find the right combination of parameter settings.

Number of features

For certain types of data, the number of features can be very large compared to the number of data points. This is often the case with genetics or textual data. The large number of features can bog down some learning algorithms, making training time unfeasibly long. Support Vector Machines are particularly well suited to this case

Advantages of some particular algorithms.

Advantages of Naive Bayes:

Super simple, you're just doing a bunch of counts. If the NB conditional independence assumption actually holds, a Naive Bayes classifier will converge quicker than discriminative models like logistic regression, so you need less training data. And even if the NB assumption doesn't hold, a NB classifier still often does a great job in practice. A good bet if want something fast and easy that performs pretty well. Its main disadvantage is that it can't learn interactions between features (e.g., it can't learn that although you love movies with Brad Pitt and Tom Cruise, you hate movies where they're together).

Advantages of Logistic Regression:

Lots of ways to regularize your model, and you don't have to worry as much about your features being correlated, like you do in Naive Bayes. You also have a nice probabilistic interpretation, unlike decision trees or SVMs, and you can easily update your model to take in new data (using an online gradient descent

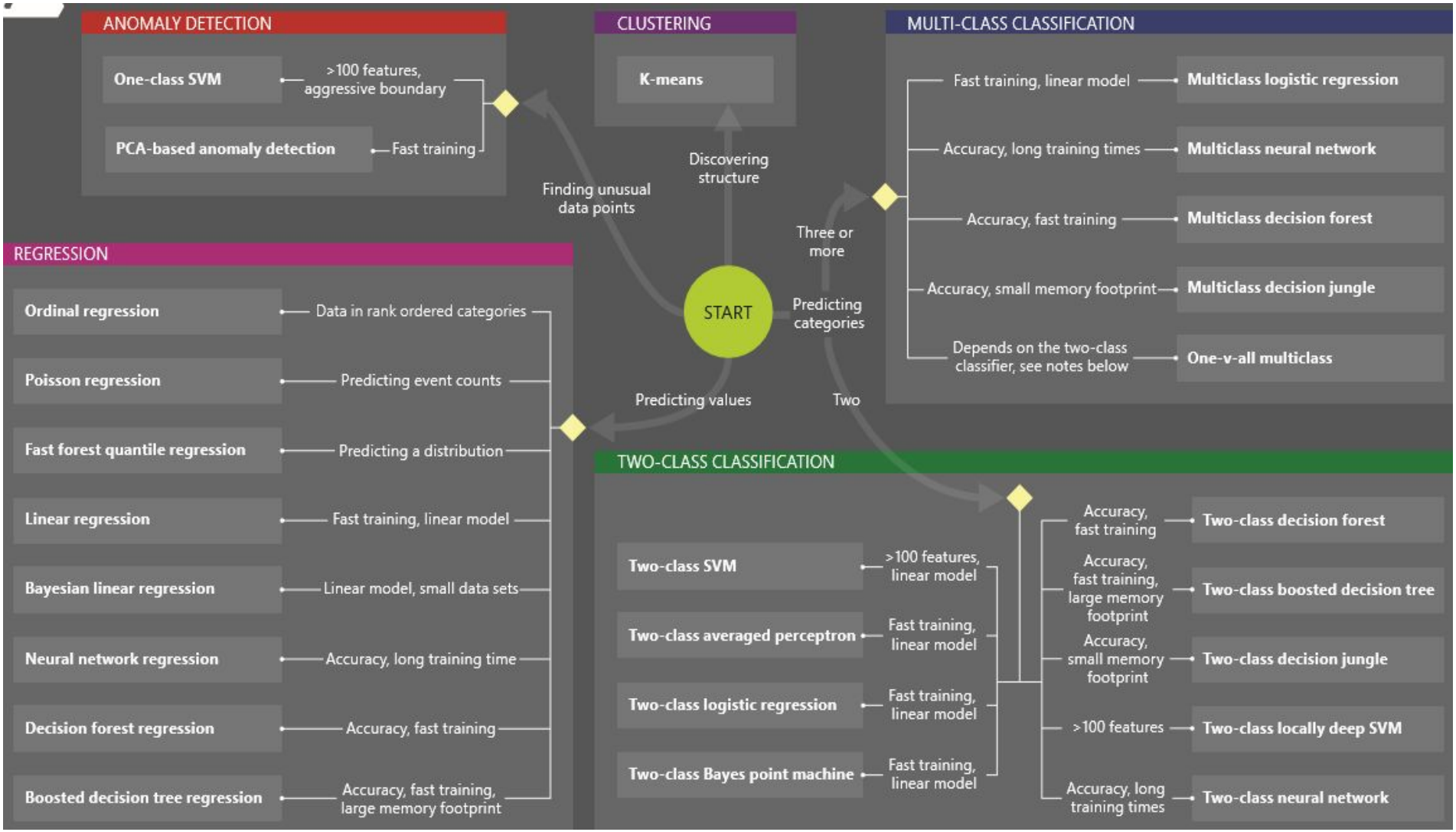
method), again unlike decision trees or SVMs. Use it if you want a probabilistic framework (e.g., to easily adjust classification thresholds, to say when you're unsure, or to get confidence intervals) or if you expect to receive more training data in the future that you want to be able to quickly incorporate into your model.

Advantages of Decision Trees:

Easy to interpret and explain (for some people – I'm not sure I fall into this camp). They easily handle feature interactions and they're non-parametric, so you don't have to worry about outliers or whether the data is linearly separable (e.g., decision trees easily take care of cases where you have class A at the low end of some feature x, class B in the mid-range of feature x, and A again at the high end). One disadvantage is that they don't support online learning, so you have to rebuild your tree when new examples come on. Another disadvantage is that they easily overfit, but that's where ensemble methods like random forests (or boosted trees) come in. Plus, random forests are often the winner for lots of problems in classification (usually slightly ahead of SVMs, I believe), they're fast and scalable, and you don't have to worry about tuning a bunch of parameters like you do with SVMs, so they seem to be quite popular these days.

Advantages of SVMs:

High accuracy, nice theoretical guarantees regarding overfitting, and with an appropriate kernel they can work well even if your data isn't linearly separable in the base feature space. Especially popular in text classification problems where very high-dimensional spaces are the norm. Memory-intensive, hard to interpret, and kind of annoying to run and tune, though, so I think random forests are starting to steal the crown.


START
Finding unusual data points
Discovering structure
Predicting categories
Predicting values
Three or more
Two
ANOMALY DETECTION
One-class SVM
>100 features, aggressive boundary
PCA-based anomaly detection
Fast training
CLUSTERING
K-means
MULTI-CLASS CLASSIFICATION
Fast training, linear model
Multiclass logistic regression
Accuracy, long training times
Multiclass neural network
Accuracy, fast training
Multiclass decision forest
Accuracy, small memory footprint
Multiclass decision jungle
Depends on the two-class classifier, see notes below
One-v-all multiclass
REGRESSION
Ordinal regression
Data in rank ordered categories
Poisson regression
Predicting event counts
Fast forest quantile regression
Predicting a distribution
Linear regression
Fast training, linear model
Bayesian linear regression
Linear model, small data sets
Neural network regression
Accuracy, long training time
Decision forest regression
Accuracy, fast training
Boosted decision tree regression
Accuracy, fast training, large memory footprint
TWO-CLASS CLASSIFICATION
Two-class SVM
>100 features, linear model
Two-class averaged perceptron
Fast training, linear model
Two-class logistic regression
Fast training, linear model
Two-class Bayes point machine
Fast training, linear model
Accuracy, fast training
Two-class decision forest
Accuracy, fast training, large memory footprint
Two-class boosted decision tree
Accuracy, small memory footprint
Two-class decision jungle
>100 features
Two-class locally deep SVM
Accuracy, long training times
Two-class neural network

RESOURCES REQUIRED

SOFTWARES

- Language -PYTHON 3.4.4
- Libraries - numpy,sci-kit learn,matplotlib,pandas
- IDE - Anaconda(Spyder)
- Microsoft Excel

DATASETS

- From sci-kit learn - Boston,Iris,Breast Cancer, Diabetes,
- From kagel/Quandl - Wiki-Google,Autos,Voice

ALGORITHM

Repeat this algorithm for all the algorithms appropriate for the problem dataset requirements.

1. Import dataset and record
    a. the problem type
    b. Size of dataset
2. Do pre processing on data
    a. Missing Data
    b. Feature Extraction
3. Repeat the following steps(6-13) N times :
4. Sample the data (Training - Testing)
5. Do pre-processing :
    a. Feature selection
    b. Feature scaling
    c. Dimension reduction
6. Train the model and observe
    a. Training speed
7. Test the model
8. Observe and record the following parameters :
    a. Prediction speed
    b. Accuracy
9. Calculate average prediction speed and average predictive accuracy

Repeat the whole process excluding pre-processing (step 2 and step 4).

Results

OUTPUT

```
In [18]: runfile('C:/Users/shash/untitled0.py', wdir='C:/Users/shash')
Linear Regression
Training time =  0.0
Testing time =  0.0
Linear Regression accuracy = 0.878867563676

Linear SVM
Training time =  0.0
  Linear SVM  accuracy = 0.883283229696
Testing time =  0.0


RBF SVM
Training time =  0.0
  RBF SVM  accuracy = 0.924756196415
Testing time =  0.0


Naive Bayes
Training time =  0.0
  Naive Bayes  accuracy = 0.966666666667
Testing time =  0.0


K Nearest Neighbors
Training time =  0.0
  K Nearest Neighbors  accuracy = 0.9
Testing time =  0.0


Decision Tree
Training time =  0.0
  Decision Tree  accuracy = 0.933333333333
Testing time =  0.0
```

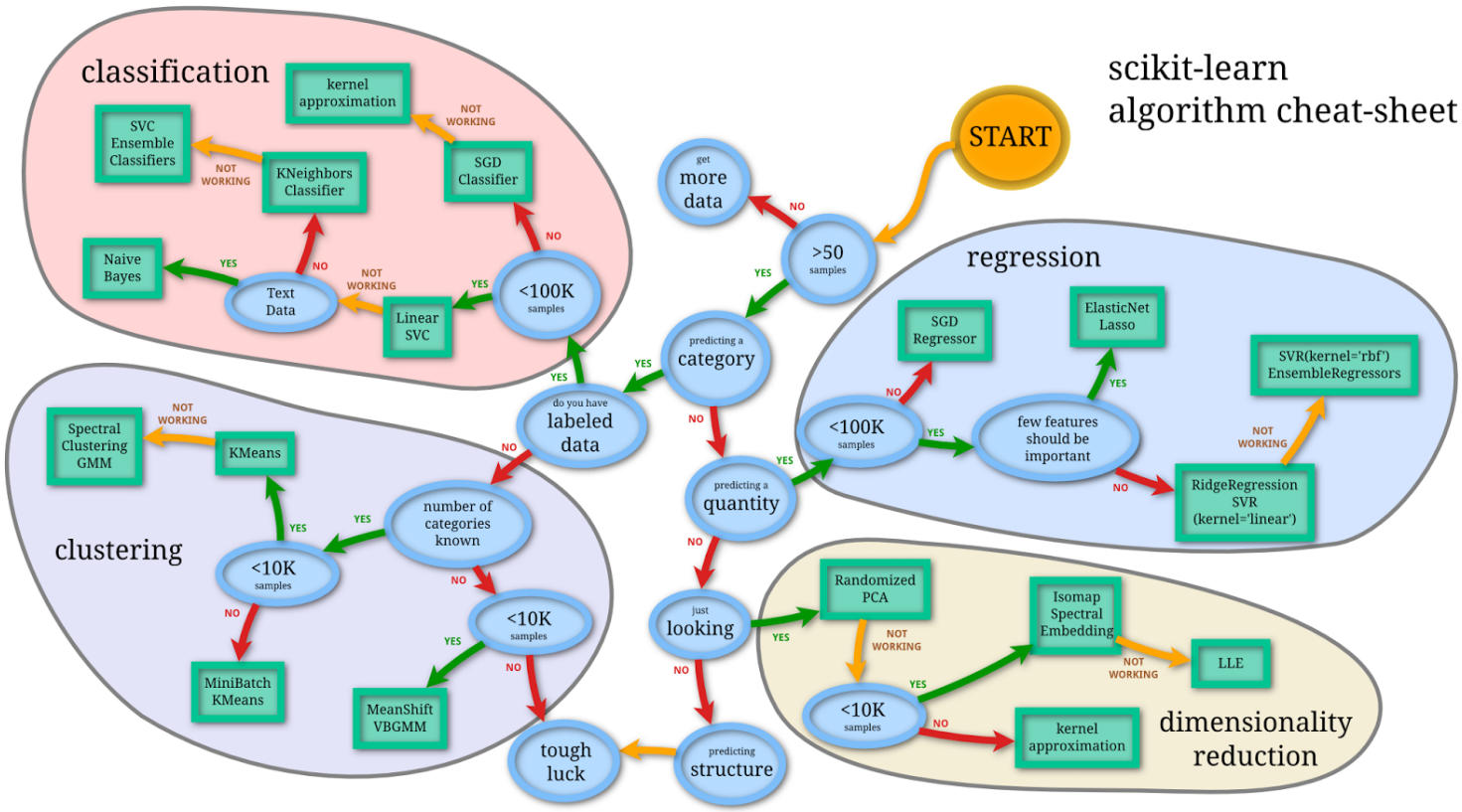
Comparison of algorithms(classifiers) across the following dimension :

Algorithm	**Problem Type**	**Results interpretable by you?**	**Average predictive accuracy**	**Training speed**	**Prediction speed**	**Amount of parameter tuning needed (excluding feature selection)**
Decision trees	Either	Somewhat	Lower	Fast	Fast	Some
KNN	Either	Yes	Lower	Fast	Depends on n	Minimal
Linear regression	Regression	Yes	Lower	Fast	Fast	None (excluding regularization)
Logistic regression	Classification	Somewhat	Lower	Fast	Fast	None (excluding regularization)
Naive Bayes	Classification	Somewhat	Lower	Fast (excluding feature extraction)	Fast	Some for feature extraction
Random Forests	Either	A little	Higher	Slow	Moderate	Some

Algorithm	**Performs well with small number of observations?**	**Handles lots of irrelevant features well (separates signal from noise)?**	**Automatically learns feature interactions?**	**Gives calibrated probabilities of class membership?**	**Parametric?**	**Features might need scaling?**
Decision trees	No	No	Yes	Possibly	No	No
KNN	No	No	No	Yes	No	Yes
Linear regression	Yes	No	No	N/A	Yes	No (unless regularized)
Logistic regression	Yes	No	No	Yes	Yes	No (unless regularized)
Naive Bayes	Yes	Yes	No	No	Yes	No
Random Forests	No	Yes (unless noise ratio is very high)	Yes	Possibly	No	No

CONCLUSION

The objective was to compare above mentioned machine learning algorithm(classifier) by training and testing all the required model. While training and testing the model accuracy , training speed, testing speed, size of datasets were the aspects under consideration. On the basis of these aspects along with the analysis of performance of these algorithms comparison is made . Although there is some value in the "brute force" approach (try everything and see what works best), there is a lot more value in being able to understand the trade-offs you're making when choosing one algorithm over another.



REFERENCES

MACHINE LEARNING -STUDY MATERIAL

https://www.coursera.org/learn/machine-learning/supplement/aAgxl/what-is-machine-learning

MACHINE LEARNING - ANALYSIS AND COMPARISON
https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice

http://www.dataschool.io/comparing-supervised-learning-algorithms/

http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/

http://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html#linear-regression