Choosing the Right Machine Learning Algorithm

Know your data

Look at Summary statistics and visualizations

- Percentiles can help identify the range for most of the data
- Averages and medians can describe central tendency
- Correlations can indicate strong relationships

Visualize the data

- Box plots can identify outliers
- Density plots and histograms show the spread of data
- Scatter plots can describe bivariate relationships

Clean your data

Deal with missing value. Missing data affects some models more than others. Even for models that handle missing data, they can be sensitive to it (missing data for certain variables can result in poor predictions)

Choose what to do with outliers

- Outliers can be very common in multidimensional data.
- Some models are less sensitive to outliers than others. Usually tree models are less sensitive to the presence of outliers. However, regression models, or any model that tries to use equations, could definitely be affected by outliers.
- Outliers can be the result of bad data collection, or they can be legitimate extreme values.

Does the data need to be aggregated?

Augment your data

Feature engineering is the process of going from raw data to data that is ready for modeling. It can serve multiple purposes:

- Make the models easier to interpret (e.g. binning)
- Reduce data redundancy and dimensionality (e.g. PCA)
- Rescale variables (e.g. standardizing or normalizing)

Different models may have different feature engineering requirements. Some have built in feature engineering.

Categorize the problem

The next step is to categorize the problem. This is a two-step process.

Categorize by input:

- If you have labelled data, it's a supervised learning problem.
-
- If you have unlabelled data and want to find structure, it's an unsupervised learning problem.
-
- If you want to optimize an objective function by interacting with an environment, it's a reinforcement learning problem.

- **Categorize by output:**
- If the output of your model is a number, it's a regression problem.
- If the output of your model is a class, it's a classification problem.
- If the output of your model is a set of input groups, it's a clustering problem.
-
- Do you want to detect an anomaly? That's anomaly detection

Understand your constraints

- What is your data storage capacity? Depending on the storage capacity of your system, you might not be able to store gigabytes of classification/regression models or gigabytes of data to be clustered. This is the case, for instance, for embedded systems.
-
- Does the prediction have to be fast? In real time applications, it is obviously very important to have a prediction as fast as possible. For instance, in autonomous driving, it's important that the

classification of road signs be as fast as possible to avoid accidents.

-
- Does the learning have to be fast? In some circumstances, training models quickly is necessary: sometimes, you need to rapidly update, on the fly, your model with a different dataset.

Find the available algorithms

Now that you a clear understanding of where you stand, you can identify the algorithms that are applicable and practical to implement using the tools at your disposal. Some of the factors affecting the choice of a model are:

- Whether the model meets the expected goals
- How much preprocessing the model needs;
- How accurate the model is;
- How explainable the model is;
- How fast the model is: How long does it take to build a model, and how long does the model take to make predictions;
- How scalable the model is.

An important criterion affecting choice of algorithm is model complexity.

Generally speaking, a model is more complex is:

- It relies on more features to learn and predict (e.g. using two features vs ten features to predict a target)

- It relies on more complex feature engineering (e.g. using polynomial terms, interactions, or principal components)

- It has more computational overhead (e.g. a single decision tree vs. a random forest of 100 trees).

Besides this, the same machine learning algorithm can be made more complex based on the number of parameters or the choice of some hyperparameters. For example,

- A regression model can have more features, or polynomial terms and interaction terms.

- A decision tree can have more or less depth.

Making the same algorithm more complex increases the chance of overfitting.

Linear Regression

These are probably the simplest algorithms in machine learning. Regression algorithms can be used for example, when you want to compute some continuous value as compared to Classification where the output is categoric.
Henceforth, whenever you are told to predict some future value of a process which is currently running, you can go with regression algorithm.
Linear Regressions are however unstable in case features are redundant, i.e. if there is multicollinearity

Logistic Regression

Logistic regression performs binary classification, so the label outputs are binary. It takes linear combination of features and applies non-linear function (sigmoid) to it, so it's a very small instance of neural network.

Logistic regression provides lots of ways to regularize your model, and you don't have to worry as much about your features being correlated. You also have a nice probabilistic interpretation, and you can easily update your model to take in new data, unlike decision trees or SVMs. Use it if you want a probabilistic framework or if you expect to receive more training data in the future that you want to be able to quickly incorporate into your model. Logistic regression can also help you understand the contributing factors behind the prediction, and is not just a black box method.

Logistic regression can be used in cases such as:

- Predicting the Customer Churn
- Credit Scoring & Fraud Detection
- Measuring the effectiveness of marketing campaigns

Decision trees

Single trees are used very rarely, but in composition with many others they build very efficient algorithms such as Random Forest or Gradient Tree Boosting.

Decision trees 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.

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.

Decision Trees can also take a lot of memory (the more features you have, the deeper and larger your decision tree is likely to be)

Trees are excellent tools for helping you to choose between several courses of action.

- Investment decisions
- Customer churn
- Banks loan defaulters
- Build vs Buy decisions
- Sales lead qualifications

K-means

Sometimes you don’t know any labels and your goal is to assign labels according to the features of objects. This is called clusterization task. Clustering algorithms can be used for example, when there is a large group of users and you want to divide them into particular groups based on some common attributes.

If there are questions like how is this organized or grouping something or concentrating on particular groups etc. in your problem statement then you should go with Clustering.

The biggest disadvantage is that K-Means needs to know in advance how many clusters there will be in your data, so this may require a lot of trials to "guess" the best K number of clusters to define.

Principal component analysis (PCA)

Principal component analysis provides dimensionality reduction. Sometimes you have a wide range of features, probably highly correlated between each other, and models can easily overfit on a huge amount of data. Then, you can apply PCA.

One of the keys behind the success of PCA is that in addition to the low-dimensional sample representation, it provides a corresponding low-dimensional representation of the variables. The sample and variable representations provide a way to visually find variables that are characteristic of a group of samples.

Support Vector Machines

Support Vector Machine (SVM) is a supervised machine learning technique that is widely used in pattern recognition and classification problems—when your data has exactly two classes.

High accuracy, nice theoretical guarantees regarding overfitting, and with an appropriate kernel they can work well even if you're data isn't linearly separable in the base feature space. Especially popular in text classification problems where very high-dimensional spaces are the norm. SVMs are however memory-intensive, hard to interpret, and difficult to tune.

SVM can be used in real-world applications such as:

- detecting persons with common diseases such as diabetes
- hand-written character recognition
- text categorization—news articles by topics
- stock market price prediction

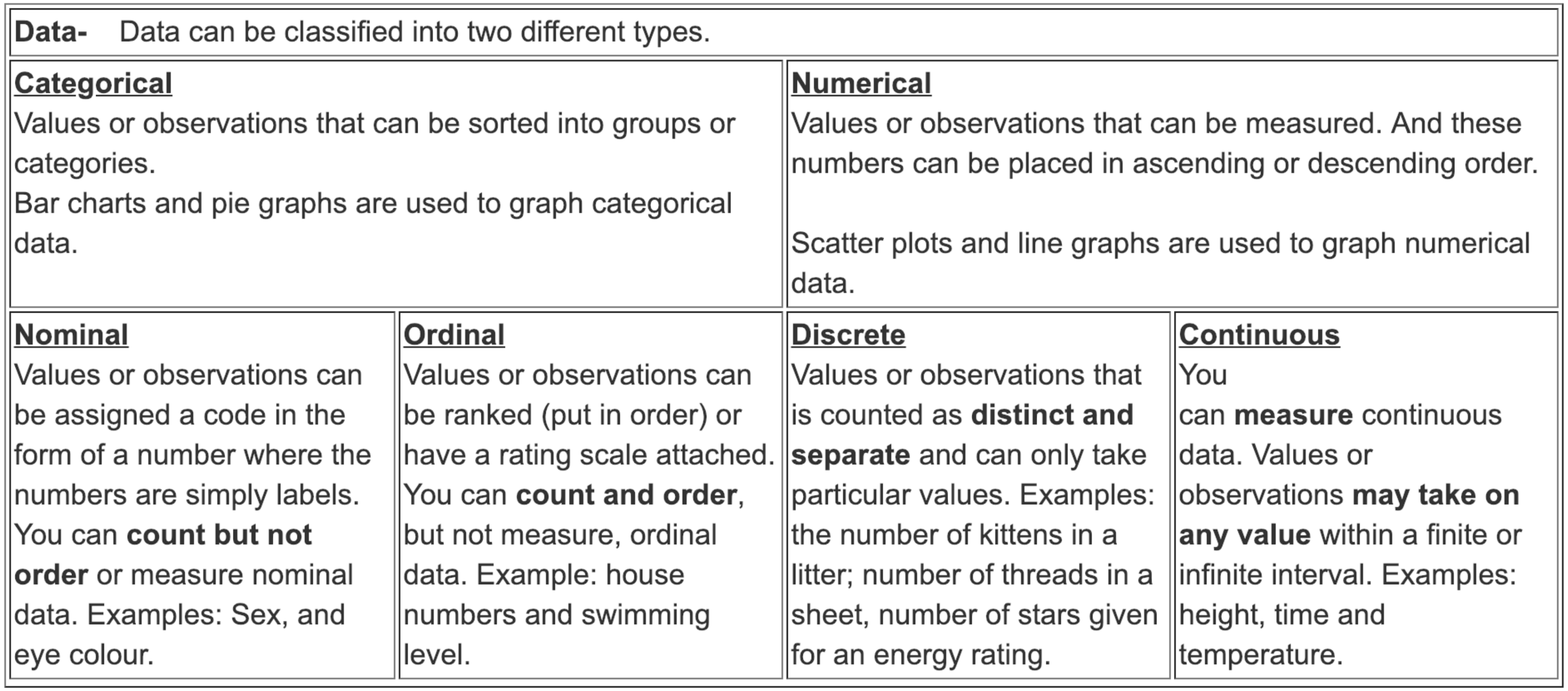
Random Forest

Random Forest is an ensemble of decision trees. It can solve both regression and classification problems with large data sets. It also helps identify most significant variables from thousands of input variables. Random Forest is highly scalable to any number of dimensions and has generally quite acceptable performances. Then finally, there are genetic algorithms, which scale admirably well to any dimension and any data with minimal knowledge of the data itself, with the most minimal and simplest implementation being the microbial genetic algorithm. With Random Forest however, learning may be slow (depending on the parameterization) and it is not possible to iteratively improve the generated models

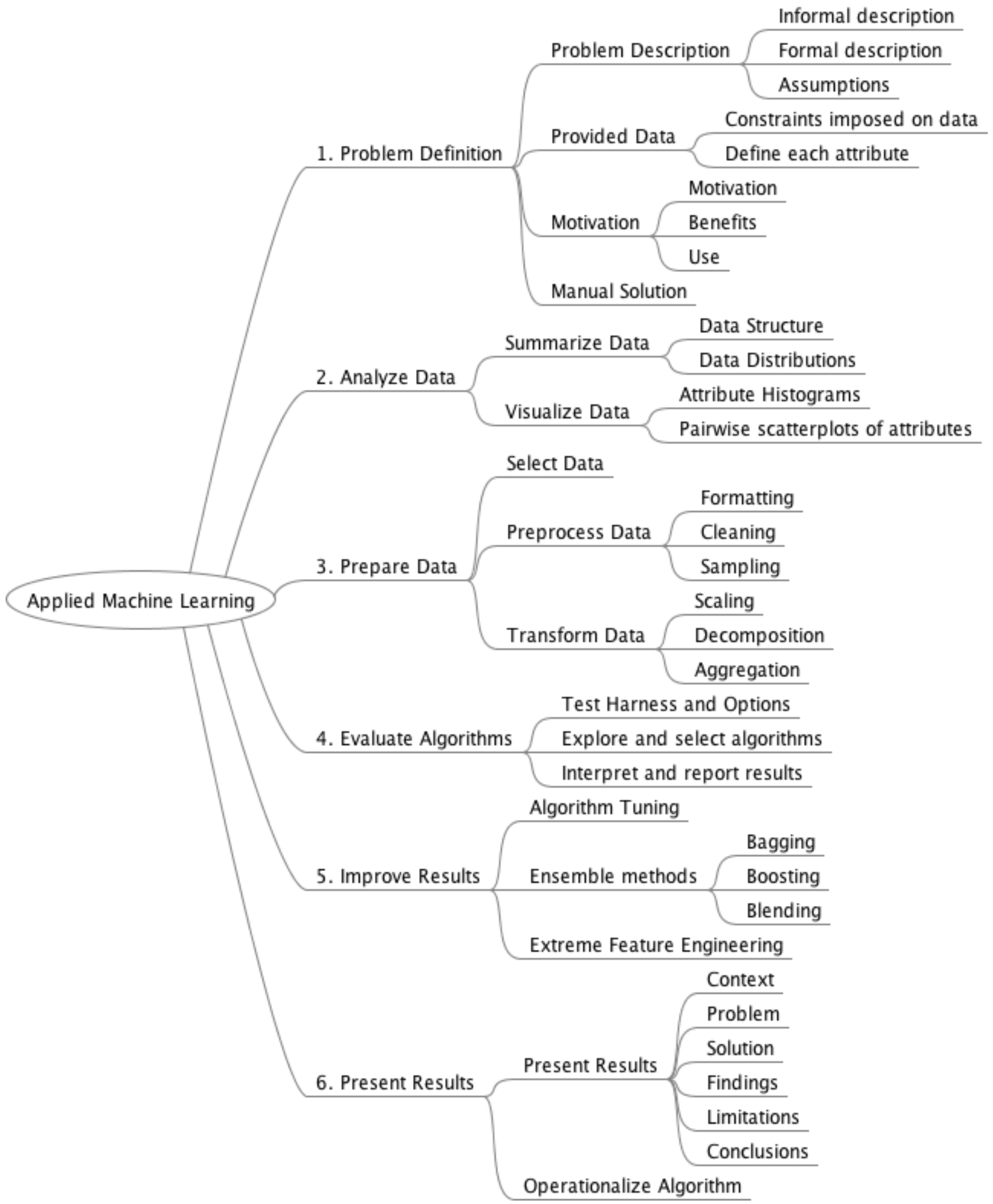
Random Forest can be used in real-world applications such as:

- Predict patients for high risks

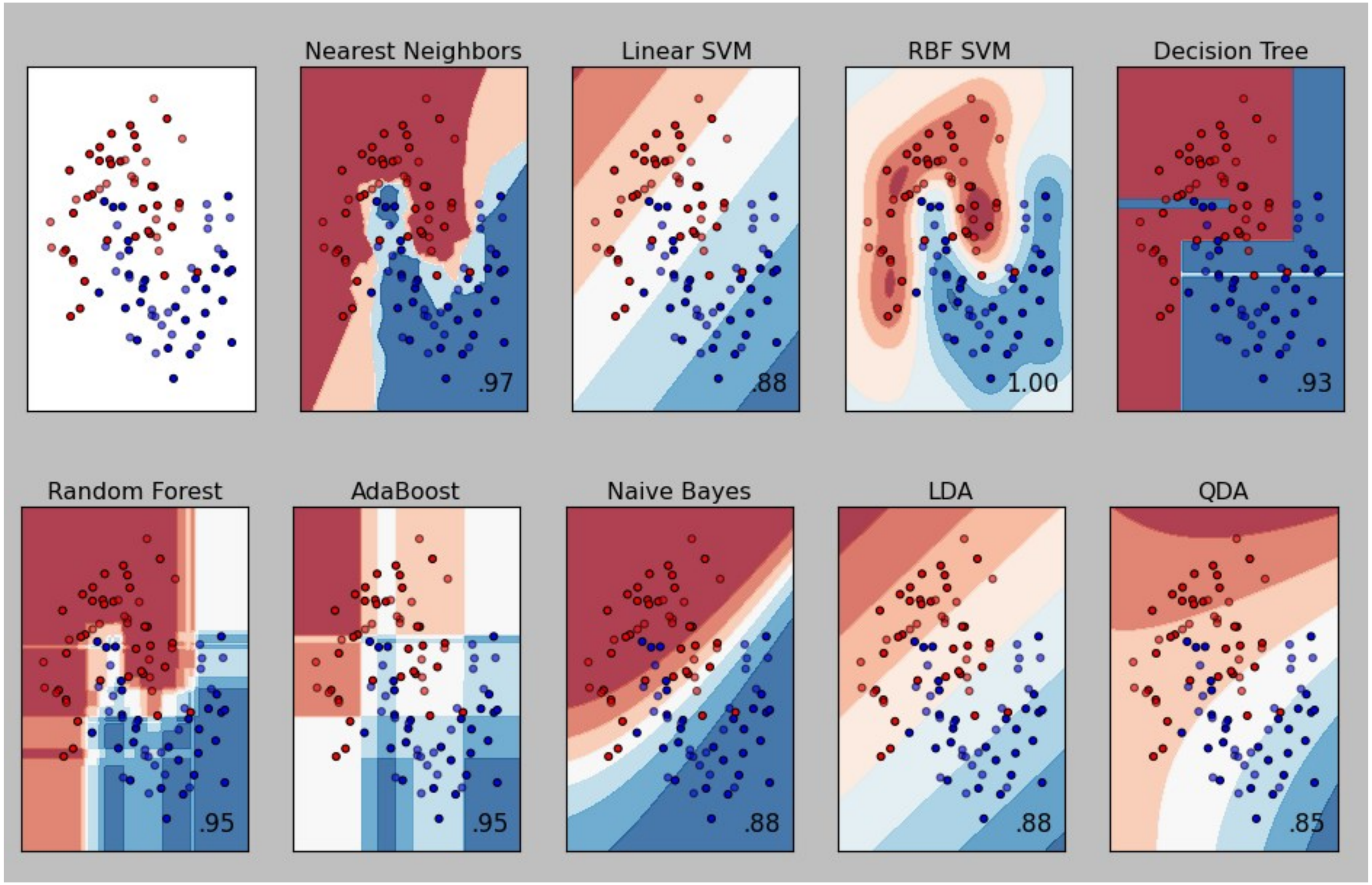
- Predict parts failures in manufacturing
- Predict loan defaulters



Data- Data can be classified into two different types.			
Categorical Values or observations that can be sorted into groups or categories. Bar charts and pie graphs are used to graph categorical data.		**Numerical** Values or observations that can be measured. And these numbers can be placed in ascending or descending order. Scatter plots and line graphs are used to graph numerical data.	
Nominal Values or observations can be assigned a code in the form of a number where the numbers are simply labels. You can **count but not order** or measure nominal data. Examples: Sex, and eye colour.	**Ordinal** Values or observations can be ranked (put in order) or have a rating scale attached. You can **count and order**, but not measure, ordinal data. Example: house numbers and swimming level.	**Discrete** Values or observations that is counted as **distinct and separate** and can only take particular values. Examples: the number of kittens in a litter; number of threads in a sheet, number of stars given for an energy rating.	**Continuous** You can **measure** continuous data. Values or observations **may take on any value** within a finite or infinite interval. Examples: height, time and temperature.



Applied Machine Learning
1. Problem Definition
Problem Description
Informal description
Formal description
Assumptions
Provided Data
Constraints imposed on data
Define each attribute
Motivation
Motivation
Benefits
Use
Manual Solution
2. Analyze Data
Summarize Data
Data Structure
Data Distributions
Visualize Data
Attribute Histograms
Pairwise scatterplots of attributes
3. Prepare Data
Select Data
Preprocess Data
Formatting
Cleaning
Sampling
Transform Data
Scaling
Decomposition
Aggregation
4. Evaluate Algorithms
Test Harness and Options
Explore and select algorithms
Interpret and report results
5. Improve Results
Algorithm Tuning
Ensemble methods
Bagging
Boosting
Blending
Extreme Feature Engineering
6. Present Results
Present Results
Context
Problem
Solution
Findings
Limitations
Conclusions
Operationalize Algorithm


Nearest Neighbors
.97
Linear SVM
.88
RBF SVM
1.00
Decision Tree
.93
Random Forest
.95
AdaBoost
.95
Naive Bayes
.88
LDA
.88
QDA
.85