Kaggle Competition: Product Classification

Machine Learning CS933

Term Project

Name:

Muping He
Jianan Duan
Sinian Zheng

Acknowledgements:

These are the complete, official rules for the Competition (the 'Competition Rules') and incorporate by reference the contents of the Competition Website listed above.

By downloading a dataset linked from the Competition Website, submitting an entry to this Competition, or joining a Team in this Competition, you are agreeing to be bound by these Competition Rules which constitute a binding agreement between you and the Competition Sponsor. The Competition is sponsored by the Competition Sponsor listed above and hosted on the Sponsor's behalf by Kaggle Inc ('Kaggle').

The competition is used for CS933 Machine Learning class term project, our team members has Muping He, Jianan Duan and Sinian Zheng. In the end, we will upload our solution to Kaggle.com; thanks for everyone’s efforts and Dr. Ming-Hwa Wang’s lectures on Machine Learning.

Abstract:

This project studies classification methods and try to find the best model for the Kaggle competition of Otto group product classification. Machine learning models deployed in this paper include decision trees, neural network, gradient boosting model, etc. We will also use cross-validation for prediction accuracy in order to compare between models.

Table of contents:

Introduction
- Objective
- What is the problem
- why this is a project related the this class
- Why other approach is no good
- Why you think your approach is better
- Statement of the problem
- Area or scope of investigation

Theoretical bases and literature review
- Definition of the problem:
- Theoretical background of the problem:
  - Decision Trees
  - Bayesian Approaches
  - Neural Networks
  - Regression-based Methods
  - Vector-based Methods - SVM
- Related research to solve the problem
- Advantage/disadvantage of those research
- Our solution to solve this problem
- Where your solution different from others
- why your solution is better

Hypothesis

Methodology
- 4.1 how to generate/collect input data
- 4.2 how to solve the problem
- 4.3 algorithm design
- 4.4 language used
- 4.5 tools used
- 4.6 how to generate output
- 4.7 how to test against hypothesis
- 4.8 how to proof correctness

5. Implementation
- 5.1 Code (refer programming requirements)
- 5.2 Design document and flowchart

6. Data analysis and discussion
- 6.1 output generation & output analysis
- 6.3 Compare output against hypothesis
- 6.4 Abnormal case explanation
- 6.5 Discussion

7. conclusions and recommendations
- 7.1 summary and conclusions

7.2 recommendations for future studies
bibliography
9. Appendices
9.1 program flowchart
9.2 program source code with documentation
9.3 input/output listing

1. Introduction

1.1. Objective

The objective of this project is to apply classification learning models on an e-commerce products dataset with 93 features for more than 200,000 products and therefore to obtain a predictive model with high accuracy for identifying future products categories.

1.2. What is the problem

Given a dataset concerning e-commerce products with 93 features for more than 200,000 products, this project is aimed to build a predictive model that is able to distinguish products between 9 main product categories. A few selected classification learning models will be trained by the dataset that includes each product's corresponding category. In order to compare among applied models, cross validation will be used to evaluate the accuracy and then the model with comparatively higher accuracy will be selected.

1.3. why this is a project related the this class

This project does not only contribute to achieve the objectives of this Machine Learning class, but also applies what we learned in class into practice. This project follows the course objective that is to learn advanced knowledge and implementation in machine learning. As an important part in machine learning, classification has many real world applications, such as business marketing segmentation, Internet search result grouping, etc. In this project, we use classification to distinguish e-commerce products between main categories, which directly help us to obtain hands-on skills of dealing with real data from the perspective of machine learning.

1.4. Why other approach is no good

Among all classification techniques, quite a few of them has restrictions and preference in attribute value types and the size of attributes. Besides, simply applying a learning model on a particular dataset normally will not yield the “best” analysis results. According to different dataset, adding more techniques can help train the model in a better performance.

1.5. Why you think your approach is better

In our project, instead of applying single models, we apply as much as possible classification models on the dataset. After a comparison of accuracies among all applied models, we choose the one with comparatively highest accuracy. This approach not only allow us to test out and get a comprehensive concept of most of those frequently used models, but also will obtain the one with highest accuracy under our administration. Besides, we add and adjust weighting factors to some of those models. Instead of each data contributing equally to the model, this approach enables us to add weights on those more important core data so that the model can be trained better for this particular dataset.

1.6. Statement of the problem

This project is aimed to build a predictive model that is able to distinguish between main product categories in an e-commerce dataset. The main dataset regarding to ecommerce products has 93 features for more than 200,000 products. The resource of the dataset comes from an open competition Otto Group Product Classification Challenge, which can be retrieved on www kaggle.com. The Otto Group is one of the world's largest e-commerce companies. They are selling millions of products worldwide everyday, with several thousand products being added to their product line. Therefore it is important for the company to do a consistent analysis on the performance of their products. However, due to the diversity of the company's global infrastructure, it is challenging to classify each product appropriately. As a result, the quality of product analysis depends heavily on the ability to accurately cluster similar products. In order to find out the model with comparatively high accuracy, we will first utilize various classification learning models including decision trees, Bayesian approaches, neural network, regression-based methods, vector-based methods, etc. After an accuracy comparison among applied models, we shall obtain the "best" model under our experimental analysis.

1.7. Area or scope of investigation

This project will mainly discuss classification models in the field of machine learning and statistics. Specifically, a variety of classification models will be used and evaluated including decision trees, Bayesian approaches, neural network, regression-based methods, vector-based methods, etc. Besides, model validation techniques, which are used to assess how the results of a statistical analysis will generalize to an independent data set, will be applied on trained models. In the process of applying models, programming languages including R, Python, and Java will be used. To conclude, this project covers knowledge in machine learning, computer science, and statistics, with a concentration on various classification learning models.

2. Theoretical bases and literature review

2.1. Definition of the problem:

In this problem, we are given a data set contains over 20k products and 93 features with them; the goal is find a predictive model to distinguish between their main product categories. In machine learning and statistics, **classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.).

2.2. Theoretical background of the problem:

2.2.1. Decision Trees

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. Tree models where the target variable can take a finite set of values are called classification trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. In data mining, a decision tree describes data but not decisions; rather the resulting classification tree can be an input for decision making. An example of it looks like:

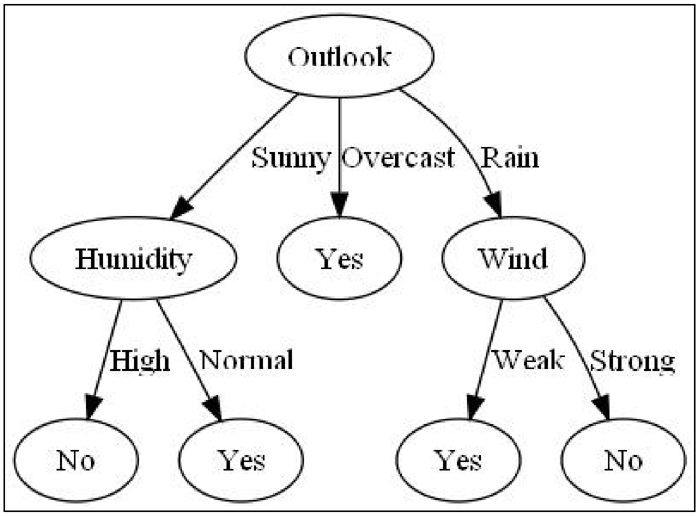


Figure 1. Decision Tree Example

Tree based methods have Boosted trees, Random forest, etc.

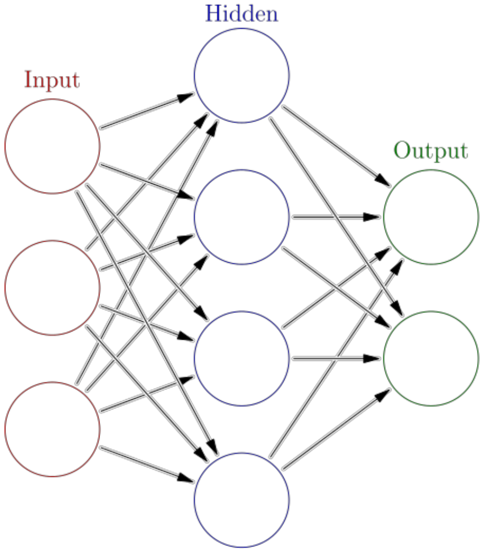
2.2.2. Bayesian Approaches

There are two groups of Bayesian approaches: Naive and non-naive Bayesian approaches. The naive part of the former is the assumption of feature independence, meaning that the feature order is irrelevant and consequently that the presence of one word does not affect the presence or absence of another. This assumption makes the computation of Bayesian approaches more efficient.

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \dots, x_n)$ representing some n features (dependent variables), it assigns to this instance probabilities $p(C_k \vert x_1, \dots, x_n)$for each of k possible outcomes or *classes*. Using Bayes' theorem, the conditional probability can be decomposed as $p(C_k \vert \mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x} \vert C_k)}{p(\mathbf{x})}$. Using Bayesian probability terminology, the above equation can be written as $\mbox{posterior} = \frac{\mbox{prior} \times \mbox{likelihood}}{\mbox{evidence}}$.

2.2.3. Neural Networks

Different neural network approaches have been applied to document categorization problems. While some of them use the simplest form of neural networks, known as perceptrons, which consist only of an input and an output layer, others build more sophisticated neural networks with a hidden layer between the two others. In general, these feed-forward-nets consist of at least three layers (one input, one output, and at least one hidden layer) and use backpropagation as learning mechanism. However, the comparatively old perceptron approaches perform surprisingly well.



In machine learning and cognitive science, artificial neural networks (ANNs) are a family of statistical learning algorithms inspired by biological neural networks, neural networks are similar to biological neural networks in performing functions collectively and in parallel by the units, rather than there being a clear delineation of subtasks to which various units are assigned.

The first layer has input neurons which send data via synapses to the second layer of neurons, and then via more synapses to the third layer of output neurons. More complex systems will have more layers of neurons with some having increased layers of input neurons and output neurons. The synapses store parameters called "weights" that manipulate the data in the calculations.

The advantage of neural networks is that they can handle noisy or contradictory data very well. Furthermore, some types of neural networks are able to comprehend fuzzy logic, but one has to change from backpropagation as learning mechanism to counterpropagation. The advantage of the high flexibility of neural networks entails the disadvantage of very high computing costs. Another disadvantage is that neural networks are extremely difficult to interpret the result.

2.2.4. Regression-based Methods

For this method the training data are represented as a pair of input/output matrices where the input matrix is identical to our feature matrix A and the output matrix B. The most methodology used here is Logistic regression, it measures the relationship between categorical dependent variables and one or more independent variables, which are usually continuous, by using probability scores as the predicted values of the dependent variable. It can be seen as a special case of generalized linear model.

Logistic regression can be binomial or multinomial. Binomial or binary logistic regression deals with situations in which the observed outcome for a dependent variable can have only two possible types. In binary logistic regression, the outcome is usually coded as "0" or "1", as this leads to the most straightforward interpretation.

2.2.5. Vector-based Methods - SVM

Support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. A support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

2.3. Related research to solve the problem

- The research <Chimera: Large-Scale Classification using Machine Learning, Rules, and Crowdsourcing> introduces how Chimera employs a combination of learning, rules (created by in-house analysts), and crowdsourcing to achieve accurate, continuously improving, and cost-effective classification.

- The research <A New Methodology for Photometric Validation in Vehicles Visual Interactive Systems> introduces a new way to identify a region of components on Instrument Clusters (IC) as homogenous or not, and showed that the proposed methodology obtains precision above 95%.

- Du et al (2011) introduced rule learning for classification based on neighborhood covering reduction. Specifically, it redefines the relative covering element reduction and analyze the difference between covering element reduction and relative covering element reduction.

- Kim et al (2010) proposed a mathematical programming method that directly mines discriminative patterns as numerical features for classification and they also proposed a technique called novel search space shrinking which addresses the inefficiencies in iterative pattern mining algorithms.

2.4. Advantage/disadvantage of those research

- The research <Chimera: Large-Scale Classification using Machine Learning, Rules, and Crowdsourcing> Advantage：The reasearch shows that Chimera is novel in four important aspects:
    1. Chimera uses both learning and hand-crafted rules (written by domain analysts) extensively.
    2. It uses a combination of crowdsourcing and in-house analysts to evaluate and analyze the system, to achieve an accurate, continuously improving, and cost-effective solution for classification.
    3. Chimera is scalable in terms of human resources, by using in-house analysts and tapping into crowdsourcing, the most “elastic" and “scalable" workforce available for general use today, and

4. Chimera uses a human-machine hybrid algorithm that treats learning, rules, crowd workers, in-house analysts, and developers as “first-class citizen".

Disadvantage: With the time going on, the write rules becomes more and more storing into the database. Then it will be slower for searching activity.The current learning-based classifiers are not precise enough.

- The research <A New Methodology for Photometric Validation in Vehicles Visual Interactive Systems> Advantage:This research gets a high precision of classification. Descriptors was given to two machine learning algorithms, namely the SVM and an ANN. Both the SVM and the ANN obtained precisions higher than 95% when classifying the regions.
  Disadvantage: This research focuses on image and visual quality identification. The training data sets are small and hard to obtain with only 7 features which is not very useful in our large-scale data problem.

- According to Du et al (2011), the relative covering element reduction learning method can be used in more complex tasks compared to previous related algorithms because it expands the application of covering element reduction to numerical type attributes. However compared to other rule learning methods, the number of rules derived from this algorithm is sometimes much more than other techniques. Only after a step of merging derived rules, the number of rules can be reduced.

- According to Kim et al (2010), the NDPMine method is an order of magnitude faster, significantly more memory efficient and more accurate than current approaches.

2.5. Our solution to solve this problem

Instead of using the simple decision trees we assume adding weights to each tree can improve the precision of the prediction, thus using boosted trees can improve our product classification. Similar to this, apart from traditional neural network model, we can also try giving weights to different neurons or adding more layers up to thousands to find out if it can improve our scenario. Besides all the methods, some simple exploratory analysis can tell us the relationship between each or more features, or a model selection before modeling can largely improve the efficiency and accuracy. For example, using variable importance in a random forest model to select the important features out and then apply to other models can help the model to make better decisions. However, all the assumptions above need to be tested out to make a standing point.

2.6. Where your solution different from others

Describe the differences between different classification methods here later

2.7. why your solution is better

The assumption is doing weighted trees or preliminary cleaning process can help with the accuracy of the prediction, this does not guarantee that our solution is better until test out.

3. Hypothesis

Successfully classify the product categories with precision 90%, in the end, visualize the model and classification such as:

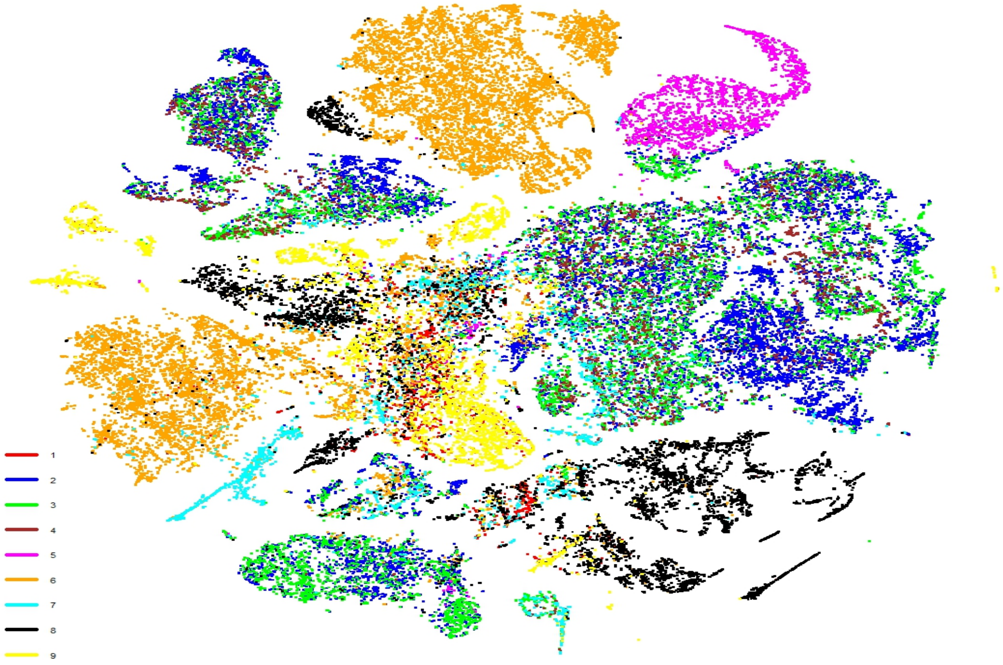


Figure 2. Visualization using t-SNE

4. Methodology

4.1 how to generate/collect input data

The input data already exists in csv format. Download the training data from: http://www.kaggle.com/c/otto-group-product-classification-challenge/data 61878 sets of training data in all

4.2 how to solve the problem

Classification is a supervised problem, so we will use training data sets to train the selected algorithms and models, then comparing the test error by cross-validation to find the best algorithm which can maximize the precision. Then using the newest technology to continuously improve the precision.

4.3 algorithm design

We will experiment and compare several algorithms as follows: SVM, neural network, logistic regression, naive bayes, random forest, decision trees.

4.4 language used

R, Python, Java

4.5 tools used

RStudio, iPython Notebook, Plot.ly, Eclipse

4.6 how to generate output

Our well trained algorithm will make the prediction from input data and output as a specific classification corresponded to the input. There are 9 classes totally in this project.

4.7 how to test against hypothesis

We will upload the prediction data to the Otto group for precision verification in order to compare with hypothesis.

4.8 how to proof correctness

We’ll use cross validation to proof the correctness of our methodology.

5. Implementation

5.1 Code (refer programming requirements)

Exploratory Analysis

Working environment: R, Rstudio, ggplot2, randomForest, Rstne
Feature correlation analysis:
https://github.com/lyladuan/kaggle/blob/master/correlation_visual.r
Feature importance analysis:https://github.com/lyladuan/kaggle/blob/master/varimp_rf.r
Classes visual using Rsne: https://github.com/lyladuan/kaggle/blob/master/rsne_visual.r
AUC ROC curve: https://github.com/lyladuan/kaggle/blob/master/roc_auc.r

Support vector machines (SVMs)

Working environment: Anaconda, IPython Notebook, Python 2.7, scikit-learn

https://github.com/edisonhmp/edisonhmp/blob/master/Support%20vector%20machines%20(SVMs)

Decision Trees (DTs)

Working environment: Anaconda, IPython Notebook, Python 2.7, scikit-learn

https://github.com/edisonhmp/Machine-Learning-Big-data-classification/blob/master/Decision%20Trees%20(DTs)

Extremely Randomized Trees

Working environment: Anaconda, IPython Notebook, Python 2.7, scikit-learn

https://github.com/edisonhmp/Machine-Learning-Big-data-classification/blob/master/Extremely%20Randomized%20Trees

Gradient Boosting Machines

Working environment: R, Rstudio, Xgboost, H2O, Python3.4, Graphlab
xgboost: https://github.com/lyladuan/kaggle/blob/master/xgboost.r
h2o: https://github.com/lyladuan/kaggle/blob/master/h2o_gbm_nn.r
Graphlab: https://github.com/lyladuan/kaggle/blob/master/graphlab.py

Deep Learning using Neural Network

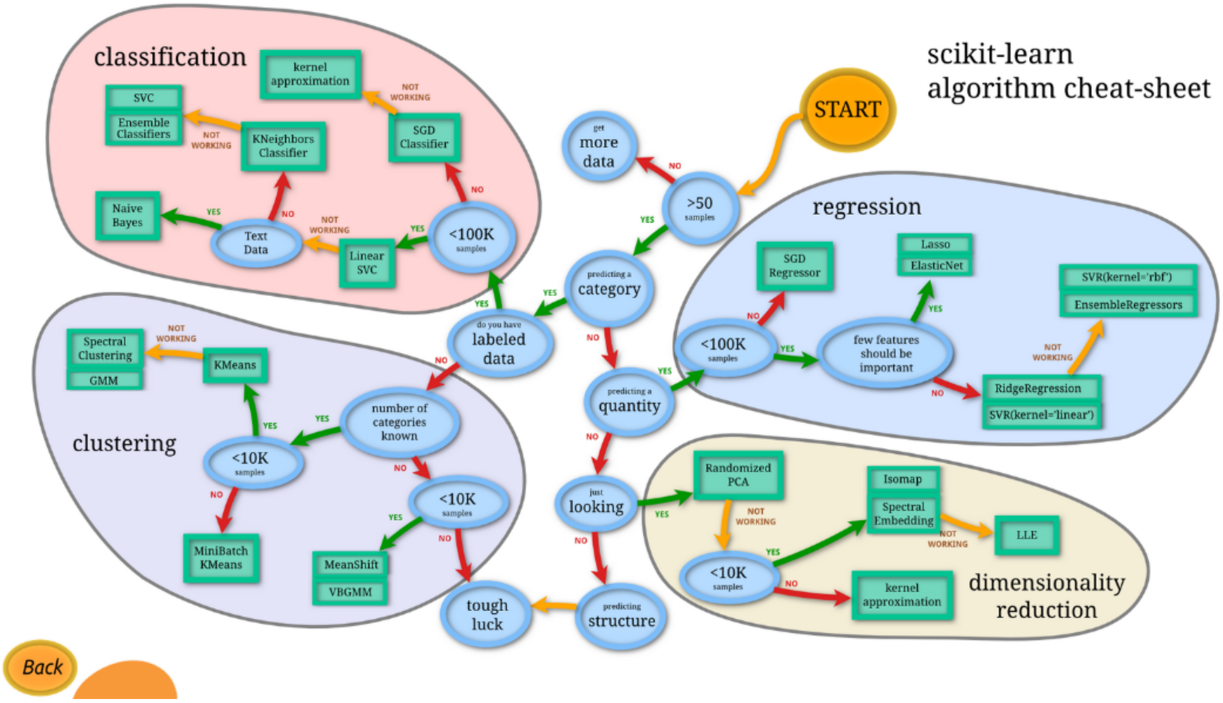
Working environment: R, Rstudio, H2O
h2o: https://github.com/lyladuan/kaggle/blob/master/h2o_gbm_nn.r
Ensemble using GBM and ANN

Working environment: R, Rstudio, H2O

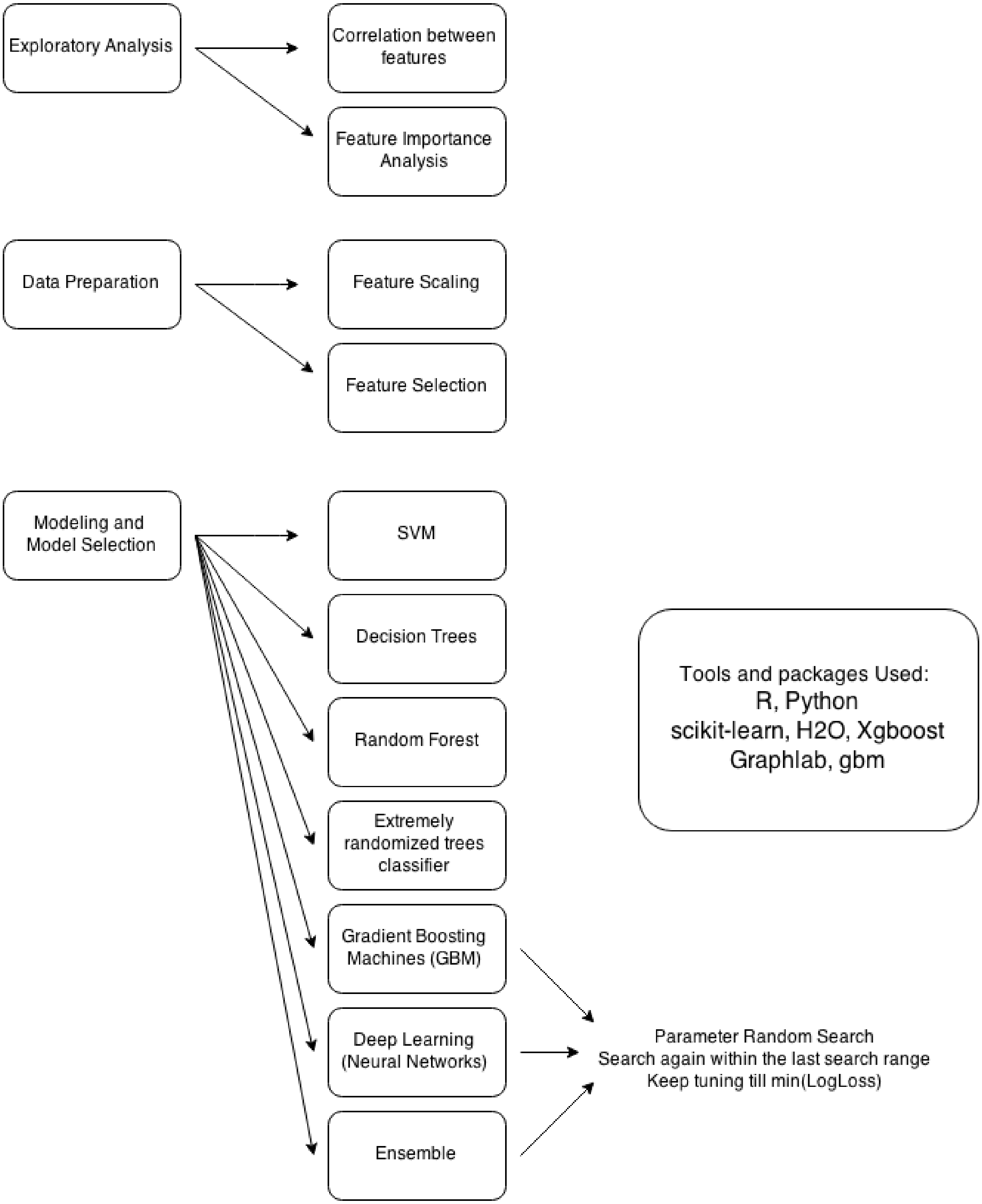
https://github.com/lyladuan/kaggle/blob/master/ensemble.r

5.2 Design document and flowchart

For python:
Design document: http://scikit-learn.org/stable/modules/classes.html
FlowChart : http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html



Flowchart of using Python and R to get the best model:


Exploratory Analysis
Correlation between features
Feature Importance Analysis
Data Preparation
Feature Scaling
Feature Selection
Modeling and Model Selection
SVM
Decision Trees
Random Forest
Extremely randomized trees classifier
Gradient Boosting Machines (GBM)
Deep Learning (Neural Networks)
Ensemble
Tools and packages Used:
R, Python
scikit-learn, H2O, Xgboost
Graphlab, gbm
Parameter Random Search
Search again within the last search range
Keep tuning till min(LogLoss)

6. Data analysis and discussion

6.1 output generation & output analysis

❖ **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:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- 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:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities, below).

1. Choosing parameters in Support Vector Classification:

C : float, optional (default=1.0)

Penalty parameter C of the error term.

kernel : Specifies the kernel type to be used in the algorithm. It must be one of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable.

degree : int, optional (default=3)

Degree of the polynomial kernel function (‘poly’). Ignored by all other kernels.

gamma : float, optional (default=0.0)

Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’. If gamma is 0.0 then 1/n_features will be used instead.

coef0 : float, optional (default=0.0)

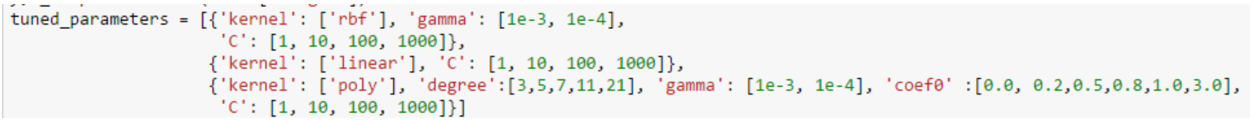
Independent term in kernel function. It is only significant in ‘poly’ and ‘sigmoid’.

2. The precision is bad if just randomly select the parameters in SVC.

3. Try several different Kernels and corresponding parameters, and evaluate their out-of-sample performance by using cross validation.

4. Doing cross validation one-by-one with changing kernels and other parameters are pretty slowly and inefficient.

5. Parameters that are not directly learnt within estimators can be set by searching a parameter space for the best Cross-validation score. Exhaustive Grid Search: exhaustively generates candidates from a grid of parameter values specified with the param_grid parameter.



```
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]},
                    {'kernel': ['poly'], 'degree':[3,5,7,11,21], 'gamma': [1e-3, 1e-4], 'coef0' :[0.0, 0.2,0.5,0.8,1.0,3.0],
                     'C': [1, 10, 100, 1000]}]
```

- **Decision Trees (DTs)** are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

  Some advantages of decision trees are:

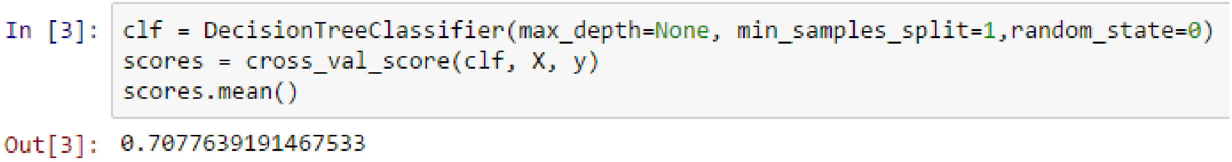
  - Simple to understand and to interpret. Trees can be visualised.
  - Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.

- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable. See algorithms for more information.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.

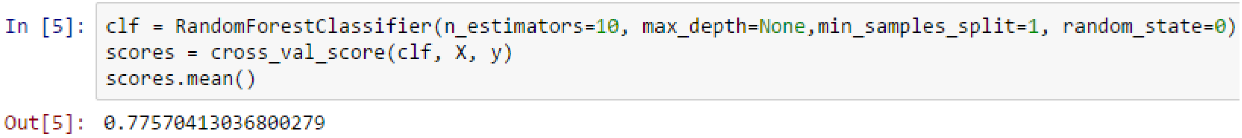
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.



```
In [3]: clf = DecisionTreeClassifier(max_depth=None, min_samples_split=1,random_state=0)
        scores = cross_val_score(clf, X, y)
        scores.mean()

Out[3]: 0.7077639191467533
```

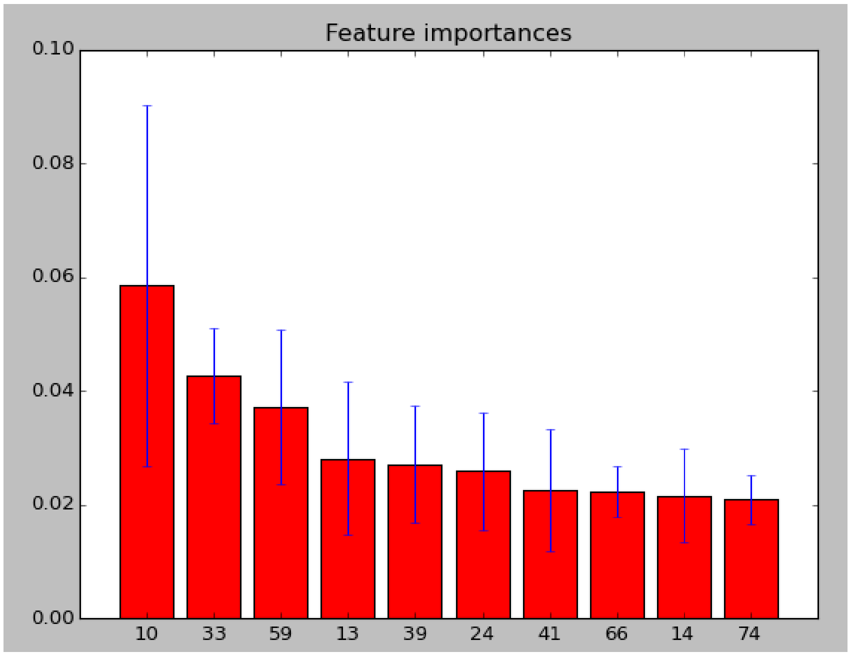
One estimator model is still not precise enough, so we use ensemble methods to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

❖ Random Forests: each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

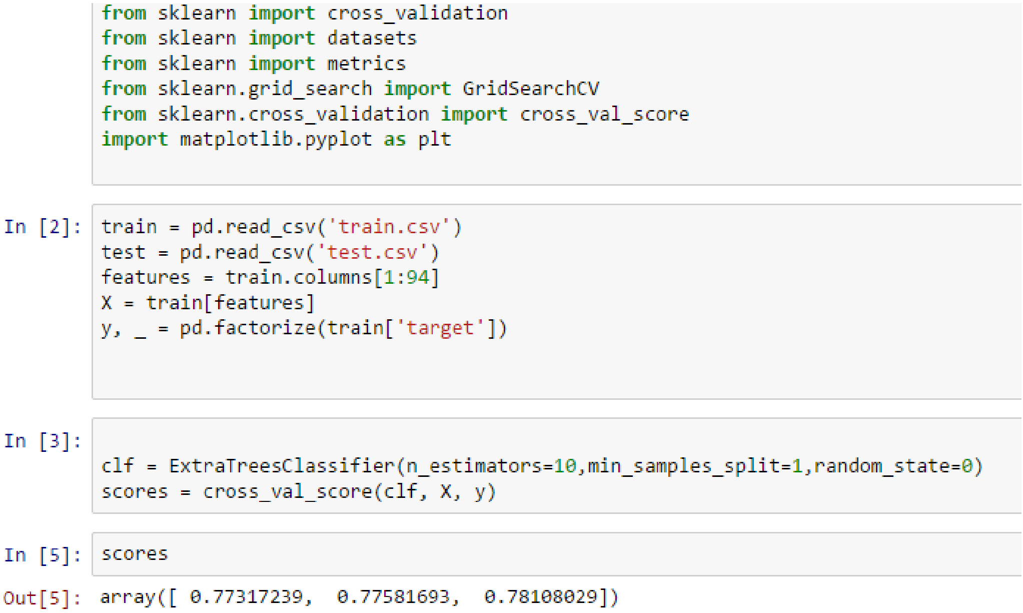


```
In [5]: clf = RandomForestClassifier(n_estimators=10, max_depth=None,min_samples_split=1, random_state=0)
        scores = cross_val_score(clf, X, y)
        scores.mean()

Out[5]: 0.7757041303680O279
```

❖ **Extremely randomized trees classifier** is one of the averaging ensemble methods, randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias.
Extremely randomized trees get the higher precision than the previous ones and also includes the technique of Feature importance evaluation. We use this function to plot the top 10 important features:



After estimate the accuracy of a Extremely randomized trees on the dataset by splitting the data, fitting a model and computing the score 3 consecutive times,the output mean score shows the precision is only around 78%

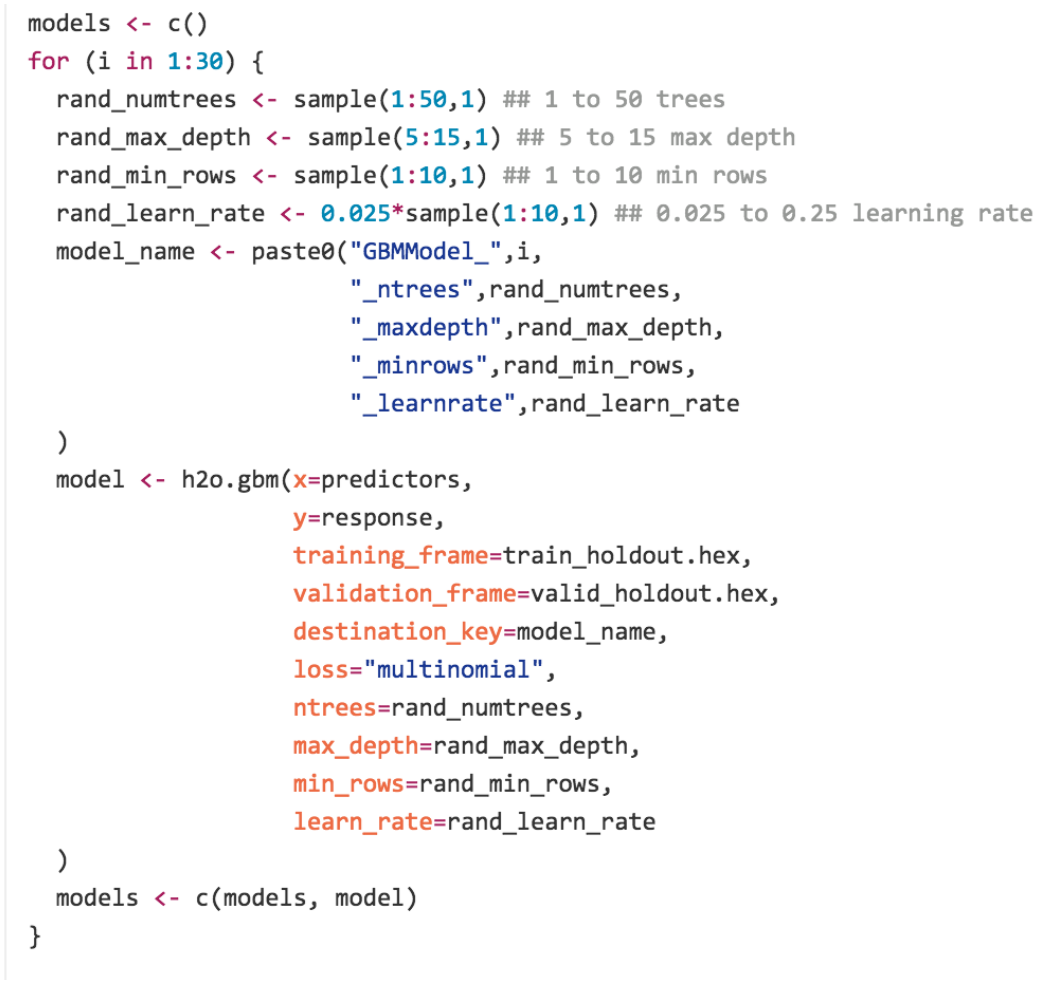


```
         from sklearn import cross_validation
         from sklearn import datasets
         from sklearn import metrics
         from sklearn.grid_search import GridSearchCV
         from sklearn.cross_validation import cross_val_score
         import matplotlib.pyplot as plt

In [2]:  train = pd.read_csv('train.csv')
         test = pd.read_csv('test.csv')
         features = train.columns[1:94]
         X = train[features]
         y, _ = pd.factorize(train['target'])

In [3]:
         clf = ExtraTreesClassifier(n_estimators=10,min_samples_split=1,random_state=0)
         scores = cross_val_score(clf, X, y)

In [5]:  scores

Out[5]:  array([ 0.77317239,  0.77581693,  0.78108029])
```

❖ Gradient Boosting Machine

Gradient boosting is a machine learning technique for regression problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The gradient boosting method can also be used for classification problems by reducing them to regression with a suitable loss function.

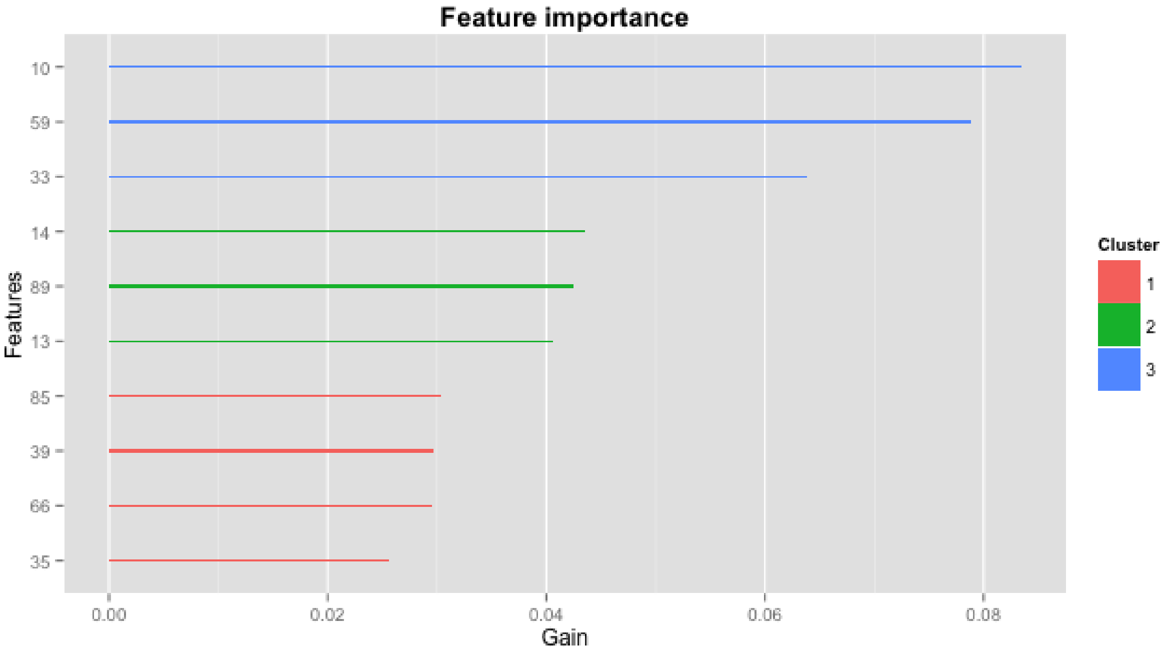
Using H2O in R, first we need to do a random search to determine the parameters that are using:



```r
models <- c()
for (i in 1:30) {
  rand_numtrees <- sample(1:50,1) ## 1 to 50 trees
  rand_max_depth <- sample(5:15,1) ## 5 to 15 max depth
  rand_min_rows <- sample(1:10,1) ## 1 to 10 min rows
  rand_learn_rate <- 0.025*sample(1:10,1) ## 0.025 to 0.25 learning rate
  model_name <- paste0("GBMModel_",i,
                       "_ntrees",rand_numtrees,
                       "_maxdepth",rand_max_depth,
                       "_minrows",rand_min_rows,
                       "_learnrate",rand_learn_rate
  )
  model <- h2o.gbm(x=predictors,
                   y=response,
                   training_frame=train_holdout.hex,
                   validation_frame=valid_holdout.hex,
                   destination_key=model_name,
                   loss="multinomial",
                   ntrees=rand_numtrees,
                   max_depth=rand_max_depth,
                   min_rows=rand_min_rows,
                   learn_rate=rand_learn_rate
  )
  models <- c(models, model)
}
```

This random search will keep tuning within the previous search range and search within the range to tune the parameters. For GBM model, the parameters need tuning are number of trees, maximum of depth, minimum of rows and the learning rate.

The following graph shows the feature importance according to the trees, feature 10, 59 33 are the top features. We choose the Logloss to be the criterion for selecting models since this is a classification problem and we try to minimize logloss.



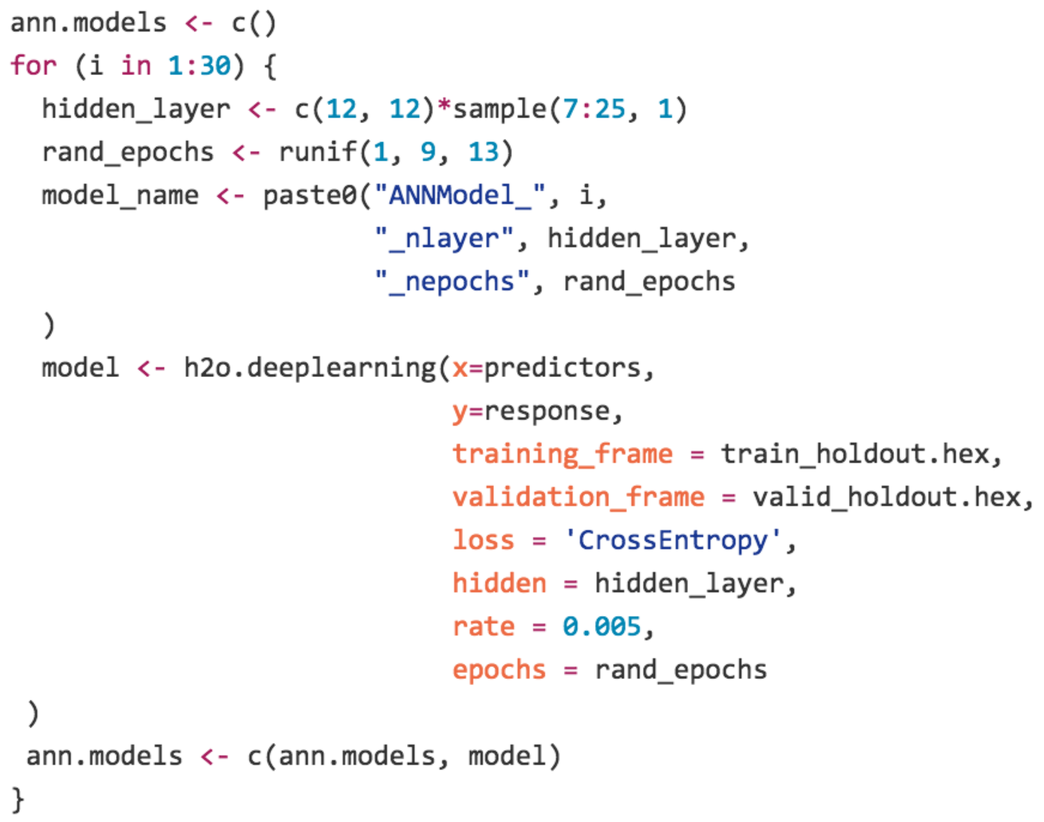
We reached the conclusion that the logloss are the same before feature scaling and after, The final parameters we chose are number of trees = 43, maximum depth = 9, minimum of rows = 9 and learning rate = 0.125; error rate is around 15%, logloss is 0.501.

- Deep learning -- Neural Network

Deep learning is part of a broader family of machine learning methods based on learning representations of data. An observation (e.g., an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, etc. Some representations make it easier to learn tasks (e.g., face recognition) from examples. One of the promises of deep learning is replacing handcrafted features with efficient algorithms for unsupervised or semi-supervised feature learning and hierarchical feature extraction.

H2O’s Deep Learning is based on a multi-layer feed-forward artificial neural network that is trained with stochastic gradient descent using back-propagation. The network can contain a large number of hidden layers consisting of neurons with tanh, rectifier and maxout activation functions. Advanced features such as adaptive learning rate, rate annealing, momentum training, dropout, L1 or L2 regularization, checkpointing and grid search enable high predictive accuracy. Each compute node trains a copy of the global model parameters on its local data with multi-threading (asynchronously), and contributes periodically to the global model via model averaging across the network.

First, perform a random search to determine the parameters.



```
ann.models <- c()
for (i in 1:30) {
  hidden_layer <- c(12, 12)*sample(7:25, 1)
  rand_epochs <- runif(1, 9, 13)
  model_name <- paste0("ANNModel_", i,
                       "_nlayer", hidden_layer,
                       "_nepochs", rand_epochs
  )
  model <- h2o.deeplearning(x=predictors,
                            y=response,
                            training_frame = train_holdout.hex,
                            validation_frame = valid_holdout.hex,
                            loss = 'CrossEntropy',
                            hidden = hidden_layer,
                            rate = 0.005,
                            epochs = rand_epochs
 )
 ann.models <- c(ann.models, model)
}
```

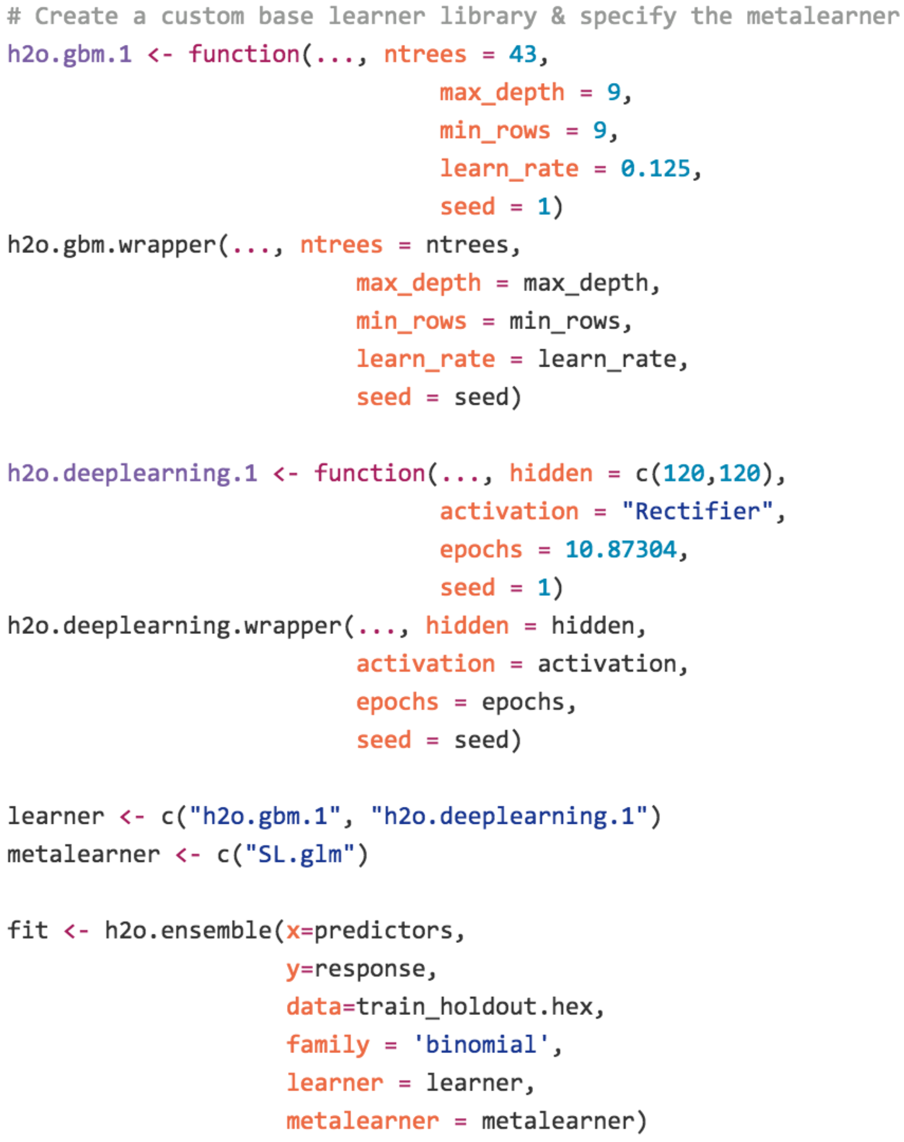
We have ran 200 Neural networks models to determine the one that minimized the logloss; the parameters are hidden layers = c(120, 120), learning rate = 0.005, epochs = 10.87304. Which give us around 16% error rate and the logloss is 0.51.

- Ensemble

In order to reach a better accuracy, we decide to try ensemble learning to mix several models together. In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms. Unlike a statistical ensemble in statistical mechanics, which is usually infinite, a machine learning ensemble refers only to a concrete finite set of alternative models, but typically allows for much more flexible structure to exist between those alternatives.

Here, we choose Bayesian model averaging to ensemble the GBM model and Neural Network model together. Bayesian model averaging (BMA) is an ensemble technique that seeks to approximate the Bayes Optimal Classifier by sampling hypotheses from the hypothesis space, and combining them using Bayes' law. Unlike the Bayes optimal classifier, Bayesian model averaging can be practically implemented. Hypotheses are typically sampled using a Monte Carlo sampling technique such as MCMC. For example, Gibbs sampling may be used to draw hypotheses that are representative of the distribution P(T|H). It has been shown that under certain circumstances, when hypotheses are drawn in this manner and averaged according to Bayes' law, this technique has an expected error that is bounded to be at most twice the expected error of the Bayes optimal classifier. Despite the theoretical correctness of this technique, it has been found to promote overfitting and to perform worse, empirically, compared to simpler ensemble techniques such as bagging; however, these conclusions appear to be based on a misunderstanding of the purpose of Bayesian model averaging vs. model combination.

To realize the above techniques:



```
# Create a custom base learner library & specify the metalearner
h2o.gbm.1 <- function(..., ntrees = 43,
                               max_depth = 9,
                               min_rows = 9,
                               learn_rate = 0.125,
                               seed = 1)
h2o.gbm.wrapper(..., ntrees = ntrees,
                         max_depth = max_depth,
                         min_rows = min_rows,
                         learn_rate = learn_rate,
                         seed = seed)

h2o.deeplearning.1 <- function(..., hidden = c(120,120),
                               activation = "Rectifier",
                               epochs = 10.87304,
                               seed = 1)
h2o.deeplearning.wrapper(..., hidden = hidden,
                         activation = activation,
                         epochs = epochs,
                         seed = seed)

learner <- c("h2o.gbm.1", "h2o.deeplearning.1")
metalearner <- c("SL.glm")

fit <- h2o.ensemble(x=predictors,
                    y=response,
                    data=train_holdout.hex,
                    family = 'binomial',
                    learner = learner,
                    metalearner = metalearner)
```

After the this step we get to error rate at 10% and logloss = 0.45598.

6.3 Compare output against hypothesis

The Forests of randomized trees or Gradient Tree Boosting are not precise as the hypothesis expected. So, we need to find another model or ensemble more other models together.

6.4 Abnormal case explanation

In the GBM boosting trees part, we found that if we keep boosting as more GBM with one tree each depth, we will eventually get better results. Some may takes weeks to finish the process, which can be essentially eliminating our error rate and good in the competition. However, this is not the purpose of learning this project, here we decide not to pursue the black box approach and decide to ensemble the two models together.

6.5 Discussion

To utilize an effective and good ensemble method, we have discussed a lot of different combinations of algorithms and core models.

7. conclusions and recommendations

7.1 summary and conclusions

In this project, we have tried 7 classification models(include the ensemble ones) to continuously decrease the error rate from 35% to 10%. The models we have used are listed as follows: SVM, Decision Tree, Random Forest, Extremely randomized trees classifier, Gradient Boosting Machine, Neural Network, Ensemble Gradient Boosting Machine and Neural Network with Bayesian model averaging. The lowest precision we can get is 65% from SVM, while the highest is 90% from Ensemble Gradient Boosting Machine and Neural Network with Bayesian model averaging. So, we can conclude that it is very hard for one pure training model to get a high precision even with the best parameters. For trees models, the more number of trees in parameter, the more precision it can get. But, in order to break the bottleneck, using ensemble techniques is necessary but time consuming.

7.2 recommendations for future studies

We can transmit csv training datasets into RDD.Then adopt spark distributed computing and Amazon EC2 to break the hardware limitation on our project. So, we can ensemble more training models and find optimal parameters for each of them. Then, we can get a higher precision.

8. bibliography

8.1. http://www.kaggle.com/c/otto-group-product-classification-challenge/data

8.2. Statistical classification in Wiki

8.3. Document Classification Methods for Organization Explicit Knowledge, Heide Brucher, Gerhard Knolmayer, Marc-Andre Mittermayer; University of Bern, Institute of Information Systems

8.4. Learning Bayesian Networks: Naïve and non-Naïve Bayes

8.5. Chimera: Large-Scale Classification using Machine Learning, Rules, and Crowdsourcing, Chong Sun, Narasimhan Rampalli, Frank Yang, AnHai Doan @WalmartLabs, University of Wisconsin-Madison

8.6. Logistic Regression as a Classifier

8.7. Using Support Vector Machines for classification tasks in Python

8.8. Supervised Learning: Support Vector Machines

8.9. Du, Y., Hu, Q., Zhu, P. and Ma, P., 2011. Rule learning for classification based on neighborhood covering reduction. Information Sciences (181) p5457-5467.

8.10. Kim, H., Kim, S., Weninger, T., Han, J. and Abdelzaher, T., 2010. NDPMine: Efficiently Mining Discriminative Numerical Features for Pattern-Based Classification. Machine learning and knowledge discovery in database (6322) p35-50

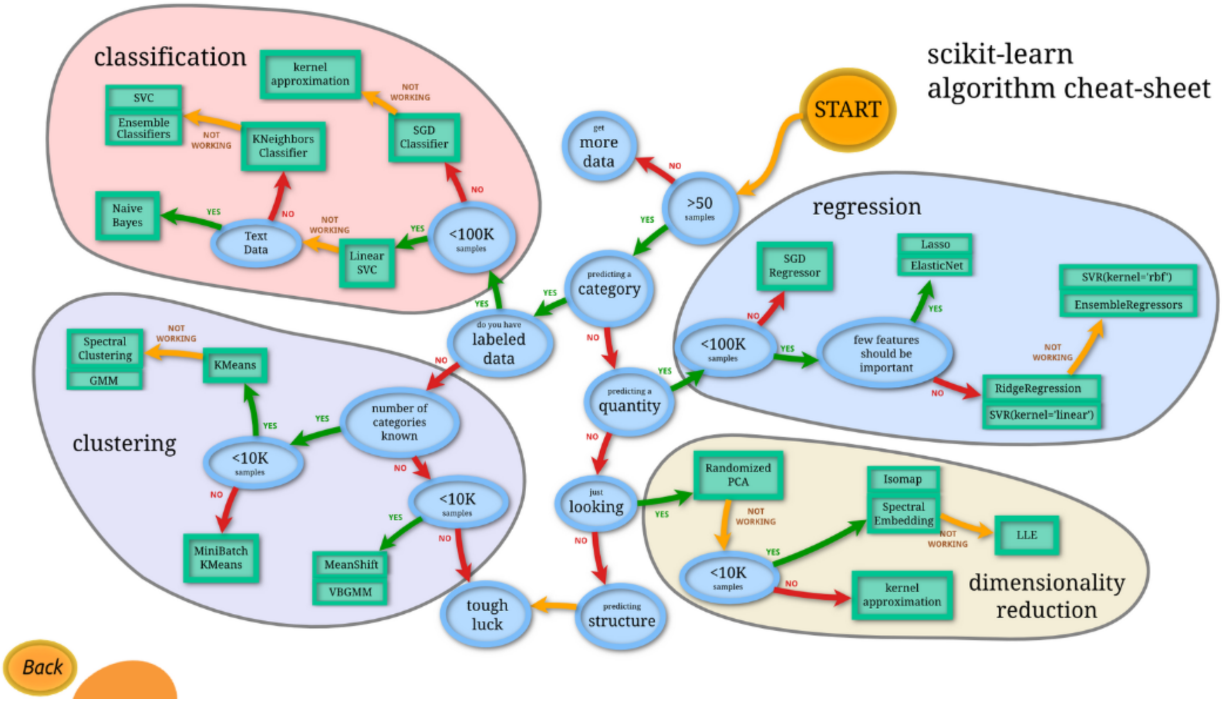
8.11. H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.

8.12. Scikit Learn

8.13. H2O documentation: http://docs.h2o.ai/

9. Appendices

9.1 program flowchart



9.2 program source code with documentation

Exploratory Analysis

Working environment: R, Rstudio, ggplot2, randomForest, Rstne
Feature correlation analysis:
https://github.com/lyladuan/kaggle/blob/master/correlation_visual.r
Feature importance analysis:https://github.com/lyladuan/kaggle/blob/master/varimp_rf.r
Classes visual using Rsne: https://github.com/lyladuan/kaggle/blob/master/rsne_visual.r
AUC ROC curve: https://github.com/lyladuan/kaggle/blob/master/roc_auc.r

Support vector machines (SVMs)

Working environment: Anaconda, IPython Notebook, Python 2.7, scikit-learn

https://github.com/edisonhmp/edisonhmp/blob/master/Support%20vector%20machines%20(SVMs)

Decision Trees (DTs)

Working environment: Anaconda, IPython Notebook, Python 2.7, scikit-learn

https://github.com/edisonhmp/Machine-Learning-Big-data-classification/blob/master/Decision%20Trees%20(DTs)

Extremely Randomized Trees

Working environment: Anaconda, IPython Notebook, Python 2.7, scikit-learn

https://github.com/edisonhmp/Machine-Learning-Big-data-classification/blob/master/Extremely%20Randomized%20Trees

Gradient Boosting Machines

Working environment: R, Rstudio, Xgboost, H2O, Python3.4, Graphlab
xgboost: https://github.com/lyladuan/kaggle/blob/master/xgboost.r
h2o: https://github.com/lyladuan/kaggle/blob/master/h2o_gbm_nn.r
Graphlab: https://github.com/lyladuan/kaggle/blob/master/graphlab.py

Deep Learning using Neural Network

Working environment: R, Rstudio, H2O
h2o: https://github.com/lyladuan/kaggle/blob/master/h2o_gbm_nn.r
Ensemble using GBM and ANN

Working environment: R, Rstudio, H2O

https://github.com/lyladuan/kaggle/blob/master/ensemble.r

9.3 input/output listing

Model Name	Parameters	Error rate
SVM	Exhaustive Grid Search	35%
Decision Tree	Default	30%

Random Forests	n_estimators=10,
	min_samples_split=1,
	random_state=0
		23%

Extremely randomized trees	n_estimators=10,	22%

classifier	min_samples_split=1, random_state=0	
Gradient Boosting Machine	number of trees = 43, maximum depth = 9, minimum of rows = 9 and learning rate = 0.125	15%
Deep learning -- Neural Network	hidden layers = c(120, 120), learning rate = 0.005, epochs = 10.87304.	16%
Ensemble with Bayesian model averaging	learner(GBM,Deep learning)	10%