Big Data Final Project Report

Titanic: Survival Prediction

This is a general statment for this project. It means nothing actually. What I want to do is just to increase nonsenses so that I can test whether genBlog.py works normally or not.

Teammate

- R02922164 邵　飛
- B00902064 宋昊恩
- B00902048 吳瑞洋
- B00902042 詹舜傑

Data Information

Input

- Features
  - Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked.
- Features Meaning
  - Pclass: class of accommodation.
  - SibSp: number of sibling and spouse on board.
  - Parch: number of parents and children on board.
  - Ticket: ticket id.
  - Fare: fare paid.
  - Cabin: cabin accomodated.
  - Embarked: port of embarkation.

Output

- Single Label
  - Survived or Not Survived (1/0).

Data Size

- Number of Instance
  - Train: 891
  - Test: 418
- Number of Feature
  - Numerical: 4
  - Categorical: 3
  - Nominated: 3

Onboard Evaluation

- Accuracy (TP/TP+FP)

Tools and Model Selection

Tools

- Pandas: Python package, used for *data manipulation*
- Sklearn: Python package, used for *data mining* and *data analysis*

Linear Model

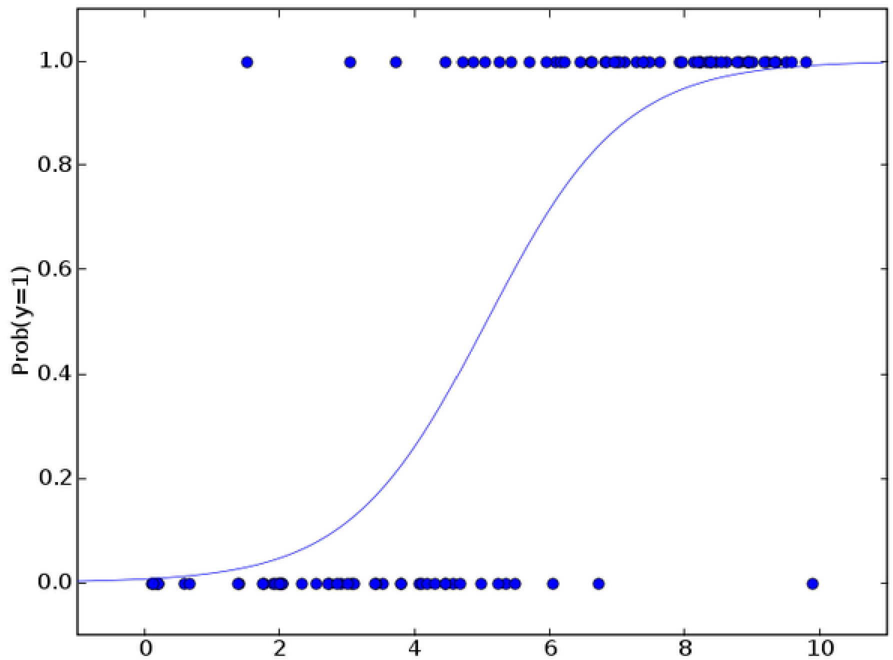
Logistic Regression

- Model Introduction
  - One of the linear models, which is widely used to solve machine learning problems

- Formula:

$$\min_{w} \quad \frac{1}{2}w^T w + C \sum_{i=1}^{l} \log(1 + e^{-y_i w^T x_i})$$

- Figure:

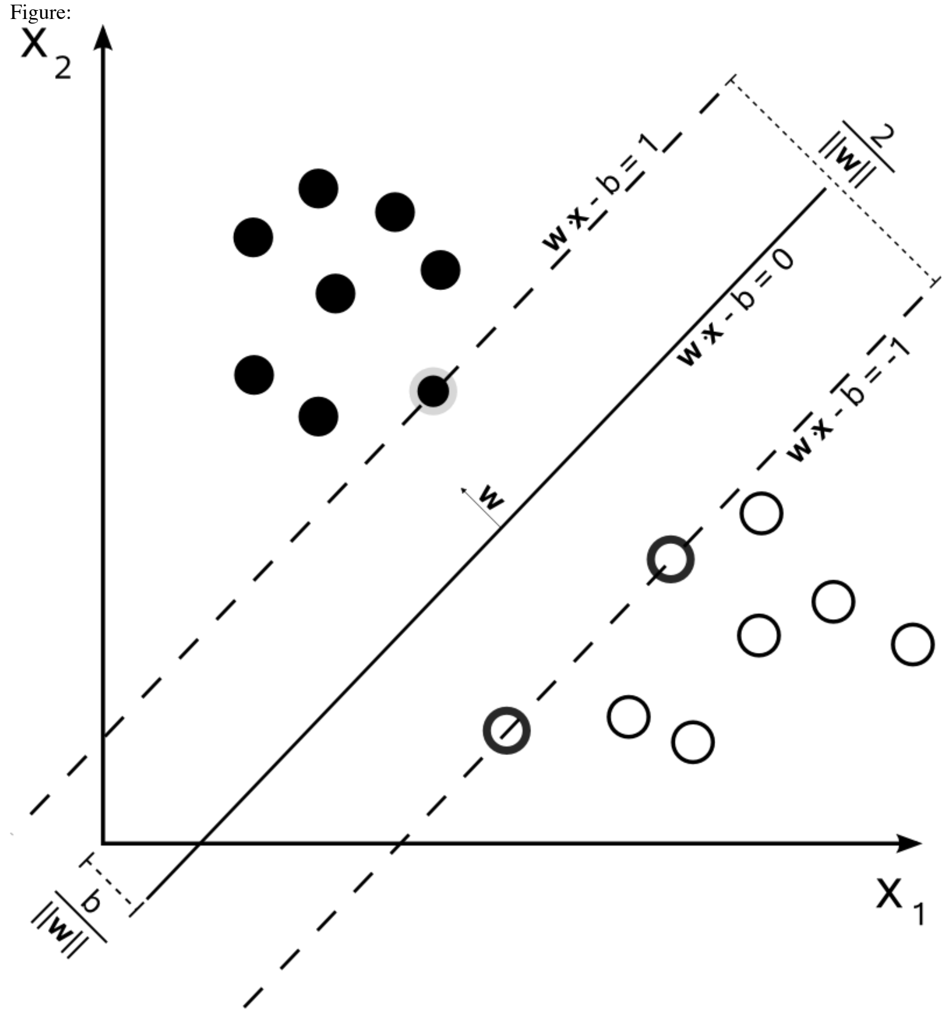


Linear Support Vector Machine

- Model Introduction
  - Model will try to find out a hyperplane to separate data points in the space spanned by features.
  - Formula:

$$\min_{w} \quad \frac{1}{2}w^T w + C \sum_{i=1}^{l} \xi(w; x_i, y_i)$$

- Figure:



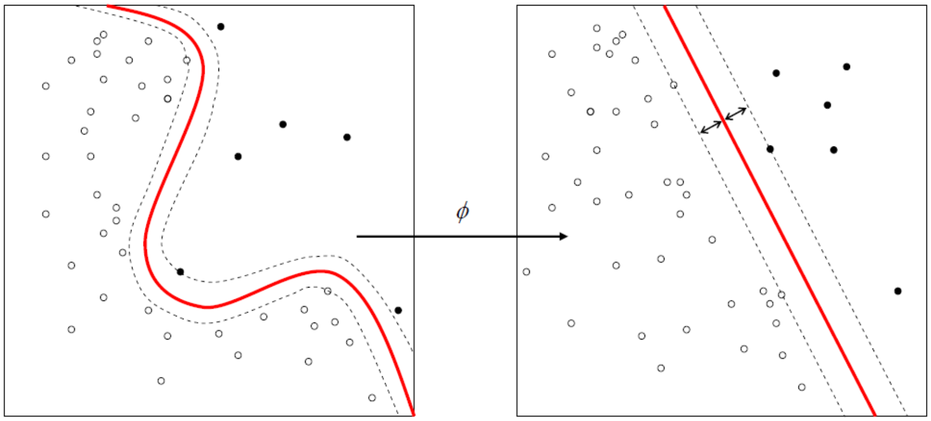
Kernel Model

Support Vector Machine

- Model Introduction
  - Model will use RBF kernel to map data points into space with infinite dimension, then try to find out a hyperplane to separate data points.
  - Its performance should cover *Linear SVC*.
  - Formula:

$$
\begin{aligned}
\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{l}\xi_i \\
\text{subject to} \quad & y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i)+b) \geq 1-\xi_i, \\
& \xi_i \geq 0, i = 1,\ldots,l,
\end{aligned}
$$

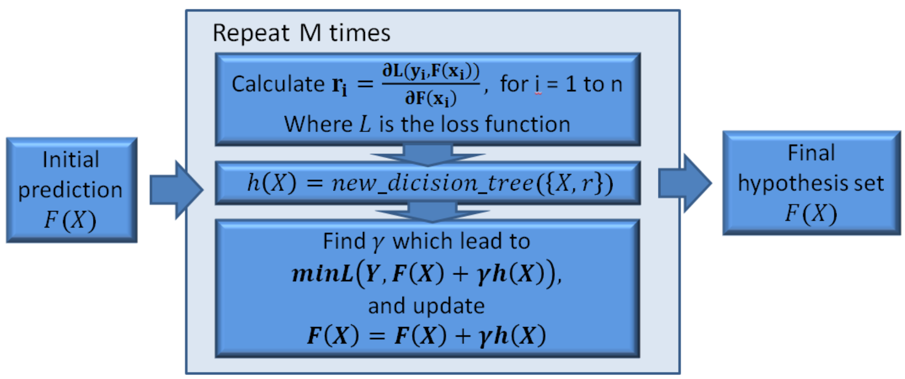
- Figure: (mapping into a space with higher dimension)



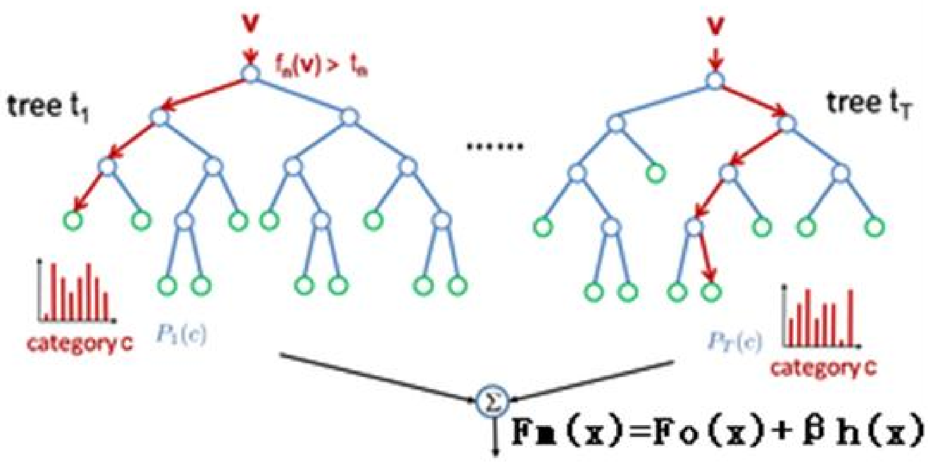
Tree-based Model

Gradient Boosting Classifier

- Model Introduction
  - Tree-based model with gradient descent update
  - Formula:



  - Figure:



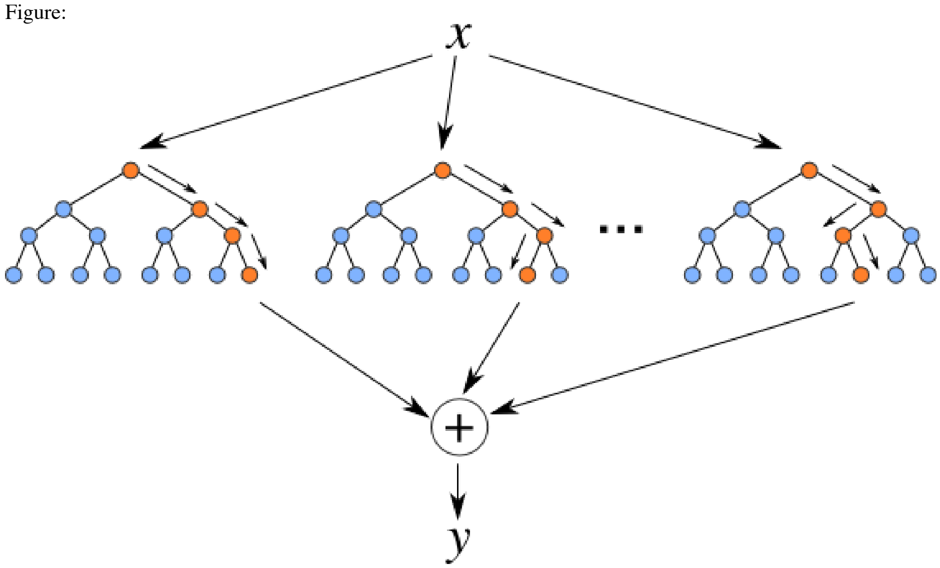
Random Forest Classifier

- Model Introduction
  - Tree-based model, ensembled with many out-of-bag decision trees

- Formula:

$$\hat{y} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} W_j(x_i, x')\right) y_i$$

- Figure:



AdaBoost Classifier

- Model Instruction
  - Selects only those features known to improve the predictive power of the model
  - Formula:

(a) Train classifier with respect to the weighted sample set $\{S, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \boldsymbol{x} \mapsto \{-1, +1\}$, i.e. $h_t = \mathcal{L}(S, \mathbf{d}^{(t)})$

(b) Calculate the weighted training error ε_t of h_t:

$$\varepsilon_t = \sum_{n=1}^{N} d_n^{(t)} \mathbf{I}(y_n \neq h_t(\boldsymbol{x}_n)) \ ,$$

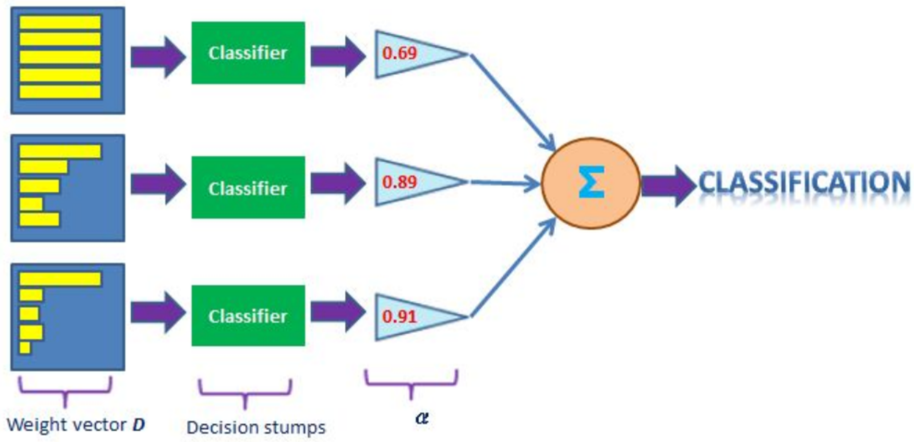
(c) Set:

$$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}$$

(d) Update weights:

$$d_n^{(t+1)} = d_n^{(t)} \exp\left\{-\alpha_t y_n h_t(\boldsymbol{x}_n)\right\} / Z_t \ ,$$

- Figure:



Feature Engineering

- Numerical
  - Features
    - Age, SibSp, Parch, Fare
  - Preprocessing
    - Impute NA with mean value
    - Standard-scaling
- Categorical
  - Features
    - Pclass, Sex, Embarked
  - Preprocessing
    - No NA is discovered
    - Binary-feature Expansion
- Nominated
  - Features
    - Name, Ticket, Cabin
  - Preprocessing
    - Lots of NA value (ex: more than 90% NA in *Cabin*)
    - Hard to use without adding human knowledge (ex: *Name*)
    - We just eliminate them in this step

Off-board Experiment Design

- Since there are few data for this problem, we must have a robust way to prevent overfitting. Then, we just apply 5-fold cross-validation for all model evaluation.

- Though we are really careful about the overfitting problem, we still find out that there are 0.04 percent difference in accuracy between off-board and on-board.

Model Performance Comparison

Linear Model

Logistic Regression

- Best Parameters C=10, random_state=514
- Performance

	Train	Test
Valid	0.80387	0.70020
Board		0.76555

Linear SVC

- Best parameters C=10, random_state=514
- Performance

	Train	Test
Valid	0.70078	0.79460
Board		0.75598

Kernel Model

Support Vector Machine

- Best Parameters: C=1, gamma=0.125, random_state=514
- Performance

	Train	Test
Valid	0.80387	0.70021
Board		0.76555

Tree-based Model

Gradient Boosting Classifier

- Best Parameters estimator=500, depth=5, random_state=514
- Performance

	Train	Test
Valid	0.89870	0.82041
Board		0.77990

Random Forest Classifier

- Best Parameters: estimator=20, depth=5, random_state=514
- Performance

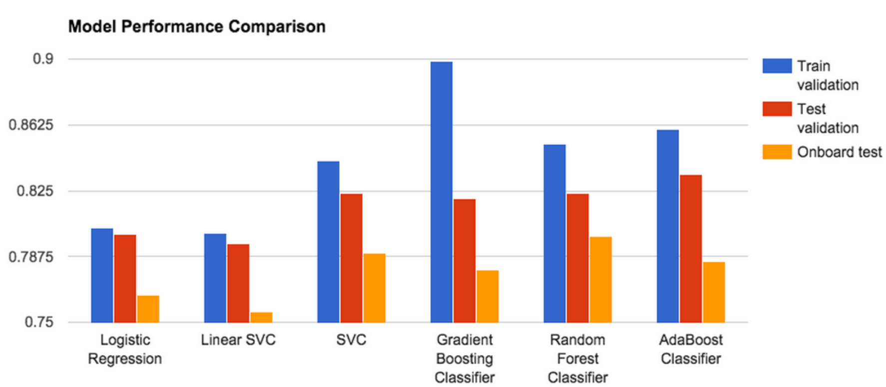
	Train	Test
Valid	0.85156	0.82378
Board		0.79904

AdaBoost Classifier

- Best Parameters estimator=30, depth=3, learning_rate= 0.2
- Performance

	Train	Test
Valid	0.85972	0.83438
Board		0.78469

Comparison from Figure



Model Ensemble

- We choose the best answer collected from each model, including SVC, GBM, Random Forest and Adaboost, and aggregate them to gain on-board score *0.79904*, which is exactly the same as the *Random Forest* one.
- One possible reason is that there is nearly nothing further can be learn from our current feature set, so different models have almost the same answer.
- To have advanced score, we can either put more efforts on nominated features or try robust feature selection for each model to enhance the model exclusiveness.

Conclusion

- We implement six ML models in this *Titanic* problem and get 0.79904 as our best result. There are several points we learn from this competition, listed as follows:
    - Some ML models have similar performance on one ML problem, i.e. Tree-based models.
    - Though some people make use of the well-known knowledge to gain 100 percent performance, this is not our main purpose in this competition. We just try to make use of what we have learned in this course.
    - There is a consistent gap between off-board and on-board score for all models. This may be caused by the imbalanced sampling in official data.

Reference

- Python Package: Scikit Learn http://scikit-learn.org/stable/
- Python Package: Pandas http://pandas.pydata.org/
- Python Software for Convex Optimization - Documentation http://cvxopt.org/
- A Library for Large Linear Classification http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf
- A Library for Support Vector Machine http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf
- Wiki page for Support Vector Machine https://en.wikipedia.org/wiki/Support_vector_machine
- Wiki page for Gradient Boosting https://en.wikipedia.org/wiki/Gradient_boosting
- Wiki page for Random Forest https://en.wikipedia.org/wiki/Random_forest
- Wiki page for AdaBoost https://en.wikipedia.org/wiki/AdaBoost