Machine Learning Practice for Multi-Classification Problem in Life Insurance

Han Zhang
28247418
hz1u16@soton.ac.uk

Fan Bi
28610016
fn1n15@soton.ac.uk

Ningning Liu
25903179
ln1u16@soton.ac.uk

Ruiqi Guo
28872738
Rg3n15@soton.ac.uk

Xin Zhang
28871219
xz5n15@soton.ac.uk

***Abstract* — the project aims to develop a predictive model that can classify risk levels using data points in the existing life insurance assessment. Successful model will help Prudential to better classify risk and offer a quote for new and existing customers quickly, enabling the corporation to make better decisions. We develop some preliminary models such as linear model, support vector machine, random forests etc., and ensemble them at later stage. The optimal quadratic weighted kappa we obtain from test set is 0.66 which represents relatively high accuracy in prediction. When dealing with massive categorical inputs and targets, random forests shows impressive performance in this project.**

I. INTRODUCTION

Life insurance is a type of insurance that the companies will pay the customers who died during the term of the contract. The product pricing will deeply depend on the health condition of clients, a correct measurement for risk classification for each client can significantly affect the company's profit. Mostly, life insurers find that it is very hard to win customers in nowadays competitive market. One strategy to win the market share is to make the quote attractive to customers without increasing too much risks. This means that insurer should offer the quotes properly according to the risk levels of customers. Traditionally the risk is assessed by manual underwriting classification on collected risk information such as BMI, age, sex, medical history etc. However, this approach is quite subjective, inaccurate and time-consuming. This conventional risk assessment method may result in many low-risk customers being overcharged and many high-risk customers being undercharged.

As time goes by, the client will run off from the company especially for low-risk customers, the loss of such portion client will significantly affect the revenue. To remain profitable, the insurer has to increase their average insurance quote that will intensifies the loss of low-risk customers. After several such vicious circles, the insurer may lose his competitiveness in the market.

Machine learning method has been highly qualified for solving classification problem and particularly performing well with the basis of large data. The purpose of the project is to develop a machine learning pipeline to increase the accuracy for client risk assessment, and further reduce the unnecessary cost.

By designing a machine learning model that can do risk assessment accurately using existing customers` information. We can help life insurers make their price both equitable and profitable. One big challenge of this project is that the data in the dataset provided is not complete. Mainly because many people want to keep their health condition confidential, and are afraid of the consequences of privacy disclosure. The dataset consists of 126 independent variables and the target value is composed of 8 distinct values which represent the customer's risk level. Almost 10% data is lack of 90% values so that a value imputing method is necessary and deserves tryout. However, the result shows the missing value is not as important as we expected. We impute the missing value with variable's mean and evaluate the performances against SVD method [1], the accuracy measured by quadratic-kappa function will just fluctuate in three decimal digits.

The heavy-imbalanced target value is another challenge, they are not distributed equally and not meet the assumption of many learning models. The most appearances target value almost has one fourth samples of total size, and the least target only has less than 5 percent appearances. Simply simulating samples for achieving a balanced distribution is an option but more bias would be introduced also. Instead of doing so, we compensate predictions with an offsetting vector which learned from original distribution to counter the imbalanced problem. Such method can provide 2 to 11 percent increase of accuracy for different models.

The project is expected to follow the subsequent stages accordingly. Firstly, pre-process the raw data in two steps begin with imputing missing values and later work on feature engineering. Next stage is dimensionality reduction over hundred variables. This can be done by feature extraction. Most importantly, we focus on model building and sustainable optimisation. Not only some preliminary models will be implemented, but also advanced method such as ensemble and offsetting will be tried out for better accuracy.

The predicted target using test set will be assessed by quadratic weighted kappa that measures the agreement between 8 possible risk ratings. Tools like Python is mainly used in this project in conjunction with several packages, e.g.

pandas, Matplotlib, numpy, scikit-learn and XGBoost. The version control and merging files is fulfilled by GitHub.

This article presents the performances of different machine learning techniques on the given data such as linear regression, support vector machine, random forests and etc. The second section provides the brief introduction of all the methods we used in this project. The third section compares the performances of the models in the experiments. Conclusions and potential future work is discussed in the last section.

II. Methodology

In this section, we will briefly discuss the methodology in filling missing data, feature selection and some preliminary models we built at initial stage.

2.1 Preprocessing Data

Data is normalized by its column mean and variance. If there is any missing value, we write NAN in the cell. Normalized data is extremely useful when we construct linear model and SVM. Random forest will take the original data without normalising.

2.2 Missing Data

We have 14 out of 126 variables containing missing values. Tables 1 shows the frequency and percentage of missing values of associated variables accordingly. There should be 79146 data points under each variable. Therefore, the percentage indicates the missing proportion out of 79146 entries. Medical history carries the most of missing values as being more than 90%. Hence, we proposed three possible solutions to handle missing data.

Variable	Counts	Percentage
Employment_Info_1	22	0.02%
Employment_Info_4	8916	11.3%
Employment_Info_6	14641	18.5%
Family_Hist_2	38536	48.7%
Family_Hist_3	45305	57.2%
Family_Hist_4	25861	32.7%
Family_Hist_5	55435	70.0%
Insurance_History_5	33501	42.3%
Medical_History_1	11861	15.0%
Medical_History_10	78388	99.0%
Medical_History_15	59460	75.1%
Medical_History_24	74165	93.7%
Medical_History_32	77688	98.2%

Table 1. Summary of missing variables

Solution 1

We deleted all of user's data containing missing values. Consequently, only 1% entries containing complete variables information. So this solution is not recommended.

Solution 2

We filled mean of each variable into corresponding missing values. Since the data was normalized in the initial stage, filling mean is a fairly common method.

Solution 3

Singular Valued Decomposition (SVD) is one of common methods in matrix completion [1, 2]. Krucz et al. used SVD to fill in missing values through iterations [1].

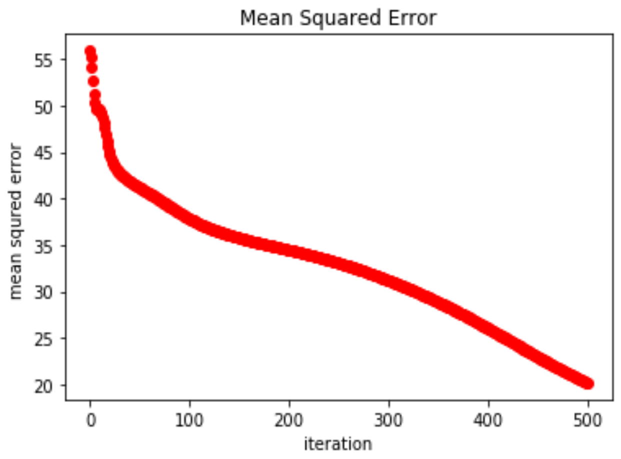

Fig 1. Mean square error of SVD data against original data converges when iteration increase

Algorithm I : Singular Valued Decomposition

```
Input: D
Output :
Initialize: error = 1000, k = 0 and iteration = 100
$\mu$= col_mean(D)
B , $\underline{B}$ = binary mark where there is / no data
$D_0 = B \otimes D + \underline{B} \otimes (\mu 1^T)$
While error > 0.01 OR k<= iteration
  $D_k = USV^T$
  $S' = S$ where $S_{ij} > s^*$
  $D'_k = US'V^T$
  error = norm ($B \otimes D - B \otimes D'_k$ )
  $D_{k+1} = B \otimes D + \underline{B} \otimes D'_k$
  If error < 0.01 OR k>= iteration
    Set K = k
  else
    Set k = k +1
End While
$D^* = D_K$
Return $D^*$
```

By applying the similar idea, **Algorithm I** shows the pseudocode of filling missing data using SVD. Suppose the original normalized data set is D, D^* is the resulting data after filling missing values. At each iteration of SVD, we only keep high rank values $S_{ij} > s^*$. In original data set, S_0 is ranging from 460 to 26. So we let s^* be 30 to eliminate zero and small ranks. Mean Squared Error (MSE) is used to measure the distance between SVD data set and original one at each iteration. Fig1 show there is clear convergence of MSE when iteration is increasing. However, SVD is a time-consuming algorithm. So, we let machine to compute SVD up to 100 times.

2.3 Feaure Selection

There are usually many features in databases in the real world, but some features may be irrelevant or redundant. Feature selection is a tool that can be used to eliminate these features. Furthermore, the reduction of features can also decrease the time of executing. To some extent, it can also prevent overfitting. Our dataset has 126 features. It is a large dimension and the target are labelled with numbers from 1 to 8.

Furthermore, there are some features which have high correlation with each other, and the combination of interacting features also have strong correlation with the target variable, even though there are weak correlation between the individuals and response variable [4]. We calculate the correlation between the continuous features in the dataset.

Features` name	BMI&Wt	Family_His_4 &Ins_Age	BMI&Ht	Family_His_4& Family_His_2
Correlation	0.85	0.63	0.61	0.60

Table 2. The first four correlation between continuous features

And we can see from the table above that there is high correlation between BMI and Wt. Hence, we combine BMI and Wt together as a new feature.

In the term of removing the irrelevant features, we used two approaches. The first one is Linear model with L1-norm as the penalty term which can increase the sparsity of model, so it can enforce the coefficient of features which have weak correlation with target variable to be zero, to get the purpose of eliminating irrelevant features. We used SVM model with linear kernel and L1-norm term to remove the redundant features [4].

The second one is Tree based methods which can estimate the importance of features, so the features which are less important can be removed with meta-transformer. In this case, we used Extra-trees classifier as the estimator which will use the whole train samples compared with other tree-based approaches [12].

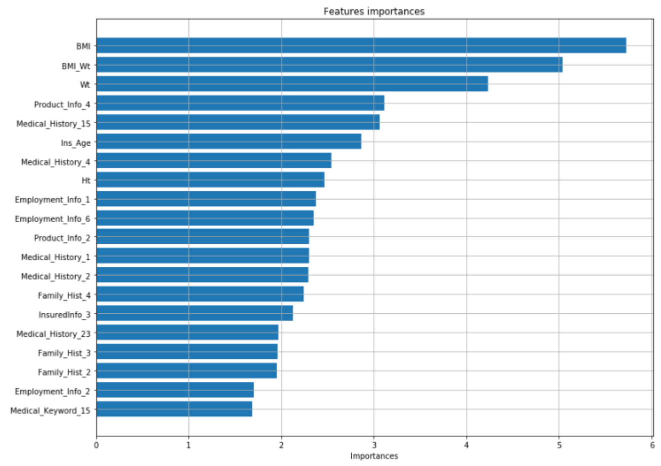

Fig 2. The first 20 important features

It is can be seen from Fig. 2 that personal information is more important than other factors.

2.4 Preliminary Modeling

In this section, the proposed models are constructed individually.

2.4.1 Linear Model

Suppose the data set is $X = \{x_{ij} | i = 1,2,...,N, j = 1,2,...,M\}$, and target y, we would like to obtain optimal weights **w** for linear model with some penalty. By weight decay, we minimise the error E(y) to obtain the weights.

$$E(y) = \| w^T X - y \|^2 + v \| W \|^2 \tag{1}$$

We have three set features selection to trial on the linear model. The results are shown in experiment part.

2.4.2 Support Vector Machine

Support vector machine is a machine learning model that separate datasets using a hyperplane. SVM selects a robust classification by maximizing the margin between different classes and the hyperplane. If the data is not linearly separable, we can use kernel functions to project the data into a higher dimensional space to make the separation easier. The using of appropriate kernels can also improve the classification ability of SVM.

There are 8 classes in this problem. SVM can separate multiclass by solving several binary classification problems which separate one of the classes and the others. After this process, we can use a max-wins voting method to finally decide every example's class.

In order to increase the generalization ability of SVM, we allow some data fall into the wrong side of the margin. The penalty parameter C controls the trade-off between the margin-size and the number of misclassified examples. Small C makes a soft margin, while large C makes a hard margin. Bias increases as C decreases, and vice-versa.

Three kernels are used in our project, which are radial basis function, polynomial and sigmoid. The radial basis function kernel is defined as:

$$K_{rbf}(x, x') = exp(\gamma||x - x'||^2)] \tag{2}$$

RBF kernel is a very popular kernel function in classification and widely used. $||x - x'||^2$ is the squared Euclidean distance between these two feature vectors. γ decides the shape of the kernel function. And bias decreases as γ increases, and vice-versa.

The polynomial kernel is defined as:

$$K(x, y) = (x^T y + c)^d \quad (3)$$

Polynomial kernel is more popular in natural language processing [10]. d sets the degree of the polynomial function. Larger degrees can reduce bias but tend to overfit the data.

An SVM model with sigmoid kernel is similar to a two-layer neural network. In general, the sigmoid kernel is not better than RBF and linear kernels [11].

We use data with features that are selected by different methods of feature selection to figure out which kind of feature selection can generate the best result and reduce computing time. From Table 4 we can see that features selected by LinearSVC with L1 can generate the best result. The performances of other feature selections are worse than that of the original data. Since the difference is small, we can shorten the computing time by keeping only the important features.

Although we use grid search to find the ideal parameters for rbf kernels, some other combinations of parameters are also tested. Table 3 shows the Kappa value of the test data achieved by different parameters using the best feature selection method. The kernels we used are radial basis, polynomial and sigmoid functions. The degrees of the polynomial function are 3 and 9. Table 3 shows that in most cases the radial basis function is better than other kernels and the accuracy of grid search of rbf kernel is the highest. Nevertheless, the result of the grid search is just a little better than default parameters in sklearn. As can be seen in Table 4, SVMs with RBF kernel have a slightly better performance than other models. Overall SVMs did a good job of insurance assessment.

Feature Selection	Fill missing data	Kappa value of the testing data				
		Kernel	C=0.01	C=1	C=10	Grid search: C=2, gamma= 0.5/89
LinearSVC with L1 norm C = 0.01	SVD	RBF	0.52406	0.56789	0.55232	0.56875
		Sigmoid	0.42696	0.33258	0.31972	
		Ploy(d=3)	0.41977	0.51852	0.50473	
		Poly(d=9)	0.20764	0.27744	0.34804	
	Mean	RBF	0.51876	0.56597	0.5922	

Table 3. Kappa value of SVM in different kernels

2.4.3 XGBoost

Xgboost (Extreme gradient boosted trees) is a fasting and accurate algorithm developed from gradient boosting scheme which performs very well with tabular data [5, 6]. The gradient boosting method is built upon weak learners through learning residual recursively instead of simply voting or averaging results. The benefit of boosting algorithm is that it tends to produce large margin classifiers, which defined as the difference between the total weights assigned to the correct label and the largest assigned to an incorrect label. Hence, the margin can be interpreted as an indication of the confidence to correct classification. Overall, if a combination of classifiers correctly classifies most of the training data with a large margin, then its error probability is small. The way of iteratively adding residual to loss function is an additive training which can shallow the depth of trees compared with random forest.

We presumed this kind of ensemble method could have a good fitting and generalizing ability for our dataset, since the variability of thousands of customers' records are dramatic and models are easily suffered from the overfitting problem. Xgboost is an ensemble method which uses many trees to take a decision so it gains power by repeating itself. Therefore, it would help to defeat overfitting through reducing variance on unseen data predicting.

Finally, there are dozens of hyper-parameters can be tuned in xgboost architecture, most of them can control the complexity and improve the generalizing ability. For example, the number of trees, the maximum depth of single tree, the objective function, the sub-sample ratio and the magnitude of regularization term. Few of them will significantly be tryout in our case, more experiment and analysis will be discussed in experiment section.

2.4.4 Random Forest

Random forest [7] is another tree-based ensemble learning method we implemented to compare against gradient boosting model. The merit is same as the inherently good with ensemble and such bagging scheme can correct overfitting problem habited from individual tree. In contrast with xgboost, it constructs multiple fully grown decision trees other than weaker learners at training time and outputs the classes with the mean of prediction of those based trees. The trees are made uncorrelated to maximize the decrease in variance (differ from gradient boosting), but the algorithm cannot reduce bias. Therefore, it requires unpruned and complexity trees, so that the bias is initially as low as possible.

The random forest model is prone to overfitting for the reason of low bias, thus the parameter for deviating the problem of low bias is way more important to be tuned. For example, the number of base tree can influence the generalizing ability a lot. With increasing the number of trees, the averaged prediction is more generic than individual tree. Furthermore, the tree depth and number of features will impact the tradeoff between variance and bias, more features and deeper layers can lead to overfitting a lot on unseen data predicting. Considering that the dataset is inherent complexity itself with more than 100 features and 50000 records. We will try to achieve a compromising result with tuning the random forest parameter in next section.

III. Experiments

3.1 Preliminary Models

In order to evaluate whether feature selection can have a positive impact on models` predicting, we set up the experiment to compare the performance of different techniques and the results can be viewed in Table 4. It shows that all feature selection processes will more or less decrease the accuracy in all kinds of preliminary models except for SVM. As less features we feed in the models, the models tend to return a lower accuracy of predicting. From where interested us of adding more useful features which would result in higher accuracy, the detail and implement will be discussed in the ensemble models part.

Feature Selection	Filling Missing Data	Linear Model	SVM('rbf' , C=1, γ=default)	Random Forest	Xgboost
Keep all variables	Mean	0.54591	0.56056	0.49346	0.52975
	SVD	0.53734	0.55990	0.48014	0.52350
LinearSVC to select features	Mean	0.54528	**0.56597**	0.52181	0.52975
	SVD	0.54318	0.56789	0.48771	0.50970
Extra Trees	Mean	0.49490	0.52695	0.48407	0.51606
	SVD	0.49332	0.52882	0.46990	0.50457

Table 4. Preliminary Kappa Value

The best performance achieved by SVM with a kappa score of **0.56597**, which is way better than random forest and xgboost. However, the tree-based ensemble scheme with fine-tuned parameter almost dominates the prediction problem with tabular data particularly, that would encourage us to give more shots on tuning parameters with random forest and xgboost. The result can be viewed below in Table 5.

One further effort we made is treating the problem as a regression rather than classification. For the reason of imbalanced dataset we have as Fig 3 shows, the targets are not distributed equally. On the one hand, classification method prone to result in a very sparse predicting (lack of several classes). On the other hand, we believe that regression method can beat the problem with a more diversity results. Furthermore, we try to compensate the values with the information learned from origin distribution instead of simply rounding up and down the results to integral number. The more details can be viewed in the offset part.

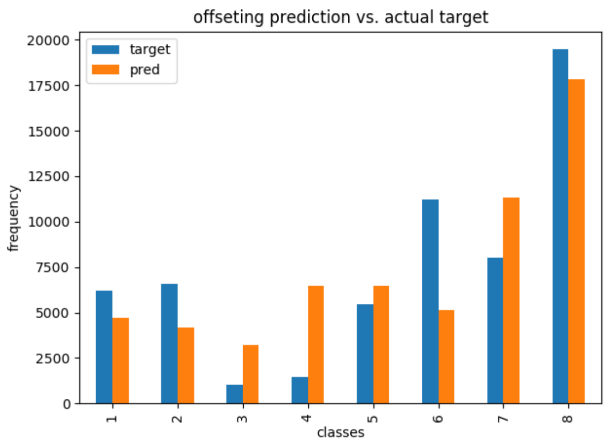

Fig 3. Offsetting Prediction vs. Actual Target

3.2 Ensemble Models

Considering that one learning model cannot give competitive results comparing with ensemble learning, a method uses multiple algorithm to obtain better predictive performance is widely implemented [8]. Random forest scheme and gradient boosting has been mentioned before are two popular and mature techniques for generating multiple hypotheses using the same base learner. However, more general ensemble approach could be used to combine several different base learners, where would be linear regression, SVM, random forest and xgboost in our case.

We propose two methods for constructing ensemble framework as Fig 4 shows, one (**Ensemble One** in Fig 4) is just averaging predictions among those four base learners. However, it turns out a worse result compared with other methods and even poorer than single models like a fine-tuned svm. A potential improvement can be made by applying an extra linear regression upon based on learner's predictions and the actual predictions. By doing so, there will be four coefficients where the bigger value means the higher importance for predicting actual classes. The scheme can be done in future work and also including more based models to construct a neural network style framework worth a tryout. This kind of implementation will not be presented in this paper due to the reason of time consuming.

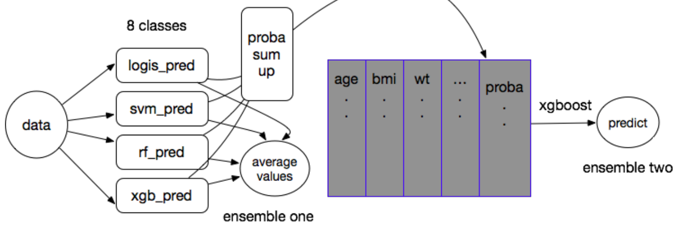

Fig 4. Ensembling Framework

A more interesting ensembling method (**Ensemble Two** in Fig 4) we implemented is focus on improving the predictive power with data itself. To exceed the limitations of normal feature engineering, we still train four based models for predicting but obtain the probabilities rather than real values this time. Since the way of predicting classes of multi-classifier model depends on probabilities, where model returns the class value with biggest corresponding probabilities. We lately feed those probabilities vector probamodel $proba_{model} = [class_1, class_2, \ldots, class_8]$ that generated by base models back to training data and add them up$proba_{add} = [class_{add1}, class_{add2}, \ldots, class_{add8}]$ to get eight new columns of input. The reason for using probabilities are comes from the diversity concerning. We assume such way can diversify the data better rather than putting back the real class value. This method has reached the highest accuracy (**0.64250**) so far in without combining offset method.

In Table 5, we show the experiment result with kappa score which derived from the ensemble methods we proposed above. We verified that converting the problem from classification to regression truly improve the accuracy a lot, and the biggest increasing happened in re-training probs model from **0.59091** to **0.64250**. As we stated before, averaging predictions returns a poorly result in comparing with other methods.

Model\Ensemble	Classifier	Regressor (round)	Regressor (offset)
Random Forest	0.53571	0.56812	0.58404
Xgboost	0.58329	0.60694	0.66038
Multiple Models Averaging	0.54355	0.55485	0.66038
Multiple Models probe re-training(xgboost)	0.59091	0.64250	**0.66605**

Table 5. Ensemble Kappa Value

3.3 Offset

A further attempt we did is trying to compensate the predicting results based on the information learned from original distribution (Fig 5). On doing so, we initialize a compensate vector $offset = [off_1, off_{2,}, \ldots off_8]$ for each class value and the final value will be added or subtracted from it. The vector was optimized from learning the best offsetting value from base model predictions to actual target value, it will iteratively move the offset value to reach a better kappa score till convergence.

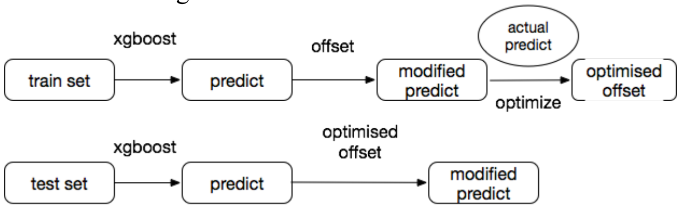

Fig 5. Offset Framework

Adding the predicting distribution with final offset will make its distribution approach to the original imbalanced shape. The result can be viewed in the last column of Table 5, a further improvement is achieved by applying such method and it completely outperform any other methods that we have implemented. It defeats the imbalanced problem with least effort and finally generate the most promising result so far with the accuracy of **0.66605**.

IV. Discussion and Future work

During the project, we become acquainted with project pipeline in machine learning through a realistic application in the industry. We intensively use automated algorithms to continuously optimise the prediction model as well as compare both of excellence and drawbacks during the exploration in each state-of-the-art approach.

The ensemble models can provide higher accuracy compared with other models we have used. The reason is that ensemble models can combine advantages of each model together. However, it may need more time to generate the model. The feature selection does not influence much on the accuracy. The problem can be that we did not choose the optimal subset of features. We may use some approaches like PCA to remove the redundant features and use cross-validation to choose the optimal subset of features in the future work.

For more complex ensembling method, we suggest that more based learners can be included in future implementation to achieve a higher accuracy. And the way of initializing offset vector can also impact the final accuracy to some extent. In contrast of assigning equal intervals among 0 to 1 to initilize the offset vector, a further method can be developed with the information of data distribution.

To sort the imbalanced dataset out entirely, one attempt can be made by generating similar sample based on existed user records to enlarge dataset. Such technique has already been implemented in sufficiency the training images for deep learning [9].

References

[1] M. Krucz, A. A. Benczur, K. Csalogany, "Methods for large scale SVD with missing values," *in Proc of Conf. on Knowledge Discovery and Data Mining*, California, Aug. 2007

[2] M. G. Vozalis, K. G. Margaritis, " Applying SVD on generalized item-based filtering," *Int J. of Computer Sci & Applications*, 2006, Vol 3, Issue 3, pp.27-51.

[3] Blum, A.L. and Langley, P., 1997. Selection of relevant features and examples in machine learning. *Artificial intelligence*, *97*(1), pp.245-271.

[4] De Silva, A. and Leong, P. (2015). Feature Selection. In: A. De Silva and P. Leong, ed., *Grammar-Based Feature Generation for Time-Series Prediction*, 1st ed. Singapore: Springer Singapore, pp.13-24.

[5] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]//Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016: 785-794.

[6] Friedman J H. Greedy function approximation: a gradient boosting machine[J]. Annals of statistics, 2001: 1189-1232.

[7] Ho T K. Random decision forests[C]//Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on. IEEE, 1995, 1: 278-282.

[8] Dietterich T G. Ensemble methods in machine learning[C]//International workshop on multiple classifier systems. Springer Berlin Heidelberg, 2000: 1-15.

[9] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[10] Chang, Y.W., Hsieh, C.J., Chang, K.W., Ringgaard, M. and Lin,

C.J., 2010. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, *11*(Apr), pp.1471-1490.

[11] Amami, R., Ayed, D.B. and Ellouze, N., 2015. Practical selection of SVM supervised parameters with different feature representations for vowel recognition. *arXiv preprint arXiv:1507.06020*.

[12] Geurts, P., Ernst, D. and Wehenkel, L., 2006. Extremely randomized trees. *Machine learning*, *63*(1), pp.3-42.

The Distribution of Marks

Name	ECS ID	Percentage	Signature	Date
Han Zhang	28247418	23.75%		12/5/2017
Ningning Liu	25903179	23.75%		12/5/2017
Fan Bi	28610016	23.75%		12/5/2017
Ruiqi Guo	28872738	5%		12/5/2017
Xin Zhang	28871219	23.75%		12/5/2017