COMP9417 Project: Santander Product Recommendation

Link to the competition:https://www.kaggle.com/c/santander-product-recommendation/overview

Team members:

Name	zID
Yucheng Lu	z5297999
Jiayu Liu	z5235411
Yikai Han	z5276871
Shanshan Wu	z5275276

1 Introduction

As customers' demands for a series of financial decisions increase, Santander provides loans to customers through personalized product recommendations. Under their current system, a small number of Santander customers will receive a lot of recommendations, while many other customers rarely see any recommendations. This results in uneven customer experience. So Tande Bank challenged Kagglers to predict which products the existing customers will use in the next month based on customers' past behavior and the behavior of similar customers.

The goal of this competition is to predict which new Santander products, if any, a customer will purchase in the following month. In this project, we aim to build a set of models to create a more effective recommendation system, and ensure that Santander can better meet the individual needs of all customers. And we will do some data cleaning, adjust some features, and do some visualization to get a sense of what features might be important predictors.

We will deal with this problem in two ways. One is binary classification on 24 separate labels. The other is multi-class classification. The evaluation metric for our project will also be separated in two parts. For binary classification, it will be accuracy of each target. For multi-class classification, it will be accuracy of the whole multi-label.

2 Implementation

2.1 Choice of model and hyperparameter

The table below shows the models and hyperparameters that we will analyze in this report.

Table 2.1 models and hyperparameters

Model	Hyperparameter	Effect	Values tested
Decision Tree	criterion	The function to measure the quality of a split.	'gini' or 'entropy'
	splitter	The strategy used to choose the split at each node.	'best' or 'random'
	max_depth	The maximum depth of the tree to avoid overfitting.	from 2 to 8
Multinomial Naive Bayes	alpha	Additive smoothing parameter to avoid some features with a probability of zero.	from 0 to 1, at 0.1 increments
K-Nearest Neighbor	n_neighbors	Number of neighbours	from 1 to 5
	weights	Weight function used in prediction	'uniform' or 'distance'
Logistic Regression	solver	Optimization strategy	'lbfgs' or 'liblinear'
	C	Inverse of regularization strength, smaller values mean stronger regulariztion	0.001, 0.01, 0.1, 1, 10
Support Vector Classification	kernel	Kernel function	'poly' or 'rbf'
	degree	The degree of 'poly' kernel function	from 1 to 4
	gamma	The coefficient for 'rbf'	0.001, 0.01, 0.1, 1
	C	Inverse of regularization strength	0.001, 0.01, 0.1, 1
Multi-layer Perceptron Classifier	activation	Activation function at hidden layer	'logistic' or 'tanh'
	hidden_layer_sizes	The number of neurons in the hidden layer	(100,), (100,30), (100,60), (100,90)
	alpha	L2 penalty parameter	0.001, 0.01, 0.1, 1

Random Forest	n_estimators	It determines the number of sub-trees. It should be determined first and choose as high value as processor can handle and the model will not be overfitting.	from 10 to 201, at 10 increments
	max_depth	It limits the maximum depth of each decision tree.	from 1 to 8
	min_sample_leaf	Leaf is the end node of a decision tree. A smaller leaf makes the model more prone to capturing noise in train data.	30, 40, 50, 60
XGBoost	booster	Select the type of model to run at each iteration	'gbtree' or 'gblinear'
	learning_rate	Learning rate in gradient boosting.	0.1, 0.3
	reg_lambda	Use L2 regularization term on weights	1, 0
	max_ depth	The maximum depth of a tree.	from 3 to 10
	min_child_weight	Defines the minimum sum of weights of all observations required in a child.	from 1 to 5

2.2 Parameter search strategy

In order to find the set of robust parameters for a model, we use GridSearchCV, which is the same search strategy used in GSK Ranjan(2019). GridSearchCV can find the best parameters which optimize the model by cross-validation over the given parameter grid. Hence, there are three important terms for GridSearchCV. The first one is 'estimator', which is the classifier model to train data. The next is 'parameter grid', which contains all parameter names with their ranges of value to be tested for the best outcomes. The final one is 'cv', which means the cross-validation splitting strategy. In this report, we use 5-fold cross validation. The figure shows the working process of GridSearchCV (GSK Ranjan 2019).

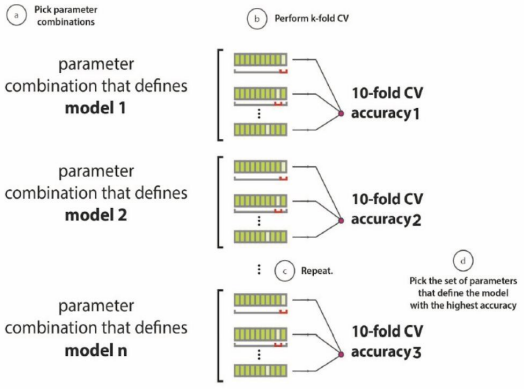


Fig 2.1 working process of GridSearchCV

2.3 Model evaluation method

The equations below show how precision and recall are computed.

Actual Class	Predicted Class	
	Positive	Negative
Positive	True Positive(TP)	False Negative(FN)
Negative	False Positive(FP)	True Negative(TN)

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

Recall checks whether the Real Positive cases are correctly predicted while precision tests whether the Predicted Positive cases are the Real Positive cases. Machine Learning should focus on whether the predicted cases are the real cases (David Martin 2008). Hence, we will use precision to evaluate the performance of the models.

Because this competition is a multi-label classification task. We use Hamming loss to evaluate the performance of model in Random Forest and XGBoost which originally support multi-label classification or use multiclass classifier to support multi-label classification. The equation of Hamming loss is as follows.

$$\text{hloss}(h) = \frac{1}{p} \sum_{i=1}^{p} |h(x_i \Delta Y)|$$

2.4 Implementation process

The first step is data preprocessing. According to the description of each feature, we deal with missing values in each feature with different strategies. We then transform variables with string values, which represent different categories, into variables with integer values.

The next step is to split data. Since the training set contains more than 13000000 records of data, we sample around 50% of training set. We then compute the correlation coefficient matrix and exclude one of the features which are highly associated from our model.

We then use GridSearchCV to search best parameters for different models. After finding the best parameters, we rebuild the model with best parameters and then compute the accuracy of each model on the test data.

The final step is to evaluate the performance of each model and come to the conclusion that which model is the best performer.

3 Experimentation

3.1 Exploratory data analysis

3.1.1 Missing values

We use different strategies to handle missing values of different features. The simplest strategy is to delete rows of data with missing values. For features representing categories, we can assign the most common category or a new category for missing values. For numeric features, we can replace missing values with the median of total data, or the median of data group by other features. Take the feature 'renta' as an example, since the median value of this feature can be influenced by the customer's province, we replace the missing values with the median 'renta' of the customer's province. The following table shows the way to process missing values for all features.

Table 3.1 way to process missing values of all features

Feature Name	Description	Processing Missing Values
ind_empleado	the employee index	This is an important feature which defines the user portrait. We use NA to fix missing values and we will encode this feature in the feature engineering process.
pais_residencia	a customer's country residence	It may indicate that people in different areas will have different consumer preferences.
age	a customer's age	Firstly, we convert age to numeric values. We then estimate the distribution of age in a histogram and we find that this is a bimodal distribution. The first peak is for young men aged 18-28 and the other is men aged 30-45. So we use 40, which is the mean age from 18 to 100, to fill in missing values. (The histogram is in appendix.)
ind_nuevo	It determines whether a customer has registered in the last 6 months or not.	We have 27734 missing values. We could use the active information to fill this value, because we find that the maximum number of active records is 6 times, then we use 1 to fill missing values.
fecha_alta	the date of consumption	We fill the missing value with the median date.
sexo	a gender feature	There are also many missing values in this column, so we use NA to replace missing values. Then we have three genders: Male, Female and NA.
antiguedad	the month-long period of customer seniority	It has an outlier value of 999999 which will affect some numerically-sensitive models like linear regression. We set all negative values to 0.
indrel	1 (First/Primary), 99 (Primary customer during the month but not at the end of the month)	We find that the common status is 1, so we replace missing values with 1.
ult_fec_cli_1t	the last date as primary customer, if it is not at the end of the month	Since almost all values are missing, we just exclude this feature from the model.
indrel_1mes	customer type at the beginning of month: 1 (First/Primary customer), 2 (co-owner), P (Potential), 3 (former primary), 4 (former co-owner)	Based on the definitions of these types, we replace missing values with 'P', which means potential customers.

tiprel_1mes	relation type at the beginning of month: A (active), I (inactive), P (former customer), R (Potential)	Based on the definitions of these types, we replace missing values with 'A', which means active relation.
indresi	The value is 'S' if the customer's residence country is the same as the bank country. Otherwise, the value is 'N'.	Although we observe that most of the customers' residence countries are the same as bank countries, we cannot replace missing values with 'S'. So we just delete this part of data.
indext	The value is 'N' if the customer's birth country is different from bank country. Otherwise, the value is 'S'.	Although we observe that most of the customers' birth countries are different from bank countries, we cannot replace missing values with 'N'. So we just delete this part of data.
conyuemp	The value of 'conyuemp' is 1 if the customer is the spouse of an employee.	Since this feature has huge amount of missing values, we decide not to use this feature in the model.
canal_entrada	channels used by the customer to join	We notice that the majority of customers join through three major channels: 'KHE', 'KAT' and 'KFC'. Since we cannot replace missing values with a certain channel, we just delete this part of data.
indfall	deceased index with value 'N' or 'S'	We assume that those missing values mean no decease. So we assign the value 'N' to those missing values.
tipodom	address type: 1 for primary address	Except for missing values, this feature only has one value. Besides, address type seems to have nothing to do with purchasing new products. So we decide to exclude this feature from our model.
cod_prov	the province code of each customer	Since these two features have the same meaning, we just need to remain one feature and handle its missing values. For this report, we remain 'cod_prov' and assign a new category for missing values. The value of this new category is 0 and its meaning is that the province of customer in this category is 'Unknown'.
nomprov	the corresponding province name	
ind_actividad_cliente	Activity index: 1 for active customer, 0 for inactive customer	The amount of missing values in 'ind_actividad_cliente' is the same as that in 'pais_residencia', 'fecha_alta' and etc. We find that several features of these records of data do not have value. Hence, we just exclude this part of data from the model.
renta	the gross income of household	Instead of replacing the missing value with the median of all gross household incomes, it is more accurate to assign missing incomes by province (Alan 2016). Hence, we replace missing values with the median of gross income by province. Note that if all values in one province are missing values, we replace these missing values with the median of all gross household incomes.
segmento	the segmentation of customer: 01 - VIP, 02 - Individuals, 03 - college graduated	The way to handle missing values is the same as that in 'cod_prov'. We set a new category for missing values. The value of this new category is 0 and the meaning is that the segmentation of customer in this category is 'Unknown'.

3.1.2 Feature engineering

There are many string values in the dataset, but many models cannot process these values. Although, from an algorithm perspective, Decision Tree and Random forest can use string values, the implementation of sk-learn can not use string values. In fact, most algorithms work better with numerical inputs like neural networks. Hence, we need to transform string values of features into numerical values.

We mainly focus on 2 methods: One-Hot-Encoder and Label-Encoder. We use Label-Encoder to convert string values, which represent different labels, to numbers. However, some categorical data is not in order. So using Label-Encoder may cause the problem of creating unreal relationships between features. Therefore, we should use One-Hot-Encoder to transform unordered category data like province names in this dataset. However, when we use features encoded by One-Hot-Encoder in model training, we find that the performances of many models decline. The reason is that One-Hot-Encoder increases model complexity.

Another problem is the imbalanced labels. We sort the number of samples in each label, this dataset has a huge current

account label where 2 index. It could affect the preference of model. Then we resample this dataset to balance the distribution of labels.

3.1.3 Correlation analysis

We compute the correlation coefficient matrix and use seaborn to draw the heatmap of Pearson correlation coefficient. The output figure is as follows. We find that the feature 'ncodpers' and feature 'antiguedad' are highly associated. This means we cannot include both of these two features in the model because it will cause difficulty to determine which feature is actually affecting Y. We also find that the feature 'tiprel_1mes' and feature 'ind_actividad_cliente' are highly associated. So we should exclude one of these two features from our model.

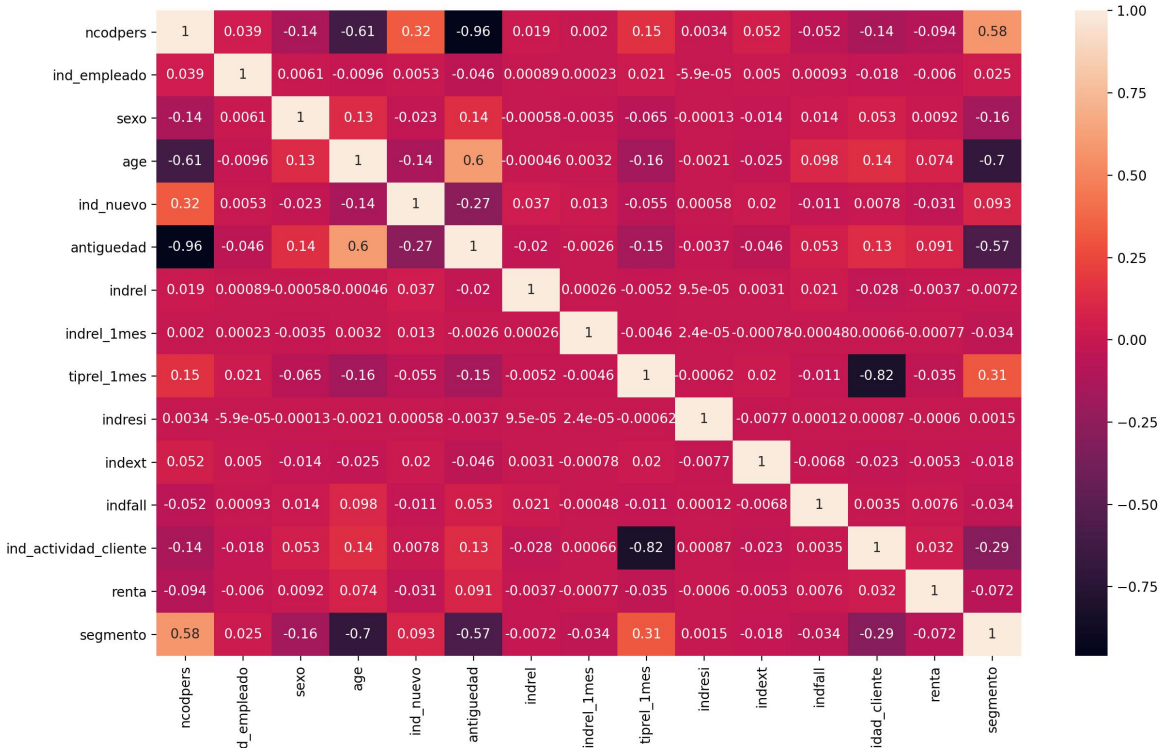


Fig 3.1 heatmap of correlation coefficient matrix

3.2 Model results

3.2.1 Linear Regression model

In order to run this model properly, we need to normalize the features in the data set such that each feature has zero mean and the one standard deviation. Besides, in order to use this model for classification task, we define the scoring rule as if the predict value more than 0.5, then the predict class is 1. Otherwise, the predict class is 0. We select the first 65000 samples to train the model and use samples from 65001 to 70000 to evaluate the accuracy by 5-fold cross validation. The results are shown in table3.3.

3.2.2 Decision Tree model

One advantage of Decision Tree is that exploratory data analysis can be easier than other models because Decision Tree can handle both numerical and categorical data (Scikit-learn). This model can also deal with multi-output problems. Hence, it is a good choice to use Decision Tree model in this task because it can make the exploratory data analysis of data set become easier. In order to compare the results with the Linear Regression model, we still train and test 20 labels one by one. We select the first 65000 samples to train the model.

Since we use GridSearchCV to search the best parameter for this model among a set of specified parameter values, we

decide to search three parameters. One is ‘criterion’ with value ‘gini’ or ‘entropy’. This is the function to measure the quality of split. ‘gini’ represents the Gini coefficient and ‘entropy’ represents the information gain. The next is ‘splitter’ with value ‘best’ or ‘random’. This is the strategy used to choose the split at each node. ‘best’ finds the optimal partition point among all the partition points of the feature while ‘random’ is to randomly find the local optimal partition point among the partial partition points. The final one is ‘max_depth’. This is the maximum depth of the tree to avoid overfitting. For this GridSearchCV, we take values from 2 to 8 to test the best value for ‘max_depth’. The table below is the outcomes.

Table 3.2 search results for two Decision Tree model

label	criterion=‘gini’				criterion=‘entropy’			
	splitter=’best’		splitter=’random’		splitter=’best’		splitter=’random’	
	best Max depth	best score	best Max depth	best score	best max depth	best score	best max depth	best score
Ahor_fin	2	0.9998	2	0.9998	2	0.9998	2	0.9998
Aval_fin	2	0.9999	2	0.9999	2	0.9999	2	0.9999
Cco_fin	8	0.7414	5	0.7301	8	0.7423	8	0.7310
Cder_fin	2	0.9997	7	0.9997	2	0.9997	2	0.9997
Cno_fin	6	0.9185	6	0.9188	7	0.9187	6	0.9183
Ctju_fin	3	0.9986	8	0.9955	4	0.9986	7	0.9949
Ctma_fin	2	0.9903	8	0.9903	2	0.9903	2	0.9903
Ctop_fin	5	0.8829	8	0.8790	7	0.8832	7	0.8819
Ctpp_fin	6	0.9574	2	0.9557	7	0.9575	7	0.9564
Deco_fin	2	0.9985	5	0.9985	2	0.9984	2	0.9984
Deme_fin	2	0.9903	2	0.9987	2	0.9987	2	0.9987
Dela_fin	5	0.9581	5	0.9575	8	0.9584	6	0.9573
Ecue_fin	7	0.9197	8	0.9194	7	0.9196	6	0.9192
Fond_fin	2	0.9809	2	0.9809	2	0.9809	2	0.9809
Gip_fin	2	0.9943	2	0.9943	2	0.9943	2	0.9943
Plan_fin	2	0.9907	2	0.9907	2	0.9907	2	0.9907
Pres_fin	2	0.9975	2	0.9974	2	0.9974	2	0.9974
Reca_fin	2	0.9479	5	0.9479	2	0.9478	6	0.9479
Tjcr_fin	2	0.9553	2	0.9553	2	0.9553	2	0.9553
Valo_fin	2	0.9734	2	0.9733	2	0.9733	2	0.9733
Viv_fin	2	0.9959	2	0.9959	2	0.9959	2	0.9959
Nomina	2	0.9441	7	0.9442	8	0.9441	5	0.9441
Nom_pens	2	0.9392	8	0.9393	2	0.9392	7	0.9393
Recibo	7	0.8736	7	0.8738	7	0.8739	8	0.8742

The outcomes show that the value for ‘max_depth’ is not stable, so we cannot find a fixed value for ‘max_depth’. This may be because that Decision Tree tends to be overfiting. We use samples from 65001 to 70000 to evaluate the accuracy by 5-fold cross validation. The results are shown in table3.3.

3.2.3 Multinomial Naive Bayes model

In order to compare the results with the Linear Regression model, we still train and test 20 labels one by one. We select the first 65000 samples to train the model. Through grid search to find best parameter smoothing alpha, we observe that the best alpha of all of the labels are 0. We use samples from 65001 to 70000 to evaluate the accuracy by 5-fold cross validation. The results are shown in table3.3. It is interesting to see that the performance of Multinomial Naive Bayes model is the worst. This may be because Multinomial Naive Bayes model is overfitting.

Table 3.3 test data accuracy for three models

Label	Linear Regression		Decision Tree		Multinomial Naive Bayes	
	mean	std	mean	std	mean	std
Ahor_fin	0.9998	0.0004	0.9997	0.0004	0.7604	0.1202
Aval_fin	1.0	0.0	1.0	0.0	1.0	0.0
Cco_fin	0.7208	0.0091	0.7373	0.0087	0.5856	0.0084
Cder_fin	0.9998	0.0004	0.9996	0.0004	0.6228	0.1894
Cno_fin	0.9234	0.0080	0.9192	0.0075	0.621	0.0152
Ctju_fin	0.9908	0.0029	0.99116	0.001	0.5948	0.0194
Ctma_fin	0.9912	0.0017	0.874	0.0017	0.5736	0.0579
Ctop_fin	0.8662	0.01	0.9575	0.004	0.7698	0.0117
Ctpp_fin	0.9622	0.0088	0.998	0.0063	0.664	0.0170
Deco_fin	0.998	0.0021	0.999	0.002	0.5248	0.0621
Deme_fin	0.999	0.0009	0.9586	0.0009	0.6974	0.0262
Dela_fin	0.96	0.0009	0.916	0.0063	0.6740	0.0142
Ecue_fin	0.9218	0.0043	0.9823	0.0147	0.6494	0.0176
Fond_fin	0.9824	0.0111	0.9934	0.0037	0.7402	0.0144
Gip_fin	0.9934	0.0037	0.9899	0.0018	0.7428	0.0185
Plan_fin	0.99	0.0018	0.998	0.0038	0.7228	0.0169
Pres_fin	0.998	0.0038	0.9549	0.0012	0.4836	0.0763
Reca_fin	0.9548	0.0012	0.954	0.0022	0.7202	0.0195
Tjcr_fin	0.954	0.0021	0.9751	0.0055	0.6998	0.0151
Valo_fin	0.9752	0.0055	0.9972	0.0062	0.746	0.0175
Viv_fin	0.9972	0.0062	0.9456	0.0011	0.7198	0.0223
Nomina	0.9508	0.0011	0.9509	0.0091	0.6486	0.0136
Nom_pens	0.947	0.0081	0.947	0.0059	0.6538	0.0119
Recibo	0.8708	0.0088	0.860	0.0102	0.6166	0.0226

3.2.4 Logistic Regression model

Since this model can only handle two classification problems, we still train and test 20 labels one by one. We select the first 7000 samples to train the model.

In this model, we use GridSearchCV to search the best parameter, and the parameter to be tested are ‘C’, a smaller value which specifies a stronger regularization, and ‘solver’, the optimization algorithm parameter selection. We set the range of values for ‘solver’ is (‘lbfgs’ , ‘liblinear’) and the range of values for ‘C’ is (0.001, 0.01, 0.1, 1, 10). The table below shows the outcomes. Therefore, we get the best parameter ‘C’ is 0.001, and the best parameter ‘solver’ is ‘lbfgs’.

Table 3.4 search results for Logistic Regression model

label	‘C’	‘solver’	best score
ind_cco_fin_ult1	0.001	lbfgs	0.648714
ind_cno_fin_ult1	0.001	lbfgs	0.911142
ind_ctju_fin_ult1	0.001	lbfgs	0.991
ind_ctma_fin_ult1	0.001	lbfgs	0.989428
ind_ctop_fin_ult1	0.001	liblinear	0.857571
ind_ctpp_fin_ult1	0.001	lbfgs	0.956714
ind_deco_fin_ult1	0.001	lbfgs	0.998285
ind_dela_fin_ult1	0.001	lbfgs	0.955142
ind_ecue_fin_ult1	0.001	lbfgs	0.920285

ind_fond_fin_ult1	0.001	lbfgs	0.981571
ind_hip_fin_ult1	0.001	lbfgs	0.994428
ind_plan_fin_ult1	0.001	lbfgs	0.991428
ind_pres_fin_ult1	0.001	lbfgs	0.996999
ind_reca_fin_ult1	0.001	lbfgs	0.942285
ind_tjcr_fin_ult1	0.001	lbfgs	0.954428
ind_valo_fin_ult1	0.001	lbfgs	0.969
ind_viv_fin_ult1	0.001	lbfgs	0.994714
ind_nomina_ult1	0.001	lbfgs	0.941142
ind_nom_pens_ult1	0.001	lbfgs	0.935285
ind_recibo_ult1	0.001	lbfgs	0.865

3.2.5 KNN model

K-Nearest Neighbor Classifier (KNN) is more suitable for automatic classification of class domains with a relatively large sample size, but those with a relatively small sample size are more likely to cause misclassification. In order to compare the results with the Logistic Regression model, we still train and test 20 labels one by one. We select the first 7000 samples to train the model.

In our model, we use GridSearchCV to search the best parameter, and the parameters to be tested are 'n_neighbors': k value in KNN; 'weights': used to identify the weights of the neighboring samples of each sample. We set the range of values for 'weights' is ('uniform' , 'distance') and the range of values for 'n_neighbors' is from 1 to 5. The table below shows the outcomes. The best parameter 'n_neighbors' is 4, and the best parameter 'weights' is 'uniform'.

Table 3.5 search results for KNN model

label	'n_neighbors'	'weights'	best score
ind_cco_fin_ult1	3	uniform	0.615714
ind_cno_fin_ult1	4	uniform	0.908
ind_ctju_fin_ult1	2	uniform	0.991
ind_ctma_fin_ult1	4	uniform	0.989571
ind_ctop_fin_ult1	2	uniform	0.856
ind_ctpp_fin_ult1	4	uniform	0.956
ind_deco_fin_ult1	2	uniform	0.998285
ind_dela_fin_ult1	4	uniform	0.953714
ind_ecue_fin_ult1	4	uniform	0.915285
ind_fond_fin_ult1	4	uniform	0.981428
ind_hip_fin_ult1	2	uniform	0.994428
ind_plan_fin_ult1	4	uniform	0.991428
ind_pres_fin_ult1	4	uniform	0.996999
ind_reca_fin_ult1	4	uniform	0.939714
ind_tjcr_fin_ult1	4	uniform	0.953428
ind_valo_fin_ult1	4	uniform	0.968714
ind_viv_fin_ult1	4	uniform	0.994714
ind_nomina_ult1	4	uniform	0.939714
ind_nom_pens_ult1	4	uniform	0.933285
ind_recibo_ult1	4	uniform	0.860571

3.2.6 SVC model

Although the Support Vector Classification (SVC) can deal with nonlinear classification problems, the length of fit time will quadratically rise with the expanding of samples. Besides, the traditional SVC model does not support multi-classification (Jose et al. 2015). Hence, our model will train and test 20 labels one by one in order to analyze the difference of accuracy of each label. Due to the fit time problem, we select the first 7000 samples to search the best parameters.

3.2.6.1 SVC model with 'poly' kernel function

The first SVC model that we use is with 'poly' kernel function. Since we use GridSearchCV to search the best parameter for this model among a set of specified parameter values, we decide to search two parameters. Due to the fit time problem, we first study one parameter 'degree', which represents the degree of 'poly' kernel function. After finding the best value for 'degree', we then use this best value to research the other parameter 'C', which is the regularization parameter. We use 5-fold cross validation and the search results are in the following table.

3.2.6.2 SVC model with 'rbf' kernel function

The second SVC model that we use is with 'rbf' kernel function. The way to study this model is the same as the first SVC model. This time we decide to search another two parameters. One is 'gamma', which denotes the coefficient for 'rbf'. The other is the regularization parameter 'C'. We still use 5-fold cross validation and the search results are in the following table.

Table 3.6 search results for two SVC model and accuracy results for the best SVC model

label	model 1 (kernel='poly')				model 2 (kernel='rbf')				best model	
	'degree'		'C'		'gamma'		'C'		precision	recall
	best parameter	best score	best parameter	best score	best parameter	best score	best parameter	best score		
Current Account	1	0.648429	0.001	0.648429	0.001	0.650714	1	0.650714	0.65	1.00
Payroll Account	1	0.911143	0.001	0.911143	0.001	0.912000	1	0.912000	0.92	1.00
Junior Account	1	0.991	0.001	0.991	0.001	0.991	0.001	0.991	0.99	1.00
Mas Particular Account	1	0.989429	0.001	0.989429	0.001	0.989429	0.001	0.989429	0.99	1.00
Particular Account	1	0.860714	0.001	0.860714	0.001	0.861857	1	0.861857	0.87	1.00
Particular Plus Account	1	0.956714	0.001	0.956714	0.001	0.957000	1	0.957000	0.96	1.00
Short-term deposits	1	0.998286	0.001	0.998286	0.001	0.998286	0.001	0.998286	1.00	1.00
Long-term deposits	1	0.955143	0.001	0.955143	0.001	0.955143	0.001	0.955143	0.96	1.00
E-account	1	0.920286	0.001	0.920286	0.001	0.920857	1	0.920857	0.92	1.00
Funds	1	0.981571	0.001	0.981571	0.001	0.981571	0.001	0.981571	0.98	1.00
Mortgage	1	0.994429	0.001	0.994429	0.001	0.994429	0.001	0.994429	0.99	1.00
Pensions	1	0.991428	0.001	0.991428	0.001	0.991428	0.001	0.991428	0.99	1.00
Loans	1	0.996999	0.001	0.996999	0.001	0.996999	0.001	0.996999	1.00	1.00
Taxes	1	0.942285	0.001	0.942285	0.001	0.942285	0.001	0.942285	0.95	1.00
Credit Card	1	0.954428	0.001	0.954428	0.001	0.954428	0.001	0.954428	0.96	1.00
Securities	1	0.969	0.001	0.969	0.001	0.968857	0.001	0.969	0.97	1.00
Home Account	1	0.994714	0.001	0.994714	0.001	0.994714	0.001	0.994714	1.00	1.00
Payroll	1	0.941143	0.001	0.941143	0.001	0.941714	1	0.941714	0.94	1.00
Payroll Pensions	1	0.935285	0.001	0.935285	0.001	0.935857	1	0.935857	0.94	1.00
Direct Debit	1	0.865	0.001	0.865	1	0.865142	0.001	0.865	0.87	1.00

3.2.6.3 best SVC model

When comparing the search results of these two SVC model, we find that the best values for 'degree' and 'C' are stable among 20 labels in the first model, while the best values for parameters vary among 20 labels in the second model. Although

the best scores in the second model are slightly higher than those in the first model, the differences are tiny. Hence, the best SVC model that we choose is the model with 'kernel' to be 'poly', 'degree' to be 1 and 'C' to be 0.001.

After defining the parameters for the best SVC model, we randomly select 40000 training data and 20000 test data to examine the accuracy of this model by using precision and recall. The precision and recall results are also in the table above. Almost all precision values are larger than 0.8 except that the precision for label Current Account is only 0.65. Besides, we notice that recall results are larger than precision results for all labels. This means all values for actual positive class are successfully predicted as positive but a few values for actual negative class are predicted as positive.

3.2.7 MLP model

The Multi-layer Perceptron Classifier (MLP) can fit large amounts of samples in shorter time when compared with the SVC model. However, the accuracy of outcome of MLP model is not stable due to problems caused by backpropagation. In order to compare the results of MLP model with the SVC model, we still train and test 20 labels one by one. For the MLP model, we select the first 20000 samples to search the best parameters.

3.2.7.1 MLP model with 'logistic' activation function

The first MLP model that we use is with 'logistic' activation function. Since we still use GridSearchCV to search the best parameter for this model, we decide to search two parameters. Due to the fit time problem, we first study one parameter 'hidden_layer_sizes', which represents the number of neurons in the hidden layer. After finding the best value for 'hidden_layer_sizes', we then use this best value to research the other parameter 'alpha', which is L2 penalty parameter. We use 5-fold cross validation and the search results are in the table.

3.2.7.2 MLP model with 'tanh' activation function

The second MLP model that we use is with 'tanh' activation function. The way to study this model is the same as the first MLP model and those two parameters that we choose are also the same as the first MLP model. Again we use 5-fold cross validation and the search results are in the table.

Table 3.7 search results for two MLP model and accuracy results for the best MLP model

label	model 1 (activation='logistic')				model 2 (activation='tanh')				best model	
	'hidden_layer_sizes'		'alpha'		'hidden_layer_sizes'		'alpha'			
	best parameter	best score	best parameter	best score	best parameter	best score	best parameter	best score	precision	recall
Current Account	(100, 90)	0.65315	0.001	0.6529	(100,30)	0.65025	0.001	0.65145	0.65	1.00
Payroll Account	(100,)	0.914300	0.0001	0.914300	(100,)	0.914300	0.0001	0.914300	0.92	1.00
Junior Account	(100,)	0.98995	0.0001	0.98995	(100,)	0.98995	0.0001	0.98995	0.99	1.00
Mas Particular Account	(100,)	0.98945	0.0001	0.98945	(100,)	0.98945	0.0001	0.98945	0.99	1.00
Particular Account	(100,30)	0.863250	0.0001	0.863250	(100,30)	0.863250	0.0001	0.863250	0.87	1.00
Particular Plus Account	(100,)	0.9558	0.0001	0.9558	(100,)	0.9558	0.0001	0.9558	0.96	1.00
Short-term deposits	(100,)	0.998500	0.0001	0.998500	(100,)	0.998500	0.0001	0.998500	1.00	1.00
Long-term deposits	(100,)	0.95695	0.0001	0.95695	(100,)	0.95695	0.0001	0.95695	0.96	1.00
E-account	(100,)	0.920899	0.0001	0.920899	(100,)	0.920899	0.0001	0.920899	0.92	1.00
Funds	(100,)	0.98145	0.0001	0.98145	(100,)	0.98145	0.0001	0.98145	0.98	1.00
Mortgage	(100,)	0.994249	0.0001	0.994249	(100,)	0.994249	0.0001	0.994249	0.99	1.00
Pensions	(100,)	0.9906	0.0001	0.9906	(100,)	0.9906	0.0001	0.9906	0.99	1.00
Loans	(100,)	0.9978	0.0001	0.9978	(100,)	0.9978	0.0001	0.9978	1.00	1.00
Taxes	(100,)	0.94695	0.0001	0.94695	(100,)	0.94695	0.0001	0.94695	0.95	1.00
Credit Card	(100,)	0.95475	0.0001	0.95475	(100,)	0.95475	0.0001	0.95475	0.96	1.00
Securities	(100,)	0.971450	0.0001	0.971450	(100,)	0.971450	0.0001	0.971450	0.97	1.00

Home Account	(100,)	0.99595	0.0001	0.99595	(100,)	0.99595	0.0001	0.99595	1.00	1.00
Payroll	(100,)	0.94295	0.0001	0.94295	(100,)	0.94295	0.0001	0.94295	0.94	1.00
Payroll Pensions	(100,)	0.937300	0.0001	0.937300	(100,)	0.937300	0.0001	0.937300	0.94	1.00
Direct Debit	(100,)	0.872300	0.0001	0.872300	(100,)	0.872300	0.0001	0.872300	0.87	1.00

3.2.7.3 best MLP model

When comparing the search results of these two MLP model, we find that the best values for 'hidden_layer_sizes' and 'alpha' are much more stable in the second model after training several times. Hence, the best MLP model that we choose is the model with 'activation' to be 'tanh', 'hidden_layer_sizes' to be (100,) and 'alpha' to be 0.0001.

After defining the parameters for the best MLP model, we randomly select 40000 training data and 20000 test data to examine the accuracy of this model by using precision and recall. The precision and recall results are also in the table above. Although the precision and recall results are the same as those in SVC model, the length of fit time in MLP model is much smaller than that in SVC model.

3.2.8 Random Forest model

Random Forest is an ensemble model. Decision trees are the building blocks of the Random Forest model. Every individual tree spitting out predictions with the most votes is the final precision of Random Forest. In Random Forest, most of trees are relatively uncorrelated with one another. The combination is better than every single one. This is the embodiment of collective wisdom.

The advantages of Random Forest is that it ensures the individual errors of trees are minimized, and that the overall variance and error is reduced. The weakness of Random Forest is that it allows predictions of the trees need to be uncorrelated, which need us to extract the uncorrelated features.

3.2.8.1 Feature extract

In Random Forest, if two variables that are exactly the same (the extremes of the correlation) appear in the variable set, then the chance of this model vote for m label will increase. The constructed tree will become more similar, and the correlation between trees will become higher. Thus, the generalization ability of the random forest will become worse. This means these similar variables squeeze the chance for variables with other useful information to come into the model, which declines the performance of model.

Therefore, we need to remove feature 'segmento' and 'trprel_1mes' because the correlations between these two and other features are higher than positive or negative 0.7 in heatmap of Pearson correlation coefficient. Then, we train this forest in sub-dataset to evaluate the importance of features and sort this result to filter features. Finally, we use features whose importance higher than 0.15.

3.2.8.2 Best parameters

We use GridSearchCV to choose the best hyper-parameter. Futhermore, in order to speed up search process, we use big step to limit the approximate range first and use small step to find the best hyper-parameters. Finally, we choose n_estimators is 200, min_sample_split is 5, min_samples_leaf is 60, max_depth is 8.

3.2.8.3 Choice of model evaluation metric

Because this competition is a multi-label classification task. We use Hamming loss to evaluate the performance of model in Random Forest and XGBoost which originally support multi-label classification or use multiclass classifier to support multi-label classification. The equation of Hamming loss is as follows.

$$\mathrm{hloss}(h) = \frac{1}{p}\sum_{i=1}^{p} |h(x_i \Delta Y)|$$

3.2.8.4 Random Forest with absoulute accuracy

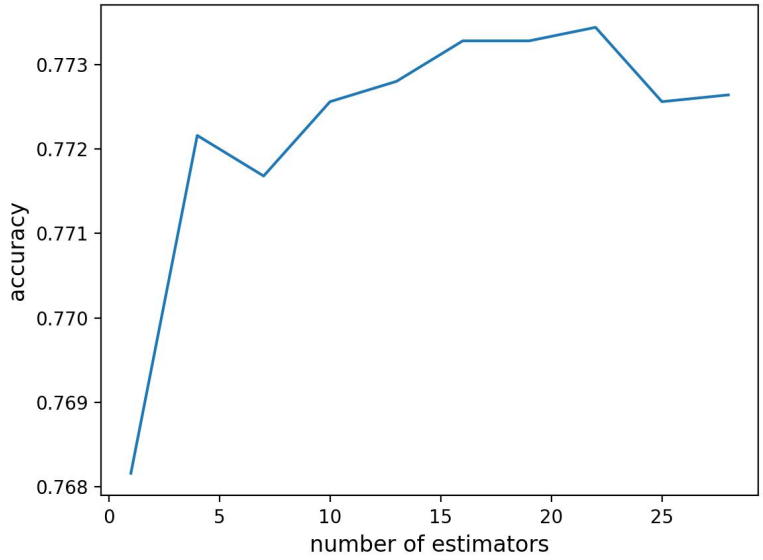


Figure 3.2 search results for the number of estimators and hamming loss in Random Forest model

3.2.9 XGBoosst model

XGBoost is a popular and optimized model in Macheicn Learning. It is an implementation of Gradient Boosting Tree. The advantage of XGBoost is that it requires less feature engineering. With regularized boosting, it can use data without scaling, normalizing and handling missing values. Because XGBoost only support multi-class classification, we use OneVsRestClassifier to implement multi-label classification.

3.2.9.1 Feature extract

Similar to Random Forest, sub-dataset help us to determine more useful features to avoid the curse of dimensionality.

3.2.9.2 Best parameters

Finally, we choose booster is gbtree, n_esimators is 16, learning_rate is 0.1, reg_lambda is 1, max_depth is 5, min_child_weight is 1.

3.2.9.3 XGBoost model with 'hamming loss function

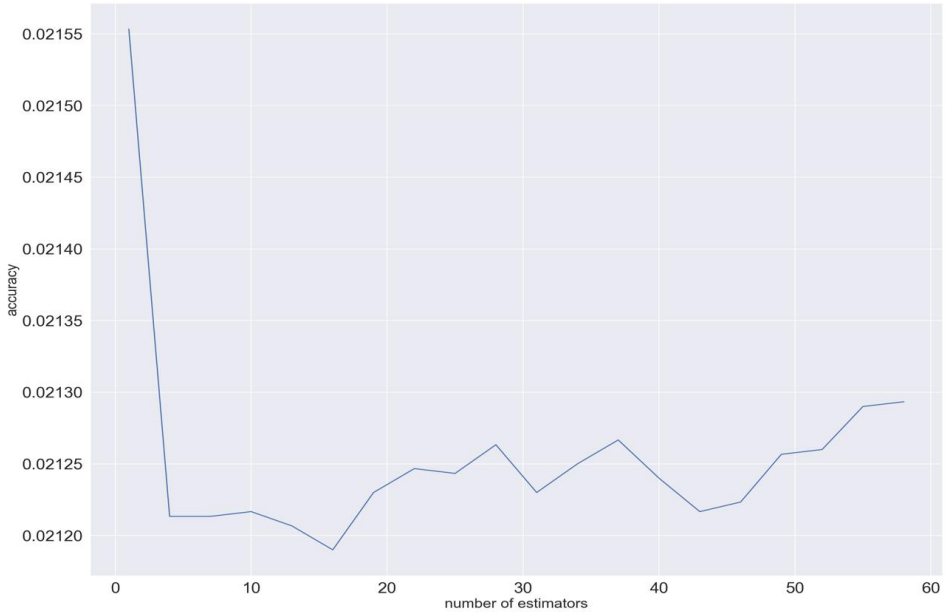


Figure 3.3 search results for the number of estimators and hamming loss in XGBoost model

4 Conclusion

Most models cannot support multi-label classification. In this report, these model transform the problem to binary classification problem and compute the accracy in every label respectively. They perform effectively in many labels, but in some label, their performance is not satisfactory. We consider the reason is sub-dataset, because of limited computing power, we have to sample the raw data-set. It affect the generalization of models.

Both Random forest and XGBoost(Grandient Boosting Decision Tree) are belong to ensemble learning. Because these two algorithm support multi-label-classification. We use absoulute accuracy to test these models. The models must predict all label correctly to gain scores. In the comparison between the two, XGBoost perform more effective than Random forest does. This is the same as our exception. Random Forest build many sub-classifier to vote, each sub-tree are independent, while XGBoost iteraters every child tree to approach optimal solution, each sub tree learn from previou tree. In addition, XGBoost training is much faster than Random forest because of parallel computing on features. The good performance of random forest is it almost never overfit in large data-set, we consider the independence of each sub-trees contributes this advantage.

Reference

GSK Ranjan, AK Verma, S Radhika 2019, *K-Nearest Neighbors and Grid Search CV Based Real Time Fault Monitoring System for Industries*, The International Conference for Convergence in Technology, India.

Jose M., Luis Gomez-Chova, Gustavo Camps-Valls 2015, *Multitask SVM learning for Remote Sensing Data Classification*, 2019 IEEE 5th International Society for Optical Engineering.

David Martin 2008, *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation*, Technical Report, Adelaide.

Jorocco 2017, *Sklearn main modules and basic usage methods*, accessed 25 July 2021, <https://blog.csdn.net/Jorocco/article/details/62892682?locationNum=8&fps=1>

Alan 2016, *Detailed Data Cleaning and Visualization*, accessed 21 July 2021, <https://www.kaggle.com/apryor6/detailed-cleaning-visualization-python>

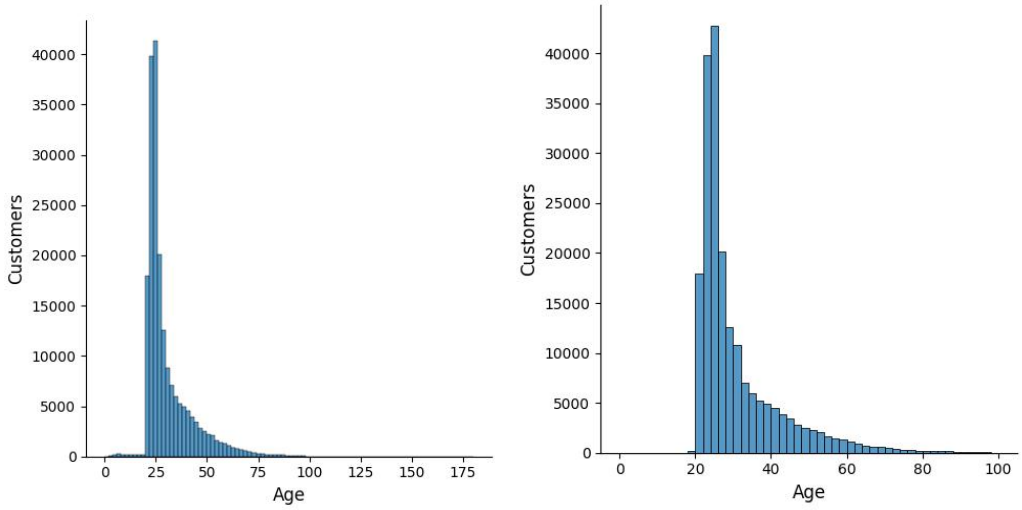
Natheer 2016, *Data Scientist at Ricoh Technology Center*, accessed 20 July 2021, <https://www.kaggle.com/alabsinatheer/comprehensive-exploration-and-visualization>

Scikit-learn 2020, Linear Regression, accessed 26 July 2021, <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html>

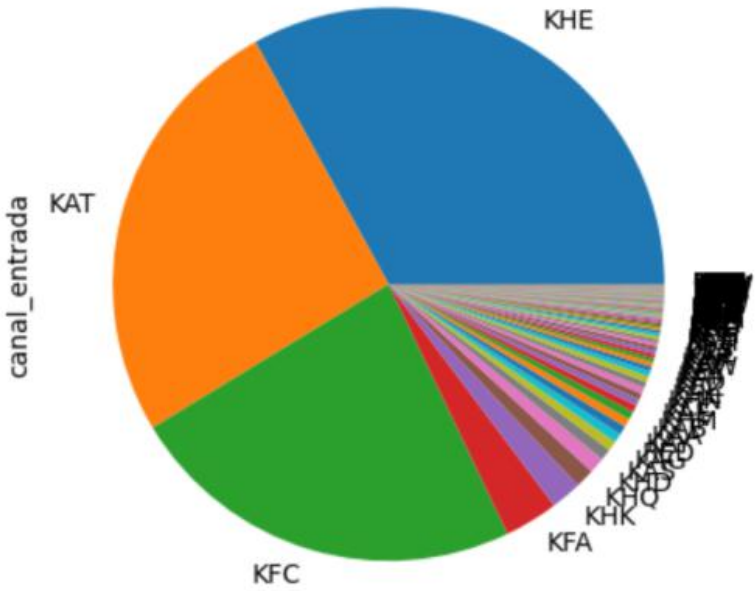
Scikit-learn 2020, Decision Trees, accessed 24 July 2021, <https://scikit-learn.org/stable/modules/tree.html>

Appendix

The difference between raw data of feature ‘age’ and processed data is as shown below.



The pie chart for feature ‘canal_entrada’.



The majority channels group by different products.