2021T1 COMP9417

Machine Learning and Data Mining

Group Project by Group 9417 HDer

TracHack Challenge 21.2

Xuanyu Jin (z5155473), Jiaying Liu (z5215876), Zhihan Liu (z5243912),

Yuewen Mao (z5210649), Shu Wang (z5211077)

I. Introduction

1.1 Background

Understanding "Does the customer ready for a device upgrade for the mobile phone" is essential so that the customer's need can be proactively provided and served. This report aims to predict, and to draw inference about the factors affecting the device upgrade by using machine learning method. In order to allow for more flexibility while maintaining interpretability, the first step will be to understand the dataset, then deal with the missing values, draw a comparison between the candidate models and interpret the result. Ultimately, select the best-performed model.

1.2 Data Description

There are 11 tables with more than 55,000 observations and around 70 variables, including the predicted value 'upgrade'. Merge has been applied by using the unique 'line_id' which contains in every table. Based on the descriptions on the model development dataset, different join methods applied, such as inner join and left join. With necessary manipulations, which will be discussed in the pre-processing session, the phone information table will be prepared in the separate spreadsheets. It is worthwhile to deal with this dataset separately as the missing value is extremely high, almost all the variables in that table contain around 40% of the missing value.

II. Data Cleaning

Initially, it is observed that in the 'redemptions', 'suspension', and 'lrp_enrollment' datasets, contained multiple records for each 'line_id'. This was due to the database inserting new updated data at different points in time. As such we selected to use only the latest record for each 'line_id'.   The reason why the latest data is selected is because it contains the most relevant information about this phone-user's next decision whether they will upgrade their phone or not, which is the aim of this project.

A similar issue appears in the ‘deactivations’ and ‘reactivations’ data set. However, instead of picking out the latest data only, the number of ‘deactivations’ and ‘reactivations’ appearances are counted and recorded as a new feature in these two data sets.

In the dataset ‘network_usage_domestic’, network usages are extremely large numbers since these are recorded in Kilobyte. Hence all Kilobyte data is converted into Gigabyte to avoid large calculations in further investigating. Additionally, there were six initial features ‘mms_in’, ‘mms_out’, ‘sms_in’, ‘sms_out’, ‘voice_min_in’ and ‘voice_min_out’ record the number of minutes the user called in and out, the number of texts the user sent and received and the number of minutes the voice messages are sent and received. We decided to aggregate them to develop three new features ‘total_mms’, ‘total_sms’ and ‘total_min_voice’.

To further extract useful information, features ‘suspension_start_date’ and ‘suspension_end_date’ in the dataset ‘suspensions’ are merged into a new feature ‘suspension_duration’, which represents the days of suspension this user had.

Additionally, all date features are converted into how many days this date is away from the ‘observed date’, which is a clearer representation showing how old or new this piece of information is. After this conversion, all date features are changed into integer type and normalization is applied on the date features to change the date values into a common scale.

After these processes have been applied, it is time to drop some relatively meaningless features in the dataset. Firstly, ‘plan_subtype’ is dropped because this feature contains only one valid data ‘SMART’ and the rest data are all null, which rarely provides any information. Same reason for dropping applies to the feature ‘gb_5g’, where feature ‘gb_5g’ contains very few non-zero values. Feature ‘revenue_type’ is dropped because it contains almost the same information as another feature ‘redemption type’ so it is redundant to have two similar features for future model development.

After comparing features ‘status’ and ‘lrp_enrolled’, it is obvious that ‘lrp_enrolled’ and ‘status’ are overlapped as both of these two features represent whether this customer is enrolled for loyalty reward programs or not – as a result, ‘status’ is dropped. The next two features in the ‘lrp_points’ dataset is ‘quantity’ which records the quantity of loyalty reward points the user received on the ‘update_date’. We decided to drop these two features since the feature ‘total_quantity’ holds the total number of loyalty reward points and provides a much more useful information.

III. Data Pre-processing

3.1 Missing Values Filling Methods

After data cleaning, this dataset still has one final issue which is the large amount of missing data in the dataset ‘phone_info’. Group decided to build two modes in parallel, one which completely ignores the data in the dataset ‘phone_info’ and another model which firstly uses a classifier to fill in empty values in ‘phone_info’ before building a model on top.

For the second model, there are several ways to fill in missing values. KNN and Tree Classifiers were ultimately chosen over mean and majority (mode) values as they provided a more realistic value for the missing entry that is based on other factors rather than just a generic “default value”.

Using KNN Classifier, the k-nearest-neighbours algorithm can be used to input the missing data by finding the closest neighbours to estimate the value. The reason why KNN Classifier is chosen is that the dataset is a high dimensional data, and KNN Classifier is specifically designed for this many features. On the other hand, KNN Classifier is also one of the most widely used imputation methods. In addition, KNN Regressor is used on continuous features instead of KNN Classifier to predict values.

For all the other categorical (discrete) features, a Decision Tree can be used. This is another popular machine learning algorithm where it aims to create a model to predict the class of a target variable based on some other input variables. It is a useful classifier to use in the project because apart from the continuous features where KNN Regressor is applied, the rest of features with missing values are all categorical features.

3.2 Missing Values Filling Models

3.2.1 Model Inputs Selection

After dropping some features with reasons mentioned in previous parts and rows with too many features missing, there are still 17 features that have missing values in them. The method of filling missing used here is to predict them by models built with features strongly correlated with them. Details of the 17 features and features used to predict them are listed in "Table 1" in the Appendix.

One thing that should be paid attention to is that since some of those features used to build models for filling missing values in other features have missing values themselves, the order of filling missing values is important. For example, 'deactivation_date' is used to predict 'suspension_start_date', so missing in it should be filled before filling in missing values in 'suspension_start_date'. The order of features in "Table 1" is the order of missing filling process.

3.2.2 Models Comparison

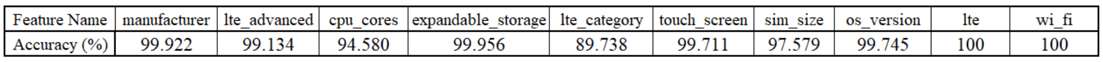
As mentioned before, missing values in categorical features will be predicted by KNN Classifier or Decision Tree Classifier, and missing values in continuous features will be predicted by KNN Regressor. However, which is the preferable choice between KNN Classifier and Decision Tree Classifier, and what is the optimal value of K, are still problems waiting to be solved.

For categorical features, accuracies of KNN Classifiers with K from 1 to 10 and Decision Tree Classifier are compared and recorded in "Table 2" in the Appendix. The table shows that Decision Tree Classifier performs always better than or at least as well as KNN Classifier no matter what the value of K is. Therefore, Decision Tree Classifier is used to fill missing values in all categorical features with missing.

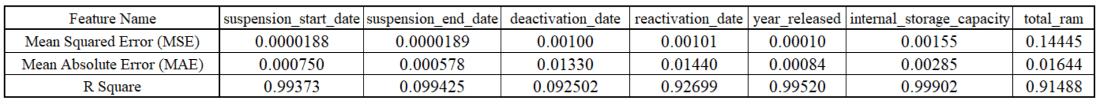
For continuous features, the mean squared error (MSE) of KNN Classifiers with K from 1 to 10 are compared and recorded in "Table 3" in the Appendix. The table shows: for 'deactivation_date', 'reactivation_date', 'suspension_start_date', and 'suspension_end_date', MSE decreases as the value of K increases for K is not larger than 10. Since the decrease of MSE is not obvious when K exceeds 7, 8 is chosen to avoid underfitting caused by large K. For the other 3 continuous features, ‘year_released’, ‘internal_storage_capacity’, and ‘total_ram’, 4 is chosen since MSE of ‘year_released’ and ‘internal_storage_capacity’ are smallest when K is 4 and 4 is neither too small to cause overfitting nor too large to cause underfitting.

3.2.3 Evaluation Metrics of Final Model

Performances of models used to fill missing values in those 17 features are shown in the two tables below. The first table, which contains accuracies of models on categorical features, shows that most of the models have prediction accuracies above 99%. The second contains mean squared errors (MSE), mean absolute errors (MAE), and R square scores (R2 score), which are useful metrics for evaluating the prediction accuracy and model performance in regression analysis, of models on continuous features, and it shows that all models have R2 scores above 0.9 and relatively low MSEs and MAEs, which means they are all good models.



Feature Name	manufacturer	lte_advanced	cpu_cores	expandable_storage	lte_category	touch_screen	sim_size	os_version	lte	wi_fi
Accuracy (%)	99.922	99.134	94.580	99.956	89.738	99.711	97.579	99.745	100	100



Feature Name	suspension_start_date	suspension_end_date	deactivation_date	reactivation_date	year_released	internal_storage_capacity	total_ram
Mean Squared Error (MSE)	0.0000188	0.0000189	0.00100	0.00101	0.00010	0.00155	0.14445
Mean Absolute Error (MAE)	0.000750	0.000578	0.01330	0.01440	0.00084	0.00285	0.01644
R Square	0.99373	0.099425	0.092502	0.92699	0.99520	0.99902	0.91488

IV. Methods for Model Selection

4.1 Model Selection Criteria

The choice of models depends on 2 major factors: the data and the performance of the feature itself.

First factor is the data. As discussed in previous sections, this problem is about binary classification, thus it is reasonable to select from models that perform well on binary classification. On the top of that, the training set itself contains 85 columns of features after pre-processing, where 84 of them are informative and 1 of them is for identification, meaning that the model of choice needs to be good at handling data with relatively high dimensionality.

Second factor is model performance. In order to measure that, the cross-validation method is used to test the performance of the model on the training data set. Due to the mechanism of the task itself, instant feedback of model performance on evaluation data is not available, CV is the main resort to measure model performance.

4.2 Available Models and Their Features

Based on reasons mentioned above, and given the limited time given to the group, 3 models were taken into account: Logistic Regression, Adaptive Boosting and Random Forest.

For Logistic Regression, it is one of the most commonly used models for binary classification. It makes no assumptions about distributions of classes in feature space, and takes relatively less time to train. However, since the training data is of relatively high dimensionality, there could be chances of overfitting. It also assumes that data are linearly separable, where that actual data may be not.

Random Forest is selected to deal with linearly non-separable cases, if there are any. It reduces the risks of overfitting, while deals with non-linear cases well. However, although it runs the testing pretty fast, it is very slow on training. If the data set is linearly separable, this method could also be inappropriate.

Adaptive Boosting is selected to reduce chances of overfitting if the Logistic Regression model performance is not ideal but data might be linearly separable. It could decrease Bias and Variance, and resolves the overfitting problem. However, given that both training and testing data size are large, the executing time could be much longer.

4.3 F1-score

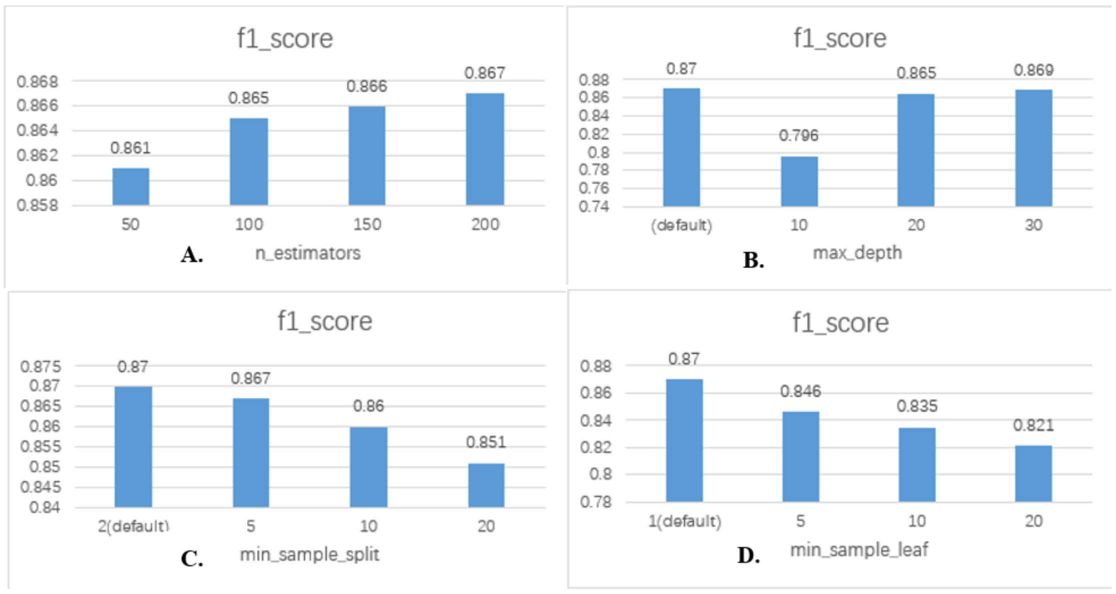
In this challenge, f1 score is the required metrics to evaluate the performance of models. It is the harmonic mean of precision and recall:

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

V. Results

In this part, 5-fold cross validation is applied in order to get a better estimate of out-of-sample accuracy.

For the random forest classifier, there are 5 different hyper-parameters tuned and compared, result of comparisons is shown in pictures below.



The first one is 'n_estimators' (A in picture above). It can be seen that increasing the number of estimators can improve the F1 score. Considering the running time, the model uses 'n_estimators' = 200.

Next one is the criterion, and the result is that using entropy gave an F1 score of 0.87, which is higher than using the gini index.

The other three are 'max_depth', 'min_sample_split', 'min_sample_leaf' (B, C, and D in picture above). It shows that changing these hyper-parameters from the default value will decrease the F1 score. The reason might be that these hyper-parameters are used to avoid over-fitting but this problem was not encountered in this challenge.

With the help of built-in ‘features_importances_’ attribute, it is found that ‘redemption_date_y’ is the most important feature in the random forest method, followed by ‘suspension_duration’. All the dummy features generated during one-hot encoding are less important than other features.

For random forest classifiers, the best F1 score is 0.87.

Tuning of Logistic Regression and AdaBoost models are similar.

For Logistic Regression, in order to achieve a highest F1 score, 3 hyper-parameters were changed from default: ‘solver’, ‘penalty’, and ‘tol’. Solver is the algorithm to use in the optimization problem, and is set to ‘liblinear’; penalty is Used to specify the norm used in the penalization, and is set to ‘l1’, standing for Lasso Regression, and ‘tol’, tolerance for stopping criteria, and is set to 0.00001. By this hyper-parameter setting, the F1 score from CV is increased from 0.5 to 0.75.

For Adaptive Boosting, ‘learning_rate’ and ‘n_estimators’ are adjusted since there is a trade-off between these 2 parameters. ‘learning_rate’ = 0.9 and ‘n_estimators’ = 100 is the adjusted optimal value for both parameters, and F1 score from CV for training becomes 0.84.

VI. Discussion

6.1 Comparison of Different Methods

6.2 Selection of Metrics

With the help of sklean.metrics.classification_report, it shows that precision, recall and F1 score are actually all the same for one class. These three metrics of class ‘no’, which means the customer will not upgrade their device, are higher than those of class ‘yes’. The former is 0.95, and the latter are 0.87. In this challenge, correct prediction on customers who are likely to upgrade their device should be focused. Therefore, the F1 score of class ‘yes’ is a proper metric to judge the performance of models.

6.3 Future Improvements

Due to the large amount of time spent on data pre-processing, the comparison between models is only based on F1 score. If there is more available time to work on this project, further comparison between model running time can be produced, which indicates the efficiency of the model.

VII. Conclusion

In conclusion, after data cleaning and using classifiers to fill in missing values, a forecasting model has been built to predict whether a customer is ready for a mobile phone upgrade or not. As a result of different models' analysis, Random Forest provides the best prediction with the highest F1 score 0.87. The final F1 score of the evaluation dataset is 0.822.

VIII. Appendix

Index	Data Type	Feature Name	Number of Missings	Related Feature	Correaltion
1	continuous	deactivation_date	10146	first_activation_date	0.42691
2	continuous	reactivation_date	15076	first_activation_date	0.42598
3	continuous	suspension_start_date	6549	deactivation_date	0.67451
				reactivation_date	0.68912
				first_activation_date	0.26667
4	continuous	suspension_end_date	6549	suspension_start_date	0.99877
				deactivation_date	0.67994
				reactivation_date	0.69398
				first_activation_date	0.27537
5	categorical	manufacturer	10256	gsma_model_name	-0.58936
6	categorical	year_released	10222	gsma_operating_system	0.56653
				manufacturer	-0.41991
				gsma_model_name	0.47350
7	categorical	internal_storage_capacity	10666	gsma_model_name	0.42136
				gsma_operating_system	0.46080
				manufacturer	-0.38433
8	categorical	lte_advanced	12824	gsma_device_type	0.58086
9	categorical	cpu_cores	10329	gsma_operating_system	-0.51343
				manufacturer	0.46142
				year_released	-0.73553
10	categorical	expandable_storage	10819	gsma_model_name	-0.78018
				gsma_operating_system	-0.94681
				manufacturer	0.84804
11	categorical	lte_category	13117	cpu_cores	0.74100

				year_released	-0.613618
12	categorical	touch_screen	10218	gsma_device_type	0.70198
13	categorical	sim_size	11579	gsma_operating_system	0.29601
				manufacturer	-0.30054
14	categorical	os_version	10419	gsma_model_name	0.56172
				gsma_operating_system	0.65010
				gsma_device_type	0.41579
15	continuous	total_ram	10434	cpu_cores	0.53874
				lte_category	0.71404
				internal_storage_capacity	0.58113
				year_released	-0.41795
				os_version	0.23080
16	continuous	lte	10240	gsma_device_type	0.32478
				HANDSET	-0.31288
				lte_advanced	0.39863
				touch_screen	0.20222
17	continuous	wi_fi	10252	gsma_device_type	0.22491
				HANDSET	-0.27079
				lte_advanced	0.27179
				touch_screen	0.26720
				lte	0.74022

Table 1: Features with Missing and Features Strongly Correlated to Them

Classifier	Feature Name	Decision Tree	K-Nearest Neighbors Classifier									
			1	2	3	4	5	6	7	8	9	10
Accuarcy (%)	manufacturer	99.922	99.911	99.878	99.833	99.778	99.811	99.778	99.756	99.733	99.722	99.689
	lte_advanced	99.134	99.134	99.134	99.134	99.134	99.134	99.134	99.134	99.134	99.134	99.134
	cpu_cores	94.580	92.403	93.947	92.381	93.913	92.370	92.359	93.025	93.003	93.014	93.014
	expandable_storage	99.956	99.956	99.911	99.888	99.822	99.857	99.845	99.845	99.833	99.833	99.789
	lte_category	89.738	89.305	62.095	86.861	89.582	89.638	89.638	89.594	89.594	89.693	89.693
	touch_screen	99.711	99.711	99.711	99.711	99.711	99.711	99.711	99.711	99.711	99.711	99.711
	sim_size	97.579	97.290	97.490	97.468	97.457	97.446	97.446	97.446	97.446	97.546	97.546
	os_version	99.745	99.711	99.522	99.567	99.40	99.367	99.311	99.234	99.223	99.167	99.045
	lte	100	100	100	100	100	100	100	100	100	100	100
	wi_fi	100	100	100	100	100	100	100	100	100	100	100

Table 2: Prediction Accuracies of Different Models on Categorical Features

Regressor	Feature Name	K-Nearest Neighbors Regressor									
		1	2	3	4	5	6	7	8	9	10
Mean Squared Error (MSE)	deactivation_date	0.00164	0.00126	0.00121	0.00107	0.00102	0.00101	0.00101	0.00100	0.00100	0.00100
	reactivation_date	0.00153	0.00124	0.00120	0.00112	0.00107	0.00106	0.00103	0.00101	0.00101	0.00101
	suspension_start_date	0.0000372	0.0000349	0.0000329	0.0000306	0.0000280	0.0000258	0.0000238	0.0000188	0.0000172	0.0000165
	suspension_end_date	0.0000364	0.0000341	0.0000321	0.0000298	0.0000271	0.0000248	0.0000229	0.0000189	0.0000174	0.0000164
	year_released	9.21813	0.00124	0.00012	0.00010	0.00013	0.00014	0.00016	0.00016	0.00016	0.00016
	internal_storage_capacity	0.00478	0.00741	0.00164	0.00155	0.00181	0.00195	0.00211	0.00226	0.00247	0.00269
	total_ram	0.37716	0.15648	0.14678	0.14445	0.14556	0.14684	0.14557	0.14837	0.14977	0.14424

Table 3: Prediction Mean Squared Errors (MSE) of Different Models on Continuous Features