FTEC4003 Data Mining for Fintech

Project Report

2020-2021 Term1

Team 04

HUANG Sida 1155124414 sdhuang@link.cuhk.edu.hk

ZHANG Xiran 1155124428 115512428@link.cuhk.edu.hk

FTEC4003 Project Task 1 Report

1. Introduction

The goal is to predict whether the client of an insurance company will buy its new transportation insurance. Given data from clients consisting of their basic information, we need to predict the class (“0” or “1”), which denotes the client’s response to the new transportation insurance. In this task, we mainly compare the performance of several common methods learned from the FTEC4003. The platform we use is python 3.8 & python 3.9.

The report is organized as follows. In section 2, we will analyze the training data and preprocess the data. In section 3, we will introduce and apply several common classification methods to this problem. And finally, the conclusions are shown in section 4.

2. Data Analysis and Preprocessing

2.1 Basic Information

The training data contains 342999 objects and 12 attributes, with 11 input attributes and 1 output attribute. The testing data contains 38110 objects and 11 input attributes. The input attributes contain the basic information of the customers, which are:

ID: Unique ID of clients (numeric)

Gender: Gender of clients (categorical: “Male”, “Female”)

Age: Age of clients (numeric)

Driving_License: whether the clients have a driving license (categorical: “0”, “1”)

Region_Code: Unique code for the region of the clients (numeric)

Previously_Insured: whether the clients have already transportation insurance (categorical: “0”, “1”)

Vehicle_Age: Age of the Vehicle (string)

Vehicle_Damage: whether the vehicle has been damaged (categorical: “No”, “Yes”)

Annual_Premium: The amount customer needs to pay as premium in the year (numeric)

Policy_Sales_Channel: Anonymised Code for the channel of outreaching to the customer (numeric)

Vintage: Number of Days that Customer has been associated with the company (numeric)

Output variable:
Response: whether the client is interested in the new transportation insurance (categorical: “0”, “1”)

As our goal is to compare the performance of different classification methods, rather than to get the best prediction results, we simply remove the ID attribute and retain all the data left as our training input.

2.2 Data Preprocessing

From the data set, we observe that there are several string attributes, including “Gender”, “Vechiel_Age”, and “Vechiel_Damage”, which cannot be fit into the training model directly. We consider using sklearn.feature_extraction.DictVectorizer, a tool to transform lists of feature-value mappings to vectors. In this task where the aimed attribute values are strings, this transformer will do a binary one-hot coding. As scikit-learn[1] introduces, “one boolean-valued feature is constructed for each of the possible string values that the feature can take on”. For instance, the attribute “Gender” that can take on the values “Female” and “Male” will become two attributes in the output, one signifying “Gender=Female”, the other “Gender=Male”.
Although there are some categorical attributes besides “Gender”, they only carry the values of “0” & “1”, which do not matter in the comparison for this task. Avoiding increasing complexity, we do not preprocess these attributes and directly put them into the training model.

3. Data Training

The purpose of this task is to compare the performance of different classification methods. In the following section, we will use f1-score as our evaluation indicator.

3.1 Classification Models

We use the following classifiers and tune their basic parameters if available in the process of training:

DT: DecisionTree	Basic parameter: max_depth
KNN: KNearestNeighbors	Basic parameter: n_neighbors, weights
GNB: GaussianNaiveBayes	
MNB: MultinomialNaiveBayes	

[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html#sklearn.feature_extraction.DictVectorizer

SVM: SupportVectorMachine Basic parameter: C

Bagging: Ensemble-Bagging Basic parameter: n_esitimators

3.2 Applying and Analysis

As we introduced above, we will apply different classifiers tuning the basic parameters respectively.

DecisionTree

```
clf = tree.DecisionTreeClassifier(max_depth = None)
```

Tune the max_depth = 10, 20, 30, 40, 50, 60, max.

max_depth	10	20	30	40
F1-score	0.00819672	0.11920758	0.22957198	0.27418438
max_depth	50	60	70	max
F1-score	0.28583935	0.29598536	0.29547158	0.29789929

We can observe that the f1-score will increase as the size of the decision tree (max_depth) increases. The highest f1-score is around 0.2980 with maximum depth.

KNearestNeighbors

```
clf = KNeighborsClassifier(n_neighbors=k, weights='distance', metric='minkowski', p=2)
```

Tune the n_neighbors = 1, 3, 5, 10; weights = ‘uniform’, ‘distance’.

n_neighbors	1	3	5	10	15
uniform	0.18695187	0.13159798	0.09149660	0.01054630	0.00766447
distance	0.18695187	0.14113694	0.10664050	0.04380694	0.02202369

We can observe that the f1-score will decrease as the number of neighbors increases, and the distance-weighted vote method will perform better than the majority vote method, especially for large n_neighbors. The highest f1-score is around 0.1870 with 1 neighbor.

GaussianNaiveBayes

```
gnb = GaussianNB()
```

f1-score = 0.38956087

MultinomialNaiveBayes

```
mnb = MultinomialNB()
```

f1-score = 0.26661178

SupportVectorMachine

```
clf = SVC(C = 1, kernel='rbf', gamma='auto')
```

Tune the C = 0.001, 0.01, 0.1, 1, 10, 100; kernel = 'rbf', 'linear', 'sigmoid', 'poly'.

	0.001	0.01	0.1	1	10	100
rbf	0	0	0	0	0	0
linear	0.01735083	0.05483745	0.05925639	0.06122449	0.06022003	0.06358935
sigmoid	0	0.07916823	0.08968809	0.09035088	0.09031127	0.09031127
poly	0	0	0	0	0	0

The reason why the f1-score of SVM is low might be that SVM has poor performance in dealing with high dimension problems such as this task.

Ensemble-Bagging

```
clf = ensemble.BaggingClassifier(DecisionTreeClassifier(max_depth = maxdepth),
                                 n_estimators=numBaseClassifiers)
```

As our goal is to get the performance of Bagging method under different values of the basic parameter n_estimators, we keep the internal classifier as DecisionTree with a max_depth of 50 to reduce the complexity.

n_estimators	2	3	4	5	10
F1-score	0.17438003	0.28076049	0.19258809	0.26555386	0.20217825
n_estimators	50	70			
F1-score	0.19930512	0.19700827			

We can observe that the f1-score will increase first and decrease afterward as the number of base estimators in the ensemble. Since the number of records is very large in this task, it is difficult to train the model with large n_estimators. The highest f1-score is around 0.2808 with 3 base estimators.

4. Conclusion

According to the results achieved in section 3, we can select the two best methods among all the classifiers in terms of the f1-scores for this task. The best one is GaussianNaiveBayes with the highest f1-score of 0.3896 approximately. The second one is DecisionTree under a parameter of max_depth = max, with the f1-score of 0.2980 approximately. The prediction results are stored in submission_1_GNB.csv and submission_1_DT.csv respectively.

FTEC4003 Project Task 2 Report

1. Introduction

Task 2 is a binary classification task. The motivation of this task is to predict whether the customers of a bank will stay or leave in the future. Given the information of the customer, the purpose of this task is to predict a class ('0' or '1'), which denotes the status of the customer. In this task, we use various classifiers and explore several methods to improve the performance of our model. The platform we use is python 3.8.

The report is organized as follows. In section 2, we'll analyse the training data and preprocess the data. In section 3, we'll introduce the techniques we used to derive our model. And finally, the conclusions are shown in section 4.

2. Data Analysis and Preprocessing

2.1 Basic Information

The training data contains 7500 objects and 14 attributes, with 13 input attributes and 1 output attribute. The testing data contains 2500 objects and 13 input attributes. The input attributes contain the basic information of the customers, which are:

- RowNumber: the number of rows
- CustomerId: the id of the customer in this bank.
- Surname: the surname of the customer
- CreditScore: personal credit score for an account.
- Geography: the location of the customer.
- Gender: the gender of the customer.
- Age: the age of the customer.
- Tenure: the valid time of the account.
- Balance: the amount of money in the account.
- NumOfProducts: the number of products the customer buys.
- HasCrCard: The number of Credit Card the customer owns.
- IsActiveMember: whether active in the recent period.
- EstimatedSalary: the estimated salary of the custome

And the output attribute is:

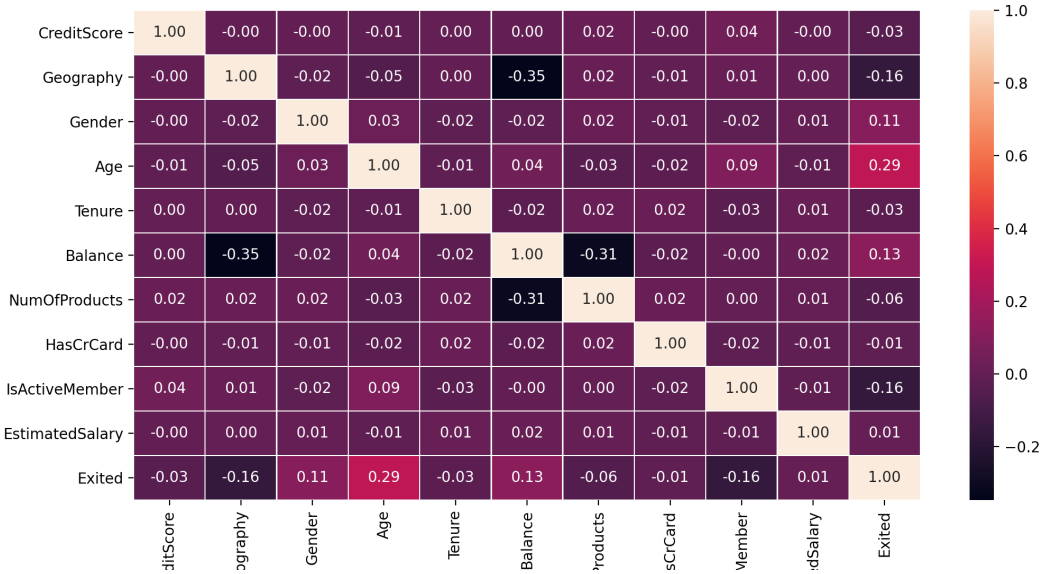
- Exited: whether this customer will leave in the future.

From the above information we can see that the input attributes contain some useless information when training. Therefore, we can remove "RowNumber", "CustomerId" and "Surname" as they have little influence on our prediction.

Within the 7500 training objects, 5941 of them are labeled as "Exited: False" and 1559 of them aer labeled as "Exited: True". Therefore, we see that the distribution of the class of the data is highly imbalanced, which will increase the difficulty of prediction and result in a low true-positive rate.

2.2 Feature selection

The pictrue below shows the correlation matrix of the training data:



We see that attribute "HasCrCard" and "EstimatedSalary" have low correlation to the output attribute : "Exited". However, after dropping these two columns, there is not an obvious improvement of the f1-score. Therefore, we'll not do feature selection at this stage.

2.3 One-hot Encoding

From the training data, we observe two categorical attributes: "Gender" and "Geography". "Gender" contains two classes: Male and Female. And "Geography" contains three classes: Spain, France and Germany. We consider using one-hot encoding to transfer these two attributes to integer.

As for "Gender", we convert Male to 0 and Female to 1. Furthermore, we remove the column of "Geography" and add three new columns: "Spain", "France", "Germany". The class is 1 for one of this attributes and 0 for the other two.

3. Data Training

The purpose of the training is to achieve higher f1-score. Therefore, in the following section, we'll use f1-score as our evaluation method.

3.1 Default Models

In the process of generating our model, we first use the default classifiers and compare their performance. Then, we select some of the best-performing classifiers to do further training.

We use the following classifiers:

- DT: Decision Tree
- KNN: K Nearest Neighbors
- GNB: Gaussian Naive Bayes

- MNB: Multinomial Naive Bayes
- LR: Logistic Regression
- SVM: Support Vector Machine
- MLP: Multi-layer Neural_Network
- Bag: Ensemble-Bagging
- RF: Ensemble-Random Forest
- GBst: Ensemble-Gradient Boosting
- LGBst: Light Gradient Boosting
- XGBst: XGBoost
- CatBst: CatBoost
- BBag: Balanced-Bagging
- BRF: Balanced-Random Forest
- RUSBst: Random under-sampling AdaBoost

The path of our code:

```
/Task-2_submission/default_models.py
```

And the results are:

Model	DT	KNN	GNB	MNB	LR	SVM	MLP	Bag
f1-score	0.512	0.126	0.115	0.300	0.077	0.222	0.239	0.572

Model	RF	GBst	LGBst	XGBst	CatBst	BBag	BRF	RUSBst
f1-score	0.606	0.609	0.619	0.590	0.619	0.587	0.567	0.559

From the results above we can see that there is a huge difference with respect to the performance of these models. The possible reason for this result is that some of the classifiers like KNN, GNB, LR, SVM have difficulty in handling high-dimensional training data.

We will drop the models whose f1-score is less than 0.5 and use the rest of models to do the future training.

3.2 Handling Imbalanced Data

From our perspective, the most important problem of this task is to handle the imbalanced data while training. And here are some ways to tackle this problem:

- Get more real training data
  - This does not work in our current task.
- Use balanced model
  - In the above subsection, we use three balanced model (BBag, BRF and RUSBst), which is designed for imbalanced data, to do the training, but the results are not satisfying. Therefore, we'll drop these 3 models in the future training.
- Adjust the class weights parameters
  - Some models contains a parameter called "class_weight", which allows users to manually adjust the weight for each class when training. For example, in this task, we have more "0"s than "1"s. If we set the weight of "0" to 1 and let the weight of "1" be larger than 1,

then the model will focusing more on the rows which are labeled as "1" while training, which will increase the true-positive rate.

- Resampling
  - The idea of resampling is to make the data balanced by manually creating or deleting data.
    - Over sampling allows us to increase the number of minority class.
    - Under sampling allows us to decrease the number of majority class.
    - We can also combine over sampling and under sampling.

We'll show the results of adjusting the class weight parameter and resampling in details in the following sub-subsections.

3.2.1 Class Weights Parameter

The path of our code:

```
/Task-2_submission/class_weight.py
```

As for fine-tuning the class_weight parameter, we use only three models (RF, LGBst, CatBst) and examine their performance using different weights.

The table below shows our results (in f1-score). The column attributes denote the weight x set for class 1 (class_weight={0:1,1:x}). The default model use class_weight={0:1,1:1}. Therefore, you can see that when x=1, the results are the same as the default models.

Weights x	**1**	**2**	**3**	**5**	**10**
RF	0.606	0.611	0.593	0.594	0.590
LGBst	0.619	0.624	0.606	0.577	0.525
CatBst	0.619	0.622	0.599	0.561	0.525

Weights x	**1.5**	**1.6**	**1.7**	**1.8**	**1.9**	**2.1**	**2.2**	**2.3**
RF	0.612	0.605	0.609	0.606	0.602	0.614	0.594	0.608
LGBst	0.620	0.621	0.615	0.628	0.617	0.621	0.611	0.620
CatBst	0.618	0.618	0.610	0.615	0.612	0.618	0.609	0.611

From the table above, we see that adjusting the class weights slightly improves the performance of the default model.

When using LGBst model and setting x = 1.8, we get the best score so far: 0.628.

3.2.2 Resampling

In this section, we show the results of using two python packages for data resampling.

- imbalanced-learn: A Python Package to Tackle the Curse of Imbalanced Datasets in Machine Learning.
  - This package provides sampler of over-sampling, under-sampling, combination of over and under-sampling, some balanced model (BBag, BRF, RUSBst we used in section 3.1).
  - The path of our code:

    ```
    /Task-2_submission/imbalanced.py
    ```

  - The table below is the result of runnig default model on resampled training data. The result of row "Without Sampling" is the same as the result in section 3.1. From row "RandomOverSampler" to row "SMOTENC" are over-samplers; From row "RandomUnderSampler" to "InstanceHardnessThreshold" are under-samplers. "SMOTEENN" and "SMOTETomek" are combinations of over- and under- samplers.

	DT	Bag	RF	GBst	LGBst	XGBst	CatBst
Without Sampling	0.512	0.572	0.606	0.609	0.619	0.590	0.619
RandomOverSampler	0.521	0.583	0.616	0.573	0.598	0.580	0.586
SMOTE	0.500	0.582	0.585	0.603	0.597	0.592	0.607
ADASYN	0.493	0.579	0.585	0.598	0.595	0.570	0.597
BorderlineSMOTE	0.504	0.580	0.601	0.606	0.612	0.586	0.608
SMOTENC	0.498	0.589	0.580	0.575	0.592	0.578	0.589
RandomUnderSampler	0.455	0.551	0.544	0.569	0.548	0.540	0.554
ClusterCentroids	0.336	0.350	0.360	0.356	0.353	0.350	0.352
NearMiss	0.392	0.387	0.388	0.390	0.390	0.390	0.387
EditedNearestNeighbours	0.484	0.588	0.599	0.610	0.589	0.560	0.596
RepeatedEditedNearestNeighbours	0.462	0.522	0.531	0.528	0.501	0.504	0.525
AllKNN	0.483	0.547	0.574	0.585	0.558	0.539	0.575
CondensedNearestNeighbour	0.496	0.561	0.605	0.609	0.595	0.578	0.598
OneSidedSelection	0.508	0.580	0.605	0.613	0.621	0.588	0.619
NeighbourhoodCleaningRule	0.487	0.577	0.609	0.622	0.595	0.570	0.595
InstanceHardnessThreshold	0.369	0.380	0.387	0.388	0.381	0.381	0.390
SMOTEENN	0.464	0.539	0.538	0.568	0.565	0.537	0.560
SMOTETomek	0.498	0.561	0.600	0.598	0.605	0.594	0.602

  - We see that we get worse results using most of the samplers, especially the under-samplers. One of the reasons may be that, if we use under-samplers, the training data will be so small with respect to the test data that there is not enough information for the model to predict the classes of the test data.

    - To conclude, we'll drop Decision Tree when do future training because of its low f1-score. "OneSidedSelection" and "NeighbourhoodCleaningRule" are worth trying. When we use "NeighbourhoodCleaningRule" and GBst model, we get the highest score in the table: 0.622.
- smote_variants: A common framework of 85 variants of SMOTE (Synthetic Minority Oversampling Technique).
    - This package only provides over-sampling techniques.
    - Note that this package does not support pandas.DataFrame. Therefore, we have to convert the data to numpy.ndarray to do the resampling and convert it back after finishing resampling.
    - The path of our code:

      ```
      /Task-2_submission/smotevariants.py
      ```

    - The table below shows the results of 5 over-samplers using 6 models

	Bag	RF	GBst	LGBst	XGBst	CatBoost
Without Sampling	0.572	0.606	0.609	0.619	0.590	0.619
polynom_fit_SMOTE	0.566	0.596	0.611	0.610	0.597	0.615
ProWSyn	0.577	0.605	0.628	0.612	0.603	0.610
SMOTE_IPF	0.573	0.606	0.620	0.618	0.601	0.612
Lee	0.576	0.601	0.619	0.606	0.591	0.617
SMOBD	0.592	0.589	0.617	0.614	0.586	0.624

    - It's clear to see that the GBst model improves a lot when using these over-samplers. When we use "ProWSyn" over-sampler and GBst, we get the highest score in the table: 0.628.

3.3 Cross validation and Fine-tuning

After choosing the best default model, class weight parameter and the sampler, we'll fine-tune the default model by adjusting its parameters and select the best one using cross-validation. As this process takes a long time to run, we'll not show our whole results. The sample code below shows how we fine-tune the GBst model using cross-validation.

```
...

x_train = train_dataframe.drop("Exited", axis=1)
y_train = train_dataframe["Exited"]

from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import GridSearchCV

r=999
gradient_boosting_params = {
```

```
    "loss":["deviance","exponential"],
    "learning_rate" : [0.001, 0.01, 0.1, 0.05],
    "n_estimators": [100,250,500],
    "criterion":['friedman_mse', 'mse', 'mae'],
    "min_samples_split": [2,5,10],
    "min_samples_leaf":[1,3,5],
    "max_depth": [3,5,10],
}

models = [
    GradientBoostingClassifier(random_state=r),
]
names = [
    "Ensemble-Gradient Boosting",
]
params = [
    gradient_boosting_params,
]
cross_val_result = {}
best_params = {}
for name, model, param in zip(names, models,params):
    clf = GridSearchCV(model, param_grid=param, cv =10, scoring = "f1", n_jobs = -1
,verbose = False)
    clf.fit(x_train,y_train)
    cross_val_result[name]=clf.best_score_    # Best f1-score
    best_params[name]=clf.best_params_        # Best parameters
...
```

3.4 Derivation of the Final Model

After all of the above process, we derive and examine the performance of our final model using the following techniques:

- Threshold moving

  Usually, the clasifier predict a input of the test data to be "0" by if the probability is greater than 0.5. Threshold moving allows us to examine different values of this probability and select the best threshold which has the highest score.

- Voting

  We select n best models and combine them together use majority voting, that is:

  ```
  final_model = []

  for i in range(0,2500):
    pred_sum = 0
    for model in models:
      if int(model[i]) == 1: pred_sum+=1
    if pred_sum/n < 0.5: final_model.append(0)
    else: final_model.append(1)
  ```

The path of our code:

```
/Task-2_submission/final_model.py
```

This will directly output submission_2.csv file.

4. Conclusion

In this project, we use various models and techniques aiming at achieving higher f1-score. However, the improvement of f1-score is not very satisfying. This may due to the lack of real training data. If this problem can be properly addressed, we believe the model will have higher performance when predicting the test data.

I am submitting the assignment for:

☐ an individual project or

☑ a group project on behalf of all members of the group. It is hereby confirmed that the submission is authorized by all members of the group, and all members of the group are required to sign this declaration.

I/We declare that: (i) the assignment here submitted is original except for source material explicitly acknowledged/all members of the group have read and checked that all parts of the piece of work, irrespective of whether they are contributed by individual members or all members as a group, here submitted are original except for source material explicitly acknowledged; (ii) the piece of work, or a part of the piece of work has not been submitted for more than one purpose (e.g. to satisfy the requirements in two different courses) without declaration; and (iii) the submitted soft copy with details listed in the <Submission Details> is identical to the hard copy(ies), if any, which has(have) been / is(are) going to be submitted. I/We also acknowledge that I am/we are aware of the University's policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the University website http://www.cuhk.edu.hk/policy/academichonesty/.

In the case of a group project, we are aware that all members of the group should be held responsible and liable to disciplinary actions, irrespective of whether he/she has signed the declaration and whether he/she has contributed, directly or indirectly, to the problematic contents.

I/We also understand that assignments without a properly signed declaration by the student concerned and in the case of a group project, by all members of the group concerned, will not be graded by the teacher(s).

黄思达；张曦然 30/11/2020

Signature(s) Date

Huang Sida; ZHANG Xiran 1155124414; 1155124428

Name(s) Student ID(s)

FTEC4003 Data Mining For Fintech

Course code Course title