ANSWER SHEET DAC 2021

DATA ANALYSIS COMPETITION 2021

TEAM NAME	Siev
TEAM ID	ID-21-0250
UNIVERSITY	Universitas Indonesia

Predicting Customer's Choice of Region Cluster with Random Forest Algorithm

As the startup industry grows rapidly, many startups have sprung up in any industry, including the property industry. The most common is the development of apps for buying houses. To optimize the user experience, app developers must make a good recommendation system. Using the data given, we are going to simulate making a recommendation system for houses based on the user's preferences.

Chapter I – Variable Selection

There are 24 variables presented in the data. The table below will show us the variables presented and the default data type.

Variable	Data Type (default)
time_date	object
site	int64
continent_id	int64
buyer_country	int64
buyer_region	int64
buyer_city	int64
distance	float64
buyer_id	int64
mobile	int64
package	int64
channel_id	int64
buying_date	object
dealing_date	object
adults	int64
children	int64
room	int64
destination_id	int64
destination_type	int64
regency_continent	int64
regency_country	int64
regency_market	int64
dealing	int64
cnt	int64
regency_cluster	int64

Table 1. Default Variables

Since the recommendation system will be used continuously in the future (not time-weighted), all the variables related to date and time will be dropped (‘time_date’, ‘buying_date’, and ‘dealing_date’). It will be shown later that there are many missing values in ‘distance’. These missing values can’t be replaced (because replacing them with zeros mean that the distance is zero that can lead to misinterpretation) and erasing the rows with missing values will just reduce the number of samples which lead to less prediction accuracy, so with those considerations, the ‘ distance’ variable will also be dropped. One last variable that will be dropped is the ‘cnt’ because it’s not interpretable enough to be used.

Then, using the SelectKBest method, 10 best variables will be chosen, that is (‘site’, ‘buyer_country’, ‘buyer_region’, ‘buyer_city’, ‘buyer_id’, ‘package’, ‘ destination_id’, ‘destinatination_type’, ‘regency_country’, ‘regency_market’). Observe that buyers in the same city should be in the same region and country. This seems to be true in the data with some inconsistencies. With this in mind, we decide to drop the ‘buyer_country’ and ‘buyer_region’ variables to reduce the inconsistencies and to reduce the number of variables selected. The table below will show the variables selected and the formatted data type.

Variable	Data Type (formatted)
site	category
buyer_city	category
buyer_id	category
package	category
destination_id	category
destination_type	category
regency_country	category
regency_market	category
regency_cluster	category

Table 2. Selected Variables

Chapter II – Methodology

2.1 Data Preprocessing

As mentioned in Chapter I, in choosing the variables we use the SelectKBest method. SelectKBest method is one of the variable/feature selection methods that select the top k-variables, in this case, k=10. The SelectKBest class just scores the features using a function (in this case we use 'chi2/2') and then "removes all but the k highest scoring features". SelectKBest will compute the statistics between each variable/feature of X and y ('regency_cluster'). If the result is small, it means the variable/feature is independent of y. A large value means the variable/feature is significantly related to y, meaning that it provides important information. After determining the variables selected, we just dropped all columns that contain the unselected variables.

2.2 Modelling

With the given dataset, we will be using the Random Forest method to model this problem. Before we discuss the Random Forest method further, we will first discuss the Decision Tree method. A Decision Tree is a supervised learning algorithm that's used for classification problems. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

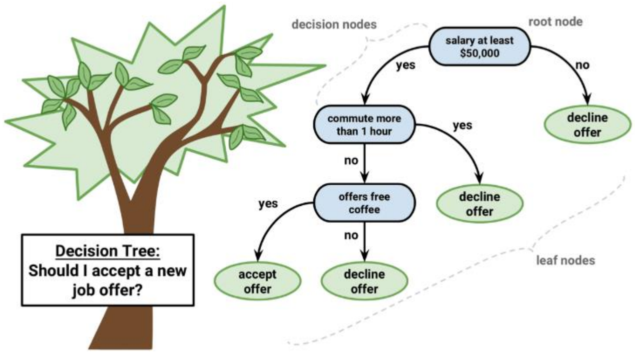


Figure 1. Decision Tree (https://dataaspirant.com/how-decision-tree-algorithm-works/)

Random Forest is an ensemble of decision tree algorithms. Each tree in the forest predicts/give their own classification and the result chosen is the prediction that wins the majority vote. This method is chosen because it's unbiased and stable. It's also flexible, easy to use, and able to give excellent results without a lot of hyperparameter tuning.

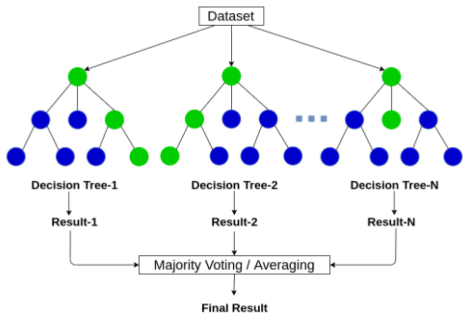


Figure 2. Random Forest (https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/)

2.3 Data Validation

The performance of the model will be assessed using *repeated stratified k-fold cross-validation*. Using this method, the dataset will be broken into *k* number of folds randomly (in this instance, *k* = 10) where one fold will be the test set and the rest will be the dataset used for training and it will be repeated *k* times. In each fold, the evaluation score will be retained, which we will be using the f1 score. The final score is the average of all evaluation scores.

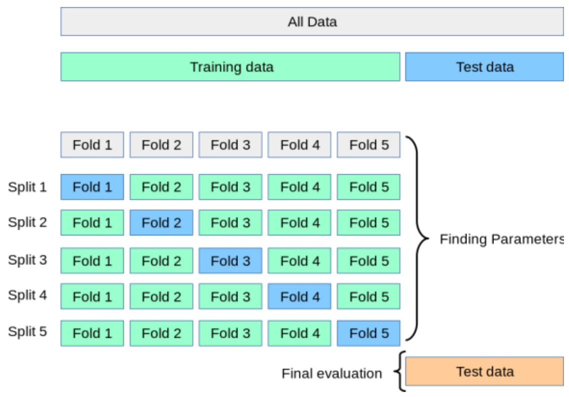


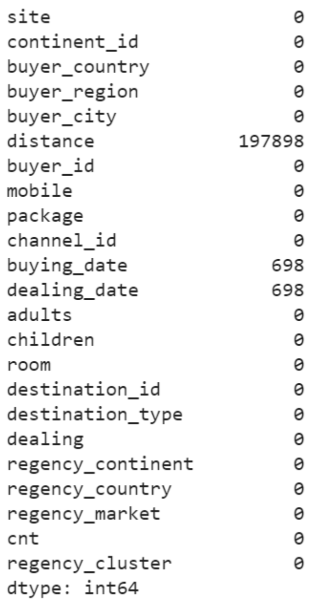
Figure 3. 5-fold cross-validation (https://www.kdnuggets.com/2020/01/data-validation-machine-learning.html)

This method is chosen because it's easy to understand, less biased than the normal train/split test, and doesn't require the training data to give up a certain portion of the set. We'll also be using the f1 score as the evaluation score because we have imbalanced class distribution. For the same reason, we will use stratification random sampling when splitting the dataset into *k* folds/groups and we will repeat the process for *n* times (*n*=3) to ensure the result given, hence it's called *repeated stratified k-fold cross-validation*.

Chapter III – Data Exploration & Analysis

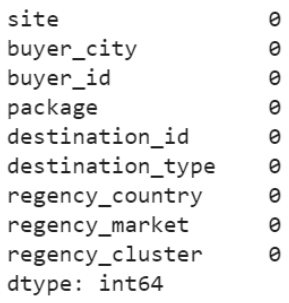
3.1 Missing Values

There are several missing values in ‘distance’, ‘buying_date’, and ‘dealing_date’ variables as shown in figure 4. But since we only use the variables that have been selected before, all variables that have missing values are dropped so that we don’t have any variables with missing values anymore.



```
site                       0
continent_id               0
buyer_country              0
buyer_region               0
buyer_city                 0
distance              197898
buyer_id                   0
mobile                     0
package                    0
channel_id                 0
buying_date              698
dealing_date             698
adults                     0
children                   0
room                       0
destination_id             0
destination_type           0
dealing                    0
regency_continent          0
regency_country            0
regency_market             0
cnt                        0
regency_cluster            0
dtype: int64
```

Figure 4a. Missing Values (before)



```
site                0
buyer_city          0
buyer_id            0
package             0
destination_id      0
destination_type    0
regency_country     0
regency_market      0
regency_cluster     0
dtype: int64
```

Figure 4b. Missing Values (after)

3.2 Visualization

Here are some visualizations of the ‘train’ dataset and their interpretations:

- **Label Class Imbalance**

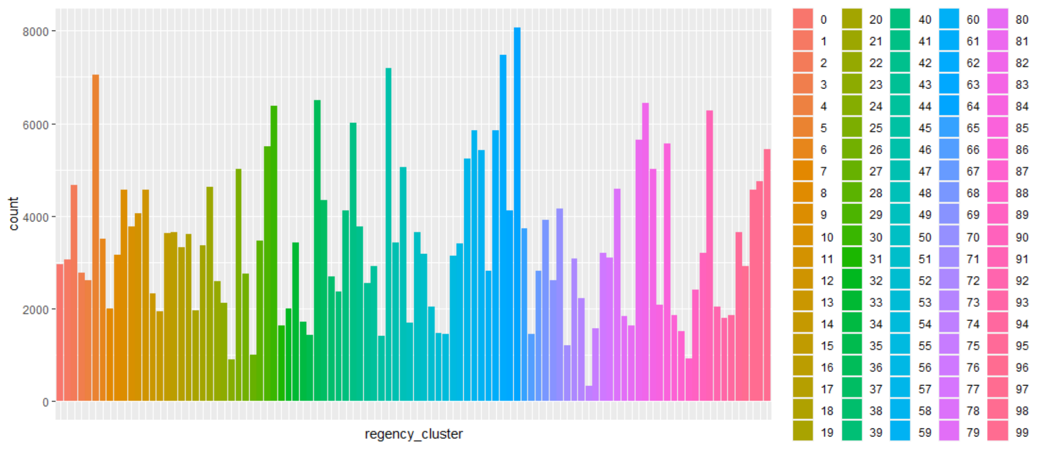


Figure 5. Label Class Imbalance

It is shown from the figure above that the label class distribution is imbalanced. That is, we need to use stratified random sampling when splitting our train and test dataset to get optimal results.

- **Successful Dealings**

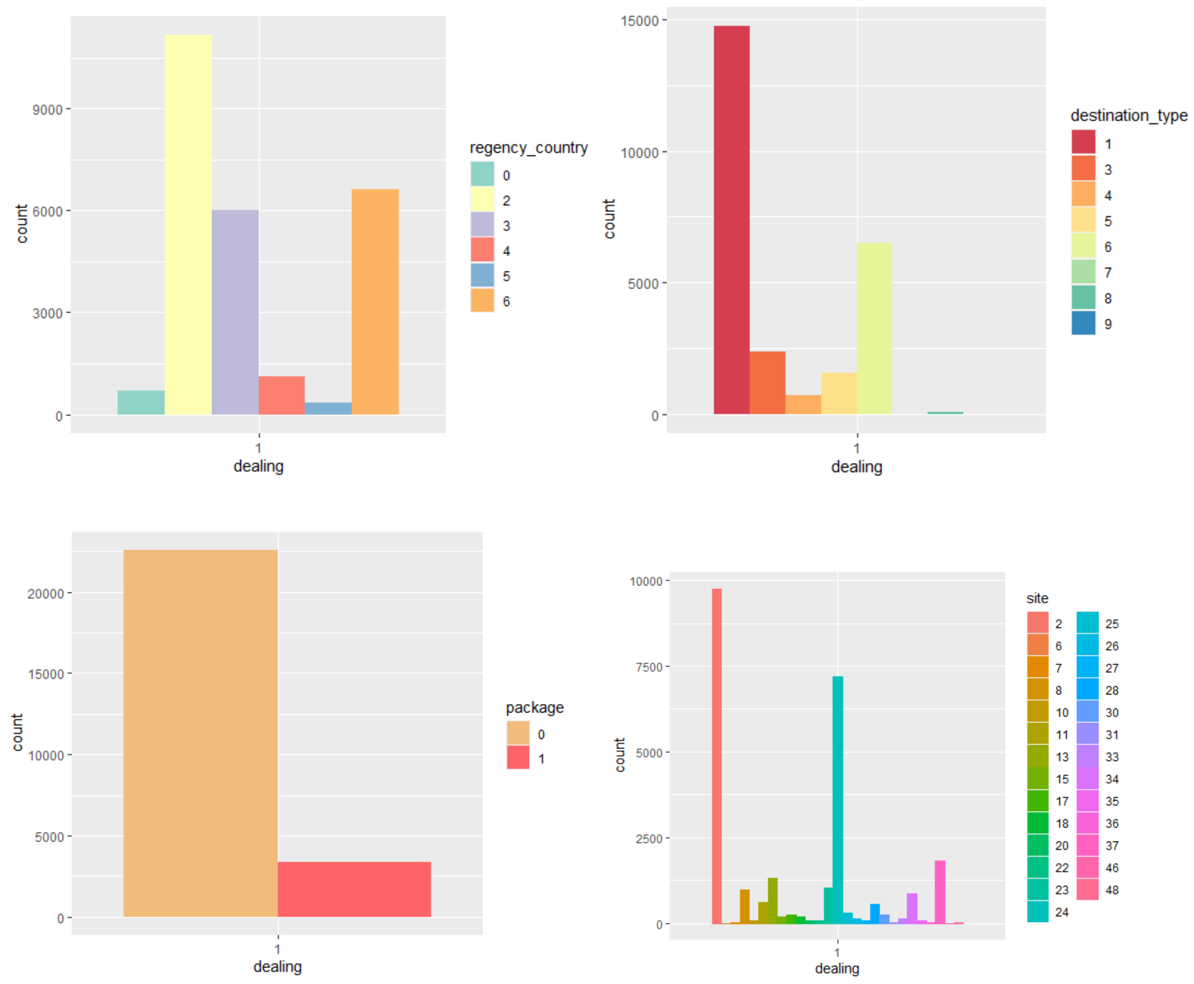


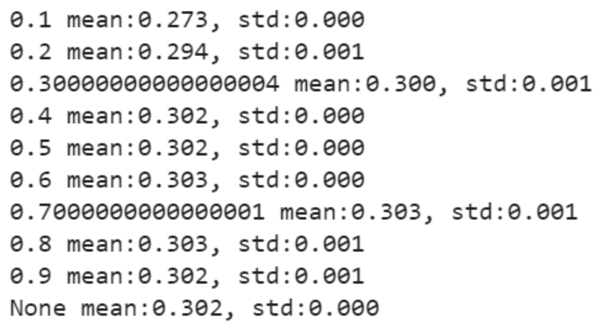
Figure 6. Successful Dealings (regency country, destination type, package, site)

The plots above show us the distribution of contributions in each variable to successful dealings. From the plots, we know that successful dealings will most likely happen from sites 2 and 24, without package (which means package doesn’t really affect on dealings), in regency country 2 and in the cluster with destination type 1.

3.3 Tuning Hyperparameters

According to the parsimony principle, our goal is to make the simplest yet accurate model. This also will help us to reduce the computational time needed. To achieve this, we'll need to tune the hyperparameters in our model to make it simpler with the same or better accuracy score than the default model.

- **Number of Samples in Each Decision Tree (‘*max_samples*’)**

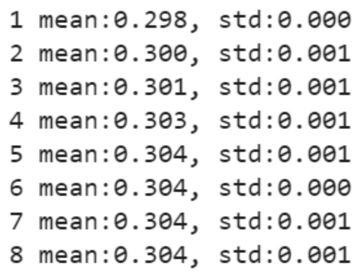


```
0.1 mean:0.273, std:0.000
0.2 mean:0.294, std:0.001
0.30000000000000004 mean:0.300, std:0.001
0.4 mean:0.302, std:0.000
0.5 mean:0.302, std:0.000
0.6 mean:0.303, std:0.000
0.7000000000000001 mean:0.303, std:0.001
0.8 mean:0.303, std:0.001
0.9 mean:0.302, std:0.001
None mean:0.302, std:0.000
```

Figure 7. max_samples

We computed the f1 score for all models with the possible values of sample sizes between 0.1 and 1. It's shown that the model that uses the 0.4 sample size in a single decision tree gives approximately the same accuracy as the model that uses all the samples in a single decision tree. Based on our parsimony principle, we will choose the simplest/smallest value, that is 0.4.

- **Number of Features Randomly Sampled at Each Split Point (‘*max_features*’)**

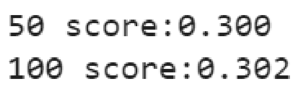


```
1 mean:0.298, std:0.000
2 mean:0.300, std:0.001
3 mean:0.301, std:0.001
4 mean:0.303, std:0.001
5 mean:0.304, std:0.001
6 mean:0.304, std:0.000
7 mean:0.304, std:0.001
8 mean:0.304, std:0.001
```

Figure 8.max_features

We computed the f1 score for all models with all possible values of features. We can see that the score stabilizes at 5 so we will use 5 as our number of features.

- **Number of Trees in The Forest (‘*n_estimators*’)**



```
50 score:0.300
100 score:0.302
```

Figure 9. n_estimators

Observe that the model with 50 trees that's not significantly different as the model with 100 trees and since we are not able to compute the model using more trees we will use 50 trees in our model.

3.4 Validating The Model

After the hyperparameter tuning, using the *repeated stratified k-fold cross-validation* with $k = 10$ and repeated 3 times, we get that the expected value of the model's f1 score is 0.313 with a standard deviation of 0.003.

Chapter IV – Conclusion and Suggestions

4.1 Conclusion

- We have successfully made the prediction model for the customer's choice of region cluster with a random forest algorithm with a score of 0.313. The score is not too high, but considering that the problem has 100 different label classes, this result is very acceptable. Customers are most likely to buy houses in regency country 2 with destination type 1.
- Sites that contributes the most for the dealing of house sales are site 2 and 24.
- The package is an important variable to predict the customer's preference of regency cluster but it doesn't really affect the house sales.

4.2 Suggestions

Suggestions for the business in the industry:

- Re-evaluate the package made so that package may really affect customers more to buy the house instead of just clicking.
- For other selling sites other than 2 and 24 should increase their advertisement and re-evaluate their business strategy so they can generate more deals in the future.

Suggestions for the data collection team:

- Change the data representation for places such as continent, city, region, and others. Instead of just using numbers to represent them, it's better if we use the real name of the place such as Asia, America, etc for the continent and so on. The other solution is to give further explanation about the numbers such as '1: USA, 2: Indonesia, 3: India' for the country and others. This will really help in pre-processing the data.

- Most of the data types are not correct and there are still a lot of missing values.
- Information about the variables is confusing. It will be better to give further examples of what the variables represent.
- A reduction in the number of region clusters may be helpful in improving the model's score. But this suggestion is not to be taken without further considerations.

ATTACHMENT

Python

```
#importing the data and initial data pre-processing
import pandas as pd
from google.colab import drive

drive.mount('/content/drive')
train=pd.read_csv("/content/drive/MyDrive/DAS/Train.csv").iloc[:,2:]
test=pd.read_csv("/content/drive/MyDrive/DAS/Test.csv").iloc[:,3:]

train = train.drop(columns=['distance','buying_date','dealing_date','cnt'])
train = train.astype('category')

#variable selection
X = train.drop(columns=['regency_cluster'])
y = train['regency_cluster']

from sklearn.feature_selection import SelectKBest, chi2
X_new = SelectKBest(chi2, k=10).fit_transform(X,y)
X_new[0:2]

#dropping the unselected variables
drop=['continent_id','buyer_country','buyer_region','mobile','channel_id','adults','childre
n','room','dealing','regency_continent']

train=train.drop(columns=drop)

X = train.drop(columns=['regency_cluster'])
y = train['regency_cluster']

#modelling and validation
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
```

```
model = RandomForestClassifier(max_samples=0.4,max_features=5,n_estimators=50)
cv = RepeatedStratifiedKFold(n_splits=10,n_repeats=3, random_state=2056)
n_scores = cross_val_score(model, X, y, cv=cv,scoring='f1_micro', error_score='raise')
print('f1 score: %.3f (%.3f)' % (np.mean(n_scores), np.std(n_scores)))

#data pre-processing for the test dataset
drop1=['continent_id','buyer_country','buyer_region','distance','mobile','buying_date','de
aling_date','channel_id','adults','children','room','regency_continent']
test=test.drop(columns=drop1).astype('category')

#model fitting and predicting the test dataset
model.fit(X,y)
pred = model.predict(test)
answer = test
answer['regency_cluster'] = pred

#finding the best number of samples
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2, stratify=y,random_
state= 2056)
for i in np.arange(0.1,1.1,0.1):
  arr=[]
  if i==1.0:
    i=None
  for j in range(0,5):
    model = RandomForestClassifier(max_samples=i)
    model.fit(X_train, y_train)
    yhat = model.predict(X_test)
    # evaluate predictions
    arr.append(f1_score(y_test, yhat,average='micro'))
  print(i,"mean:{0:.3f}, std:{1:.3f}".format(np.mean(arr), np.std(arr)))
```

```
#finding the best number of trees
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2, stratify=y,random_
state= 2056)
for i in [50,100]:
  model = RandomForestClassifier(max_samples=0.4,n_estimators=i)
  model.fit(X_train, y_train)
  yhat = model.predict(X_test)
  f1=f1_score(y_test, yhat,average='micro')
  print(i,"score:{0:.3f}".format(f1))

#finding the best number of features
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2, stratify=y,random_
state= 2056)
for i in range(1,len(X.columns)+1):
  arr=[]
  for j in range(0,5):
    model = RandomForestClassifier(max_samples=0.4,max_features=i,n_estimators=50
)
    model.fit(X_train, y_train)
    yhat = model.predict(X_test)
    # evaluate predictions
    arr.append(f1_score(y_test, yhat,average='micro'))
  print(i,"mean:{0:.3f}, std:{1:.3f}".format(np.mean(arr), np.std(arr)
```

R

```
#data visualization
drop<-c('X1','id','time_date','buying_date','dealing_date','cnt')
train_new=Train[,!(names(Train) %in% drop)]

str(train_new)
col<-colnames(train_new)
train_new[col]<-lapply(train_new[col], as.factor)

library(tidyverse)
library(wesanderson)

train_new %>%
  ggplot(aes(regency_cluster,fill=regency_cluster))+
  geom_bar()+
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

train_new %>%
  filter(dealing==1)%>%
  ggplot(aes(dealing,fill=site,color=site))+
  geom_bar(position='dodge')+
  scale_fill_discrete(name='site')
  xlab('dealing')

train_new %>%
  filter(dealing==1)%>%
  ggplot(aes(dealing,fill=package))+
  geom_bar(position='dodge')+
  scale_fill_manual(name="package",values=wes_palette("GrandBudapest1"))

train_new %>%
  filter(dealing==1)%>%
  ggplot(aes(dealing,fill=destination_type))+
  geom_bar(position='dodge')+
  scale_fill_brewer(palette = "Spectral")
```

```
train_new %>%
  filter(dealing==1)%>%
  ggplot(aes(dealing,fill=regency_country))+
  geom_bar(position='dodge')+
  scale_fill_brewer(palette = "Set3")
```

DATA ANALYSIS COMPETITION 2021

Himpunan Mahasiswa Statistika ITS (HIMASTA-ITS)

Gedung H Lantai III, Jl. Arief Rahman Hakim
Kampus ITS Sukolilo Surabaya
Email: eventhimastaits@gmail.com
Contact Person: Catur (085155430660)
Wanda (081327522030)