ANSWER SHEET DAC 2021

DATA ANALYSIS COMPETITION 2021

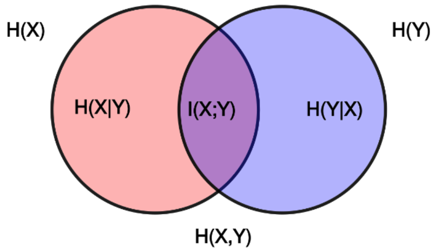
TEAM NAME Hmm Okay

TEAM ID ID-21-0114

UNIVERSITY Universitas Gadjah Mada

Chapter I

This dataset consists of many variables as written in the question sheet. To make predictions, we must choose which variables have a strong relevance to the target variable. The method we use to see the relevance between the variables used for prediction and the target variable is mutual information.

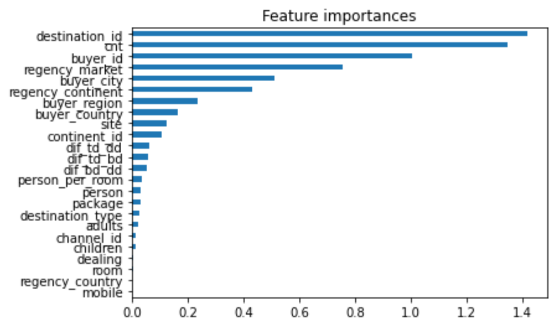


Mutual Information (MI) is a selection method that shows how much information whether a term contributes to making a right or wrong classification decision. X is input variable and Y is output or target variable. Mutual information can be calculated by marginal entropies H(X) H(Y)), conditional entropy H(X|Y) H(Y|X) and joint entropy H (X, Y). Mutual information can be expressed as:

$I (X, Y) = H(X) – H(X|Y)$
$I (X, Y) = H(Y) – H(Y|X)$
$I (X, Y) = H(X) + H(Y) – H (X, Y)$
$I (X, Y) = H (X, Y) – H(X|Y) – H(Y|X)$

In Python, we can used mutual_info_classif from sklearn.feature_selection here is the result :

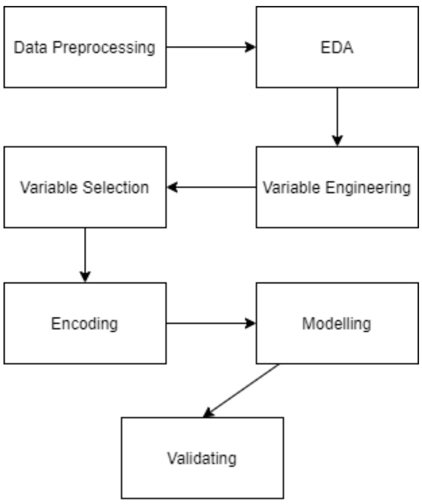
```
Text(0.5, 1.0, 'Feature importances')
```



Based on the table we use variables destination_id, buyer_id, regency_market, buyer_city, regency_continent. We do not use cnt because we did not find these variables in the test.csv.

dif_td_dd, dif_td_bd, dif_bd_dd, person, person_per_room are variable result from "variables enginering" which will be explained in the next chapter.

Chapter II



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)

The chart above shows the process of making predictions. We will discuss them one by one.

2.1. Data Preprocessing

A. Handling Missing Value

We found the missing values in train.csv and test.csv, we found the missing values in three columns: distance, buying_date, and dealing_date. In distance column we found we found more than 46% in test.csv and more than 57% in train.csv. In buying_date and dealing_date column we found 0.2% missing value in train.csv and 0% in test.csv. **We remove the distance column and delete the rows that contain missing values in this step.**

B. Swapping regency_continent and regency_country in train.csv

We swapped it out because regency_continent and regency_country have so many unique values. In reality there are only 6 or 7 unique values.

C. Handling Timeseries Columns

We create dif_tb_dd, dif_tb_bd, dif_bd_dd variables. dif_tb_dd is days difference between time_date and dealing_date, dif_tb_bd is days difference between time_date and buying_date, dif_bd_dd is days difference between buying_date and dealing_date.

2.2. EDA

We did EDA to find insights, more details will be explained in the next chapter.

2.3. Variable Engineering

We create person and person_per_room variable. Person is person is the sum of adults and children and person_per_room is the ratio of the number of persons to the number of rooms.

2.4. Variable Selection

As explained in chapter 1, we use mutual information to choose which variables are relevant to the target variable (regency_cluster). We use destination_id, buyer_id, regency_market, buyer_city, regency_continent variables for prediction. We do not use cnt because we did not find these variables in the test.csv.

2.5. Encoding

A. Target Encoding by Mean in destination_id

We do one hot encoder on target variable (regency_cluster), for example, in that row, the regency cluster value is 6, then there will be a column named is_6 which has a value of 1. Thats the clearly example:

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Then we calculate the probability regency_cluster which is selected for each destination_id and give it the name destination_id_enc_n where n is 0 to 99 cluster.

B. Target Encoding by Mode in buyer_id
   We create a variable buyer_id_enc which contains the most cluster bought by the buyer

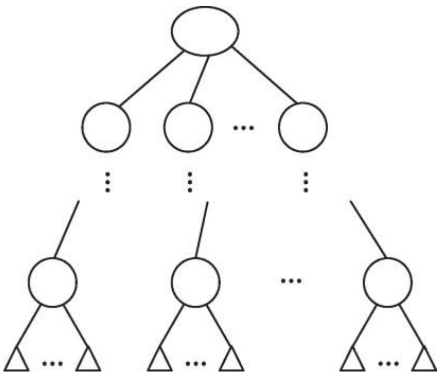
We perform encoding on these two variables(buyer_id and destination_id) because these variables are the top 2 highest relevance values

2.6. Modelling

The train.csv data that we have processed earlier, we separate it into 80% training data and 20% test data. Then model it using a decision tree where the target variable is regency_cluster.

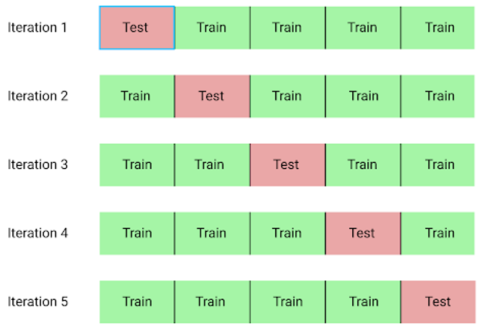
A decision tree is a flow diagram shaped like a tree structure in which each internal node represents a test of an attribute, each branch represents the output of the test and the leaf node represents the classes or class distributions.

Decision tree is used to classify a data sample whose class is not yet known into existing classes. The data test path is first through the root node and the last is through the leaf node which will conclude the class prediction for the data. Attribute data must be categorical data, if it is continuous then the attribute must be discretized first.



We use a decision tree because it can handle multi-class target variables as in the case we are solving now. We get an accuracy of 31.46%
With this model we can predict regency_cluster on test.csv.

2.7.Validating



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)

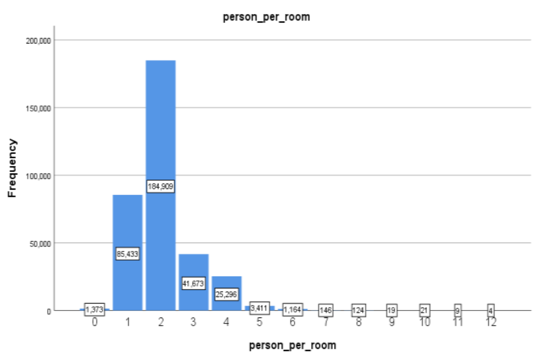
Cross-validation (CV) is a statistical method that can be used to evaluate the performance of a model or algorithm where the data is separated into two subsets, namely learning process data and validation/evaluation data. The model or algorithm is trained by a subset of learning and validated by a subset of validity (in this case subset of validity is test data).

We are sampling the train and test data as shown and enter it into the model that has been created. The average accuracy of this validation is 29.74%.

Chapter III
Data Exploration

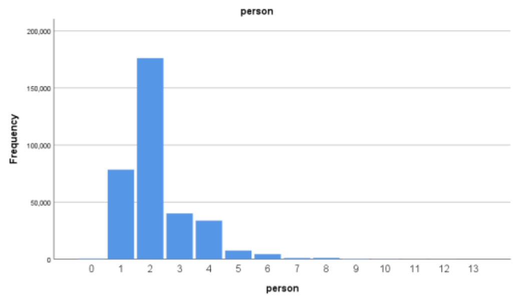
Based on the data, there are several customer characteristics that influence the possibility of customer in choosing a house according to the desired preferences. These characteristics are as follows.

DATA TRAINING
Person per Room



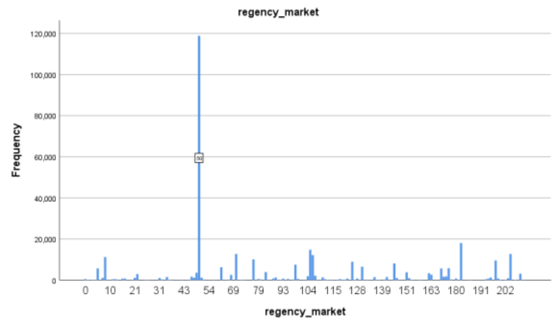
In general, the number of people in each room is two people. The number of people living will be directly related to the ratio of the room where the ration of the room is one of the factors in determining the cluster of houses.

Person



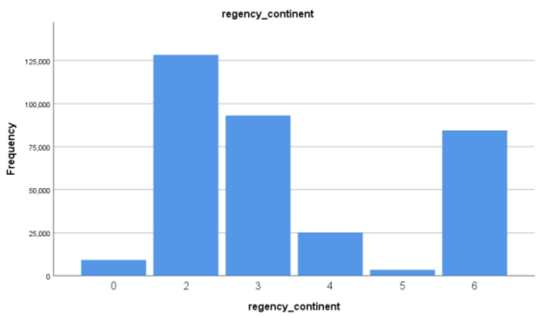
In general, the number of occupants of the house is two people. The number of occupants of the house will affect the determination of the cluster.

Regency Market



In general, customers will choose a cluster with consideration of a standard price with adequate facilities. This can be seen from the chart which shows that most buyers choose relatively low cluster prices.

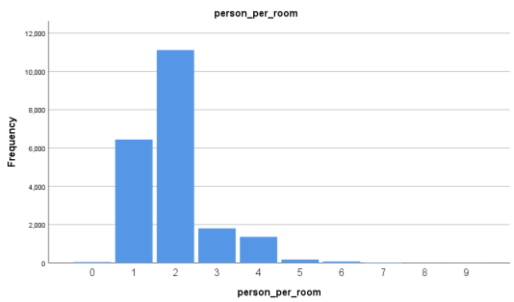
Regency_continent



In general, customers choose the continent with the label number 2. The choice of the destination of the continent affects the clustering because each continent has its own characteristics such as weather. Customers will choose the district environment according to the weather they want.

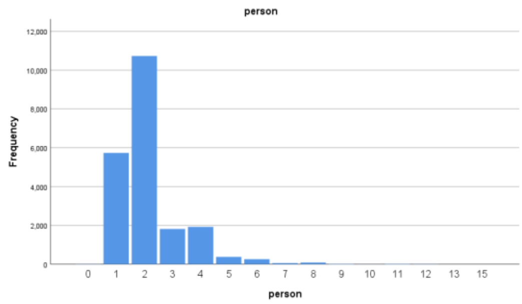
DATA TEST

Person per Room



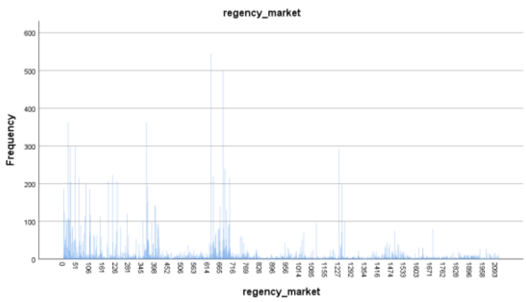
In general, the number of people in each room is two people. The number of people living will be directly related to the ratio of the room .

Person



In general, the number of occupants of the house is two persons. The number of occupants of the house will affect the determination of the cluster.

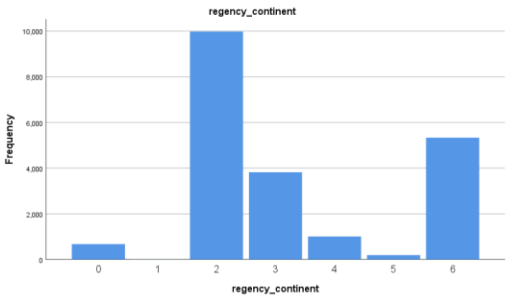
Regency Market



In general, customers will choose a cluster with consideration of a standard price with adequate facilities. This can be seen from the chart which shows that most buyers choose the middle cluster prices.

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)

Regency Continent



In general, customers choose the continent with the label number 2. The choice of the destination of the continent affects the grouping.

Chapter IV

Based on our analysis as a consultant, we have determined residence's cluster that will be chosen by customer based on several clusters that we determine according to the specifications we have determined. We use variables destination_id, buyer_id, regency_market, buyer_city, regency_continent. Beside that, dif_td_dd, dif_td_bd, dif_bd_dd, person, person_per_room are variable results from engineering variables. In our modelling, we use decision trees because they can handle multi-class target variables. We get 31.46% accuracy.

Based on the results of exploration on data training, we can conclude that the number of people in each room is two people, the number of occupants of the house is two persons, and customers choose the continent with the label number 2.

From the results of exploration on data testing, we can conclude that the number of people in each room is two people, the number of occupants of the house is two persons, and customers choose the continent with the label number 2.

1. Data Praprocessing

1.1. Data Input

```
df_train =
pd.read_csv("https://drive.google.com/uc?id=1aHBmV7ZA_lmIC9dL8P-
6yCF8nqZNda-c")
df_test =
pd.read_csv("https://drive.google.com/uc?id=1rlx8gYDUozbpF3dPFMil7x4g
tNBOMhk")
```

1.2. Data Describe 1

```
df_train.describe
df_test.describe
df_train.head(5)
df_test.head(5)
```

1.3. Looking for Missing Value in Dataset

```
missing_data = pd.DataFrame({'total_missing':
df_train.isnull().sum(), 'perc_missing':
(df_train.isnull().sum()/len(df_train.index))*100})
missing_data

missing_data = pd.DataFrame({'total_missing': df_test.isnull().sum(),
'perc_missing': (df_test.isnull().sum()/len(df_test.index))*100})
missing_data
```

1.4. Count the values each column

```
for column in df_train.columns:
      print("Nilai unik pada variabel "+column)
      print(df_train[column].value_counts())
      print("\n\n")

for column in df_test.columns:
      print("Nilai unik pada variabel "+column)
      print(df_test[column].value_counts())
      print("\n\n")

#Menghapus Kolom yang ada missing valuenya
df_train = df_train.drop(columns=["distance"])
df_test = df_test.drop(columns=["distance"])

#Menghapus Row yang ada missing valuenya
df_train.dropna(inplace=True)
df_test.dropna(inplace=True)
```

```
df_train.head(10)
df_test.head(10)
df_train.sample(10)
#Menghapus Kolom yang tidak penting
df_train = df_train.drop(columns=["Unnamed: 0"])
df_test = df_test.drop(columns=["Unnamed: 0"]) 1

df_train = df_train.rename(columns={"regency_country":
"regency_continent", "regency_continent": "regency_country"})

column_=["regency_country","regency_continent"]
for column in column_:
      print("Nilai unik pada variabel "+column)
      print(df_train[column].value_counts())
      print("\n\n")

df_train.head()

#Handling time series
import datetime
df_train['time_date'] =
pd.to_datetime(df_train['time_date']).dt.strftime("%m/%d/%y")
df_test['time_date'] =
pd.to_datetime(df_test['time_date']).dt.strftime("%m/%d/%y")

#Ubah tipe data
df_train['time_date'] = pd.to_datetime(df_train['time_date'])
df_test['time_date'] = pd.to_datetime(df_test['time_date'])

df_train["buying_date"] = pd.to_datetime(df_train['buying_date'])
df_test["buying_date"] = pd.to_datetime(df_test['buying_date'])

df_train["dealing_date"] = pd.to_datetime(df_train['dealing_date'])
df_test["dealing_date"] = pd.to_datetime(df_test['dealing_date'])

#Selisih time_date x buying_date
df_train["dif_td_bd"] = abs(df_train['time_date'] -
df_train['buying_date'])
df_test["dif_td_bd"] = abs(df_test['time_date'] -
df_test['buying_date'])

#Selisih time_date x buying_date
df_train["dif_td_dd"] = abs(df_train['time_date'] -
df_train['dealing_date'])
df_test["dif_td_dd"] = abs(df_test['time_date'] -
df_test['dealing_date'])
```

```
    #Selisih dealing_date x buying_date
    df_train["dif_bd_dd"] = abs(df_train['buying_date'] -
    df_train['dealing_date']) df_test["dif_bd_dd"] =
    abs(df_test['buying_date'] - df_test['dealing_date'])
```

2. Feature Engineering

```
#Jumlahin Adult sama Child jadi person
df_train["person"] = df_train["adults"] + df_train["children"]
df_test["person"] = df_test["adults"] + df_test["children"]

#person/room ratio
df_train["person_per_room"] = df_train["person"]/df_train["room"]
df_test["person_per_room"] = df_test["person"]/df_test["room"]

days = ["dif_td_bd", "dif_td_dd", "dif_bd_dd"]

for y in days :
    df_train[y] = df_train[y].dt.days
    df_test[y] = df_test[y].dt.days
    df_train[y].astype(int)
    df_test[y].astype(int)

categorical = ["site", "continent_id", "buyer_country",
"buyer_region","buyer_city", "buyer_id", "mobile", "package",
"channel_id", "destination_id", "dest :
target = df_train["regency_cluster"]

df_train_num = df_train.drop(columns=categorical)
df_train_cat = df_train[categorical]

df_train_num.head(5)

#df_train_num = df_train_num.drop(columns=days)
df_train =
df_train.drop(columns=["time_date","buying_date","dealing_date","regency
_cluster"])
#df_test = df_test.drop(columns=days) df_test =
df_test.drop(columns=["time_date","buying_date","dealing_date"])
```

3. Feature Selection

```
def select_features_cat(X_train, y_train):
    fs = SelectKBest(score_func=mutual_info_classif, k='all')
    fs.fit(X_train, y_train) X_train_fs = fs.transform(X_train)
    #X_test_fs = fs.transform(X_test) \
    return X_train_fs, fs #X_test_fs
```

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif

X_train_fs_cat, fs_cat = select_features_cat(df_train, target)

i=0

for x in df_train.columns:
    print('Feature %s: %f' % (x, fs_cat.scores_[i]))
    i=i+1

import matplotlib.pyplot as plt

(pd.Series(fs_cat.scores_,
index=df_train.columns).sort_values(ascending= True)
    .plot(kind='barh'))
plt.title('Feature importances')

#from sklearn.feature_selection import f_classif

#def select_features_num(X_train, y_train):
    #fs = SelectKBest(score_func=f_classif, k='all')
    #fs.fit(X_train, y_train)
    #X_train_fs = fs.transform(X_train)
    #X_test_fs = fs.transform(X_test)
    #return X_train_fs, fs #X_test_fs

#X_train_fs_num, fs_num = select_features_num(df_train_num, target)
#i=0
#for x in df_train_num.columns:
    #print('Feature %s: %f' % (x, fs_num.scores_[i]))
    #i=i+1

#(pd.Series(fs_num.scores_,
    index=df_train_num.columns).sort_values(ascending= True)
    #.plot(kind='barh'))
#plt.title('Feature importances numerical data')

#sel_cat = ["destination_id",
"buyer_id","buyer_city","regency_continent","regency_market"] #sel_cat =
["destination_id"]
#sel_num = ["person_per_room"]#,"dif_td_dd","dif_bd_dd"]

#data testnya jangan lupa
#df1 = df_train_cat[sel_cat]
#df2 = df_train_num[sel_num]
```

```
train = df_train
#import math
train["regency_cluster"] = target

y = pd.get_dummies(train.regency_cluster, prefix='is')
train = pd.concat([train,y], axis=1, join='inner')
test = df_test
train_ = df_train

for x in range(0,100) :
    mean = train['is_'+str(x)].mean()
    mean_encode = train.groupby('destination_id')['is_'+str(x)].mean()
    train_['destination_id_enc_'+str(x)] =
train['destination_id'].map(mean_encode).fillna(mean)
    test['destination_id_enc_'+str(x)] =
test['destination_id'].map(mean_encode).fillna(mean)


sel = []
for x in range(0,99) :
    x=str(x)
    sel.append("destination_id_enc_"+x)
sel.append("buyer_id")
sel.append("regency_market")
sel.append("buyer_city")
sel.append("regency_continent")

#Saving Id
value Id = test["id"]

train_ = train_[sel]
pred = test[sel]

train.columns
Index(['site', 'continent_id', 'buyer_country', 'buyer_region',
'buyer_city', 'buyer_id', 'mobile', 'package', 'channel_id', 'adults',
... 'is_90', 'is_91', 'is_92', 'is_93', 'is_94', 'is_95', 'is_96',
'is_97', 'is_98', 'is_99'], dtype='object', length=125)

mode = train["regency_cluster"].mode()
mode_encode = train.groupby("buyer_id")['regency_cluster'].agg(lambda
x:x.value_counts().index[0])
train_["buyer_id_enc"] = train["buyer_id"].map(mode_encode).fillna(mode)
pred["buyer_id_enc"] = pred["buyer_id"].map(mode_encode).fillna(mode)
```

```
train_ =train_.drop(columns="buyer_id")
pred = pred.drop(columns="buyer_id")

#sel_cat = ["destination_id"]
#sel_num = ["regency_market"]
#sel_feature =
["buyer_id_enc","destination_id_enc","buyer_city_enc","regency_market_en
c"]
#sel_feature =
["destination_id","buyer_id","buyer_city","regency_continent","regency_m
arket","buyer_id_enc","destination_id_enc","buyer_city_enc","regency_
#train = train[sel_feature]

#train_ = train_.drop(columns=["buyer_id","regency_cluster"]) from
sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(train_, target,
train_size = 0.8,test_size=0.2, random_state=123,stratify = target)

from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline

#over = SMOTE()
#under = RandomUnderSampler()
#steps = [('o', over), ('u', under)]
#pipeline = Pipeline(steps=steps)
# transform the dataset
#X_train, y_train = over.fit_resample(X_train, y_train)

# Feature Scaling
#from sklearn.preprocessing import StandardScaler
#X_train = X
#y_train = y

#sc = StandardScaler()
#X_train = sc.fit_transform(X_train)
#X_test = sc.transform(X_test)
```

4. Modelling

```
#Modeling
from sklearn.multiclass import OneVsOneClassifier
from sklearn.svm import SVC
from sklearn.metrics import make_scorer, accuracy_score,
precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split, KFold,
cross_validate, cross_val_predict
```

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)

```
from sklearn.linear_model import LogisticRegression
import warnings
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier,
ExtraTreesClassifier, VotingClassifier, GradientBoostingClassifier

from sklearn.naive_bayes import MultinomialNB, GaussianNB
from sklearn.model_selection import train_test_split from
sklearn.metrics import confusion_matrix from sklearn.metrics import
roc_auc_score
from sklearn.metrics import classification_report
from sklearn.datasets import make_multilabel_classification
from sklearn.svm import SVC
from sklearn.multioutput import MultiOutputClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression

#model = GaussianNB()
model = DecisionTreeClassifier()
#model = LogisticRegression(solver='liblinear')
#model = OneVsRestClassifier(model)

#model = RandomForestClassifier()
#model = GradientBoostingClassifier()
#model = KNeighborsClassifier()
#model = ExtraTreesClassifier()
#model = DecisionTreeRegressor()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
#f1 = f1_score(y_test, y_pred, average='weighted')
print('Accuracy : ', "%.2f" % (accuracy*100))
#print('F1 : ', "%.2f" % (f1*100))
```

5. Validating

```
#Cross-Validation
scoring = {'accuracy' : make_scorer(accuracy_score)}
    #'precision' : make_scorer(precision_score),
    #'recall' : make_scorer(recall_score)}
    #'f1_score' : make_scorer(f1_score(average='weighted'))}
kfold = KFold(n_splits=5, random_state=1234, shuffle = True)
results_clf = cross_validate(estimator=model,X=X_train,
```

```
            y=y_train,
            cv=kfold,
            scoring=scoring)
results_clf

a = results_clf.get('test_accuracy')
print('Validation Mean Accuracy : ', "%.2f" % (a.mean()*100))
```

6. #Submission

```
Id.to_numpy()
Y_pred = model.predict(pred)
submission = pd.DataFrame({
    "id": Id,
    "regency_cluster": Y_pred
    })
submission.to_csv("submission.csv", index=False)

submission
```

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)