ANSWER SHEET
DAC 2021

DATA ANALYSIS COMPETITION 2021

TEAM NAME Diandaru Suchrady

TEAM ID ID-21-0105

UNIVERSITY Institut Teknologi Bandung

CHAPTER I

Professionals Opinion

Before making any assumptions about the features contained in the dataset, the proper way to find a clue of whether a feature is considered necessary or not is by asking the professionals. According to Susi Sundari (2021), a senior property marketer, there are three factors of a customer's preference for buying a house:

1. Price

   The price depends on the customer's budget.

2. Location

   Customers are very likely to buy a house near their workplace, relatives, and their old residence.

3. House Shape

   House shape affects the customer's interest.

Susi also stated that there is no correlation between customer's preferences and date of purchase. On the other hand, Dady Suchrady (2021), a public notary inferred that dates play a role in the event of house or property transactions. From his past experience, documents for transactions are usually done by the month of April, hence it is inferred that transactions mostly occur in the second semester of the year. In addition to that, many of Dady's clients have more budget during the second semester since they gain more income that time of the year (In Indonesia, when the government gives projects to companies, their employees may gain bonuses). Therefore, it can be assumed that there is a chance that a house with a higher price will have more demand in the second semester of a year.

Research Study

According to the results of a study conducted by Amrin Fauzi (2012) about what affects a consumer's decision in buying a house using a regression analysis, consumer's decision in buying a house depends with the rate of 64.2% on location, economic status, and lifestyle.

In another study with the same topic conducted by Sutianingsih (2010), quality of a building, pricing, location, and promotion determine the decision making of a housing consumer by 61.1% and the 38.9% is affected by other variables.

Assumptions

Since the test dataset ‘Test.csv’ does not contain the ‘cnt’ and ‘dealing’ columns, backed up by the decision to make the machine learning model to only evaluate categorical variables, the ‘cnt’ and ‘dealing’ columns are not to be included in the model construction. In addition, the ‘buyer_id’ will not be involved in the model to prevent any overfitting to happen since a buyer’s identity is deemed irrelevant.

Decision

It is concluded that it is best to not include the ‘cnt’, ‘dealing’, and ‘buyer_id’ column. In the further process of preparing the dataset, data reduction and feature selection will be held to reduce the dimension of the dataset, so there is a chance of excluding other columns.

CHAPTER II

Preprocessing

Data Cleansing

In this process, firstly the information of the data must be seen.

```
Data columns (total 24 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   time_date          343583 non-null  object
 1   site               343583 non-null  int64
 2   continent_id       343583 non-null  int64
 3   buyer_country      343583 non-null  int64
 4   buyer_region       343583 non-null  int64
 5   buyer_city         343583 non-null  int64
 6   distance           145685 non-null  float64
 7   buyer_id           343583 non-null  int64
 8   mobile             343583 non-null  int64
 9   package            343583 non-null  int64
 10  channel_id         343583 non-null  int64
 11  buying_date        342885 non-null  object
 12  dealing_date       342885 non-null  object
 13  adults             343583 non-null  int64
 14  children           343583 non-null  int64
 15  room               343583 non-null  int64
 16  destination_id     343583 non-null  int64
 17  destination_type   343583 non-null  int64
 18  dealing            343583 non-null  int64
 19  regency_continent  343583 non-null  int64
 20  regency_country    343583 non-null  int64
 21  regency_market     343583 non-null  int64
 22  cnt                343583 non-null  int64
 23  regency_cluster    343583 non-null  int64
dtypes: float64(1), int64(20), object(3)
```

As the ‘cnt’, ‘dealing’, and the ‘buyer_id’ have been decided to be dropped, the dataset will remain the rest columns (21 columns).

From the data information, it is seen that the ‘distance’, ‘buying_date’, and ‘dealing_date’ contain null values. Since the majority of the data types are categorical, before deciding to drop the null values, it is best to check the class loss when the null values are dropped.

Following are the class numbers for each feature before and after dropping the null values:

Table 2.1. Class Value Counts of Each Feature

Column Name	Nothing to drop	Drop distance, buying_date, and dealing_date	Drop distance	Drop buying_date	Drop dealing_date
site	30	23	23	30	30
continent_id	5	5	5	5	5
buyer_country	155	17	17	155	155
buyer_region	653	182	182	653	653
buyer_city	7256	3073	3074	7249	7249
mobile	2	2	2	2	2
package	2	2	2	2	2
channel_id	11	11	11	11	11
adults	10	10	10	10	10
children	9	8	8	9	9
room	9	8	8	8	8
destination_id	9831	6636	6647	9811	9811
destination_type	8	8	8	8	8
regency_continent	40	36	36	40	40
regency_country	6	6	6	6	6
regency_market	179	157	157	179	179
regency_cluster	100	100	100	100	100

Preventing the loss information in categorical features, it is decided to fill the null values and not drop the outliers.

Data Transformation

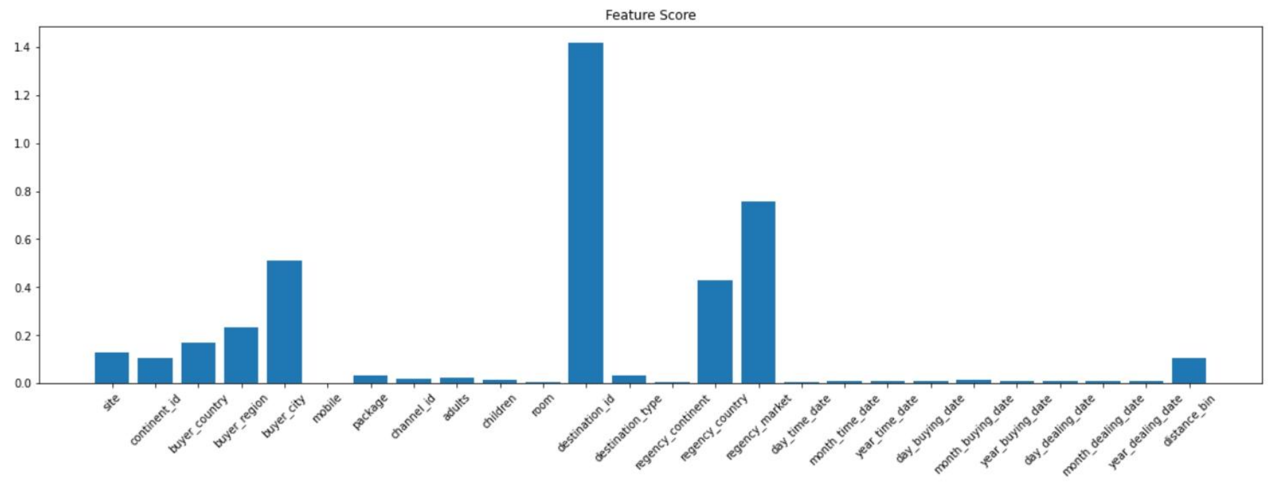
In the data transformation part, the distance column is grouped into 5 corresponding bins with the interval of every bins are as the following:

1. (0.0046, 326.593] labeled as 0
2. (326.593, 1100.383] labeled as 1
3. [(1100.383, 2808.036] labeled as 2
4. (2808.036, 11761.396]] labeled as 3
5. Null values labeled as -1

The date columns (time_date, buying_date, and dealing_date) are extracted to day, month, and year. This process results in all columns bening of categorical type.

Data Reduction

To reduce the burden of the model while fitting/training the dataset, feature selection should be conducted. Mutual information feature selection is chosen because the dataset's majority columns are categorical or numeric.

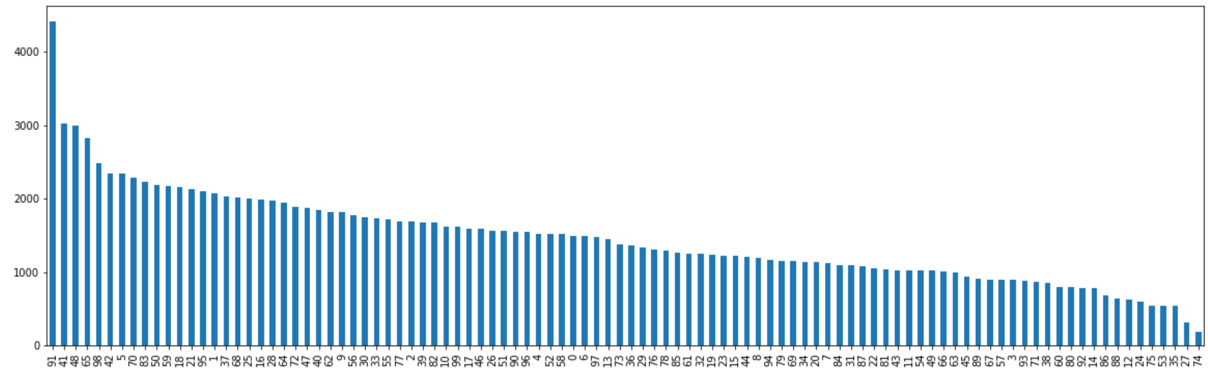


Graph 2.1. Feature score bar plot

After running the feature selection, it is concluded that the chosen columns for the modelling process are ‘site’, ‘continent_id’, ‘buyer_country’, ‘buyer_region’, ‘buyer_city’, ‘destination_id’, ‘regency_country’, ‘regency_market’, and ‘distance_bin’.

Data Balancing

In order to minimize overfitting in our model, we need to take a look at the frequency of each class in the target feature. A dominant class in the target feature may result in a model that assumes classes that have small frequencies as an insignificant class, thus making it less considered by the model. The following is the bar plot for the frequency distribution of the target class ‘regency_cluster’ of the dataset Train.csv



Graph 2.2. Target class frequency distribution

As seen on the bar chart, there is an imbalance between the distribution of class frequencies. Over-sampling is done to resolve this issue, that is extrapolating classes of the target attribute that have less entries using Synthetic Minority Over-sampling Technique for Nominal and Continuous (SMOTENC) and Synthetic Minority Over-sampling Technique for Nominal (SMOTEN) through scikit library. Thus, every class from the target attribute now has 8077 entries.

Modelling and Validation

An experiment is done to determine the best algorithm to predict the target attributes. The experiment compares the performance of Random Forest Classifier and the Decision Tree Classifier using the accuracy as a metric. Each of the algorithms will use both the entropy and gini function. The models evaluate the result using ten fold cross validation.

There are twelve combinations of model:

1. Random forest classification with entropy function splitting using the SMOTEN transformed dataset
2. Random forest classification with gini function splitting using the SMOTEN transformed dataset
3. Random forest classification with entropy function splitting using the SMOTENC transformed dataset
4. Random forest classification with gini function splitting using the SMOTENC transformed dataset
5. Random forest classification with entropy function splitting using the imbalanced dataset
6. Random forest classification with gini function splitting using the imbalance dataset
7. Decision tree classification with entropy function splitting using the SMOTEN transformed dataset
8. Decision tree classification with gini function splitting using the SMOTEN transformed dataset
9. Decision tree classification with entropy function splitting using the SMOTENC transformed dataset
10. Decision tree classification with gini function splitting using the SMOTENC transformed dataset
11. Decision tree classification with entropy function splitting using the imbalanced dataset
12. Decision tree classification with gini function splitting using the imbalanced dataset

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)

CHAPTER III

The result of the experiment is as follows

Model	Accuracy
1	49.63 %
2	49.63 %
3	47.23 %
4	47.23 %
5	28.22 %
6	28.22 %
7	49.81%
8	49.77%
9	47.29%
10	47.19%
11	28.40%
12	28.38%

It can be inferred from the table that models trained with unbalanced dataset (5, 6, 11, 12) performed worse than models that were trained using balanced datasets either using SMOTEN or SMOTENC. This may be due to the overfitting of the model due to the dominance of some target attribute class that makes up a large percentage of the training dataset. It can also be concluded from the results that SMOTEN (1, 2, 7, 8) data balancing works better than SMOTENC (3, 4, 9, 10), meaning that it is better to assume all attributes in this dataset as categorical. Although only by a small percentage, the entropy function works better than the gini function in terms of feature selection.

CHAPTER IV

Conclusion

Based on the conducted experiment, it is found that the balance of the data greatly affects the performance of the model, therefore data balancing is a crucial part of data preprocessing. In addition, mutual information can be used to determine which categorical attributes are best to be put into the model which concludes that in this case, consumer preference in choosing the residence cluster is highly dependent on location represented by continent_id, buyer_country, buyer_region, buyer_city, destination_id, regency_country, and distance_bin.

Suggestion

More data preprocessing methods may be done to improve the performance of the model and the experiment may involve more varying algorithms.

Attachment

Residence Reccomendation Data Analysis

In [112…
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [113…
```python
# df = pd.read_csv('https://raw.githubusercontent.com/rdyzakya/DAC/master/Train.csv?token=ANMX7UVSDMRT7YXXECIV3KLA7T7TW')
df = pd.read_csv('Train.csv')
```

Check data information

In [114…
```python
#delete unnecessary column
del df['Unnamed: 0']

#check information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 343583 entries, 0 to 343582
Data columns (total 24 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   time_date          343583 non-null  object
 1   site               343583 non-null  int64
 2   continent_id       343583 non-null  int64
 3   buyer_country      343583 non-null  int64
 4   buyer_region       343583 non-null  int64
 5   buyer_city         343583 non-null  int64
 6   distance           145685 non-null  float64
 7   buyer_id           343583 non-null  int64
 8   mobile             343583 non-null  int64
 9   package            343583 non-null  int64
 10  channel_id         343583 non-null  int64
 11  buying_date        342885 non-null  object
 12  dealing_date       342885 non-null  object
 13  adults             343583 non-null  int64
 14  children           343583 non-null  int64
 15  room               343583 non-null  int64
 16  destination_id     343583 non-null  int64
 17  destination_type   343583 non-null  int64
 18  dealing            343583 non-null  int64
 19  regency_continent  343583 non-null  int64
 20  regency_country    343583 non-null  int64
 21  regency_market     343583 non-null  int64
 22  cnt                343583 non-null  int64
 23  regency_cluster    343583 non-null  int64
dtypes: float64(1), int64(20), object(3)
memory usage: 62.9+ MB
```

Drop cnt, dealing, buyer_id to match the Test.csv

In [115…
```python
#since Test.csv doesn't have cnt and dealing it's best to drop it
#and drop the buyer_id as the buyer_id isn't needed for the modelling
df = df.drop(labels=['cnt','buyer_id','dealing'],axis=1)
```

There are null values in the distance, buying_date, dealing_date, before dropping any null values, check class loss in the categoric variables

In [116…
```python
#checking the unique values of each columns
def check_classloss(df,nclabel=[],dropped_label=[]):
  df2 = df.dropna(subset=dropped_label)
  df2 = df2.drop(labels=nclabel,axis=1)
  uni_len = {}
  for col in df2.columns:
    uni_len[col] = len(df2[col].unique())
  return uni_len
```

In [117…
```python
drop_nothing = check_classloss(df,['distance','time_date','buying_date','dealing_date'],[])
drop_allna = check_classloss(df,['distance','time_date','buying_date','dealing_date'],['distance','buying_date','dealing_date'])
drop_distance = check_classloss(df,['distance','time_date','buying_date','dealing_date'],['distance'])
drop_bdate = check_classloss(df,['distance','time_date','buying_date','dealing_date'],['buying_date'])
drop_ddate = check_classloss(df,['distance','time_date','buying_date','dealing_date'],['dealing_date'])
```

In [118…
```python
#if drop nothing
drop_nothing
```

Out[118…
```
{'site': 30,
 'continent_id': 5,
 'buyer_country': 155,
 'buyer_region': 653,
 'buyer_city': 7256,
 'mobile': 2,
 'package': 2,
 'channel_id': 11,
 'adults': 10,
 'children': 9,
 'room': 9,
 'destination_id': 9831,
 'destination_type': 8,
 'regency_continent': 40,
 'regency_country': 6,
 'regency_market': 179,
 'regency_cluster': 100}
```

In [119…
```python
#drop distance,buying_date,dealing_date
drop_allna
```

Out[119…
```
{'site': 23,
 'continent_id': 5,
 'buyer_country': 17,
 'buyer_region': 182,
 'buyer_city': 3073,
 'mobile': 2,
 'package': 2,
 'channel_id': 11,
 'adults': 10,
 'children': 8,
 'room': 8,
 'destination_id': 6636,
 'destination_type': 8,
 'regency_continent': 36,
 'regency_country': 6,
 'regency_market': 157,
 'regency_cluster': 100}
```

In [120…
```python
#drop distance only
drop_distance
```

Out[120…
```
{'site': 23,
 'continent_id': 5,
 'buyer_country': 17,
 'buyer_region': 182,
 'buyer_city': 3074,
 'mobile': 2,
 'package': 2,
 'channel_id': 11,
 'adults': 10,
 'children': 8,
 'room': 8,
 'destination_id': 6647,
 'destination_type': 8,
 'regency_continent': 36,
 'regency_country': 6,
 'regency_market': 157,
 'regency_cluster': 100}
```

In [121…
```python
#drop buying_date
drop_bdate
```

Out[121…
```
{'site': 30,
 'continent_id': 5,
 'buyer_country': 155,
 'buyer_region': 653,
 'buyer_city': 7249,
 'mobile': 2,
 'package': 2,
 'channel_id': 11,
 'adults': 10,
 'children': 9,
 'room': 8,
 'destination_id': 9811,
 'destination_type': 8,
 'regency_continent': 40,
 'regency_country': 6,
 'regency_market': 179,
 'regency_cluster': 100}
```

In [122…
```python
#drop dealing_date
drop_ddate
```

Out[122…
```
{'site': 30,
 'continent_id': 5,
 'buyer_country': 155,
 'buyer_region': 653,
 'buyer_city': 7249,
 'mobile': 2,
 'package': 2,
 'channel_id': 11,
 'adults': 10,
 'children': 9,
 'room': 8,
 'destination_id': 9811,
 'destination_type': 8,
 'regency_continent': 40,
 'regency_country': 6,
 'regency_market': 179,
 'regency_cluster': 100}
```

Data Transformation

since there are class losses in the categoric features if the distance/buying_date/dealing_date is dropped hence, decides to fill the null values

In [125…
```python
#extracting the dates for filling the null values

#changing data types
df['time_date'] = pd.to_datetime(df['time_date'])
df['buying_date'] = pd.to_datetime(df['buying_date'])
df['dealing_date'] = pd.to_datetime(df['dealing_date'])

#time_date
df['day_time_date'] = df['time_date'].dt.day
df['month_time_date'] = df['time_date'].dt.month
df['year_time_date'] = df['time_date'].dt.year

#buying_date
df['day_buying_date'] = df['buying_date'].dt.day
df['month_buying_date'] = df['buying_date'].dt.month
df['year_buying_date'] = df['buying_date'].dt.year

#dealing_date
df['day_dealing_date'] = df['dealing_date'].dt.day
df['month_dealing_date'] = df['dealing_date'].dt.month
df['year_dealing_date'] = df['dealing_date'].dt.year

#drop the dates
df = df.drop(labels=['time_date','buying_date','dealing_date'],axis=1)
```

Fill Null Values

In [126…
```python
df['distance_bin'] = pd.qcut(df['distance'],q=4,labels=[0,1,2,3])
df['distance_bin'] = df['distance_bin'].cat.add_categories(-1)
df['distance_bin'] = df['distance_bin'].fillna(value=-1)
df['distance_bin'] = df['distance_bin'].astype(int).astype('category')

#to bin the distance in Test.csv
distance_bin = pd.qcut(df['distance'],q=4).unique()
```

In [127…
```python
#fill null values with -1
df = df.fillna(value=-1)
```

Feature Selection

In [128…
```python
#define the x columns and y column
x_cols = df.columns.to_list()
y_col = 'regency_cluster'
x_cols.remove('regency_cluster')
x_cols.remove('distance')
```

In [130…
```python
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.feature_selection import mutual_info_classif
from sklearn.feature_selection import f_classif

def create_fs(xtrain,ytrain,method):
  fs = SelectKBest(score_func=method,k='all')
  fs.fit(xtrain,ytrain)
  return fs

def print_fs_score(x_cols,fs):
  for i in range(len(fs.scores_)):
    print('Feature %s: %f' % (x_cols[i], fs.scores_[i]))
```

In [131…
```python
#create feature selection using mutual information
fs_mi = create_fs(df[x_cols],df[y_col],mutual_info_classif)
```

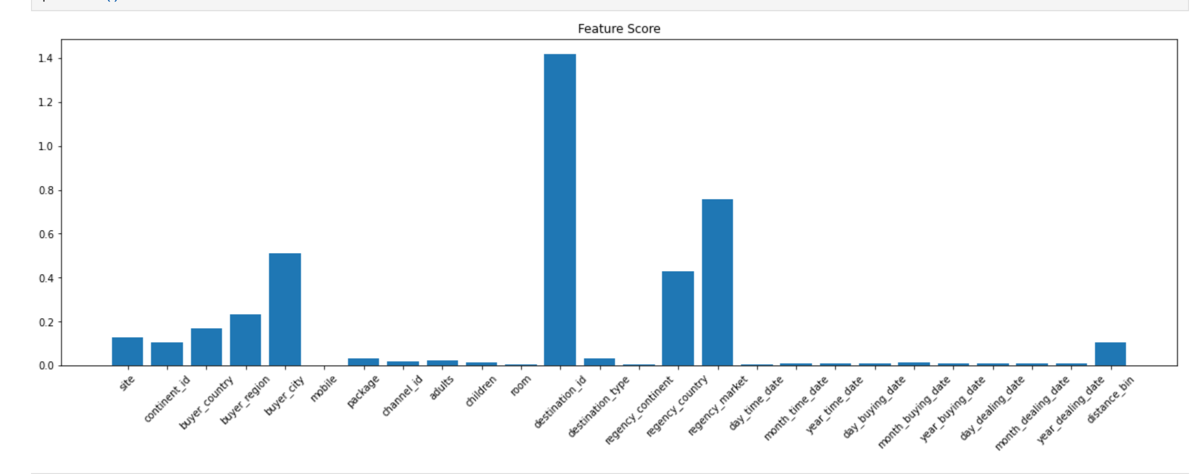
In [132…
```python
print("Feature score for mutual information feature selection\n")
print_fs_score(x_cols,fs_mi)
```

```
Feature score for mutual information feature selection

Feature site: 0.126481
Feature continent_id: 0.106386
Feature buyer_country: 0.167690
Feature buyer_region: 0.234650
Feature buyer_city: 0.511225
Feature mobile: 0.000000
Feature package: 0.030740
Feature channel_id: 0.016649
Feature adults: 0.023396
Feature children: 0.012530
Feature room: 0.004241
Feature destination_id: 1.416510
Feature destination_type: 0.032782
Feature regency_continent: 0.003482
Feature regency_country: 0.427398
Feature regency_market: 0.758817
Feature day_time_date: 0.004808
Feature month_time_date: 0.007425
Feature year_time_date: 0.007240
Feature day_buying_date: 0.008460
Feature month_buying_date: 0.014674
Feature year_buying_date: 0.009268
Feature day_dealing_date: 0.010553
Feature month_dealing_date: 0.009297
Feature year_dealing_date: 0.009055
Feature distance_bin: 0.105644
```

In [134…
```python
#plotting the feature selection result

fig,ax = plt.subplots(figsize=(20,6))
plt.bar(x_cols,fs_mi.scores_)
plt.xticks(rotation=45)
plt.title('Feature Score')
plt.show()
```



In [21]:
```python
#select certain columns chosen from the feature selection
df2 = df[['site','continent_id','buyer_country','buyer_region','buyer_city','destination_id','regency_country','regency_market','distance_bin','reg
```

In [22]:
```python
#changing the datatypes into categories
for col in df2.columns:
  df2[col] = df2[col].astype('category')
```

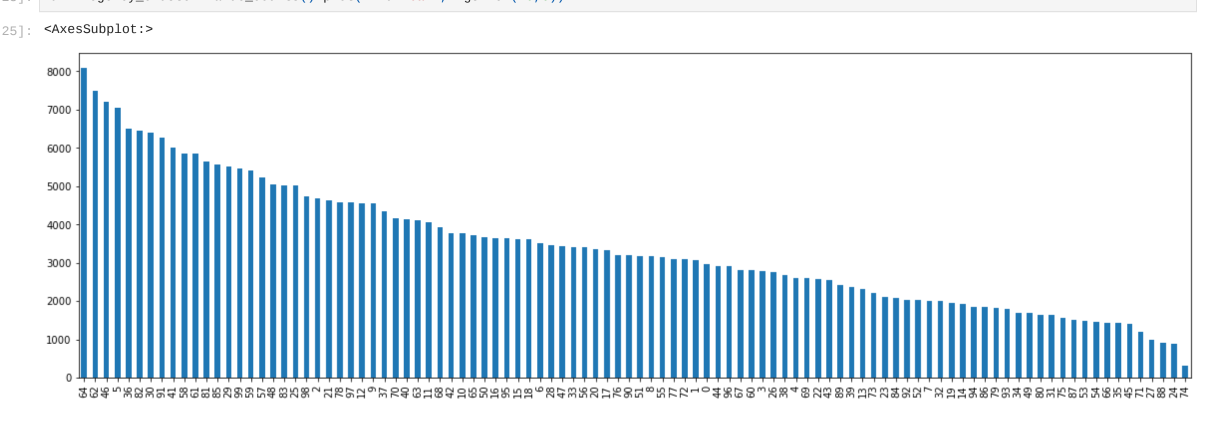
In [23]:
```python
#make sure there are no class loss in the way of preprocessing
check_classloss(df2,[],[])
```

Out[23]:
```
{'site': 30,
 'continent_id': 5,
 'buyer_country': 155,
 'buyer_region': 653,
 'buyer_city': 7256,
 'destination_id': 9831,
 'regency_country': 6,
 'regency_market': 179,
 'distance_bin': 5,
 'regency_cluster': 100}
```

Check the balance of the target feature

In [25]:
```python
df2.regency_cluster.value_counts().plot(kind='bar',figsize=(20,6))
```

Out[25]: <AxesSubplot:>



Data Balancing

In [26]:
```python
#class with least value counts
valcnt = df2['regency_cluster'].value_counts()
print("Class with the least value counts : " + str(valcnt[valcnt == valcnt.min()].index[0]))
print("with the value counts : " + str(valcnt.min()))
```

```
Class with the least value counts : 74
with the value counts : 324
```

In [27]:
```python
#shuffle the dataset
df2 = df2.sample(frac=1, random_state=13519000)

#get classes
classes = df2['regency_cluster'].unique()

#initial dataset
df_balanced = df2.loc[df2['regency_cluster'] == valcnt[valcnt == valcnt.min()].index[0]]

#trimming each class to 324 and concat to new dataframe
for c in classes:
  if c != 74:
    trimmed_class = df2.loc[df2['regency_cluster'] == c][:valcnt.min()]
    df_balanced = pd.concat([df_balanced,trimmed_class])
```

In [28]:
```python
check_classloss(df_balanced,[],[])
```

Out[28]:
```
{'site': 28,
 'continent_id': 5,
 'buyer_country': 126,
 'buyer_region': 523,
 'buyer_city': 4046,
 'destination_id': 4187,
 'regency_country': 6,
 'regency_market': 151,
 'distance_bin': 5,
 'regency_cluster': 100}
```

There are class loss if undersampling is chosen choosing oversampling rather than undersampling to prevent any class loss, so the model would handle all class in the features

In [31]:
```python
# OVER SAMPLING
from collections import Counter
from imblearn.over_sampling import SMOTENC
from imblearn.over_sampling import SMOTEN

# shuffle the dataset
dfnew = df2.copy()
X = dfnew.loc[:, dfnew.columns != 'regency_cluster'].copy()
X = X.to_numpy()
y = dfnew.loc[:, dfnew.columns == 'regency_cluster'].copy()
y = y.to_numpy().flatten()
```

In [32]:
```python
print(f'Original dataset shape {X.shape}')
# Original dataset shape (jml row, jml colom)
print(f'Original dataset samples per class {Counter(y)}')
# Original dataset samples per class Counter({kelas1: jml, kelas2: jml})
```

```
Original dataset shape (343583, 9)
Original dataset samples per class Counter({64: 8077, 62: 7486, 46: 7195, 5: 7041, 36: 6491, 82: 6445, 30: 6386, 91: 6277, 41: 6008, 58: 5850, 61:
5849, 81: 5649, 85: 5562, 29: 5503, 99: 5451, 59: 5414, 57: 5237, 48: 5057, 83: 5021, 25: 5009, 98: 4745, 2: 4671, 21: 4627, 78: 4582, 97: 4572, 1
2: 4561, 9: 4559, 37: 4346, 70: 4163, 40: 4124, 63: 4120, 11: 4051, 68: 3918, 42: 3778, 10: 3769, 65: 3732, 50: 3656, 16: 3649, 95: 3646, 15: 3626,
18: 3606, 6: 3510, 28: 3457, 47: 3426, 33: 3415, 56: 3408, 20: 3356, 17: 3324, 76: 3206, 90: 3192, 51: 3186, 8: 3169, 55: 3139, 77: 3091, 72: 3085,
1: 3058, 0: 2959, 44: 2922, 96: 2907, 67: 2819, 60: 2808, 3: 2778, 26: 2749, 38: 2690, 4: 2610, 69: 2604, 22: 2587, 43: 2548, 89: 2409, 39: 2373, 1
3: 2320, 73: 2221, 23: 2119, 84: 2071, 92: 2032, 52: 2029, 7: 2004, 32: 2001, 19: 1952, 14: 1935, 94: 1859, 86: 1847, 79: 1827, 93: 1804, 34: 1704,
49: 1688, 80: 1636, 31: 1634, 75: 1565, 87: 1503, 53: 1471, 54: 1450, 66: 1440, 35: 1435, 45: 1406, 71: 1197, 27: 998, 88: 920, 24: 897, 74: 324})
```

In [33]:
```python
# fitting using SMOTENC
sm1 = SMOTENC(random_state=42, categorical_features=[0,1,2,3,4,5,6,7])
X_res1, y_res1 = sm1.fit_resample(X, y)
print(f'Resampled dataset samples per class {Counter(y_res1)}')
```

```
Resampled dataset samples per class Counter({98: 8077, 36: 8077, 99: 8077, 21: 8077, 50: 8077, 46: 8077, 77: 8077, 64: 8077, 91: 8077, 86: 8077, 5:
8077, 42: 8077, 18: 8077, 41: 8077, 59: 8077, 70: 8077, 87: 8077, 79: 8077, 17: 8077, 60: 8077, 53: 8077, 25: 8077, 62: 8077, 3: 8077, 78: 8077, 8
2: 8077, 45: 8077, 38: 8077, 61: 8077, 15: 8077, 26: 8077, 30: 8077, 85: 8077, 11: 8077, 0: 8077, 7: 8077, 92: 8077, 67: 8077, 22: 8077, 96: 8077,
81: 8077, 43: 8077, 23: 8077, 20: 8077, 19: 8077, 76: 8077, 72: 8077, 48: 8077, 10: 8077, 31: 8077, 29: 8077, 57: 8077, 35: 8077, 8: 8077, 16: 807
7, 95: 8077, 94: 8077, 58: 8077, 83: 8077, 47: 8077, 6: 8077, 24: 8077, 54: 8077, 39: 8077, 90: 8077, 13: 8077, 89: 8077, 32: 8077, 1: 8077, 49: 80
77, 68: 8077, 2: 8077, 12: 8077, 14: 8077, 97: 8077, 28: 8077, 63: 8077, 9: 8077, 40: 8077, 34: 8077, 66: 8077, 37: 8077, 80: 8077, 4: 8077, 51: 80
77, 56: 8077, 44: 8077, 93: 8077, 75: 8077, 73: 8077, 33: 8077, 84: 8077, 71: 8077, 65: 8077, 52: 8077, 74: 8077, 27: 8077, 55: 8077, 88: 8077, 69:
8077})
```

In [34]:
```python
# fitting using SMOTEN
sm2 = SMOTEN(random_state=42)
X_res2, y_res2 = sm2.fit_resample(X, y)
print(f'Resampled dataset samples per class {Counter(y_res2)}')
```

```
Resampled dataset samples per class Counter({98: 8077, 36: 8077, 99: 8077, 21: 8077, 50: 8077, 46: 8077, 77: 8077, 64: 8077, 91: 8077, 86: 8077, 5:
8077, 42: 8077, 18: 8077, 41: 8077, 59: 8077, 70: 8077, 87: 8077, 79: 8077, 17: 8077, 60: 8077, 53: 8077, 25: 8077, 62: 8077, 3: 8077, 78: 8077, 8
2: 8077, 45: 8077, 38: 8077, 61: 8077, 15: 8077, 26: 8077, 30: 8077, 85: 8077, 11: 8077, 0: 8077, 7: 8077, 92: 8077, 67: 8077, 22: 8077, 96: 8077,
81: 8077, 43: 8077, 23: 8077, 20: 8077, 19: 8077, 76: 8077, 72: 8077, 48: 8077, 10: 8077, 31: 8077, 29: 8077, 57: 8077, 35: 8077, 8: 8077, 16: 807
7, 95: 8077, 94: 8077, 58: 8077, 83: 8077, 47: 8077, 6: 8077, 24: 8077, 54: 8077, 39: 8077, 90: 8077, 13: 8077, 89: 8077, 32: 8077, 1: 8077, 49: 80
77, 68: 8077, 2: 8077, 12: 8077, 14: 8077, 97: 8077, 28: 8077, 63: 8077, 9: 8077, 40: 8077, 34: 8077, 66: 8077, 37: 8077, 80: 8077, 4: 8077, 51: 80
77, 56: 8077, 44: 8077, 93: 8077, 75: 8077, 73: 8077, 33: 8077, 84: 8077, 71: 8077, 65: 8077, 52: 8077, 74: 8077, 27: 8077, 55: 8077, 88: 8077, 69:
8077})
```

In [35]:
```python
#balanced data
df3 = pd.DataFrame(X_res2,columns=['site','continent_id','buyer_country','buyer_region','buyer_city','destination_id','regency_country','regency_ma
df3['regency_cluster'] = pd.Series(y_res2)
```

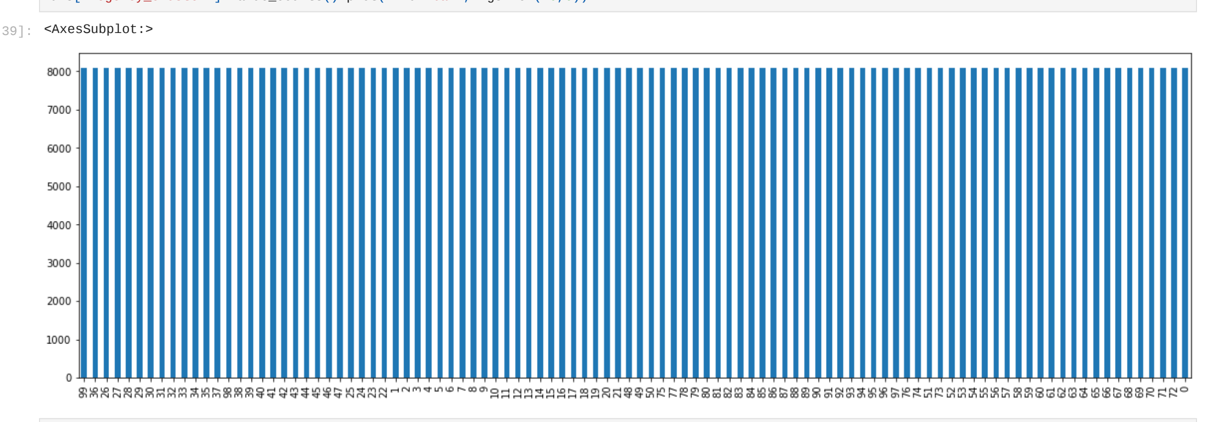
In [52]:
```python
#balanced data
df4 = pd.DataFrame(X_res1,columns=['site','continent_id','buyer_country','buyer_region','buyer_city','destination_id','regency_country','regency_ma
df4['regency_cluster'] = pd.Series(y_res1)
```

In [38]:
```python
#changing the datatypes into categories
for col in df3.columns:
  df3[col] = df3[col].astype(int).astype('category')
```

In [53]:
```python
#changing the datatypes into categories
for col in df4.columns:
  df4[col] = df4[col].astype(int).astype('category')
```

In [39]:
```python
#plotting the bar
df3['regency_cluster'].value_counts().plot(kind='bar',figsize=(20,6))
```

Out[39]: <AxesSubplot:>



In [40]:
```python
check_classloss(df3,[],[])
```

Out[40]:
```
{'site': 30,
 'continent_id': 5,
 'buyer_country': 155,
 'buyer_region': 653,
 'buyer_city': 7256,
 'destination_id': 9831,
 'regency_country': 6,
 'regency_market': 179,
 'distance_bin': 5,
 'regency_cluster': 100}
```

Modelling

In [137…
```python
x_cols = df3.columns.to_list()
x_cols.remove('regency_cluster')
y_col = 'regency_cluster'
```

In [42]:
```python
# Import the model we are using
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

# Instantiate model with 30 decision trees using entropy
rf1 = RandomForestClassifier(n_estimators = 30, random_state = 13519000, criterion='entropy')

# rf1.fit(x_train,y_train)

# Instantiate model with 30 decision trees using gini
rf2 = RandomForestClassifier(n_estimators = 30, random_state = 13519000, criterion='entropy')

# rf2.fit(x_train,y_train)
```

In [48]:
```python
#SMOTENed dataset
df3_x = df3[x_cols].copy()
df3_y = df3[y_col].copy()
```

In [54]:
```python
#SMOTENC-ed dataset
df4_x = df4[x_cols].copy()
df4_y = df4[y_col].copy()
```

In [81]:
```python
#imbalance dataset
df2_x = df2[x_cols].copy()
df2_y = df2[y_col].copy()
```

Validation

In [50]:
```python
scores_entropy = cross_val_score(rf1, df3_x, df3_y, cv=10)
scores_gini = cross_val_score(rf2, df3_x, df3_y, cv=10)
```

In [55]:
```python
print(scores_entropy)
```

```
[0.33892534 0.35124427 0.41281416 0.46211465 0.52484833 0.55373282
 0.5688003  0.57953448 0.58427634 0.58718584]
```

In [56]:
```python
print(scores_gini)
```

```
[0.33892534 0.35124427 0.41281416 0.46211465 0.52484833 0.55373282
 0.5688003  0.57953448 0.58427634 0.58718584]
```

In [68]:
```python
scores_entropy2 = cross_val_score(rf1, df4_x, df4_y, cv=10)
scores_gini2 = cross_val_score(rf2, df4_x, df4_y, cv=10)
```

In [69]:
```python
print(scores_entropy2)
```

```
[0.35752136 0.36668317 0.40974372 0.44803764 0.49144484 0.51259131
 0.52536833 0.53549585 0.53793488 0.53909868]
```

In [70]:
```python
print(scores_gini2)
```

```
[0.35752136 0.36668317 0.40974372 0.44803764 0.49144484 0.51259131
 0.52536833 0.53549585 0.53793488 0.53909868]
```

In [82]:
```python
scores_entropy3 = cross_val_score(rf1, df2_x, df2_y, cv=10)
scores_gini3 = cross_val_score(rf2, df2_x, df2_y, cv=10)
```

In [89]:
```python
print("Random Forest Classifier:")
print("Accuracy for using SMOTEN-ed dataset using entropy criterion:", scores_entropy.mean()*100, "%")
print("Accuracy for using SMOTEN-ed dataset using gini criterion:", scores_gini.mean()*100, "%")
```

```
Random Forest Classifier:
Accuracy for using SMOTEN-ed dataset using entropy criterion: 49.63476538318684 %
Accuracy for using SMOTEN-ed dataset using gini criterion: 49.63476538318684 %
```

In [90]:
```python
print("Random Forest Classifier:")
print("Accuracy for using SMOTENC-ed dataset using entropy criterion:", scores_entropy2.mean()*100, "%")
print("Accuracy for using SMOTENC-ed dataset using gini criterion:", scores_gini2.mean()*100, "%")
```

```
Random Forest Classifier:
Accuracy for using SMOTENC-ed dataset using entropy criterion: 47.23919772192646 %
Accuracy for using SMOTENC-ed dataset using gini criterion: 47.23919772192646 %
```

In [96]:
```python
print("Random Forest Classifier:")
print("Accuracy for using imbalance dataset using entropy criterion:", scores_entropy3.mean()*100, "%")
print("Accuracy for using imbalance dataset using gini criterion:", scores_gini3.mean()*100, "%")
```

```
Random Forest Classifier:
Accuracy for using imbalance dataset using entropy criterion: 28.226368070840703 %
Accuracy for using imbalance dataset using gini criterion: 28.226368070840703 %
```

In [84]:
```python
from sklearn.tree import DecisionTreeClassifier

dt1 = DecisionTreeClassifier(criterion='entropy', random_state=13519000)

dt2 = DecisionTreeClassifier(criterion='gini', random_state=13519000)
```

In [85]:
```python
scores_entropy_dt1 = cross_val_score(dt1, df3_x, df3_y, cv=10)
scores_gini_dt1 = cross_val_score(dt2, df3_x, df3_y, cv=10)
```

In [92]:
```python
print("Decision Tree Classifier:")
print("Accuracy for using SMOTEN-ed dataset using entropy criterion:", scores_entropy_dt1.mean()*100, "%")
print("Accuracy for using SMOTEN-ed dataset using gini criterion:", scores_gini_dt1.mean()*100, "%")
```

```
Decision Tree Classifier:
Accuracy for using SMOTEN-ed dataset using entropy criterion: 49.81775411662747 %
Accuracy for using SMOTEN-ed dataset using gini criterion: 49.779868763154624 %
```

In [87]:
```python
scores_entropy_dt2 = cross_val_score(dt1, df4_x, df4_y, cv=10)
scores_gini_dt2 = cross_val_score(dt2, df4_x, df4_y, cv=10)
```

In [93]:
```python
print("Decision Tree Classifier:")
print("Accuracy for using SMOTENC-ed dataset using entropy criterion:", scores_entropy_dt2.mean()*100, "%")
print("Accuracy for using SMOTENC-ed dataset using gini criterion:", scores_gini_dt2.mean()*100, "%")
```

```
Decision Tree Classifier:
Accuracy for using SMOTENC-ed dataset using entropy criterion: 47.297635260616566 %
Accuracy for using SMOTENC-ed dataset using gini criterion: 47.1994552432834 %
```

In [88]:
```python
scores_entropy_dt3 = cross_val_score(dt1, df2_x, df2_y, cv=10)
scores_gini_dt3 = cross_val_score(dt2, df2_x, df2_y, cv=10)
```

In [95]:
```python
print("Decision Tree Classifier:")
print("Accuracy for using imbalance dataset using entropy criterion:", scores_entropy_dt3.mean()*100, "%")
print("Accuracy for using imbalance dataset using gini criterion:", scores_gini_dt3.mean()*100, "%")
```

```
Decision Tree Classifier:
Accuracy for using imbalance dataset using entropy criterion: 28.409439088215382 %
Accuracy for using imbalance dataset using gini criterion: 28.380333463333457 %
```

Predict

In [105…
```python
def prediction(model,df,xcols,ycol):
    df_ = df.copy()
    df_['distance_bin'] = pd.cut(df_.distance,[0.0046, 326.593, 1100.383, 2808.036, 11761.396],labels=[0,1,2,3])
    df_['distance_bin'] = df_.distance_bin.cat.add_categories(-1)
    df_['distance_bin'] = df_.distance_bin.fillna(value=-1)
    df_2 = df_[xcols]
    df_[ycol] = model.predict(df_2)
    return df_[['id',ycol]]
```

In [98]:
```python
submission = pd.read_csv('Test.csv')
del submission['Unnamed: 0']
```

In [99]:
```python
#decision tree with SMOTEN-ed dataset and entropy criterion is chosen

#fit the model
dt1.fit(df3_x,df3_y)
```

Out[99]: DecisionTreeClassifier(criterion='entropy', random_state=13519000)

In [106…
```python
#predict
submission_result = prediction(dt1,submission,x_cols,y_col)
```

In [111…
```python
#create csv file of the prediction
submission_result.to_csv('DecisionTreeSubmission.csv')
```

In [ ]: