S11_T01_Unsupervised_Learning_Grouping

June 21, 2022

S11T01: Unsupervised Learning Grouping

Libraries

[2]:
```
#python built-in modules
import math

#Data Manipulation
import pandas as pd
import numpy as np

#Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt

#Data Modeling
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.cluster import KMeans
from kneed import KneeLocator
import scipy.cluster.hierarchy as shc
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
```

Notebook Function

[3]:
```
#1 function to the data type to another
def astype_convertion(df,cols,dtype):
    df[cols] = df[cols].astype(dtype)
```

[4]:
```
#2 function to apply to Time a cyclical encoding
def time_cyclical_encoding(df,col,sin_cos):

    hour = df[col].dt.hour*60
    minute = df[col].dt.minute

    ct = hour + minute
```

```
    if sin_cos == "sin":

        ct = np.sin(2*math.pi*ct/1440)
    else:

        ct = np.cos(2*math.pi*ct/1440)
    return ct
```

[5]:
```
#3 function to apply to day and month a cyclical encoding
def date_cyclical_encoding_sin(df,col,time,sin_cos):

    if time == "day" :
        day = df[col].dt.day-1
        if sin_cos == "sin":

            ct = np.sin(2*math.pi*day/31)

        else:

            ct = np.cos(2*math.pi*day/31)

        return ct
    elif time == "month" :
        month = df[col].dt.month-1
        if sin_cos == "sin":

            ct = np.sin(2*math.pi*month/12)

        else:

            ct = np.cos(2*math.pi*month/12)

        return ct

    else:
        print("Only Day, Month or DayOfweek")
```

[6]:
```
#4 function to apply to DayOfWeek a cyclical encoding
def dayOfWeek_cyclical_encoding(df,col,sin_cos):

    dayOfWeek = df[col]  - 1

    if sin_cos == "sin":

        ct = np.sin(2*math.pi*dayOfWeek/7)
    else:
```

```
        ct = np.cos(2*math.pi*dayOfWeek/7)
    return ct
```

[7]:
```
#6: correlation matrix

def matrix_plot (df,name):

    fig, ax = plt.subplots(figsize=(30, 30))
 # Heatmap Matrix
    sns.heatmap(df, annot=True,fmt='.2g',
                cmap=sns.color_palette("YlGnBu", as_cmap=True),
                annot_kws = {"size": 10})
    ax.set_xticklabels(ax.get_xticklabels(),rotation = 90,
                       horizontalalignment = 'right',weight='bold')
    ax.set_yticklabels(ax.get_yticklabels(),rotation = 0,
                       horizontalalignment = 'right',weight='bold')
    ax.tick_params(labelsize = 15)
    fig.suptitle(name, fontsize = 25, fontweight = "bold")
    plt.show()
```

This dataset is composed by the following variables: Year 2008 **Month** 1-12 **DayofMonth** 1-31 **DayOfWeek** 1 (Monday) - 7 (Sunday) **DepTime** actual departure time (local, hhmm) **CRSDepTime** scheduled departure time (local, hhmm) **ArrTime** actual arrival time (local, hhmm) **CRSArrTime** scheduled arrival time (local, hhmm) **UniqueCarrier** unique carrier code **FlightNum** flight number **TailNum** plane tail number: aircraft registration, unique aircraft identifier **ActualElapsedTime** in minutes **CRSElapsedTime** in minutes **AirTime** in minutes **ArrDelay** arrival delay, in minutes: A flight is counted as "on time" if it operated less than 15 minutes later the scheduled time shown in the carriers' Computerized Reservations Systems (CRS). **DepDelay** departure delay, in minutes **Origin** origin IATA airport code **Dest** destination IATA airport code **Distance** in miles **TaxiIn** taxi in time, in minutes **TaxiOut** taxi out time in minutes **Cancelled** was the flight cancelled **CancellationCode** reason for cancellation (A = carrier, B = weather, C = NAS, D = security) **Diverted** 1 = yes, 0 = no **CarrierDelay in minutes:** Carrier delay is within the control of the air carrier. Examples of occurrences that may determine carrier delay are: aircraft cleaning, aircraft damage, awaiting the arrival of connecting passengers or crew, baggage, bird strike, cargo loading, catering, computer, outage-carrier equipment, crew legality (pilot or attendant rest), damage by hazardous goods, engineering inspection, fueling, handling disabled passengers, late crew, lavatory servicing, maintenance, oversales, potable water servicing, removal of unruly passenger, slow boarding or seating, stowing carry-on baggage, weight and balance delays. **WeatherDelay in minutes:** Weather delay is caused by extreme or hazardous weather conditions that are forecasted or manifest themselves on point of departure, enroute, or on point of arrival. **NASDelay in minutes:** Delay that is within the control of the National Airspace System (NAS) may include: non-extreme weather conditions, airport operations, heavy traffic volume, air traffic control, etc. **SecurityDelay in minutes:** Security delay is caused by evacuation of a terminal or concourse, re-boarding of aircraft because of security breach, inoperative screening equipment and/or long lines in excess of 29 minutes at screening areas. **LateAircraftDelay in minutes:** Arrival delay at an airport due to the late arrival of the same aircraft at a previous airport. The ripple effect of an earlier delay at downstream airports is referred to as delay propagation.

The original dataset used in this notebook can be found in Kaggle. Use the below link to access it. Airlane Delay

I Will use a pre-procesed datased of previous work.

EDA

[8]:
```
data = pd.read_csv("/Volumes/GoogleDrive/Mi unidad/Barcelona Activa/Itinerario␣
↪Data Science/S10/DelayedFlightsPreprocesed.csv")
```

[9]:
```
df = data.copy()
```

[10]:
```
df.shape
```

[10]:
```
(1928368, 28)
```

[11]:
```
df.head().T
```

[11]:

	0	1	2	3	4
Date	2008-01-03	2008-01-03	2008-01-03	2008-01-03	2008-01-03
DayOfWeek	4	4	4	4	4
DepTime	20:03:00	07:54:00	06:28:00	18:29:00	19:40:00
CRSDepTime	19:55:00	07:35:00	06:20:00	17:55:00	19:15:00
ArrTime	22:11:00	10:02:00	08:04:00	19:59:00	21:21:00
CRSArrTime	22:25:00	10:00:00	07:50:00	19:25:00	21:10:00
UniqueCarrier	WN	WN	WN	WN	WN
FlightNum	335	3231	448	3920	378
TailNum	N712SW	N772SW	N428WN	N464WN	N726SW
ActualElapsedTime	128	128	96	90	101
CRSElapsedTime	150	145	90	90	115
AirTime	116	113	76	77	87
ArrDelay	0	0	0	1	0
DepDelay	8	19	8	34	25
Origin	IAD	IAD	IND	IND	IND
Dest	TPA	TPA	BWI	BWI	JAX
Distance	810	810	515	515	688
TaxiIn	4	5	3	3	4
TaxiOut	8	10	17	10	10
Cancelled	0	0	0	0	0
CancellationCode	N	N	N	N	N
Diverted	0	0	0	0	0
CarrierDelay	0	0	0	2	0
WeatherDelay	0	0	0	0	0
NASDelay	0	0	0	0	0
SecurityDelay	0	0	0	0	0

```
LateAircraftDelay          0          0          0         32          0
OrdinalDate           733044     733044     733044     733044     733044
```

[12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1928368 entries, 0 to 1928367
Data columns (total 28 columns):
 #   Column             Dtype
---  ------             -----
 0   Date               object
 1   DayOfWeek          int64
 2   DepTime            object
 3   CRSDepTime         object
 4   ArrTime            object
 5   CRSArrTime         object
 6   UniqueCarrier      object
 7   FlightNum          int64
 8   TailNum            object
 9   ActualElapsedTime  int64
 10  CRSElapsedTime     int64
 11  AirTime            int64
 12  ArrDelay           int64
 13  DepDelay           int64
 14  Origin             object
 15  Dest               object
 16  Distance           int64
 17  TaxiIn             int64
 18  TaxiOut            int64
 19  Cancelled          int64
 20  CancellationCode   object
 21  Diverted           int64
 22  CarrierDelay       int64
 23  WeatherDelay       int64
 24  NASDelay           int64
 25  SecurityDelay      int64
 26  LateAircraftDelay  int64
 27  OrdinalDate        int64
dtypes: int64(18), object(10)
memory usage: 411.9+ MB
```

[13]:
```
df.shape
```

[13]:
```
(1928368, 28)
```

Preproccesing Data

Time Cyclical Encoding

```
[14]: time_list = ['DepTime','CRSDepTime','ArrTime','CRSArrTime']
```

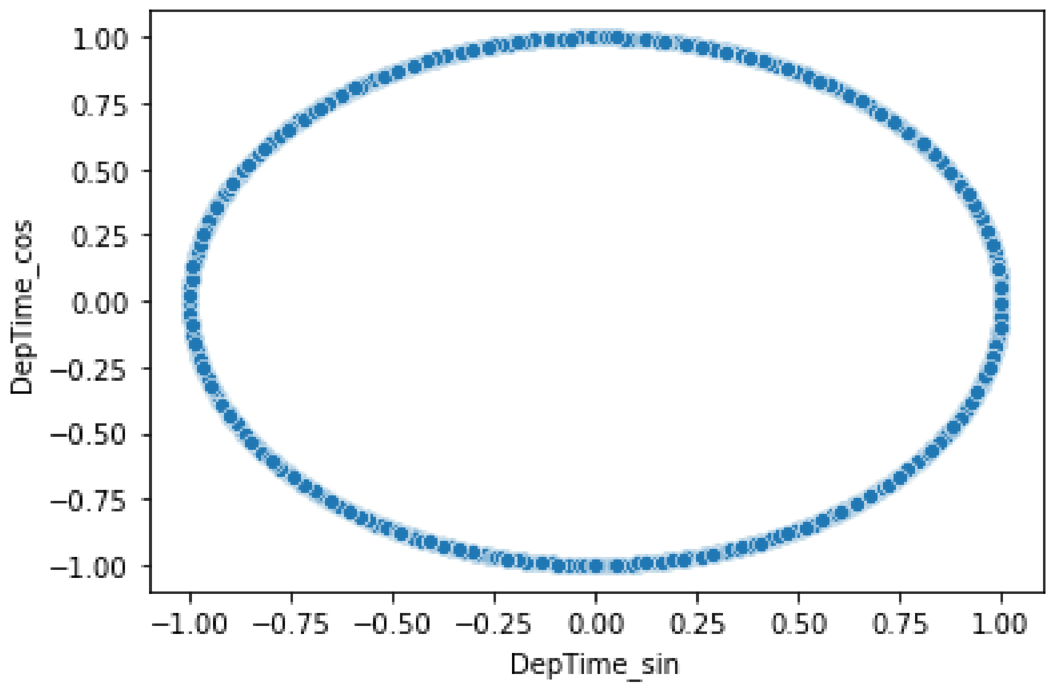
```
[15]: for t in time_list:
          df[t + "_dt"] = pd.to_datetime(df['Date'] + df[t], format='%Y-%m-%d%H:%M:
       ↪%S')
```

```
[16]: time_dt_list = ['DepTime_dt','CRSDepTime_dt','ArrTime_dt','CRSArrTime_dt']
```

```
[17]:  for dt,t in zip(time_dt_list,time_list):
             df[t + "_sin"] = time_cyclical_encoding(df,dt,"sin")
             df[t + "_cos"] = time_cyclical_encoding(df,dt,"cos")
```

```
[18]: sns.scatterplot(x="DepTime_sin",y="DepTime_cos",data=df)
```

```
[18]: <AxesSubplot:xlabel='DepTime_sin', ylabel='DepTime_cos'>
```



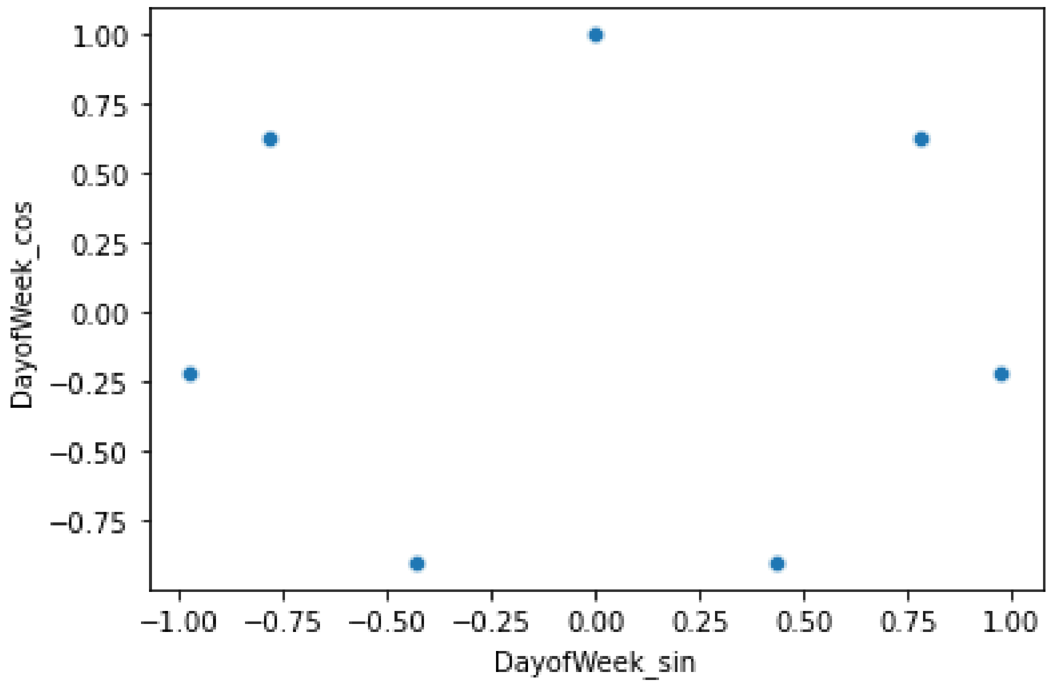
The plot show the cyclical form of the DepTime in its Sin and Cos values.

Day of Week Cyclical Encoding

```
[19]: df["DayofWeek" + "_sin"] = dayOfWeek_cyclical_encoding(df,"DayOfWeek","sin")
      df["DayofWeek" + "_cos"] = dayOfWeek_cyclical_encoding(df,"DayOfWeek","cos")
```

```
[20]: sns.scatterplot(x="DayofWeek_sin",y="DayofWeek_cos",data=df)
```

[20]: <AxesSubplot:xlabel='DayofWeek_sin', ylabel='DayofWeek_cos'>



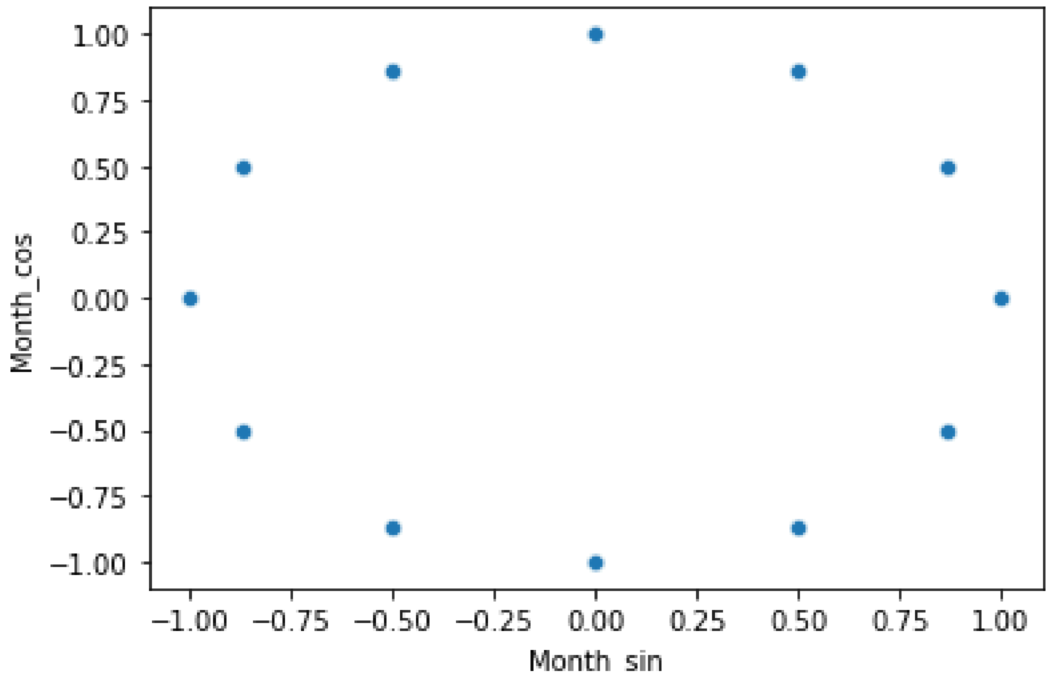
The plot show the cyclical form of the day of week in its Sin and Cos values.

Month Cyclical Encoding

[21]:
```
df["Month" + "_sin"] = date_cyclical_encoding_sin(df,"DepTime_dt","month","sin")
df["Month" + "_cos"] = date_cyclical_encoding_sin(df,"DepTime_dt","month","cos")
```

[22]:
```
sns.scatterplot(x="Month_sin",y="Month_cos",data=df)
```

[22]: <AxesSubplot:xlabel='Month_sin', ylabel='Month_cos'>



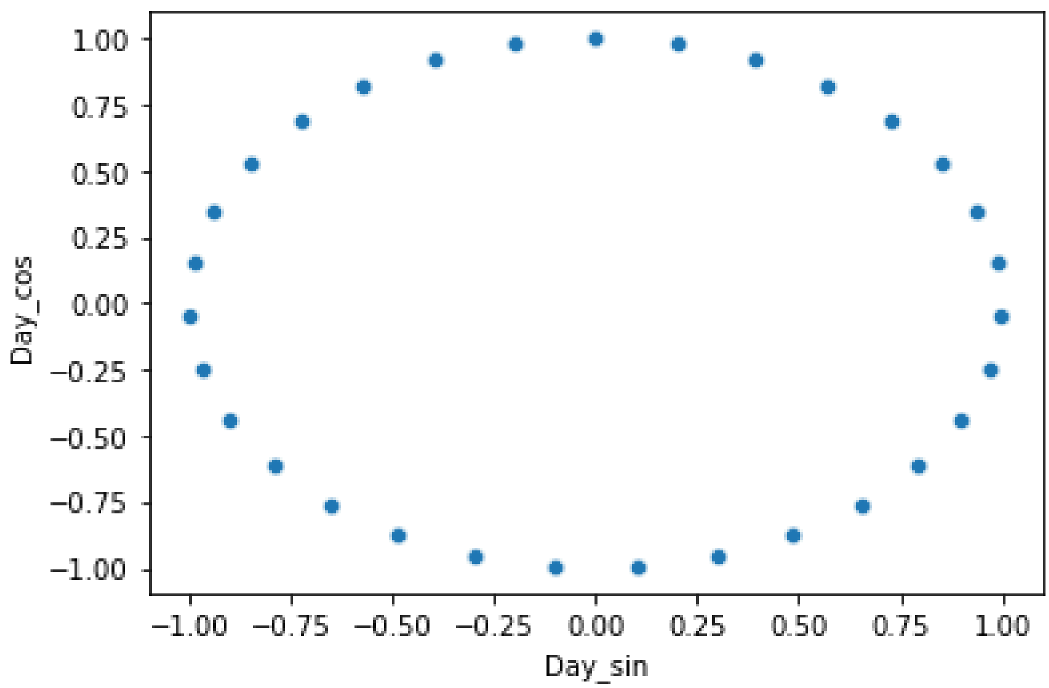
The plot show the cyclical form of the Month in its Sin and Cos values.

Day Of the Month Cyclical Encoding

[23]:
```
df["Day" + "_sin"] = date_cyclical_encoding_sin(df,'ArrTime_dt',"day","sin")
df["Day" + "_cos"] = date_cyclical_encoding_sin(df,'ArrTime_dt',"day","cos")
```

[24]:
```
sns.scatterplot(x="Day_sin",y="Day_cos",data=df)
```

[24]: <AxesSubplot:xlabel='Day_sin', ylabel='Day_cos'>



The plot show the cyclical form of the Day in its Sin and Cos values.

Drop columns

[25]:
```
#create a copy of the dataframe before drop the columns
df_copy = df.copy()
```

Some of the columns have been coded in a cyclic form so that they can be safely drop, another feature can be removed because it doesn't add more insight.

[26]:
```
drop_columns = ['Date','DayOfWeek', 'DepTime', 'CRSDepTime', 'ArrTime',
                'CRSArrTime','CancellationCode','OrdinalDate',
                'Cancelled','Diverted']
```

[27]:
```
dtime_columns = df.select_dtypes(include='datetime').columns
dtime_columns
```

[27]:
```
Index(['DepTime_dt', 'CRSDepTime_dt', 'ArrTime_dt', 'CRSArrTime_dt'],
dtype='object')
```

[28]:
```
df.drop(columns=drop_columns,inplace=True)
df.drop(columns=dtime_columns,inplace=True)
```

Change Data Type

[29]:
```
object_columns = df.select_dtypes(include = ["O"]).columns
```

[30]:
```python
astype_convertion(df,object_columns,"category")
```

[31]:
```python
astype_convertion(df,'FlightNum',"category")
```

[32]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1928368 entries, 0 to 1928367
Data columns (total 32 columns):
 #   Column              Dtype
---  ------              -----
 0   UniqueCarrier       category
 1   FlightNum           category
 2   TailNum             category
 3   ActualElapsedTime   int64
 4   CRSElapsedTime      int64
 5   AirTime             int64
 6   ArrDelay            int64
 7   DepDelay            int64
 8   Origin              category
 9   Dest                category
 10  Distance            int64
 11  TaxiIn              int64
 12  TaxiOut             int64
 13  CarrierDelay        int64
 14  WeatherDelay        int64
 15  NASDelay            int64
 16  SecurityDelay       int64
 17  LateAircraftDelay   int64
 18  DepTime_sin         float64
 19  DepTime_cos         float64
 20  CRSDepTime_sin      float64
 21  CRSDepTime_cos      float64
 22  ArrTime_sin         float64
 23  ArrTime_cos         float64
 24  CRSArrTime_sin      float64
 25  CRSArrTime_cos      float64
 26  DayofWeek_sin       float64
 27  DayofWeek_cos       float64
 28  Month_sin           float64
 29  Month_cos           float64
 30  Day_sin             float64
 31  Day_cos             float64
dtypes: category(5), float64(14), int64(13)
memory usage: 414.3 MB
```

Encoding Categorical Variables

There are categorical feautures that must be coded, these features will be encoded by Label Encoding since there is no target feature and we do not want to make the dataset larger.

[33]:
```
objects_to_encoding = df.select_dtypes(include = ["category"]).columns
objects_to_encoding
```

[33]:
```
Index(['UniqueCarrier', 'FlightNum', 'TailNum', 'Origin', 'Dest'],
dtype='object')
```

[34]:
```
#create the categorical values names for encoding

encoding_variables_names = []

for i in objects_to_encoding:
    name = "encoded_" + i
    encoding_variables_names.append(name)

encoding_variables_names
```

[34]:
```
['encoded_UniqueCarrier',
 'encoded_FlightNum',
 'encoded_TailNum',
 'encoded_Origin',
 'encoded_Dest']
```

[35]:
```
LabelEncoder = preprocessing.LabelEncoder()

for i in objects_to_encoding:
    df[i] = LabelEncoder.fit_transform(df[i])
```

[36]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1928368 entries, 0 to 1928367
Data columns (total 32 columns):
 #   Column             Dtype
---  ------             -----
 0   UniqueCarrier      int64
 1   FlightNum          int64
 2   TailNum            int64
 3   ActualElapsedTime  int64
 4   CRSElapsedTime     int64
 5   AirTime            int64
 6   ArrDelay           int64
 7   DepDelay           int64
 8   Origin             int64
 9   Dest               int64
```

```
 10  Distance           int64
 11  TaxiIn             int64
 12  TaxiOut            int64
 13  CarrierDelay       int64
 14  WeatherDelay       int64
 15  NASDelay           int64
 16  SecurityDelay      int64
 17  LateAircraftDelay  int64
 18  DepTime_sin        float64
 19  DepTime_cos        float64
 20  CRSDepTime_sin     float64
 21  CRSDepTime_cos     float64
 22  ArrTime_sin        float64
 23  ArrTime_cos        float64
 24  CRSArrTime_sin     float64
 25  CRSArrTime_cos     float64
 26  DayofWeek_sin      float64
 27  DayofWeek_cos      float64
 28  Month_sin          float64
 29  Month_cos          float64
 30  Day_sin            float64
 31  Day_cos            float64
dtypes: float64(14), int64(18)
memory usage: 470.8 MB
```

[37]:

```
df.describe().T
```

[37]:

```
                       count         mean          std        min  \
UniqueCarrier      1928368.0    11.123611     5.933309   0.000000
FlightNum          1928368.0  2176.468894  1933.658113   0.000000
TailNum            1928368.0  2710.599580  1525.180412   0.000000
ActualElapsedTime  1928368.0   133.305891    72.060116  14.000000
CRSElapsedTime     1928368.0   134.197721    71.233433 -21.000000
AirTime            1928368.0   108.277192    68.642652   0.000000
ArrDelay           1928368.0     0.630072     0.482785   0.000000
DepDelay           1928368.0    43.091598    53.265773   6.000000
Origin             1928368.0   146.495721    80.105233   0.000000
Dest               1928368.0   149.970765    80.759155   0.000000
Distance           1928368.0   764.949030   573.886107  11.000000
TaxiIn             1928368.0     6.811386     5.268054   0.000000
TaxiOut            1928368.0    18.217313    14.308382   0.000000
CarrierDelay       1928368.0    12.407419    36.204244   0.000000
WeatherDelay       1928368.0     2.395748    17.376209   0.000000
NASDelay           1928368.0     9.717681    28.143350   0.000000
SecurityDelay      1928368.0     0.058311     1.627458   0.000000
LateAircraftDelay  1928368.0    16.364624    35.920819   0.000000
DepTime_sin        1928368.0    -0.407427     0.600257  -1.000000
```

DepTime_cos	1928368.0	-0.258933	0.637690	-1.000000
CRSDepTime_sin	1928368.0	-0.369760	0.639927	-1.000000
CRSDepTime_cos	1928368.0	-0.345332	0.578375	-1.000000
ArrTime_sin	1928368.0	-0.471719	0.508052	-1.000000
ArrTime_cos	1928368.0	-0.000912	0.720669	-1.000000
CRSArrTime_sin	1928368.0	-0.485883	0.522865	-1.000000
CRSArrTime_cos	1928368.0	-0.089970	0.694576	-1.000000
DayofWeek_sin	1928368.0	0.002018	0.686000	-0.974928
DayofWeek_cos	1928368.0	-0.014619	0.727452	-0.900969
Month_sin	1928368.0	0.101658	0.660404	-1.000000
Month_cos	1928368.0	0.027914	0.743474	-1.000000
Day_sin	1928368.0	-0.002359	0.712360	-0.998717
Day_cos	1928368.0	-0.024358	0.701388	-0.994869

	25%	50%	75%	max
UniqueCarrier	6.000000	1.300000e+01	17.000000	19.000000
FlightNum	609.000000	1.538000e+03	3418.000000	7497.000000
TailNum	1415.000000	2.647000e+03	3987.000000	5359.000000
ActualElapsedTime	80.000000	1.160000e+02	165.000000	1114.000000
CRSElapsedTime	82.000000	1.160000e+02	165.000000	660.000000
AirTime	58.000000	9.000000e+01	137.000000	1091.000000
ArrDelay	0.000000	1.000000e+00	1.000000	1.000000
DepDelay	12.000000	2.400000e+01	53.000000	2467.000000
Origin	80.000000	1.550000e+02	210.000000	302.000000
Dest	80.000000	1.570000e+02	215.000000	301.000000
Distance	338.000000	6.060000e+02	997.000000	4962.000000
TaxiIn	4.000000	6.000000e+00	8.000000	240.000000
TaxiOut	10.000000	1.400000e+01	21.000000	422.000000
CarrierDelay	0.000000	0.000000e+00	10.000000	2436.000000
WeatherDelay	0.000000	0.000000e+00	0.000000	1352.000000
NASDelay	0.000000	0.000000e+00	6.000000	1357.000000
SecurityDelay	0.000000	0.000000e+00	0.000000	392.000000
LateAircraftDelay	0.000000	0.000000e+00	18.000000	1316.000000
DepTime_sin	-0.913545	-6.360782e-01	-0.013090	1.000000
DepTime_cos	-0.857167	-4.146932e-01	0.288196	1.000000
CRSDepTime_sin	-0.923880	-6.360782e-01	0.108867	1.000000
CRSDepTime_cos	-0.876727	-5.000000e-01	0.087156	1.000000
ArrTime_sin	-0.906308	-6.259235e-01	-0.160743	1.000000
ArrTime_cos	-0.740218	4.363309e-03	0.728371	1.000000
CRSArrTime_sin	-0.915311	-6.593458e-01	-0.207912	1.000000
CRSArrTime_cos	-0.782608	-1.521234e-01	0.580703	1.000000
DayofWeek_sin	-0.781831	0.000000e+00	0.781831	0.974928
DayofWeek_cos	-0.900969	-2.225209e-01	0.623490	1.000000
Month_sin	-0.500000	1.224647e-16	0.866025	1.000000
Month_cos	-0.866025	6.123234e-17	0.866025	1.000000
Day_sin	-0.724793	0.000000e+00	0.724793	0.998717
Day_cos	-0.758758	-5.064917e-02	0.688967	1.000000

PCA

[38]:
```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
scaled_df = pd.DataFrame(scaled_data,columns=df.columns)
scaled_df.head().T
```

[38]:

	0	1	2	3	4
UniqueCarrier	0.990407	0.990407	0.990407	0.990407	0.990407
FlightNum	-0.953358	0.542770	-0.894920	0.899089	-0.931121
TailNum	0.685428	0.921465	-0.500006	-0.381332	0.739847
ActualElapsedTime	-0.073631	-0.073631	-0.517705	-0.600969	-0.448319
CRSElapsedTime	0.221838	0.151646	-0.620463	-0.620463	-0.269504
AirTime	0.112507	0.068803	-0.470221	-0.455653	-0.309971
ArrDelay	-1.305077	-1.305077	-1.305077	0.766238	-1.305077
DepDelay	-0.658802	-0.452291	-0.658802	-0.170684	-0.339648
Origin	-0.143508	-0.143508	-0.081090	-0.081090	-0.081090
Dest	1.659617	1.659617	-1.262653	-1.262653	-0.012020
Distance	0.078502	0.078502	-0.435538	-0.435538	-0.134084
TaxiIn	-0.533667	-0.343844	-0.723491	-0.723491	-0.533667
TaxiOut	-0.714079	-0.574301	-0.085077	-0.574301	-0.574301
CarrierDelay	-0.342706	-0.342706	-0.342706	-0.287464	-0.342706
WeatherDelay	-0.137875	-0.137875	-0.137875	-0.137875	-0.137875
NASDelay	-0.345292	-0.345292	-0.345292	-0.345292	-0.345292
SecurityDelay	-0.035829	-0.035829	-0.035829	-0.035829	-0.035829
LateAircraftDelay	-0.455575	-0.455575	-0.455575	0.435273	-0.455575
DepTime_sin	-0.752976	2.142822	2.332289	-0.973879	-0.831112
DepTime_cos	1.207838	-0.342213	0.214938	0.603949	1.068781
CRSDepTime_sin	-0.792226	2.008156	2.134550	-0.984491	-0.901931
CRSDepTime_cos	1.428694	-0.099269	0.446382	0.559355	1.152836
ArrTime_sin	0.027251	1.897725	2.615651	-0.780393	-0.330124
ArrTime_cos	1.234865	-1.206440	-0.713401	0.689815	1.068110
CRSArrTime_sin	0.159001	1.885540	2.625713	-0.853232	-0.362823
CRSArrTime_cos	1.447331	-1.117309	-0.535260	0.651344	1.191010
DayofWeek_sin	0.629541	0.629541	0.629541	0.629541	0.629541
DayofWeek_cos	-1.218431	-1.218431	-1.218431	-1.218431	-1.218431
Month_sin	-0.153934	-0.153934	-0.153934	-0.153934	-0.153934
Month_cos	1.307492	1.307492	1.307492	1.307492	1.307492
Day_sin	0.556902	0.556902	0.556902	0.556902	0.556902
Day_cos	1.344929	1.344929	1.344929	1.344929	1.344929

[39]:
```
pca_pipe = make_pipeline(PCA(random_state=7))
```

[40]:
```
pca_pipe.fit(scaled_df)
```

[40]:
```
Pipeline(steps=[('pca', PCA(random_state=7))])
```

```
[41]: pca_model = pca_pipe.named_steps['pca']
      pca_model
```

```
[41]: PCA(random_state=7)
```

```
[42]: pc_list = []

      for i,val in enumerate(df.columns):
          pc = "PC_" + str(i+1)
          pc_list.append(pc)
```

```
[43]: pca_df = pd.DataFrame(data= pca_model.components_,
                            columns = scaled_df.columns,
                            index   = pc_list)

      pca_df.T
```

[43]:

	PC_1	PC_2	PC_3	PC_4	PC_5	PC_6 \
UniqueCarrier	-0.027080	-0.142284	0.006641	-0.075226	0.216622	-0.421120
FlightNum	-0.061380	-0.200328	0.033681	0.102887	-0.081297	-0.021494
TailNum	-0.013712	-0.052588	0.008904	0.015128	-0.113183	0.336078
ActualElapsedTime	0.082673	0.467543	-0.046094	0.052185	-0.039007	-0.036038
CRSElapsedTime	0.081185	0.466504	-0.056259	-0.019968	0.065890	0.015230
AirTime	0.075497	0.466827	-0.056736	-0.016557	0.079091	-0.001092
ArrDelay	0.076054	-0.013616	0.143962	0.435903	0.039872	-0.061914
DepDelay	0.094056	-0.008287	0.217202	0.497929	0.311808	0.065285
Origin	-0.012131	0.026592	-0.018003	-0.015209	0.218268	-0.322504
Dest	0.004152	0.038795	0.010106	-0.055169	0.162881	-0.501154
Distance	0.075865	0.464135	-0.051887	-0.042013	0.097191	0.019995
TaxiIn	0.024546	0.067766	-0.018489	0.196492	-0.278273	0.220397
TaxiOut	0.045136	0.090151	0.046852	0.269899	-0.473424	-0.257406
CarrierDelay	-0.002417	0.010585	0.152663	0.274803	0.311666	0.318656
WeatherDelay	0.014935	-0.008751	0.069074	0.154756	-0.057615	-0.024634
NASDelay	0.043681	0.041219	0.031179	0.348812	-0.451793	-0.283260
SecurityDelay	-0.003838	0.004093	0.011113	0.003618	0.003159	-0.061154
LateAircraftDelay	0.110563	-0.030809	0.133652	0.275512	0.327307	-0.093715
DepTime_sin	-0.368001	0.114922	0.262659	-0.044650	-0.022734	-0.010820
DepTime_cos	0.347355	-0.053575	0.339052	-0.125601	-0.033722	0.000200
CRSDepTime_sin	-0.398102	0.115177	0.183988	0.021554	-0.004103	-0.003718
CRSDepTime_cos	0.299573	-0.039203	0.370113	-0.223434	-0.079468	0.009953
ArrTime_sin	-0.151147	0.096287	0.526210	-0.131628	-0.059628	-0.021721
ArrTime_cos	0.426468	-0.017046	0.107415	-0.049243	-0.028178	0.000099
CRSArrTime_sin	-0.250421	0.093302	0.454121	-0.121524	-0.055947	-0.004969
CRSArrTime_cos	0.407812	-0.005626	0.140536	-0.156738	-0.042520	0.014525
DayofWeek_sin	0.001855	-0.003580	0.007275	-0.004210	-0.054910	-0.085090
DayofWeek_cos	-0.004100	0.001720	0.002912	0.011252	0.008668	0.047882
Month_sin	0.007941	0.001716	0.018050	0.003033	0.024005	-0.141511

Month_cos	-0.021871	-0.006957	-0.007153	0.032969	0.057748	-0.058390
Day_sin	0.002493	-0.000986	-0.000204	0.001309	-0.039289	-0.008422
Day_cos	0.000379	-0.000631	0.003223	0.012296	0.003081	-0.018459

	PC_7	PC_8	PC_9	PC_10	…	PC_23 \
UniqueCarrier	0.101767	0.083113	0.335346	0.065444	…	-0.000565
FlightNum	0.174666	0.132739	0.317568	-0.021219	…	-0.022950
TailNum	0.176859	0.212828	0.271977	-0.054076	…	0.003620
ActualElapsedTime	0.040651	0.033650	0.052970	-0.002774	…	0.008494
CRSElapsedTime	0.035159	0.031889	0.062028	-0.006580	…	0.003752
AirTime	0.036996	0.030541	0.062844	-0.007502	…	0.007397
ArrDelay	-0.016494	-0.019203	-0.098908	-0.008043	…	-0.000581
DepDelay	0.040597	-0.004004	0.034822	0.016160	…	-0.008310
Origin	-0.368889	-0.085872	0.431319	0.124057	…	0.017312
Dest	0.364725	0.140719	-0.180670	-0.065187	…	-0.005114
Distance	0.025067	0.022531	0.071508	-0.000921	…	0.003629
TaxiIn	-0.421113	-0.088717	0.143745	0.063102	…	0.015227
TaxiOut	0.182288	0.055614	-0.087642	-0.001214	…	0.001688
CarrierDelay	0.433682	-0.041603	-0.025573	0.037664	…	0.007744
WeatherDelay	0.069346	0.163273	0.477144	0.077203	…	0.007328
NASDelay	0.002173	-0.011690	0.003951	0.019151	…	-0.010385
SecurityDelay	-0.025774	-0.005586	-0.000601	0.047108	…	0.000950
LateAircraftDelay	-0.399708	-0.027643	-0.175051	-0.061580	…	-0.009918
DepTime_sin	-0.050063	-0.008795	-0.002197	0.002738	…	-0.563176
DepTime_cos	-0.007541	0.011897	0.004969	-0.002363	…	-0.159990
CRSDepTime_sin	-0.025948	-0.007859	0.002879	0.004705	…	-0.460184
CRSDepTime_cos	0.013455	0.014437	0.009058	0.000446	…	-0.256105
ArrTime_sin	-0.054893	-0.002625	0.001937	0.000071	…	0.403668
ArrTime_cos	-0.008574	0.006924	0.034508	0.004207	…	-0.121727
CRSArrTime_sin	-0.015616	0.001699	0.002039	0.001026	…	0.438707
CRSArrTime_cos	-0.004679	0.008232	0.037503	0.004302	…	-0.089321
DayofWeek_sin	0.002144	0.019556	-0.208104	-0.247919	…	0.002856
DayofWeek_cos	-0.021523	0.133046	0.240811	-0.547885	…	0.000724
Month_sin	0.007287	-0.480514	-0.075598	-0.047389	…	-0.001181
Month_cos	-0.238814	0.476224	-0.160851	-0.229447	…	0.003540
Day_sin	0.153541	-0.595254	0.193098	-0.109920	…	-0.000111
Day_cos	-0.012615	-0.167433	0.084339	-0.725149	…	-0.001293

	PC_24	PC_25	PC_26	PC_27	PC_28	PC_29 \
UniqueCarrier	-0.000998	-0.001167	-0.016242	0.001900	-0.004652	-0.000073
FlightNum	-0.007044	-0.000728	0.003898	0.004782	0.017726	-0.002093
TailNum	-0.006641	0.001369	-0.002480	-0.000013	-0.008315	0.000146
ActualElapsedTime	0.079407	-0.008002	0.340963	-0.044052	-0.148469	-0.001666
CRSElapsedTime	0.040370	-0.021370	-0.401893	0.029479	-0.473117	-0.006725
AirTime	0.080940	-0.009782	0.389655	-0.049138	-0.160546	-0.002507
ArrDelay	-0.021479	-0.035566	-0.073312	0.015879	-0.013251	-0.004951
DepDelay	-0.008643	0.028302	-0.410350	0.110144	-0.191641	-0.019970

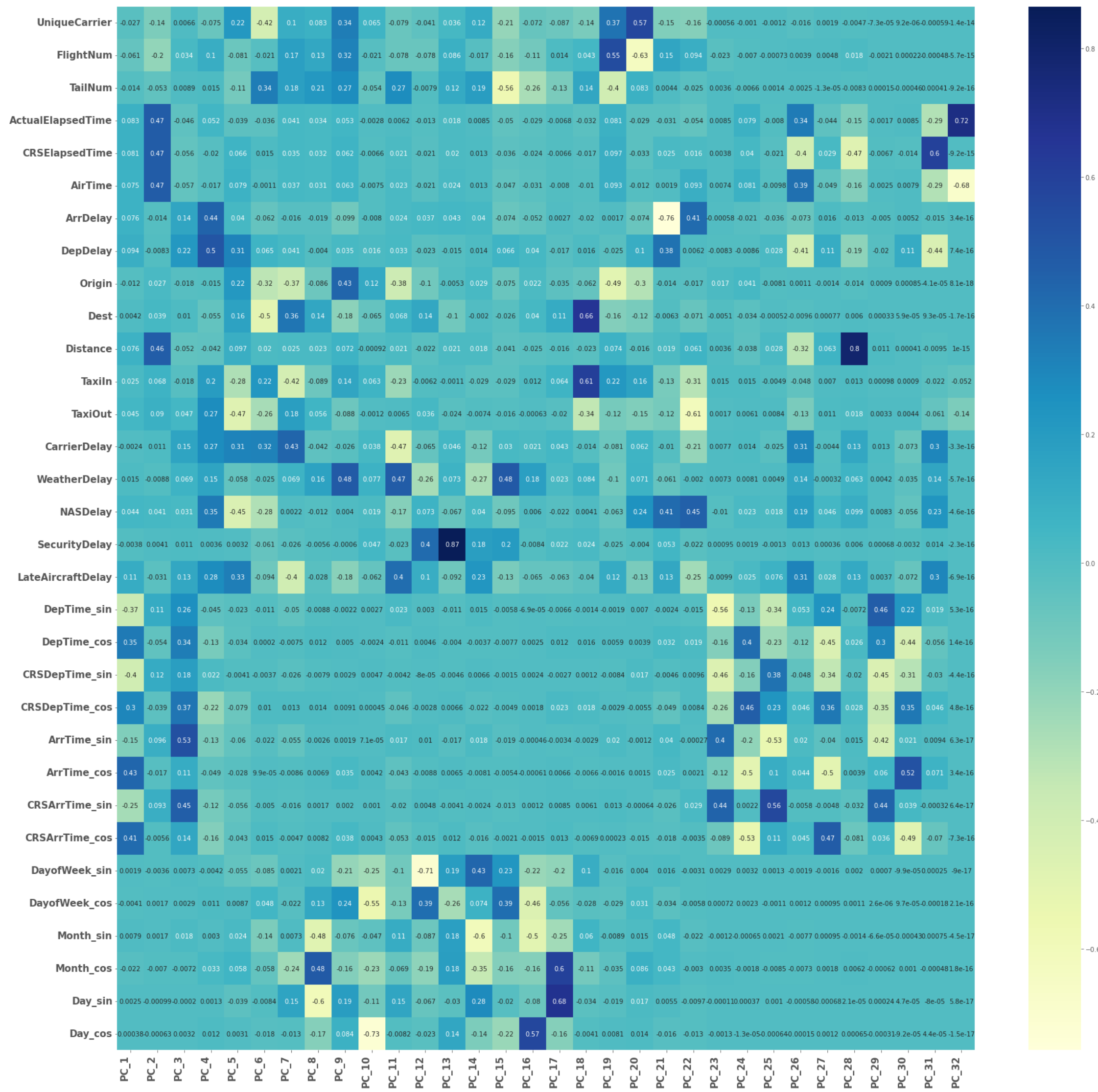
Origin	0.041304	-0.008139	0.001071	-0.001358	-0.014242	0.000895
Dest	-0.034152	-0.000520	-0.009556	0.000769	0.005954	0.000327
Distance	-0.038455	0.027796	-0.323322	0.063033	0.795475	0.010807
TaxiIn	0.014979	-0.004912	-0.048222	0.007039	0.012654	0.000977
TaxiOut	0.006093	0.008433	-0.134407	0.011289	0.017821	0.003276
CarrierDelay	0.014350	-0.024906	0.306529	-0.004410	0.131985	0.012589
WeatherDelay	0.008138	0.004871	0.139670	-0.000323	0.062973	0.004199
NASDelay	0.023286	0.017652	0.187831	0.046175	0.098787	0.008285
SecurityDelay	0.001891	-0.001328	0.013223	0.000357	0.006044	0.000680
LateAircraftDelay	0.024945	0.076020	0.309985	0.027613	0.128782	0.003688
DepTime_sin	-0.134985	-0.339394	0.053221	0.239665	-0.007242	0.462863
DepTime_cos	0.396424	-0.226797	-0.120684	-0.452425	0.025514	0.302933
CRSDepTime_sin	-0.155944	0.380855	-0.048449	-0.338332	-0.019910	-0.447098
CRSDepTime_cos	0.455591	0.232801	0.045592	0.356520	0.027885	-0.352844
ArrTime_sin	-0.196479	-0.534355	0.019828	-0.039889	0.015341	-0.416441
ArrTime_cos	-0.498589	0.103104	0.043634	-0.497886	0.003856	0.059929
CRSArrTime_sin	0.002248	0.562494	-0.005792	-0.004752	-0.031862	0.436340
CRSArrTime_cos	-0.533250	0.105471	0.045186	0.471257	-0.081143	0.036267
DayofWeek_sin	0.003160	0.001304	-0.001945	-0.001602	0.001970	0.000703
DayofWeek_cos	0.002293	-0.001136	0.001186	0.000954	0.001142	0.000003
Month_sin	-0.000646	0.002105	-0.007713	0.000953	-0.001356	-0.000066
Month_cos	-0.001767	-0.008544	-0.007325	0.001751	0.006221	-0.000622
Day_sin	0.000370	0.001013	-0.000584	-0.000684	0.000021	0.000243
Day_cos	-0.000013	-0.000645	-0.000145	0.001180	0.000650	-0.000309

	PC_30	PC_31	PC_32
UniqueCarrier	0.000009	-0.000589	-1.363250e-14
FlightNum	0.000224	-0.000479	-5.739295e-15
TailNum	-0.000461	0.000412	-9.166983e-16
ActualElapsedTime	0.008505	-0.293177	7.157167e-01
CRSElapsedTime	-0.014142	0.598202	-9.213198e-15
AirTime	0.007937	-0.293283	-6.817737e-01
ArrDelay	0.005209	-0.015040	3.396115e-16
DepDelay	0.114120	-0.441632	7.359183e-16
Origin	0.000847	-0.000041	8.078986e-18
Dest	0.000059	0.000093	-1.663552e-16
Distance	0.000406	-0.009542	9.965257e-16
TaxiIn	0.000899	-0.022146	-5.232346e-02
TaxiOut	0.004425	-0.061360	-1.421140e-01
CarrierDelay	-0.073024	0.300688	-3.278480e-16
WeatherDelay	-0.035408	0.143940	-5.748015e-16
NASDelay	-0.055937	0.232017	-4.617829e-16
SecurityDelay	-0.003172	0.013537	-2.318497e-16
LateAircraftDelay	-0.071816	0.298139	-6.885666e-16
DepTime_sin	0.224487	0.019321	5.320727e-16
DepTime_cos	-0.439680	-0.055754	1.351169e-16
CRSDepTime_sin	-0.305620	-0.029955	-4.366957e-16

```
CRSDepTime_cos     0.353510  0.046256  4.830227e-16
ArrTime_sin        0.020643  0.009428  6.270286e-17
ArrTime_cos        0.516157  0.071337  3.443779e-16
CRSArrTime_sin     0.038685 -0.000323  6.372919e-17
CRSArrTime_cos    -0.491205 -0.069584 -7.344244e-16
DayofWeek_sin     -0.000099  0.000248 -8.954884e-17
DayofWeek_cos      0.000097 -0.000181  2.072070e-16
Month_sin         -0.000430  0.000749 -4.478376e-17
Month_cos          0.000999 -0.000481  1.830967e-16
Day_sin            0.000047 -0.000080  5.784894e-17
Day_cos           -0.000092  0.000044 -1.525802e-17

[32 rows x 32 columns]
```

[44]:
```
plt.style.use('ggplot')
matrix_plot (pca_df.T,"The influence of the variables by principal component.")
```

The influence of the variables by principal component.

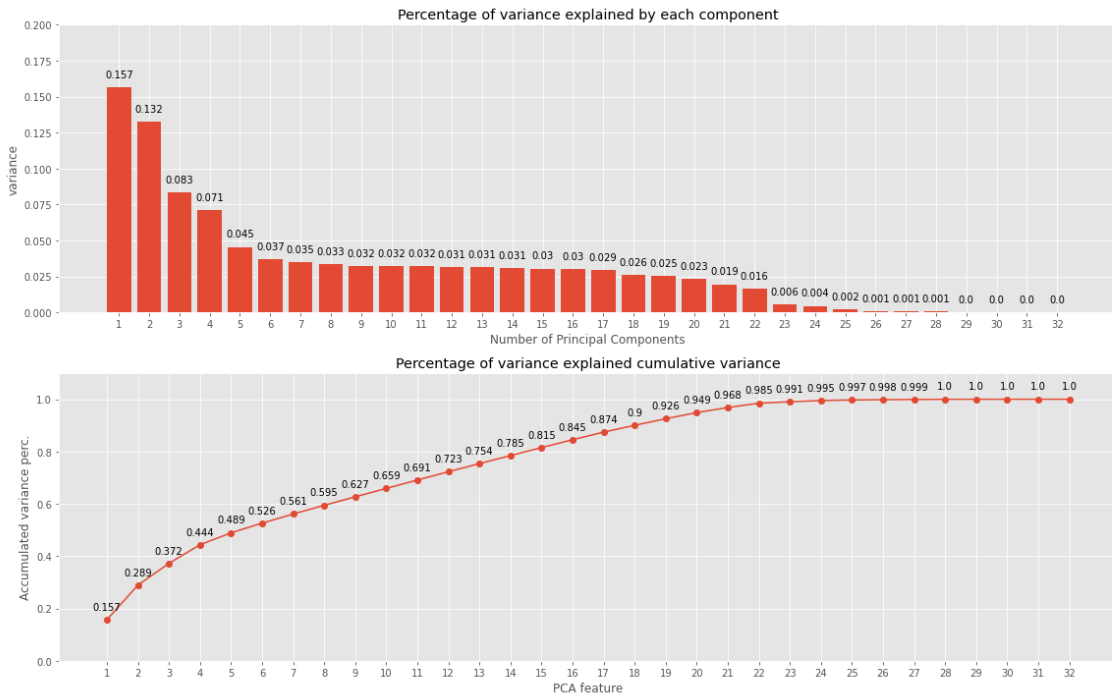


The matrix shows that the time features have more weight in the for PC_1 and the rest have a similar impact.

[45]:
```
plt.style.use('ggplot')

pc_n = np.arange(pca_model.n_components_) + 1
prop_varianza_acum = pca_model.explained_variance_ratio_.cumsum()
y_max = round(pca_model.explained_variance_ratio_.max()+.05,1)

fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(16, 10))
```

```
axes[0].bar(
        x      = pc_n,
        height = pca_model.explained_variance_ratio_)


axes[0].set_xlabel('Number of Principal Components')
axes[0].set_ylabel('variance')
axes[0].set_title('Percentage of variance explained by each component')
axes[0].set_xticks(pc_n)
axes[0].set_ylim(0, y_max)

for x, y in zip(pc_n, pca_model.explained_variance_ratio_):
    label = round(y, 3)
    axes[0].annotate(
        label,
        (x,y),
        textcoords="offset points",
        xytext=(0,10),
        ha='center'
    )



axes[1].plot(
np.arange(len(df.columns)) + 1,
prop_varianza_acum,
marker='o')


for x, y in zip(pc_n, prop_varianza_acum):
         label = round(y, 3)
         axes[1].annotate(
             label,
             (x, y),
             textcoords="offset points",
             xytext=(0, 10),
             ha='center'
         )

axes[1].set_ylim(0, 1.1)
axes[1].set_xticks(pc_n)
axes[1].set_title('Percentage of variance explained cumulative variance')
axes[1].set_xlabel('PCA feature')
axes[1].set_ylabel('Accumulated variance perc.')

plt.xticks(pc_n)
```

```
plt.tight_layout()
plt.grid(True)
plt.show()
```



The first 6 PCs have more than 50% of the variance and then have a similar distribution until PC_21 where 95% of the variance is reached, maybe be can use only 6 PC for the K-Means but we need to verify it later.

[46]:
```
pca_clustering = PCA(n_components=21, random_state=42)

clustering_array = pca_clustering.fit_transform(scaled_df)
```

[47]:
```
pc21_list = []
for i,val in enumerate( np.arange(1, 22)):
    pc = "PC_" + str(i+1)
    pc21_list.append(pc)
```

[48]:
```
pca_PC21_df = pd.DataFrame(data= clustering_array , columns=pc21_list)
pca_PC21_df.T
```

[48]:

	0	1	2	3	4	5	6 \
PC_1	2.275411	-3.777021	-3.477211	1.779164	2.133572	2.890394	-3.578725
PC_2	-0.172807	0.681180	0.005890	-1.941730	-1.015811	2.295075	1.000346
PC_3	0.502399	1.939654	3.266709	-0.477973	-0.027515	-0.306244	2.447513
PC_4	-2.542552	-1.671754	-2.201118	-0.123806	-2.006273	-0.157187	-1.929189

```
PC_5   0.392693  0.247743 -0.443187  0.550135  0.182517  1.628193 -0.255723
PC_6  -0.953999 -1.032474 -0.271197 -0.146222 -0.126387 -0.231073 -0.650887
PC_7   0.667881  0.597577 -0.898689 -0.534643  0.068138  0.052512 -0.184196
PC_8   0.459871  0.567266 -0.329147 -0.081637  0.187041  0.365349  0.127676
PC_9  -0.296550  0.064770 -0.420802 -0.044159 -0.059421 -0.001330 -0.331270
PC_10 -0.931066 -0.954951 -0.660970 -0.783690 -0.808033 -0.946140 -0.809849
PC_11  0.225161  0.513647 -0.026560  0.079336  0.124017  0.860016  0.398426
PC_12 -1.081329 -1.104988 -1.315744 -1.413867 -1.291362 -1.265403 -1.128205
PC_13  0.716965  0.730621  0.676350  1.002029  0.862293  1.003043  0.667334
PC_14 -0.121273  0.086827 -0.192303 -0.073862 -0.116523  0.378930  0.042304
PC_15 -1.251000 -1.658658 -0.500518 -0.971566 -1.144285 -1.806145 -1.210294
PC_16  0.944581  0.737822  1.205812  0.824023  0.915313  0.512673  0.947933
PC_17  1.009766  0.886981  0.737440  0.744822  0.798597  0.619301  0.758916
PC_18  0.901887  1.059212 -1.465038 -1.294682 -0.210811 -0.425760 -0.136122
PC_19 -0.595676  0.163025  0.068771  1.064564 -0.542261  0.165740 -0.486181
PC_20  1.108194  0.320153  1.341645 -0.004533  1.336976  0.913743  1.342388
PC_21  0.472996  0.747329  0.446665 -0.500108  0.594517 -0.402372  0.349313

              7         8         9  …   1928358   1928359   1928360  \
PC_1   1.076130 -3.994953  0.039042  … -2.086043  0.101677  0.395328
PC_2  -0.529541 -1.439652  2.462341  … -0.934171 -0.694163 -0.649398
PC_3  -1.095483  1.052571 -2.388028  … -0.974118 -1.597091 -2.086929
PC_4   1.135676 -1.383171 -0.765786  …  0.858832  0.799358 -0.682656
PC_5   1.937012  0.129607  1.279193  … -0.334711  0.307826 -0.611375
PC_6  -0.920376 -1.029287 -1.063508  …  2.050316 -0.897170  2.015988
PC_7  -0.802575 -0.543823 -0.014056  … -1.085722 -0.130781 -0.862285
PC_8  -0.220344 -0.288309 -0.002191  …  0.361821  0.278059  0.422934
PC_9  -0.889171 -0.665529 -0.351092  …  0.223879 -0.274299  0.815246
PC_10 -0.850792 -0.716062 -0.790226  …  0.890878  1.021787  1.000935
PC_11  0.579802 -0.220306  0.028095  …  0.877511 -0.451269  0.187082
PC_12 -1.051529 -1.145387 -1.254114  …  0.675710  0.846974  0.435961
PC_13  0.578965  0.496867  0.652031  … -0.088859 -0.576487 -0.063442
PC_14  0.149502 -0.361470 -0.245583  …  0.294124 -0.249795 -0.031714
PC_15 -0.657108 -0.194311 -0.479823  … -0.994887  0.132170 -0.718841
PC_16  1.094650  1.396589  1.209996  … -0.388409  0.349402 -0.221401
PC_17  0.865144  1.035355  0.997240  …  1.635074  2.150321  1.730517
PC_18 -0.073940  0.098233  0.321417  … -0.704869 -0.447630 -0.769057
PC_19  0.682138  0.547240  0.861065  … -0.886343 -1.506266 -0.950125
PC_20  0.671030  0.715792  0.947433  …  0.123005 -1.090547  0.063311
PC_21 -0.146695  0.582916  0.636962  … -0.656659 -0.654276  0.647679

        1928361   1928362   1928363   1928364   1928365   1928366   1928367
PC_1  -3.557138  0.115496 -1.147780 -3.084274 -2.970038 -1.979045 -2.623612
PC_2   0.207100 -2.462140  0.504526  0.698802  1.113943 -0.261187  0.390798
PC_3   1.902111 -1.514621 -1.650896  3.723065  1.663712 -1.721465 -1.332070
PC_4   0.779735  0.530019  0.987514  0.907405  1.718672 -0.255153 -0.603554
PC_5  -0.563584 -0.274348 -0.105918 -1.868415 -0.600769 -0.752367 -0.191138
```

```
PC_6   1.348429  2.138103  1.719085  0.439026  0.971997  1.953672  1.636630
PC_7   1.163128 -1.095754 -1.144147 -1.982425 -0.392666 -1.301819 -1.316509
PC_8   0.774398  0.258699  0.390703  0.306152  0.743051  0.120781  0.301709
PC_9  -0.905277  0.087344  0.704109  1.951179 -0.991387  0.328363  1.159649
PC_10  0.659758  0.914291  0.978863  1.570265  0.535439  1.041928  1.106365
PC_11  0.610113  0.581897  0.546005  0.606771  2.108219  0.077932  0.064476
PC_12  1.077844  0.665361  0.527605 -0.253675  1.239278  0.533155  0.364267
PC_13 -0.340655 -0.101071 -0.025021 -0.227039 -0.491817 -0.241715 -0.118566
PC_14 -0.041332  0.097180  0.302958 -1.136998  0.650428 -0.195276  0.101757
PC_15 -0.764643 -0.654949 -1.113293  1.668886 -1.163731 -0.124918 -0.817760
PC_16 -0.171964 -0.233571 -0.430060  0.915041 -0.425306  0.057121 -0.212469
PC_17  2.077976  1.765683  1.598502  2.092365  1.716953  1.887663  1.631274
PC_18  0.661738 -0.552992 -0.965973 -0.583969 -0.345752 -0.390615 -0.897697
PC_19 -0.605175 -0.864484 -0.981029 -0.762716 -0.020480 -0.116429 -1.309146
PC_20  0.458286  0.203882 -0.247027  0.011570  0.141716  0.300714 -0.197214
PC_21 -0.500606 -0.636158 -0.721117 -0.898580  0.002542  0.574980  0.682379

[21 rows x 1928368 columns]
```

[49]:
```python
print("original shape:    ", df.shape)
print("transformed shape:", pca_PC21_df.shape)
```

```
original shape:     (1928368, 32)
transformed shape: (1928368, 21)
```

The PCA helps to reduce the dimension of the DataSet to 21 features without loosing important insights.

Level 1

Exercise 1

Group the different flights using K-means algorithm

[50]:
```python
#I know this dataset from previous exercise so I will sample it to 5%
sample_df = pca_PC21_df.sample(frac=0.05, random_state=7)

print("the shape of the sample DF is {}".format(sample_df.shape))
sample_df.head().T
```

```
the shape of the sample DF is (96418, 21)
```

[50]:
```
        258182    437414    1499466   1549589   23091
PC_1   1.921690  0.794537  2.724801  1.673346 -0.669639
PC_2  -0.419542 -3.165526  2.304652 -1.740521 -1.346261
PC_3  -0.627985 -1.200955  2.831202 -0.756361 -1.946598
PC_4  -1.643473 -1.165235 -0.673770 -0.705609 -0.808675
PC_5  -0.187700 -0.061525 -0.016147 -1.225132  0.362436
PC_6   0.636896 -1.203451  1.142069  0.674653  0.139501
```

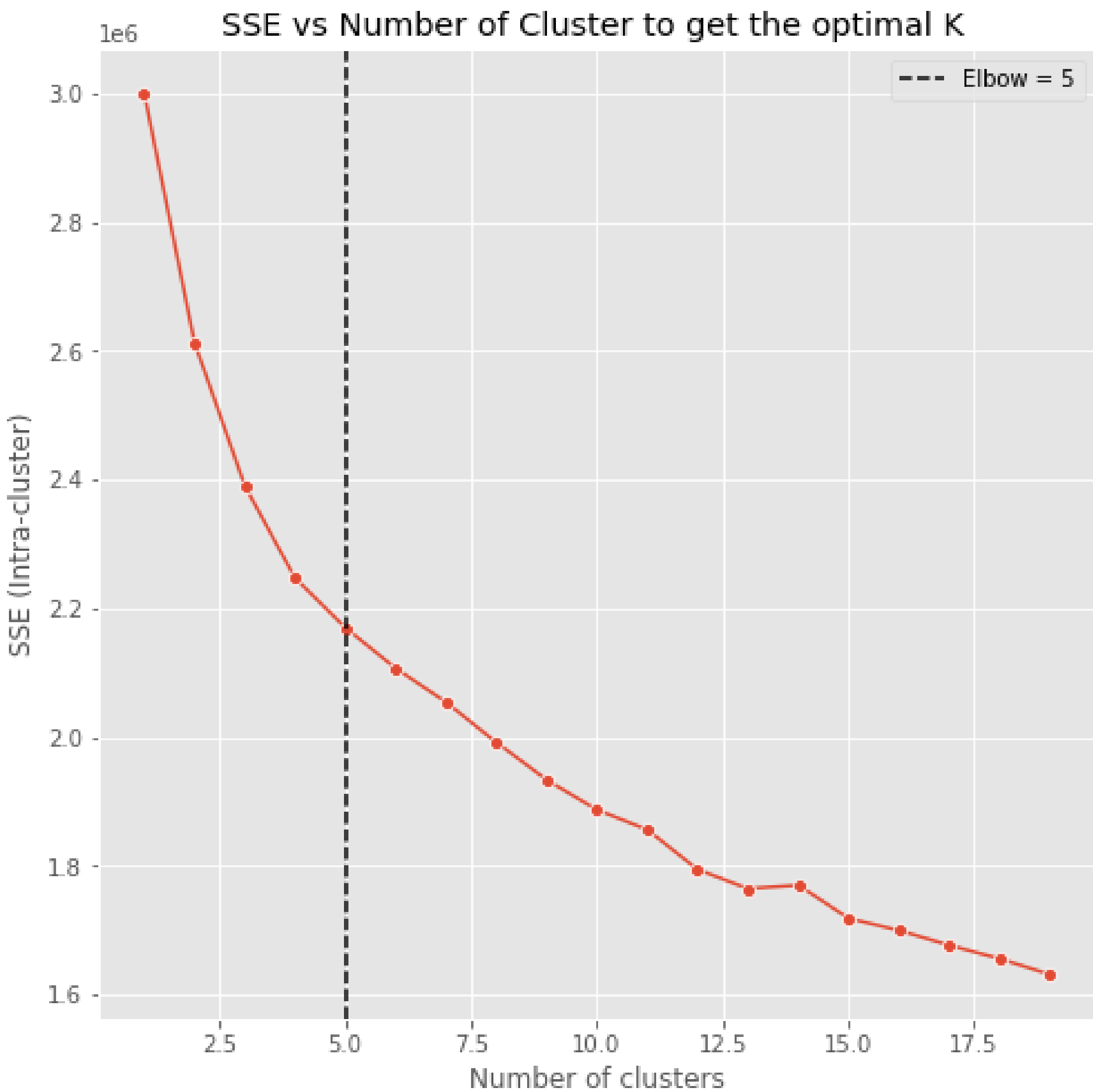
```
PC_7  -0.574910  1.086485 -1.379043 -0.932403 -1.213489
PC_8  -0.642948 -0.370764  0.502022  0.862362  0.864232
PC_9   1.319948 -0.113368 -0.694290  1.646801 -0.312376
PC_10  0.273544 -0.015163  0.756859  1.056357  0.724662
PC_11  0.103826 -0.029284  0.112497 -1.543417 -0.614565
PC_12  0.200860 -1.377873  0.316913 -1.338791 -0.444134
PC_13  0.305960  0.466993 -0.186066  0.081024  0.479591
PC_14 -0.825791 -0.069224  0.307157  1.754712 -1.356400
PC_15 -1.269450  0.048120 -0.060711 -0.066255 -0.184937
PC_16 -0.419026 -1.444978  2.082991 -0.405890  1.083830
PC_17  1.441208  0.879181 -0.722602  0.804527 -0.043071
PC_18 -1.249324  0.694426 -0.359201  2.059185 -1.071290
PC_19 -0.889164  0.178956  0.010769  1.123853  0.936937
PC_20  0.975333 -0.928283 -0.961088 -1.026354  0.851064
PC_21  0.443202  0.969344 -0.293592  0.503962  0.564795
```

Elbow method

[51]:
```
#get the optimal number of cluster
sse = []
for k in range(1, 20):
    kmeans = KMeans(n_clusters=k, random_state=7)
    kmeans.fit(sample_df)
    sse.append(kmeans.inertia_)
```

[52]:
```
kl = KneeLocator(
    range(1, 20), sse, curve="convex", direction="decreasing"
    )

elbow = kl.elbow
```

[53]:
```
fig, ax = plt.subplots(figsize=(8, 8))

sns.lineplot(x =range(1, 20), y=sse, marker='o')

ax.set_title("SSE vs Number of Cluster to get the optimal K ")
ax.set_xlabel('Number of clusters')
ax.set_ylabel('SSE (Intra-cluster)')
ax.axvline(x=elbow, c = 'black', linestyle='--', label=f'Elbow = {elbow}')
ax.legend()
plt.show()
```



The elbow method computes 5 clusters.

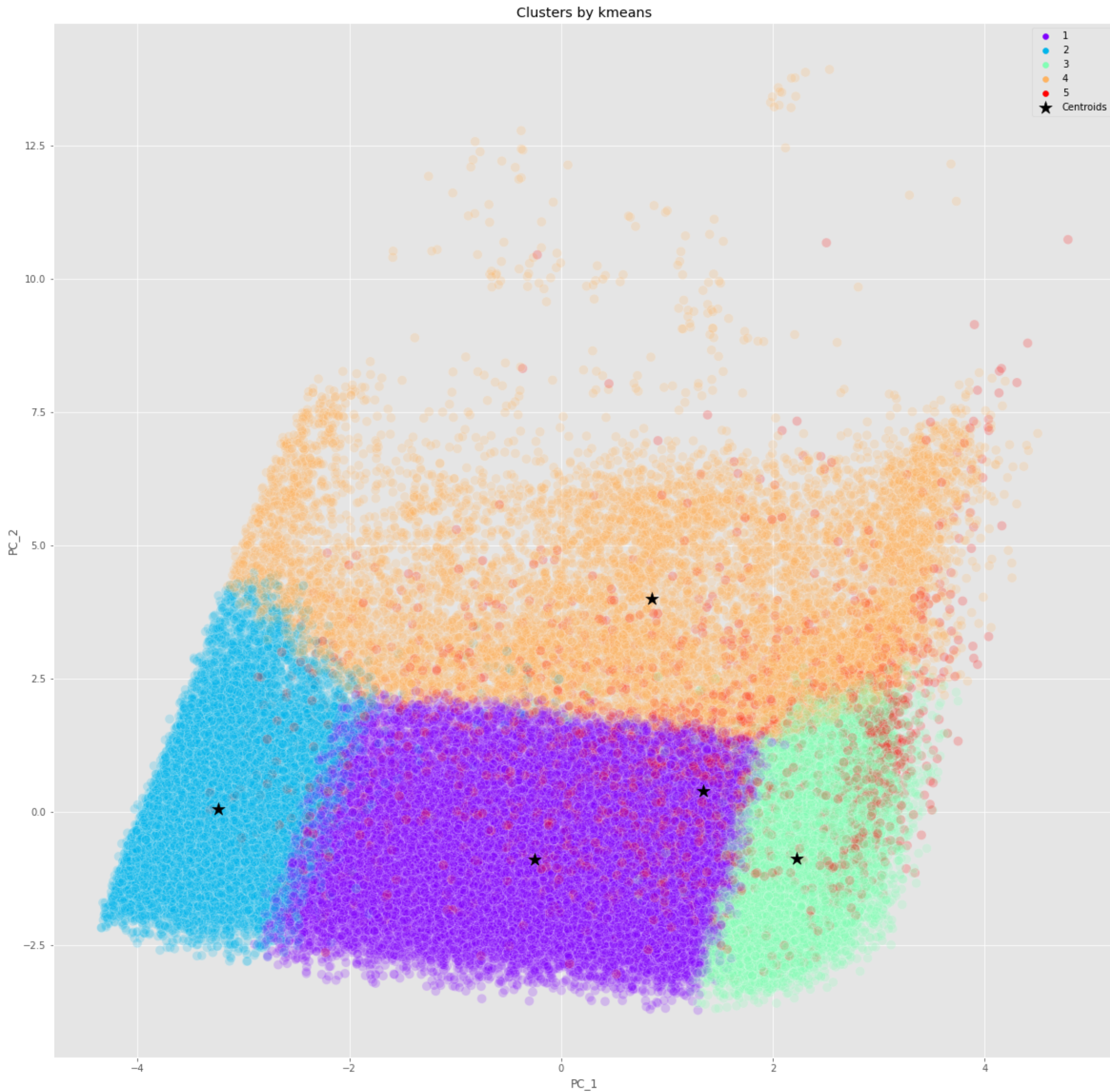
KMeans

[54]:
```
#initializing the model with 5 clusters
kmeans = KMeans(n_clusters=5, random_state=7)
kmeans.fit(sample_df)
```

[54]: KMeans(n_clusters=5, random_state=7)

[55]:
```
fig, ax = plt.subplots(figsize=(20, 20))

sns.scatterplot(x="PC_1", y="PC_2",
                data=sample_df,
```

```
                hue=kmeans.labels_+1,
                alpha=.2,
                palette = 'rainbow',
                s=100
               )
ax.set_title("Clusters by kmeans")
ax.set_xlabel('PC_1')
ax.set_ylabel('PC_2')
ax.scatter(
    x = kmeans.cluster_centers_[:, 0],
    y = kmeans.cluster_centers_[:, 1],
    c = 'black',
    s = 200,
    marker = '*',
    label  = 'Centroids'
)
ax.legend()
plt.show()
```



The variance for the 2 PC is 28.9% the first 3 cluster have a more defined shape, cluster 4 and 5 are more dispersed.

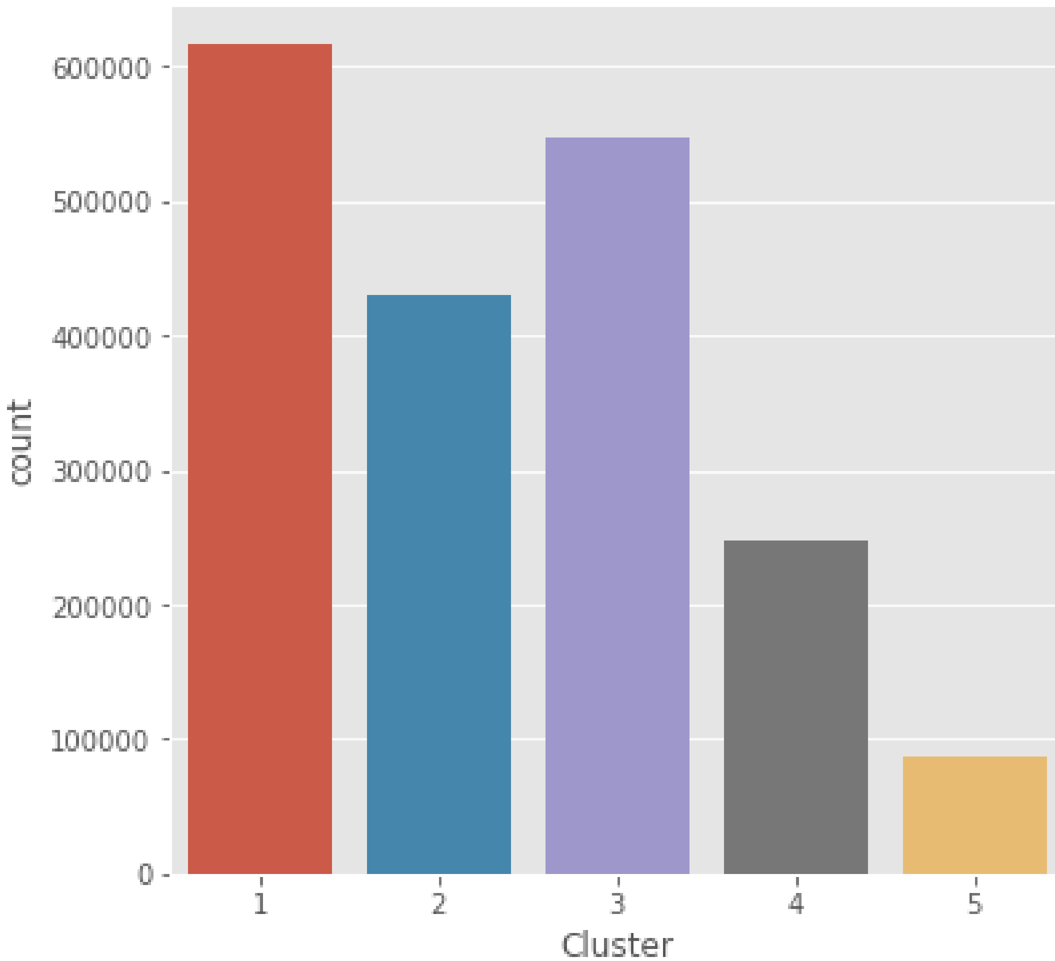
[56]:
```python
#add observation cluster to the dataframe
clustering = kmeans.predict(pca_PC21_df ) + 1

kmeans_classified = scaled_df.copy()
kmeans_classified['Cluster'] = clustering
kmeans_classified.head().T
```

[56]:

	0	1	2	3	4
UniqueCarrier	0.990407	0.990407	0.990407	0.990407	0.990407
FlightNum	-0.953358	0.542770	-0.894920	0.899089	-0.931121
TailNum	0.685428	0.921465	-0.500006	-0.381332	0.739847

```
ActualElapsedTime -0.073631 -0.073631 -0.517705 -0.600969 -0.448319
CRSElapsedTime     0.221838  0.151646 -0.620463 -0.620463 -0.269504
AirTime            0.112507  0.068803 -0.470221 -0.455653 -0.309971
ArrDelay          -1.305077 -1.305077 -1.305077  0.766238 -1.305077
DepDelay          -0.658802 -0.452291 -0.658802 -0.170684 -0.339648
Origin            -0.143508 -0.143508 -0.081090 -0.081090 -0.081090
Dest               1.659617  1.659617 -1.262653 -1.262653 -0.012020
Distance           0.078502  0.078502 -0.435538 -0.435538 -0.134084
TaxiIn            -0.533667 -0.343844 -0.723491 -0.723491 -0.533667
TaxiOut           -0.714079 -0.574301 -0.085077 -0.574301 -0.574301
CarrierDelay      -0.342706 -0.342706 -0.342706 -0.287464 -0.342706
WeatherDelay      -0.137875 -0.137875 -0.137875 -0.137875 -0.137875
NASDelay          -0.345292 -0.345292 -0.345292 -0.345292 -0.345292
SecurityDelay     -0.035829 -0.035829 -0.035829 -0.035829 -0.035829
LateAircraftDelay -0.455575 -0.455575 -0.455575  0.435273 -0.455575
DepTime_sin       -0.752976  2.142822  2.332289 -0.973879 -0.831112
DepTime_cos        1.207838 -0.342213  0.214938  0.603949  1.068781
CRSDepTime_sin    -0.792226  2.008156  2.134550 -0.984491 -0.901931
CRSDepTime_cos     1.428694 -0.099269  0.446382  0.559355  1.152836
ArrTime_sin        0.027251  1.897725  2.615651 -0.780393 -0.330124
ArrTime_cos        1.234865 -1.206440 -0.713401  0.689815  1.068110
CRSArrTime_sin     0.159001  1.885540  2.625713 -0.853232 -0.362823
CRSArrTime_cos     1.447331 -1.117309 -0.535260  0.651344  1.191010
DayofWeek_sin      0.629541  0.629541  0.629541  0.629541  0.629541
DayofWeek_cos     -1.218431 -1.218431 -1.218431 -1.218431 -1.218431
Month_sin         -0.153934 -0.153934 -0.153934 -0.153934 -0.153934
Month_cos          1.307492  1.307492  1.307492  1.307492  1.307492
Day_sin            0.556902  0.556902  0.556902  0.556902  0.556902
Day_cos            1.344929  1.344929  1.344929  1.344929  1.344929
Cluster            3.000000  2.000000  2.000000  3.000000  3.000000
```

[57]:
```
fig, ax = plt.subplots(figsize=(6, 6))
sns.countplot(x = kmeans_classified['Cluster'])
plt.show()
```



The first cluster have a more data.

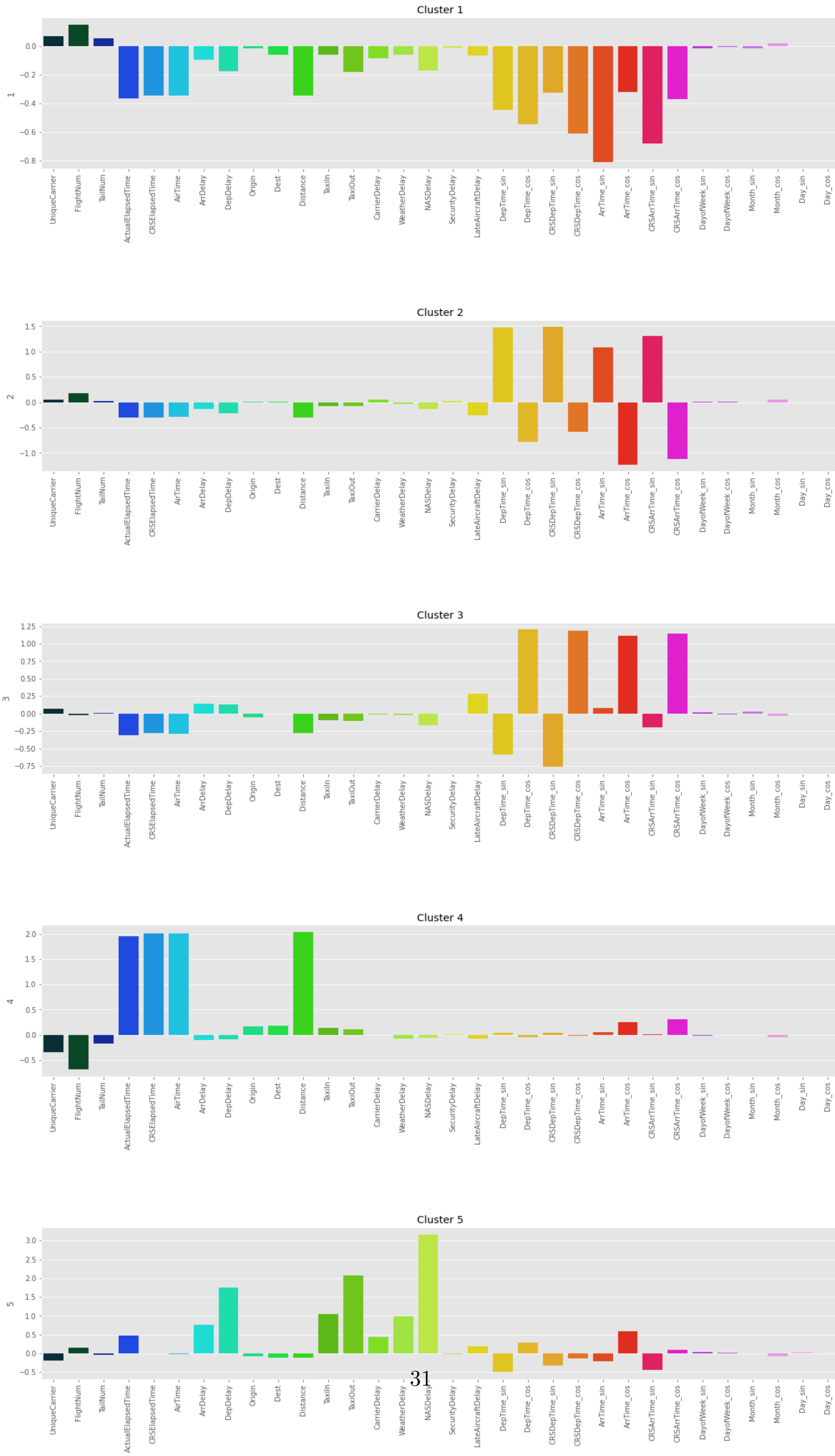
[60]:
```
cluster_group = kmeans_classified.groupby("Cluster").mean()
cluster_group
```

[60]:
```
         UniqueCarrier  FlightNum   TailNum  ActualElapsedTime  \
Cluster
1             0.067745   0.148634  0.049985          -0.367888
2             0.055504   0.178451  0.028416          -0.300336
3             0.069816  -0.018780  0.007600          -0.306774
4            -0.352028  -0.689707 -0.178790           1.947849
5            -0.191398   0.152667 -0.032649           0.470191

         CRSElapsedTime   AirTime  ArrDelay  DepDelay    Origin      Dest  \
Cluster
1             -0.347393 -0.343802 -0.095799 -0.174704 -0.019299 -0.062041
2             -0.304392 -0.292776 -0.138458 -0.212379  0.011232  0.003266
```

```
3              -0.277482 -0.292087  0.142590  0.128484 -0.050887  0.001688
4               2.004890  2.012463 -0.103586 -0.091653  0.164139  0.183796
5              -0.006284 -0.018778  0.766118  1.753419 -0.066894 -0.112403

         …  ArrTime_sin  ArrTime_cos  CRSArrTime_sin  CRSArrTime_cos  \
Cluster  …
1        …    -0.809003    -0.319706       -0.678491       -0.371410
2        …     1.076906    -1.229542        1.302661       -1.122825
3        …     0.077590     1.114261       -0.193873        1.147392
4        …     0.049303     0.257770        0.009783        0.301715
5        …    -0.216971     0.586442       -0.435360        0.087503

         DayofWeek_sin  DayofWeek_cos  Month_sin  Month_cos   Day_sin  \
Cluster
1            -0.016325      -0.005235  -0.017631   0.019034 -0.002332
2             0.009688       0.013014  -0.008656   0.058285 -0.004755
3             0.014574      -0.009192   0.027490  -0.032115  0.001084
4            -0.019411       0.001564  -0.004582  -0.050979 -0.001075
5             0.031647       0.026507   0.007614  -0.075571  0.036552

          Day_cos
Cluster
1       -0.002443
2       -0.000781
3       -0.000959
4       -0.002251
5        0.033915

[5 rows x 32 columns]
```

[73]:

```
fig, ax = plt.subplots(nrows=5, figsize=(20, 35))
plt.subplots_adjust(hspace = 1)


for cluster in range(0,5):

    sns.barplot(y=cluster_group.iloc[cluster] ,  x=cluster_group.columns,
                palette="gist_ncar",ax=ax[cluster])
    ax[cluster].set_title(f'Cluster {cluster+1 }')
    ax[cluster].set_xticklabels(labels=cluster_group.columns, rotation=90)
```



Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
UniqueCarrier
FlightNum
TailNum
ActualElapsedTime
CRSElapsedTime
AirTime
ArrDelay
DepDelay
Origin
Dest
Distance
TaxiIn
TaxiOut
CarrierDelay
WeatherDelay
NASDelay
SecurityDelay
LateAircraftDelay
DepTime_sin
DepTime_cos
CRSDepTime_sin
CRSDepTime_cos
ArrTime_sin
ArrTime_cos
CRSArrTime_sin
CRSArrTime_cos
DayofWeek_sin
DayofWeek_cos
Month_sin
Month_cos
Day_sin
Day_cos

Cluster 1 is mostly composed of flight schedules. Cluster 2 is composed mostly of flight schedules in the cyclic form sin. Cluster 3 is mostly composed of the flight schedules in their cyclic form cos. Cluster 4 is mostly composed of flight duration and distance. Cluster 5 is mostly composed of delays.

Level 2

Exercise 2

Group the different flights using the hierarchical clustering algorithm.

[74]:
```
cluster_five = pca_PC21_df[["PC_1","PC_2","PC_3","PC_4","PC_5"]]
```

[75]:
```
#concat the 5 PC with the Kmean dataframe
cluster_df = pd.concat([kmeans_classified,cluster_five],axis=1)
```

[76]:
```
cluster_sample_df = cluster_df.sample(frac=0.009, random_state=7)
cluster_sample_df
```

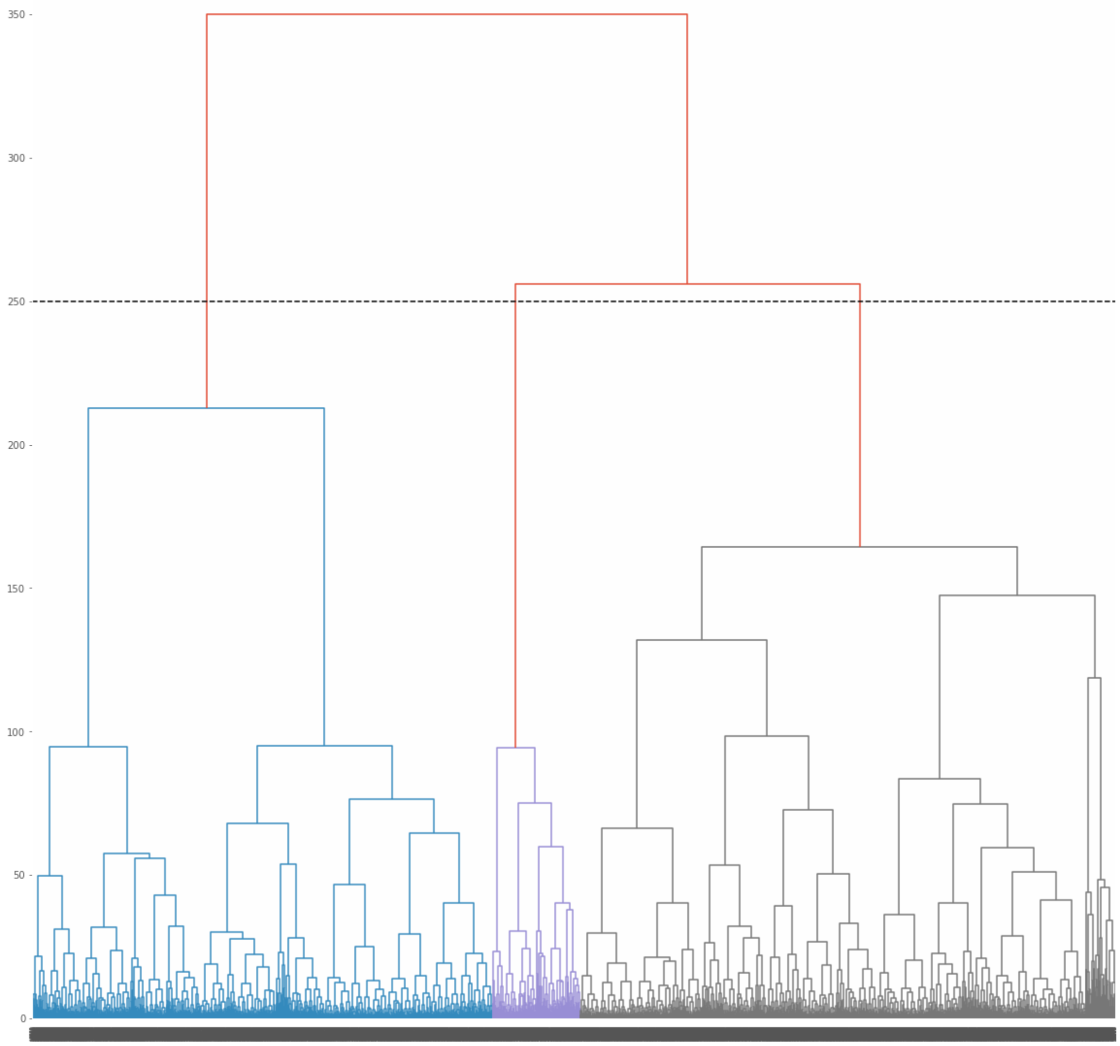
[76]:
```
         UniqueCarrier  FlightNum   TailNum  ActualElapsedTime  \
258182        0.653327  -0.847342  1.127343          -0.170773
437414        0.484787   1.694473  0.055994          -1.267080
1499466      -1.706234  -0.718053 -0.627860           0.911657
1549589       0.316247   2.265929  0.582489          -0.434441
23091         0.990407  -0.046269 -1.094035          -0.573214
…                  …          …        …                  …
356393       -1.200614  -0.688058 -0.014162           2.757339
1790088       0.484787   1.321605 -0.559671          -0.684233
1025022       0.147707  -0.540669 -0.057436          -0.642601
35587         1.158947   0.156455 -1.381869           0.620234
1381707      -1.706234  -0.949221  0.043536          -0.656478

         CRSElapsedTime   AirTime  ArrDelay  DepDelay    Origin      Dest  \
258182         0.067416 -0.149720 -1.305077 -0.621255  0.792761 -1.398861
437414        -1.252189 -1.242336 -1.305077 -0.527386  0.131131  1.250995
1499466        1.274715  0.972032  0.766238  0.561494  0.193549 -0.854031
1549589       -0.311619 -0.615903 -1.305077 -0.471064  0.967531  0.000362
23091         -0.550272 -0.470221 -1.305077 -0.546159  0.031262 -1.423626
…                   …         …         …         …         …         …
356393         2.327591  2.574534  0.766238 -0.395969  0.043746  1.436732
1790088       -0.676617 -0.645039  0.766238 -0.414743  0.642958  1.436732
1025022       -0.578348 -0.674176  0.766238 -0.358422 -1.229579 -0.779736
35587          0.923756  0.695236  0.766238  0.430077 -0.630367  0.309925
1381707       -0.620463 -0.543062 -1.305077 -0.508612 -0.817621 -0.730206

         …  Month_sin  Month_cos   Day_sin   Day_cos  Cluster      PC_1  \
```

```
258182    …   0.603178   1.127292  1.405294 -0.037484        3  1.921690
437414    …   1.157423   0.634973  1.113390 -0.837979        1  0.794537
1499466   …  -1.465290  -0.710065 -1.313090  0.529898        3  2.724801
1549589   …  -1.668158  -0.037546  0.423553 -1.325631        3  1.673346
23091     …  -0.153934   1.307492 -1.384285  0.250626        1 -0.669639
…      …        …         …        …        …        …        …
356393    …   0.603178   1.127292  0.145330 -1.383702        4  2.922178
1790088   …  -0.911046   1.127292 -1.014141  1.017021        1 -1.902201
1025022   …   0.603178  -1.202383 -1.398671 -0.037484        2 -3.557005
35587     …  -0.153934   1.307492  0.556902  1.344929        1 -1.287937
1381707   …  -0.911046  -1.202383  0.285891  1.431290        3  2.105725

             PC_2      PC_3      PC_4      PC_5
258182  -0.419542 -0.627985 -1.643473 -0.187700
437414  -3.165526 -1.200955 -1.165235 -0.061525
1499466  2.304652  2.831202 -0.673770 -0.016147
1549589 -1.740521 -0.756361 -0.705609 -1.225132
23091   -1.346261 -1.946598 -0.808675  0.362436
…           …        …        …        …
356393   5.203907  2.612439 -0.501508 -1.284480
1790088 -1.391663 -1.143748  0.553791  0.836591
1025022 -0.366642  1.216338  0.199933 -0.672273
35587    1.197773 -1.187212  1.034577  1.328356
1381707 -0.882859  1.611582 -2.448200 -0.967342

[17355 rows x 38 columns]
```

I had to reduce the sample size to run the models in my computer.

Dendrogram

[62]:
```python
fig, ax = plt.subplots(figsize=(20,20))
dendogram = shc.dendrogram(shc.
 ↪linkage(cluster_sample_df[["PC_1","PC_2","PC_3","PC_4","PC_5"]],
                                       method='ward'))
ax.axhline(y = 250 , color = 'k', linestyle = '--')
plt.show()
```



the highest vertical distance that doesn't intersect with any clusters is the intersection with the red dot line, this line say that are at least 3 clusters.

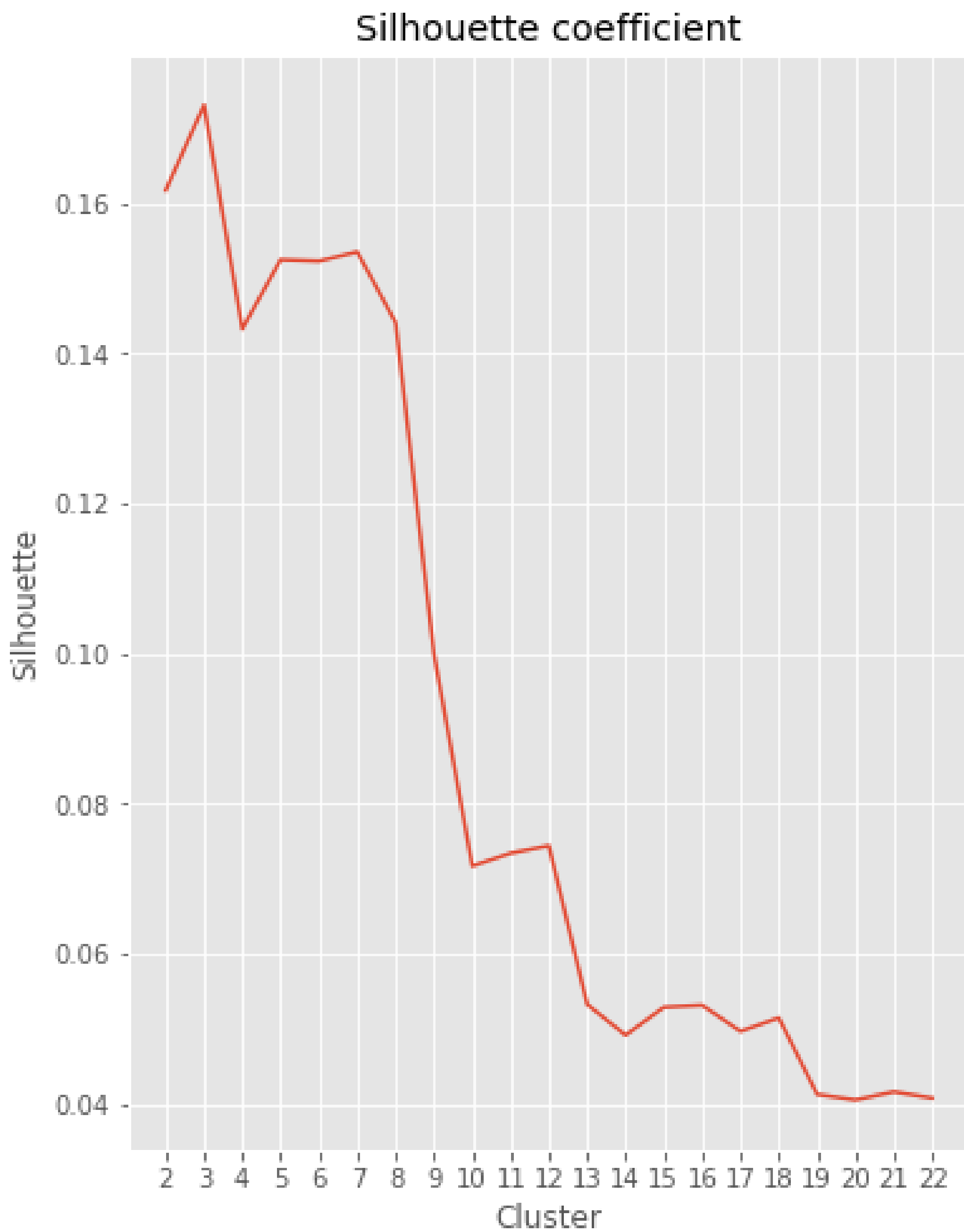
Silhouette Coefficient

[67]:
```
range_clusters = range(2, 23)
mean_silhouette = []

for n_clusters in range_clusters:
    hier = AgglomerativeClustering(
                    linkage    = 'ward',
                    n_clusters = n_clusters
             )

    cluster_labels = hier.fit_predict(cluster_sample_df)
```

```python
    silhouette_avg = silhouette_score(cluster_sample_df, cluster_labels)
    mean_silhouette.append(silhouette_avg)

fig, ax = plt.subplots(figsize=(6, 8))
sns.lineplot(range_clusters, mean_silhouette,markers="o")
ax.set_title("Silhouette coefficient")
ax.set_xlabel('Cluster')
ax.set_ylabel('Silhouette')
plt.xticks(range_clusters)
plt.show
```

```
/Users/franciscoregalado/opt/anaconda3/lib/python3.9/site-
packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables
as keyword args: x, y. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

[67]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



The Silhouette coefficient computes 3 cluster for the sample.

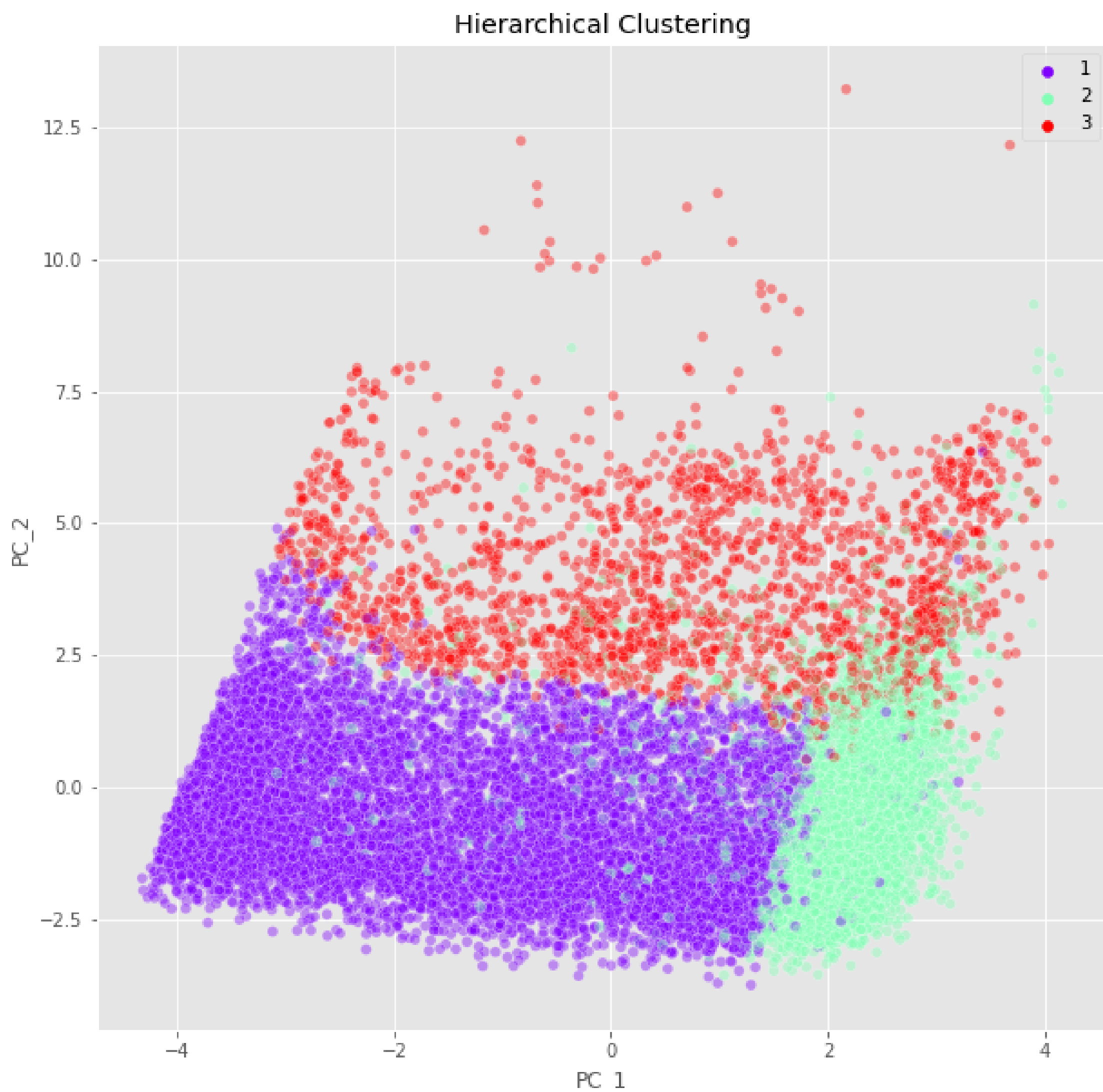
[77]:
```
#create the aglomerative clustering model for 3 clusters
hier = AgglomerativeClustering(
                    linkage    = 'ward',
                    n_clusters = 3
             )
```

```
hier.fit(cluster_sample_df)
```

[77]: AgglomerativeClustering(n_clusters=3)

[78]:
```
fig, ax = plt.subplots(1, 1, figsize=(10, 10))

sns.scatterplot(x=cluster_sample_df["PC_1"], y=cluster_sample_df["PC_2"],
                hue=hier.labels_+1 , alpha=.4,
                palette = 'rainbow')
ax.set_title("Hierarchical Clustering")
ax.set_xlabel('PC_1')
ax.set_ylabel('PC_2')
plt.show
```

[78]: <function matplotlib.pyplot.show(close=None, block=None)>



It seem that the cluster 1 is overlaps cluster 2.

Level 3

Exercise 3

Calculate clustering performance using a parameter such as silhouette

[79]:
```
cluster_sample_df['Hierarchical_Cluster'] = hier.labels_
```

[80]:
```
cluster_sample_df.T
```

[80]:
```
                      258182    437414    1499466   1549589   23091    \
UniqueCarrier        0.653327  0.484787 -1.706234  0.316247  0.990407
FlightNum           -0.847342  1.694473 -0.718053  2.265929 -0.046269
TailNum              1.127343  0.055994 -0.627860  0.582489 -1.094035
ActualElapsedTime   -0.170773 -1.267080  0.911657 -0.434441 -0.573214
CRSElapsedTime       0.067416 -1.252189  1.274715 -0.311619 -0.550272
AirTime             -0.149720 -1.242336  0.972032 -0.615903 -0.470221
ArrDelay            -1.305077 -1.305077  0.766238 -1.305077 -1.305077
DepDelay            -0.621255 -0.527386  0.561494 -0.471064 -0.546159
Origin               0.792761  0.131131  0.193549  0.967531  0.031262
Dest                -1.398861  1.250995 -0.854031  0.000362 -1.423626
Distance             0.177824 -1.177846  1.087413 -0.740476 -0.489556
TaxiIn              -0.154020 -0.723491  0.225627  2.883155  0.035803
TaxiOut             -0.085077 -0.154966 -0.154966 -0.294744 -0.644190
CarrierDelay        -0.342706 -0.342706 -0.342706 -0.342706 -0.342706
WeatherDelay        -0.137875 -0.137875 -0.137875 -0.137875 -0.137875
NASDelay            -0.345292 -0.345292 -0.345292 -0.345292 -0.345292
SecurityDelay       -0.035829 -0.035829 -0.035829 -0.035829 -0.035829
LateAircraftDelay   -0.455575 -0.455575  0.852859 -0.455575 -0.455575
DepTime_sin         -0.985613 -0.984091  0.095325 -0.967809 -0.615934
DepTime_cos          0.474451  0.310314  1.874900  0.644602 -0.580827
CRSDepTime_sin      -0.984862 -0.972369 -0.400301 -0.980565 -0.574311
CRSDepTime_cos       0.597073  0.378877  1.945480  0.725205 -0.571011
ArrTime_sin         -0.394936 -1.025146  1.412990 -0.698494 -0.758679
ArrTime_cos          1.028392  0.170371  1.346170  0.782213 -0.713401
CRSArrTime_sin      -0.306468 -0.978611  1.012694 -0.688218 -0.660949
CRSArrTime_cos       1.228379  0.229962  1.567889  0.897791 -0.670338
DayofWeek_sin       -1.424121  1.418236 -0.635426  1.418236 -0.635426
DayofWeek_cos       -0.285795 -0.285795 -1.218431 -0.285795 -1.218431
Month_sin            0.603178  1.157423 -1.465290 -1.668158 -0.153934
Month_cos            1.127292  0.634973 -0.710065 -0.037546  1.307492
Day_sin              1.405294  1.113390 -1.313090  0.423553 -1.384285
Day_cos             -0.037484 -0.837979  0.529898 -1.325631  0.250626
Cluster              3.000000  1.000000  3.000000  3.000000  1.000000
```

```
PC_1                 1.921690  0.794537  2.724801  1.673346 -0.669639
PC_2                -0.419542 -3.165526  2.304652 -1.740521 -1.346261
PC_3                -0.627985 -1.200955  2.831202 -0.756361 -1.946598
PC_4                -1.643473 -1.165235 -0.673770 -0.705609 -0.808675
PC_5                -0.187700 -0.061525 -0.016147 -1.225132  0.362436
Hierarchical_Cluster  1.000000  0.000000  1.000000  0.000000  0.000000

                      101715    398768    241648    598168   1213280  …  \
UniqueCarrier       -0.694994  0.990407  0.484787  1.158947 -1.874774  …
FlightNum            1.099228  0.818930  2.173358 -1.117814 -0.086090  …
TailNum              0.337928 -0.654742  1.675475 -1.636266 -1.686751  …
ActualElapsedTime   -1.086675 -0.684233  0.051264  0.120651 -0.254036  …
CRSElapsedTime      -1.111806 -0.760847  0.025301  0.362222 -0.044891  …
AirTime             -1.271472 -0.543062  0.054235  0.272758 -0.193425  …
ArrDelay             0.766238  0.766238 -1.305077 -1.305077  0.766238  …
DepDelay             0.411304 -0.489838 -0.621255 -0.302100  0.749232  …
Origin              -1.604087 -0.205926 -0.830105  0.780277 -1.604087  …
Dest                 0.322307  1.684382 -0.593998  1.226230  0.309925  …
Distance            -1.195271 -0.543573 -0.020821  0.700576 -0.127114  …
TaxiIn              -0.343844 -0.723491 -0.343844 -0.723491 -0.533667  …
TaxiOut              0.753593 -0.574301  0.124590 -0.434523 -0.154966  …
CarrierDelay         1.452664  0.099231 -0.342706 -0.342706  1.507906  …
WeatherDelay        -0.137875 -0.137875 -0.137875 -0.137875 -0.137875  …
NASDelay            -0.345292 -0.203163 -0.345292 -0.345292 -0.345292  …
SecurityDelay       -0.035829 -0.035829 -0.035829 -0.035829 -0.035829  …
LateAircraftDelay   -0.455575 -0.427736 -0.455575 -0.455575 -0.455575  …
DepTime_sin         -0.446747 -0.718431  0.533557  1.622360 -0.335414  …
DepTime_cos          1.562217  1.260130 -1.156145 -0.886314  1.650154  …
CRSDepTime_sin      -0.758136 -0.792226  0.509654  1.608163 -0.751011  …
CRSDepTime_cos       1.494023  1.428694 -1.130265 -0.702845  1.506888  …
ArrTime_sin         -0.018246 -0.228454 -0.316870 -0.085265  0.135758  …
ArrTime_cos          1.217811  1.123857 -1.073281 -1.188139  1.271351  …
CRSArrTime_sin      -0.423100 -0.331755 -0.208352  0.009360 -0.312825  …
CRSArrTime_cos       1.147573  1.211976 -1.027803 -1.132716  1.224309  …
DayofWeek_sin        0.629541 -1.142639  1.136754  1.136754 -0.635426  …
DayofWeek_cos       -1.218431  0.877183  0.877183  0.877183 -1.218431  …
Month_sin           -0.153934  1.157423  0.603178  1.360290 -0.153934  …
Month_cos            1.307492  0.634973  1.127292 -0.037546 -1.382584  …
Day_sin              0.556902 -1.355660  1.020764  1.390908  1.263635  …
Day_cos              1.344929 -0.322638  1.017021  0.250626 -0.593161  …
Cluster              3.000000  3.000000  1.000000  2.000000  3.000000  …
PC_1                 2.183129  2.173726 -2.314800 -2.763234  2.670746  …
PC_2                -2.661745 -1.863137 -0.388702  1.285087 -0.305432  …
PC_3                 1.605115  0.766095 -1.430796 -0.648203  1.583418  …
PC_4                 0.351377 -1.072626 -0.289922 -1.192106 -0.024766  …
PC_5                -0.665757  0.673434 -0.631580  1.224962  0.088327  …
Hierarchical_Cluster  1.000000  1.000000  0.000000  0.000000  1.000000  …
```

	36359	1040300	1461557	350918	1602376	\
UniqueCarrier	1.158947	-1.706234	0.653327	-1.369154	-1.706234	
FlightNum	0.276952	-0.574284	-0.859236	-0.774423	-0.475508	
TailNum	-1.471695	-0.281016	-0.405591	1.586305	-0.627860	
ActualElapsedTime	0.772884	-1.045043	1.161449	-0.073631	0.356565	
CRSElapsedTime	0.769334	-0.901230	1.302791	0.263953	0.221838	
AirTime	0.331031	-0.980108	1.263396	0.068803	0.301894	
ArrDelay	0.766238	0.766238	-1.305077	0.766238	-1.305077	
DepDelay	1.087160	0.448851	-0.602481	0.129697	-0.696350	
Origin	-0.630367	1.292104	-0.143508	1.391973	-0.842588	
Dest	1.535792	-0.854031	-0.866413	0.062275	0.446132	
Distance	0.186537	-0.902530	1.197191	0.176082	0.270177	
TaxiIn	0.035803	0.415450	0.225627	0.035803	1.364568	
TaxiOut	2.291153	-0.714079	-0.294744	-0.714079	-0.154966	
CarrierDelay	-0.342706	-0.342706	-0.342706	-0.342706	-0.204601	
WeatherDelay	0.092325	-0.137875	-0.137875	-0.137875	-0.137875	
NASDelay	-0.345292	-0.345292	-0.345292	0.543017	-0.025501	
SecurityDelay	-0.035829	-0.035829	-0.035829	-0.035829	-0.035829	
LateAircraftDelay	2.244810	1.075571	-0.455575	-0.455575	-0.427736	
DepTime_sin	-0.358326	-0.854763	-0.483732	-0.446747	-0.907892	
DepTime_cos	1.633303	1.018777	-0.717228	1.562217	0.884124	
CRSDepTime_sin	-0.823708	-0.975574	-0.457646	-0.702255	-0.922433	
CRSDepTime_cos	1.361782	0.785302	-0.697859	1.588777	1.080892	
ArrTime_sin	0.714203	-0.633074	-0.902858	0.569791	-0.129083	
ArrTime_cos	1.380617	0.845982	-0.507291	1.365629	1.171555	
CRSArrTime_sin	-0.070029	-0.821300	-0.847115	0.378083	-0.201633	
CRSArrTime_cos	1.357101	0.709377	-0.403970	1.508174	1.290592	
DayofWeek_sin	0.629541	0.629541	-1.424121	-1.142639	0.629541	
DayofWeek_cos	-1.218431	-1.218431	-0.285795	0.877183	-1.218431	
Month_sin	-0.153934	0.603178	-1.465290	0.603178	-1.668158	
Month_cos	1.307492	-1.202383	-0.710065	1.127292	-0.037546	
Day_sin	0.556902	1.113390	-1.188002	0.556902	-0.550279	
Day_cos	1.344929	-0.837979	0.788897	1.344929	1.344929	
Cluster	3.000000	3.000000	4.000000	3.000000	3.000000	
PC_1	3.116731	2.228469	0.065798	2.547038	2.419835	
PC_2	0.701773	-1.959436	2.178156	0.246471	0.649488	
PC_3	2.107743	0.107770	-2.479177	1.630790	-0.129065	
PC_4	0.940504	0.297731	-0.845125	-0.733832	-1.395011	
PC_5	0.347810	0.279935	0.360290	-0.342170	-1.301999	
Hierarchical_Cluster	1.000000	1.000000	2.000000	1.000000	1.000000	

	356393	1790088	1025022	35587	1381707
UniqueCarrier	-1.200614	0.484787	0.147707	1.158947	-1.706234
FlightNum	-0.688058	1.321605	-0.540669	0.156455	-0.949221
TailNum	-0.014162	-0.559671	-0.057436	-1.381869	0.043536
ActualElapsedTime	2.757339	-0.684233	-0.642601	0.620234	-0.656478

```
CRSElapsedTime        2.327591 -0.676617 -0.578348  0.923756 -0.620463
AirTime               2.574534 -0.645039 -0.674176  0.695236 -0.543062
ArrDelay              0.766238  0.766238  0.766238  0.766238 -1.305077
DepDelay             -0.395969 -0.414743 -0.358422  0.430077 -0.508612
Origin                0.043746  0.642958 -1.229579 -0.630367 -0.817621
Dest                  1.436732  1.436732 -0.779736  0.309925 -0.730206
Distance              2.132917 -0.573196 -0.621986  0.569888 -0.372808
TaxiIn                1.934038  0.035803  1.364568 -0.533667 -0.723491
TaxiOut               0.823482 -0.364633 -0.504412 -0.015188 -0.434523
CarrierDelay         -0.094117 -0.342706  0.154473 -0.342706 -0.342706
WeatherDelay         -0.137875 -0.137875 -0.137875 -0.137875 -0.137875
NASDelay              0.791744 -0.345292 -0.345292 -0.345292 -0.345292
SecurityDelay        -0.035829 -0.035829 -0.035829 -0.035829 -0.035829
LateAircraftDelay    -0.093668  0.073366 -0.455575  0.769342 -0.455575
DepTime_sin          -0.109774  0.108965  1.579992  0.136373 -0.018713
DepTime_cos           1.787426 -1.067540 -0.912836 -1.076676  1.830162
CRSDepTime_sin       -0.290361  0.179956  1.555935  0.509654 -0.173814
CRSDepTime_cos        2.034669 -1.074933 -0.751334 -1.130265  2.112918
ArrTime_sin           1.752535 -0.362837  1.304056 -0.545686  0.312084
ArrTime_cos           1.261404 -1.045969 -1.360840 -0.918186  1.319067
CRSArrTime_sin        1.302388 -0.201633  1.440374 -0.235011  0.236093
CRSArrTime_cos        1.541595 -1.031529 -1.257833 -1.012681  1.471369
DayofWeek_sin        -1.424121 -1.142639  1.136754  0.629541 -1.424121
DayofWeek_cos        -0.285795  0.877183  0.877183 -1.218431 -0.285795
Month_sin             0.603178 -0.911046  0.603178 -0.153934 -0.911046
Month_cos             1.127292  1.127292 -1.202383  1.307492 -1.202383
Day_sin               0.145330 -1.014141 -1.398671  0.556902  0.285891
Day_cos              -1.383702  1.017021 -0.037484  1.344929  1.431290
Cluster               4.000000  1.000000  2.000000  1.000000  3.000000
PC_1                  2.922178 -1.902201 -3.557005 -1.287937  2.105725
PC_2                  5.203907 -1.391663 -0.366642  1.197773 -0.882859
PC_3                  2.612439 -1.143748  1.216338 -1.187212  1.611582
PC_4                 -0.501508  0.553791  0.199933  1.034577 -2.448200
PC_5                 -1.284480  0.836591 -0.672273  1.328356 -0.967342
Hierarchical_Cluster  2.000000  0.000000  0.000000  0.000000  1.000000

[39 rows x 17355 columns]
```

[81]:
```python
kmeans_silhouette = silhouette_score(cluster_sample_df[['PC_1', 'PC_2',
                                                       'PC_3','PC_4','PC_5']],
                                     cluster_sample_df['Cluster']).round(3)
print('K-means score:', kmeans_silhouette)
```

```
K-means score: 0.266
```

[75]:
```python
hier_silhouette = silhouette_score(cluster_sample_df[['PC_1','PC_2',
                                                     'PC_3','PC_4','PC_5']],
```

```
                                 cluster_sample_df['Hierarchical_Cluster']).
↪round(3)
print('Hirearchical Clustering Score:', hier_silhouette)
```

```
Hirearchical Clustering Score: 0.242
```

The Kmeans behave better than the Hirearchical Clustering, but both are overlaps data. Kmeans is less computationally intensive since my laptop can finish the model with out a dead kernel on jupyter notebook.