Лабораторная работа 2

Тема: Кластеризация

Вариант 6 (датасет flight_delays)

Разбить заданный датасет на 5 кластеров и на 2 кластера, используя агломеративный иерархический метод и метод к-средних; сравнить результаты;

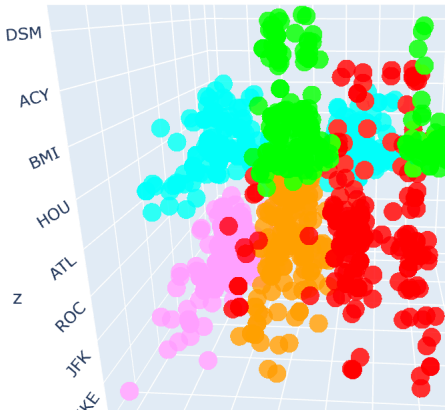
```
import pandas as pd
import numpy as np
import datetime
import scipy
import matplotlib.pyplot as plt
import random
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

#plotly imports
import plotly as py
import plotly.graph_objs as go
import plotly.io as pio

# Для отображения Plotly в Colab
def enable_plotly_in_cell():
  import IPython
  from plotly.offline import init_notebook_mode
  display(IPython.core.display.HTML('''
        <script src="/static/components/requirejs/require.js"></script>
  '''))
  init_notebook_mode(connected=False)

get_ipython().events.register('pre_run_cell', enable_plotly_in_cell)
```

```
def clastirize(clusters_amount, col_a, col_b, col_c, ClastirizeMethod):
  df = pd.read_csv('flight_delays.csv', sep=',')[:1000]
  cats = ["Month","DayofMonth","DayOfWeek", "UniqueCarrier","Origin","Dest","dep_delayed_15min"]

  for cat in cats:
    df[f'{cat}_cat'] = pd.factorize(df[cat], sort=True)[0]

  # Выбираем категориальные и числовые данные
  numer = df[["DepTime","Distance"]]
  cater = df[[f"{cat}_cat" for cat in cats]]

  # Подготавливаем числовые данные
  scaler = StandardScaler()
  numer = pd.DataFrame(scaler.fit_transform(numer))
  numer.columns = ["DepTime_Scaled", "Distance_Scaled"]

  # Сливаем подготовленные данные в новый датасет
  X = pd.concat([numer, cater], axis=1, join='inner')

  clasterizer = ClastirizeMethod(n_clusters=5)
  clusters = clasterizer.fit_predict(X)

  labels = clasterizer.labels_
  silhouette_avg = silhouette_score(X, labels)

  X["Cluster"] = clusters
  X = pd.concat([X, df], axis=1, join='inner')

  plotX = pd.DataFrame(np.array(X))
  plotX.columns = X.columns

  data = []

  colors = [
      dict(color = 'rgba(255, 128, 255, 0.8)'),
      dict(color = 'rgba(255, 128, 2, 0.8)'),
      dict(color = 'rgba(0, 255, 200, 0.8)'),
      dict(color = 'rgba(255, 0, 0, 0.8)'),
      dict(color = 'rgba(0, 255, 0, 0.8)'),
  ]
```

```
    for i in range(0, clusters_amount):
      cluster = plotX[plotX["Cluster"] == i]
      trace = go.Scatter3d(
          x = cluster[col_a],
          y = cluster[col_b],
          z = cluster[col_c],
          mode = "markers",
          name = f"Cluster{i}",
          marker = colors[i % len(colors)],
          text = None
      )
      data.append(trace)

    title = f"Визуализация задержек полётов; {ClastirizeMethod.__name__}; кластеров: {clusters_amount}; x - расстояние, км; y - точка отпра

    layout = dict(title = title,
                  xaxis= dict(title= 'PC1',ticklen= 5,zeroline= False),
                  yaxis= dict(title= 'PC2',ticklen= 5,zeroline= False),
                 )

    fig = dict(data = data, layout = layout)

    pio.show(fig)
```

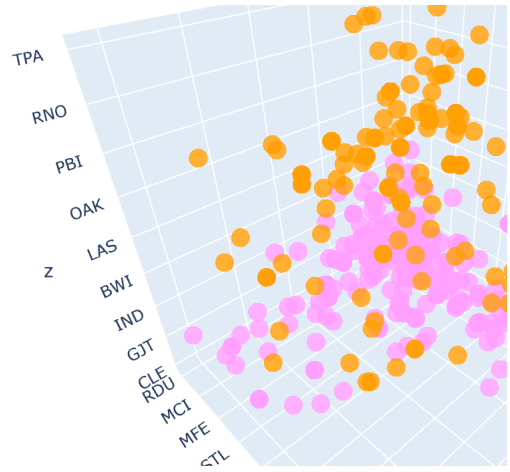
```
clastirize(5, "Distance", "Origin", "Dest", KMeans)
```

Визуализация задержек полётов; KMeans; кластеров: 5; x - расстояние, км; y - точка отправки; z - то



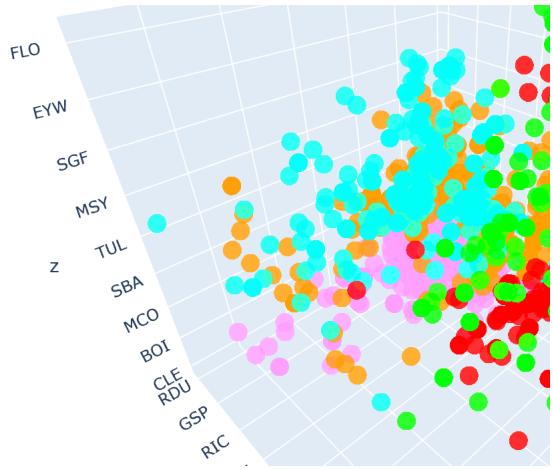

```
clastirize(2, "Distance", "Origin", "Dest", KMeans)
```

Визуализация задержек полётов; KMeans; кластеров: 2; x - расстояние, км; y - точка отправки; z - то




```
clastirize(5, "Distance", "Origin", "Dest", AgglomerativeClustering)
```

Визуализация задержек полётов; AgglomerativeClustering; кластеров: 5; x - расстояние, км; y - точка




```
clastirize(2, "Distance", "Origin", "Dest", AgglomerativeClustering)
```

Визуализация задержек полётов; AgglomerativeClustering; кластеров: 2; x - расстояние, км; y - точка