- **خوشه‌بندی آفلاین**

به‌منظور خوشه‌بندی آفلاین داده‌ها، از بستر PySpark استفاده شده است. با توجه به حجم زیاد داده‌ها، استفاده از این بستر موجب افزایش سرعت عملیات خوشه‌بندی داده‌ها می‌شود. در ادامه گام‌های زیر انجام می‌شود:

❖ پس از تعریف spark session، با استفاده از دستور نشان داده شده در شکل زیر داده‌ها خوانده می‌شوند.

```
data = sc.read \
  .option('header', 'True')\
  .option('inferSchema', 'True')\
  .option('sep', ',')\
  .csv('/content/drive/MyDrive/data.csv')
```

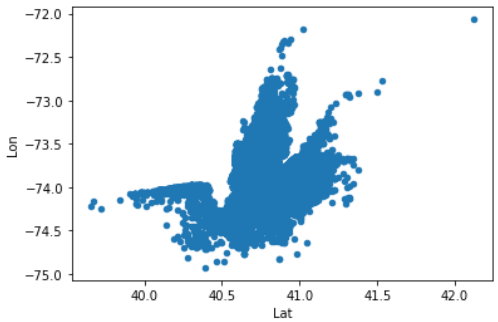
❖ در گام بعدی با استفاده از قطعه کد قابل مشاهده در شکل زیر، حجم ۲۰ درصد از کل داده‌ها محاسبه می‌شود.

```
import math

data_len = data.count()
length = math.trunc(data_len * 0.2)
```

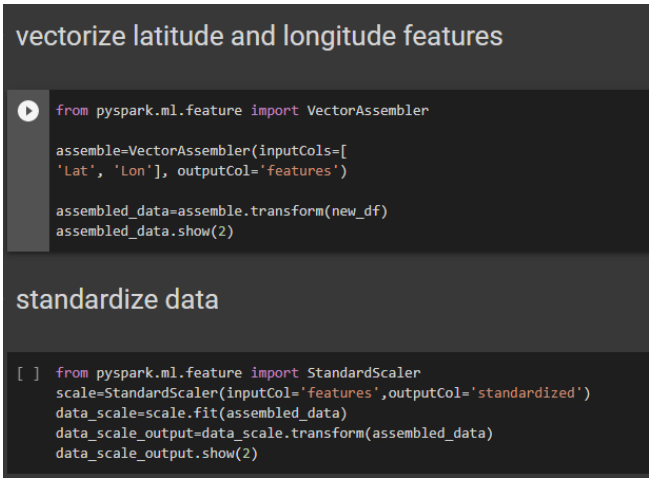
❖ ویژگی‌هایی که برای خوشه‌بندی داده‌ها مورد استفاده قرار گرفته‌اند، ویژگی‌های latitude و longitude می‌باشند. بنابراین سایر ستون‌ها از دیتافریم حذف می‌شوند. سپس با کمک تابع distinct داده‌های تکراری حذف می‌شوند.

```
data = data.na.drop()

# pick Longitude and Latitude features
unimportant_features = ['Date/Time', 'Base']
data = data.drop(*tuple(unimportant_features))

df = data.distinct()
```

❖ درگام بعدی بیست درصد داده‌ها جداسازی شده و شافل می‌شوند. شکل زیر نشان‌دهنده‌ی نمودار پراکندگی داده‌های مورد استفاده برای خوشه‌بندی آفلاین است.



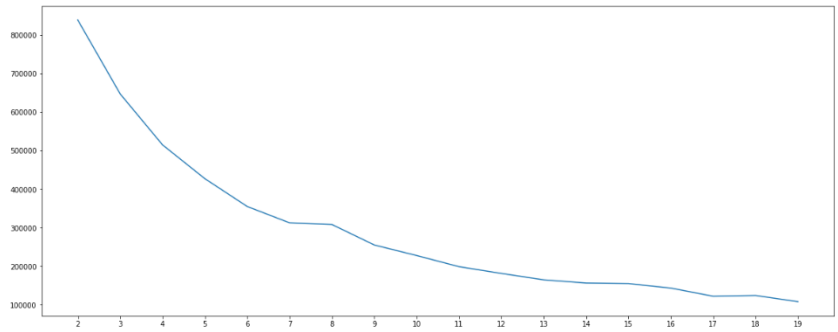
❖ در ادامه با استفاده از vectorizer و standardscaler مقادیر ستون‌ها به بردار تبدیل شده و نرمال‌سازی می‌شوند.



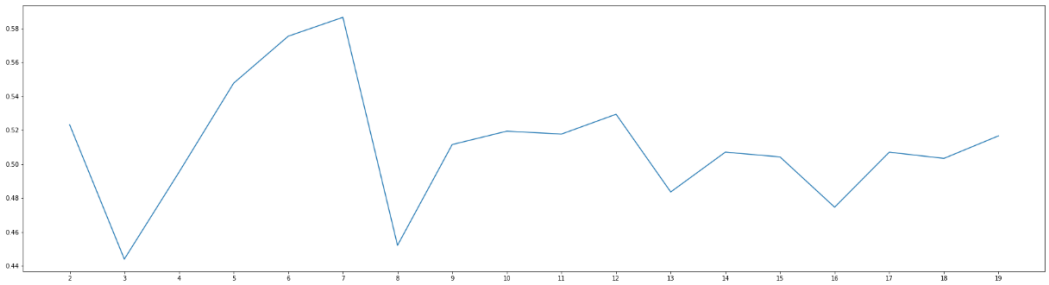
❖ الگوریتم خوشه‌بندی مورد استفاده در این فاز، الگوریتم k-means می‌باشد. به‌منظور یافتن بهترین تعداد خوشه، الگوریتم به‌ازای مقادیر مختلف k اجرا می‌شود. سپس به‌ازای هر خوشه‌بندی مقادیر SSE[1] و silhouette به‌طور مجزا ذخیره می‌شوند. در ادامه الگوریتم خوشه‌بندی به‌ازای دو تا بیست خوشه اجرا شده است.

با استفاده از روش آرنج[2]، می‌توان بهترین مقدار برای متغیر k را تخمین زد. بر مبنای این روش نموداری بر اساس مقادیر خطا به‌ازای هر k رسم می‌شود. نقطه‌ای که نمودار در آن دچار شکست شده باشد، می‌تواند انتخاب مناسبی برای k باشد. شکل زیر نشان‌دهنده‌ی نمودار خطا بر حسب تعداد خوشه است.

[1] Sum of Squared Errors
[2] Elbow Method

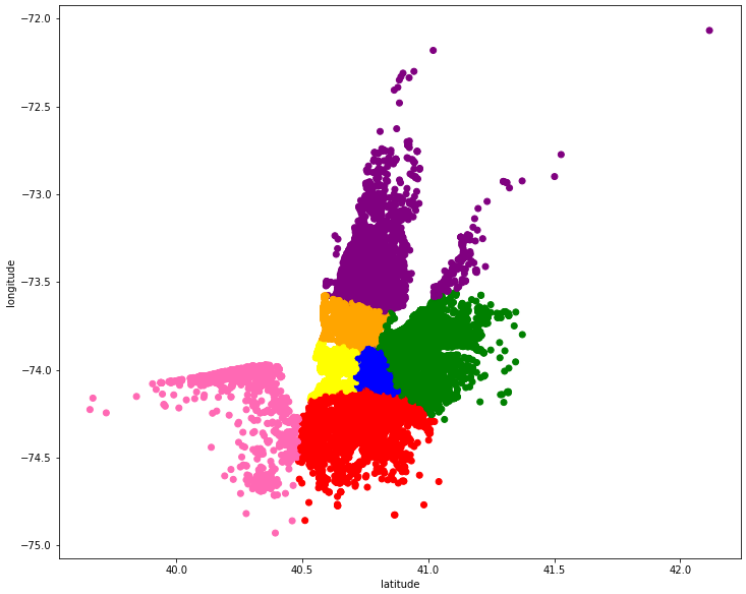


با توجه به نمودار بالا مقدار مناسب برای k یکی از دو مقدار ۷ یا ۸ است زیرا نمودار در این مقدار دچار شکست شده است. به‌منظور تصمیم‌گیری بهتر در انتخاب تعداد خوشه، نمودار silhouette score بر حسب تعداد خوشه نیز ترسیم شده است. هر قدر معیار silhouette score برای یک خوشه‌بندی بالاتر باشد، آن خوشه‌بندی مناسب‌تر است. با توجه به نمودار به‌دست‌آمده، مقدار k=7 بالاترین مقدار silhouette score را دارد. بنابراین عدد هفت به‌عنوان تعداد خوشه‌ها انتخاب می‌شود.



در ادامه خوشه‌بندی به‌ازای هفت خوشه انجام شده و مدل ذخیره می‌شود. با استفاده از قطعه کد زیر می‌توان نمودار نشان‌دهنده‌ی داده‌های خوشه‌بندی شده را نمایش داد.

```
import numpy as np
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

transformed = c_pred.select('Lat', 'Lon', 'prediction')
rows = transformed.collect()
df_pred = sqlContext.createDataFrame(rows)
pddf_pred = df_pred.toPandas()

colors=("red","blue","green", "yellow","hotpink","purple","orange","aqua","lightgreen","lightcoral","grey","darkviolet",
        "gold","wheat","rosybrown")

thowdee = plt.figure(figsize=(12,10)).gca()
thowdee.scatter(pddf_pred.Lat, pddf_pred.Lon, c= np.take(colors, pddf_pred.prediction))
thowdee.set_xlabel('latitude')
thowdee.set_ylabel('longitude')
plt.show()
```



- **پرسش داده‌ها در elasticsearch**

با استفاده از کوئری‌های زیر، می‌توان موارد خواسته‌شده در صورت پروژه را به‌دست آورد.

```
GET /taxi_service_index/_search
{
  "aggs":{
    "geo":{
      "geohash_grid":{
        "field": "Location",
        "precision": 7,
        "size":10
      },
      "aggs":{
        "point": {
          "geo_centroid": {
            "field": "Location"
          }
        }
      }
    }
  }
}
```

```
{
  "took" : 546,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 10000,
      "relation" : "gte"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "taxi_service_index",
        "_id" : "973",
        "_score" : 1.0,
        "_source" : {
          "Date/Time" : "09/04/2014 10:12:00",
          "Lat" : 40.7356,
```

```
GET /taxi_service_index/_search
{
  "aggs": {
    "filter_date":{
      "filter":{
        "range":{
          "Date/Time": {
            "gte": "09/01/2014 00:01:00",
            "lte": "09/01/2014 00:20:00"
          }
        }
      },
      "aggs": {
        "geo":{
          "geohash_grid": {
            "field": "Location",
            "precision":7,
            "size":10
          },
          "aggs": {
            "point": {
              "geo_centroid": {
                "field": "Location"
              }
```

```
        "_source" : {
          "Date/Time" : "09/04/2014 10:12:00",
          "Lat" : 40.7356,
          "Lon" : -73.975,
          "Base" : "B02512",
          "Cluster_number" : 0,
          "Location" : {
            "lat" : 40.7356,
            "lon" : -73.975
          }
        }
      },
      {
        "_index" : "taxi_service_index",
        "_id" : "974",
        "_score" : 1.0,
        "_source" : {
          "Date/Time" : "09/04/2014 10:16:00",
          "Lat" : 40.763,
          "Lon" : -73.9699,
          "Base" : "B02512",
          "Cluster_number" : 0,
          "Location" : {
            "lat" : 40.763,
            "lon" : -73.9699
```

- **نوشتن داده‌ها در کانال کافکا (فاز چهارم)**

به‌منظور نوشتن داده‌ها در تاپیک مربوط به این فاز (elastic topic) از کد زیر استفاده شده است.

```
def _normal_run(self):
    """
    run method for normal type
        create normal
    :return:
    """
    self._generate_index()
    self._normal_setup()
    data_id = 0

    if self.consumer:
        for each in self.consumer:
            produced_data, key, timestamp = self._produce_answer(each, data_id=data_id)
            data_id += 1
            self._send_data(data=produced_data, key=key, timestamp_ms=timestamp)

    else:
        print("No data in previous phase topic")
```

- **پرسش داده‌ها در کاساندرا**

به‌منظور اجرای کوئری‌ها می‌توان از cqlsh در محیط cmd استفاده کرد. اما به‌منظور استفاده از نتایج کوئری‌ها در پایتون لازم است cassandra driver با دستور pip نصب شود. کوئری‌های اجراشده به‌صورت زیر هستند.

```
query = "select * from TaxiServiceKeyspace.midday_table limit 2"
rows = session.execute(query)
```

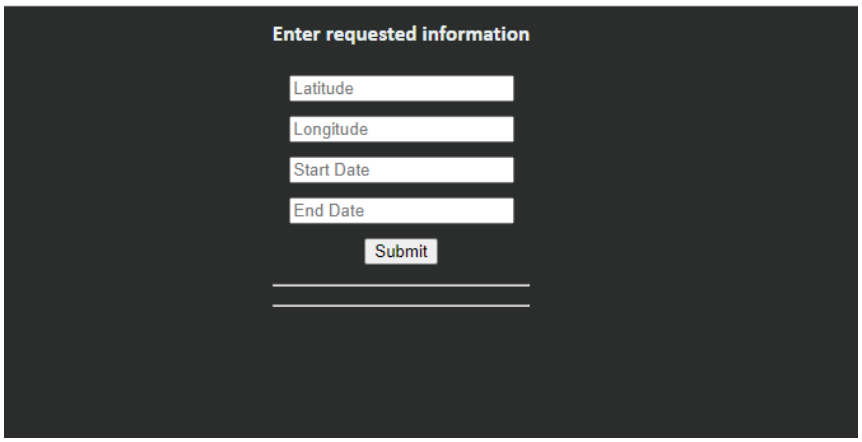
```
lat = request.form.get("lat")
lon = request.form.get("lon")

point_lat = round(float(lat), 4)
point_lon = round(float(lon), 4)
start_point = (point_lat, point_lon)

session.encoder.mapping[tuple] = session.encoder.cql_encode_tuple
query = "select * from TaxiServiceKeyspace.start_coordinates_table WHERE start=%s"

rows = session.execute(query, (start_point, ))
```

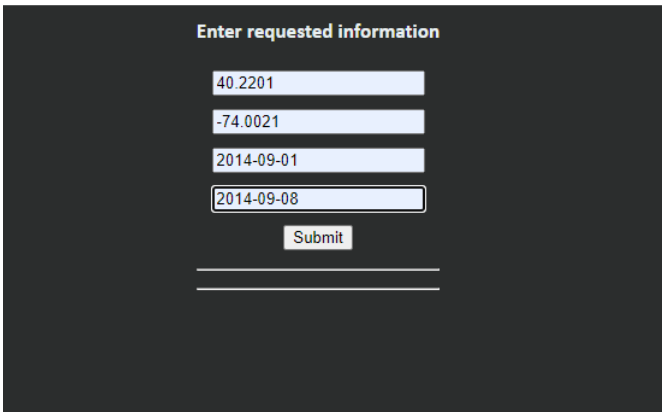
```
lat = request.form.get("lat")
lon = request.form.get("lon")

point_lat = round(float(lat), 4)
point_lon = round(float(lon), 4)

start = request.form.get("start")
end = request.form.get("end")
week_date = (start, end)

session.encoder.mapping[tuple] = session.encoder.cql_encode_tuple
query = "select * from TaxiServiceKeyspace.week_table WHERE week=%s AND lat CONTAINS %s AND lon CONTAINS %s AL
rows = session.execute(query, (week_date, point_lat, point_lon))
```

```
lat_list = rows[0].lat
lon_list = rows[0].lon
index_list = []
flag = False
for i in range (0, len(lat_list)):
    if round(lat_list[i], 4)==point_lat and round(lon_list[i], 4)==point_lon:
        flag = True
        index_list.append(i)

print(flag)
if len(index_list) == 0:
    return "no data found"

else:
    form_data = {}
    form_data['time_data'] = []
    form_data['base'] = []
    for i in index_list:
        form_data['time_data'].append(rows[0].date_time[i])
        form_data['base'].append(rows[0].base[i])
```

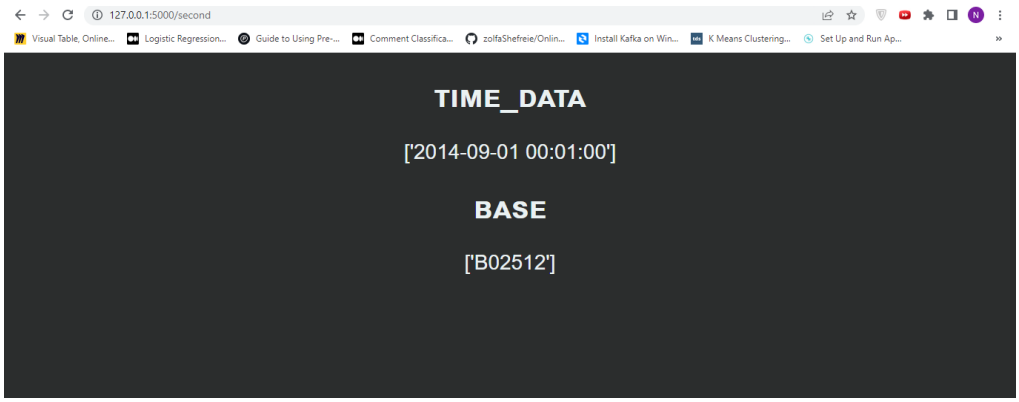
- **بازیابی داده‌ها در فلسک**

با استفاده از فلسک می‌توان داده‌ها را به‌صورت زیر بازیابی نمود. برای مثال با توجه به شکل زیر کاربر می‌تواند اطلاعات سفرهای انجام‌شده برای یک مختصات خاص را در بازه‌ی زمانی دلخواه به‌دست آورد.





برای مثال در بازه‌ی زمانی مشخص‌شده توسط کاربر تنها یک سفر با اطلاعات زیر وجود داشته است.



همچنین می‌توان سایر کوئری‌ها را نیز به همین صورت اجرا نمود. برای این کار لازم است یک تابع در فلسک ایجاد کرده و آدرس و متد مورد نظر را در آن مشخص نمود.

آدرس گیت‌هاب پروژه:

https://github.com/zolfaShefreie/Online-Taxi-Service