به نام خدا

گزارش تمرین سوم درس machine learning

امیرحسین میرزاده، ۹۶۲۲۲۰۸۲

توجه: سرویس‌ها در سرویس ابری فندق بارگزاری شده‌اند.

آدرس دسترسی سرویس‌ها: https://fandogh-service-a-mirzade.fandogh.cloud

برای پیاده‌سازی سرویس‌های موردنظر، از ای پی آی FLASK استفاده شده است.

توابع اصلی و سرویس‌ها در فایل app.py و توابع پیاده‌سازی در فایل functions.py، توابع مربوط به سیستم درون یابی در فایل interpolation_methods.py و توابع مربوط به تشخیص ناهنجاری و داده پرت در فایل outlier_and_anomoly_detection.py و در نهایت توابع مربوط به سرویس متوازن سازی داده ها، در فایل imbalanced_method.py پیاده سازی شده اند.

با ورود به دایرکتوری ریشه برنامه، پیام SYSTEM IS RUNNING! نمایش داده می شود.

هدف تمرین اول پر کردن داده های ناموجود در یک سری زمانی می باشد. سرویس اول در دایرکتوری Service_1 پیاده سازی شده است.

ورودی های تمرین اول به صورت فایل input.json است که در آن داده های سری زمانی ما و سپس تنظیمات (config) این داده ها قرار دارند. برای مثال:

```
{
  "data": {
    "time": {
      "0": "2022-01",
      "1": "2022-02",
      "2": "2022-04"
    },
    "vol": {
      "0": 20,
      "1": 40,
      "2": 100
    }
  },
  "config": {
    "type": "gregorian",
    "time": "monthly",
    "interpolation": "linear"
  }
}
```


که مقدار ۲۰ را در ماه اول سال ۲۰۲۲ نشان میدهد. همچنین تایپ تاریخ ما میلادی، دقت تاریخ تا ماه و نوع درون یابی ما خطی است. انواع ورودی ها با کانفیگ های مختلف در فایل input.json در دایرکتوری service_1 قرار دارد.

تابع ip_service()، برای درون یابی سری زمانی ورودی از فایل بالا استفاده می شود. ابتدا داده ها خوانده و ذخیره شده و سپس تابع interpolation از کلاس interpolation_methods روی آن ها اجرا می شود. ابتدا داده ها بر اساس کانفیگ زمانی شان (روزانه، ماهانه، ساعتی یا دقیقه ای) توسط پکیج پاندا resample شده و سپس، درون یابی مورد نظر (درون یابی اسپلاین و چندجمله ای نیز پیاده سازی شده است) روی آن ها اجرا می شود.

نمونه درون یابی شده ورودی بالا:


```
{
  "0": {
      "time": "2022-01-01T00:00:00Z",
      "vol": 20.0
  },
```


موارد امتیازی پیاده سازی شده: اجرا روی فندق، درون یابی خطی، اسپلاین و چندجمله ای، درون یابی داده های ساعتی و دقیقه ای

برای پیاده سازی سرویس دوم، از همان توابع سرویس اول استفاده می شود، با این تفاوت که ابتدا تاریخ ها به تاریخ جلالی (شمسی) تبدیل شده و سپس درون یابی برای پر کردن داده های ناموجود انجام می شود.

این سرویس در دایرکتوری service_2 پیاده سازی شده است. همان کارهای سرویس قبلی روی ورودی ها انجام می شود.

نمونه ورودی:


```
{
  "data": {
    "time": {
      "0": "2022-01",
      "1": "2022-02",
      "2": "2022-04"
    },
    "vol": {
      "0": 20,
      "1": 40,
      "2": 100
    }
  },
  "config": {
    "time": "monthly",
    "interpolation": "linear"
  }
```


تابع Gregorian_to_jalali در فایل functions تعریف شده است. نمونه خروجی این سرویس:


```
{
  "0": {
      "time": "1400-10-11 00:00:00",
      "vol": 20.0
  },
```


موارد امتیازی: اجرا روی سرویس ابری فندق، روش های درون یابی دیگر مانند اسپلاین، درون یابی داده های ساعتی و دقیقه ای

سرویس سوم ما، سرویس تشخیص ناهنجاری یا داده پرت است.

نمونه داده های ورودی:

```
{
  "time series":{
    "seasonal":
    {
      "data": {
        "time": {
          "0": "2022-01-01 00:00:00",
          "1": "2022-01-05 00:00:00",
          "2": "2022-01-10 00:00:00",
          "3": "2022-01-15 00:00:00",
          "4": "2022-01-20 00:00:00",
          "5": "2022-01-25 00:00:00",
          "6": "2022-01-30 00:00:00",
          "7": "2022-02-05 00:00:00",
          "8": "2022-02-10 00:00:00",
          "9": "2022-02-15 00:00:00",
          "10": "2022-02-20 00:00:00"
        },
        "vol": {
          "0": 100,
          "1": 200,
          "2": 100,
          "3": 200,
          "4": 100,
          "5":200,
          "6":100,
          "7":200,
          "8":1000,
          "9":200,
          "10":100
        }
      },
```

```
  "config": {
      "time_series": true,
      "method": "seasonal",
      "freq": 5
  }
},
```

بعد از خواندن داده های ورودی، تابع read_and_anomoly_detection از فایل function.py اجرا می شود. در صورتی که داده های ما از نوع داده سری زمانی باشند، تابع anomaly_detection و در غیر این صورت، تابع outlier_detection برای تشخیص داده پرت اجرا می شود.

این توابع در فایل outlier_and_anomoly_detection.py پیاده سازی شده اند. برای پیاده سازی این توابع، از پکیج adtk و برای استفاده از رگرسیون خطی، از پکیج sklearn استفاده شده است.

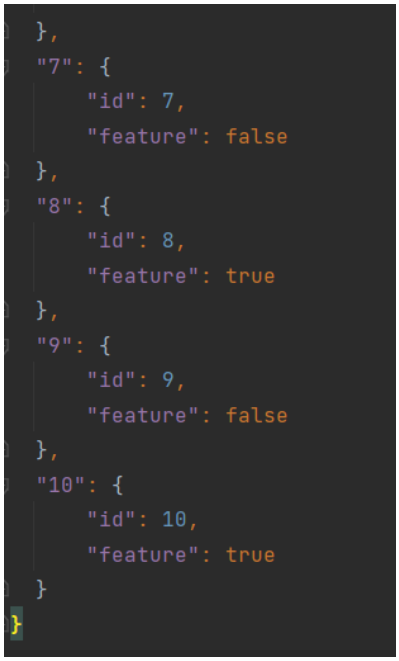
تابع anomaly_detection، با استفاده از پکیج adtk، با تنظیمات seasonal, threshold, persist و regressionAD کار تشخیص ناهنجاری را انجام می دهد.

تابع outlier_detection، با دو متد zscore و IQR کار تشخیص داده پرت را برای داده های غیر سری زمانی انجام می دهد. محاسبه zscore توسط پکیج stats انجام می شود (که برابر میانگین است). هر چه امتیاز زد یک مشاهده از ۰ دورتر باشد، داده پرت تری محسوب می شود. آستانه پرت بودن، داشتن فاصله ۳ از ۰ است.

متند IQR، داده هایی را که از یک و نیم برابر فاصله دو چارک اول و سوم دور تر از این چارک ها باشند، داده پرت محسوب می کند.

همچنین همین تابع، توسط تابع outlier_detector کلاس adtk به صورت کامنت (فعال نیست در کد) پیاده سازی شده است.

نمونه خروجی:



مشاهده می شود که ورودی های ۸ و ۱۰، داده پرت محسوب می شوند.

موارد امتیازی: پیاده سازی روی سرویس ابری فندق، اجرای چندین متد مختلف برای تشخیص داده پرت

سرویس چهارم، که در دایرکتوری service_4 پیاده سازی شده است، برای مدیریت و متوازن سازی داده های غیر متوازن به کار می رود.

تابع imbalanced با استفاده از تابع read_and_balance، و آن هم با استفاده از تابع balance_the_data که در فایل imbalanced_methods.py پیاده سازی شده است، کار متوازن سازی داده ها را انجام می دهد.

نمونه داده ورودی:

"RandomOverSample":
{
"data": {
"id": {
"0": 1,
"1": 2,
"2": 3,
"3": 4,

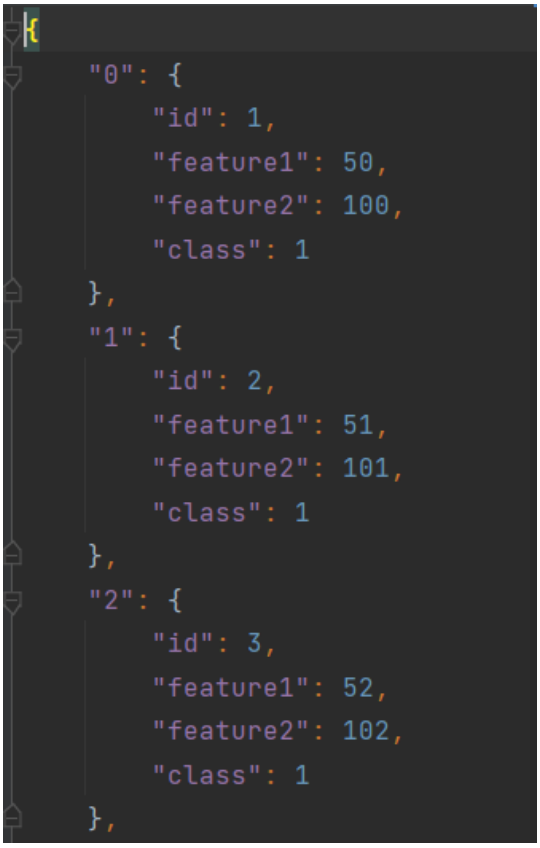
```
  "4": 5,
  "5": 6,
  "6":7,
  "7":8,
  "8":9,
  "9":10,
  "10":11,
  "11":12,
  "12":13,
  "13":14,
  "14":15,
  "15":16,
  "16":17,
  "17":18
},
"feature1": {
  "0": 50,
  "1": 51,
  "2": 52,
  "3": 53,
  "4": 54,
  "5": 150,
  "6": 151,
  "7":152,
  "8": 153,
  "9":154,
  "10":155,
  "11":156,
  "12":157,
  "13":158,
  "14":159,
  "15":160,
```

```
  "16":55,
  "17":160

},
"feature2":{
  "0": 100,
  "1": 101,
  "2": 102,
  "3": 103,
  "4": 104,
  "5": 200,
  "6":201,
  "7":202,
  "8": 203,
  "9": 204,
  "10": 205,
  "11": 206,
  "12": 207,
  "13": 208,
  "14": 209,
  "15": 210,
  "16":105,
  "17":211

},
"class": {
  "0": 1,
  "1": 1,
  "2": 1,
  "3": 1,
  "4": 1,
  "5": 0,
```

```
      "6":0,
      "7":0,
      "8":0,
      "9":0,
      "10":0,
      "11":0,
      "12":0,
      "13":0,
      "14":0,
      "15":0,
      "16":1,
      "17":1
    }
  },
  "config": {
    "method": "RandomOverSample",
    "class_name": "class"
  }
},
```


که با متد RandomOverSample پیاده سازی می شود. متدهای دیگر مانند smote, adasyn, randomundersample, clustercentroids و nearmiss نیز پیاده سازی شده اند. این توابع توسط کلاس imblearn پیاده سازی شده اند.

نمونه خروجی این سرویس:



موارد امتیازی پیاده سازی شده: اجرا روی سرویس ابری فندق، پیاده سازی چندین روش مدیریت داده نامتوازن متفاوت.