گزارش پروژه نهایی درس یادگیری ماشین – عرفان کرمی – ۹۸۲۲۲۰۷۹

موارد عمومی:

- فرمت تاریخ ورودی (چه شمسی و چه میلادی) برای داده های سری زمانی به صورت $Y-M-D$ خواهد بود که Y نشان دهنده سال، M نشان دهنده ماه و D نشان دهنده روز میباشد.
- $index$ نشان دهنده اندیس داده وروی است و از صفر شروع میشود و $volume$ نشان دهنده مقدار برای داده با اندیس مشخص شده است.
- . در قسمت $config$ برای فیلد هایی که با / جدا شده اند نشان دهنده مقادیر مختلفی است که فیلد مورد نظر میتواند داشته باشد
- قسمت $Data$ میتواند شامل هر تعداد رکوردی باشد این مورد با ". . ." نشان داده شده است. فقط در داده های سری زمانی باید در هر کانفیگ مختلف برای $time$ حداقل دو مقدار مختلف به ازای آن کانفیگ وجود داشته باشند تا بتوان بر روی داده ها یک خط درونیابی کرد. مثلا اگر کانفیگ روزانه انتخاب شد باید در داده های ورودی حداقل دو داده با روز های مختلف وجود داشته باشند.

۱. سرویس ۱ : در این سرویس هدف درونیابی داده های سری زمانی میباشد.
ورودی این سرویس یک رشته $JSON$ به فرمت زیر است :

```
{
    "data": {
        "time": {
            "index": "Y-M-D"
             ...

        },
        "vol": {
            "index": volume
             ...

        }
    },
    "config": {
        "type": "shamsi/miladi",
        "time": "daily/monthly",
        "interpolation": "linear"
    }
}
```

خروجی این سرویس در صورتی که درخواست ورودی صحیح باشد:

```
{
    "data": {
        "time": {
            "index": "Y-M-D"
             ...
        },
        "vol": {
            "index": volume
             ...
        }
    }
}
```

که داده های خروجی همان داده های درونیابی شده اند.

خروجی این سرویس در صورتی که درخواست ورودی صحیح نباشد:

```
{
    "data": "invalid type config!/invalid time config!"
}
```

که یکی از پیغام ها زمانی است که کانفیگ زمان نادرست باشد یا کانفیگ نوع نادرست باشد.

۲. سرویس۲ : در این سرویس هدف تبدیل داده های سری زمانی میلادی به داده شمسی و سپس درونیابی آنها میباشد.

ورودی این سرویس یک رشته $JSON$ به فرمت زیر است :

```
{
    "data": {
        "time": {
            "index": "Y-M-D"
             ...
        },
        "vol": {
            "index": volume
             ...
        }
    },
    "config": {
        "time": "daily/monthly",
        "interpolation": "linear"
    }
}
```

همانطور که مشخص است ورودی این سرویس کاملا مانند حالت سرویس قبلی است به جز اینکه داده های ورودی حتما باید از نوع میلادی باشند.

خروجی این سرویس در صورتی که درخواست ورودی صحیح باشد:

```
{
    "data": {
        "time": {
            "index": "Y-M-D"
             ...
        },
        "vol": {
            "index": volume
             ...
        }
    }
}
```

تفاوت این سرویس با سرویس قبل این است که تایم سری خروجی در فرمت شمسی میباشند

خروجی این سرویس در صورتی که درخواست ورودی صحیح نباشد:

```
{
    "data": "invalid time config!"
}
```

که مربوط به زمانی است که کانفیگ زمان به درستی وارد نشده باشد

۳. سرویس ۳: در این سرویس هدف کشف داده های پرت برای داده های سری زمانی و داده های جدولی میباشد که تنها داده های جدولی پیاده سازی شده اند و به علت عدم نصب موفق پکیج $fbprophrt$ پیاده سازی انجام نگرفت و در صورتی که داده ورودی سری زمانی باشد خروجی سرویس $null$ خواهد بود.

البته سرویس برای داده های جدولی قابلیت مدیریت هر تعداد فیچر را دارد

ورودی این سرویس یک رشته $JSON$ به فرمت زیر است :

```
{
    "data": {
      "id": {
        "index": index number,
        ...
      },
      "feature1": {
        "inedx": volume,
        ...
      },
      ...
      "featureN": {
        "inedx": volume,
        ...
      },
    },
    "config": {
      "time_series": false/true
    }
}
```

خروجی این سرویس در صورتی که داده ها سری زمانی باشند به صورت زیر خواهد بود :

```
{
    "data":null
}
```

خروجی این سرویس در صورتی که داده ها سری زمانی نباشند برای N فیچر به صورت زیر خواهد بود :

```
{
  "data": {
    "id": {
      "index": index number,
      ...
    },
    "IQR_method_feature1": {
      "index": false/true,
      ...
    },
    "zscore_method_feature1": {
      "index": false/true,
      ...
    },
    ...
    ,
    "IQR_method_featureN": {
      "index": false/true,
      ...
    },
    "zscore_method_featureN": {
      "index": false/true,
      ...
    }
  },
  "config": {
    "time_series": false
  }
}
```

که نشان میدهد هر کدام از روش ها فیچر مورد نظر را داده پرت تشخیص داده اند یا نه . که $false$ نشان دهنده تشخیص داده به عنوان داده معمولی و $true$ نشان دهنده تشخیص داده به عنوان داده پرت میباشد

متد هایی که برای تشخیص داده پرت استفاده شده اند عبارتند از IQR و $Z-score$ که هر کدام به صورت زیر اند :

- IQR: در این روش دامنه میان چارکی IQR ، چارک اول $Q1$ و چارک سوم $Q3$ تعیین میشوند و سپس یک آستانه بالا و آستانه پایین محاسبه میشوند و اگر مقداری خارج از این دو آستانه قرار گرفت به عنوان داده پرت شناسایی میشود.

$$IQR = Q1 - Q3$$

Lower Bound: (Q1 − 1.5 ∗ IQR)
Upper Bound: (Q3 + 1.5 ∗ IQR)

- $Z - score$ در این روش برای هر فیچر مقدار میانگین μ و انحراف معیار σ محاسبه میشوند حال برای هر رکورد اگر مقدار فیچر مربوطه آن x خارج از محدوده زیر قرار بگیرد به عنوان داده پرت شناسایی میشود.

$$-3 \leq \frac{x - \mu}{\sigma} \leq 3$$

۴. هدف این سرویس مدیریت داده های نامتوازن میباشد و از سه متد $oversampling$ ، $undersampling$ و $SMOTE$ که در پکیج $imbalance - learn$ پیاده سازی شده اند استفاده شده است. این سرویس امکان هندل کردن بیش از دو کلاس را هم دارا میباشد.
ورودی این سرویس یک رشته $JSON$ به فرمت زیر است :

```
{
    "data": {
      "id": {
        "index": index number,
        ...
      },
      "feature1": {
        "index": volume,
        ...
      },
      ...
      "featureN": {
        "index": volume,
        ...
      },
      "class": {
        "index": class tag,
        ...
      }
    },
    "config": {
      "major_class": major_class tag,
      "minor_class": minor_class tag,
      "method": "SMOTE/under sampling/over sampling"
    }
}
```

که خروجی داده های مدیریت شده در قالب دیتاست جدید به فرمت زیر هستند :

```
{
    "data": {
      "id": {
        "index": index number,
        ...
      },
      "feature1": {
        "index": volume,
        ...
      },
      ...
      "featureN": {
        "index": volume,
        ...
      },
      "class": {
        "index": class tag,
        ...
      }
    }
}
```