با سلام

چهار پروژه در قالب چهار فایل ایجاد شده اند. فایل requirements هر پروژه تکمیل هست. هر پروژه دارای یک داکر فایل هست و قابلیت پیاده سازی بر سرویس ابری را دارد. فایل ها با pycharm تولید و ایجاد شده اند. هر پروژه دارای swagger مخصوص به خود است. و اگر بر لوکال هاست اجرا شوند در هر سرویس با زدن <http://127.0.0.1:5000/report> به صفحه swagger هدایت میشوید. هر پروژه در virtual environment خود ایجاد شده است.

------------------------------------------------------------------------------

سرویس اول

ورودی ها و خروجی ها:

ورودی در سرویس اول به دو صورت میتواند باشد صورت اول آن هست که تاریخ ها به میلادی باشند و صورت دوم آن هست که تاریخ ها شمسی باشند. مثالی از صورت اول->

{  
 "data": {  
 "time": {  
 "0": 1577836800000,  
 "1": 1577923200000,  
 "2": 1578096000000  
 },  
 "vol": {  
 "0": 20,  
 "1": 40,  
 "2": 100  
 }  
 },  
 "config": {  
 "type": "miladi",  
 "time": "daily",  
 "interpolation": "linear",  
 "tool": "Pandas"  
 }  
}

type داخل فیلد config میتواند “miladi” یا “shamsi” باشد. و time داخل فیلد config میتواند “daily” یا “monthly” یا “hourly” یا “minutes” یا “seconds” باشد. و interpolation داخل فیلد config میتواند “linear” یا “polynomial” یا “spline” باشد. و tool دخل فیلد config میتواند “Pandas” یا “Dask” باشد. بر اساس فیلد های config برنامه از pandas یا Dask بهره میبرد. برنامه میتواند با تاریخ های شمسی و میلادی اجرا شود. اگر تاریخ شمسی باشد باید حتما به فرمت خاصب ورودی دهیم در زیر مثالی از ورودی با تاریخ شمسی را مینویسم.

{  
 "data": {  
 "time": {  
 "0": "1390-11-14 10:10:10.99",  
 "1": "1390-11-15 11:11:11.99",  
 "2": "1390-11-16 12:12:12.89",  
 "3": "1390-11-17 13:12:12.78"  
 },  
 "vol": {  
 "0": 20,  
 "1": 40,  
 "2": 70,  
 "3": 100  
 }  
 },  
 "config": {  
 "type": "miladi",  
 "time": "daily",  
 "interpolation": "linear",  
 "tool": "Pandas"  
 }  
}

در خروجی این سرویس فیلد config نیست و زمان ها با عنوان vol ها وجود دارند.

توضیحاتی راجع به خود کد: در فایل app که قسمت اصلی برنامه است. در ابتدا کد های مربوط به swagger را میبینیم که فایل report.yml داخل پروژه مربوط به swagger است. بوسیله تابع read\_json\_time\_series ما json ورودی را به دیتا فریم پانداز تبدیل میکنیم اگر زمان ورودی به میلادی باشد این تابع تنها میاید زمان ها را به کلاس datetime میبرد و دیتا فریم را خروجی میدهد اما اگر زمان ها به شمسی باشند این تابع توسط تابع دیگری داخل فایل common به نام convert\_shamsi\_strings\_to\_datetime تمام زمان ها را به حالت datetime میبرد و دیتا فریم را خروجی میدهد.

در ادامه در فایل app دو حالت در نظرگرفته شده است اگر خروجی ها بخواهند میلادی باشد آنگاه ما دیتا فریم را به تابع interpolate در فایل commom میدهیم این تابع با توجه به فیلد های config میفهمد که ما خواستار کدام نوع از درونیابی هستیم و بر اساس نوع درونیابی دیتا فریم ما را به یکی از توابع درونیابی داخل فایل interpolation\_methods میفرستد. حال بعد از درونیابی دیتا فریم به جیسون تبدیل و در خروجی نمایش داده میشود اما اگر بخواهیم خروجی شمسی باشد باز هم ابتدا دیتا فریم را به تابع interpolate میدهیم و بعد با تابع convert\_dattime\_to\_JalaliDatetime داخل فایل common تاریخ ها را به نوعی استرینگ شمسی تبدیل میکنیم و بعد هم آن دیتا فریم را به جیسون میبریم و خروجی میدهیم.

سرویس دوم

ورودی در سرویس دوم به یک صورت میتواند باشد و تاریخ های ورودی باید به میلادی باشند مثالی از ورودی->

{  
 "data": {  
 "time": {  
 "0": 1328227200000,  
 "1": 1328400000000,  
 "2": 1328486400000  
 },  
 "vol": {  
 "0": 20,  
 "1": 40,  
 "2": 100  
 }  
 },  
 "config": {  
 "time": "daily",  
 "interpolation": "linear",  
 "skip\_holiday": false,  
 "tool": "Pandas"  
 }  
}

time داخل فیلد config میتواند “daily” یا “monthly” یا “hourly” یا “minutes” یا “seconds” باشد. و interpolation داخل فیلد config میتواند “linear” یا “polynomial” یا “spline” باشد. و tool دخل فیلد config میتواند “Pandas” یا “Dask” باشد و در آخر قابلیت skip\_holiday میتواند true یا false باشد. بر اساس فیلد های config برنامه از pandas یا Dask بهره میبرد. اگر قابلیت skip\_holiday ما true باشد قبل از درونیابی روز های پنجشنبه و جمعه از دیتا فریم حذف میشوند.

در خروجی این سرویس فیلد config نیست و تنها زمان و مقادیر متناظر با عنوان vol وجود دارند.

توضیحاتی درباره خود کد: در فایل app که قسمت اصلی برنامه است. در ابتدا کد های مربوط به swagger را میبینیم که فایل report.yml داخل پروژه مربوط به swagger است. بوسیله تابع read\_json\_time\_series ما json ورودی را به دیتا فریم پانداز تبدیل میکنیم چون زمان ورودی به میلادی است این تابع تنها میاید زمان ها را به کلاس datetime میبرد و دیتا فریم را خروجی میدهد .در ادامه در فایل app دیتا فریم را به تابع skip\_holidays در فایل common میدهیم اگر skip\_holidays در فیلد config فعال باشد. این تابع ابتدا توسط تابعی به نام convert\_datetime\_to\_JalaliDatetime\_for\_skip\_holiday\_version میاید و تمام زمان های داخل دیتا فریم را به فرمت JalaliDatetime میبرد در ادامه توسط پکیج خیام روزه های پنجشنبه و جمعه را از دیافریم حذف میکند. و با تابع convert\_JalaliDatetime\_to\_datetime\_for\_skip\_holiday\_version میاید و تمام زمان ها را به حالت datetime میبرد و دیتا فریم حاصله را خروجی میدهد. اگر هم skip\_holiday داخل فیلد config فعال نباشد تابع skip\_holidyas تنها با تابع convert\_datetime\_to\_JalaliDatetime\_for\_skip\_holiday\_version زمان ها را datetime میبرد و در ادامه بدون هیچ تغییری با تابع convert\_JalaliDatetime\_to\_datetime\_for\_skip\_holiday\_version زمان ها را به حالت datetime برمیگرداند و دیتا فریم حاصل خروج می یابد. در ادامه فایل app دیتا فریم را به تابع interpolate در فایل commom میدهیم این تابع با توجه به فیلد های config میفهمد که ما خواستار کدام نوع از درونیابی هستیم و بر اساس نوع درونیابی دیتا فریم ما را به یکی از توابع درونیابی داخل فایل interpolation\_methods میفرستد. حال بعد از درونیابی با تابع convert\_dattime\_to\_JalaliDatetime داخل فایل common تاریخ ها را به نوعی استرینگ شمسی تبدیل میکنیم و بعد هم آن دیتا فریم را به جیسون میبریم و خروجی میدهیم.

سرویس سوم

ورودی ها و خروجی ها: در این سرویس یکسری داده داریم که هر کدام یک feature دارند بعد با دو متد isolation forest (متد اول) ومتد LOF (متد دوم) می آییم و داده های پرت را شناسایی میکنیم فیلد time\_series در config میتواند true یا false باشد بر حسب آنکه ورودی ما سری زمانی یا داد عادی باشد دو گونه از ورودی ها را شاهد خواهیم بود در حال سری زمانی زمان ها باید میلادی و به صورت عددی باشند.

مثال ورودی سری زمانی:

{  
 "data": {  
 "time": {  
 "0": 1577836800000,  
 "1": 1577923200000,  
 "2": 1578096000000,  
 "3": 1579096000000  
 },  
 "feature": {  
 "0": 100,  
 "1": 20,  
 "2": 35,  
 "3": 67,  
 "4": 89,  
 "5": 90  
 }  
 },  
 "config": {  
 "time\_series": true  
 }  
}

مثال ورودی عادی:

{  
 "data": {  
 "id": {  
 "0": 1,  
 "1": 2,  
 "2": 3,  
 "3": 4,  
 "4": 5,  
 "5": 6  
 },  
 "feature": {  
 "0": 100,  
 "1": 20,  
 "2": 35,  
 "3": 67,  
 "4": 89,  
 "5": 90  
 }  
 },  
 "config": {  
 "time\_series": false  
 }  
}

خروجی ما فیلد config را ندارد و در آن بجای ویژگی ها دو فیلد متد اول و متد دوم داریم که درآنها به ازای هر داده یا زمان مشخص کرده ایم که آن داده یا زمان پرت هست یا نه اگر میزان متناظر با آن داده یا زمان داخل متد اول true بود به معنای آن هست که متد اول آن داده را پرت شناسایی کرده است.

مثالی از خروجی:

{  
 "data": {  
 "id": {  
 "0": 1,  
 "1": 2,  
 "2": 3,  
 "3": 4,  
 "4": 5,  
 "5": 6  
 },  
 "method1": {  
 "0": false,  
 "1": false,  
 "2": false,  
 "3": false,  
 "4": false,  
 "5": false  
 },  
 "method2": {  
 "0": true,  
 "1": false,  
 "2": false,  
 "3": true,  
 "4": true,  
 "5": false  
 }  
 }  
}

در خروجی حالت هایی که ورودی سری زمانی بوده است زمان ها به صورت میلادی یا integer هستند.

توضیحاتی درباره خود کد برنامه:

در فایل اصلی برنامه یا app در ابتدا کد های مربوط به swagger را میبینیم و فایل report.yml داخل فولدر پروژه مربوط به swagger است. در تابع اصلی، برنامه با تابع read\_json جیسون ورودی را میخواند و داخل یک دیتا فریم میریزد و خروجی را میدهد. در ادامه یک کپی از دیتا فریم گرفته میشود و نسخه اول دیتافریم به تابع isolation\_forest در فایل anomaly\_detection\_methods داده میشود در این تابع اگر داده سری زمانی باشد ایندکس آن را به time و اگر داده عادی باشد ایندکس آن را به id میبرد بعد الگوریتم بر روی دیتا فریم اجرا میشود و یک ستون به دیتا فریم افزوده میشود که داده های پرت مقدار منفی یک را در آن ستون میگیرند در ادامه این تابع به جای مقادیر منفی یک true و به جای مقادیر غیر منفی یک false میگذارد و همچنیم اسم ستون را به method1 تغییر میدهد و دیتا فریم کلی را خروجی میدهد. در ادامه نسخه دوم دیتا فریم اولی به تابع LOF داده میشود. این تابع اگر داده سری زمانی باشد ایندکس آن را به time و اگر داده عادی باشد ایندکس آن را به id میبرد بعد الگوریتم بر روی دیتا فریم اجرا میشود و یک ستون به دیتا فریم افزوده میشود که داده های پرت مقدار منفی یک را در آن ستون میگیرند در ادامه این تابع به جای مقادیر منفی یک true و به جای مقادیر غیر منفی یک false میگذارد و همچنیم اسم ستون را به method2 تغییر میدهد در ادمه فایل app برنامه می آید و دو تا دیتا فریم حاصله از دو الگوریتم را با هم ترکیب میکند و method1 و method2 را در یک جیسون میگذارد و خروجی میدهد.

سرویس چهارم

ورودی ها و خروجی ها:

در ورودی این تابع یکسری داده همرا با یک یا چند feature هستند در فیلد data بخشی به نام class هست که میگوید هر کدام از داده های متناظر در کلاس صفر هستن یا در کلاس یک. در فیلد config major\_class کلاس اکثریت و minor\_class کلاس اقلیت را نمایش میدهند. فیلد متد میتواند “SMOTE” یا “RANDOMOVERSAMPLING” یا “RANDOMUNDERSAMPLING” باشد.

مثالی از ورودی:

{  
 "data": {  
 "id": {  
 "0": 1,  
 "1": 2,  
 "2": 3,  
 "3": 4,  
 "4": 5,  
 "5": 6  
 },  
 "feature1": {  
 "0": 50,  
 "1": 12,  
 "2": 50,  
 "3": 500,  
 "4": 60,  
 "5": 12  
 },  
 "class": {  
 "0": 1,  
 "1": 1,  
 "2": 1,  
 "3": 1,  
 "4": 0,  
 "5": 0  
 }  
 },  
 "config": {  
 "major\_class": 1,  
 "minor\_class": 0,  
 "method": "SMOTE"  
 }  
}

نکته مهم: متد SMOTE هر کلاس باید حداقل دو عضو داشته باشد.

خروجی بخش config را ندارد و بنابر نوع متد انتخابی تعداد داده کمتر یا بیشتری دارد

توضیحاتی راجع به خود کد:

در فایل اصلی برنامه یا app در ابتدا کد های مربوط به swagger را میبینیم و فایل report.yml داخل فولدر پروژه مربوط به swagger است. در این فایل ابتدا به وسیله تابع read\_json داخل فایل common جیسون وردی به دیتافریم تبدیل میشود . در ادامه با توجه به فیلد method داخل بخش config دیتا فریم مربوطه به تابع الگوریتم متناظر در داخل فایل functions هدایت میشود. و بعد از عملیات imbalancing با توجه به کم شدن یکسری از سطر ها یا زیاد شدن آن ها و به هم ریختن بخش “id” داخل قسمت “data” ایندکس بندی این قسمت توسط تابع cleaning\_result دوباره انجام میشود تا ترتیب id ها از یک تا تعداد داده ها باشد.