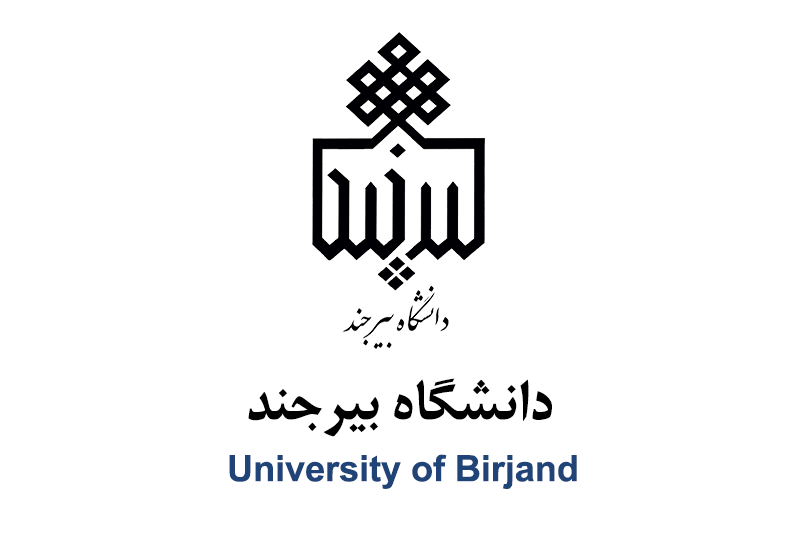
بسم الله الرحمان الرحیم



رشته کامپیوتر-گرایش نرم‌ افزار

کارشناسی پیوسته

موضوع پروژه:

پیش‌بینی ارزان ترین قیمت بلیت هواپیما در بازه زمانی مشخص

استاد راهنما:

امیر کیوان شفیعی

دانشجو:

مصطفی وحدانی دهکلانی

سال تحصیلی:

۱۴۰۱-۱۴۰۲

**فهرست مطالب**

[**​ مقدمه 3**](#__RefHeading___Toc454_1174935635)

[**​ فصل ۱ 4**](#__RefHeading___Toc456_1174935635)

[**​ معرفی پروژه و اهداف 4**](#__RefHeading___Toc458_1174935635)

[**​ مروری بر دیتاست ۴**](#__RefHeading___Toc1167_1374526161)

[**​ ساختار دایرکتوری ها ۵**](#__RefHeading___Toc460_1174935635)

[**​ کمی به ساختار عمیق‌تر بنگریم… 8**](#__RefHeading___Toc1169_1374526161)

[**​ ماژول ها ۹**](#__RefHeading___Toc462_1174935635)

[**​ مابقی فایل‌ها ... ۱**](#__RefHeading___Toc1171_1374526161)**۱**

[**​ چگونه از پروژه اجرا بگیریم؟ 12**](#__RefHeading___Toc1173_1374526161)

[**​ فصل ۲ 14**](#__RefHeading___Toc464_1174935635)

[**​ تکنولوژی ها و ابزارهای به کار رفته 14**](#__RefHeading___Toc466_1174935635)

[**​ کتابخانه‌های داخلی زبان پایتون 15**](#__RefHeading___Toc5477_2184025443)

[**​ کتابخانه و فریمورک های مشترک 16**](#__RefHeading___Toc5479_2184025443)

[**​ کتابخانه‌ها و فریمورک های زیر پروژه flight\_tickets\_scraper 16**](#__RefHeading___Toc5481_2184025443)

[**​ کتابخانه‌ها و فریمورک های زیر پروژه flight\_tickets\_preprocessing 18**](#__RefHeading___Toc5483_2184025443)

[**​ ابزارهای مرتبط استفاده شده در طول پروژه 20**](#__RefHeading___Toc5485_2184025443)

[**​ API های استفاده شده در پروژه 21**](#__RefHeading___Toc5487_2184025443)

[**​ فصل ۳ 22**](#__RefHeading___Toc468_1174935635)

[**​ عملکرد و توضیح کد ها 22**](#__RefHeading___Toc470_1174935635)

[**​ فاز اول پروژه 22**](#__RefHeading___Toc5489_2184025443)

[**​ فاز دوم پروژه 52**](#__RefHeading___Toc5491_2184025443)

[**​ فاز سوم پروژه 7**](#__RefHeading___Toc5493_2184025443)**4**

[**​ فاز چهارم پروژه 7**](#__RefHeading___Toc5495_2184025443)**9**

[**​ فصل ۴ 8**](#__RefHeading___Toc1175_1374526161)**5**

[**​ خروجی های پروژه و اصلاحات مورد نیاز 8**](#__RefHeading___Toc474_11749356351)**5**

[**​ خروجی ها 8**](#__RefHeading___Toc6132_2184025443)**5**

[**​ اصلاحات 8**](#__RefHeading___Toc6134_2184025443)**6**

[**​ فصل ۵ 8**](#__RefHeading___Toc476_1174935635)**8**

[**​ جمع‌بندی و نتیجه‌گیری 8**](#__RefHeading___Toc478_1174935635)**8**

[**​ منابع 8**](#__RefHeading___Toc6136_2184025443)**9**

# مقدمه

اگر در تاریخ بنگریم از تمدن کهن گرفته که ساختار نوشتاری خاصی نداشتند و اطلاعات را با تعریف داستان در قالب افسانه‌ها به نسل های بعدی منتقل می کردند تا تمدن مدرن دیجیتال که اطلاعات را در بستر اینترنت و یا در قالب کتاب‌ها می‌نویسند و یا همین امروز که شما شروع به خواندن این داکیومنت کرده اید تا بوده، داشتن اطلاعات و طریقه استفاده کردن از آن‌ها همیشه مهم‌ترین بخش زندگی ما را تشکیل می‌دادند و نقش حیاتی را در اتخاذ تصمیم های شخصی روزانه و رویداد های تاریخی ایفا کرده‌اند و خواهند کرد ولی فقط و فقط در عصر امروزی است که اکثریت جامعه متوجه اهمیت به سزای آن در داشتنش شده اند. از نقش حیاتی استفاده از اطلاعات در تاریخ می‌توان به جنگ انقلابی آمریکا که نشان دهنده مبارزه سیزده مستعمره آمریکا علیه سلطه بریتانیا بود که در نهایت منجر به تولد ایالات متحده آمریکا شد اشاره کرد. جمع آوری، انتشار و مدیریت اطلاعات نقشی اساسی در شکل دادن به روند و نتیجه این جنگ داشت. هر دو طرف درگیری نیز از تبلیغات و گسترش شایعه برای شکل دادن به افکار عمومی استفاده می کردند. این تنها یکی از هزاران نمونه تاریخی است که در آن داشتن اطلاعات نقش مهمی را ایفا کرده است. اما شما نیز از داشتن اطلاعات در زندگی روزمره خودتان بهره برده‌اید یه مثال ساده می‌تواند این باشد که فرض کنید می‌خواهید با خانواده خود به یک سفر تفریحی با هواپیما به شهری دیگر بروید برای شما باید مهم باشد که شرایط آب و هوایی در شهر مقصد چگونه است و یا چگونه می‌توانید یک بلیت ارزان‌قیمت را در یک بازه زمانی مشخص خریداری کنید این دقیقاً کاری است که ما می‌خواهیم توسط داشتن اطلاعات در این پروژه با کمک علم داده (Data Science) انجام دهیم. امروزه داده‌ها در جهان به شدت رو به افزایش هستند. باید بدانید که داده در عصر امروزی به معنای واقعی، یعنی همه چیز. همه چیز ما یک داده هست. پیشنینه ما، آگاهی ما، داشته های ما، تاریخچه مرورگر ما، شماره های مخاطبین گوشی ما، پیام‌های درون پیام رسان های ما، دارایی ما، دانستنی های ما، علایقی که دنبال می‌کنیم و غیره همه و همه یک نوع داده محسوب می شوند. به طور کلی از طریق جمع آوری این داده‌ها می‌توان اطلاعات آماری خاصی را استخراج کرده و از آن‌ها در زمینه‌های متفاوت در جهت اهداف خیر و یا خصمانه بهره برد و یا اینکه با آن نیاز روزمره افراد را برطرف کرد. علم داده، ابزارها و روش‌هایی را به ما پیشنهاد می‌دهد که به ما در استخراج این اطلاعات آماری به بهترین شکل ممکن کمک خواهد. این ابزارها، روش‌ها، الگوریتم ها و مدل ها به پیدا کردن الگوهای پنهان درون دیتاست های خام جمع آوری شده می پردازند. باید اشاره کرد، به اشخاصی که در زمینه علم داده فعالیت می‌کنند دیتا ساینتیست (Data Scientist)، متخصص داده و یا دانشمند داده گفته می‌شوند و لازم است قبل از اینکه شما یک دیتا ساینتیست باشید حداقل در مباحتی مثل ریاضیات، آمار و احتمال، برنامه نویسی، پایگاه داده ها، یادگیری ماشین و مدل سازی تسلط کافی داشته باشید تا بتوانید قدم در دنیای بزرگ علم داده بگذارید.

# فصل **۱**

## معرفی پروژه و اهداف

پروژه ایی که در پیش روی شما قرار دارد پروژه تحلیل قیمت بلیت هواپیما در یک بازه زمانی مشخص می باشد. **تحلیل قیمت بلیت هواپیما برای پیدا کردن ارزان ترین قیمت در یک بازه زمانی** معمولاً کار سخت و دشواری است زیرا که قیمت هواپیما در لحظه دچار تغییر و نوسانات می‌باشد. قیمت بلیت هواپیمایی که امروز می‌خرید می‌تواند با قیمت بلیت هواپیمایی مشابه که فردا می‌خرید بسیار متفاوت باشد و این نوسانات به سیاست‌های جداگانه هر شرکت هواپیمایی، آژانس های چارتر کننده بلیت، مسیرهایی متفاوت هواپیما برای مقاصد یکسان، فصل سال و غیره بستگی دارد. دیتاست **خام** جمع آوری شده دارای داده‌های سطری از تاریخ ۱۴۰۲/۰۵/۱۷ تا تاریخ ۱۴۰۲/۰۷/۱۵ در ۱۴۷۵۲ سطر و ۱۵ ستون و همچنین دیتاست **پیش پردازش شده** دارای داده‌های سطری از تاریخ ۱۴۰۲/۰۵/۱۷ تا تاریخ ۱۴۰۲/۰۷/۱۵ در ۱۳۹۴۷ سطر و ۱۶ ستون می‌باشد و نیر دیتاست **نهایی** ما دارای داده‌های سطری از تاریخ ۱۴۰۲/۰۵/۱۷ تا تاریخ ۱۴۰۲/۰۷/۱۵ در ۱۳۹۴۷ سطر و ۱۱ ستون می‌باشد. البته باید به این نکته نیز اشاره کرده که پروژه ما در مبحث یادگیری ماشین از نوع رگرسیون می‌باشد زیرا که ما میخواهیم یک مقدار پیوسته (هزینه بلیت) را پیش‌بینی کنیم پس باید توجه کنیم که معیارهای ارزیابی ما نیز باید بررسی میزان خطای موجود به ازای هر پیش‌بینی باشد که در این مورد می‌توان از روش‌های ارزیابی MSE و یا MAE استفاده کرد. گفتنی است که این پروژه در چهار فاز تهیه شده است و روند انجام آن به ترتیب زیر می باشد:

1. کراول کردن سایت هدف در جهت جمع آوری یک دیتاست خام. در این پروژه از سایت tcharter.ir که یک موتور جستجو بلیت های هواپیما می‌باشد استفاده کرده ام.
2. پیش پردازش داده‌های خام و آماده سازی یک دیتاست تمیز شده.
3. آماده کردن داده‌های تمیز شده برای مدل یادگیر (اعمالی مثل نرمال سازی، تبدیل داده ها و غیره)
4. آموزش دادن مدل و پیش‌بینی قیمت.

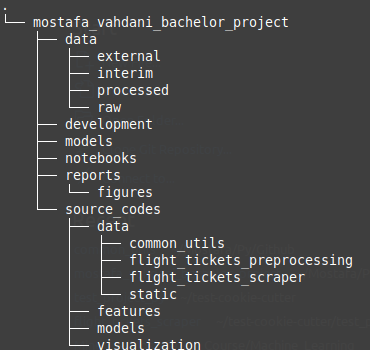
برای دریافت پروژه به صورت clone شده نیز می‌توانید با کلیک برروی [Mostafa-mvd/Flight-Tickets-DataScience-Py](https://github.com/Mostafa-mvd/Flight-Tickets-DataScience-Py) به لینک گیت هاب پروژه بنده مراجعه کنید. اگر بعد از مطالعه داکیومنت، این پروژه موجب خرسندی شما شده بسیار سپاس گزار خواهم بود که با یک **ستاره** پروژه را در گیت هاب حمایت کنید.

## مروری بر دیتاست

1. **national\_departure\_code** : کد استاندارد بین‌المللی برای شناسایی فرودگاه مقصد. (سه حرفی می باشد)
2. **departure\_city** : اسم شهر مقصد.
3. **departure\_airport** : اسم فرودگاه مقصد.
4. **national\_arrival\_code** : کد استاندارد بین‌المللی برای شناسایی فردوگاه مبدا. (سه حرفی می باشد)
5. **arrival\_city** : اسم شهر مبدا.
6. **arrival\_airport** : اسم فردوگاه مبدا.
7. **orthodromic\_distance\_KM** : فاصله دایره بزرگ (Great-circular distance) بین شهر مبدأ و مقصد. به عبارتی فاصله مستقیم بین شهر مبدأ و مقصد می باشد(با فاصله جاده ایی تفاوت دارد.) این فاصله به کیلومتر می‌باشد که با KM در نام ستون مشخص شده است.
8. **flight\_length\_min** : مقدار زمانی که طول می‌کشد تا پرواز تمام شود که واحد آن به دقیقه (min) می باشد.
9. **departure\_date\_YMD\_format** : تاریخ حرکت هواپیما از مقصد که به فرمت Year-Month-Day می باشد.
10. **local\_departure\_time** : ساعت محلی حرکت هواپیما از مقصد.
11. **local\_arrival\_time** : ساعت محلی رسیدن هواپیما به مبدا.
12. **company\_name** : اسم شرکت هواپیمایی یا همان ایرلاین.
13. **flight\_number** : شماره پرواز. (با این شماره می‌توان به اطلاعات پرواز از جمله ساعت حرکت، ساعت رسیدن، نوع بلیت و غیره دسترسی داشت)
14. **flight\_sale\_type** : نوع فروش بلیت هواپیمایی که به صورت چارتری یا سیستمی می باشد.
15. **fare\_class\_code** : کد **کلاس نرخی** بلیت هواپیمایی. (کد کلاس نرخی بعضی شرکت های هواپیمایی می‌تواند یکسان ولی مفهوم آن متفاوت از یکدیگر باشد.) این کلاس نرخی می‌تواند با توجه به ایرلاین مربوطه مشخص کند که **کلاس پروازی** شما از چه نوعی است اکونومی، بیزنس، اکونومی پرمیوم و یا فرست.
16. **ticket\_price\_T** : هزینه بلیت هواپیمایی که واحد آن به تومان(T) می باشد.
17. **flight\_capacity** : مقدار ظرفیت موجود برای هر بلیت هواپیمایی می باشد.
18. **ticket\_id** : مقدار id برای هر بلیت هواپیمایی یا همان داده سطری.

## ساختار **دایرکتوری** ها

بعد از معرفی پروژه و اهداف آن حال به توضیح ساختار پروژه و دایرکتوری ها (فولدر یا همان پوشه) می‌پردازیم تا متوجه شویم هر قسمت از ساختار، مربوط به چه فعالیت و عملی می باشد. ساختار کلی پروژه ما بدون در نظر گرفتن ماژول های پایتونی و دیگر فایل ها به شرح عکس زیر می باشد.



همانطور که مشاهد می‌کنید دایرکتوری روت و اصلی پروژه به نام mostafa\_vahdani\_bachelor\_project می‌باشد که زیر دایرکتوری های آن عبارت اند از:

1. **data** : داده‌هایی که مرتبط با بخش Train و Predict پروژه ما باشند در این دایرکتوری قرار خواهند گرفت مثل دیتاست های پردازش شده، تمیزه شده و غیره.

* external : دیتاهایی که از طریق Third Party یا به فارسی همان شخص ثالت بدست آمده در این پوشه قرار خواهند گرفت. به عبارت ساده‌تر دیتاهایی خارجی ای که توسط خود شما تولید نشده‌اند را در این مسیر قرار دهید.
* interim : داده‌هایی که در اواسط کار بر روی ماشین تولید می‌شوند در این پوشه قرار خواهند گرفت .
* processed: مکان قرارگیری تمام داده‌های پیش پردازش شده و تمیز شده در این دایرکتوری می باشد.
* raw : داده‌های خامی را که از مرحله Scraping یا هر جای دیگر بدست آمده‌اند و یا به طور کلی داده‌هایی که هنوز آماده نیستند تا برای آموزش ماشین استفاده شوند را در این پوشه قرار می دهیم.

1. **development** : در این دایرکتوری فایل‌های توسعه و یا فایل‌ پیش نیازهای پروژه قرار خواهند گرفت مثل requirements.txt و یا در صورت لزوم فایل‌های Docker
2. **models :** مدل Train شده یا Serialized شده ما و یا خلاصه‌ای از مدل نهایی در اینجا قرار می گیرند.
3. **notebooks** : این دایرکتوری برای بخش یادگیری ماشین و اعمالی مثل پیش بینی، آموزش و پیش پردازش بسیار مهم می‌باشد زیرا که استفاده از ابزاری مثل Jupyter کار را برای ما آسان‌تر می کند(هرچند باید توجه شود که باز هم به دانش برنامه نویسی مثل Python همچنان نیاز است زیرا که ژوپیتر صرفاً یک محیط برای اجرا گرفتن از کد شما مثل Google Colab می باشد). در این پروژه فایل‌های مربوط به ژوپیتر با فرمت اسمی مشخص و سراسری قرار خواهند گرفت. به طور مثال فرمت اسمی زیر را در نظر بگیرید:

1.1-arrival-time-data-preprocessing.ipynb

عدد 1.1 ترتیب استفاده از فایل‌های ژیپیتر را به شما می‌گوید یعنی اگر فایلی در این دایرکتوری بود که با عدد ۱.۳ شروع می‌شد یعنی اینکه باید بعد از ۱.۱ از آن استفاده کرد و اگر به فرض ۱.۰ بود باید قبل از ۱.۱ از آن استفاده کرد. در ادامه برای اینکه نشان دهیم چه کاری را در چه مرحله ایی از یادگیری ماشین داریم انجام می‌دهیم توضیحات آنرا به صورت مختصر و مفید با ”-” از یکدیگر جدا می‌کنیم.

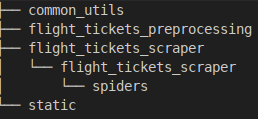
مثال بالا را در نظر بگیرید ما از کلمه arrival-time استفاده کرده‌ایم یعنی اینکه می‌خواهیم در این فایل ژوپیتر روی خروجی arrival-time کار کنیم و این فعالیت را در مرحله data-preprocessing داریم انجام می دهیم.

1. **report** : گزارش های تحلیلی نهایی خود را می‌توانید با تایپ های متفاوت مثل PDF ، HTML ، Figures ، Graphical و غیره را در این دایرکتوری تولید کنید و به افراد نشان دهید.
2. **source\_codes** : تمامی **کدهای اصلی** شما که می‌توانند شامل Scraping ، Pre-Processing ، Visualization و غیره می‌شود در این پوشه قرار می گیرند.

* data : در ابتدا باید توجه شود که این پوشه data با پوشه دیتایی که در بالا توضیح داده شد بسیار تفاوت دارد و یکسان نیستند پوشه دیتا بالایی برای دیتاست های تولید شده ما در هر مرحله از یادگیری ماشین بوده‌اند و این پوشه ی دیتایی که در مسیر source\_codes قرار دارد پوشه ایی است که تمامی کدهای ما برای جمع آوری و تولید دیتاست خام از طریق Scraping و انجام اعمال پیش پردازش بر روی دیتاست خام در آن قرار دارند. جلوتر به توضیح بیشتر در رابطه با دایرکتوری های موجود در این پوشه می پردازیم.
* features : بعد از عمل پیش پردازش، برای تولید مقادیر مناسب ویژگی ها در جهت استفاده از آن‌ها در مدل یادگیر مد نظر(به طور مثال تبدیل مقادیر Categorical به Numeric). از این دایرکتوری استفاده خواهیم کرد که به صورت پیش‌فرض یک فایل برای این امر به نام build\_features.py در آن قرار دارد.
* models : کدهای مرتبط با شروع آموزش مدل، پیش‌بینی و ارزیابی مدل را می‌توانیم در این قسمت پیاده‌سازی کنیم (هرچند ما در پروژه خود از فایل‌های ژوپیتر برای آسانی کار استفاده کردیم). به صورت پیش‌فرض فایل‌هایی مثل predict\_model.py و train\_model.py در آن قرار دارند.
* visualization : خروجی کار بر روی دیتاست خود را می‌توانید در این پوشه نمایش دهید و دیتاست خود را به صورت بصری مشاهده کنید تا از این طریق ارزیابی هایی را روی دیتاست خود در جهت بهبود آن بدست آورید. به صورت پیش‌فرض یک فایل برای این امر به نام visualize.py در آن قرار دارد.

### کمی به ساختار عمیق‌تر بنگریم…

حال که یک توضیح اجمالی در رابطه با دایرکتوری ها دادیم میخواهیم به ساختار دایرکتوری source\_codes/data بیشتر نگاه بندازیم.



وقتی به ساختار پوشه دیتا درون مسیر mostafa\_vahdani\_bachelor\_project/source\_codes/data نگاه کنیم همچون ساختاری را مثل تصویر بالا مشاهده خواهیم کرد. که این ساختار عبارت اند از:

1. **common\_utils** : معمولاً ازutils.py زمانی استفاده می‌شود که ما در پروژه خود ابزارهایی داریم که در آینده حتما به آن‌ها نیاز پیدا خواهیم کرد پس این ابزارها که اعم از function و یا class های متعددی می‌باشند را در این فایل قرار خواهیم داد. اما ما در اینجا کار دیگری نیز انجام داده‌ایم ما در پروژه خود دو زیر پروژه غیر وابسته و مستقل به نام های flight\_ticket\_preprocessing و flight\_tickets\_scraper داشته‌ایم با اینکه این دو پروژه به هم وابسته نیستند ولی یکسری از اعمال در این زیر پروژه یکسان هستند و استفاده زیادی دارند پس با تعریف یک package سراسری به اسم common\_utils این ابزارها را به شکل سراسری در هر دو زیر پروژه می‌توانیم استفاده کنیم. در مبحث توضیح پروژه نمی‌گنجد که راجع به چگونگی ایجاد پکیج و استفاده از آن توضیح دهیم پس از این مرحله گذر می‌کنیم و توضیح ساختار درون این دایرکتوری نمی‌پردازیم ولی در جلوتر حتماً راجبه ابزارهایی که استفاده کردیم توضیحاتی را به شما ارائه خواهیم داد.
2. **flight\_tickets\_preprocessing** : این دایرکتوری همانطور که از اسمش کاملاً مشخص است و جای فکر برای شما باقی نمی‌گذارد از مجموعه ایی از ماژول های پایتونی تشکیل شده است که وظیفه اعمال پیش پردازش و تمیز کردن داده‌های خام جمع آوری شده را بر عهده دارند. بعداً راجبه این ماژول ها توضیحاتی ارائه خواهیم داد.
3. **flight\_tickets\_scraper** : از اسم این دایرکتوری هم کاملاً مشخص است که به عنوان یک کراولر/ اسپایدر وظیفه جمع آوری داده خام را از سایت هدف بر عهده دارد. این دایرکتوری که دایرکتوری ریشه زیر پروژه Flight Tickets Scraping می‌باشد دارای یک نام به اسم flight\_tickets\_scraper و همچنین یک زیر دایرکتوری به همین اسم می‌باشد که تمامی کدهای **مربوط** به کراولر یا اسپایدر خود را در این بخش می نویسیم. بعداً در بخش ماژول ها بیشتر راجبه عملکرد این کراولر توضیح خواهیم داد. همچنین این دایرکتوری در دل خودش یک دایرکتوری به اسم spiders نیز دارد که در آن تمامی اسپایدر های مورد نیاز برای کراول کردن سایت هدف را تعریف خواهیم کرد.
4. **static** : در این پوشه تمامی فایل‌های استاتیک یا همان ایستا که در طول پروژه از آن‌ها استفاده خواهیم کرد چه برای کراول و چه برای پیش پردازش را قرار می‌دهیم که شامل فایل‌هایی مثل:

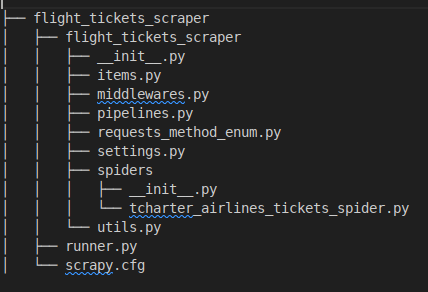
* airlines\_code.json : این فایل شامل کدهای استاندارد شناسایی IATA و یا ICAO مرتبط با هر ایرلاین یا همان شرکت هواپیمایی می باشد. (محتوای این فایل به صورت دستی فراهم شده است)
* airport\_city\_codes.json : این فایل شامل کدهای استاندارد شناسایی هر Airport یا همان فرودگاه می باشد. ( محتوای این فایل توسط سایت هدف فراهم شده ولی به صورت دستی آن را ویرایش کرده‌ایم تا راحت‌تر قابل خواندن باشد.)
* airports\_geometry.json : این فایل شامل مختصات جغرافیایی هر فرودگاه به صورت lat یا همان latitude (عرض جغرافیایی- x) و lng یا همان longitude (طول جغرافیایی - y) در پارامتر geometry می باشد. ( این فایل توسط یک اسکریپت پایتونی فراهم شده است)
* airports\_info.json : این فایل شامل اطلاعات هر فرودگاه مثل نام رسمی و محلی فردوگاه به فارسی و انگلیسی، کد شناسایی فرودگاه، اسم شهر فرودگاه، ناحیه یا منطقه ایی که فردوگاه در آن قرار دارد و کد IATA دو حرفی کشور آن فرودگاه. (این فایل به صورت دستی فراهم شده است)

نکته: توجه شود که این فایل‌ها متناسب با فرودگاه های موجود در سایت هدف فراهم شده‌اند و ما اطلاعات کل فرودگاه های جهان یا ایران را فراهم نکرده ایم.

* months.json : دوازده ماه شمسی با شماره متناسب با آن‌ها در این فایل قرار دارد.

نکته: این فایل‌ها در طول نوشتن این پروژه و به مرور زمان ایجاد شده‌اند و از اولین زمان ایجاد پروژه وجود نداشتند.

## ماژول ها

حال که ساختار دایرکتوری های پروژه را مورد بررسی قرار دادیم در ادامه به توضیح ماژول های موجود در هر دایرکتوری می پردازیم:

در ابتدا به توضیح ماژول های موجود در زیر پروژه **flight\_tickets\_scraper** می پردازیم. همانطور که قبلاً هم بیان کردیم این زیر پروژه وظیفه کروال کردن سایت هدف را برعهده دارد.

همانطور که از عکس بالا مشخص است ماژول های این زیر پروژه عبارت اند از:

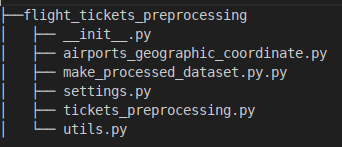
1. **\_\_init\_\_**: این ماژول در هر جایی باشد موجب می‌شود که دایرکتوری مد نظر به صورت یک پکیج برای زبان برنامه نویسی پایتون دیده شود.
2. **items.py** : قبل از اینکه داده‌های جمع آوری شده را ذخیره یا پردازش کنیم و مورد تغییر قرار دهیم مثل یک Container موقتی برای ذخیره داده های استخراج شده و تولید شده توسط اسپایدر عمل می کند.
3. **middlewares.py** : این ماژول وظیفه تغییر در Request و Response ها را بر عهده دارد. در‌واقع قبل از اینکه Request شما به درون اینترنت و سایت هدف ارسال شود و یا قبل از اینکه Response به اسپایدر شما برسد توسط این ماژول می‌توانیم آن‌ها را دستکاری کنیم.
4. **piplines.py** : اگر می‌خواهیم تغییر اضافی بر روی داده استخراج شده که در قالب یک item می‌باشد انجام دهیم از این ماژول استفاده می‌کنیم و به طور کلی از این ماژول برای اعمالی مثل:

* Data Validation
* Cleaning
* Duplicate Checking
* Saving to database

استفاده می شود.

1. **settings.py** : این ماژول همانطور که از اسمش هم پیداست کانفیگ ها و تمام تنظیمات کراولر ما در آن قرار دارد.
2. **requests\_method\_enum.py** : از این ماژول برای تعریف انواع Request ها به صورت مقادیر شمارشی استفاده شده است.
3. **tcharter\_airlines\_tickets\_spider.py** : اسپایدر مربوطه ما برای استخراج داده‌ها از سایت هدف در این ماژول نوشته شده است
4. **utils.py** : از این ماژول برای تعریف توابع کمکی و مشترک بین اسپایدرها در طول پروژه استفاده می کنیم.
5. **runner.py** : این ماژول در زمان استفاده از قابلیت دیباگ کردن پروژه Scraping درIDE مربوطه کاربرد دارد.
6. **scrapy.cfg** : تنظیمات فریمورک Scrapy برای اجرا کردن پروژه در این فایل قرار دارند.

حال به سراغ توضیح ماژول های موجود در زیر پروژه **flight\_tickets\_preprocessing** می رویم. این زیر پروژه وظیفه پیش پردازش کردن و تمیز کردن داده‌های خام ما را بعد از مرحله Scraping بر عهده دارد.



همانطور که از تصویر بالا مشخص است ماژول های زیر پروژه مربوطه عبارت اند از:

1. **settings.py** : کانفینگ ها و تنظیمات سراسری مختص به این زیر پروژه در این فایل قرار دارند.
2. **tickets\_preprocessing.py** : ما در این ماژول توابع پیش پردازش خود را می‌نویسم تا از طریق آن‌ها داده‌های خام را تمیز کنیم. این فایل و محتوای آن فقط مختص به این پروژه می‌باشد زیرا محتوای درون آن متناسب با داده‌های جمع آوری شده برای تمیز و مرتب کردن آن نوشته است.
3. **utils.py** : از این ماژول برای تعریف توابع کمکی، مشترک و پرکاربرد استفاده کرده ایم.
4. **make\_processed\_dataset.py** :بعد از این توابع مورد نظر خود را در ماژول ticket\_preprocessing.py نوشتیم در جهت اعمال بر روی دیتاست در اجرا آن‌ها را وارد این ماژول می کنیم. در‌واقع به مثل تابع main در cpp می‌ماند با این تفاوت که به جای تابع از ماژول استفاده کرده ایم. در‌واقع ما با اجرای این فایل تغییرات را روی دیتاست خود اعمال می کنیم.
5. **airports\_geographic\_coordinate.py** : از این فایل برای نوشتن یک اسکریپت پایتونی در جهت پیدا کردن مختصات جغرافیایی هر فرودگاه به کمک یک API و ذخیره آن‌ها در فایل airports\_geometry.json استفاده کرده ایم.

## مابقی فایل‌ها ...

فایل‌هایی مخفی ای مثل gitignore ، gitkeep و غیره در پروژه ما وجود دارند و اما این فایل‌ها چه نقشی در پروژه ما دارند؟

1. **gitignore** : در زمانی که نمیخواهیم یک سری فایل‌ها و دایرکتوری های ما وارد ریپوزیتوری گیت هابمان شوند از این فایل استفاده می‌کنیم و با مراجعه به محتوای این فایل به راحتی می‌توانید متوجه شوید که چه فایل‌ها و دایرکتوری نیاز نیست که وارد مخزن یا ریپو ما شوند.
2. **gitkeep** : زمانی که در پروژه ما یک دایرکتوری وجود دارد ولی این دایرکتوری در حال حاضر خالی است اما بعداً از فایل و ماژول هایی پر می‌شود از این فایل استفاده می‌کنیم چرا که ابزار گیت به صورت پیش‌فرض نمی‌تواند دایرکتوری های خالی را وارد گیت هاب ما کند یک روش این می‌باشد که ما میتوانیم هر نوع فایلی حتی فایل gitignore را درون آن دایرکتوری خالی قرار دهیم تا بتوانیم آن دایرکتوری را به داخل گیت هاب خود Push کنیم اما روش بهتری که همه از آن استفاده می‌کنند این است که به درون آن دایرکتوری فایل gitkeep را اضافه می کنند.
3. **env** : این فایل مخفی به صورت پیش فرض در پروژه ایی که دست شماست وجود ندارد و شما باید خودتان آنرا در کنار دایرکتوری ریشه mostafa\_vahdani\_bachelor\_project ایجاد کنید چرا که اطلاعات حیاتی مثل پسورد و لاغیر در آن ذخیره می‌شود که نباید در محیط پابلیک مثل گیت هاب قرار گیرند.

## چگونه از پروژه اجرا بگیریم؟

نکته: در اینجا مراحل اجرا گرفتن را با جزئیات توضیح نمی‌دهیم چرا که اگر به سطحی رسیدید که دارید این داکیومنت را مشاهده و مطالعه می‌کنید پس جزئیات و چرایی مراحل را خودتان می دانید.

1. یک محیط مجازی برای نصب کتابخانه‌ها در جهت اجرا گرفتن کد های پروژه با هر ابزاری که می‌شناسید ایجاد کنید می‌توانید از virtualenv و یا virtualenvwrapper استفاده کنید.
2. از طریق Command Line خود به درون دایرکتوری development پروژه بروید و در آنجا قطعه کد زیر را اجرا کنید.

**pip install -r requirements.txt**

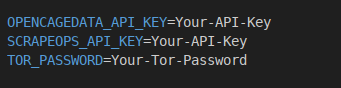
از قبل بررسی کنید اینترنت و همچنین VPN شما وصل باشد.

1. به درون دایرکتوری common\_utils در مسیر source\_codes/data پروژه بروید و در آنجا قطعه کد زیر را اجرا کنید.

**pip install .**

اون نقطه رو اشتباهی نذاشتم اونجا اون نقطه هم باید باشه وقتی کد رو کپی می کنی :)

1. سرویس Tor را روی کامپیوتر خود برروی پورت ۹۰۵۰ و TorControl را برروی 9051 تنظیم کنید.
2. سرویس Privoxy را برروی سیستم خود فعال کنید تا Scrapy از طریق آن بتواند به سرویس Tor متصل شود.
3. مرحله آخر هم این هست که باید در کنار دایرکتوری ریشه پروژه فایل مخفی env رو ایجاد کنید و پارامترهای زیر رو طبق عکس بهش بدید.



**نکته** : توجه شود که برای گرفتن کلید API باید در سایت‌های OpenCageData و ScrapeOps ثبت نام کنید و کلید خودتون رو در پارامترهای بالا قرار دهید.

**نکته** : پارامترها دقیقاً باید به همان اسمی باشند که در تصویر می‌بیند پس پارامترهارا به همین شکل نامگذاری

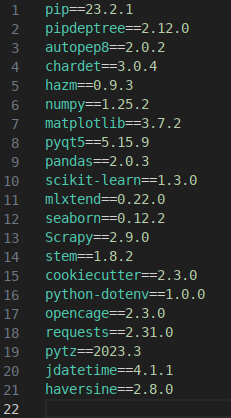
کنید و به هیچ وجه آن‌ها تغییر ندهید.

# فصل **۲**

## **تکنولوژی ها و ابزارهای به کار رفته**

تنها زبان برنامه نویسی و تکنولوژی به کار رفته در این پروژه **زبان برنامه نویسی قدرتمند و بسیار ساده پایتون** می‌باشد از این زبان برای **تمامی فازهای پروژه** استفاده شده است. زبان برنامه نویسی پایتون از ماژول‌ها (modules) و بسته‌ها (packages) استفاده می‌کند، بدین معنا که برنامه‌های این زبان قابل طراحی به سبک ماژولار (modular) هستند و کدهای نوشته شده در یک پروژه در پروژه‌های گوناگون دیگر نیز قابل استفاده مجدد محسوب می‌شوند. هنگامی که کاربری ماژول یا بسته مورد نیاز خود را توسعه داد، خودش یا دیگر علاقمندان (در صورتی که کد در اختیار عموم قرار بگیرد) می‌توانند آن را برای استفاده در دیگر پروژه‌ها گسترش دهند.

در این پروژه نیز از کتابخانه، فریمورک و ابزارهای Built-in و External متفاوتی به صورت مشترک و یا مستقل استفاده شده است اگر به دایرکتوری development مراجعه کنید فایلی به نام requirements.txt را مشاهده خواهید کرد که محتوای آن که در عکس زیر موجود است و نشان دهنده موارد External می‌باشد که در پروژه وجود دارند.



همانطور که در عکس بالا مشاهده می‌کنید از موارد بالا در پروژه استفاده کرده‌ایم در ادامه نیز راجبه استفاده هرکدام توضیحاتی را ارائه می‌دهم و حتماً به این نکته نیز توجه کنید برای اجرا گرفتن پروژه حتماً باید موارد بالا بر روی سیستم شما همانطور که در بخش قبلی گفته شد نصب شده باشند.

### کتابخانه‌های **داخلی** زبان پایتون

1. **کتابخانه os** : ماژول os در پایتون راهی برای تعامل با سیستم عامل ارائه می دهد که به شما امکان می دهد تسک های مختلف مربوط به سیستم را انجام دهید.
2. **کتابخانه sys** : ماژول sys در پایتون دسترسی به پارامترها و توابع مختلف سیستم را فراهم می کند. معمولاً برای تعامل باPython runtime environment و سیستم عامل استفاده می شود.
3. **کتابخانه itertools** : ماژول itertools در پایتون مجموعه ای قدرتمند از توابع است که ابزارهای مختلف سریع و کارآمدی را برای کار باIterator ها ارائه می دهد. بخشی از کتابخانه استاندارد پایتون است و معمولاً برای ایجاد باIterator برای لوپ زدن کارآمد و دستکاری داده‌ها استفاده می‌شود.
4. **کتابخانه pathlib** :ماژول pathlib در پایتون یک رویکرد شی گرا برای کار با مسیرها و دایرکتوری های سیستم فایل ارائه می دهد.
5. **کتابخانه enum** : شمارش (enum) مجموعه‌ای از نام‌های نمادین (اعضا) است که به مقادیر ثابت و منحصربه‌فرد محدود می‌شوند. Enum ها با دادن نام های معنی دار به ثابت ها، راهی برای ایجاد کد قابل خواندن و نگهداری بیشتر ارائه می دهند.
6. **کتابخانه random** : به شما امکان می دهد با اعداد تصادفی کار کنید.
7. **کتابخانه time** : این ماژول توابع مختلف مرتبط با زمان را ارائه می دهد.
8. **کتابخانه urllib** :کتابخانه urllib در پایتون ماژولی برای کار با URL ها است. و چندین ماژول برای کار با URL ها و ایجاد درخواست های HTTP فراهم می کند.
9. **کتابخانه json** : پایتون کتابخانه داخلی را برای کار با فایل‌هایی با فرمت JSON از طریق ماژول json فراهم می کند.

### کتابخانه و فریمورک های مشترک

1. **پکیج autopep8** : یک ابزار کامند لاینی و کتابخانه پایتونی است که به طور خودکار کد پایتون را برای مطابقت با راهنمای سبک PEP-8 پایتون (راهنمای طریقه نوشتن کد تمیز پایتون) قالب بندی می کند. این به حفظ استایل کد یکنواخت کمک می کند و کد شما را خواناتر و قابل نگهداری تر می کند.
2. **پکیج یا کتابخانه pipdeptree** : این ابزار کامند لاینی به نمایش پکیج های پایتونی نصب شده در سیستم شما کمک می کند. هرچند که این ابزار در روند اصلی پروژه و ایجاد خروجی تأثیری ندارد ولی صرفاً به دلیل استفاده از آن تصمیم گرفتم آنرا در لیست قرار دهیم.
3. **پکیج pip** : این پکیج برای نصب پکیج های نرم‌افزار مثل کتابخانهها و فریمورک های زبان برنامه نویسی پایتون می باشد.
4. **کتابخانه python-dotenv** : این کتابخانه از یک فایل مخفی به نام env که توسط شما هم باید ایجاد شود جفت‌های کلید-مقدار را می‌خواند و آن‌ها را در Environment Variables سیستم شما قرار می‌دهد تا بتوانید از این عمل برای افزایش امنیت و تمیزی کدهای کد استفاده کنید.
5. **کتابخانه common\_utils** : این یک پکیج خیلی ساده است که ابزارهای درون آن توسط بنده برروی پکیج json **جمع آوری** شده است و **در حال حاضر** همانطور که از اسم پکیج مشخص است صرفاً در آن یکسری تابع **پرکابرد** در انواع پروژه ها برای کار با فایل و آبجکت های JSON قرار دارد.

### **کتابخانه‌ها و فریمورک های زیر پروژه flight\_tickets\_scraper**

* **فریمورک scrapy** : مواردی مثل Web Scraping و Web Crawling دو مورد از مهم ترین موارد بررسی و تحلیل صفحات وب هستند. در سال 2008 فریمورکی تحت عنوان Scrapy انتشار یافت که برای تازه کاران ساده و برای حرفه ای ها به اندازه کافی دارای ویژگی های مثبت است. وبسایت ها حاوی اطلاعات معنی دار هستند که می توانیم آن ها را استخراج کنیم و در کاربرد مورد نظرتان ( طراحی موتور جستجو یا ایجاد یک دیتاست برای کار تحلیل داده یا …) از این داده ها بهره ببرید. از Scrape کردن می‌توان برای جمع آوری محتویات خام از صفحات وب و ذخیره کردن آنها برای استفاده بعدی در یک قالب ساختارمند و معنادار استفاده کرد. بهتر از یک نگاه به معماری که این فریمورک از الان پیروی می‌کند هم نگاهی بی اندازیم که چرا که در بخش توضیح کدها بسیار با این فریمورک درگیر خواهیم شد. معماری این فریمورک قدرتمند به شرح عکس زیر می باشد:



* عکسی که مشاهده می‌کنید نشان می‌دهد که Engine مهم‌ترین بخش این معماری می‌باشد چرا که به گفته داکیومنت وظیفه کنترل جریان دیتاهای عبوری کل سیستم(جا به جا کردن Request و Response بین تمام کامپوننت ها و تحویل دادن Item ها) و همچنین تحریک کردن Event ها را بر عهده دارد. رکیوئست شما در ماژول اسپایدر مد نظرتان تولید می‌شود و سپس این رکیوئست ها توسط Engine در Scheduler زمابندی شده تا Engine بتواند دوباره از اسپایدر Request بعدی را تحویل بگیرد در ادامه Request زمان بندی شده دوباره به Engine داده می‌شود تا آن را به Downloader منتقلش کند. وظیفه Downloader این می‌باشد که Request شما را به درون اینترنت ارسال کند، صفحات وب دریافتی را fetch کند و در نهایت Response را در اختیار Engine قرار دهد. Engine نیز Response دریافتی را توسط Middleware ها جهت parse شدن به اسپایدر بر می گرداند و در پایان اسپایدر نیز داده‌های استخراج شده را با ساختن یک Item و yield کردن، توسط Engine به Pipeline ها منتقل می‌کند به طور مختصر Pipeline ها وظیفه پردازش Item ها را بر عهده دارند.
* **کتابخانه stem** : این کتابخانه دارای ابزاری برای تعامل با شبکه تور می باشد. با استفاده از این کتابخانه و ابزارهایی که برای شما فراهم آورده است می‌توانید اسکریپت هایی را بنویسید که instance های شبکه تور را کنترل و اطلاعات لازمه را از relay و exit node هاو مدارهای ایجاد شده دریافت کند.
* **پکیج chardet** : کتابخانه chardet یک کتابخانه پایتونی است که برای تشخیص Encoding کاراکتر استفاده می شود. با تجزیه و تحلیل بایت ها و الگوهای آماری موجود، به تعیین Encoding کاراکتر یک متن یا فایل معین کمک می کند. این امر به ویژه هنگام کار با داده های متنی که ممکن است از منابع مختلف آمده و دارای رمزگذاری های متفاوت باشند، مفید است.
* **کتابخانه requests** : یک پکیج محبوب برای ایجاد درخواست های HTTP می‌باشد که فرآیند ارسال درخواست های HTTP و رسیدگی به پاسخ ها را ساده تر می کند. به طور گسترده برای کارهایی مانند واکشی داده ها از API ها، دانلود صفحات وب و تعامل با سرویس های وب استفاده می شود.

### **کتابخانه‌ها و فریمورک های زیر پروژه flight\_tickets\_preprocessing**

* **کتابخانه hazm** : کتابخانه hazm یک پکیج پایتون است که برای پردازش متن فارسی طراحی شده است. عملکردهای مختلفی را برای نرمال سازی متن، ریشه یابی، واژه سازی، نشانه گذاری، برچسب گذاری بخشی از گفتار و موارد دیگر فراهم می کند. این یک ابزار ارزشمند برای پردازش زبان طبیعی وظایف مربوط به زبان فارسی است.
* **کتابخانه matplotlib** : یک کتابخانه پایتونی است که به طور گسترده برای ایجاد شکل و نمودار های بصری ایستا، تعاملی و متحرک در پایتون استفاده می شود. این کتابخانه برای مصورسازی و تحلیل داده‌ها بسیار کارآمد می باشد.
* **کتابخانه pandas** : یک کتابخانه OpenSource پایتونی است که برای دستکاری و تجزیه و تحلیل داده ها استفاده می شود. ساختارهای داده ایی و توابع مورد نیاز برای دستکاری کارآمد داده های ساخت یافته را فراهم می کند. pandas به طور گسترده در علم داده، یادگیری ماشین و سایر زمینه های مرتبط با داده استفاده می شود.
* **کتابخانه pyqt5** : یک انقیاد پایتونی برای فریمورک Qt است که به توسعه دهندگان اجازه می دهد تا برنامه های کراس پلتفرمی را با رابط کاربری گرافیکی ایجاد کنند. به طور گسترده ای برای ساخت برنامه های دسکتاپ که بر روی سیستم عامل های مختلف از جمله ویندوز، مک و لینوکس اجرا می شوند، استفاده می شود. در این پروژه صرفاً ما این پکیج را نصب می‌کنیم تا بتوانیم نمودارهای خود را به نمایش در آوریم.
* **کتابخانه scikit-learn** : یک کتابخانه یادگیری ماشین OpenSource بسیار پرکاربرد و قدرتمند پایتونی است. ابزارهای ساده و کارآمدی برای داده کاوی و تجزیه و تحلیل داده ها فراهم می کند. در یک ساختار انتزاعی با تکیه بر دیگر ابزارها بر روی کتابخانه های محبوب دیگر مانند NumPy، SciPy و Matplotlib ساخته شده است. Scikit-Learn برای کار با وظایف مختلف یادگیری ماشین، از جمله طبقه بندی، رگرسیون، خوشه بندی، کاهش ابعاد و غیره طراحی شده است و در این پروژه نیز بسیار به ما کمک خواهد کرد.
* **ابزار cookiecutter** : اگر مثل بنده همیشه به دنبال این باشید که پروژه شما باید یک ساختار منظم و مرتب داشته باشد تا هم خودتان هم بقیه که بعدها کد شما را می‌خوانند بدانند دقیقاً کجا دنبال چه چیزی بگردند متوجه می‌شوید در این موضوع پروژه های دیتاساینس یک ضعف بزرگ دارند و همچنین یک سردرد بزرگ برای شما هستند چرا که اصلاً و به هیچ وجه یک ساختار منظم و واحد بر خلاف پروژه های دیگر مثل پروژه های طراحی بک اند با django یا طراحی یک کراولر با scrapy ندارند و با شروع هر پروژه دیتاساینی جدید شما ساختار قبلی را فراموش می‌کنید و یک ساختار جدید را برای کد های پروژه خود ایجاد می کنید. اما با مقداری سرچ متوجه خواهید شد که پروژه هایی مثل cookiecutter وجود دارند که به راحتی به شما امکان ایجاد یک Template برای پروژه دیتاساینسی یا هر پروژه دیگری را به شما خواهند داد و این مورد برای ما فوق‌العاده خواهد بود.

Cookiecutter یک ابزار کامند لاینی است که به شما امکان می دهد پروژه ها را در Template های متفاوت ایجاد کنید که در آن شما با یک الگو شروع می کنید، و این ابزار به شما کمک می کند یک ساختار پروژه را بر اساس آن الگو ایجاد کنید. این به ویژه زمانی مفید است که می‌خواهید یک پروژه جدید راه‌اندازی کنید که از یک ساختار دایرکتوری خاص پیروی می‌کند، شامل فایل‌های خاصی است و دارای تنظیمات از پیش تعریف‌شده می باشد.

گفتنی است که ما دقیقاً نمیخواهیم دوباره شروع کنیم چرخ را اختراع کرده و ساختار پروژه های دیتاساینسی را تعریف کنیم چرا که قبلاً یکی اینکار رو برای ما انجام داده ( این است زیبایی دنیای نرم افزار) اگر مقدار بیشتر سرچ کنیم متوجه می‌شویم پروژه ایی به نام CookieCutter Data Science وجود دارد که می‌توانیم از آن بهره ببریم.

* **ابزار Cookie Cutter Data Science** : یک الگوی پروژه است که به شما کمک می کند پروژه های علم داده خود را به شیوه ای سازگار و سازمان یافته ساختار دهید. این یک ابزار کامند لاینی است که به شما امکان می دهد با پیروی از یک ساختار دایرکتوری از پیش تعریف شده و پر کردن آن با فایل های ضروری، الگوهای پروژه خود را ایجاد کنید. این می تواند در زمان شروع پروژه های جدید علم داده در زمان و تلاش شما صرفه جویی کند، زیرا مجبور نخواهید بود که ساختار و فایل های مشابه را به طور مکرر تنظیم کنید.
* **کتابخانه opencage** : ما در ادامه درون پروژه خود از API سایت OpenCage برای جمع آوری مختصات جغرافیایی هر فرودگاه استفاده خواهیم کرد این سایت برای راحتی استفاده برنامه نویس ها از API خودش یک کتابخانه ارائه کرده که می‌توانیم از آن بهره ببریم.
* **کتابخانه mlxtend** : یک کتابخانه پایتونی است که توسط سباستین راشکا توسعه یافته است که بر ارائه مجموعه‌ای از ابزارها و برنامه‌های افزودنی برای ارتقای قابلیت‌های Sikit-Learn و دیگر کتابخانه‌های یادگیری ماشین تمرکز دارد. این شامل پیاده‌سازی الگوریتم‌های مختلف یادگیری ماشین، معیارهای ارزیابی، و توابع کاربردی است که می‌تواند هم برای مبتدیان و هم برای متخصصان با تجربه مفید باشد.
* **کتابخانه seaborn** : یک کتابخانه است که بر روی matplotlib ساخته شده است. برای مصورسازی داده ها و تجزیه و تحلیل داده های اکتشافی استفاده می شود. Seaborn به راحتی با دیتافریم ها و کتابخانه pandas کار می کند. نمودارهای ایجاد شده را نیز می توان به راحتی شخصی سازی کرد.
* **کتابخانه pytz** : قابلیت کار با مناطق زمانی مختلف را فراهم می کند. به شما امکان می دهد تاریخ را محلی سازی کنید، مناطق زمانی مختلف را به هم تبدیل کنید. این کتابخانه به ویژه در هنگام برخورد با محاسبات مربوط به نمایش تاریخ و زمان در مناطق زمانی مختلف مفید است.
* **کتابخانه haversine** : فرمول haversine برای محاسبه فاصله بین دو نقطه روی سطح یک کره مانند زمین استفاده می شود. این فرمول اغلب در جاهای استفاده می شود که شامل موقعیت جغرافیایی و محاسبه فاصله بین مختصات می شود. پکیج haversine روشی ساده برای محاسبه فاصله بین دو مجموعه مختصات با استفاده از فرمول haversine ارائه می دهد. این می تواند برای برنامه هایی که شامل محاسبه فاصله بین نقاط جغرافیایی هستند مفید باشد.
* **کتابخانه numpy** : یک کتابخانه قدرتمند در پایتون برای عملیات عددی و ریاضی است. پشتیبانی از آرایه ها و ماتریس ها، همراه با طیف وسیعی از توابع ریاضی برای کار بر روی این آرایه ها را فراهم می کند. Numpy یک پکیج اساسی برای محاسبات علمی با پایتون است.
* **کتابخانه jdatetime** : کتابخانه jdatetime به شما امکان می دهد با تاریخ و زمان جلالی (فارسی) در پایتون کار کنید. قابلیت هایی برای تبدیل، فرمت کردن و دستکاری تاریخ های جلالی فراهم می کند.

علاوه بر موارد گفته شده ابزارهای دیگری نیز به کمک ما برای توسعه کدهای پروژه آمده اند. البته باید گفت که این ابزار در کد نویسی پروژه تأثیر **غیر مستقیم** داشته اند.

### ابزارهای مرتبط استفاده شده در طول پروژه

1. **Git** : یک سیستم **کنترل نسخه** توزیع شده است که به توسعه دهندگان کمک می کند تغییرات را در کد های خود پیگیری و دنبال کرده و با دیگران همکاری کنند تا تاریخچه پروژه را به طور موثر مدیریت کنند. این نرم افزار توسط لینوس توروالدز در سال 2005 ایجاد شد و از آن زمان به یک ابزار اساسی در توسعه نرم افزار تبدیل شده است.
2. **Tor Service** : با فعال کردن این سرویس در سیستم خود شما به یک شبکه ایی از اتصال **نظیر به نظیر** گره‌های متصل بهم دسترسی دارید. مسیریابی ترافیک اینترنتی شما از طریق یک شبکه ای از سرورهای داوطلب به نام گره ها انجام می شود. هر گره در شبکه Tor لایه ای از رمزگذاری را به داده های شما اضافه می کند و ردیابی به مکان یا هویت اصلی شما را دشوار می کند. استفاده ما از تور در پروژه به عنوان یک پروکسی رایگانی است که در هر چند ثانیه IP ما را در اینترنت عوض می کند. هر چند اگر می‌خواهید از تور در پروژه های Scraping دیگر خود استفاده کنید توجه کنید که باید سرور مقصد یا همان سایت هدف باید از exit node های شبکه تور پشتیبانی کند.
3. **Privoxy** : یک Web Proxy با قابلیت های فیلترینگ پیشرفته برای افزایش حریم خصوصی، دستکاری کوکی ها، کنترل دسترسی و موارد دیگر است. اغلب در ترکیب با ابزارهای دیگر مانند Tor برای ارائه یک لایه اضافی از ناشناس بودن و امنیت استفاده می شود. Privoxy به عنوان یک **واسطه** بین کامپیوتر شما، نرم‌افزار مربوطه با شبکه تور و اینترنت عمل می کند و محتوا و درخواست ها را با توجه به کانفیگ های آن فیلتر می کند. ما در پروژه خود برای اتصال Scrapy به سرویس Tor و سپس انتساب IP که Relay های Tor در اختیارمان قرار می‌دهند به Request ها به سایت هدف بهره برده ایم. به همین دلیل به Privoxy نام Intermediate Proxy نیز داده اند.
4. **Postman** : نرم افزاری با رابط گرافیکی است که به ما امکان می‌دهد تست API را انجام دهیم. مانند مرورگری است که html را رندر نمی‌کند. در مرورگر می‌توانیم فقط GET HTTP Request ارسال کنیم، اما در اینجا می‌توانیم GET، POST، PUT، PATCH، DELETE و بسیاری دیگر از درخواست‌های HTTP را ارسال کنیم. در postman ما می توانیم درخواست را ذخیره کرده و مجموعه‌ای از پوشه ها را برای سازماندهی Request ها ایجاد کنیم. کاربرد ما از این نرم‌افزار در این پروژه در بخش‌هایی مثل بررسی Response های درخواست های Ajax سایت هدف، استفاده از API ها و همچنین کمک به نوشتن اسکریپت پایتونی برای API ها و یک کراولر چند سطحی بوده است.
5. **Jupyter** : نوت بوک های Jupyter به طور گسترده برای تجزیه و تحلیل داده ها، یادگیری ماشینی، تحقیقات علمی و آموزش استفاده می شود. این نوت بوک از زبان برنامه نویسی Python و R پشتیبانی می‌کند و یکی از قابلیت‌هایی این نوت بوک که از آن در پروژه استفاده کردیم قابلیت Interactive Code Execution می‌باشد که به ما این اجازه را می‌دهد هر خط کد را به صورت جداگانه اجرا کنیم و نتیجه‌ آن را به صورت inline مشاهده کنیم.

### API های استفاده شده در پروژه

1. **OpenCageAPI** : یک سرویس geocoding است که روشی آسان برای تبدیل مختصات جغرافیایی (طول و عرض جغرافیایی) به اطلاعات مکان قابل خواندن توسط انسان و بالعکس ارائه می دهد. معمولاً در برنامه هایی که به خدمات موقعیت جغرافیایی نیاز دارند استفاده می شود.
2. **ScrapOpsAPI** : سرویسی است که به شما این امکان را می دهد تا پارامتر های موجود در Web Scraping یک سایت را کنترل کنید. این سرویس برای ساده سازی و خودکارسازی اجزای عملیات Web Scraping طراحی شده است و جمع آوری داده ها از وب سایت ها را برای توسعه دهندگان برای اهداف مختلف آسان تر می کند. یکی از امکاناتی که این API در اختیارمان قرار می‌دهد این می‌باشد که برای ما FakeUserAgent تولید می‌کند تا سایت هدف Request ما را مثل رکیوئستی در نظر بگیرد که از یک مرورگر واقعی در حال ارسال شدن است به عبارتی به کمک این API می‌توانیم خودمان را جای یک کاربر واقعی جا بزنیم تا سایت هدف به ربات بودن ما شک نکند.

# فصل **۳**

## **عملکرد و توضیح کد ها**

تا به اینجای کار به معرفی پروژه و اهداف پرداختیم و در رابطه با ساختار دایرکتوری ها و ماژول های پروژه، ساختار دیتاست، تکنولوژی و ابزارهای به کار رفته در پروژه و همچنین طریقه اجرا کردن کدها توضیحاتی را به شما ارائه دادیم. حال میخواهیم کدهای نوشته شده را به صورت فنی و تکنیکی مورد بررسی قرار دهیم تا متوجه شویم پروژه نوشته شده در جزئیات به چه شکل کار می‌کند بدین طریق شما خواننده گرامی بتوانید تغییرات دلخواه را در جهت بهبود پروژه به آن اضافه کنید. ترتیب توضیح دادن کدها به ترتیب فازهای پروژه است که در ابتدای بخش معرفی پروژه به شما توضیح دادیم.

**نکته** : ممکن است در عکس‌های تهیه شده کدهایی را مشاهده کنید کامنت شده‌اند، بنده در مورد آن کدها توضیحاتی ارائه نمی‌دهم زیرا که واضح است آن کدهای کامنت شده تأثیری در روند برنامه ندارند. و ممکن است کدهای تولید شده توسط خود فریمورک باشند که ما در حال حاضر از آن‌ها در پروژه استفاده نکرده ایم.

### فاز اول پروژه

کراول کردن سایت هدف در جهت جمع آوری یک دیتاست خام که در این پروژه از سایت tcharter.ir که یک موتور جستجو بلیت های هواپیما می‌باشد استفاده کرده ام.

**ماژول settings.py :**

* **load\_dotenv()**

این خط کد تمام کانفیگ های موجود در فایل env را در Python Environments Variable برای ما لود می‌کند و اجازه دسترسی به متغیرهای محیطی تعریف شده را به ما می دهد.

* **BASE\_PATH = pathlib.Path(\_\_file\_\_).parent.parent.parent**

ذخیره کردن آدرس دایرکتوری data درون دایرکتوری source\_codes از این آدرس به عنوان آدرس پایه برای دسترسی به دایرکتوری های زیر مجموعه دایرکتوری data مثل دایرکتوری static استفاده می شود. در‌واقع با قطعه کد pathlib.Path(\_\_file\_\_) به آدرس مسیر فعلی فایل setting.py دسترسی پیدا می‌کنیم و با هر .parent یک دایرکتوری به عقب بر‌می‌گردیم تا به دایرکتوری data برسیم.

* **BOT\_NAME = "flight\_tickets\_scraper"**

فریمورک اسکرپی موقع اجرا ربات کراولر ما را با ذخیره شده در متغیر BOT\_NAME می‌شناسد که با اسم زیر پروژه بخش Scraping یکی می باشد.

* **SPIDER\_MODULES = ["flight\_tickets\_scraper.spiders"]**

اسکرپی برای پیدا کردن اسپایدر های نوشته شده توسط شما در این لیست به دنبالش می گردد.

* **NEWSPIDER\_MODULE = "flight\_tickets\_scraper.spiders"**

ماژول پیش فرضی که اسپایدرهای جدید شما در آن تعریف می‌شود در این متغیر قرار می گیرند.

* **ROBOTSTXT\_OBEY = False**

اگر می‌خواهید از robot.txt سایت هدف پیروی کنید مقدار این متغیر باید به True تغییر پیدا کند (مقدار پیش‌فرض این متغیر True می باشد) این فایل مشخص می‌کند ربات های خزنده به چه مسیرهایی در سایت می‌توانند با چه محدودیتی دسترسی داشته باشند. در این پروژه بنده مقدار آنرا برابر با False قرار دادم زیرا که در حین درخواست به سایت هدف، سایت ما را با کد ۳۰۲ به URL اصلی tcharter.ir هدایت می‌کرد با تغییر دادن این متغیر به False دیگر این اتفاق نمی افتد.

* **DOWNLOAD\_DELAY = 3.75**

مقدار تأخیر در ارسال هر Request را با این متغیر مشخص می‌کنیم که ما مقدار آنرا با آزمون خطا های متعدد ترجیه دادیم روی ۳.۷۵ ثانیه تنظیم کنیم.

* **RANDOMIZE\_DOWNLOAD\_DELAY = True**

**MinSec = 0.5 \* DOWNLOAD\_DELAY**

**MaxSec = 1.5 \* DOWNLOAD\_DELAY**

**[MinSec, MaxSec]**

با قرار دادن True در این متغیر میزان تأخیر در ارسال درخواست به سایت هدف را تصادفی می‌کنیم که مقدار آن در بازه [MinSec, MaxSec] تصادفی تغییر می کند.

* **DEFAULT\_REQUEST\_HEADERS**

س در بعضی مواقع هدرهایی که خود اسکرپی به صورت پیش برای ما میسازد تا پارامترهای مورد نیاز برای ارسال رکیوئست را پر کند توسط سایت هدف پذیرفته نمی‌شود و ممکن است که وقتی ما را به عنوان یک ربات شناسایی می‌کند Reponse مورد انتظار ما را به ما برنگرداند در نتیجه می‌توانیم از این پارامتر برای ارسال تمام رکیوئست های خود استفاده کنیم.

با توجه به عکس بالا می‌بینید که ما برای پارامتر گفته شده مقادیر بالا را در نظر گرفته‌ایم باید توجه کرد که این مقادیر، مقادیر من درآوردی نیست و توسط Developer Tools مرورگر بخش Network با توجه به رکیوئست مد نظر پیدا شده و صرفاً مقدار پیدا شده را کپی کردیم و در این متغیر قرار دادیم. تا سایت ما از طریق هدر به عنوان ربات شناسایی نکند.

* **DOWNLOADER\_MIDDLEWARES = {**

**'flight\_tickets\_scraper.middlewares.ScrapeOpsFakeUserAgentMiddleware': 400,**

**'flight\_tickets\_scraper.middlewares.TorMiddleware': 610,**

**}**

در این پارامتر Middleware هایی که برای دستکاری رکیوئست یا ریسپانس می نویسیم را قرار می‌دهیم توجه شود عدد هایی که به هر Middleware ها انتساب می‌دهیم در‌واقع ترتیب اجرا شده آن‌ها توسط اسکرپی می باشد.

* **ITEM\_PIPELINES = {**

**"flight\_tickets\_scraper.pipelines.DuplicatesItemsPipeline": 300,**

**"flight\_tickets\_scraper.pipelines.TicketArrivalTimePipeline": 800,**

**}**

در این پارامتر هم pipline هایی که نوشتیم را با انتساپ عددهایشان برای اجرا قرار می دهیم.

* **SOURCES = {**

**"airport\_city\_codes\_path": BASE\_PATH.joinpath("static", "airport\_city\_codes.json"),**

**}**

توسط این متغیر و همچنین به کمک متغیر BASE\_PATH می‌توانیم آدرس سورس هایی که در طول پروژه احتمال استفاده از آن‌ها وجود دارد را مشخص می‌کنیم و با فرمت کلید-مقدار به آن‌ها دسترسی داشته باشیم. این قطعه کد به خوانا تر و منعطف تر شدن کد ما کمک شایانی خواهد کرد زیرا که هم کد مارا داینامیک تر می‌کند و اگر بعدها آدرس سورس تغییر کرد فقط از اینجا آنرا تغییر می‌دهیم و لازم نیس چندین جای کد را دست بزنیم و هم اینکه نیاز نیست هر جایی از پروژه از آدرس مطلق فایل سورس دیتا استفاده کنیم.

* **SCRAPEOPS\_API\_KEY = os.getenv("SCRAPEOPS\_API\_KEY")**

تابع getenv برای دسترسی به متغیر محیطی سیستم استفاده می‌شود که در مورد قطعه کد بالا مشخص است که ما می‌خواهیم به مقدار کلید API ای که قبلاً راجبه آن توضیح دادیم دسترسی داشته باشیم و آنرا در متغیر ذکر شده ذخیره کنیم. ( بعداً از این متغیر در FakeUserAgent Middleware استفاده می کنیم.)

* **SCRAPEOPS\_FAKE\_USER\_AGENT\_ENABLED = True**

برای اینکه بتوانیم از FakeUserAgent Middleware استفاده کنیم علاوه بر اینکه باید این Middleware را درون Download\_Middleware قرار دهیم باید مقدار این متغیر را طبق داکیومنت سایت ScrapeOps به True تغییر دهیم.

* **INTERMEDIATE\_PROXY = 'http://127.0.0.1:8118**

در متغیر آدرس دسترسی به سرویس Privoxy که بر روی سیستم خود فعال کردیم را قرار می‌دهیم تا scrapy بتواند به سرویس Tor دسترسی داشته باشد. ( بعداً از این متغیر در TorMiddleware استفاده می کنیم.)

* **IP\_CHECKER\_SITE = "http://icanhazip.com**

از این متغیر استفاده می‌کنیم تا آدرس سایتی را در آن قرار دهیم که خروجی سایت نمایش آدرس IP فعلی سیستم ما می باشد. ( بعداً از این متغیر در TorMiddleware استفاده می کنیم.)

* **TOR\_PASSWORD = os.getenv("TOR\_PASSWORD")**

پسورد سرویس تور را می‌توانیم از طریق این کد بخوانیم و در متغیر ذکر شد ه قرار دهیم. ( بعداً از این متغیر در TorMiddleware استفاده می کنیم.)

* **TOR\_RANDOMIZE\_CHANGING\_IP\_DELAY = True**

**MinSec = 0.5 \* TOR\_CHANGING\_IP\_DELAY\_SEC**

**MaxSec = 1.5 \* TOR\_CHANGING\_IP\_DELAY\_SEC**

**[MinSec, MaxSec]**

با True قرار دادن مقدار این متغیر می‌توانیم موجب تغییر IP تور در بازه های زمانی(ثانیه) متفاوت به صورت رندوم انجام شویم. ( بعداً از این متغیر در TorMiddleware استفاده می کنیم.)

* **TOR\_CHANGING\_IP\_DELAY\_SEC = 120**

مقدار تأخیر در ارسال هر Request برای تغییر IP را با این متغیر مشخص می‌کنیم. ( بعداً از این متغیر در TorMiddleware استفاده می کنیم.)

* **TOR\_CONTROL\_PORT = 9051**

تنظیم پورت سرویس کنترل تور از طریق این متغیر انجام می شود. ( بعداً از این متغیر در TorMiddleware استفاده می کنیم.)

* **"REQUEST\_FINGERPRINTER\_IMPLEMENTATION = "2.7**

مشخص می‌کند که باید از کدام اسکرپی باید از کدام الگوریتم Request Fingerprinting استفاده کند.

* **TWISTED\_REACTOR = "twisted.internet.asyncioreactor.AsyncioSelectorReactor"**

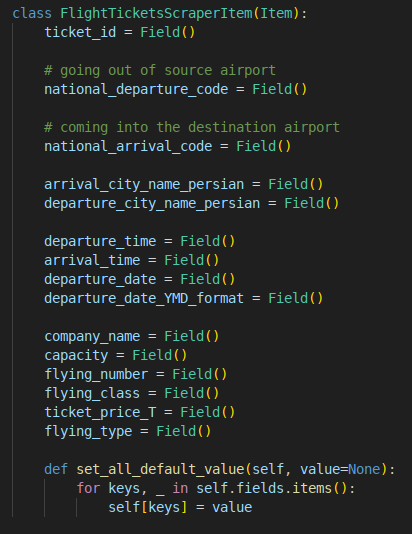
اسکرپی به صورت Async کار می‌کند و برای این امر از ری اکتور Twisted استفاده می‌کند.

* **FEED\_EXPORT\_ENCODING = "utf-8"**

با تنظیم این پارامتر اسکرپی از UTF-8 Encoding استفاده می کند.

**ماژول items.py :**

* **کلاس FlightTicketsScraperItem**

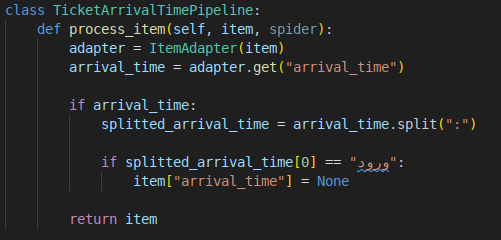


همانطور که در تصویر بالا مشاهده می کنید یک کلاس به اسم FlightTicketsScraperItem تعریف کرده‌ایم که از کلاس والدی به نام Item که توسط فریمورک Scrapy نوشته شده ارث بری کرده است. Class Attribure های این کلاس از کلاس دیگری به اسم Field ایجاد می‌شوند که در ورودی خود مقداری نمی‌گیرد. زمانی که یک Element را از سایت هدف استخراج کرده‌ایم برای اینکه مقدار آن را به صورت موقتی در زمان اجرای اسپایدر در دسترس داشته باشیم میتوانیم یک کلاس Item به شکلی که دیدید با فیلد های مد نظر ایجاد کنیم.

در بدنه کلاس از تابعی به نام set\_all\_default\_value استفاده کردیم زمانی که این تابع صدا زده شود به ازای تمام فیلد هایی که در آیتم شما موجود می‌باشد مقدار پیش‌فرض None را قرار می‌دهد که البته مشخص است می‌توان مقدار پیش‌فرض None را توسط پارامتر ورودی value می‌توان به صورت دلخواه تغییر داد.

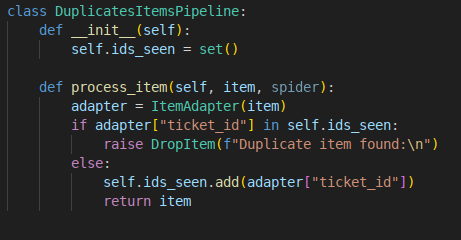
**ماژول pipelines.py :**

* **کلاس TicketArrivalTmePipeline**



وظیفه این کلاس ایجاد تغییر بر روی فیلد arrival\_time آن آیتم مربوطه می باشد. اسکرپی به صورت خودکار از این کلاس استفاده می‌کند و تابع process\_item را صدا می‌زند تا تغییر لازمه روی فیلد انجام شود. اتفاقی در این تابع می‌افتد این می‌باشد که در زمان کراول کردن سایت و استخراج فیلد arrival\_time زمانی که مقدار این فیلد وجود داشت و مقدار این فیلد برابر با ”ورود” بود مقدار آن را به None تغییر می‌دهیم به عبارتی ساده‌تر اگر مقدار فیلد arrival\_time برابر با فرمت «hour : minute» نبود مقدار آنرا برابر با None قرار می‌دهیم و سپس آبجکت item را بر می گردانیم و ادامه پروسه را به فریمورک اسکرپی میسپاریم.

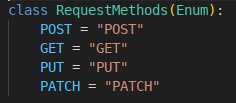
* **کلاس DuplicatesItemsPipeline**



همانطور که از اسم کلاس پیداست این کلاس توسط تابع \_\_init\_\_ یک Instance Variable به نام ids\_seen ایجاد می‌کند در این متغیر مجموعه id های هر بلیت قرار خواهد گرفت تا از ذخیره کردن یک آیتم تکراری جلوگیری شود. در ادامه نیز در تابع process\_item مقدار id هر بلیت را بررسی می‌کنیم اگر مقدار آن در مجموعه ids\_seen وجود نداشت یعنی این آیتم از قبل وجود ندارد و می‌تواند با return به ادامه رند اسکرپینگ بازگشت داده شود در مقابل اگر مقدار id در مجموعه وجود داشت با raise کردن یک ارور به کاربر اطلاع می‌دهیم که یک آیتم تکراری پیدا شده است.

**ماژول requests\_method\_enum.py :**

* **کلاس RequestMethods**

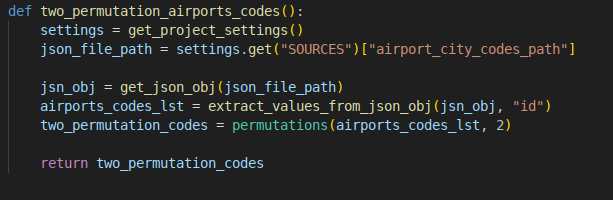


این کلاس از کلاس دیگر به Enum ارث بری کرده است که موجب می‌شود قابلیت یک مقدار شمارشی به این کلاس اضافه شود مقدار های شمارشی ای که ما از آن‌ها استفاده کردیم انواع اکشن رکيوئستHTTP مربوطه می‌باشد مثل POST ، GET ، PUT ، PATCH

استفاده از این ساختار کد به جای استفاده مستقیم مقادیر String موجب می‌شود کد ما خواناتر و تمیز تر باشد و علاوه بر آن مشخص است که این مقادیر رشته ایی برای چه چیزی می‌باشند که در مورد ما برای نمایش انواع روش‌های ارسال درخواست HTTP می باشند.

**ماژول utils.py :**

* **تابع two\_permutation\_airports\_codes**

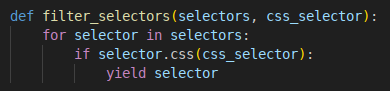
 با خواندن settings.py توسط تابع get\_project\_settings و دسترسی به مسیر فایل airport\_city\_codes.json توسط کلید airport\_city\_codes\_path در متغیر دیکشنری SOURCE میتوانیم این فایل json را توسط تابعی که در پکیج common\_utils به نام get\_json\_obj قرار دارد بخوانیم و در متغیر jsn\_obj قرار دهیم سپس توسط تابع extract\_values\_from\_json\_obj می‌توانیم به دنبال تمام کلیدهای id درون jsn\_obj بگردیم و آن‌ها را درون متغیر لیست airports\_codes\_lst نگه داریم و در نهایت توسط تابع permutations که در پکیج itertools قرار دارد می‌توانیم عمل جایگشت دوتایی مجموعه (نمونه برداری ترتیبی بدون جایگزینی دوباره) را انجام دهیم و خروجی این جایگشت را توسط کلمه رزرو شده return برگردانیم.

به طور مثال داریم:

[‘THR’, ‘MHD’, ‘ISF’] → Permutations 2 → [ (‘THR’, ‘MHD’), (‘THR’, ‘ISF’), (‘MHD’, ‘THR’), (‘MHD’, ‘ISF’), (‘ISF’, ‘THR’), (‘ISF’, ‘MHD’) ]

بدین ترتیب می‌توانیم در سایت هدف بررسی کنیم که بین چه مبدأ و مقصد هایی هواپیما وجود دارد.

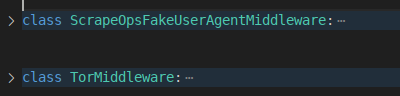
* **تابع filter\_selectors(selectors, css\_selector) :**

****

در ابتدا باید توضیح دهیم زمانی که درخواستی را به سایت هدف ارسال می‌کنیم سایت هدف برای ما یک response حاوی کدهای HTML بر می گرداند برای جستجو در این کدها و پیدا کردن Element مربوطه فریمورک اسکرپی دو روش به نام های Select via css و Select via xpath را اختیار ما می‌گذرد و اما کاربردی که تابع filter\_selectors برای ما دارد همانطور که از اسمش پیداست وظیفه فیلتر کردن تمامی selector ها را به ازای css\_selector که به آن می‌دهیم بر عهده دارد و در نهایت آن selector پیدا شده را با کلمه رزرو شده yield نگه می‌داریم تا بعداً در صورت نیاز از آن استفاده کنیم(کاربرد return با yield متفاوت می‌باشد وقتی از return استفاده می‌کنیم کار تابع در همان لحظه تمام می‌شود اما وقتی از yield استفاده می‌کنیم وضعیت فعلی تابع در حین اجرا نگه داشته خواهد شد تا زمانی که تابع دوباره فراخوانی شود و ما به وضعیت بعدی تابع برویم در‌ واقع ما یک lazy function توسط کلمه کلیدی yield ایجاد کرده‌ایم که هر موقع به آن نیاز داشتیم صدایش می‌زدیم تا از خروجی این generator استفاده کنیم.)

**نکته** : توابع دیگری نیز در ماژول utils.py وجود دارند ولی چون از آن‌ها استفاده ایی نشده از توضیح آن‌ها امتناع خواهیم کرد.

**ماژول middlewares.py** :

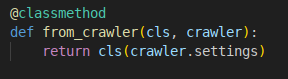


در این ماژول دو کلاس ScrapeOpsFakeUserAgentMiddleware و TorMiddleware را نوشته‌ایم این دو کلاس درخواست های HTTP را قبل از اینکه به درون اینترنت بروند به دلخواه ما دستکاری می‌کنند. وظیفه کلاس ScrapeOpsFakeUserAgentMiddleware این می باشدکه به API سایت ScrapeOps یک درخواست جهت دریافت FakeUserAgent تصادفی دورانی ارسال می کند (منظور از تصادفی دورانی این می‌باشد که تعداد این user-agent های فیک از طرف API محدود بوده و این سایت حول یک صف به صورت دورانی این user-agent ها را در اختیار ما قرار می‌دهد) تا در هدر user-agent خود به جای user-agent اسکرپی جایگزین کنیم. همانطور که مشخص است **یکی از راه‌های** جلوگیری از تشخیص ربات بودن کراولر توسط سایت هدف استفاده از این user-agent ها می باشد. از راه‌های دیگری که ما در settings.py به آن‌ها اشاره کردیم استفاده از Download\_Delay و Randomize\_Download\_Delay می باشند. در ادامه راجبه مابقی راه‌های موجود توضیحاتی را ارائه خواهیم داد.

البته ناگفته نماند که این روش‌های گفته شده با استفاده از متغیر ROBOTTXT\_OBEY متفاوت هستند چرا که با استفاده از این متغیر همچنان سایت متوجه می‌شود که ما ربات هستیم ولی استفاده از این متغیر موجب می‌شود که ربات ما از محدودیت موجود که توسط سایت هدف برای کراولر شدن توسط اسپایدرها تعیین شده فراتر نرود تا IP ربات توسط سایت بلاک نشود.

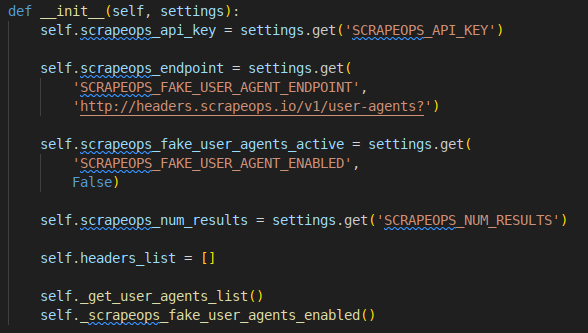
در ادامه می‌توان بیان کرد که وظیفه کلاس TorMiddleware تغییر IP کراولر به صورت تصادفی دروانی می‌باشد شبکه تور لیستی از گره‌های موجود با وضعیت آماده باش را که هرروز آپدیت می‌شوند در اختیار کاربران خودش قرار می‌دهد با وجود این لیست می‌توانیم درخواست هایی را به شبکه تور ارسال کنیم تا IP در مدت زمان های متفاوتی تغییر دهد. این عمل ما را از کمین سایت هدف برای بلاک کردن اسپایدرها نجات خواهد داد هرچند که باید در نظر داشت برای انجام این عمل سایت هدف باید و حتماً از exit node های شبکه تور پشتیبانی لازمه را داشته باشد.

* **کلاس ScrapeOpsFakeUserAgentMiddleware**
  + **تابع from\_crawler**



زمانی که وارد کلاس ذکر شده فوق شویم اولین تابعی که با آن برخورد خواهیم کرد تابع from\_crawler می‌باشد که با دکوریتور classmethod ترکیب شده است این دکوریتور به ما این اجازه را خواهد داد که بدون نیاز به ساختن یک obj و فقط با صدا زدن نام کلاس به این تابع دسترسی داشته باشیم خروجی تابع فوق کانفیگ هایی می‌باشد که در settings.py قرار دارند و قرار است در کلاس ذکر شده استفاده شوند.

* + **تابع \_\_init\_\_**

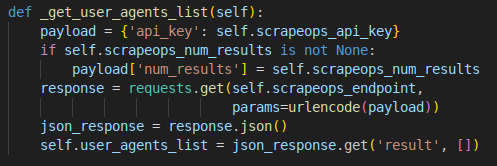


با استفاده از تابع \_\_init\_\_ در زمان ساخته شدن یک Instance از یک کلاس می‌توانیم پارامتر های اولیه لازمه را در آن مقدار دهی کنیم اگر دقت کنید می‌بیند که در ورودی تابع از آرگومان settings استفاده شده است که این آرگومان توسط تابع from\_crawler و crawler.settings مقدار دهی می‌شود تا setting های ما در دسترس middleware ها قرار گیرند. همانطور که تا الان متوجه شده‌اید کانفیگ هایی مثل SCRAPEOPS\_API\_KEY و SCRAPEOPS\_FAKE\_USER\_AGENT\_ENABLED را قبلاً دیده ایم و راجبه آن‌ها صحبت کردیم فقط اتفاقی که در تابع \_\_init\_\_ می‌افتد این است که با استفاده از تابع get (دیگه حتماً متوجه شدین که settings یه دیکشنریه چون تابع get برای دسترسی به مقدار یه کلید استفاده میشه) به مقدار تنظیم شده دسترسی خواهیم داشت حال اگر این کانفیگ در settings.py وجود داشت که مقدارش در Instance Variable مد نظر قرار خواهد گرفت اگر وجود نداشت میتوانی توسط تابع get مشخص کنیم چه مقدار پیش‌فرض در صورت عدم وجود کلید در آن قرار بگیرد(این کار موجب می‌شود تا در حین اجرا برنامه به ارور KeyError مواجه نشود) هر چند که اگر این مقدار پیش‌فرض را مشخص نکنیم تابع get مقدار None را در صورت عدم وجود برای عمل انتساب مد نظر قرار خواهد داد. با این توضیحات می‌توان به راحتی حدس زد که در هنگام خواندن مقدار کلید های SCRAPEOPS\_FAKE\_USER\_AGENT\_ENDPOINT و SCRAPEOPS\_NUM\_RESULTS چه اتفاقی رخ خواهد داد.

در self.headers\_list مقادیر header ها قرار خواهند گرفت( البته ما از این متغیر استفاده نمی‌کنیم ولی به پیشنهاد داکیومنت ScrapeOps تو کد قرارش دادیم)

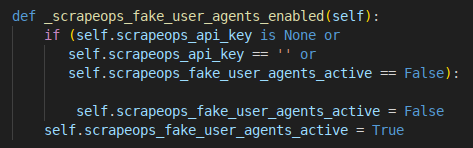
در رابطه‌های تابع های صدا زده شده در تابع \_\_init\_\_ مثل get\_user\_agents\_list و scrapeops\_fake\_user\_agents\_enabled در ادامه توضیح خواهم داد هرچند که از روی اسم آن‌ها واضح است چه عملی را برای ما انجام می دهند.

* + **تابع get\_user\_agents\_list**



طبق داکیومنت سایت ScrapeOps در هنگام استفاده از API باید یک payload ، form یا همان body را به سمت API ارسال کرد که این payload می‌تواند شامل api\_key و num\_results باشد. همانطور که از کد مشخص است استفاده از api\_key اجباری ولی استفاده از num\_results اختیاری است اگر آنرا تنظیم کنید می‌توانید مشخص کنید که چه تعداد user-agent فیک را به شما برگرداند. باید توجه کرد که فرمت ارسال payload توسط کتابخانه requests حتماً باید به صورت دیکشنری و همچنین به گفته داکیومنت ScrapeOps حتماً باید به صورت Encode شده درون URL یا همان Endpoint مد نظر ارسال شوند. تابع get نیر در کتابخانه requests وظیفه ارسال درخواست HTTP با نوع GET را برعهده دارد در نهایت نیز بعد از آماده سازی payload و ارسال آن توسط پارامتر params به تابع get نوبت خواندن response دریافتی توسط تابع json و ذخیره json در متغیر json\_respone می‌باشد در مرحله آخر با کمک تابع get دیکشنری (دیگه حتماً می دونید که فرمت دیکشنری پایتون مثل json می‌ماند و در‌واقع پایتون با فرمت json مثل یک دیکشنری رفتار می کند) می‌توانیم مقدار کلید result موجود در json را بخوانیم و خروجی آنرا در متغیر Instance به نام user\_agents\_list ذخیره کنیم.

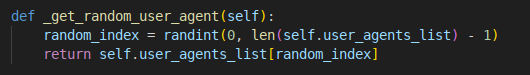
* + **تابع scrape\_fake\_user\_agents\_enabled**



راستش خود بنده هم نفهمیدم **هدف** از ایجاد این تابع توسط داکیومنت ScrapeOps چی می باشد زیرا که در نهایت خروجی self.scrapeops\_fake\_user\_agents\_active چه شرط اجرا بشود و چه نشود برابر با مقدار True خواهد بود اما در هرحال به پیشنهاد داکیومنت ما از آن در کد خود استفاده کرده ایم.

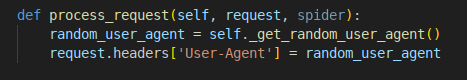
شرط فوق به ما می‌گوید اگر self.scrapeops\_api\_key هویتی برابر با None داشته باید یا مقدارش برابر با ”” باشد یا مقدار self.scrapeops\_fake\_user\_agents\_active برابر با مقدار لیترال False باشد. در نهایت مقدار self.scrapeops\_fake\_user\_agents\_active برابر با False شود.

* + **تابع get\_random\_user\_agent**



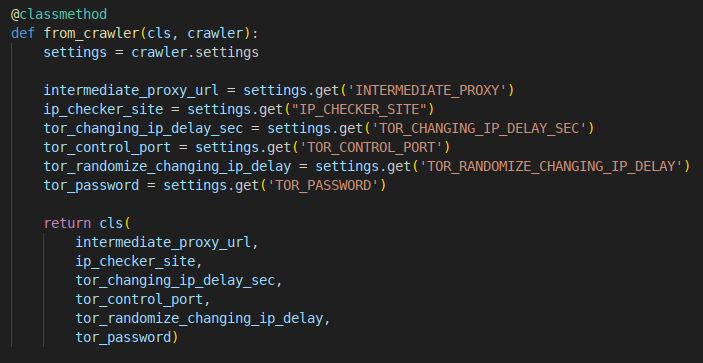
اگر یادتان باشد خروجی تابع get\_user\_agents\_list مقدار self.user\_agents\_list می‌باشد که همان response سایت ScrapeOps به ما بوده است. فقط ما در تابع فوق صرفاً عمل انتخاب تصادفی بین از اعضای لیست موجود را با تابع randint کتابخانه random در بازه ۰ تا تعداد اعضای لیست مربوطه انجام داده ایم.

* + **تابع process\_request**

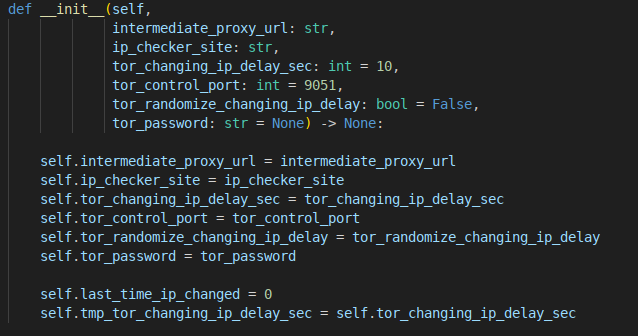


همانطور که اشاره شد خروجی ScrapeOpsFakeUserAgentMiddleware باید یک user-agent فیک باشد که توسط سایت ScrapeOps به ما ارائه می‌شوند. در نهایت خروجی مد نظر را توسط تابع process\_request در بخش headers درخواست خود قرار خواهیم داد تا تغییر مربوطه بر روی درخواست HTTP اعمال شود.

* **کلاس TorMiddleware**
  + **تابع from\_crawler**

با تابع from\_crawler در ScrapeOpsFakeUserAgentMiddleware آشنا شده‌ایم همان عمل‌کردی که در کلاس گفته شده داشتیم همان عمل‌کرد را در کلاس TorMiddleware خواهیم داشت با این تفاوت که فقط در اینجا کانفیگ های بیشتری را از settings.py می‌خوانیم و به تابع \_\_init\_\_ می فرستیم باید توجه شود که حتماً **ترتیب** برگرداندن متغیرها در تابع from\_crawler و دریافت آن‌ها در تابع \_\_init\_\_ باید یکسان باشد وگرنه مقادیر متغیرها جا به جا خواهند شد و این اتفاق مساعد نیست و کد شما را با مشکلاتی همراه خواهد کرد.

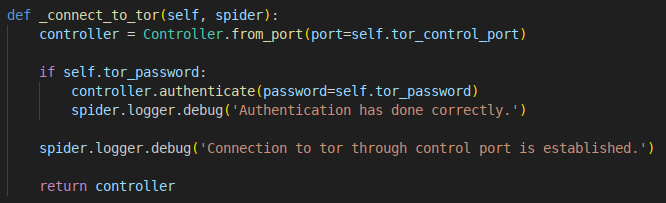
* + تابع \_\_init\_\_

****

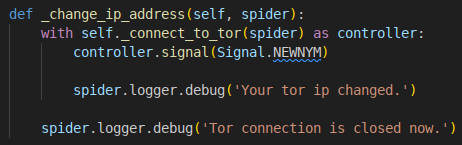
کلاس TorMiddleware برای اینکه بتواند درست کار کند به مقادیر دهی اولیه ایی که توسط تابع \_\_init\_\_ انجام می‌شود نیاز دارد همانطور که مشاهد می‌کنید یکسری از آرگومان های تابع فوق دارای مقدار پیش‌فرض هستند این یعنی اینکه اگر شما این متغیرها را از طریق settings.py مقدار دهی نکرده باشید تا به واسطه تابع from\_crawler در اختیار تابع \_\_init\_\_ قرار گیرند در نتیجه این آرگومان توسط مقدارهای پیش‌فرض مشخص شده درون عکس مقدار دهی خواهند شد ( به عبارتی این متغیرها اختیاری هستند و میتوانید در صورت لزوم settings.py آن‌ها مقدار دهی نکنید.) اما حتماً باید مقدار آرگومان های intermediate\_proxy\_url و ip\_checker\_size از قبل اجرا در settings.py مشخص کنید(اگر نمیدانید این متغیرها چیکاری انجام می‌دهند به بخش ماژول settings.py در زیر پروژه flight\_tickets\_scraper مراجعه کنید.)

علاوه بر موارد گفته شده متغیرهای Instance به نام self.last\_time\_ip\_changed و self.tmp\_tor\_changing\_ip\_delay\_sec وجود دارند به ترتیب این متغیر ها برای آخرین زمانی که Tor IP تغییر کرده و مقدار تأخیر موقتی مورد نیاز ( اگر مقدار TOR\_RANDOMIZE\_CHANGING\_IP\_DELAY برابر با True باشد آنگاه مقدار self.tmp\_tor\_changing\_ip\_delay\_sec تغییر خواهد کرد) برای تغییر IP فعلی می باشد.

* + **تابع connect\_to\_tor**

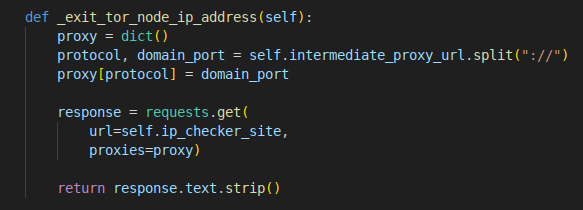
از این تابع برای وصل شدن به سرویس تور استفاده کرده‌ایم این تابع به این طریق کار می‌کند که در ابتدا با استفاده از کلاس Controller و classmethod به نام from\_port به TorController که پورت آن را از قبل روی سیستم باز کرده‌ایم متصل می‌شویم و مطابق تصویر یک آبجکت controller اینجا خواهیم کرد حال اگر پسوردی بر روی سرویس تور تنظیم کرده باشیم عمل احراز هویت را توسط متد authenticate انجام می‌دهیم و در یک لاگ به گزارش می‌کنیم که عمل احراز هویت با موفقیت انجام شد و موفقیت آمیز بود احراز هویت به معنی ایجاد اتصال با سرویس تور می‌باشد که آنرا هم در لاگ سیستم گزارش می کنیم. در نهایت آبجکت controller را بر میگردانیم.

* + **تابع change\_ip\_address**

****

این تابع توسط تابع connect\_to\_tor و کلمه رزرو شده with یک اتصال باز با سرویس تور روی سیستم شما ایجاد می‌کند و با ارسال سیگنال Signal.NEWYM توسط متد signal به سرویس تور بازگو می‌کند که IP را برای کراولر ما تغییر دهد. در نهایت بعد از انجام این مراحل لاگ تغییر IP را در سیستم ثبت می‌کنیم و همچنین گزارش می‌کنیم که اتصال با سرویس تور توسط مدیریت کننده ریسورس with بسته شد.

* + **تابع exit\_tor\_node\_ip\_address**

****

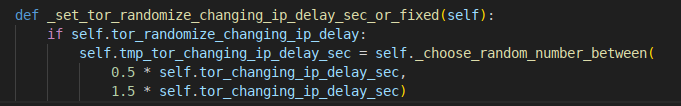
شبکه تور به طور **پیش‌فرض** از سه لایه relay (همان گره) استفاده می‌کند تا ترافیک مد نظر شما را منتقل کند در لایه آخر گره ایی وجود دارد به نام exit node یا گره خروجی که وظیفه ارسال درخواست شما به سایت هدف را برعهده دارد (در واقع این گره با سایت هدف ارتباط می‌گیرد و نتیجه را به شما می‌دهد بدون اینکه سایت هدف بداند آدرس IP واقعی شما چیست این همان چیزی است که ما به آن معجزه Tor Proxy می گوییم) اما کتابخانه stem این قابلیت را ندارد که آدرس IP گره خروجی را برای ما گزارش کند(در اصل این کار خیلی سخت و دشواره) و تنها راه فهمیدن آدرس گره خروجی استفاده از سایت‌ها و یا API هایی هست که به ما آدرس IP فعلی مارا گزارش می‌کنند به همین منظور است که ما متغیر IP\_CHECKET\_SITE را در setting.py مقدار دهی می‌کنیم چرا که می‌خواهیم با استفاده از سایت مد نظر به IP گره خروجی خود پی ببریم و این دقیقاً کاری است که تابع exit\_tor\_node\_ip\_address برای ما انجام خواهد داد از تابع split استفاده می‌کنیم تا مقدار protocol و domain port موجود در متغیر مقدار دهی شده INTERMEDIATE\_PROXY را از هم جدا کنیم از این امر استفاده می‌کنیم تا به requests بگوییم برای ارسال درخواست به سایت مد نظر از چه پروکسی باید گذر کنی (کتابخونه requests به صورت پیش‌فرض نمیدونه که سرویس تور روی سیستم ما فعاله و باید از اون استفاده کنه تا IP گره خروجی رو بتوانیم مشاهده کنیم به همین دلیل است که از Privoxy کمک گرفته‌ایم اگر از Privoxy استفاده نکنیم اونوقت سایت مد نظر آدرس IP واقعی ما را بر می گرداند نه آدرس IP گره خروجی شبکه تور)

* + **تابع choose\_random\_number\_between**

****

وظیفه تولید یک عدد تصادفی اعشاری با تخمین یک رقم اعشار در بازه عددی مشخص را بر عهده دارد. به دلیل اینکه از تابع uniform در کتابخانه random استفاده کرده‌ایم این قابلیت را نیز به ما می‌دهد تا انتخاب تصادفی هر کدام از اعداد بین بازه مشخص شده باهم برابر باشد.

* + **تابع \_set\_tor\_randomize\_changing\_ip\_delay\_sec\_or\_fixed**

در صورتی که مقدار self.tor\_randomize\_changing\_ip\_delay برابر با True باشد، مقدار متغیر Instance به نا

self.tmp\_tor\_changing\_ip\_delay\_sec در بازه عددی مشخص تغییر خواهد کرد. این امر موجب خواهد شد تا IP ما در بازه های زمانی تصادفی تغییر کند.

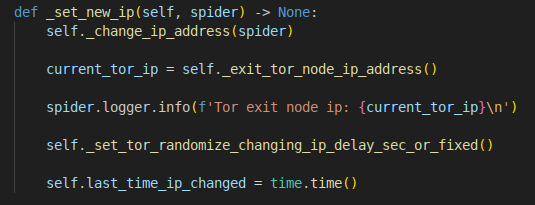
بازه ها به صورت زیر می باشد:

[ MinDelay, MaxDelay ]

MinDelay = 0.5 \* self.tor\_changing\_ip\_delay\_sec

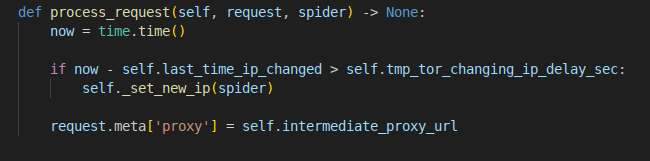
MaxDelay = 1.5 \* self.tor\_changing\_ip\_delay\_sec

* + **تابع set\_new\_ip**

****

حال که توابع بالا را توضیح دادیم کاملاً مشخص است که تابع set\_new\_ip به چه بخش‌هایی تقسیم شده و با صدا زدن تابع set\_new\_ip هر بخش چه کاری را برعهده دارد فقط لازم به ذکر است که هر بار یک IP جدید تنظیم کردیم باید مقدار متغیر self.last\_time\_ip\_changed را با تابع time کتابخانه time مقدار دهی کنیم تا بعداً بتوانیم بررسی کنیم که در چه زمانی باید دوباره تابع set\_new\_ip را صدا بزنیم تا IP جدیدی را به کراولر ما انتساب دهد.

* + **تابع process\_request**



هر بار که اسکرپی درخواستی ارسال می‌کند و درخواست به TorMiddleware می‌رسد این تابع به صورت خودکار صدا زده خواهد شد زمانی که این تابع صدا زده شد مقدار متغیر now برابر زمان فعلی می‌باشد که این تابع صدا زده شده از این متغیر استفاده می‌کنیم تا اختلافش را با آخرین زمانی که IP تغییر کرده بسنجیم و اگر این اختلاف بیشتر از مقدار تأخیر مورد نیاز برای تغییر IP بوده است باید تابع set\_new\_ip صدا زده شود تا IP جدیدی به کراولر انتساب داده شود. در نهایت هم برای اتصال اسکرپی با سرویس تور از آدرس Privoxy استفاده می‌کنیم و آن را در کلید proxy دیکشنری meta آبجکت request قرار می دهیم.

**نکته** : کلاس‌های دیگری نیز در ماژول middlewares.py وجود دارند ولی این کلاس‌ها توسط خود فریمورک اسکرپی ایجاد شده‌اند و ما از آن‌ها در طول پروژه استفاده نکرده‌ایم به همین دلیل از توضیح ها امتناع خواهم کرد.

**ماژول tcharter\_airlines\_tickets\_spider.py** :

در این ماژول اسپایدرهای خود را کد نویسی می‌کنیم ما در این ماژول صرفاً یک اسپایدر به نام AirlinesTickets نوشته‌ایم و این اسپایدر در **چهار سطح** عمل Parse کردن سایت را برای ما انجام خواهد داد. به طور کلی parse کردن به معنی تجزیه و تحلیل کردن یک ساختار می‌باشد و به طور دقیق‌تر در مبحث Web Scraping عمل parse به معنی استخراج کردن اطلاعات در جهت یافتن Element های مد نظر از سورس های HTML و همچنین فرمت کردن آن‌ها برای استفاده کردن از آن‌ها در آینده می باشد.

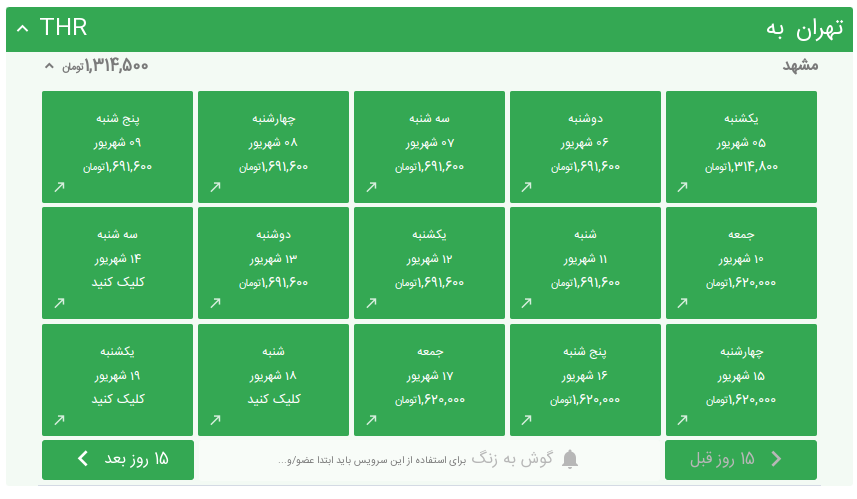
همانطور که در متن اشاره کردیم parse کردن ما در چهار سطح انجام خواهد شد این چهار سطح به این معنی می‌باشد که همه اطلاعات مد نظر ما فقط با فرستادن درخواست به یک URL بدست نمی آیند و ما باید مرحله به مرحله به اندازه چهار سطح جلو برویم تا بتوانیم تمام فیلدهای مد نظر را برای آیتم های خود جمع آوری کنیم. قبل از اینکه به سراغ توضیح کدهای اسپایدر برویم این چهار سطح را به صورت گرافیکی به شما نشان می‌دهیم تا متوجه شوید که یک کاربر واقعی چگونه با یک مرورگر به این اطلاعات دسترسی خواهد داشت و بعد به همان شکل باید کدهای خود را بنویسم.

**سطح اول:**

در سطح اول با ارسال درخواست به آدرس زیر:

**www.tcharter.ir//tickets/search/0/{source}-{destination}**

تصویر زیر را در نهایت مشاهده خواهیم کرد:



مقدار source و destination در‌واقع شناسه سه حرفی بین‌المللی فردوگاه مبدأ و مقصد می باشند.

هر بار که با آدرس ذکر شده درخواستی را ارسال می‌کنیم نتیجه ایی را **مشابه** تصویر بالا مشاهده خواهیم کرد نتیجه ایی که سایت هدف به ما نمایش می‌دهد از چهار section تشکیل شده که در هر section تعداد روزهایی به اندازه عدد ۱۵ وجود خواهد داشت. در‌واقع کلیک برروی دکمه **”۱۵ روز بعد”** یا **”۱۵ روز قبل”** section بعدی و قبلی را به شما نشان می دهد.

**سطح دوم:**

در سطح دوم درخواستی به آدرس زیر ارسال می کنیم:

**www.tcharter.ir/tickets/dates/{source}-{destination}-airplane?section={section\_counter}**

مقدار source و destination همان مقدار هایی هستند که در سطح قبلی توضیح دادیم و اما مقدار section\_counter این مقدار به هر section که در سایت مشاهده می‌کنید انتساب داده می‌شود که مقدار آن بین بازه [4, 1] می باشد.

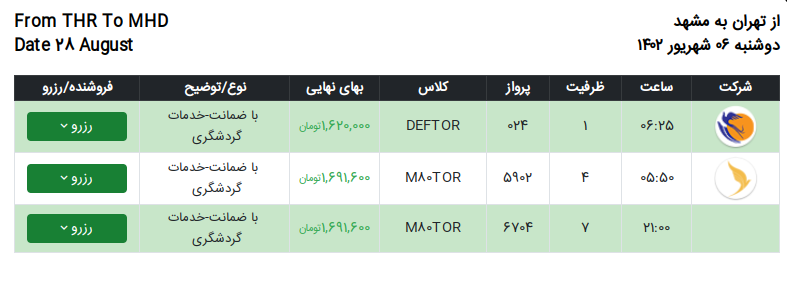
در‌ واقع با آدرسی که در بالا دارید می‌توانید بین هر section جا به جا شوید دقیقاً همان عملی که با کلیک بر روی دکمه های ”۱۵ روز بعد” و ”۱۵ روز قبل” اتفاق می افتد.

**سطح سوم:**

در این سطح با ارسال درخواست به آدرس زیر:

**www.tcharter.ir/tickets/tickets/{city\_airline\_date\_code}/**

تصویری **مشابه** تصویر زیر را مشاهده خواهیم کرد:

مقدار city\_airline\_data\_code متفاوت با هر پرواز بین مبدأ و مقصد و همچنین متناسب با تاریخ و روز حرکت متغیر است. در‌واقع مقدار این پارامتر ترکیبی به طول ۲۴ از کاراکتر ها و اعداد است که توسط خود سایت با توجه به مواردی که به آن‌ها اشاره کردیم تولید می شود. فرض می‌شود که سایت از این کد تولید شده برای بازیابی پروازهای مد نظرخ خویش در تاریخ انتخابی استفاده می کند.

توجه شود ممکن است در این تصویر که مشاهده می‌کنید یک جدول در چند صفحه با ایجاد حالت Pagination داشته باشیم. برای اینکه بتوانیم به هر صفحه این جدول دسترسی داشته باشیم از آدرس زیر استفاده می کنیم:

**www.tcharter.ir/tickets/tickets/{city\_airline\_date\_code}?page={next\_page\_number}**

آدرس بالا مشابه آدرس سطح سومی است که مشاهده کردید منتهی یک پارامتر page به آن اضافه شده که با next\_page\_number مقدار دهی خواهد شد. با استفاده از آدرس بالا می‌توانید بین هر صفحه جدول منتقل شوید.

**سطح چهارم:**

در این سطح **به ازای هر سطر** جدول بلیت ها با ارسال درخواست به آدرس زیر:

**www.tcharter.ir/reservations/step1/airplane/{reservation\_number}**

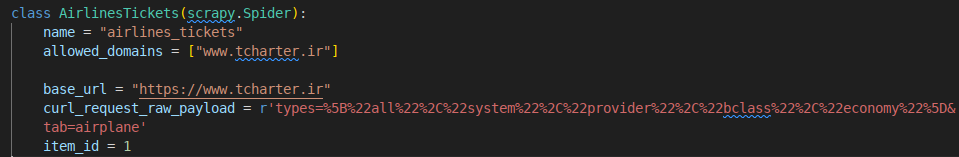
تصویر **مشابه** زیر را مشاهده خواهید کرد:



در‌واقع وقتی که تا سطح سوم پیش برویم اطلاعات لازمه را از هر سطر جدول می‌توانیم استخراج کنیم اما به نظر بنده در مرحله اسکرپینگ هر آنچه که سایت ارائه می‌دهد را استخراج کنیم بهتر می‌باشد(بعدا اگر نیازشان نداشتیم خیلی راحت می‌توانیم آن‌ها حذف کنیم ولی به فرض اگر نیاز داشتیم باید برگردیم و با بررسی لاجیک کار خود دوباره کراولر خود را کامل‌تر کنیم حال فرض کنید که مجبور باشید داده‌های زیادی را از سایت استخراج کنید آن وقت باید مدت زمان بیشتری را منتظر باشید تا کراول شما تمام شود.) پس ترجیه دادم یک سطح دیگر هم به کراولر خود اضافه کنم تا اطلاعات اضافی مثل ساعت ورود، ساعت خروج و تاریخ حرکت را نیز استخراج کرده باشم.

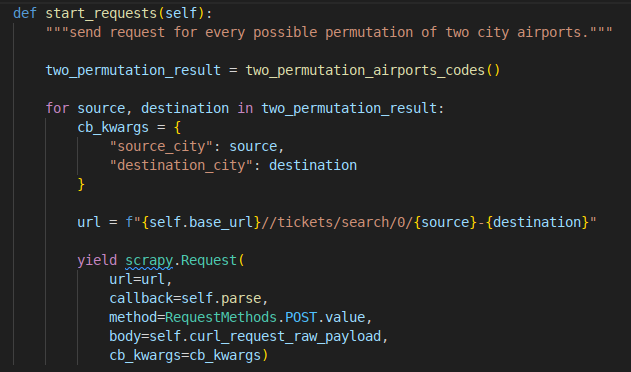
* **کلاس AirlinesTickets**

خب بعد از تمامی توضیحاتی که تا به الان راجبه ساختار فریمورک اسکرپی و ماژول هایی که در زیر پروژه flight\_tickets\_scraper وجود دارند ارائه دادیم حال نوبت به بخش اصلی پروژه یعنی استفاده از تمام قابلیت‌هایی کد نویسی کردیم و یا در فایل settings.py قرار دادیم می‌باشد. قبل از ورود به متد های موجود در کلاس AirlinesTickets ابتدا باید نگاهی به Class Variable این کلاس بی اندازیم:

متغیرهای کلاس را در عکس مشاهده می‌کنیم حال به ترتیب راجبه هرکدام توضیحاتی می دهم:

1. **name** : این متغیر اسم اسپایدر ما می‌باشد و اسکرپی اسپایدر ما را به این نامی که مشخص کرده‌ایم خواهد شناخت و هر موقع نیاز بود که اسپایدر ما اجرا شود و عملیات خود را آغاز کند از این نام استفاده خواهیم کرد.
2. **allowed\_domain** : این متغیر مشخص می‌کند به چه دامنه هایی مجاز به دسترسی هستیم.
3. **base\_url** : این متغیر URL اصلی سایت هدف می‌باشد از طریق این URL به Endpoint های سایت هدف می‌توانیم دسترسی داشته باشیم.
4. **curl\_request\_raw\_payload** : مقدار خام payload را مشخص می‌کند که در درخواست های cURL استفاده می شوند. این مقدار با آزمون و خطاهای متعدد بدست آمده در‌واقع سایت هدف با فرستادن payload به روش عادی به ما خروجی مناسب را نمی‌دهد در نتیجه از این روش برای فرستادن payload مورد نیاز استفاده کرده ایم.
5. **item\_id** : از این متغیر برای انتساب id به هر بلیت استفاده خواهیم کرد.

* **تابع start\_requests**

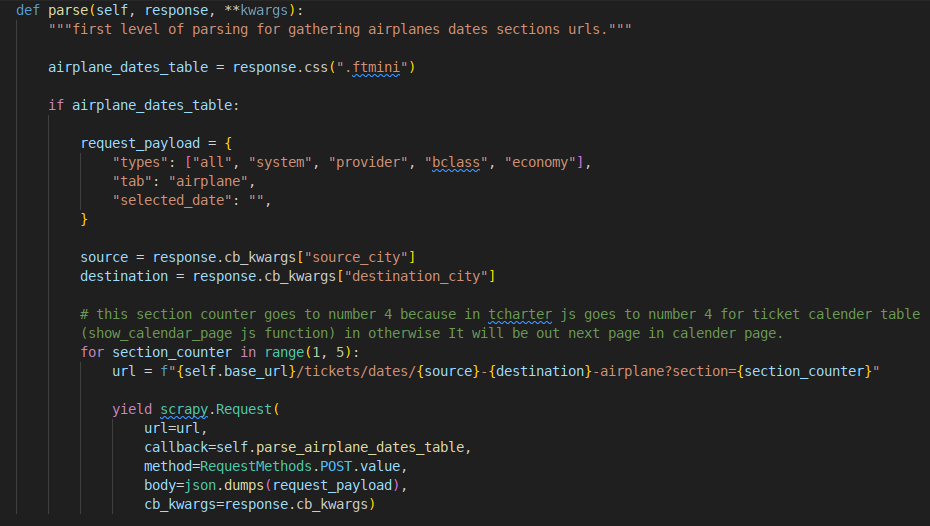


تابع start\_request همانطور که از اسمش پیداست اولین تابعی است که شما از آن برای شروع عملیات درخواست HTTP و کراول استفاده خواهید کرد و همچنین گفتنی است که این تابع متعلق به والدی است که ما در کلاس AirlinesTickets از آن ارث بری کرده ایم( به عبارتی از Polymorphism در اینجا بهره برده ایم). خب با توضیحاتی که از ابتدای داکیومنت تا به الان ارائه کردیم توضیح دادن این قطعه کد و فهمیدنش نباید کار سختی باشه در خط اول تابع فوق شاهد استفاده از تابع two\_permutation\_airports\_codes هستید که از ماژول utils می‌باشد و همانطور که قبلاً گفتیم این تابع جایگشت های دوتایی مجموعه ایی از شناسه های فردوگاه ها را به ما می‌دهد و در ادامه در حلقه ایی که نوشتیم به ازای جایگشت های دوتایی بدست آمده ما یک شناسه برای فردوگاه مبدأ و یک شناسه برای فردوگاه مقصد خواهیم داشت این مقادیر را با کلید هایی درون دیکشنری cd\_kwargs قرار خواهیم داد و همچنین از شناسه ها در url استفاده می‌کنیم و در نهایت یک درخواست HTTP با کلاس Request ایجاد می کنیم. طبق عکس فوق کلاس Request برای ورودی های خود به مقادیر زیر احتیاج دارد:

1. **url** : آدرسی که است که میخواهیم آنرا کراول کنیم.
2. **callback** : ورودی این پارامتر یک تابع parser می‌باشد به عبارت دیگر با این پارامتر مشخص می‌کنیم بعد اینکه response از سایت هدف بازگشت برای parse کردن آن به کدام تابع آنرا ارجاع دهیم.
3. **method** : روشی که با آن درخواست HTTP باید ارسال شود. در اینجا ما از روش POST استفاده کردیم که به صورت Enum می باشد.
4. **body** : بدنه یا payload که نیاز آدرس URL مد نظر به آن نیاز دارد تا خروجی مطلوب را به تحویل دهد.
5. **cb\_kwargs** : از این پارامتر استفاده کرده‌ایم تا مقادیری که در parser های بعدی به آن‌ها نیاز داریم را منتقل کنیم

تا الان باید متوجه شده باشید که چگونه ما یک درخواست HTTP که روزانه در مرورگر شما و میلیون ها کاربر عادی اتفاق می‌افتد را در کدنویسی با فریمورک اسکرپی شبیه سازی کرده ایم. این نوع شبیه سازی به کرات در دنیای برنامه استفاده می‌شود و در ادامه کدهای همین پروژه نیز به شکل مشابه چندین بار استفاده شده است.

* **تابع parse**



تابع parse نیز تابعی می‌باشد که توسط والد کلاس اسپایدر AirlinesTickets پیاده‌سازی شده است و ما آن را در اینجا Override کرده‌ایم. این تابع وظیفه parse کردن در **اولین سطح** را کراول را برعهده دارد.

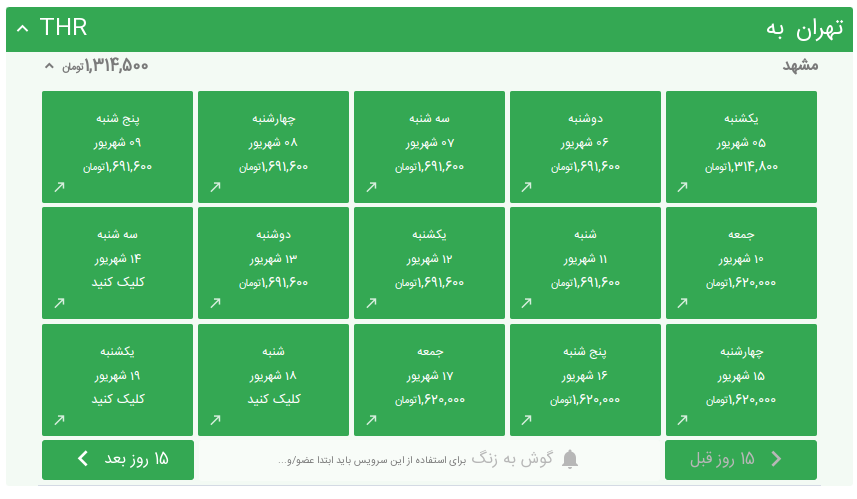
در اولین خط تابع توسط css\_selector در respone دریافتی به دنبال جدول تاریخی که قبلا آن را به شما نشان دادیم می‌گردیم و خروجی را درون متغیر airplane\_dates\_table قرار می‌دهیم و با یک شرط وجود یا عدم وجود جدول را بررسی می‌کنیم زیرا که منطقی است که ما از همه فرودگاه ها به همه فردوگاه بلیت برای فروش نخواهیم داشت این شرط موجب خواهد شد که در ادامه رکیوئست های اضافی را به سمت سایت هدف نفرستیم. در ادامه مقادیر source و destination را از طریق متغیر cb\_kwargs که در مرحله قبل به تابع parse انتقال دادیم مقدار دهی کردیم و همچنین payload مورد نیاز را برای رکيوئست مد نظر را فراهم کردیم.

یک حلقه ایجاد کردیم که به ازای محدوده ۱ تا ۵ آدرس URL هر section را برای میسازد و در نهایت با کلاس Request باز هم درخواست HTTP را برای مرحله بعدی parse شبیه سازی می کنیم. اگر به پارامتر body کلاس Request دقت کنید متوجه خواهید شد که مقداری متفاوت از body است که در مرحله قبل استفاده کردیم در اینجا از تابع dumps کتابخانه json بهره برده ایم که تابع به ما کمک می‌کند تا آبجکت دیکشنری ساخته شده را به رشته تبدیل کنیم(ساختار دیکشنری دست نخواهد خورد فقط کل دیکشنری ما با رشته تبدیل خواهد شد).

* **تابع parse\_airplane\_dates\_table**



این تابع وظیفه استخراج داده در سطح دوم عملیات کراول را بر عهده دارد. در خط اول این تابع می‌بینید که از response دریافت شده توسط css\_selector که استفاده کردیم (.daterow) تمام grid ها را از جدول تاریخ ها در اولین section بیرون کشیدیم و در متغیر city\_airline\_date\_rows قرار دادیم و سپس توسط تابع filter\_selector آن grid هایی که دارای سی اس اس سلکتور .currency\_symbole هستند را استخراج کردیم. به تصویر زیر قبلاً به شما نشان دادیم دوباره دقت کنید:



این تصویری بود که در سطح اول به شما نشان دادیم در سطح دوم میخواهیم آن grid هایی که دارای قیمت بلیت هواپیما هستند را استخراج کنیم از دلایلی که چرا همه grid ها را در هر section با هم استخراج نمی‌کنیم در زیر آمده است:

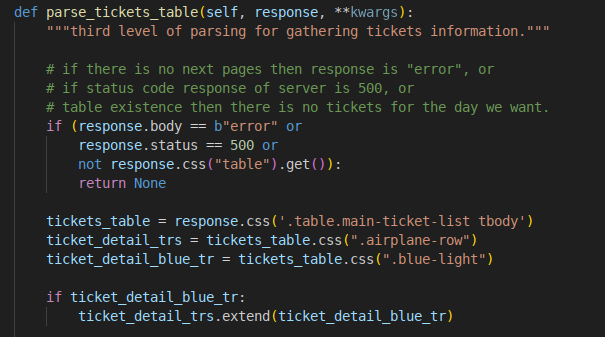
1. کاهش فزاینده تعداد رکیوئست هایی که به سایت هدف ارسال می‌کنیم چرا که نیاز نیست وقتی در آن grid سایت هدف جدولی به ما ارائه نمی‌دهد ما یک رکیوئست اضافی به سایت ارسال و برای سرور هایش سربار ایجاد کنیم. (چرا؟ چون فقط میخوایم مجانی و با زور از سایتش داده جمع آوری کنیم؟)
2. سایت هدف در grid هایی که قیمت در آن‌ها وجود ندارد جدول بلیت ها را ارائه نمی دهد.

در ادامه کد ها می بینید که به ازای grid های فیلتر شده متغیرهای city\_airline\_date\_code و date\_values را از طریق css\_selector هایی که از سایت هدف پیدا کرده‌ایم مقدار دهی می کنیم. دیکشنری cb\_kwargs را طبق عکس با کلیدهایی مقدار دهی کرده تا در تابع parse بعدی از آن‌ها استفاده کنیم و در نهایت درخواست HTTP را شبیه سازی و ارسال می کنیم.

* **تابع parse\_tickets\_table**

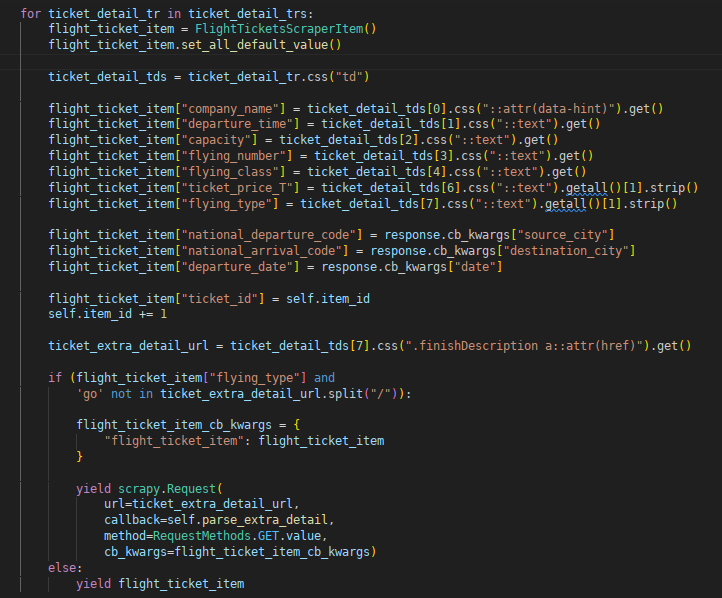
کدهای مربوط به این تابع را در چند بخش توضیح خواهیم داد:

**بخش اول:**



این تابع وظیفه کراول در سطح سوم را برعهده دارد در‌واقع درخواست از سطح دوم ارسال می‌شود و نتیجه آن در تابع فوق کراول خواهد شد. در بدو شروع بدنه تابع مشاهده می‌کنید که ما از یک شرط استفاده کرده‌ایم این شرط در‌واقع شرط تابع بازگشتی ما می‌باشد در‌واقع تابع فوق یک Parser بازگشتی است به این دلیل که وقتی شما به سطح سوم کراول برسید نیاز دارید همه اطلاعات جدول بلیت ها را که در قالب pagination به صورت صفحه به صفحه ارائه شده‌اند را به شکلی **یکنواخت** parse کنید. خیلی واضح این شرط به ما می‌گوید اگر body ریسپانس برابر با رشته ”error” بود یا status برابر با مقدار ۵۰۰ بود و یا جدول بلیت ها پیدا نشد مقدار None را برگردان و دیگر به کراول کردن ادامه نده این بدان معنی است که یا صفحات pagination تمام شده و صفحه بعدی وجود ندارد یا سایت هدف در آن لحظه جدولی نمی‌تواند ارائه دهد با اینکه ما از grid هایی استفاده کردیم که هزینه بلیت در آن‌ها نوشته شده بود و یا اینکه به دلایلی سایت هدف به اررور ۵۰۰ خورده است. توسط css\_selector هایی که در تصویر می‌بینید respone را parse کرده و متغیر های tickets\_table و ticket\_detail\_trs و ticket\_detail\_blue\_tr را مقدار دهی می‌کنیم. گفتنی است که ticket\_table در‌واقع کدهای HTML جدول بلیت ها، ticket\_detail\_trs تمام سطرهایی با رنگ سبز و سفید و ticket\_detail\_blue\_tr تمام سطرهایی با رنگ آبی می باشند. شرط بعدی هم مشخص می‌کند که اگر مقدار ticket\_detail\_blue\_tr پر بود اعضای مجموعه این متغیر به متغیر ticket\_detail\_trs اضافه شوند.

بخش دوم:

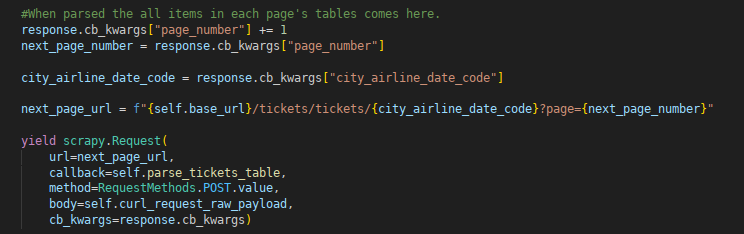
****

حال به ازای تمامی سطرهایی که از جدول بلیت ها در آن تاریخ مد نظر پیدا کرده‌ایم وقت این است که یک آبجکت از کلاس FlightTicketsScraperItem که نوشته‌ایم ایجاد کنیم و در ابتدا توسط متد set\_all\_defualt\_value آیتم خود را مقدار دهی اولیه کرده و سپس بعد از استخراج داده‌ها توسط css\_selector هایی که در تصویر می‌بینید مقدار کراول شده را به آیتم انتساب می دهیم. همانطور که مشاهده می‌کنید مقداردهی به آبجکت آیتم دقیقاً به مانند مقدار دهی به صورت کلید-مقدار در دیکشنری می باشد. در رابطه با متدهای دیگر که در تابع فوق استفاده شده می‌توان به متد get ، getall و strip اشاره کرد به ترتیب متد getall تمام المنت هایی که selector پیدا کرده را بر‌می‌گرداند، متد get صرفاً اولین المنت پیدا شده توسط selector را بر‌می‌گرداند و متد strip نیز متد آبجکت String می‌باشد که وظیفه آن پاک کردن تمامی white space های درون رشته مد نظر می باشد. حال بعد از مقدار دهی به آیتم خود باید آدرس URL سطح بعدی یعنی سطح چهارم را توسط attribute با نام href از طریق سلکتور “finishDescription a” استخراج کنیم ولی باید توجه کرد که طبق سایت هدف این آدرس همیشه آن آدرسی که ما برای سطح چهارم می‌خواهیم نیست از شرایطی که آدرس استخراج شده مناسب نمی باشد در زیر بیان شده اند:

1. کلید flying\_number در آیتم مربوطه باید دارای مقداری غیر از مقدار None باشد.
2. آدرس URL استخراج شده در متغیر ticket\_extra\_detail\_url نباید حاوی کلمه ”go” باشد.

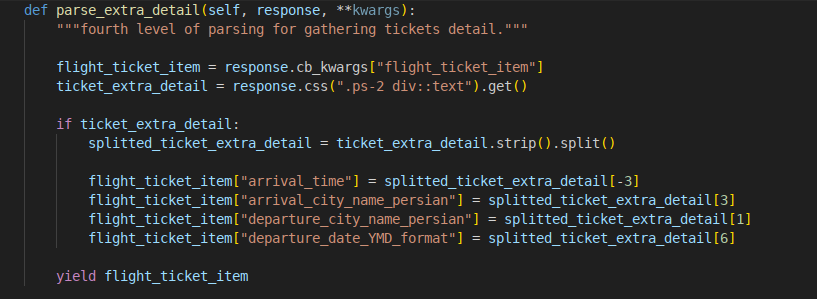
اگر شرایط فوق درست بود در نتیجه می‌توانیم درخواست HTTP را توسط کلاس Request ایجاد کرده و توسط پارامتر cb\_kwargs آیتم فعلی را برای کامل‌تر شدن آن به پارسر سطح چهارم انتقال دهیم. در غیر این صورت اگر شرایط فوق معتبر نبود آیتم فعلی را فقط توسط کلمه رزرو شده yield برای اعمالی مثل Validation ، Saving to database ، Manipulation و غیره به pipline ها انتقال می‌دهیم یا اینکه خیلی ساده آن‌ را درون فایل csv ذخیره می کنیم.

**بخش سوم:**

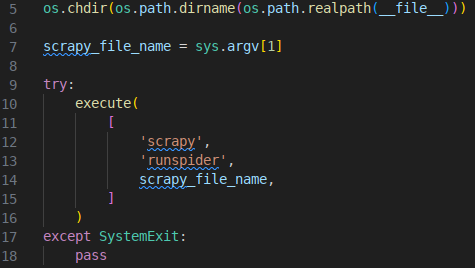
****

نکته ایی که باید آن اشاره کنم این می‌باشد که فریمورک اسکرپی به صورت **Asynchronous** کار می‌کند این بدان معنی است که وقتی اسکرپی یک درخواست HTTP را ارسال می‌کند **منتظر نمی‌ماند** که Response‌ آن به سیستم شما برگردد تا رکيوئست بعدی را ارسال کند، بلکه تمام رکیوئست های شما را **پشت سر هم** ارسال، و با وجود یک Scheduler تمام Response ها را بعداً درون callback function که شما موقع ارسال رکیوئست مشخص کردید بررسی می کند. طبق کامنتی که درون عکس نوشتیم حتماً متوجه شدید بعد از اینکه **هربار** بخش دوم تابع parse\_tickes\_table تمام شد(یعنی زمانی که تمام سطر های درون جدول **فعلی** استخراج شد) ما به بخش سوم تابع مورد بحث خود می‌رسیم باید توجه کنید که این بدان معنا نیست که کار بخش دوم **کاملاً** تمام شده بلکه رکیوئست های باقی‌مانده باز هم پشت سر هم ارسال خواهند شد.( اینجاست که مفهوم Asynchronous نمود پیدا خواهد کرد). به طور ساده‌تر شما در نظر بگیرید که در **سطح سوم کراول** خود هستید و یک جدول چند صفحه ایی دارید زمانی که تمامی داده‌های سطرهای جدول صفحه اول را استخراج کردید تازه به **بخش سوم تابع مورد بحث** خواهید رسید. از آنجایی که با توجه به ساختار سایت هدف به راحتی نمی‌توان تشخیص داد که جدول ما چند صفحه ایی می‌باشد تا بدانیم که تا کی باید parse کردن جدول را ادامه دهیم باز هم به ابزار Network مروگر خود در بخش Developer مراجعه کرده و درخواستی که برای هر صفحه ارسال می‌شود را پیدا می‌کنیم، آن آدرس دقیقاً مشابه آدرسی است که در **سطح سوم** به شما توضیح دادیم و در **بخش سوم تابع** نیز آن را مشاهده می کنید. از آنجایی که هر درخواست اسکرپی از هم مستقل بوده و جدا از هم parse می‌شوند ما یک کلید به نام page\_number در دیکشنری cb\_kwargs دقیقاً بعد از parse شدن **تمامی سطر های هر جدول** (یعنی هربار تمام شدن بخش دوم تابع مورد بحث) قرار می‌دهیم تا بتوانیم با آن آدرس URL **صفحه بعدی هر جدول** را مشخص کند. دلیل اینکه از cb\_kwargs چرا استفاده کردیم هم باید برایتان واضح باشد برای اینکه بتوانیم هر بار به صفحه بعدی هر جدول برویم باید مقدار صفحه فعلی هر جدول را داشته باشیم اینگونه مطمئن خواهیم شد که تمامی سطر های تمام صفحات هر جدول را parse خواهیم کرد. دقت شود برای اینکه آدرس URL صفحه بعدی ما کامل شود باید از مقدار city\_airline\_date\_code که از سطح دوم با دیکشنری cb\_kwargs به سطح سوم انتقال داده‌ایم هم استفاده کنیم و آن را در URL خود قرار دهیم بعد از طی کردن این مراحل حال نوبت به ایجاد درخواست با کلاس Request می‌باشد فقط حتماً توجه کنید که مقدار callback این کلاس باید با نام تابع سطح سوم یعنی parse\_tickets\_table دوباره پر شود چرا که در متن هم به آن اشاره کردیم که پارسر سطح سوم یک پارسر بازگشتی می‌باشد و در رابطه با شروط پایان این پارسر بازگشتی هم توضیحات کاملی را ارائه کردیم.

* **تابع parse\_extra\_detail**

این تابع به عنوان یک پارسر سطح چهارم اطلاعات اضافی مثل ساعت ورود هواپیما به شهر مقصد، ساعت خروج هواپیما از شهر مبدأ، تاریخ حرکت هواپیما به فرمت YMD و اسم شهر مبدأ و مقصد به فارسی را در صورت وجود المنت ticket\_extra\_detail در آیتم مربوطه مقدار دهی کرده و در نهایت آن را yield می کند.

**ماژول runner.py** :



دیباگر Visual Studio Code به طور پیش‌فرض عمل دیباگ یک کراولر را پشتیبانی نمی‌کند و ما باید به طریقی بتوانیم عمل دیباگ را برروی کراولر خود انجام دهیم این قابلیت توسط ماژول runner به کدهای ما اضافه شده و در صورت نیاز می‌توانیم به راحتی کد خود را دیباگ کنیم. زمانی که روی دکمه دیباگ vscode کلید کنید، ماژول runner از طریق فایل مخفی launch.json اجرا خواهد شد (بعدا در رابطه با چگونگی تنظیم ماژول runner در این فایل توضیحاتی خواهم داد) خط اول این ماژول از چند بخش تشکیل شده است:

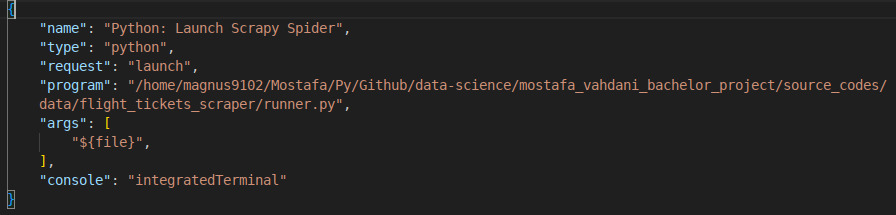
1. **os.path.realpath(\_\_file\_\_)** : آدرس مطلق **ماژول** runner را برمی گرداند.
2. **os.path.dirname** : آدرس مطلق **دایرکتوری** runner را تشخیص خواهد کرد.
3. **os.chdir** : مسیر فعلی را به مسیر دایرکتوری runner تغییر خواهد داد.

هدف از قطعه کد بالا این می‌باشد که بتوانیم از قطعه کد ”scrapy runspider {spider\_file\_name}” در پروژه اسکرپی که ایجاد کردیم استفاده کنیم چرا که این قطعه کد در جایی اجرا خواهد شد که اولاً اسکرپی در آن نصب باشد دوماً پروژه مد نظر ایجاد شده باشد.

در خط بعدی کد بالا از sys.argv[1] استفاده شده زمانی که شما فایل runner را اجرا کنید به همراه آن باید یک آرگومان هم به این فایل ارسال کنید(دقیقا مثل ارسال یک مقدار به آرگومان تابع فقط اینجا ما به ماژول داریم مقداری را ارسال می کنیم) که مقدار این آرگومان باید فایل اسپایدر مد نظر شما باشد.

در نهایت با قرار دادن یک Exception برای مواقعی که دیباگ ما تمام شد یا به هر دلیلی دیباگ ما متوقف شد و همچنین استفاده از تابع execute از فریمورک اسکرپی برای اجرای قطعه کد ”scrapy runspider {spider\_file\_name}” کار ما در نوشتن و توضیح ماژول runner تمام می شود.

Vscode برای اینکه بتواند عمل دیباگ را انجام دهد به فایل کانفیگ launch.json نیاز خواهد داشت در‌واقع اگر می‌خواهید هر نوع دیباگی انجام دهید در ابتدا باید این فایل را کانفیگ کنید. اگر میخواهیم یک اسپایدر را دیباگ کنیم می‌توانیم از کانفیگ زیر استفاده کنیم:

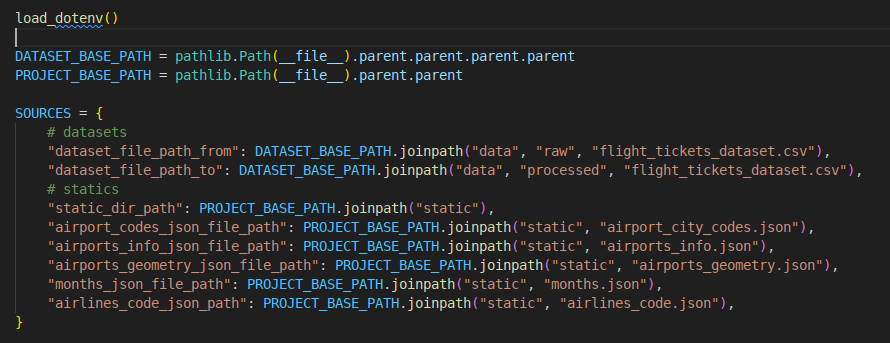
توضیح هرکدام از کلید ها به شرح زیر خواهد بود:

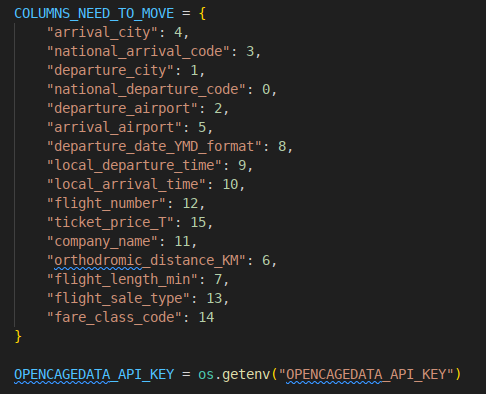
1. **name** : نرم‌افزار vscode کانفیگ شما را به اسمی که در این کلید مشخص خواهید کرد خواهد شناخت.
2. **type** : در کلید باید مشخص کنید که کد شما بر اساس چه زبان برنامه نویسی نوشته شده است.
3. **request** : خواسته شما از این کانفیگ چیست، در مورد دیباگ کردن ما میخواهیم که برنامه یا ماژول مربوطه لانچ یا همان اجرا شود. پس از مقدار launch استفاده خواهیم کرد.
4. **program** : نام برنامه یا ماژولی که می‌خواهید برای انجام عمل دیباگ اجرا شود ( بنده از آدرس دهی مطلق برای ماژول runner استفاده کردم اگر شما می‌توانید از آدرس دهی نسبی استفاده کنید بهتر است)
5. **args** : آرگومان هایی که ماژول runner انتظار دارد به آن بدیم، این آرگومان باید در یک لیست داده شوند و ترتیب آن‌ها دارای اهمیت می باشد. ”{file}$” در‌واقع یک متغیر محیطی است که vscode از آن استفاده می‌کند تا آدرس فایلی که میخواهیم دیباگ شود را مشخص کند.
6. **console** : محیطی که ما میخواهیم دیباگر ما در آن کد ها را اجرا کند و به آن مقدار **intergratedTerminal** را انتساب داده‌ایم. در صورت استفاده از این مقدار vscode‌ از ترمینال خودش برای اجرای کد های استفاده می کند.

### **فاز دوم پروژه**

در این فاز به پیش پردازش داده‌های خام و آماده سازی یک دیتاست تمیز شده خواهیم پرداخت. تا به اینجای کار در رابطه با تمام قسمت‌های زیر پروژه flight\_tickets\_scraper توضیحات کاملی را ارائه کردیم از اینجا به بعد در رابطه با زیر پروژه flight\_tickets\_preporcessing توضیح خواهیم داد.

**ماژول settings.py** :

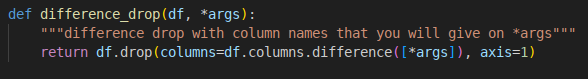




1. **تابع load\_dotenv** : تابعی برای لود کردن پارامترهای مشخص شده در فایل مخفی env درون متغیرهای محیطی.
2. **DATASET\_BASE\_PATH** : آدرس مطلق دایرکتوری data که دیتاست های ما درون آن قرار دارند.
3. **PROJECT\_BASE\_PATH** : آدرس مطلق دایرکتوری data که درون دایرکتوری source\_codes وجود دارد
4. **SOURCES** : دیکشنری ای می‌باشد که مقدار آن آدرس متصل شده DATASET\_BASE\_PATH و یا PPOJECT\_BASE\_PATH به دایرکتوری و نام فایل‌های مد نظر برای ایجاد کردن مسیر مطلق آنها می‌باشد. و کلید هم رشته ایی برای دسترسی آسان و راحت به مسیر مطلق فایل‌ها می باشد. برای متصل کردن آدرس‌ها نیز می‌توانیم از تابع joinpath استفاده کنیم.
5. **COLUMNS\_NEED\_TO\_MOVE** : بعد از ایجاد تغییرات، ستون‌های بالا ها را خواهیم داشت و می‌توانیم ترتیب قرار گرفتن این ستون‌ها را در دیتاست خود با یک عدد مشخص کنیم.
6. **OPENCAGEDATA\_API\_KEY** : مقدار کلید API که از متغیرهای محیطی توسط تابع getenv خوانده شده است.

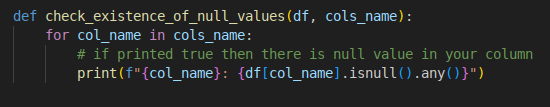
**ماژول utils.py :**

* **تابع difference\_drop**

****

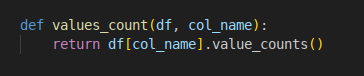
این تابع به ما کمک می‌کند تا **هر ستونی به غیر از ستون هایی** که در ورودی این تابع به عنوان args می‌دهیم از دیتاست مربوطه **پاک کنیم**. تابع difference لیستی از نام ستون‌ها را از شما خواهد گرفت و هر ستونی غیر از آن ستون هایی بهش دادین را به شما برمیگرداند در نتیجه با برگرداندن خروجی تابع difference و دادن این خروجی به تابع drop هر ستونی به غیر ستون هایی که به تابع difference\_drop داده‌ایم را می‌توانیم از دیتاست حذف کنیم.

* **تابع check\_existence\_of\_null\_values**



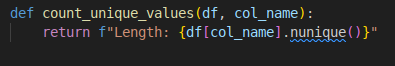
در این تابع به ازای لیستی از نام ستون‌ها می‌توانیم بررسی کنیم که آیا مقدار null در آن ستون وجود دارد یا خیر. تابع isnull به ازای هر سطر مقدار True برای اعلام وجود null و مقدار False را برای اعلام عدم وجود null بر می گرداند و تابع any در صورتی که حتی یک سطر را به عنوان True شناسایی کند مقدار True را برای ما چاپ خواهد کرد که در نتیجه این به معنی وجود مقدار null در ستون مد نظر می باشد.

* **تابع values\_count**



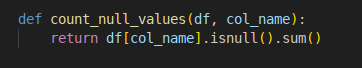
وظیفه شمارش تعداد **هرکدام** از مقادیر Categorical در ستون مد نظر را بر عهده دارد.

* **تابع count\_unique\_values**



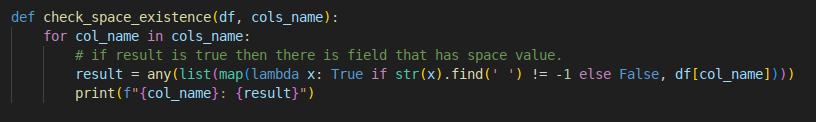
وظیفه شمارش تعداد سطر های **یکتا** موجود در ستون مد نظر را بر عهده دارد.

* **تابع count\_null\_values**



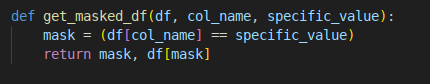
تعداد مقادیر null موجود در ستون مد نظر را شمارش می کند.

* **تابع check\_space\_existence**



در این تابع به ازای لیستی از نام ستون‌ها می‌توانیم بررسی کنیم که آیا space در آن ستون وجود دارد یا خیر. تابع find به ازای هر سطر مقدار True برای اعلام وجود space و مقدار False را برای اعلام عدم وجود space بر می گرداند و تابع any در صورتی که حتی یک سطر را به عنوان True شناسایی کند مقدار True را برای ما چاپ خواهد کرد که در نتیجه این به معنی وجود space در ستون مد نظر می باشد. تابع map در‌واقع عمل نگاشت تابع خطی lambda را بروی Iterable مد نظر (df[col\_name]) انجام می‌دهد و به ما یک generator بر میگرداند که با تابع list خروجی map را به تابع any می‌دهیم تا بررسی حضور و عدم space را انجام دهد.

* **تابع get\_masked\_df**

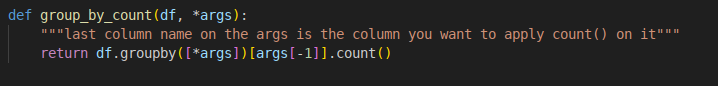


ماسک کردن، فرآیند پنهان کردن مقادیر یا ردیف های خاص از یک چارچوب داده بر اساس معیارهای خاص است. خروجی این تابع یک دیتافریم با شرایط ماسک کردن فوق(بررسی برابری با مقداری به خصوص) و خود ماسک مد نظر می باشد.

* **تابع apply\_to\_df\_by\_mask**

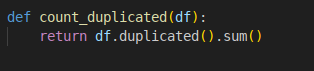
با این تابع می‌توانیم توسط دیتاست ماسک شده، دیتاست اصلی و با استفاده از یک تابع خارجی (your\_func) روی ستون مد نظر تغییرات دلخواهمان را اعمال کنیم. به عبارت ساده‌تر اگر بخواهیم روی مقادیر سطرهای خاصی از یک ستون دیتافریم تغییری به وجود آوریم از این تابع استفاده می کنیم. تابع loc برای انتخاب داده‌ها جهت انتساب مقدار جدید از یک DataFrame بر اساسlabel ها یا شرایط Boolean استفاده می کند. تابع apply نیز برای اعمال کردن یک تابع خارجی روی محور ستونی یا سطری (axis) استفاده می شود. تا به الان باید متوجه شده باشید که از تابع فوق باید در کنار تابع get\_masked\_df استفاده کنیم.

* **تابع group\_by\_count**



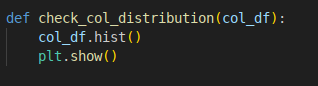
این تابع لیستی از نام ستون‌ها را از شما خواهد گرفت و در خروجی بر اساس ترتیبی که در لیست ستون‌ها مشخص کردید بعد از اعمال گروه بندی مد نظرتان، برای شما عمل شمارش را روی آخرین عضو لیست ستون‌ها انجام خواهد داد.

* **تابع count\_duplicated**



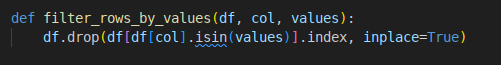
این تابع برای شما سطرهای تکراری را در دیتافریم شمارش خواهد کرد. در‌واقع تابع duplicated هر سطری که مقدار تکراری آن را پیدا کند یا برابر با True قرار خواهد داد و تابع sum تعداد True ها را برای شما شمارش خواهد کرد.

* **تابع check\_col\_distribution**



نوع توزیع داده‌های ستون مد نظر را با نمودار هیستوگرام به شما نمایش خواهد داد. توزیعی که در خروجی مشاهده می‌کنید می‌تواند normal ، right skewed ، left skewed ،‌ symmetric و غیره باشد.

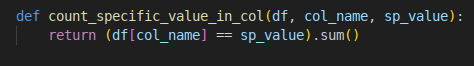
* **تابع filter\_rows\_by\_values**



این تابع وظیفه فیلتر کردن سطرهایی با مقادیر خاص روی ستون مد نظر را برعهده دارد. اجزای این تابع در زیر به ترتیب شرح داده شده اند:

1. **df[col].isin(values)** : توسط تابع isin بررسی می‌شود که به ازای سطرهایی که یکی از مقادیر خاص ما (values) در آن وجود داشته باشد مقدار True برای آن سطر در نظر گرفته می شود.
2. **df[df[col].isin(values)].index** : بعد از پیدا کردن سطر های مد نظر، اندیس آن ها توسط index برگشت داده خواهد شد.
3. **تابع drop** : ایندکس های سطر های مد نظر بعد از قرار گرفتن در این تابع حذف خواهند شد.
4. **آرگومان inplace** : این آرگومان اگر True باشد به ما این اجازه را خواهد داد تا بدون نیاز به عمل انتساب دیتافریم خود را آپدیت کنیم.

* **تابع count\_specific\_value\_in\_col**

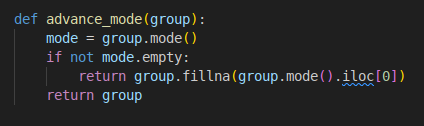


با استفاده از این تابع می توانیم یک مقدار خاص را درون ستون مد نظر شمارش کنیم. قطعه کد زیر را در نظر بگیرید:

* **(df[col\_name] == sp\_value)**

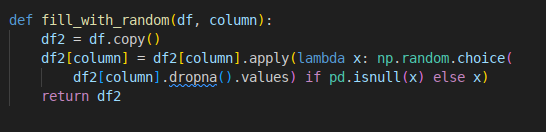
با این قطعه کد هر جایی در ستون مد نظر مقداری برابر با مقدار مشخص شده (sp\_value) بشود مقدار True برایش در نظر گرفته شده و سپس توسط تابع sum این مقادیر True شمارش خواهند شد.

* **تابع advance\_mode**



این تابع برای پر کردن مقادیر گم شده (missing values) Categorical یک ستون گروه بندی شده توسط تابع mode استفاده می‌شود حال به چه علت از تابع mode به تنهایی استفاده نکردیم، همانطور که از تصویر فوق مشخص است تابعmode در جایی که **تعدا فراوانی های یک مقدار Categorical برابر با None باشد** مقدار None را به ما بر خواهد گرداند در نتیجه با ایجاد یک شرط None بودن مقدار mode را روی گروه مد نظر مشخص می‌کنیم اگر None بود که همان گروه را برمی‌گردانیم ولی اگر مقدار mode برایمان None نبود تمامی سطرهای None آن گروه را برابر با مقدار mode قرار می دهیم. گفتنی که خروجی تابع mode یک دیتافریم می‌باشد پس برای دسترسی به مقدار mode می‌توانیم از تابع iloc با ایندس صفر استفاده کنیم.

* **تابع fill\_with\_random**



این تابع برای پر کردن مقادیر گم شده Categorical استفاده می‌شود و از انتخاب تصادفی روی خود مقادیر موجود درون ستون مد نظر عمل می کند. کد بالا را به ترتیب زیر شرح می دهیم:

1. **تابع copy** : یک کپی از دیتافریم به ما می‌دهد و آنرا درون متغیر df2 قرار می‌دهیم این کار انجام دادیم چون جلوتر نمی خواستیم تغییری روی دیتافریم اصلی ما اتفاق بی افتد.
2. **تابع dropna** : این تابع هرجا که در ستون مد نظر مقدار null داشته باشد برای ما حذف می کند.
3. **.values** : این attribute تمام مقادیر ستون مد نظر را در قالب یک آرایه به بر خواهد گرداند. توجه شود که استفاده از این attribute بعد از حذف کردن مقادیر null درون ستون اتفادق افتاده.
4. **تابع isnull** : بررسی می‌کند که آیا مقدار x یک مقدار null می‌باشد یا خیر.
5. **تابع apply** : اعمال کردن یک تابع برای ایجاد تغییر برروی مقادیر ستون مد نظر.
6. **تابع choice** : وظیفه انتخاب تصادفی از اعضای یک آرایه را بر عهده دارد.

طبق توضیحات بالا باید متوجه شده باشید که روی ستون مدنظرتان تابع apply اعمال می‌شود و تابع apply نیز درون خودش دارای یک تابع خطی برای اعمال تغییرات می‌باشد. این تابع خطی هر بار مقدار هر فیلد درون ستون را برمی‌دارد و در متغیر x قرار می‌دهد حال توسط تابع isnull مقدار x بررسی می‌شود اگر این مقدار برابر با مقدار null نبود خود x برای ما برگردانده می‌شود چرا که نمی‌خواهیم جایی از ستون که مقدار دارد را تغییر دهیم ما صرفاً می‌خواهیم مقادیر گم شده را پر کنیم در ادامه اگر مقدار x برابر با null بود از دیتافریتم df2 ستون مد نظر را انتخاب کرده با تابع dropna مقادیر null حذف می‌کنیم و با attribute به نام values یک آرایه از تمامی مقدار باقی‌مانده در ستون ایجاد می‌کنیم و سپس توسط تابع choice یک انتخاب تصادفی انجام می‌دهیم و مقدار بدست آمده را برمی‌گردانیم تا در مکان null مد نظر در ستون قرار گیرد.

**ماژول airports\_geographic\_coordinate.py :**

* **تابع get\_airport\_geographic\_coordinate**

کدهای مربوط به این تابع را در چند بخش توضیح خواهیم داد:

**بخش اول:**

****

این تابع مختص به API سایت OpenCageGeocode نوشته شده است و وظیفه آن بدست آوردن مختصات جغرافیایی فردوگاه ها می باشد. این تابع به صورت بازگشتی نوشته شده است و شرط پایان آن توسط یک متغیر سراسری به نام max\_recursive\_depth مشخص می‌شود، مقدار اولیه این متغیر صفر می‌باشد و زمانی که مقدار این متغیر برابر با مقدار لیترال ۲ بشود مقدار None برگشت داده شده و کار تابع بازگشتی پایان می یابد. دلیل اینکه خواستیم عمق بازگشتی تابع فوق برابر با ۲ باشد این می‌باشد که ما یکبار می‌خواهیم با کد فرودگاه به دنبال مختصات جغرافیایی آن بگردیم و اگر با کد فرودگاه نتوانستیم مختصات را پیدا کنیم می‌خواهیم که با اسم فرودگاه دوباره درخواست به API بدهیم تا مختصات جغرافیایی را به ما باز گرداند. از کد بالا مشخص است که متغیر هایی مثل q ، country\_code و airport\_name حتماً باید از طریق kwargs مقدار دهی شود اگر این کلید ها در دیکشنری kwargs وجود نداشته باشد آن وقت برنامه با اررور KeyError مواجه خواهد شد مابقی متغیرهای محلی همانطور که در کد مشاهده می‌کنید توسط تابع get مقدار دهی پیش‌فرض خواهند شد. در ادامه نیز از آبجکت geo\_coder متد geocode که ایجاد و ارسال درخواست به سمت API فرا می‌خوانیم و پارامترهای لازمه را توسط متغیرهای محلی مقدار دهی می‌کنیم. البته طبق داکیومنت باید گفت که سایت OpenCage پارامترهای بیشتری را برای ارسال رکیوئست قبول می‌کند ولی ما برای دستیابی به مختصات جغرافیای فرودگاه مد نظر صرفاً به پارامترهای تهیه شده نیاز خواهیم داشت پارامترها را به ترتیب زیر شرح می دهیم:

1. query : کوئری که نیاز داریم باید توسط این پارامتر ارسال کنیم در مورد مسأله ایی که ما باهاش درگیر بوده‌ایم کوئری ما می‌تواند کد فرودگاه یا اسم فردوگاه مد نظر باشد.
2. countrycode : کدی ۲ حرفی IATA که یک کشور را با آن شناسایی خواهند کرد.
3. language : با این پارامتر می‌توان مشخص کرد که response به چه زبانی به شما ارسال شود.
4. pretty : اگر مقدار آن برابر با ۱ باشد خروجی را به شکلی نمایش خواهد داد که خواندن و دیباگ کردن راحت‌تر باشد.
5. no\_dedupe : اگر مقدار این پارامتر برابر با ۱ باشد از ارسال مقادیر تکراری در response جلوگیری خواهد شد.
6. no\_annotation : اگر مقدار این پارامتر برابر با ۱ باشد از ارسال اطلاعات اضافی جلوگیری خواهد شد.

شاید متوجه شده باشید ما نیازی به مورد ۵ و ۶ نخواهیم داشت دلیل استفاده کردن از این موارد صرفاً به خاطر کمتر از سربار به روی سرور می باشد(هرچند که ما درخواست زیادی به سمت سرور مربوطه ارسال نخواهیم کرد ولی استفاده از این پارامتر ها خالی از لطف نیست)

در ادامه از تابع sleep در کتابخانه time استفاده کرده‌ایم و مقدار آن را برابر با ۱ ثانیه قرار داده‌ایم. دلیل اینکه از ۱ ثانیه تأخیر استفاده کرده‌ایم به گفته داکیومنت می‌باشد چرا که در داکیومنت بیان شده برای API هایی که از اکانت رایگان استفاده می‌کنند درخواست های خود را باید با تأخیر ۱ ثانیه ایی ارسال کنند. در نهایت نیز response خود را درم متغیر result\_lst برای ادامه کار ذخیره می کنیم.

**بخش دوم:**



گفتنی که result\_lst همانطور که از اسمش مشخص است ساختمان داده ایی از نوع لیست پایتونی می‌باشد که اعضای آن دیکشنری می باشد.

حال به ازای دیکشنری هایی که درون این لیست قرار دارند که حلقه با for ایجاد می‌کنیم و متغیرهای component\_dict ، \_type و \_category را توسط مقادیر کلیدهایی که مشاهده می‌کنید مقدار دهی خواهیم کرد. طبق گفته داکیومنت مقدار \_category می‌تواند مقادیری مثل transportation ، road ، commerce ، social ، place و غیره باشد. در این API فردوگاه در دسته trasportation قرار دارند پس منطقی است که در چندین مکانی که API بابت کوئری که زدیم به ما پیشنهاد داده به دنبال این مقدار در \_category باشیم علاوه بر اینکه مقدار \_type دقیقاً مشخص می‌کند در دسته transportation ما به دنبال حمل و نقل هوایی هستیم که مقدار آن با aeroway مشخص می‌شود. در نتیجه فقط با این دو مقدار می‌توانیم محدوده بررسی را کوچک کرده و فردوگاه مد نظر را از بین چند مکانی که API پیشنهاد می‌دهد پیدا کنیم در نهایت هم با معتبر بودن شرط طبق مطالبی که توضیح دادیم از کلیدهای مرتبطه دیکشنری مقدار طول و عرض جغرافیایی را پیدا می‌کنیم و با توجه به ساختار فوق درون تاپل result\_tpl قرار می‌دهیم و مقدار آنرا به خارج از تابع باز می گردانیم. اگر بعد از تمام شدن حلقه for همچنان مقداری که به دنبالشان بودیم را پیدا نکردیم به مقدار max\_recursive\_depth یکی اضافه کرده و تابع فوق را با مقادیری که مشاهده می‌کنید دوباره فرا می‌خوانیم و همین فرایند را دوباره از سر می‌گیریم فقط با این تفاوت که query ما این بار نام فرودگاه می باشد.

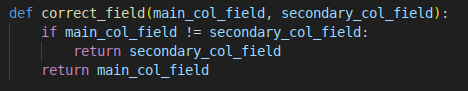
**طریقه استفاده از تابع get\_airport\_geographic\_coordinate** :

****

در اولین خط یک **متغیر خاص** پایتونی به نام \_\_name\_\_ را مشاهده می‌کنید این متغیر زمانی که مستقیم ماژول پایتونی را اجرا کنید مقدار آن با ” \_\_main\_\_” برابر خواهد بود ولی زمانی که ماژول پایتون را در ماژول دیگر import کنید مقدارش برابر نام آن ماژول خواهد بود. پس طبق توضیحات فقط در صورتی که ماژول airports\_geographic\_coordinate.py را مستقیماً اجرا کنیم شرطی که در خط اول عکس قرار دارد معتبر خواهد بود. یک متغیر به نام airports\_geometry\_dict ایجاد کردیم که این متغیر مقادیر مرتبط با مختصات جغرافیایی هر فرودگاه را درون خودش جای خواهد داد. مسیر فایل airports\_info.json را از طریق کلید خودش در دیکشنری SOURCES بدست می آوریم و توسط تابع get\_json\_obj متحوای آنرا میخوانیم و درون متغیر airports\_info\_dict قرار می دهیم. در ادامه یک آبجکت با کلاس OpenCageGeoCode با کلید API ایی که در setting.py قرارداد ایجاد می‌کنیم تا بتوانیم با اکانت خود از API مربوطه استفاده کنیم. سپس توسط متد values به ازای تمام مقادیر درون کلیدهای دیکشنری airports\_info\_dict یک حلقه ایجاد می‌کنیم و تابع get\_airport\_geographic\_coordinate را با مقداردهی به پارامترهای مورد نیازش صدا می‌زنیم و خروجی اش را در متغیر airport\_geometry\_tpl نگهداری می‌کنیم. اگر مقدار این متغیر برابر با None باشد یعنی ما نتوانسته‌ایم نه با کد فرودگاه و نه با اسم فردوگاه مختصات جغرافیایی آنرا پیدا کنیم پس شرط مربوطه معتبر نخواهد اما درغیر این صورت اگر مقدار این متغیر برابر None نبود مقدار key و value را جدا می‌کنیم و درون دیکشنری airports\_geometry\_dict قرار می‌دهیم باید دقت شود که مقدار key طبق متغیر airport\_geometry\_tpl برابر با کد فرودگاه و مقدار value برابر با یک دیکشنری می‌باشد که خود این دیکشنری شامل نام فرودگاه و مختصات جغرافیایی آن می باشد. بعد از اینکه کار حلقه ما تمام شد تعداد مقادیر پیدا شده درون متغیر airports\_geometry\_dict را با تابع len بدست می آوریم و با کلید ”Total Length” دوباره درون دیکشنری airports\_geometry\_dict قرار می دهیم. در نهایت هم محتوای این متغیر را در فایل json مد نظر ذخیره سازی می‌کنیم تا بعد از آن استفاده لازمه را برای تمیز کردن دیتاست خام خود ببریم.

**ماژول tickets\_preprocessing.py** :

* **تابع correct\_field**

****

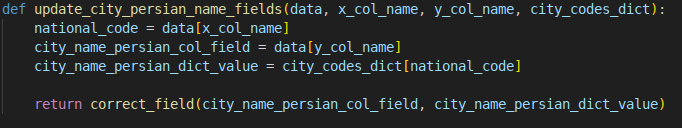
در زمانی که می‌خواهیم برابری دو فیلد متفاوت از هم را بسنجیم و در صورت برابر نبود آن‌ها مقدار فیلد دوم و در صورت برابر بودن آن‌ها فیلد اول (همان فیلدی است که درحال حاضر در ستون مد نظر مد ما وجود دارد) برگشت داده شود و در دیتاست بنشیند.

* **تابع semi\_space\_correction**



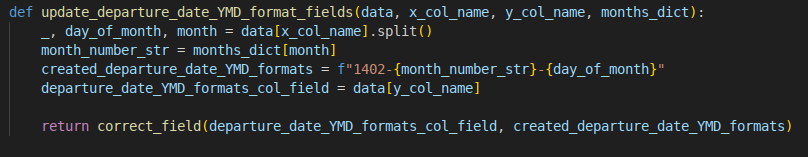
بر خلاف زبان انگلیسی در زبان فارسی کلمات مرکب و افعالی وجود دارند که برای خواناتر و زیبا‌تر دیده شدن آن‌ها در متن باید از یک نیم فاصله (semi-space یا half-space) به جای فاصله کامل (space) برای آن‌ها استفاده کرد چرا که باید این نوع کلمات در واحد یک کلمه توسط مخاطب دیده شوند تا متن مورد نظر برای آن‌ها خواناتر باشد. به طور مثال در متون فارسی ”آن ها” با ” آن‌ها” و یا «”سه شنبه” با ”سه‌شنبه” متفاوت است چرا که در اولی از فاصله کامل و در دومی از نیم فاصله استفاده شده است. به طور خلاصه و کلی نیم فاصله در جایی از متن به کار می‌رود که می‌خواهیم دوکلمه از هم جدا باشند ولی چسبیده دیده شوند و بین آن‌ها فاصله نباشد. این امر موجب خواهد شد که بعداً در کد نویسی آن ترکیب کلمه ایی مد نظر در واحد یک کلمه در دسترس قرار گیرد. در ساختار Unicode Encoding مقدار نیم فاصله با **u200c\** دیده خواهد شد. در‌واقع به دید شما در متن چنین چیزی بین ترکیب کلمه ایی مد نظر وجود ندارد ولی حتماً در ساختار Unicode آنرا مشاهده خواهید کرد. تابع فوق به کمک آبجکت normalizer از کلاس Normalizer در کتابخانه hazm میتوانید در مقدار x ، فاصله را به نیم فاصله تبدیل کند.

* **تابع update\_city\_persian\_name\_fields**

****

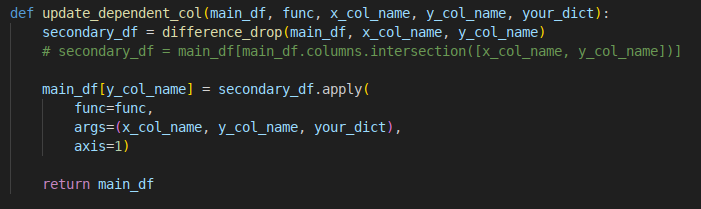
هدف ما از نوشتن این تابع این می‌باشد که به ما کمک کند تا فیلدهای درون ستون city\_persian\_name دیتاست خام خود را در صورت null بودن فیلد مد نظر پر و آپدیت کنیم. توجه شود که آرگومان data یک دیتافریم تهیه شده از روی دیتاست خام می‌باشد و مقدار x\_col\_name و y\_col\_name نام ستون هایی می‌باشند که علاوه بر اینکه حتماً باید در دیتافریم وجود داشه باشند همچنین باید مقدار y\_col\_name بر اساس x\_col\_name تعریف شده باشد چرا که هدف این تابع این می‌باشد که از مقدار x\_col\_name و y\_col\_name به کمک دیکشنری city\_codes\_dict به نام فارسی فرودگاه مد نظر دست یابد و همانگونه که در تصویر مشاهده می‌کنید data[x\_col\_name] مقدار national\_code را به ما می‌دهد (این متغیر می‌تواند برحسب مکان استفاده از این تابع شناسه فرودگاه مقصد یا مبدأ باشد) و همچنین data[y\_col\_name] مقدار فارسی نام فردوگاه می باشد(این متغیر نیز بر حسب مکان استفاده از این تابع می‌تواند نام فارسی فرودگاه مقصد یا مبدأ باشد.) و در ادامه نام فارسی فردوگاه مد نظر در دیکشنری city\_codes\_dict را توسط کلید national\_code استخراج کرده و در نهایت نیز توسط تابع correct\_field که قبلاً آنرا تعریف کردیم می‌توانیم در مکان هایی که مقدار null وجود دارد تغییر مربوطه را ایجاد کنیم.

* **تابع update\_departure\_date\_YMD\_format\_fields**

****

هدف ما از نوشتن این تابع این می‌باشد که به ما کمک کند تا فیلدهای درون ستون departure\_date\_YMD\_formats دیتاست خام خود را در صورت null بودن فیلد مد نظر پر و آپدیت کنیم. توجه شود که آرگومان data یک دیتافریم تهیه شده از روی دیتاست خام می‌باشد و مقدار x\_col\_name و y\_col\_name نام ستون هایی می‌باشند که علاوه بر اینکه حتماً باید در دیتافریم وجود داشه باشند همچنین باید مقدار y\_col\_name بر اساس x\_col\_name تعریف شده باشد چرا که هدف این تابع این می‌باشد که از مقدار x\_col\_name و y\_col\_name به کمک دیکشنری months\_dict به تاریخ حرکت هواپیما دست یابد و همانگونه که در تصویر مشاهده می‌کنید data[x\_col\_name] مقدار ستون departure\_date موجود در دیتاست خام را در سه متغیر که جدا از هم می باشند به ما می‌دهد. و همچنین data[y\_col\_name] مقدار فعلی فیلد departure\_date\_YMD\_formats را به ما خواهد داد. و در ادامه شماره ماه مربوطه را از دیکشنری months\_dict توسط کلید month استخراج کرده و در نهایت نیز توسط تابع correct\_field که قبلاً آنرا تعریف کردیم می‌توانیم در مکان هایی که مقدار null وجود دارد تغییر مربوطه را ایجاد کنیم.

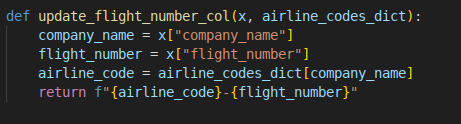
* **تابع update\_dependent\_col**

****

از طریق این تابع می‌توانیم فیلدهای ستون y را به کمک مقدارهای ستون x و دیکشنری مد نظر شما آپدیت کنیم. توجه شود که ستون y باید و حتماً به ستون x وابستگی داشته باشد. در کد خود از ترکیب این تابع با توابعی مثل update\_departure\_date\_YMD\_format\_fields و update\_city\_persian\_name\_fields می‌توانیم بهره ببریم.

در رابطه با بدنه تابع فوق می‌توان اضحار کرد که از توابعی مثل difference\_drop و apply استفاده شده که در رابطه با این توابع قبلاً توضیحاتی را خدمتتان ارائه کردیم تنها مورد جدیدی که در این قسمت وجود دارد این است که در تابع apply علاوه بر آرگومان func از آرگومان های args و axis هم استفاده شده که به ترتیت آرگومان args برای انتقال پارامترهای مورد نیاز تابع func و آرگومان axis=۱ برای ارسال مقادیر ستونی روی هر سطر(ارسال هر سطر دیتافریم) به تابع func استفاده شده است.

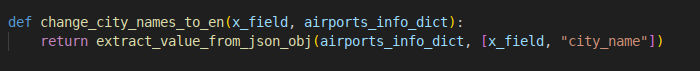
* **تابع update\_flight\_number\_col**

****

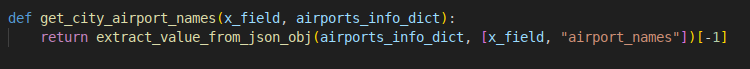
کد پرواز ترکیبی از حروف و اعداد می باشد که با در دست داشتن آن می‌توانیم به اطلاعاتی دیگر مثل ساعت حرکت از مبدا، ساعت ورود به مقصد، کلاس پروازی بلیت، فرودگاه مبدأ و مقصد، تعداد توقف های هواپیما در بین مسیر و غیره و غیره به راحتی دسترسی داشته باشیم. ما در دیتاست خام خود به صورت پیش‌فرض چنین فقط بخش شماره ایی کد پرواز را در دست داریم اما با استفاده از فایل airlines\_codes.json که با توجه به شرکت های هواپیمایی موجود در دیتاست خود فراهم کرده‌ایم می‌توانیم فرمت این کد پروازی را ایجاد کرده به صورتی که در عکس فوق مشادهده می‌کنید می باشد. در‌واقع به طور کلی هر شرکت هواپیمایی می‌تواند ساختار کد پروازی خودش را داشته باشد اما با مقداری سرچ متوجه خواهیم شد که اکثریت شرکت ها از استاندارد زیر استفاده می کنند:

**"{airline\_code}-{flight\_number}"**

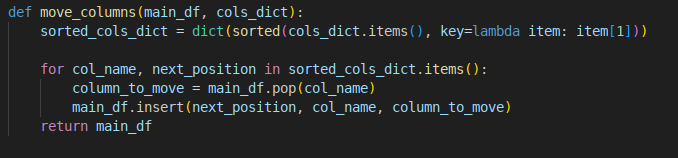
* **تابع change\_city\_names\_to\_en**

این تابع به کمک دیکشنری airports\_info\_dict و از طریق تابع extract\_value\_form\_json\_obj معادل انگلیسی اسم فارسی فرودگاه بدست می‌آورد و آن را برمی گرداند. هر چند باید خاطر نشان کرد که در دیکشنری airports\_info\_dict در ابتدا به دنبال کلیدی با مقدار x\_field خواهیم گشت و پس آن در دیکشنری بدست آمده مقدار کلید city\_name را بر می گردانیم. پس توجه شود ترتیبی که در لیست کلیدها ایجاد می‌کنید دارای اهمیت در پیدا کردن مقدار نهایی می باشد.

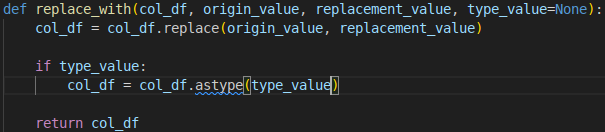
* **تابع get\_city\_airport\_names**

دقیقاً به همانند تابع changes\_city\_name\_to\_en عمل می‌کند عمل می‌کند با این تفاوت که خروجی تابع extract\_value\_from\_json\_obj با توجه به ترتیب کلیدهای استفاده شده براساس ساختار فایل airports\_info.json یک لیست از نام فرودگاه ها می‌باشد که در هر لیست فقط آخرین عضو این لیست را بر می گردانیم.

* **تابع move\_columns**

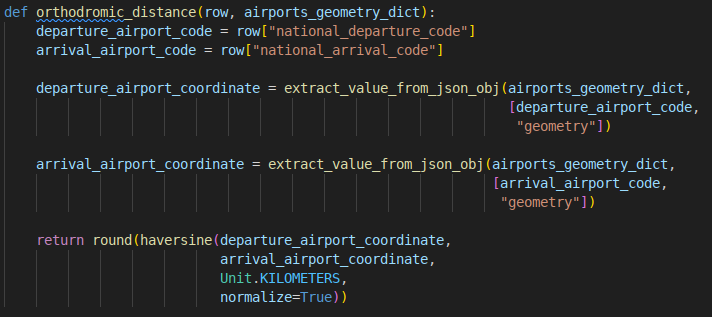
همانطور که از اسم این تابع پیداست وظیفه مرتب سازی ستون‌های دیتاست را بر اساس ترتیبی که در متغیر COLUMNS\_NEED\_TO\_MOVE فراهم شده بر عهده دارد. برای انجام این عمل در ابتدا باید تمامی مقادیر موجود در دیکشنری COLUMNS\_NEED\_TO\_MOVE را بر اساس value آن‌ها مرتب سازی کنیم و سپس به ازای جفت Item های key و value موجود در این دیکشنری یک حلقه ایجاد کرده و در ابتدا ستون مربوطه را از دیتافریم مد نظر را توسط تابع pop حذف کرده و درون متغیر column\_to\_move قرار می‌دهیم و سپس با استفاده از تابع insert در موقعیت مد نظر درون دیتافریم جای خواهیم داد و بعد از اتمام حلقه دیتافریم آپدیت شده را به عنوان باز می گردانیم.

* **تابع replace\_with**

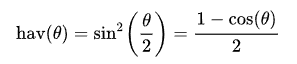
****

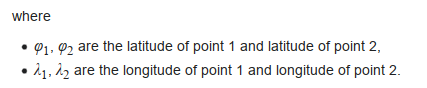
این تابع وظیفه جا به جایی مقدار اصلی که فیلد با مقدار جایگزین آن را بر عهده دارد و همچنین با پارامتر type\_value می‌تواند نوع یک ستون را توسط تابع astype عوض کند به طور مثال اگر یک ستون از نوع int64 بود می‌توان آن را با این پارامتر به float تبدیل کرد.

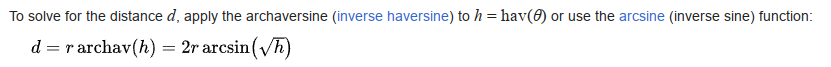
* **تابع orthodromic\_distance**

این تابع وظیفه تخمین فاصله دایره-بزرگ (فاصله مستقیم یا فاصله هوایی) بین فرودگاه مبدأ و مقصد را بر عهده دارد. آرگومان row یک سطر به صورت دیتافریم می‌باشد که توسط آن می‌توان مقدار ستون national\_departure\_code و national\_arrival\_code سطر مد نظر را بدست آورد. توسط این شناسه ها هم می‌توان از دیکشنری airports\_geometry\_dict مختصات جغرافیای فرودگاه مبدأ و مقصدا را با تابع extrac\_value\_from\_json\_obj و کلیدهای ترتیبی airport\_code و geometry بدست آورد. در نهایت نیز با فرمول haversine زیر فاصله دایره-بزرگ بین مبدأ و مقصد محاسبه کرد:



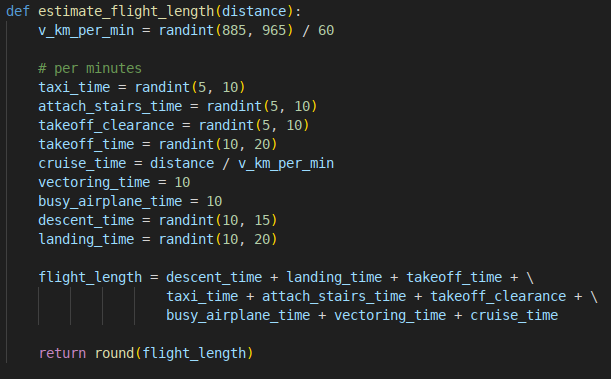
****

****



می‌توانیم خودمان این فرمول را با استفاده از توابع موجود در کتابخانه math پیاده‌سازی کرده و فاصله را محاسبه کنیم اما به جای اختراع کردن دوباره چرخ پکیج haversine را در محیط مجازی ای که ایجاد کردیم نصب می‌کنیم و از تابع havesine که در این کتابخانه موجود است برای محاسبه فاصله دایره-بزرگ استفاده می کنیم. این تابع همانطور که می‌بینید فقط از ما طول و عرض جغرافیایی فرودگاه و همچنین یکا محاسباتی (کیلومتر) را از ما می‌گیرد و همچنین اگر مقدار پارامتر normalize برابر با True باشد در صورتی که طول و عرض جغرافیایی شما خارج از محدوده بود آن را در محدوده مجاز نرمال می کند. و سپس به محاسبه فاصله haversine می پردازد. در نهایت زیر با استفاده از تابع round خروجی بدست آمده را به یک عدد صحیح رند می کنیم.

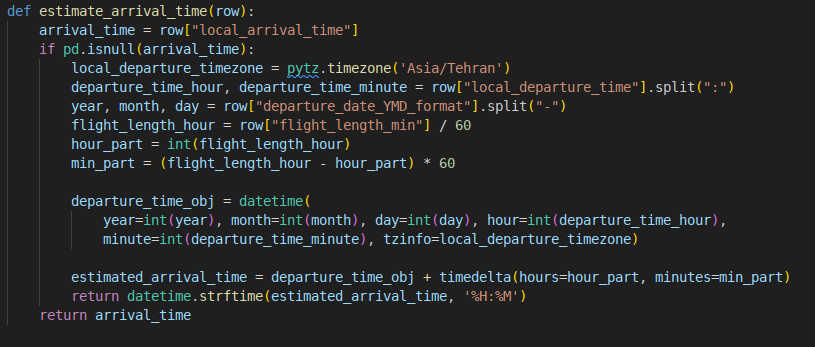
* **تابع estimate\_flight\_length**

از این تابع برای محاسبه تخمینی مدت زمان پرواز از مبدأ به مقصد می‌توان استفاده کرد همانطور که مشاهده می‌کنیم در ورودی این تابع از آرگومان distance استفاده کرده‌ایم این آرگومان مقدار فاصله هوایی یا همان فاصله دایره-بزرگ بین مبدأ و مقصد می‌باشد احتمالاً متوجه شده‌اید که بعد از اینکه از تابع orthodromic\_distance استفاده کردیم می‌توانیم از تابع فوق برای محاسبه مدت زمان پرواز بهره ببریم. شاید به این فکر بی افتید که خب صرفاً تقسیم کردن مقدار فاصله به روی میانگین سرعت هواپیماهای مسافربری می‌تواند خروجی مد نظر ما را تولید کند ولی باید بگویم که سخت در اشتباهید چرا که مدت زمان پرواز صرفاً برابر با مقدار تقسیم فاصله به میانگین سرعت نیست و پارامتر های دیگری هم در مدت زمان پرواز دخیل می‌باشند که هرکدام از این عوامل به شرح زیر هستند:

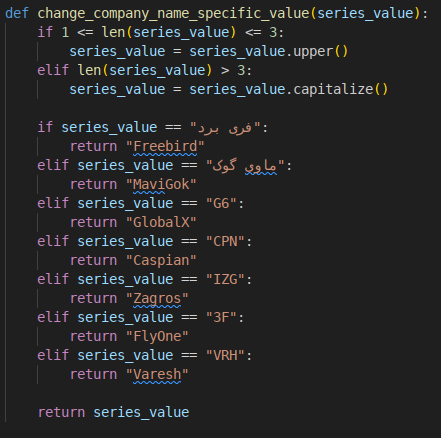
1. **v\_km\_per\_min** : میزان میانگین سرعت هواپیما مسافربری (کیلومتر بر دقیقه)
2. **taxi\_time** : میزان زمان ترک کردن سوله / ترمینال تا بلند شدن هواپیما از زمین
3. **attach\_stairs\_time** : میزان زمان وصل کردن پله ها به هواپیما برای سوار شدن و پیاده شدن مسافران
4. **takeoff\_clearance** : میزان زمانی که طول می‌کشد تا اجازه پرواز از برج مراقبت به خلبان هواپیما داده شود.
5. **takeoff\_time** : میزان زمان بلند شدن هواپیما از زمین تا رسیدن به ارتفاع کروز.
6. **cruise\_time** : از زمانی که در مبدأ هواپیما به ارتفاع کروز خودش رسید تا زمانی که برای فرود به هر دلیلی (رسیدن به مقصد، توقف اجباری، سقوط و غیره) ارتفاع خودش را از ارتفاع کروز کاهش دهد می گویند. در بعضی جاها به این زمان، flight duration یا flight length نیز گفته می شود. در‌واقع وقتی شما فاصله دایره-بزرگ را تقسیم بر میانگین سرعت هواپیما مسافربری کنید مثل این می‌ماند که دارید cruise time را محاسبه می کنید.
7. **vectoring\_time** : در نظر بگیرید که فاصله ایی که شما بدست آوردید فاصله مستقیم بین مبدأ و مقصد می‌باشد اما مسیر هواپیما هیچ وقت به صورت مستقیم نیست. بلکه می‌توان گفت مسیر هواپیما با تغییر مسیرهایی جزئی در **امتداد** خط مستقیم بین مبدأ و مقصد می باشد و خلبان هواپیما همیشه در بین مسیر به دلایل متفاوت در حین پرواز، تغییر زاویه ایی را در مسیر خود ایجاد خواهد کرد.
8. **busy\_airplane\_time** : میزان زمانی که هواپیما باید در آسمان معطل بماند تا خطوط هوایی برای فرود خلوت شود. در نظر بگیرید که هواپیما به محض رسیدن به مقصد عملیات کاهش ارتفاع خود آغاز نمی‌کند بلکه در صورت لزوم باید صبر کند تا شرایط برای فرود سالم مهیا شود.
9. **descent\_time** : مقدار زمانی که طول می‌کشد تا هواپیما از ارتفاع کروز به ارتفاع فرود برسد.
10. **landing\_time** : میزان زمانی که طول می‌کشد تا هواپیما از ارتفاع فرود به سطح زمین برسد و توقف کامل در فرودگاه مقصد داشته باشد.

تمام پارامترهایی که در بالا ذکر شده‌اند بر حسب دقیقه هستند و مجموع تمام پارامترهای بالا نیز مسلماً به دقیقه می باشند. تمام بازه های زمانی ای که در عکس فوق در تابع randint استفاده شده میانگین مقادیری می‌باشند که در سطح اینترنت گزارش شده‌اند. برای اینکه تخمین ما در خروجی به واقعیت نزدیک‌تر باشد از تابع randint برای انتخاب تصادفی در بازه های استفاده شده بهره برده ایم.

* **تابع estimate\_arrival\_time**

از این تابع میتوان برای تخمین زمان رسیدن هواپیما به فرودگاه مقصد استفاده کرد اما همانطور که مستحضر هستید تنها زمانی می‌توانید زمان رسیدن به مقصد را تخمین بزنید که پارامترهایی مثل ساعت حرکت از مبدا، فاصله بین مبدأ و مقصد، سرعت هواپیما و غیره را داشته باشید پس کاملاً مشخص است که در ابتدا باید از تابع orthodromic\_distance و بعد از آن از تابع estimate\_flight\_length و سپس از تابع فوق استفاده کنید. از آرگومان row ستون local\_arrival\_time را می‌خوانیم و درون متغیر arrival\_time قرار می‌دهیم سپس با تابع isnull به بررسی null نبودن این متغیر می‌پردازیم اگر این متغیر مقدارش برابر با null نبود یعنی اینکه سایت هدف زمان رسیدن را به ما ارائه کرده و نیاز نیست که آن را تخمین بزنیم پس مقدار خودش را برمی‌گردانیم اما اگر این مقدار برابر با null بود باید مقدار تخمینی خود را با مقدار null عوض کنیم برای اینکار ابتدا باید توسط تابع timezome کتابخانه pytz یک آبجکت از روی کلاسBaseTzInfo با منطقه زمانی Asia / Tehran ایجاد می‌کنیم چرا که میخواهیم زمان ورود هواپیما به مبدأ را به منطقه زمانی کشور خودمان ذخیره کنیم. از row ستون local\_departure\_time را خوانده و توسط تابع split مقادیر درون آن را با ”:” از هم جدا کرده و درون متغیرهای مد نظر قرار می دهیم. همین عمل را برای ستون departure\_date\_YMD\_format درون row دوباره تکرار می کنیم. اینبار از row مقدار flight\_length\_min که قبلاً آنرا محاسبه کردیم بدست آورده و بر ۶۰ تقسیم می‌کنیم تا میزان ساعت پرواز بر ساعت به دست آید با استفاده از این مقدار قسمت‌های ساعت و دقیقه را محاسبه می‌کنیم حال می‌توانیم با کتابخانه jdatetime یک آبجکت از نوع کلاس datetime ایجاد کنیم. و همچنین توسط کلاس timedelta می‌توانیم اختلاف زمانی بدست آمده را در قالب یک آبجکت داشته باشیم تا بتوانیم با مقدار departure\_time\_obj آن را جمع کرده تا زمان تخمینی رسیدن به مقصد را بدست آوریم در نهایت هم با متد strftime می‌توانی خروجی را فرمت”Hour : Minute” به خروجی تحویل دهیم.

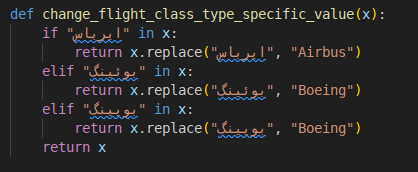
* **تابع change\_company\_name\_specific\_value**

****

برای یک‌دست کردن مقادیر درون فیلد company\_name از این تابع استفاده می‌کنیم تا یکسری مقادیر خاص را که که دارای شرایط زیر هستند تغییر دهیم:

1. اگر طول مقدار فیلد مد نظر بین[3, 1] بود کل مقدار به حروف بزرگ تبدیل شود.
2. در غیر این صورت اگر بزرگ‌تر از ۳ بود فقط کلمه اول آن با تابع Capitalize بزرگ شود
3. اگر مقادیر فیلد فارسی باشند با معادل انگلیسی آن‌ها جایگزین شوند.
4. در غیر این صورت اگر بعضی فیلد ها از کدهای IATA دو حرفی یا ICAO سه حرفی به جای نام شرکت ها استفاده شده این مقادیر با نام انگلیسی شرکت جایگزین شوند.
5. در این صورت اگر هیچ کدام از شرایط بالا ملاقات نشد مقدار فعلی فیلد بدون تغییر برگشت داده خواهد شد.

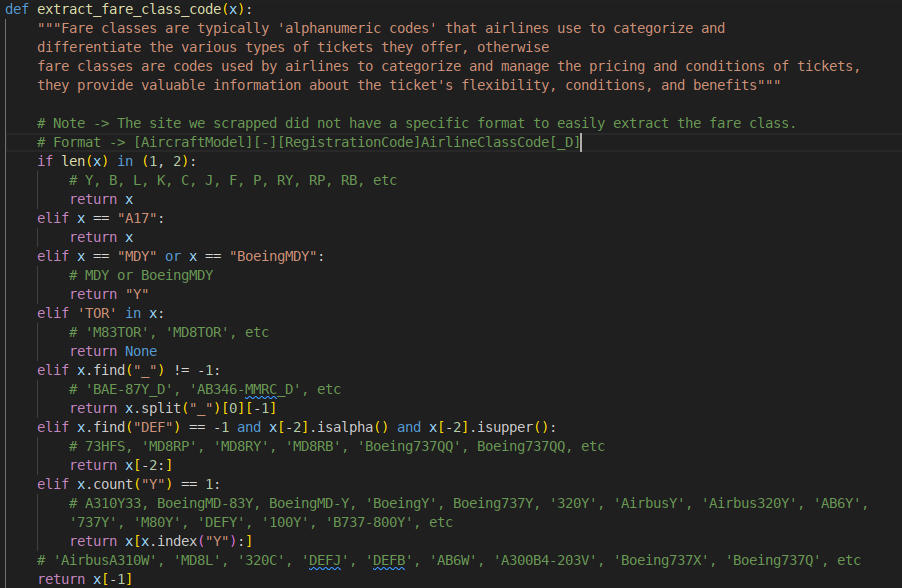
* **تابع change\_flight\_class\_type\_specific\_value**



این تابع نیز برای یک‌دست کردن مقادیر فیلدهای درون ستون flight\_class\_type کاربرد دارد در شرایط زیر مقدار فیلد تغییر خواهد کرد:

1. اگر کلمه ”ایرباس” در x وجود داشت باید توسط متد آبجکت String به نام replace مقدار ”ایرباس” با Airbus جا به جا شود.
2. در غیر این صورت اگر کلمه ”بوئینگ” یا ”بویینگ” در x وجود داشته باشد باید با Boeing جا به جا شود.
3. در غیر این صورت مقدار x بدون تغییر برگشت داده شود.

* **تابع extract\_fare\_class\_code**

سایت هدف مقادیر یک دست و تمیزی برای ستون **کلاس نرخی** به ما ارائه نمی‌دهد ولی با بررسی کلاس نرخی شرکت ها از طریق سایت رسمی آن‌ها و سرچ در اینترنت و همچنین بررسی مقادیر کلاس نرخی ای که سایت هدف در اختیار ما قرار می‌دهد می‌توان متوجه شد که **اکثریت** شرکت ها از کلاس‌های نرخی تک حرفی یا دو حرفی استفاده می‌کنند تا اطلاعتی مثل کلاس پروازی، میزان تخفیف بلیت، جریمه کنسلی و غیره را برای بلیت خریداری شده توسط مسافر مشخص کنند. ولی تعداد کمی از شرکت ها هم هستند که از کلاس نرخی ۳ حرفی هم استفاده می کنند. مثل شرکت هواپیمایی Emirates که از کلاس نرخی A17 برای نمایش کلاس پروازی اکونومی خود استفاده می کند. همچنین اگر به بررسی عمیق تر مقادیر کلاس نرخی سایت هدف بپردازیم متوجه می‌شویم که **تقریباً** از ساختار زیر برای نمایش کلاس نرخی هواپیماها استفاده می‌کند که برای ما مطلوب نیست و باید آن را در دیتاست خود اصلاح کنیم:

**[AircraftModel][-][RegistrationCode]AirlineClassCode[\_D]**

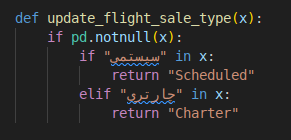
این ساختاری است که بنده با بررسی تعداد زیادی کلاس نرخی متوجه آن شده‌ام و  **اکثریت** کلاس‌های نرخی فراهم شده توسط سایت هدف این ساختار را دنبال می کنند. احتمالاً متوجه شده‌اید تنها مقداری که همیشه وجود دارد مقدار AirlineClassCode است و مابقی موارد حالت اختیاری دارند **می‌توانند** وجود نداشته باشند و ما در تابع فوق به دنبال استخراج بخش AirlineClassCode می باشیم به همین دلیل در پیرو این هدف باید شروط زیر را طبق عکس فوق پیاده‌سازی کنیم:

1. اگر طول x برابر با ۱ و یا ۲ بود آنگاه خود x را برگردان.
2. در غیر این صورت اگر x برابر با MDY یا BoeingMDY بود آنگاه مقدارY برگردان.
3. در غیر این صورت اگر مقدار TOR درون x وجود داشت آنگاه مقدار None را برگردان (مقدار TOR هرجا بوده سایت هدف مشخص نکرده این بلیت برای چه کلاس نرخی است)
4. در غیر این صورت اگر مقدار ”\_” درون x وجود داشت آنگاه یک کاراکتر قبل از ”\_” را برگردان.
5. در غیر این صورت اگر تعداد Y درون x برابر با ۱ بود آنگاه از ایندکس Y تا آخر رشته x را برگردان.
6. در غیر این صورت آخرین کاراکتر رشته x را برگردان.

**نکته** : توجه شود که ترتیب این شروط بسیار دارای اهمیت است و باید حتماً رعایت شوند.

طبق عکس می‌توانید به راحتی متوجه شوید که هر شرط با توجه به کدام مقدار رشته x اجرا خواهد شد نمونه‌هایی را فراهم کرده‌ایم تا فهم این تابع برای شما ساده‌تر شود.

* **تابع update\_flight\_sale\_type**



از این تابع نیز برای تبدیل مقادیر درون ستون flight\_sale\_type به معادل انگلیسی آن‌ها استفاده خواهد شد. همانطور که از تابع بالا مشخص است مقدار x با تابع notnull بررسی خواهد شد اگر این مقدار برابر با null بود تغییری رخ نخواهد داد ولی اگر این مقدار برابر با null نبود، اگر کلمه ”سیستمی” در x وجود داشت مقدار Scheduled برگشت داده شود و در غیر این صورت اگر کلمه ”چارتری” در x وجود داشت آنگاه رشته Charter برگشت داده خواهد شد.

**ماژول make\_processed\_dataset.py.py** :

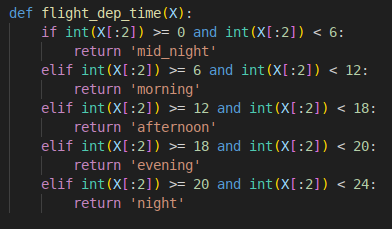
این ماژول صرفاً یک فایل **اجرایی** است از تمام توابعی که در فاز دوم به شما توضیح دادیم به عبارتی دیگر علاوه بر تمام توابعی که در ماژول هایی مثل tickets\_preprocessing.py و utils.py وجود دارند از کانفیگ های ماژول settings.py و توابع پر کاربرد پکیج common\_utils و کتابخانه‌هایی مثل pandas و hazm بهره برده ایم. در‌واقع با به اجرا در آوردن این ماژول تمامی موارد گفته شده به اجرا خواهند درآمد و تغییرات لازمه را روی دیتاست خام ما انجام خواهند داد تا بتوانیم از خروجی این فاز یک یک دیتاست تمیز شده می‌باشد در فازهای بعدی پروژه استفاده کنیم.

### **فاز سوم پروژه**

آماده کردن داده‌های تمیز شده برای مدل یادگیر (اعمالی مثل نرمال سازی، تبدیل داده ها و غیره)

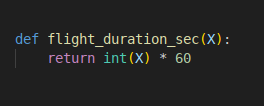
**ماژول build\_features.py**

* **تابع flight\_dep\_time :**



از این تابع برای تبدیل ساعت خروج از مبدأ به۴ دسته داده Categorical استفاده می‌شود. مقدار X[:2] برابر با قسمت ساعت می‌باشد به طور مثال اگر X برابر با ”12:33” باشد مقدار X[:2] برابر با ۱۲ خواهد بود که برای انجام عمل مقایسه ریاضیاتی، آن را توسط تابع int به عدد صحیح تبدیل می کنیم.

* **تابع flight\_duration\_sec :**



کاملاً واضح است که این تابع با دریافت مدت زمان طول پرواز در متغیر X آن‌ها از دقیقه به ثانیه تبدیل می کند.

* **مابقی کدهای درون ماژول build\_features.py :**
* df['departure\_date\_YMD\_format']=df['departure\_date\_YMD\_format'].apply(lambda d: datetime.strptime(d, '%Y-%m-%d'))

تبدیل نوع مقادیر درون ستون departure\_date\_YMD\_format از آبجکت String به آبجکت jdatetime توسط اعمال با تابع apply و تابع خطی lambda با فرمت تاریخ ”Y-m-d”

* df["year"] = df['departure\_date\_YMD\_format'].map(lambda x: x.year)
* df["month"] = df['departure\_date\_YMD\_format'].map(lambda x: x.month)
* df["day"] = df['departure\_date\_YMD\_format'].map(lambda x: x.day)

قطعه کدهای بالا برای تجزیه کردن ستون departure\_date\_YMD\_format به سه ستون سال، ماه و روز می باشد. البته باید توجه شود که مقدار x یک آبجکت از نوع jdatetime می‌باشد به همین دلیل خیلی راحت می‌توانیم مقادیر سال، روز و ماه را جدا کنیم.

* # ‌Converting the flight local\_departure\_time into proper time

df['dep\_flight\_time'] = df['local\_departure\_time'].apply(flight\_dep\_time)

* # Converting the flight duration to seconds

df['duration\_sec'] = df['flight\_length\_min'].apply(flight\_duration\_sec)

از قطعه کدهای بالا به ترتیب برای تبدیل مقادیر ستون local\_departure\_time به مقادیر Categorical و قرار دادنشان در ستون جدیدی به نام dep\_flight\_time و تبدیل مقادیر flight\_length\_min از دقیقه به ثانیه قرار دادن آن‌ها در ستون جدیدی به نام duration\_sec استفاده خواهد شد.

* df.drop(["national\_departure\_code",

"departure\_airport",

"national\_arrival\_code",

"arrival\_airport",

"orthodromic\_distance\_KM",

"flight\_length\_min",

"departure\_date\_YMD\_format",

"local\_departure\_time",

"local\_arrival\_time",

"flight\_number"],

inplace=True,

axis=1)

از تابع drop برای حذف ستون‌های فوق استفاده شده است. آرگومان inplace به ما این اجازه را خواهد داد که بدون انجام عمل انتساب تغییرات را برروی دیتافریم خود اعمال کنیم. اگر هم مقدار axis برابر با ۱ باشد آنگاه ستون‌های مد نظر در تمام سطرها پاک می شوند.

* df.to\_csv("path of your csv file", index=False)

بعد انجام تغییرات برای دیتافریم خود توسط متد to\_csv می‌توانیم تغییرات اعمال شده را برروی یک فایل csv ذخیره کنیم.

* # Converting categorical column

df = pd.get\_dummies(df, columns=['departure\_city', 'arrival\_city',

'company\_name', 'flight\_sale\_type',

'fare\_class\_code', 'dep\_flight\_time'])

تابع get\_dummies در کتابخانه pandas برای تبدیل متغیرهای طبقه‌بندی (Categorical) (همچنین به عنوان متغیرهای اسمی شناخته می‌شوند) به **مجموعه‌ای از ستون‌هایی با مقدار باینری** استفاده می‌شود که اغلب به آنها متغیرهای ساختگی یا متغیرهای نشانگر می‌گویند. این فرآیند هنگام کار با الگوریتم‌های یادگیری ماشینی که نیاز به ورودی **عددی** دارند ضروری است، زیرا بسیاری از الگوریتم‌ها **نمی‌توانند** مستقیماً داده‌های طبقه‌بندی را مدیریت کنند.

**فایل ژوپیر 1.4check-correlation-build-features.ipynb :**

* # Correlation Matrix

df.corr()

تابع corr ماتریس همبستگی بین ویژگی‌ها را در خروجی اش به ما نمایش خواهد داد از این تابع زمانی استفاده می‌کنیم که تا متوجه شویم کدام ویژگی با ویژگی هدف ارتباط نزدیک تر دارد و یا اینکه کدام ویژگی‌ها به هم مشابه هستند. در خصوص پروژه ایی که ما با آن درگیر هستیم خروجی زیر را مشاهده خواهیم کرد.



ضریب همبستگی برای سنجیدن میزان همبستگی بین داده‌ها استفاده می‌شود حال اگر:

1. ضریب همبستگی نزدیک به مقدار ۱ باشد بدین معنی است که ضریب همبستگی مثبت قوی است.
2. ضریب همبستگی نزدیک به مقدار ۱- باشد بدین معنی است که ضریب همبستگی منفی قوی است.
3. ضریب همبستگی نزدیک به مقدار ۰.۵ باشد بدین معنی است که ضریب همبستگی مثبت ضعیف است
4. ضریب همبستگی نزدیک به مقدار ۰.۵- باشد بدین معنی است که ضریب همبستگی منفی ضعیف است.
5. ضریب همبستگی نزدیک به مقدار صفر باشد بدین معنی است که همبستگی بین داده‌ها وجود نداشته و داده‌ها با هم رابطه‌ایی ندارند.

حال به طبق تصویر بالا داریم:

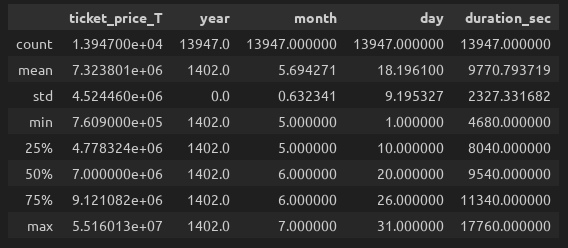
1. بین ticket\_price\_T و ticket\_price\_T مقدار ۱ مشاهده شده این بدین معنی است که این دو ویژگی با هم کاملاً یکسان هستند و این منطقی می‌باشد چرا ویژگی ticket\_price\_T با خودش مقایسه شده است. همین مورد برای month ، year و دیگر ویژگی‌ها هم وجود دارد.
2. بین ticket\_price\_T و duration\_sec مقدار نزدیک به 0.5 مشاهده شده این بدین معنی است این دو ویژگی باهم ارتباط دارند ولی این ارتباط از نوع مثبت ضعیف می باشد. ساده‌تر توضیح دهیم یعنی اینکه هرچی مقدار duration\_sec بیشتر شود مقدار ticket\_price\_T هم بیشتر خواهد شد هرچند که باید توجه کرد این ارتباط همچنان ضعیف است یعنی اینکه ticket\_price\_T صرفاً به duration\_sec وابستگی ندارد و باید ویژگی‌های دیگر را هم بررسی کنیم.
3. بین day و month مقداری نزدیک به ۰.۵- مشاهده شده این بدان معنی است که این دو ویژگی باهم ارتباط داشته و این ارتباط از نوع منفی ضعیف می باشد.
4. همانطور که متوجه شده‌اید در رابطه با ویژگی year با دیگر ویژگی ها و بلعکس و همچنین ارتباط ویژگی year با خودش، مقدار NaN گزارش شده است این امر به چند دلیل امکانش هست اتفاق بی افتد:

* **وجود Missing Values** : اگر در ویژگی مد نظر مقدار null وجود داشته باشد ضریب همبستگی این ویژگی برابر با مقدار NaN خواهد شد.
* **وجود Constant Values** : اگر در ویژگی مد نظر مقادیر یکسانی وجود داشته باشند به طوری که موجب شوند واریانس مجموعه پایین بیاید(هر چه واریانس پایین‌تر باشد مقادیر به خط میانگین نزدیک‌تر خواهند شد این بدان معنی است که مقادیر مجموعه شما خیلی به هم نزدیک هستند) حال اگر همه مقادیر ویژگی مد نظر یکسان باشند در نتیجه تابع corr مقدار NaN را برای نمایش خواهد داد.
* **Division By Zero** : ممکن است حین محاسبه همبستگی تقسیم برروی عدد صفر یا یک عدد کوچک خیلی نزدیک به صفر اتفاق می‌افتد که این موجب نمایش NaN خواهد شد.

در مورد پروژه ما اگر به دیتاست آماده سازی شده دقت کنید متوجه خواهید شد که در همه سطرهای ستون year مقدار 1402 وجود دارد که این همان دلیل نمایش NaN در ضریب همبستگی این ویژگی با خودش و بقیه ویژگی‌ها می‌باشد چرا که Constant Values وجود دارد.

* df.describe()

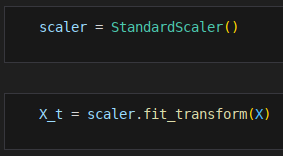
این متد خلاصه ایی از اطلاعات آماری هر یک از ستون‌های یک دیتافریم را به شما نمایش خواهد داد. در خصوص پروژه ما خروجی زیر را مشاهد خواهید کرد:



طبق تصویر بالا داریم:

1. مقدار انحراف معیار (std) ویژگی year برابر با صفر است این یعنی اینکه تمام مقادیر موجود در ویژگی year مقدار یکسانی دارند.
2. تعداد مقادیر موجود در ویژگی ticket\_price\_T برابر ۱۳۹۴۷ می باشد.
3. مقدار ماکسیمم ویژگی day برابر با 31 می باشد.
4. مقدار انحراف معیار ویژگی month عدد کوچکی می‌باشد این بدان معنی است که تنوع مقداری زیادی در تعداد ماه ها نداریم که منطقی است ما در پروژه فقط بلیت های موجود در ماه مرداد، شهریور و مهر را کراول کردیم یعنی مقادیر ما ۵ ، ۶ و ۷ می باشد.
5. مقدار انحراف معیار ویژگی ticket\_price\_T بسیار زیاد می‌باشد که این نشان از پراکندگی فراوان مقادیر موجود در این ویژگی است.
6. مقادیری که در %25 ، %50 و %75 قرار دارد به ترتیب مقادیری هستند که نشانه گر چارک اول، چارک دوم (میانه) و چارک سوم می باشد.

**فایل ژوپیتر 1.5train-and-predict-model.ipynb :**



StandardScaler کلاسی برای انجام یک نوع تکنیک پیش پردازش در فریمورک scikit-learn (sklearn) می‌باشد که برای استانداردسازی ویژگی ها با حذف میانگین و مقیاس بندی به واریانس استفاده می شود. این فرآیند می تواند برای بسیاری از الگوریتم های یادگیری ماشین مفید باشد زیرا به بهبود همگرایی و عملکرد کمک می کند. به عبارتی ساده‌تر در نظر بگیرید که ویژگی هایی به نام حقوق و سن وجود داشته، این دو ویژگی با هم فاصله مقداری زیادی دارند و این امر تأثیر منفی را در پیش‌بینی مدل ما خواهند گذاشت برای جلوگیری از این تأثیر می‌توانیم با یک تبدیل مقیاس این دو ویژگی را در یک بازه عددی مشخص نرمال سازی سازی کنیم. در ابتدا یک آبجکت از کلاس StandardScaler ایجاد می‌کنیم و سپس توسط متد fit\_transform تبدیل مربوطه را روی ویژگی‌های مد نظر اعمال خواهیم کرد.

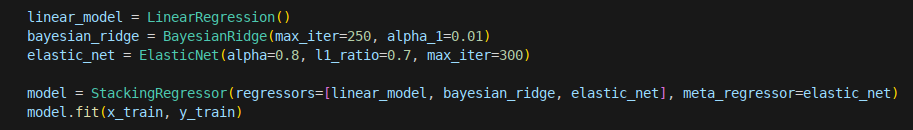
* x\_train, x\_test, y\_train, y\_test = train\_test\_split(X\_t, y, test\_size=0.2)

از تابع train\_test\_split برای جداکردن داده‌های تست و آموزش برای تعیلم مدل یادگیر و سنجش پیش‌بینی آن از طریق روش **نمونه برداری تصادفی بدون جایگزینی** به صورت ۲۰-۸۰ یا ۳۰-۷۰ می‌توان استفاده کرد. مقدار X\_t در‌واقع ویژگی‌های مقیاس بندی شده و مقدار y ویژگی هدف ما می باشد. از آرگومان test\_size هم برای اندازه تقسیم داده استفاده کرده‌ایم به این صورت که اگر اگر میزان 20-80 استفاده کنیم یعنی اینکه ۸۰ درصد داده‌ها برای آموزش مدل و ۲۰ درصد آن‌ها برای تست استفاده خواهند شد. خروجی این تابع چهار مقداری می‌باشد که در قطعه کد بالا مشاهده می کنید. x\_train و y\_train مقادیری می‌باشند که برای آموزش مدل و y\_test و x\_test مقادیری هستند که برای تست پیش‌بینی مدل استفاده خواهند شد.

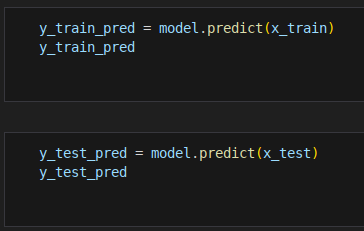
### **فاز چهارم پروژه**

این فاز مربوط به آموزش دادن مدل و پیش‌بینی قیمت می باشد. در بخش معرفی پروژه اشاره کردیم، مسأله ایی که ما با آن رو به رو هستیم یک مسأله از نوع **رگرسیون** می‌باشد چرا که در نهایت می‌خواهیم هزینه یک بلیت هواپیمایی را پیش‌بینی کنیم و این مقدار، مقداری از نوع عددی و پیوسته می‌باشد و مسائلی که در آن‌ها به پیش‌بینی یک عدد پیوسته می‌پردازیم مسائلی هستند که **باید** به روش رگسیون حل شوند. پس مدل هایی که در این فاز استفاده می‌کنیم مدل هایی هستند که به عنوان یک **Regressor** عمل خواهند کرد.

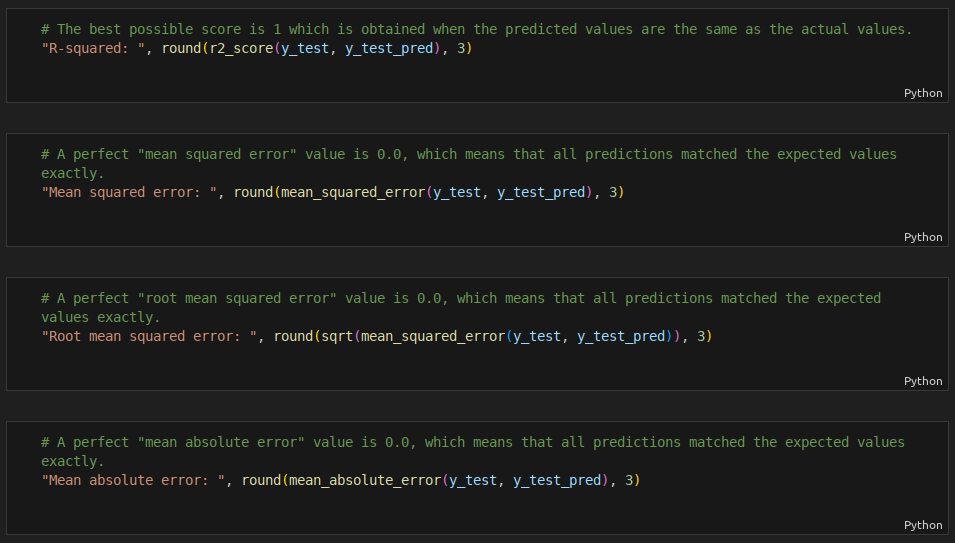
**فایل ژوپیتر 1.5train-and-predict-model.ipynb :**

در قطعه کد بالا از سه کلاس LinearRegression , BayesianRigde و ElasticNet استفاده شده است این سه کلاس به عنوان یک Regressor روی داده‌های ما عمل خواهند کرد. کلاس StackingRegressor یک نوع مدل یادگیر به روش Ensemble می‌باشد که با انباشته کردن(Stacking) اجازه می دهد تا با استفاده از خروجی هر تخمینگر(مدل یادگیر) به عنوان ورودی تخمینگر دیگر، از قدرت هر تخمینگر استفاده شود. در بعضی مسائل صرفاً استفاده از یک مدل یادگیر پیش‌بینی خوبی را به عمل نخواهد آورد در نتیجه با ترکیب چند مدل با هم دیگر می‌توان پیش‌بینی بهتری را انجام داد. در نهایت نیز با استفاده از متد fit میتوان مدل یادگیر را روی داده‌های خود با استفده از x\_train و y\_train فیت کرد. سینتکس استفاده از این متد برای تمامی مدل های یادگیر یکسان است نمونه‌ها را همرا با مقدار هدف به متد فیت می‌دهیم تا اعمال لازمه را برای آموزش مدل مدنظر به کار ببرد. کلمه فیت اصلاح عامیانه ایی است که برای هر مدل یادگیر چه مدل های رگرسیون و چه مدل های دسته بندی می‌توانیم به کار ببریم اما اینکه تابع fit دقیقاً برای هر مدل به چه شکل کار می‌کند نیازمند بررسی و مطالعه عمیق‌تر مدل مد نظر می‌باشد اما به طور کلی می‌توان گفت که در مدل های رگرسیون زمانی که آن‌ها را فیت می کنیم به دنبال یک θ می‌گردیم که این θ ها در‌واقع ضریب ویژگی‌های مد نظر ما می‌باشند به عبارتی باید گفت که کل عملیات fitting طریقه رسیدن به مقدار مناسب و بهینه ای برای θ های مد نظر خواهد بود. در ادامه در رابطه با آرگومان های استفاده شده توضیحاتی را خدمت شما ارائه خواهم داد:

1. **max\_iter** : حداکثر پیمایشی که نیاز است تا به همگرایی برسیم. باید گفته شود که در الگوریتم مدل یادگیر رگرسیونی مد نظر برای اینکه بهترین مقدار را برای θ ها پیدا کنیم از الگوریتم گرادیان کاهشی بهره خواهیم برد. گرادیان کاهشی نیز همیشه به دنبال یک کمینه محلی برای تابع هزینه مد نظر خودش است. تابع هزینه نیز برای الگوریتم گرادیان کاهشی باید یک تابع محدب باشد که ضرایب θ را به گونه ایی پیدا خواهد کرد که مقدار هزینه یا همان خطای تابع کم باشد.
2. **alpha\_1** : مقدار پارامتر alpha معروف به پارامتر میزان نرخ یادگیر مدل می‌باشد این پارامتر وظیفه دارد که از overfit شدن مدل جلوگیری کند به عبارتی وظیفه کنترل ضرایب ویژگی‌ها یا همان θ ها برعهده دارد و همچنین به سرعت همگرا شدن در الگوریتم گرادیان کاهشی کمک خواهد کرد.
3. **l1\_ratio** : وظیفه ایجاد توازن بین تنظیم کننده های (Regularizations) L1 و L2 را در تابع هزینه برعهده دارد. وقتی مقدار این آرگومان برابر با ۱ باشد یعنی داریم از Lasso Regression و و قتی برابر با مقدار صفر باشد یعنی داریم از Rigde Regression و همچنین وقتی مقداری بین ۱ و صفر داشته باشد یعنی داریم از ترکیبی از هردوی این‌ها استفاده می کنیم. گفتی است در هردوی این‌ها از عبارتی به نام Term استفاده شده که اگرچه اسمشان یکسان است ولی کاربردی متفاوت از هم دارند. عبارت Term در Lasso ضریب ویژگی‌هایی که اثر کمی در پیش‌بینی دارند را صفر می کند( به نوعی ویژگی را حذف می کند) اما عبارت Term در Rigde تأثیر ویژگی‌های کم اهمیت را کاهش خواهد داد و به نوعی جلوی بزرگ‌تر شدن ضرایب را خواهد گرفت.
4. **regressors** : به کمک این آرگومان تمامی ‌Base Regressor های ساخته شده را برای انجام عمل یادگیری به روش Esemble به مدل StackingRegressor خواهیم داد.
5. **meta\_regressor** : مقدار این آرگومان یک مدل یادگیر می‌باشد که پیش بینی هایی را بر اساس خروجی های ترکیبی آرگومان regressors انجام خواهد داد.

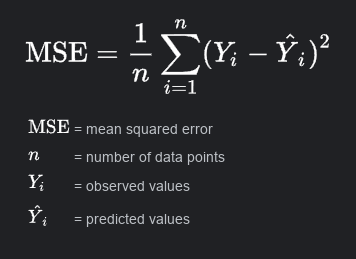


در تصویر بالا مشخص است که عمل پیش‌بینی را برروی داده‌های Train و Test توسط متد predict انجام داده‌ایم و خروجی را در متغیرهای مد نظر ذخیره کردیم. حال چرا برروی نمونه‌های آموزشی خود عمل پیش‌بینی را انجام داده ایم؟ این عمل به ما کمک می‌کند که بعداً توسط روش‌های ارزیابی مدل متوجه شویم که مدل ما برروی خود داده‌های آموزشی که قبلاً آن‌ها را در حین آموزش خود دیده بود چگونه عمل کرده است بدین ترتیب می‌توانیم متوجه overfit یا underfit شدن مدل خود شویم.

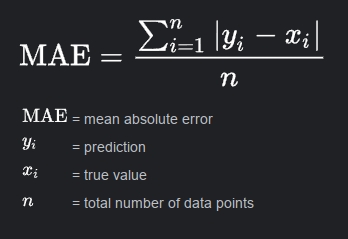


همانطور که در بخش تعریف پروژه اشاره کردم روش‌های ارزیابی یک مدل یادگیر رگرسیونی با استفاده از معیارهای میزان سنجش خطا انجام می شود. این معیارها مشخص خواهند کرد پیش‌بینی شما چقدر دچار خطا هستند مطلوب ترین مقدار برای وجود عدم خطا مقدار صفر می‌باشد. خطای صفر نشان می‌دهد که مدل شما تمام پیش‌بینی ها را درست انجام داده است که در دنیای واقعی محال ممکن است که پیش‌بینی شما بدون خطا باشد چرا که در دنیای واقعی دیتاهایی که شما برای مدل خود فراهم می‌کنید دارای نویز خواهند بود و این نویز ها پیش‌بینی شما را دچار خطا خواهد کرد هرچند که روش‌هایی برای تشخیص نویز و کاهش آن‌ها وجود دارد اما کافی نخواهند بود و باز هم مواردی وجود خواهند داشت که موجب خواهند شد پیش‌بینی شما دارای خطا باشد در‌واقع تلاش ما در این است که بتوانیم مدل خود را در سمتی بهبود ببخشیم که به عدد صفر فقط و فقط نزدیک تر شود. از معیارهای خطایی که ما در این پروژه استفاده کردیم می‌توان موارد زیر را نام برد:

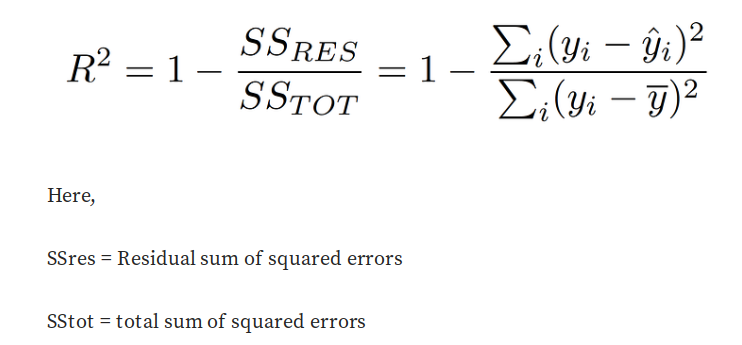
1. **mean\_squared\_error** : میانگین مربعات خطاها را اندازه گیری می‌کند که در واقع همان میانگین مجذور (مربع کامل- عدد به توان دو) اختلاف بین مقادیر پیش‌بینی شده و مقادیر واقعی آن‌ها می باشد. MSE معیاری برای سنجش کیفیت یک مدل است که از مجذور فاصله اقلیدسی به دست می‌آید و همیشه یک مقدار مثبت می باشد که با نزدیک شدن به مقدار صفر یعنی خطا کاهش یافته است. این معیار از فرمول زیر پیروی خواهد کرد:



1. **Root of mean\_squared\_error** : ریشه ی میانگین مربعات خطا می باشد. به عبارتی جذر از میانگین مربعات خطا می باشد. در برخی مواقع RMSE ترجیح داده می شود چرا که به خطاهای بزرگتر وزن بیشتری می دهد و آن را به موارد پرت حساس می کند. این بدان معنی است که خطاهای بزرگتر تأثیر نسبتاً بیشتری بر مقدار RMSE دارند. معیار خوبی برای زمانی است که می خواهید انحرافات قابل توجه بین مقادیر واقعی و پیش بینی شده را بدست آورید.
2. **mean\_absolute\_error** : میانگین خطای مطلق، میانگین تفاضل بین مقادیر واقعی و خروجی مدل است. علامت خروجی نادیده گرفته می‌شود چرا که اگر علامت را نادیده نگیریم، MAE محاسبه شده احتمالاً بسیار کمتر از تفاوت واقعی بین مدل و داده خواهد بود. میانگین خطای مطلق از فرمول زیر پیروی خواهد کرد:



1. **r2\_score** : در مورد اینکه یک مدل رگرسیون چقدر خوب با **داده‌های واقعی** فیت شده است استفاده می شود. به عنوان مثال، مدلی با مجذور r حدود 80 درصد به این معنی است که 80 درصد داده ها با مدل رگرسیونی فیت شده اند. از حالتی که بیان شد، می توان فهمید که مقادیر بالاتر مربع r نشان دهنده مدل بهتری است. اگرچه، این همیشه درست نیست. در برخی موارد بسیار نادر، ممکن است مقدار مربع r پایین را نیز بخواهیم. مقدار مربع r از صفر تا 1 است که صفر نشان دهنده یک مدل ضعیف و 1 نشان دهنده یک مدل مناسب است. چیزی که در اینجا باید به آن توجه کرد این است که r2\_score در sklearn می‌تواند یک مقدار منفی نیز باشد این یعنی اینکه مدل شما حتی از حالت صفر هم بدتر کار می کند. این زمانی اتفاق می افتد که خط پیش بینی رگرسیون شما بدتر از حالت مقدار میانگین عمل می کند. همچنین، نکته ای که باید به خاطر داشته باشید این است که مقدار r2\_score تحت تأثیر مقادیر پرت نیز قرار می گیرد. در نهایت این معیار از فرمول زیر پیروی خواهد کرد:



از فرمول بالا می‌توان متوجه شد که هرگاه دقیقاً پیش‌بینی های مدل مشابه با مقادیر واقعی باشد صورت کسر همواره صفر خواهد شد و در نتیجه تفاضل ۰ – ۱ برابر با ۱ خواهد بود که همانطور که گفته شد وقتی مقدار r2\_score برابر با عدد یک باشه یعنی ما یک مدل مناسب را دارا هستیم اما در مقابل اگر صورت کسر برابر با مخرج کسر باشد طبیعتاً تفاضل ۱ با ۱ را خواهیم داشت که برابر با صفر می‌باشد و این یعنی اینکه مدل ما ضعیف عمل کرده است.

# فصل **۴**

## **خروجی های پروژه و اصلاحات مورد نیاز**

در این بخش از داکیومنت به ارائه خروجی های پروژه می‌پردازیم و به قسمت‌هایی از پروژه اشاره خواهیم کرد که می‌توان در جهت گرفتن خروجی مناسب‌تر و بهتر، بیشتر روی آن‌ها کار کرد.

### **خروجی ها**

خروجی ها در چهار فاز آماده شده و خدمت شما در زیر ارائه خواهند شد:

1. با مراجعه به دایرکتوری **/mostafa\_vahdani\_bachelor\_project/data/raw/…** می‌توانید خروجی **فاز اول پروژه** را در فایل csv به نام **flight\_tickets\_dataset** مشاهده کنید. این فایل شامل تمام **داده‌های خام** ممکن است که سایت هدف از تاریخ ۱۴۰۲/۰۵/۱۷ تا تاریخ ۱۴۰۲/۰۷/۱۵ در اختیار ما قرار داده و مورد کراول قرار گرفته اند.
2. خروجی **فاز دوم پروژه**  در دو مرحله تهیه شده است در مرحله اول شما با مراجعه به دایرکتوری **/mostafa\_vahdani\_bachelor\_project/data/processed/…**فایلcsv به نام**flight\_tickets\_dataset** را مشاهده کنید. این فایل شامل تمام **داده‌های تمیز شده** ایی است که از خروجی فاز اول پروژه بدست آورده‌ و از مرحله پیش پردازش گذر داده ایم. و در مرحله دوم نیز شما با مراجعه به دایرکتوری **/mostafa\_vahdani\_bachelor\_project/data/interim/…** می‌توانید فایلی با فرمت csv به نام **build\_features\_flight\_tickets\_dataset** مشاهده کنید. این فایل شامل تمام داده‌های است که برای آموزش دادن مدل یا مدل های مد نظر به آن‌ها نیاز داریم.
3. و در نهایت نیز با مراجعه به دایرکتوری **/mostafa\_vahdani\_bachelor\_project/data/interim/…** می‌توانید خروجی **فاز سوم پروژه** را در فایل csv به نام **final\_flight\_tickets\_dataset** مشاهده کنید. این فایل شامل تمام داده‌های **آماده سازی شده** برای مدل های یادگیر مد نظر می باشد.
4. در فاز چهارم پروژه همانطور که می‌دانید و در بخش توضیح کدها مشاهده کردید به آموزش، پیش‌بینی و ارزیابی مدل خود پرداخته ایم. خروجی زیر نشان دهنده ارزیابی ما از پیش‌بینی مدل یادگیر برروی **داده‌های آموزش دیده**  می باشد:

**نکته** : توجه کنید که ارزیابی را روی داده پیش‌بینی شده توسط داده‌های آموزش دیده بروری خود مدل انجام داده‌ایم نه ارزیابی داده‌های پیش‌بینی شده توسط داده‌های تست.

* **('R-squared: ', 0.734)**
* **('Mean squared error: ', 5386668392338.481)**
* **('Root mean squared error: ', 2320919.73)**
* **('Mean absolute error: ', 1425455.501)**

از میزان خطای بدست آمده باید متوجه شده باشید که اولاً خروجی تابع r2\_score به تنهایی برای ارزیابی مدل های ما کافی نمی باشد چرا که می‌بینید از نظر معیار r2\_score مدل ما خوب روی داده‌های آموزشی فیت شده است و دوماً از نظر معیارهای دیگر حتی مدل روی انجام عمل پیش بینی برروی داده‌های آموزش دیده خودش هم اصلاً خوب عمل نکرده و میزان خطای بدست آمده با عدد صفر فاصله بسیار زیادی دارد. برای بهبود هر چه بهتر این میزان و نزدیک‌تر شدن آن به عدد صفر در بخش اصلاحات چند راهکار را ارائه خواهم داد.

### اصلاحات

1. با توجه به بالا بودن معیارهایی مثل MSE و MAE میتوان نتیجه گرفت که یا از مدل مناسبی استفاده نمی‌کنیم و یا دیتاست ما به خوبی پیش پردازش نشده است. با استفاده کردن از مدل های رگرسیونی مختلف با پارامترهای متفاوت به طور قطع می‌توان گفت که نقص کار از انتخاب مدل یادگیر نمی باشد و ما باید برروی آماده سازی داده‌ها بیشتر کار کنیم. چرا که آماده سازی هرچه بهتر داده‌ها از انتخاب مدل یادگیر اولویت بسیار بالا و حیاتی تری دارد.
2. می‌توانید ویژگی‌های مرتبط بیشتری را اضافه و یا ویژگی‌های نامرتبط را حذف کنید و یا به دنبال این باشید که مقادیر بهتر و دقیقتری را برای ستون مد نظر پیدا کنید به طور مثال هنگامی که ما در حال محاسبه فاصله دایره-بزرگ (فاصله هوایی) در تابع orthodromic\_distance بودیم تعداد توقف هایی که بین مسیر وجود دارند و همچنین موقعیت جغرافیایی ایستگاه های توقف را در نظر نگرفتیم. ممکن است در یک پرواز، مسیر مستقیم(بدون توقف) بین مبدأ و مقصد وجود نداشته باشد و هواپیما مربوطه مجبور باشد بین مسیر توقف هایی را انجام دهد. گفتنی است به پروازهایی که دارای توقف هستند Connecting Flight و یا Direct Flight و به پروازهای بدون توقف Non-Stop Flight گفته می شود. اطلاعات مربوط به توقف یا عدم توقف هواپیما را می‌توان توسط **کد پرواز** تهیه شده در دیتاست در ستون flight\_number پیدا کرد.
3. با در نظر گرفتن مورد ۲ که اشاره کردیم باید حواستان باشد که اگر تعداد توقف های شما بیشتر شود پس مدت زمان پرواز شما نیز بیشتر خواهد شد اینجاست که باید Layover Time را با بدست آوردن تعداد توقف ها بدست آورید و در تابع estimate\_flight\_length آن را اضافه کنید تا تخمین بهتری را از مدت زمان پرواز بدست آورید و نتیجه را در ستون‌های flight\_length\_min و local\_arrival\_time جایگذاری کنید.
4. با توجه به اینکه سایت هدف به درستی کلاس پروازی(اکونومی، بیزنس، فرست و غیره) را برای بلیت های خود مشخص نمی‌کرد، شما می‌توانید با استفاده از کلاس نرخی تهیه شده و نام شرکت هواپیمایی به دنبال کلاس پروازی شرکت بگردید و آن‌ها را در دیتاست خود جایگذاری کنید چرا که کلاس پروازی نقش موثری در تعیین قیمت بلیت های هواپیمایی خواهد داشت.
5. به جای استفاده از روش get\_dummies می‌توانید از روش‌های دیگر برای تبدیل داده Categorical به داده عددی (ایندکس گذاری شده) استفاده کنید چرا که اگر کاردینالیتی داده‌های Categorical ها دیتاست شما بالا باشد ( تعداد داده‌های Categorical یکتا درون ستون‌های دیتاست زیاد باشد)، تبدیل آن‌ها به ستون هایی با مقادیر باینری صرفاً تعداد ستون‌های شما را زیاد می‌کند و ستون بیشتر به معنی ویژگی بیشتر می‌باشد و استفاده از تعداد ویژگی‌هایی زیاد از حد می‌تواند روی پیش‌بینی مدل شما تأثیر منفی بگذارد چرا که ابعاد ویژگی‌های شما بیشتر شده اند.
6. قیمت بلیت هواپیما در روزهای تعطیل، مناسبت‌های مختلف و بعضی فصل های سال به دلیل تقاضای بیشتر مسافرین افزایش می‌یابد مخصوصاً تعطیلاتی که براساس ماه های قمری می‌باشند برای نمونه روز میلاد امام رضا به طور قطع برروی قیمت بلیت مشهد تأثیرگذار خواهد بود. می‌توانید با دسته بندی کردن انواع تعطیلات و اختصاص دادن آن‌ها به تاریخ های مربوطه ورودی ای با جزئیات مرتبط بیشتر برای مدل خود فراهم کنید تا پیش‌بینی دقیق‌تری داشته باشید.

هدف ما در نهایت این بوده که بتوانیم ارزان ترین قیمت بلیت را در یک بازه زمانی مشخص پیش‌بینی کنیم برای این امر می‌توان بعد از اینکه آموزش، پیش‌بینی و ارزیابی های مد نظر را انجام دادیم با ذخیره پیش‌بینی ها و نمونه‌های مد نظر و سپس مینیمم گیری روی قیمت بلیت در بازه زمانی دلخواه، ارزان ترین قیمت بلیت را بازه زمانی مد نظر قبل از سفر خود پیدا کنیم.

# فصل **۵**

## **جمع‌بندی و نتیجه‌گیری**

در این داکیومنت از ابتدا سعی بر این بوده که شما خواننده گرامی را از اهمیت و طریقه استفاده از اطلاعات آگاه کرده، به تعریف پروژه و اهداف مد نظر پرداخته، ساختار دایرکتوری ها و ماژول ها را به شما توضیح داده، به توضیح بخش بخش کدها بپردازیم، خروجی های پروژه را در فازهای متفاوت به شما توضیح دهیم و در نهایت به توضیح نکات تکمیلی پروژه در جهت بهبود آن بپردازیم و برای اصلاح آن‌ها راه حل‌هایی را پیشنهاد دهیم. در نهایت باید گفت که دنیای علم داده بسیار دنیای عظیمی می‌باشد و اگر شما علاقه به ادامه در این زمینه به عنوان یک متخصص علم داده را دارید و می خواهید در این زمینه پروژه ایی به عمل می آورید همانطور که در مقدمه نیز به آن اشاره کردم نه تنها باید برنامه نویس خوبی باشید بلکه باید از علوم های دیگر به‌خصوص ریاضیات سر در بیاورید، مطالعه عمیق و متوالی و همچنین تمرین دائمی در این زمینه داشته باشید تا بتوانید با علم به اینکه می‌دانید دقیقا دارید چه کاری انجام می دهید پروژه های خود را جلو برده و به اتمام برسانید.

در پایان نیز از شما خواننده گرامی تشکر می‌کنم که به این داکیومنت توجه لازمه را داشته و آن را مطالعه کرده اید، امیدوارم که این پروژه و داکیومنت برای شما مفید و ارزشمند واقع شود و نکات خوبی را از آن یاد گرفته باشید تا بتوانید بعدها از آن نکات در پروژه های خویش بهره ببرید.

# **منابع**

* www.geeksforgeeks.org
* [medium.com](http://medium.com/)
* [en.wikipedia.org](http://en.wikipedia.org/)
* [scikit-learn.org](http://scikit-learn.org/)
* [stackoverflow.com](http://stackoverflow.com/)
* [stackexchange.com](http://stackexchange.com/)
* www.datacamp.com
* www.kaggle.com
* www.linkedin.com
* [github.com](http://github.com/)
* www.gps-coordinates.net
* [sites.utoronto.ca/webdocs/HTMLdocs/Book/Book-3ed/appe/iso3166.html](http://sites.utoronto.ca/webdocs/HTMLdocs/Book/Book-3ed/appe/iso3166.html)
* https://www.iata.org/en/publications/directories/code-search/
* https://metar-taf.com/airport/
* [https://www.flightradar24.com/data/airlines/#t](https://www.flightradar24.com/data/airlines/" \l "t)
* https://en.wikipedia.org/wiki/List\_of\_airline\_codes
* [https://en.wikipedia.org/wiki/List\_of\_aircraft\_registration\_prefixes#Post-1928\_table\_notes](https://en.wikipedia.org/wiki/List_of_aircraft_registration_prefixes" \l "Post-1928_table_notes)
* <https://en.wikipedia.org/wiki/List_of_aircraft_type_designators>