به نام خدا

گزارش تمرین 2 داده کاوی

1. پاکسازی داده ها و استفاده از dask و pyspark

ابتدا کتابخانه های مورد نیاز را وارد میکنیم و سپس دیتافریم مورد نظر را می خوانیم سپس توضیحاتی از داده ها می بینیم و متوجه می شویم که هم تعدادی outlier داریم و هم تعداد زیادی مقدار خالی.

در این جا کار پر کردن داده های خالی را هم با pandas و هم با dask و pyspark انجام می دهیم. بنابراین ابتدا موارد موردنیاز را برای استفاده از این دو نصب می کنیم. کاری که در این جا می کنیم این است که با استفاده از filln موارد مورد نیاز را پر می کنیم و سطرهایی که باقی می مانند حذف می کنیم. برای این کار در هر سه کتابخانه از دیکشنری استفاده می کنیم. موضوع جالب توجه این است که در هر سه دستورات تقریبا یکی است و تفاوتی که من در pyspark دیدم این بود که بعضی مقادیر خالی را غلط می خواند و به جای آن ها NA قرار می داد. برای حل این مشکل هرکجا NA بود حذف کردم و همچنین مشکل دیگری که وجود داشت در تبدیل دیتا فریم pandas به pyspark بود که برای حل آن فایل را مستقیم خواندم. هیچ کدام از این مشکل ها در کار با dask وجود نداشتند. از لحاظ زمان اجرا هم dask و pyspark هر دو سریع تر از pandas بودند و زمان اجرای دستورات با pandas 1.72 زمان اجرای دستورات با pyspark 0.88 و زمان اجرای دستورات با dask تنها 0.13 ثانیه بود.

در ادامه داده های تکراری را حذف کرده و همچنین 0.001 بالا و پایین همه داده های عددی را حذف می کنیم که outlier ها از بین بروند.

1. دیداری سازی داده ها و multiprocessing

برای این کار نمودار داده های قیمت را می کشیم و می توان دید که قیمت اکثر اتاق های ما چیزی کمتر از 1000 یورو است و میزانی که بیش ترین تکرار را در داده ها دارد چیزی حدود 400 تا 600 یورو است.

در مرحله بعد داده های مناطق را می کشیم و می بینیم که Nordhein\_Westfallen و Sachsen بیش ترین تعداد تکرار را در داده های ما دارند و این دو به اضافه Bayern بیش از نصف مناطق ما را تشکیل می دهند.

در این جا می خواهیم داده های ریز تر منطقه ای را به نمایش بکشیم و برای این کار ابتدا تعداد تکرار داده ها را محاسبه می کنیم. می بینیم که 198 داده داری که بعضی 200 بار تکرار شده اند و بعضی حتی بیش از 8000 بار. موضوعی که باید اینجا مدنظر قرار گیرد این است که این میزان برای نمایش در نمودار چندان مناسب نیست برای همین تابعی با نام neighbourhood\_clean تعریف کرده و داده هایی که کمتر از 1000 بار تکرار شده اند را به نام other تغییر نام می دهیم. همچنین این کار را بوسیله multiprocessing هم انجام می دهیم و می بینیم که انجام این کار با multiprocessing تقریبا 0.3 ثانیه سریع تر است. سپس نمودار regio2 را می کشیم و می بینیم که در اینجا Lepzig و Chemnitz و berlin از بقیه تکرار بیش تری دارند البته این بعد از میزان تکرار other (که داده های با تعداد پایین ما است) می باشد.

در نمودار بعدی انواع تبلیغات را بررسی می کنیم که اینجا می بینیم بیش تر آگهی ها از نوع آپارتمان هستند و بعد از آن آپارتمان هایی هستند که نوعشان مشخص نشده است. roof\_storey و ground\_floor هم در رده های بعدی قرار دارند.

نمودار بعدی میزان قیمت در هر شهر را نشان می دهد. همان طور که می بینیم هامبورگ و برلین از بقیه شهر ها گران تر هستند برلین پایتخت است و گران بودن آن منطقی است اما درباره هامبورگ می توان گفت دلیل گران تر بودن آن از برلین مرکز حمل و نقل بودن آن و همچنین توریستی بودن آن است.

در قسمت بعد می توان میزان میانگین قیمت انواع اتاق ها را مدنظر داد و قیمت ها با وجود کم بودن داده ها در بعضی قسمت ها مثل زیر شیروانی بنظر منطقی می آیند همان طور که انتظار می رود penthouse از بقیه انواع خانه ها گران تر است و ظاهرا خانه های زیر شیروانی هم بدلیل بزرگ بودن فضا به نسبت گران هستند. و ارزان ترین نوع خانه هم خانه های زیر زمین یا half basement هستند که این موضوع هم منطقی است.

نمودار بعدی میزان رابطه اندازه فضای خانه با قیمت را می سنجد و همان طور که انتظار می رود این میزان رابطه مستقیم دارد.

نمودار بعدی نوع لوکس بودن یا ساده بودن خانه ها را بررسی می کند و این نمودار هم به ما نشان می دهد که خانه های لوکس هزینه ای بسار بیش تر از خانه های معمولی و ساده دارند.

در نمودار بعدی می توان اجاره خانه های با شرایط مختلف را دید که خانه های دست اول از همه گران تر هستند و خانه های تمیز و تعمیر شده و تعمیر شده دست اول در رده های بعدی هستند.

نمودار بعدی سال ساخت و هزینه اجاره را نشان می دهند و با وجود این که outlier ها حذف شده اند ولی می توان دید که بعضی خانه های چند صد ساله هستند که هزینه های ماهیانه زیادی دارند دلیل این موضوع می تواند بزرگ بودن و حتی تاریخی بودن این بنا ها باشد. همچنین هر چه بیش تر به جلو می آییم می بینیم که قیمت ها به هم نزدیک تر می شوند و خانه هایی با هر قیمت وجود دارند اما این موضوع از حدود سال 1900 تا 1960 به این صورت نیست و قیمت خانه های در این بین اکثرا کم تر از 3000 یورو است که این موضوع می تواند بدلیل نوع ساخت در این دوره زمانی بازسازی نشدن ساختمان های این دوره یا حتی کم بودن داده های ما در این دوره باشد.

1. ساخت مدل و مهندسی ویژگی

در این جا قبل از شروع به حذف داده ها ابتدا داده های از نوع object که ممکن است تاثیری در قیمت داشته باشند را عددی کرده و سپس ستون های object باقی مانده را حذف می کنیم. (نکته ای که در این جا باید اشاره شود این است که ما از baseRent به جای totalRent استفاده می کنیم و دلیل این انتخاب ما تعداد بالای سطرهای خالی totalRent است.)

حالا ماتریس وابستگی را می کشیم و با توجه با مقادیر قیمت و بقیه مقادیر آن هایی که رابطه کمی با هزینه دارند حذف می کنیم.

حالا سعی می کنیم با استفاده از رگرسیون و ریج و لسو و randomforrestregressor مدل هایی برای داده های خود بسازیم.

می بینیم که Randomforrestregressor از بقیه بهتر عمل می کند.

با وجود این مدل ها ما هنوز هم تعداد زیادی ستون داریم که برای کم کردن آن ها از PCA استفاده می کنیم. با استفاده از PCA تعداد ستون ها را به 6 کاهش می دهیم ودوباره رگرسیون و randomforrestregressor را استفاده می کنیم. نمره رگرسیون نسبتا قابل قبول است و خیلی کم نشده است اما موضوع جالب توجه این است که امتیاز randomforrestregressor به اندازه رگرسیون پایین آمده است که نشان می دهد این روش به تعداد بالای ستون ها حساس است و با تعداد بیش تر خیلی بهتر عمل می کند.

1. مدلی سازی با dask و pyspark

برای مدل سازی با دسک از dask\_ml استفاده می کنیم نکته ای که اینجا وجود دارد این است که دسک به distributed حداقل 2.4.0 نیاز دارد و نسخه این پکیج در colab پایین تر است. به همین دلیل قبل از نصب این بسته ابتدا distributed را به روز رسانی می کنیم. برای انجام رگرسیون خطی ابتدا داده ها را از پاندا تبدیل به ddf می کنیم و سپس برای این که وارد predictor شوند آن ها را تبدیل به array می کنیم.

مدل سازی با pyspark کمی تفاوت دارد و باید پس از تبدیل کل دیتافریم به دیتافریم اسپارک ستون هایی که می خواهیم برای prediction استفاده کنیم جدا کرده و در یک ستون قرار دهیم و سپس ستون های فیچر و لیبل را مشخص کرده و مدل سازی را انجام می دهیم.

انجام رگرسیون خطی با sklearn برابر 0.12 و زمان انجام رگرسیون با dask\_ml 0.91 و زمان انجام رگرسیون با pyspark 17.37 ثانیه طول کشید.