پروژه درس داده‌کاوی – ترم پاییز ۱۴۰۱ – فاز 4 – دسته شماره ۲

سید محمد طاها طباطبایی – ملیکا ذبیحی نیشابوری

سوال اول : نسبت تعداد کالاهای خریداری شده به بازگشت به انبار به تفکیک سازمان

برای پاسخ‌دهی به این سوال از دیتاست PRODUCTINSTANCE استفاده کردیم. از طریق ویژگی AD_ORG_REF_ID به آی دی سازمان ها و از طریق ویژگی RETURNAMVALTOANBAR به وضعیت کالاها دسترسی پیدا کردیم. با پردازش این دو ستون در دیتاست نسبت تعداد کالاهای خریداری شده به بازگشت به انبار به تفکیک سازمان محاسبه شد.

توضیح فنی پیاده سازی

ابتدا مرحله ی پیش پردازش انجام می شود. برای دو ویژگی RETURNAMVALTOANBAR و AD_ORG_REF_ID داده هایی که id سازمان آنها ثبت نشده و یا وضعیت بازگشت به انبار مشخصی ندارند دور ریخته می شوند.

برای پردازش داده ها از یک دیکشنری به نام orgInfo استفاده می کنیم. کلید آن id سازمان بوده و value آن یک لیست است که در آن تعداد کالاهای خریداره شده و تعداد کالا های بازگشت به انبار ذخیره می شود.

ابتدا بر روی id سازمان ها پیمایش می شود و مقادیر ویژگی RETURNAMVALTOANBAR آنها مورد بررسی قرار می گیرد.

این ویژگی دارای چهار مقدار زیر است:

۱- مستعمل قابل بهره برداری
۲- مستعمل ملزم به تعمیر
۳- اسقاط قابل تفکیک
۴- اسقاط غیر قابل تفکیک

اگر کالایی دارای مقدار ۳ و یا ۴ باشد به این معناست که این کالا به انبار بازگشت داده شده است و اگر هرکدام از این ۴ مقدار را داشته باشد یعنی کالا بهرحال خریداری شده است.

بدین ترتیب هنگامی که بر روی ستون AD_ORG_REF_ID پیمایش انجام می شود، id هر سازمان در دیکشنری ذخیره می شود. هربار که توسط سازمان خریدی صورت بگیرد با بررسی مقدار RETURNAMVALTOANBAR ، value دیکشنری آن بروز رسانی می شود.

نتایج

با این روش تمام اطلاعات مورد نیاز در دیکشنری orgInfo ذخیره شده است. با پیمایش کلید های آن (idسازمان ها) به تعداد کالاهای خریداری شده و تعداد کالا های بازگشت به انبار آن دسترسی پیدا کرده و نسبت آن ها را نمایش می دهیم.

نمونه خروجی:

Organization Id	469732738
Returned Items Count	183
Purchased Item Count	864
returned-purchased Ratio	0.21

با استفاده از این نتایج می توان فهمید کدام سازمان ها مقدار بازگشت به انبار بیشتری دارند، علت آن را ریشه یابی کرده و برای جلوگیری از آن تمهیداتی صورت گیرد. به طور مثال سازمان 200000137 نرخ بازگشت به انبار بسیار زیادی دارد (0.89) و باید علت آن بررسی شود. درصورتی که در نمونه ی نشان داده شده این نرخ به نسبت مطلوب تر است.

سوال دوم: تحلیل ورود دارایی به خروج دارایی، به تفکیک زمان و معاونت

حل این سوال در دو بخش صورت گرفته است. تحلیل ورود به خروجی دارایی به تفکیک زمان و تفکیک آن بر اساس معاونت. بدین منظور دو کلاس ConsumerInfo برای ذخیره اطلاعات بهربردار(معاونت) و ItemInfo برای ذخیره اطلاعات هر نوع کالا ایجاد شده اند که در ادامه کارکرد هرکدام توضیح داده می شود. برای تحلیل این سوال از دیتاست INOUTLINE استفاده شده و پردازش بر روی ویژگی های M_PRODUCT_ID (شناسه کالا) ، CONSUMER (بهربردار) ، CREATED (تاریخ ایجاد و ورود دارایی) ، OPERATIONDATE (تاریخ بهره برداری و خروج دارایی) صورت گرفته است. در بخش نتایج علاوه بر تعداد ورود و خروجی کالا ها در هر سال، بیشترین کالای خریداری شده، بیشترین کالای بهره برداری شده، بهره برداری که بیشترین خرید را داشته و... نمایش داده می شود. همچنین نشان داده می شود هر بهره بردار به کدام کالا ها علاقه ی بیشتری داشته و میزان خرید آن در سال های مختلف به چه صورت بوده است.

توضیح فنی پیاده سازی

ابتدا در مرحله ی پیش پردازش، ویژگی های M_PRODUCT_ID ، CONSUMER ، CREATED ، OPERATIONDATE از دیتاست انتخاب می شود. از آنجایی که null بودن هریک اینها در تحلیلمان بی معنی است کالاهایی که دارای این مقدار باشند را دور میاندازیم. زیرا به طور مثال زمانی که تاریخ بهره برداری مشخص شده است، مشخص نبودن بهره بردار بی معنی است.

در مرحله اول به توضیح کلی کد برای هر کدام از بخش های صورت سوال پرداخته و سپس کلاس های ConsumerInfo و ItemInfo معرفی می شوند.

تحلیل ورود دارایی به خروج آن به تفکیک زمان

ابتدا دیکشنریی با نام itmes ایجاد می شود که کلید آن "سال" + "شناسه کالا" بوده و value آن یک شی از کلاس ItemInfo می باشد. با این روش اطلاعات هر کالا با توجه به سالی که خریداری شده ذخیره می شود. سپس سطر های دیتاست را پیمایش می کنیم. اگر تاریخ ورود یا خروج کالا به فرمت درستی نباشد (مثلا بجای تاریخ ، رشته ی بی معنی نوشته شده باشد) و یا تاریخ ورود کالا بعد از خروج آن باشد کالا دیگر به عنوان یک کالای معتبر پردازش نمی شود. تنها در صورت صحت این اطلاعات، مقادیر آن ها در متغیر ها ذخیره می شوند.

با پیمایش هر سطر اگر شناسه آن کالا + سال ورود آن در کلید های items باشد، اطلاعات ان کالا بروز رسانی شده و در غیر این صورت یک ابجکت ItemInfo جدید به items اضافه می شود.

تحلیل ورود دارایی به خروج آن به تفکیک معاونت

ابتدا دیکشنریی با نام consumers ایجاد می شود که کلید آن "شناسه ی بهره بردار" بوده و value آن یک شی از کلاس ConsumerInfo می باشد. با این روش اطلاعات کالاهای خریداری شده برای هر بهره بردار ذخیره می شود.

مشابه قسمت قبل تاریخ ها بررسی می شوند و اگر شناسه بهره بردار در کلید های consumers باشد، اطلاعات ان بهره بردار بروز رسانی شده و در غیر این صورت یک ابجکت ConsumerInfo جدید به consumers اضافه می شود.

کلاس ItemInfo

این کلاس اطلاعات کلی هر کالا در هر سال را ذخیره می کند و شامل ویژگی های شناسه ی کالا، تاریخ ورود و خروج آن، تعداد آن کالا (در هر سال)، لیست بهره بردارانی که این کالا را خریداری کرده اند (دیکشنریی شامل id و تعداد خرید آنها)، شناسه ی بهره برداری که این کالا را بیشتر از همه خریداری کرده و تعداد خرید آن است. همچنین دیکشنری هایی به صورت استاتیک تعریف شده اند که تعداد کل ورود و خروج کالا در سال و بیشترین کالای بهره برداری شده در هر سال را ذخیره کرده اند.

تابع update_item_info: هربار که سطر های دیتاست پیمایش می شود ، اگر شناسه ی این کالا + سال مورد بررسی در کلید های items موجود باشد یعنی کالا قبلا وجود داشته و این تابع فراخوانی می شود. در این صورت یا بهره بردار جدیدی کالا را خریداری کرده و یا یک بهره بردار مجددا در آن سال کالا را خریده است. در هر صورت شناسه ی بهره بردار با لیست بهره بردارانی که کالا را تابحال خریده اند مقایسه می شود. اگر شناسه وجود داشته باشد به تعداد خرید های آن یکی اضافه شده و در غیر ان صورت شناسه بهره بردار به دیکشنری consumers کالا اضافه می شود. سپس از طریق توابعی که در ادامه ذکر میشوند تعداد کل ورودی/خروجی کالا بروز رسانی می شود.

توابع update_input_per_year_counter و update_output_per_year_counter : این توابع مقادیر دیکشنری های inputPerYear و outputPerYear که به صورت استاتیک تعریف شده اند را بروز رسانی می کنند. کلید این دیکشنری ها سال ورود/خروج کالا بوده و value آن تعداد کالا می باشد. با هر بار فرخوانی این تابع سال مورد نظر در کلید ها پیدا شده و مقدار کالا ی آن یکی اضافه می شود. بدین ترتیب

تعداد ورودی ها و خروجی های کل کالا در سال شمرده می شود. در تابع دوم مقدار دیکشنری operatedItemPerYear هم بروز رسانی می شود. این دیکشنری تعداد خروج هر کالا در سال را می شمرد و در تابع get_top_operated_item_per_year کاربرد دارد.

تابع get_top_operated_item_per_year: این تابع به صورت استاتیک پیاده سازی شده است و با گرفتن یک سال بیشترین کالا ی بهره برداری شده در آن سال و تعدادش را در خروجی می دهد.

تابع update_top_consumer_per_year: این تابع بر روی دیکشنری consumers هر کالا پیمایش انجام می دهد. هر بهره بردار که بیشتر از بقیه کالا را خریداری کرده باشد در متغیر top_consumer کالا ذخیره شده و تعداد خرید آن در top_consumer_purchase_count ذخیره می شود. در توابع getter این دو متغیر ابتدا این تابع صدا زده می شود تا مقادیر top_consumer و top_consumer_purchase_count کالا را بروز رسانی کند.

کلاس ConsumerInfo

این کلاس اطلاعات کلی هر بهره بردار را ذخیره می کند و شامل ویژگی های شناسه بهره بردار، تعداد کل خرید ها، بیشترین کالای خریداری شده و تعداد آن، لیست کالاهای خریداری شده و تعداد آنها (به صورت دیکشنری) و لیست کالاهای بهربرداری شده و تعداد آنها است.

تابع update_consumer_info: هر بار که سطر هایی دیتاست پیمایش می شود، اگر شناسه ی بهره بردار در کلید ها consumers موجود باشد یعنی این بهره بردار قبلا وجود داشته و این تابع فراخوانی می شود. در این صورت اگر بهره بردار کالای جدیدی را خریداری کرده باشد شناسه ی آن کالا به دیکشنری purchasedItems بهره بردار اضافه می شود و اگر کالا را قبلا خریداری کرده باشد مقدارآن در دیکشنری یکی اضافه می شود. سپس مشابه کلاس ItemInfo تعداد کل خرید ها و ورودی/خروجی کل کالا های بهره بردار بروز رسانی می شود.

تابع get_most_purchased_item: این تابع بر روی دیکشنری purchasedItems هر بهره بردار پیمایش انجام می دهد. با مقایسه تعداد هر کالا، کالایی که توسط بهره بردار بیشتر از همه خریداری شده باشد (ورود) به عنوان mostPurchasedItem بهره بردارو تعداد خرید آن در mostPurchasedItemCount ذخیره می شود.

نتایج

همانطور که در تعاریف کلاس ها بیان شد با استفاده از توابع پیاده سازی شده و دیکشنری های استاتیک اطلاعات بدست آمده از کل کالا ها و بهره برداران نمایش داده می شود.

تابع print_inout_per_year: این تابع تمام اطلاعات گفته شده را چاپ می کند و برای بررسی بخش اول سوال استفاده می شود.

تابع print_purchased_info: این تابع یک سال را به عنوان ورودی می گیرد و با پیمایش دیکشنری items کالایی که بیشترین مقدار خرید (ورود) را داشته باشد چاپ می کند. همچنین بهره برداری که در آن سال بیشترین خرید را داشته باشد چاپ می شود.

تابع print_top_consumers: این تابع اطلاعات گفته شده برای بهره بردار را چاپ می کند و برای بررسی بخش دوم سوال استفاده می شود. برای بررسی راحتتر نتایج یک ورودی به نام min_purchase_count به آن داده می شود و اطلاعات بهره بردارانی که حداقل یک تعداد خاصی کالا خریداری کرده باشند نمایش داده می شود.

نمونه خروجی بخش اول (اعداد ذکر شده فقط برای سال 2018 هستند)

2018	
overall input	4486
overall output	4483
most purchased item id	104014836
most purchased item count	437
top consumer id	105003590
top consumer purchase count	130
most operated item id	104014836
most operated item count	437

با بررسی خروجی های این بخش متوجه می شویم در هر سال کدام کالا ها بیشترین طرفدار را داشته اند و کدام بهره برداران بیشترین خرید را کرده اند. به طور مثال کالا ی 104010517 در سال های 2015, 2019, 2021 بیشترین ورود و خروج را داشته است و این به معنای پرطرفدار بودن ان نسبت به کالا های دیگر است. همچنین مشاهده می شود در هر سال نرخ ورود تقریبا برابر با خروج کالا بوده و فروش خوبی داشته ایم. اگر در سالی تعداد کالا ی خارج شده بیشتر از کالا های وارد شده باشد بدین معناست که از کالا های وارد شده در سال های قبل بهره برداری شده است.

نمونه خروجی بخش دوم (اعداد ذکر شده فقط برای یک بهره بردار هستند)

consumer id	300002668
most purchased item	210035088
most purchased item count	104

output	input	year
90	90	2016
186	186	2017
22	22	2018
9	9	2019
7	7	2020
1	1	2021

با بررسی این نتایج میتوانیم متوجه شویم هر بهره بردار به کدام نوع کالا بیشتر علاقمند است. به طور مثال کالای300003322 بین سه بهره بردار 105003097 و 105003766 و 105003604 محبوب است زیرا بیشترین میزان ورود و خروج را از این کالا داشته اند. همچنین می توان فهمید هر بهره بردار چقدر تمایل همکاری با ما را دارد. به طور مثال در نمونه ی نشان داده شده تعداد ورود و خروج کالاها برای بهره بردار 300002668 طی سال ها کم شده است و این نشان دهنده ی عدم رضایت اوست. در صورتی که بهره بردار 300081261 طی سال های 2018 تا 2021 میزان خرید و فروش به نسبت خوبی داشته است.

لازم به ذکر است میزان کالاهای وارد و خارج شده به انبار بیشتر از مقادیرذکر شده می باشد. به دلیل وجود ایرادات در دیتاست و ناقص بودن اطلاعات برخی کالاها، پردازش بر روی آنها انجام نشده و در همان مرحله ی پیش پردازش حذف شده اند. نتایج نشان داده شده بر اساس تعداد کالاهای معتبر می باشد.

سوال سوم: وضعیت کالاهای مشابه، و پیش بینی قیمت هر کالا متناسب با خوشه آن

برای پاسخ به این سوال، باید یک مدل خوشه‌بندی پیشنهاد دهیم که دیتای ترین[1] را خوشه‌بندی کند، تا بتوانیم، در گام بعدی، دیتای تست را به وسیله خوشه های موجود، پیش‌بینی کنیم. در این بخش، چند مدل خوشه بندی، از جمله DBSCAN, K-means, OPTICS, Brich و HDBSCAN تست و نتایج آن ها بررسی شد، و در نهایت سه مدل K-means و DBSCAN و HDBSCAN که به نظر برای داده های ما مناسب تر بود، انتخاب شدند تا نتیجه خوشه‌بندی روی آن ها به طور جدی‌تر بررسی شود. در این گزارش، برای هر کدام از این مدل ها یک اسکریپت پایتون نوشته شد و نتایج را بر اساس پیاده‌سازی انجام شده بررسی کردیم. در میانه بررسی اسکریپت، چالش های پیش‌آمده نیز بررسی می‌شوند. با توجه به اینکه بهترین پیاده‌سازی، روی مدل مبتنی بر خوشه‌بندی K-means به‌دست آمد، تنها این مدل با جزئیات شرح داده خواهد شد، اما اسکریپت های دو مدل دیگر نیز در فایل های ارسالی قرار داده شده اند.

در گام بعدی، با استفاده از یک مدل رگرسیون خطی[2]، براساس قیمت ابتدایی و فعلی کالا، یک مدل رگرسیون خطی پیاده‌سازی می‌شود که قیمت آینده کالا را با تابع به‌دست آمده از این مدل می‌توان پیش‌بینی کرد.

توضیح فنی پیاده‌سازی، متمرکز بر مدل K-means[3]

برای خوشه‌بندی، در اولین گام نیاز به تشخیص ویژگی های مناسب برای پردازش داده ها بود. برای این کار، تمام تیبل های داده شده، به صورت دستی بررسی شدند. ایده اولیه انتخاب ویژگی ها، خوشه‌بندی بر اساس نوع و قیمت بود. دلیل این انتخاب، این بود که بتوان در مرحله پیش‌بینی نیز، با اتکا به اینکه قیمت در خوشه‌بندی نیز لحاظ شده، و همچنین این اصل بدیهی که کالاهای هم نوع را می‌توان تا حد خوبی، مشابه هم فرض کرد، به یک خوشه بندی با دقت مناسب برسیم. بررسی تیبل ها با این ایده اولیه، منجر به انتخاب سه ویژگی شد. ویژگی BOOKVALUE از تیبل productInstance ، و ویژگی های NAME و VALUE از تیبل products. ویژگی اول، طبق تعریف ارائه شده، ارزش روز یک کالا را بیان می‌کند. ویژگی دوم، یک رشته به زبان فارسی است که اسم یک کالا را نمایش می‌دهد. در مورد این ویژگی لازم به ذکر است که رشته های فارسی ارائه شده در دیتا، بعضا شامل الگوی خاصی در نام گذاری بودند که در بخش های بعدی توضیح بیشتری داده خواهد شد. ویژگی سوم نیز، شامل اعدادی است که متناظر با نوع کالا، بر اساس شباهت کالا ها، مقدار دارند. شاید بتوانید برای تقریب ذهنی، عملکرد این ستون را به عنوان یک بازنمایی(مشابه embedding vector) از رشته فارسی به عدد در نظر بگیرید، به این صورت که

[1] train

[2] Linear regression

[3] پیاده‌سازی نهایی در فایل "Kmeans - regression.py" انجام شده است.

کالاهایی با اسم مشابه، اعداد نزدیک به هم در ستون VALUE دارند. ویژگی M_PRODUCT_ID نیز از هر دو تیبل، به این دلیل انتخاب شد تا بتوان به عنوان کلید مرحله مرج، برای ادغام دو سری دیتا استفاده شود. با این توضیحات برای انتخاب ویژگی، وارد مرحله پیش پردازش می‌شویم.

با بررسی ویژگی ها NAME و VALUE، یک نکته جلب توجه کرد. مقادیری در ستون NAME وجود داشتند، که به فرمت:

اسم کالا + " . ."

یا به فرمت:

اسم کالا + " ."

بودند.

پس در دیتاست، یک حالت رخ می‌داد، مشابه شکل زیر:

پوشش یو پی وي سي	836950015
پوشش یو پی وي سي .	8.3695E+12
پوشش یو پی وي سي . .	8.3695E+16

که به ازای یک اسم مشابه، بدون تفاوت در دیگر ستون های رکورد مورد نظر، و فقط وجود یک پسوند شامل نقاط در اسم، تعدادی صفر به انتهای عدد بازنمایی در ستون VALUE، اضافه می‌شد. در اولین گام پیش پردازش دیتا، این صفر های اضافی از مقادیر رکورد های ستون VALUE حذف می‌شوند.

در خطوط ۳۲ تا ۶۷، این تبدیل روی رکورد هایی که به این فرمت هستند، اعمال می‌شود. با توجه به اینکه وجود دو نقطه، تعداد صفر بیشتری اضافه می‌کرد، برای پاکسازی، دو رشته رجیکس متفاوت استفاده کردیم تا بتوانیم براساس تعداد صفر های اضافه شده، تخمین مورد نظر را اعمال کنیم. بر اساس رجیکس مچ شده با رشته و دیتاتایپ تشخیص داشته شده(برخی مقادیر عددی، رشته تشخیص داده می‌شدند که باید به شکل رشته با آنها برخورد می‌شد) تبدیل روی مقادیر انجام می‌شود.

همچنین در حلقه یاد شده، در انتهای حلقه، یکبار دیتاها را به مقدار عددی تبدیل می‌کنیم(با کمک متد to_numeric). علت این کار، این است که همچنان برخی مقادیر در دیتاست، طول عددی زیادی دارند که باعث می‌شد، این اعداد به عنوان رشته در لایبرری های مورد استفاده پردازش شوند. با این تغییر فرمت، خواستیم تا این اشکال نیز برطرف شود(خطوط ۵۹ تا ۶۴). در خط ۶۷ نیز، یکبار کل ستون به فرمت float64 تبدیل می‌شود، تا از این بابت، دیگر هیچ مشکلی در پردازش مقادیر عددی نداشته باشیم.

گام بعد، ادغام دیتاهای دو تیبل است. با کمک ویژگی ID انتخاب شده، ستون های موجود را با یکدیگر مرج می‌کنیم. متد ادغام، inner در نظر گرفته شده تا فقط اشتراکات با یکدیگر ادغام شوند، و دیتای بلااستفاده جدید تولید نشود.(خط۶۹)

سپس، مقادیر null از دیتا حذف شده.(خط ۷۳)

در خط ۷۵، دو فیچر مورد نیاز برای پردازش در مدل خوشه‌بندی، VALUE و BOOKVALUE به عنوان دیتای نهایی انتخاب شده اند. این دیتاها را برای استفاده بهتر در لایبرری ها، به فرمت numpy تبدیل می‌کنیم.

در خط ۸۱، تابع train_test_split از کتابخانه sci-kit برای دسته بندی دیتا به دو بخش train و test استفاده می‌شود. ۷۰ درصد دیتای موجود، به عنوان train و ۳۰ درصد به عنوان test انتخاب می‌شود.

گام بعدی، اسکیل دیتا به رنج [0,1] است(خطوط ۸۵ تا ۸۷). تبدیل MaxAbsScaler نیز امتحان شد، که در اجرا تفاوت خاصی با روش فعلی(MinMaxScaler) نداشت. پس از اسکیل کردن، مرحله پیش‌پردازش داده ها به اتمام می‌رسد.

در این بخش، به توضیح مدل ساخته شده می‌پردازیم. برای خوشه بندی، در مقایسه بین ۳ مدل DBSCAN، HDBSCAN و K-means، مدل K-means خروجی بهتری داشت، به همین جهت انتخاب نهایی این مدل بود. چالش اصلی مدل K-means برای خوشه‌بندی، انتخاب تعداد بهینه خوشه ها است، به شکلی که خوشه های با کیفیت مناسبی تولید شود. برای حل این موضوع، با اجرای الگوریتم elbow، تعداد خوشه های مناسب را پیدا کردیم.

الگوریتم elbow، با اجرای k های مختلف و شبیه سازی خوشه‌بندی، سعی می‌کند تا بهترین k را انتخاب کند. معیار انتخاب، بهترین دقت کسب شده، به نسبت زمان اجراست. الگوریتم، خوشه‌بندی با k های مختلف را شبیه سازی می‌کند. سپس دقت خوشه‌بندی های به‌دست آمده را محاسبه می‌کند. نقطه elbow که در واقع، نقطه ای است که تغییرات بهبود دقت خوشه‌بندی، به نسبت زمان صرف شده بسیار کم می‌شود، نقطه مورد نظر ما است. مقدار k این نقطه، به عنوان تعداد خوشه های بهینه انتخاب می‌شود. به طور بدیهی، به نظر می‌رسد که شاید در نمودار، فقط یک نقطه به این شکل وجود داشته باشد، اما به طور تجربی مشخص می‌شود که ممکن است متناسب با توزیع دیتا، چند نقطه به این شکل داشته باشیم. به همین دلیل، باید متد elbow در چند رنج مختلف k اجرا شود، تا احتمال انتخاب k بهینه به طور گلوبال را افزایش دهیم. با توجه به این نکته، ما نیز در ساخت مدل خود، چند بار در رنج های مختلف k، الگوریتم را اجرا کردیم، و در انتها مقدار k=48 انتخاب شد. جدول زیر، نمایش حاصل چندبار اجرای متد elbow و مقادیر k به‌دست آمده، زمان تقریبی اجرا و امتیاز کسب شده است.

K = 7	K = 11	K = 34	K = 48	K = 79	K = 101	
0.576	0.173	0.018	0.008	0.003	0.002	Score
0.5	0.7	2.2	2.8	5.0	7.1	Run time

متد محاسبه امتیاز در این مقایسه، distortion است. محاسبات به این شکل انجام می‌شود که مجموع مربعات فاصله هر نقطه از مرکز انتصاب داده شده اش، محاسبه می‌شود. طبعا هرچقدر این مقدار کمتر باشد، یعنی خوشه های متمرکز تری ایجاد شده، پس عدد پایین‌تر، مطلوب‌تر است.

در مرحله بعد، مدل K-means را با پارامتر k=48 ترین می‌کنیم.

گام نهایی، مرحله تست است. در این بخش به طور مرسوم، دیتای تست را به مدل دادیم و با استفاده از متد predict، لیبل های اختصاص یافته را محاسبه کردیم.

نتایج خوشه‌بندی

برای بررسی دقت مدل، از متد امتیاز دهی Silhouette استفاده کردیم، که یک متد مرسوم در روش های یادگیری بدون ناظر است. این متد برای هر نمونه از دیتا، مطابق فرمول زیر، ضریب امتیاز آن خوشه را محاسبه می‌کند.

$$x = \frac{\beta - \alpha}{\max(\alpha, \beta)}$$

که α، فاصله درون خوشه ای برای هر نمونه دیتا و β، فاصله از نزدیک ترین خوشه به جز خوشه ای که نمونه در آن قرار دارد است. عدد نهایی، میانگین ضرایب تمامی نمونه ها است. این عدد، در رنج [-1,1] است. عدد ۱ نشان دهنده بهترین حالت خوشه‌بندی، و عدد ۱- ، نشان دهنده انتساب نمونه به خوشه اشتباه است. اعداد در محدوده ۰ نیز، بیانگر هم پوشانی بین خوشه ها است، که مطلوب نیست.

ضریب Silhouette را برای دیتای train و test محاسبه کردیم. نتایج به شکل زیر است.

Test	Train	
0.838	0.841	Silhouette score

با توجه به امتیازات، مدل ما به دقت میانگین بسیار خوب ۸۴ درصد روی نمونه train و ۸۳ درصد روی نمونه test دست یافت. همچنین به طور رندوم، ۱۰ اندیس از دیتای test تولید می‌شود که در خروجی مقدار لیبل پیش‌بینی شده برای آن‌ها را چاپ می‌کنیم.

همچنین، نتایج ویژال خوشه‌بندی، در ۳ فولدر جداگانه، با نام های Train Result، Test Predictions و Selected Points که به ترتیب، نتایج اجرای مدل برای دیتای train، دیتای test و لیبل نقاط منتخب است. تصویر زیر دقت مدل در یکی از اجراها است.

```
Silhouette Coefficient (train): 0.851
Silhouette Coefficient (test): 0.853
```

به این ترتیب موفق شدیم تا یک خوشه‌بندی بر اساس نوع کالا و قیمت کالا بسازیم که در ادامه موفق به پیش‌بینی لیبل روی دیتای تست با دقت مناسبی است.

توضیح فنی پیاده‌سازی مدل رگرسیون

در این بخش، روال منطقی کار به این شکل است که به ازای هر خوشه‌بندی تولید شده در مرحله قبل، می‌توان برای دیتاهای آن خوشه، یک مدل پیش‌بینی آموزش داد. یک داده تست به طور رندوم انتخاب می‌شود، با توجه به اینکه در کدام خوشه قرار دارد، دیتای آن خوشه به مدل رگرسیون داده می‌شود و در انتها، قیمت نمونه انتخابی پیش‌بینی می‌شود. ایده این است که اگر مدلی با داشتن قیمت اولیه و قیمت به‌روز کالا بسازیم، از تابع فیت شده این، مدل می‌توان به عنوان یک تابع پیش‌بینی، استفاده کرد. در واقع ما معادله(مدل پیاده سازی شده ما معادله درجه اول است) تابع رگرسیون به‌دست آمده را، به عنوان یک تابع رشد قیمت در نظر می‌گیریم.

در کد پیاده‌سازی شده، ابتدا یک نمونه از دیتای تست را به طور رندوم انتخاب کرده ایم.(خط ۱۴۷)

در خط ۱۵۲ تا ۱۵۶، از دیتای ترین، دیتا هایی که لیبل مشابه نمونه تست را دارند، انتخاب می‌کنیم. با این کار، تمام داده های ترین مربوط به خوشه مورد نظر را داریم. با کمک این داده ها، مدل رگرسیون را خواهیم ساخت.

در خطوط ۱۵۹ و ۱۶۰، پارامتر های x1 و y1 که بعدا ورودی مدل هستند را مشخص می‌کنیم.

در خطوط ۱۶۳ تا ۱۶۹، دیتای به‌دست آمده را نمایش می‌دهیم.

در خط ۱۷۷ و ۱۷۸، مدل پیش‌بینی را آموزش می‌دهیم. در انتها نیز(خط ۱۸۴)، قیمت را برای داده نمونه تست پیش‌بینی می‌کنیم. به طور مثال، در تصویر زیر، شیب تابع و مقدار پیش‌بینی شده قیمت آینده کالا، به ترتیب نمایش داده شده اند.

```
Coefficients:
     [1.34281287]
the predicted price is:  [1268182.13691664]
```

نتایج

در انتها توانستیم، با ترکیب روش خوشه‌بندی و رگرسیون، یک مدل پیش‌بینی قیمت بر اساس شباهت کالاها بسازیم. در این روش، از متد خوشه‌بندی K-means و مدل رگرسیون خطی ساده استفاده شد.