دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

درس تحلیل کلان داده ها

استاد درس جناب آقای دکتر چهرقانی

(تمرین دوم)

محسن عبادپور | ۴۰۰۱۳۱۰۸۰ | m.ebadpour@aut.ac.ir

نیمسال دوم سال تحصیلی ۱۴۰۲-۱۴۰۱

فهرست پاسخ ها

فهرست مطالب

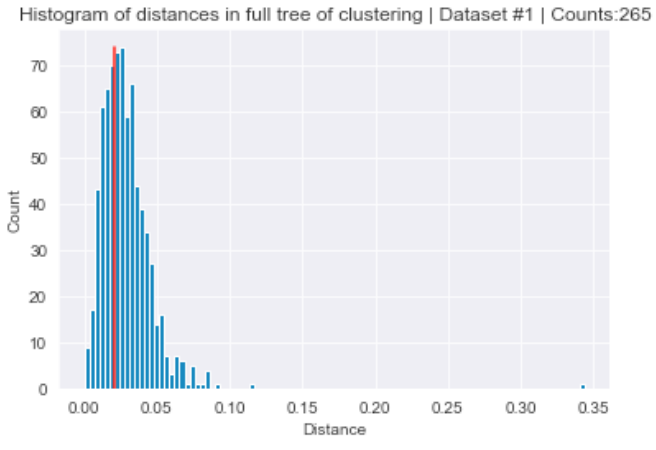
بخش اول: خوشه‌بندی CURE/BFR

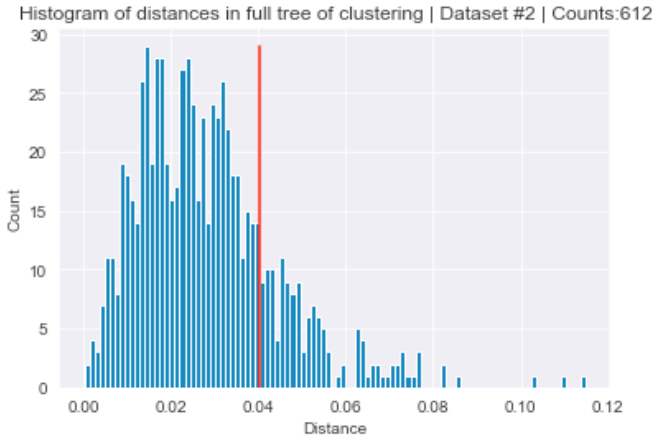
در این بخش پاسخ‌های مربوط به مبحث خوشه‌بندی آورده شده است که مشتمل بر شش سوال می‌باشد که در ادامه مورد بررسی و تحلیل قرار گرفته می‌شود.

سوال الف) پیاده‌سازی خوشه‌بندی اولیه

پیاده سازی با تابع مربوطه از کتابخانه sklearn انجام شده و بر اساس خوشه‌بندی سلسله مراتبی single-linkage در نظر گرفته شده است. طبق گفت و گوی تلگرامی، میتوانستیم تعداد نمونه‌ها را افزایش دهیم. تعداد کل نمونه‌های تولیدی را برای هر دو دیتاست به ۲۰۰۰ افزایش داده و ۷۵۰ نمونه را به عنوان داده اولیه انتخاب کردیم. سپس با سعی و خطا تعداد نقاط بازنمایی(representation points) برای دیتاست اول ۳۰٪ از ۷۵۰ نمونه (۲۶۵ نمونه) و برای دیتاست دوم ۸۰٪ از ۷۵۰ نمونه (۶۱۲ نمونه) انتخاب گردید. لازم به ذکر است در ابتدا همه‌ی داده‌ها shuffle شده و سپس ۷۵۰ نمونه از هر دیتاست انتخاب شده است. در ادامه نمودار فراوانی فاصله‌ی اقلیدسی بین نقاط قابل مشاهده است. (نزدیک‌ترین نمونه به هر نمونه در چه فاصله‌ی اقلیدسی قرار دارد.)

برای رسم این نمودار ابتدا درخت کامل خوشه‌بندی را ایجاد کرده ایم و خط قرمز نیز حدآستانه انتخاب نقاط بوده است. از یکنواخت/نرمال نبودن نمودار دوم نسبت به نمودار اول قابل انتظار است که قرارگیری نقاط در دیتاست دوم چالشی بوده و نیازمند نقاط بیشتری برای تعداد نقاط بازنمایی می باشد تا خوشه بندی درست انجام گیرد. و طبق نمودار بهتر است جایی باشد که توزیع فاصله‌ی نقاط نزول کرده باشد.





سوال ب) رسم نقاط بازنمایی

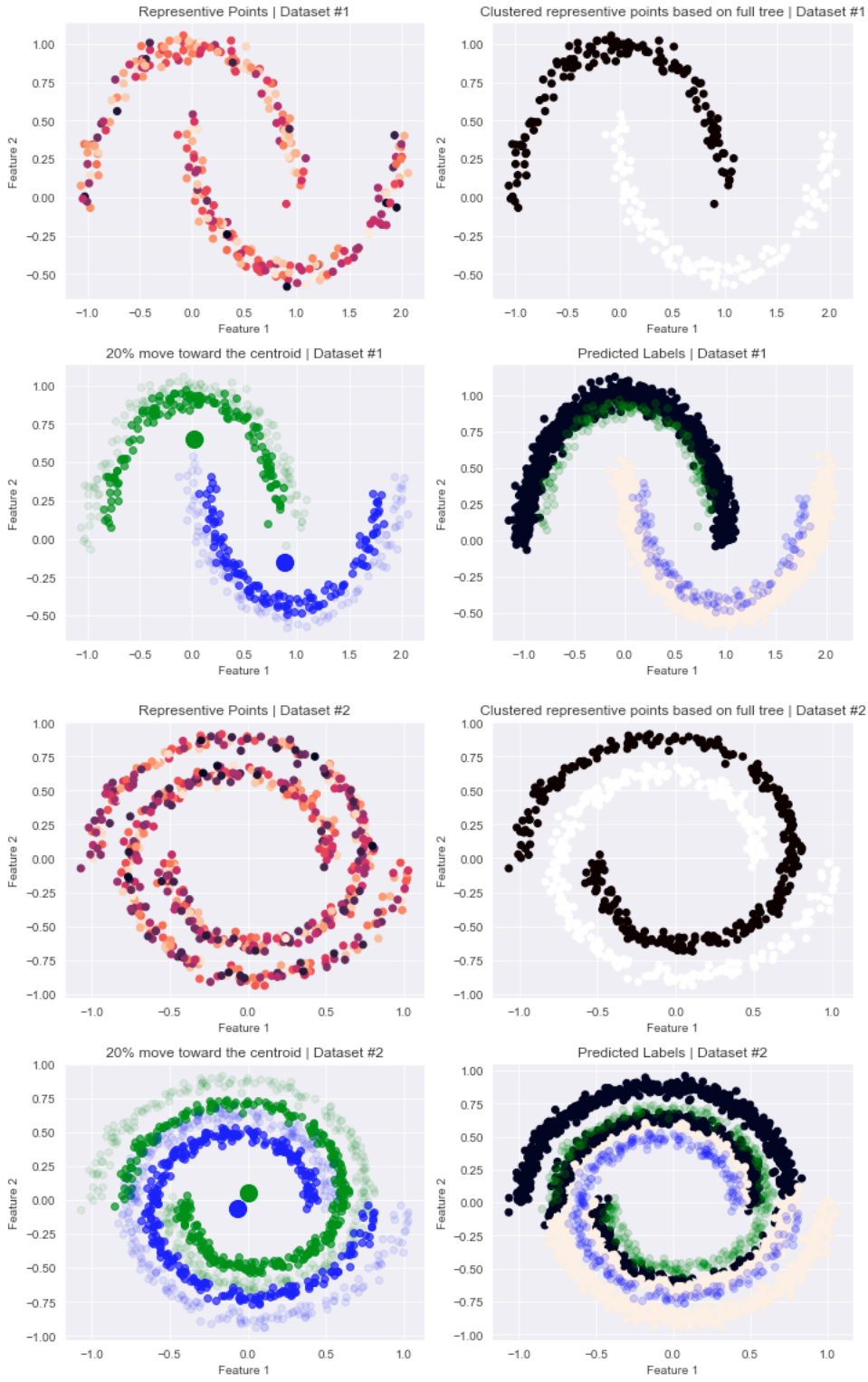
رسم نقاط بازنمایی اولیه، روند حرکت ۲۰ درصدی به سمت centroid و خود centroid های خوشه‌ها و نقاط نهایی در ادامه و همراه قسمت (ج) آورده شده است.

سوال ج) فاز دوم الگوریتم CURE

فاز دوم الگوریتم CURE مبتنی اعمال خوشه‌بندی بر روی کل داده‌ها و در مرتبه O(N) و با یکبار اسکن می باشد که در آن به ازای هر نمونه مواجه شده‌ی p، فاصله‌ی اش با نقاط بازنمایی سنجیده می شود و به خوشه‌ی متعلق به نزدیک ترین نقطه‌ی بازنمایی تخصیص پیدا میکند. در ادامه خروجی های مقتضی آورده شده است. شکل اول نقاط انتخابی بر اساس درخت کامل خوشه‌بندی و فاصله‌ی نقاط با هم دیگر می‌باشد. شکل دوم نتیجه خوشه‌بندی نقاط انتخابی می باشد که نشان می‌دهد خوشه بندی در هر دو دیتاست بدرستی انجام شده است. شکل سوم مربوط نقاط بازنمایی هر خوشه، centroid هر خوشه، و حرکت ۲۰۰ درصدی نقاط به سمت آن می باشد(نقاط کمرنگ نشان‌دهنده‌ی محل اولیه نقاط و محل پررنگ نشان‌دهنده‌ی محل نقاط پس از ۲۰ درصد حرکت می‌باشد).

شکل سوم چهارم نیز خوشه بندی کل داده‌ها توسط نقاط بازنمایی می باشد. در این نمودار نقاط بازنمایی نیز رسم شده است تا بتوان مقایسه‌های لازم را انجام داد. همانطور که میبینیم در دیتاست اول خوشه‌بندی با تقریب خیلی خوبی درست انجام شده است اما در دیتاست دوم خوشه‌بندی تقریبا با شکست مواجه شده است و دلیل آن شکل و توزیع داده‌ها می‌باشد که در هم تنیده است. این مشکل زمانی تاثیر خود را بیشتر میگذارد که ما نقاط بازنمایی را ۲۰ درصد به سمت centroid حرکت میدهیم؛ از جایی که مراکز خوشه ها خیلی نزدیک هم و تقریبا روی هم هستند، وقتی حرکت نقاط بازنمایی اتفاق

می افتد محل نقاط بازنمایی با هم ادغام می‌شود و خوشه بندی با شکست تقریبی مواجه می شود چرا که نقاط داخلی کمتر به مرکز حرکت میکنند(۲۰ درصد از یک فاصله‌ی کم) و نقاط خارجی بیشتر به داخل حرکت میکنند(۲۰ درصد از یک فاصله‌ی زیاد) و در نتیجه نقاط بازنمایی خوشه های مختلف روی هم می افتند (در دیتاست اول اینگونه نبوده و centroid ها و حرکت نقاط بازنمایی متداخل نبوده اند و از هم مجزا است).



سوال د) الگوریتم BFR و مقایسه‌ی آن

الگوریتم BFR یک نسخه‌ای از الگوریتم خوشه‌بندی K-Means می‌باشد که هدف آن مقابله با داده‌های بسیار حجیم و کلان می‌باشد. فرضی که در الگوریتم BFR اتخاذ میشود این است که توزیع داده‌ها نرمال بوده و حول یک میانگین در یک فضای اقلیدسی انجام شده است که توزیع هر خوشه نیز میتواند انحراف معیار یا بعد های مختلف داشته باشد.

راه حل قطعی برای این موضوع ذخیره داده‌ها می باشد که مرتبه ذخیره سازی آن O(data) بوده و خارج از منابع می‌باشد؛ ایده‌ی BFR ذخیره سازی صرفا تعدادی نمونه و نقاط اندک به همراه برخی از ویژگی‌های آماری از نقاط می باشد. روند کلی الگوریتم بدین گونه است که ابتدا K تا centroid اولیه تعیین می‌شود و سپس یک دسته (bag/batch) از نمونه‌ها را در حافظه لود کرده و آنان را با در نظر گرفتن یک حد آستانه‌ی فاصله‌ی اقلیدسی به خوشه‌های کنونی تخصیص می‌دهیم.

نقاطی که به خاطر حد آستانه در هیچ خوشه‌ای قرار نگرفته اند، ترکیب شان برای ایجاد یک گروه جدید بررسی می شود و این ادغام تا زمانی که همه‌ی نقاط مورد بررسی قرار گیرند، تکرار می‌شود؛ لذا در نتیجه میتوان نقاط مورد نظر را در سه دسته قرار DS، CS و RS قرار داد که DS نقاطی هستند که بقدر کافی برای قرار گرفتن در یک خوشه نزدیک بوده و میتوانند با یک خوشه حفظ و نگهداری شود(با حفظ و بروزرسانی ویژگی های آماری) و نیاز به نگهداری خود نمونه در حافظ نیست. CS نقاطی هستند که به یکدیگر بقدر کافی به هم نزدیک هستند اما از خوشه‌های موجود دور هستند که این نقاط نیز میتوانند حفظ و نگهداری شوند (حفظ با ویژگی های آماری). نقاط RS نیز نقاطی هستند که ایزوله و تنها بوده و نمیتوان آن را به هیچ خوشه یا گروهی تخصیص داد.

برای نگهداری، بارگذاری، بازنمایی و بروزرسانی K خوشه‌های موجود و گروه های نقاط (CS) نیاز نیست کل نمونه های هر یک ذخیره شود(ایده مقابله با مصرف حافظه‌ی بالا) و صرفا لازم است ویژگی های آماری نظیر مجموع، فاصله، تعداد نقاط و... برای هر یک نگهداری شود و با افزوده شدن نمونه جدید، این آمارگان آپدیت می‌شود و زمانی که بخواهیم ببینیم که نمونه جدید به اینان تعلق میگیرد یا نه، صرفا کافی است با این آمارگان تصمیم بگیریم که بقدر کافی نزدیک بوده و اختصاص پیدا میکند یا خیر و به این صورت خوشه بندی عادی را ادامه میدهیم.

مزیتی که BFR نسبت به CURE دارد این است که به خاطر حفظ ویژگی‌های آماری به جای خود نمونه‌ها، نیازمند حافظه کمتری بوده و محاسبات کمتری دارد و میتواند در مقابله با داده‌های حجیم عملکرد بهتری داشته باشد و با سرعت‌تر ظاهر شود و همچنین چون در مرحله‌ی تخصیص نمونه به گروه و خوشه ها یک حد آستانه فاصله قرار دارد، در مقابله با داده های پرت و نویزی عملکرد بهتری دارد اما مشکل و ضعف بزرگی که دارد فرض الگوریتم است که توزیع و شکل خوشه‌ها بصورت نرمال است که در واقعیت اینگونه نبوده و خوشه‌ها میتوانند شکل های متفاوت داشته باشد در نتیجه این الگوریتم در خوشه بندی نمونه‌هایی که توزیع نرمال نداشته باشند با شکست مواجه می‌شود(مانند دیتاست دوم).

در مقابل الگوریتم CURE بخاطر حفظ و نگهداری نقاط representation و محاسبه فاصله اینان با نمونه ورودی، نیازمند حافظه و محاسبات بیشتری بوده و کندتر است اما مزیتی که نسبت به BFR دارد این است که فرضی نسبت به توزیع و شکل خوشه ها نداشته و خوشه‌ها میتوانند شکل های متفاوت داشته باشند و خوشه‌بندی در فضای ویژگی با قدرت و دقت بالاتری انجام گیرد.

سوال ه) امکان اعمال الگوریتم BFR بر دیتاست های کنونی

خیر، به نظر من امکان استفاده مناسب و اعمال خوشه‌بندی BFR در دیتاست های کنونی وجود ندارد چرا که شکل و توزیع هر خوشه بصورت نرمال نبوده و فرض خوشه‌بندی BFR مبنی بر نرمال بودن در اینجا به هیچ عنوان صحت ندارد و در صورت اعمال خطای زیادی وجود خواهد داشت.

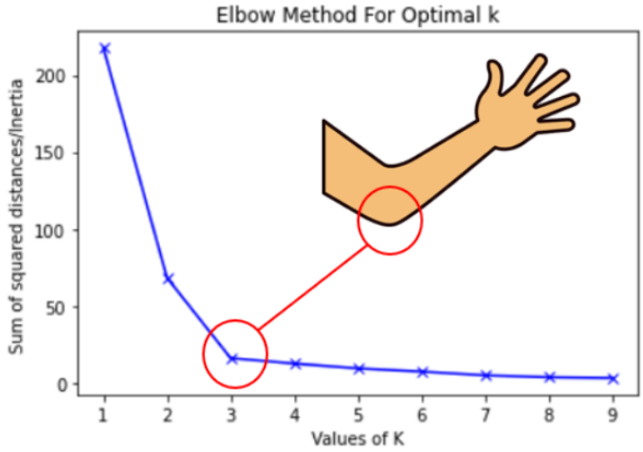
سوال ی) متد Elbow

متد Elbow یک روش معمول استفاده شده در خوشه‌بندی با الگوریتم K-means است. هدف از استفاده از این روش انتخاب تعداد بهینه خوشه‌ها است که به عنوان یک هایپرپارامتر در الگوریتم K-Means بایستی مشخص و تعیین شود.

انتخاب تعداد خوشه‌ها مناسب و بهینه برای مجموعه داده می‌تواند چالشی و سخت باشد و می‌تواند تأثیر قابل توجهی در کیفیت خوشه‌بندی و فضای مسئله داشته باشد. برای انتخاب بهینه تعداد خوشه‌ها، متد Elbow مورد استفاده قرار می‌گیرد که روشی تجربی بوده و بایستی بروی هر دیتاست جداگانه اعمال شود تا تعداد بهینه‌ی K بدست آید. روش اجرای متد

Elbow به صورت این صورت است که ابتدا برای تعداد خوشه‌های مختلف (مثلاً از یک تا ۲۰)، الگوریتم K-means را اجرا می‌کنیم و سپس برای هر تعداد خوشه، خطای Sum of Squared Error (SSE) را بین نقاط و مرکز خوشه‌ها را محاسبه کرده و نمودار SSE را بر حسب تعداد خوشه‌ها رسم می‌کنیم(البته لازم به ذکر است که از سایر معیار های سنجش کیفیت خوشه بندی نیز میتوانیم استفاده کنیم).

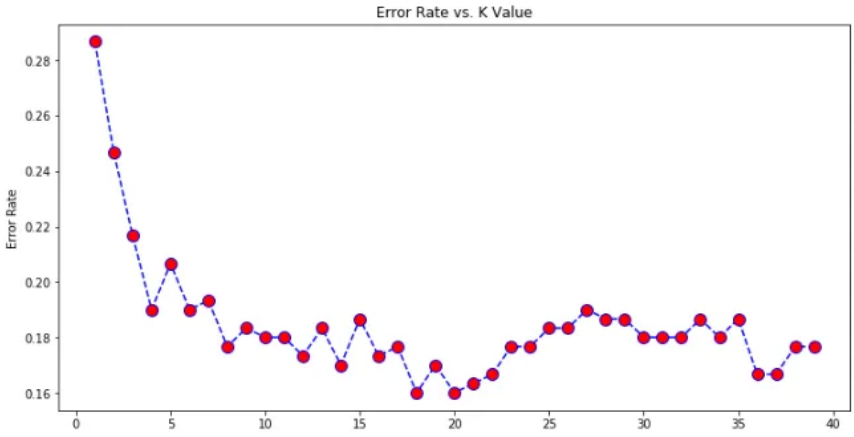
در نمودار، هر چه تعداد خوشه‌ها بیشتر می‌شود، SSE کاهش می‌یابد چرا که هر خوشه کمترین مجموع فاصله را خواهد داشت؛ اما به مرور اضافه شدن تعداد خوشه‌ها، مقدار SSE کاهش یافته ولی شدت کاهش آن کمتر می‌شود(شیب کاهش کمتر می شود). حال بر اساس نمودار هدف پیدا کردن تعداد خوشه ای است که پس از آن، کاهش SSE محسوس نبوده و شیب آن بسیار کمتر است. این نقطه به عنوان تعداد خوشه بهینه انتخاب می شود. نمونه‌ای از نمودارهای مذکور در زیر آورده شده است.



Line plot between K and inertia

بصورت کلی و به عنوان یک رویکرد ابتدایی میتوانیم از متد Elbow در روش دسته‌بندی K-NN نیز استفاده کنیم(لینک) و K مناسب برای تعداد نزدیکترین همسایگی مناسب برای دسته بندی را بدست آوریم(محل کمترین نرخ خطای دسته‌بندی) اما بایستی توجه کرد که در روند مشابه و برای K-NN نمودار Elbow همواره کاهشی نبوده و با نوسان های متعدد همراه می‌باشد و شکل نمودار متفاوت است و صرفا رسم این نمودار و پیدا گرفتن اولین نقطه که پس از آن خطا با شیب کمتری کاهش پیدا میکند لزوما تعداد K بهینه را برای مجموعه داده‌ی آموزشی فراهم نمیسازد و استفاده از این رویکرد صرفا یک دید تجربی و مفید اولیه به فرد می دهد و به او کمک می کند. برای مثال در شکل زیر، همانطور

که میبینیم روند خطا همواره کاهشی نبوده و با نوسان همراه است و اگر بخواهیم اولین نقطه که شیب خطا کاهش پیدا میکند را انتخاب کنیم باید k=4 را در نظر بگیریم اما ملاحظه میکنیم که K=20 برای مجموعه داده‌ی آموزشی بهینه است.



بخش دوم: الگوریتم Bloom Filter

در این بخش پاسخ‌های مربوط به مبحث Bloom Filter آورده شده است که مشتمل بر شش زیربخش می‌باشد؛ تمام قسمت های پیاده‌سازی با زبان برنامه نویسی پایتون انجام شده است.

سوال الف) مناسب بودن الگوریتم Bloom Filter برای انتخاب نام‌کاربری

بصورت کلی هدف مسئله در فروشگاه های آنلاین و ثبت نام کاربران جدید این است که نام‌کاربری تکراری به کاربران جدید اختصاص پیدا نکند چرا که نماد هویتی انان است ولی از طرفی میدانیم برای جواب قطعی کاملا درست برای وجود یا عدم وجود یک نام کاربری بخصوص، بایستی کل نام کاربری ها بررسی شود که مرتبه O(N) است که در آن N تعداد کل کاربران تا کنون است. الگوریتم Bloom-Filter با تضمین اینکه نرخ false negative صفر است این اطمینان را میدهد که اگر نام کاربری در پایگاه داده‌ی ما موجود باشد آن را استخراج کرده و بدرستی به ما میگوید که قبلا این نام کاربری ایجاد شده است اما در مقابل خطای false positive دارد به این معنا که ممکن است نام کاربری در پایگاه داده موجود نباشد اما خروجی الگوریتم این باشد که قبلا این نام کاربری ایجاد شده و موجود است. از جایی که برای ما false negative مهم است و نمیخواهیم یک نام کاربری که نماد هویتی است به دو نفر تعلق داشته باشد، استفاده از bloom filter مناسب است. خطای false positive در همین الگوریتم را نیز میتوان تا حد زیادی کاهش داد هر چند وجود این خطا حیاتی نمی‌باشد و قابل قبول است چرا که کاربر میتواند نام کاربری انتخابی در صفحه ثبت نام را براحتی عوض کرده و رشته دیگری را انتخاب کند. طبیعتا تعویض یک رشته دلخواه نام کاربری برای کاربر در زمان ثبت نام در مقابل بحث هویتی هیچ اهمیتی ندارد.

سوال ب) پیاده‌سازی تابع هش نوع اول

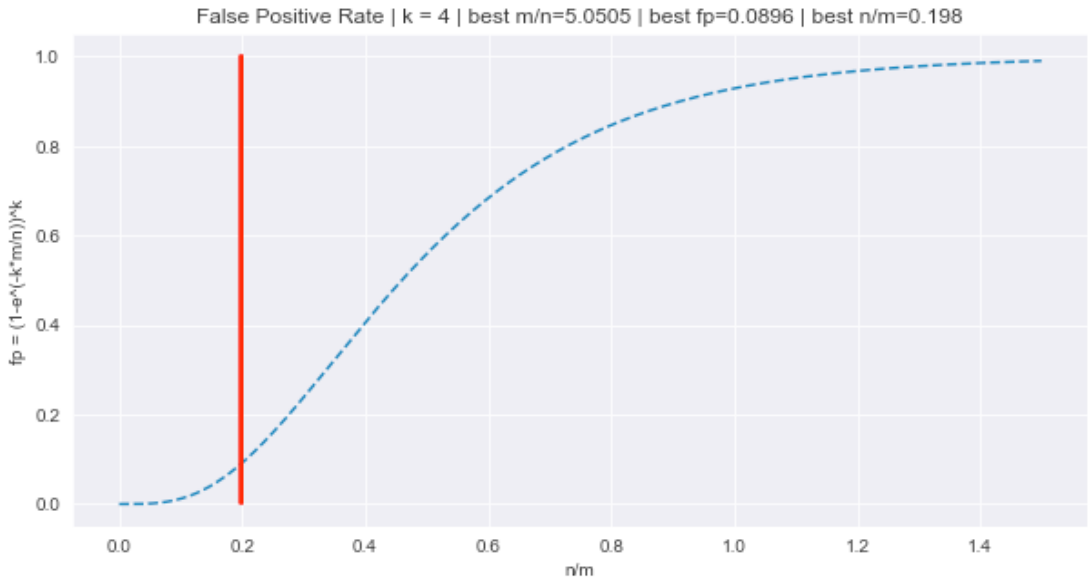
برای پیاده‌سازی این بخش و بخش های بعدی بصورت منظم تعدادی توابع توسعه داده شده است. تابع IHash برای هش نوع اول پیاده سازی شده است که رشته مورد نظر، آرایه‌ای مشتمل بر p های مفروض و M (طول آرایه‌ی بیتی) را دریافت کرده و یک لیستی از اندیس‌ها را بازمیگرداند که بایستی در آرایه‌ی بیتی آن اندیس ها به ۱ تغییر وضعیت داده

شود. فرآیند تعریف و بروزرسانی آرایه بیت‌ها در تابعی با عنوان ApplyHash انجام می شود که یک آرایه بیتی با نام BI و به طول M ساخته می‌شود و با اسکن خطی دیتاست و لود کردن نام کاربران و با استفاده از تابع IHash، آرایه‌ی بیتی BI بروزرسانی می شود و در نهایت به عنوان یکی از خروجی‌ها بازگردانده می‌شود.

برای تعیین اندازه جدول هش که چند برابر اندازه دیتاست باشد از رابطه موجود در اسلاید های درسی که در زیر نیز آورده شده است استفاده می‌کنیم:

$$\text{false positive probability} = (1 - e^{-km/n})^k$$

هم میتوانیم این رابطه را بصورت معادله و دستی حل کنیم و نسبت را بدست آوریم و هم میتوانیم نمودار نرخ false positive را از این معادله بر حسب m/n (یا n/m) که همان نسبت مورد نظر ماست رسم کنیم و وقتی که خطا برابر با ۰.۰۹ با تقریب ۰.۰۰۲ شد، نسبت را استخراج کنیم. بنده از رویکرد دوم استفاده کرده و این نمودار را رسم کردم.



نتیجه حاصل تقریبا برابر با پنج حاصل شد به این معنا که اگر سایز جدول هش یا همان آرایه‌ی بیتی را برابر با پنج برابر اندازه دیتاست قرار دهیم به خطای false positive تقریبا ۰.۰۹ خواهیم رسید.(به عبارتی دیگر بایستی دیتاست ۰.۲ برابر جدول هش یا همان آرایه بیتی مان باشد)

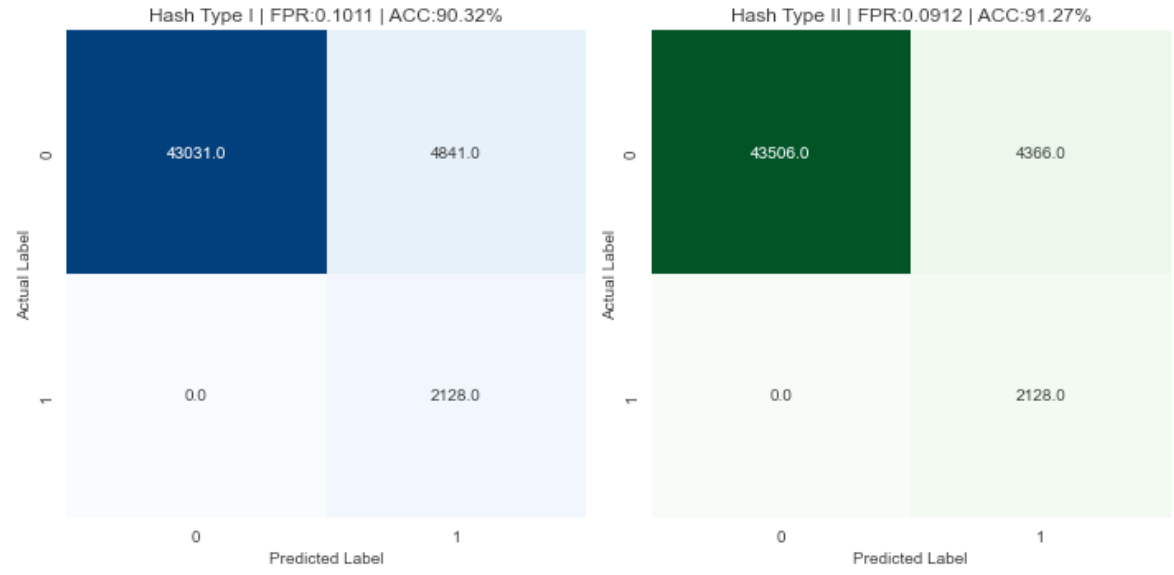
سوال ج) پیاده‌سازی تابع هش نوع دوم

تابع IIHash برای هش نوع دوم پیاده سازی شده است که رشته مورد نظر، آرایه‌ای مشتمل بر p های مفروض و M (طول آرایه‌ی بیتی) را دریافت کرده و یک لیستی از اندیس‌ها را بازمیگرداند که بایستی در آرایه‌ی بیتی آن اندیس ها به ۱ تغییر وضعیت داده شود. فرآیند تعریف و بروزرسانی آرایه بیت‌ها همانند قسمت قبل در تابع ApplyHash انجام می‌شود که یک آرایه بیتی با نام BII و به طول M ساخته می‌شود و با اسکن خطی دیتاست و لود کردن نام کاربران و با استفاده از تابع IIHash، آرایه‌ی بیتی BII بروزرسانی می شود و در نهایت به عنوان یکی از خروجی‌ها بازگردانده می‌شود.

سوال د) و سوال ه) سنجش و ارزیابی توابع هش نوع اول و دوم

تابع Measure برای ارزیابی و سنجش توابع هش پیاده سازی شده است. این تابع آرایه های بیتی بدست آمده در قسمت‌های قبل(BII و BI) را به همراه طول اندازه جدول هش(M) ورودی گرفته و شروع به اسکن مجموعه داده‌ی user_requests می کند. در مقابله با هر رشته نام کاربری، به ازای هر تابع هش، ابتدا اندیس‌های متناظر دریافت می‌شود و سپس در آرایه بیتی مقتضی بررسی می شود که آیا مقدار همه اندیس‌ها یک است یا خیر. اگر همگی یک بود یعنی این نام کاربری قبلا وارد شده است و (۱) پیش‌بینی می شود و در غیر این صورت یعنی نام کاربری آزاد است(۰).

برای هر تابع هش، فرآیند فوق بصورت همزمان انجام شده و پیش‌بینی ها حاصل می شود. بر اساس پیش‌بینی‌ها، ماتریس درهمریختگی (confusion matrix)، دقت (accuracy) و نرخ FPR حاصل شده و در زیر قابل مشاهده است:



سوال ی) مقایسه و بررسی مقدار FPR

مقدار FPR حاصل شده برای توابع هش نوع اول و دوم در قسمت قبل و در عنوان نمودار ماتریس‌های درهم ریختگی درج شده است. مقدار FPR مورد انتظار که ۰.۰۹ با تقریب ۰.۰۰۲ بود، برای تابع هش اول تطابقت ندارد چرا که برای آن مقدار ۰.۱۰۱۱ حاصل شده است که کمی بیشتر از حد انتظار است اما برای تابع هش دوم مقدار FPR حاصل با FPR مورد انتظار تطابقت داشته و برابر با ۰.۰۹۱۲ حاصل شده است.

دلیل عدم تطابقت در تابع هش اول را میتوان ماهیت خود تابع معرفی شده دانست چرا که این تابع هش هیچ الگوریتم یا رویکردی در خصوص اعمال ارزش جایگاه کاراکتر های یک رشته در هش نداشته و جابه‌جایی جایگاهی کاراکترهای یک رشته تاثیری در هش آن ندارد و این ضعف تابع هش است چرا که بنابر مسئله تغییر ترتیب کاراکترها میتواند نام کاربری دیگری ایجاد کند و این بایستی در هش کردن مد نظر قرار گرفته شود و این دقیقا موردی است که در تابع هش اول در نظر گرفته نشده اما در هش نوع دوم جایگاه ارزشی هر کاراکتر اعمال شده و در نتیجه از قدرت بالاتری نسبت به هش نوع اول در این مسئله برخوردار شده است و توانسته به حد قابل انتظار دست یابد. به عبارتی دیگر مجموع گرفتن، مینیمم یا ماکسیمم گرفتن و ضرب کردن یک مجموعه از اعداد که در تابع هش اول انجام می شود مستقل از ترتیب و جایگاه آنان است که باعث ضعف در تابع هش نوع اول و در نتیجه عدم تطابقت FPR حاصل با FPR مورد انتظار می‌شود. در زیر مثالی از عملکرد توابع هش به ازای جا به جایی کاراکترها آورده شده است:

```
1  print("------- Hash Type I --------")
2  print("mohammadAli",IHash("mohammadAli",M=1000))
3  print("Alimohammad",IHash("Alimohammad",M=1000))
4  print("------- Hash Type II --------")
5  print("mohammadAli",IIHash("mohammadAli",M=1000))
6  print("Alimohammad",IIHash("Alimohammad",M=1000))
7  print("---------------")
```

✓ 0.0s

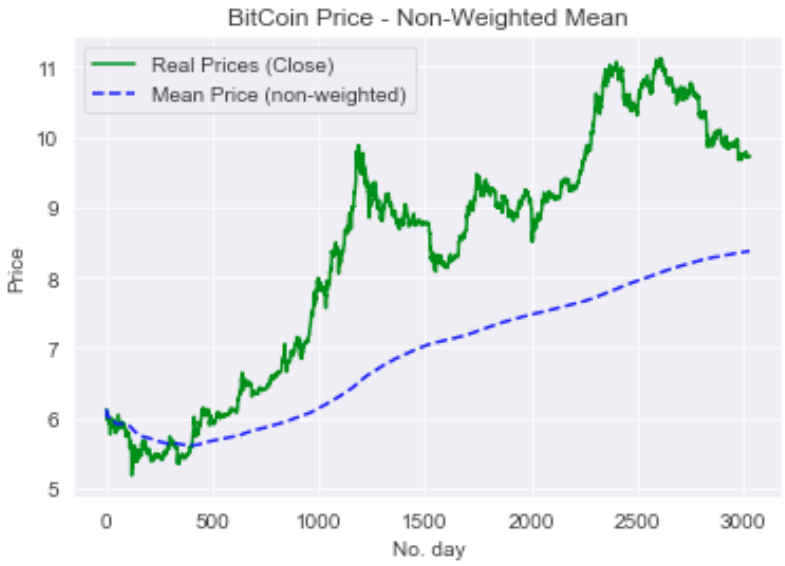
```
------- Hash Type I --------
mohammadAli [816 152 568 792]
Alimohammad [816 152 568 792]
------- Hash Type II --------
mohammadAli [794 864 544 214]
Alimohammad [450 484 740 654]
---------------
```

بخش سوم: Data Stream

در این بخش پاسخ‌های مربوط به مبحث Data Stream آورده شده است که مشتمل بر کار با لگاریتم قیمت close کندل‌های معاملاتی بیت‌کوین می‌باشد.

سوال الف) میانگین قیمت در طول زمان

برای این قسمت صرفا از دو متغیر استفاده شده است. یکی از متغیر ها تعداد کندل‌های گذشته را ذخیره کرده و متغیر دیگر مجموع قیمت‌های کندل‌های گذشته را تا کنون ذخیره می‌کند. حال دیتاست را بصورت خطی اسکن کرده و این دو متغیر را با هر قیمت جدید بروز میکنیم و میانگین جدید را با تقسیم مجموع جدید بر تعداد کل بدست می‌آوریم. در این میانگین قیمت روز های مختلف تاثیر یکسانی در قیمت میانگین داشته و میانگین حاصل بصورت غیروزن‌دار می باشد. نمودار میانگین بصورت شکل زیر حاصل می‌شود:



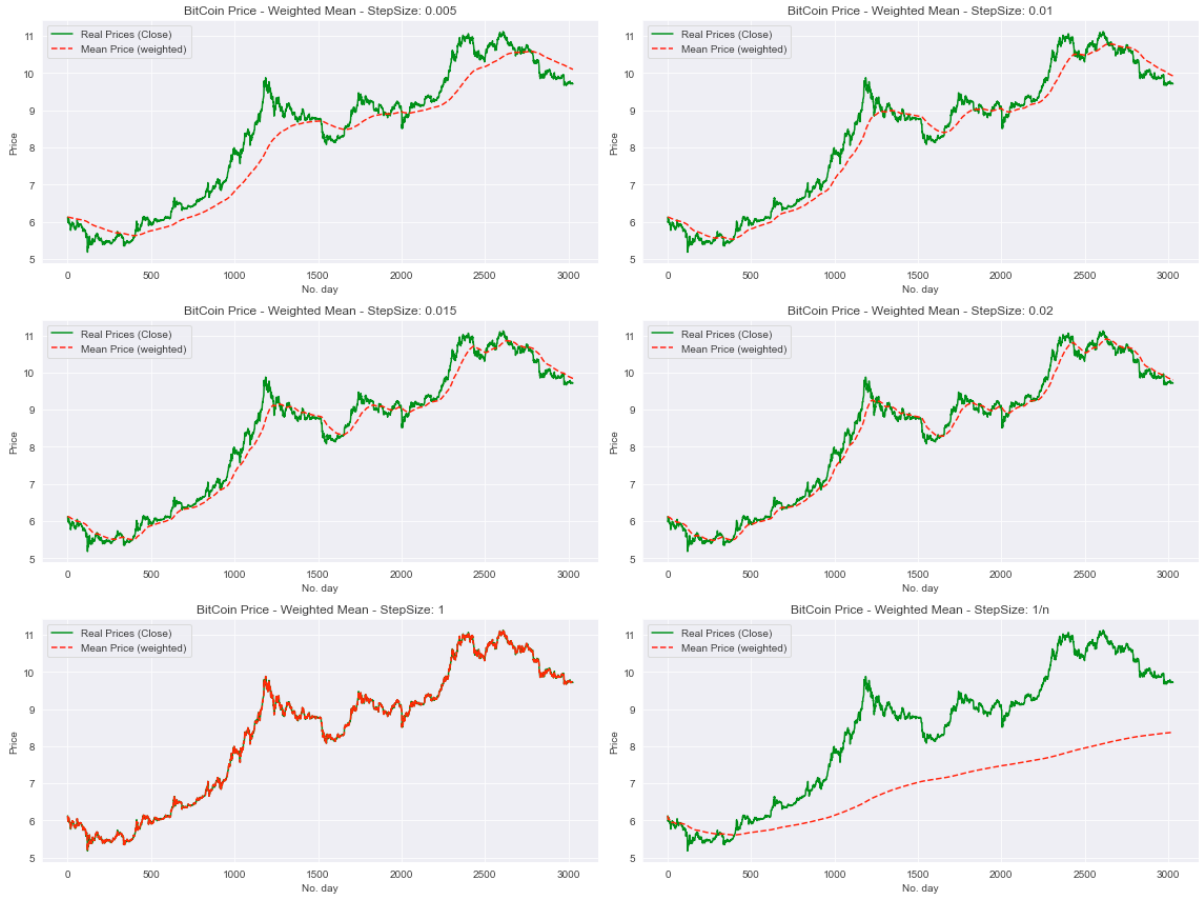
سوال ب) میانگین وزن‌دار

همانطور که در متن سوال نیز گفته شده است، رویکرد اول مبنی بر ذخیره‌ی c قیمت اخیر در یک لیست با طول ثابت c مناسب نبوده و دارای چالش‌های متعدد نظیر تعیین طول لیست و ضریب تاثیر قیمت هر روز از آنان است که پاسخگویی

به هر یک نیازمند مطالعه یا تخمین های متفاوت از جریان داده و دامنه‌ی مسئله است. از این رو، از رویکرد دوم معمولا استفاده می‌شود که در آن میانگین وزن‌دار اعمال می‌شود و این مورد معمولا به عنوان exponential moving average اطلاق می‌شود که طبق رابطه زیر می‌توان محاسبه نمود که در آن alpha همان stepsize است:

MeanToNow = new_price*(alpha) + MeanToNow *(1-alpha)

طبق رابطه‌ی فوق، قابل مشاهده است که اگر alpha را در هر مرحله برابر با 1/n قرار دهیم که n تعداد قیمت ورودی تا بدین لحظه است، میانگین وزن‌دار ما همانند میانگین‌غیروزن‌دار قسمت الف حاصل خواهد شد و اگر alpha را برابر با ۱ قرار دهیم، خود نمودار قیمتی حاصل خواهد شد(هر قیمت ورودی جدید خود میانگین جدید می‌شود) و برای بدست آوردن نتایجی مانند شکل دو(در متن سوال) که نمودار میانگین قیمتی بصورت smooth می‌باشد بایستی alpha را بین ۱ و 1/n تعیین کنیم. در زیر تعدادی رسم آورده شده است که قابل مشاهده است که مقادیر تجربی ۰.۰۱ الی ۰.۰۲ مناسب است:



سوال ج) شناسایی تغییر توزیع در جریان داده

همانطور که در متن سوال اشاره شده است، می‌توان از تغییرات واریانس و میانگین آن در طول ورود جریان داده استفاده کرده و رخداد تغییرات را در توزیع شناسایی کرده و هشدار تغییر داد. برای این منظور پیاده‌سازی انجام شده به این صورت بوده است که با ورودی هر داده(پس از گرفتن لگاریتم)، میانگین وزن‌دار داده‌ها با وزن ۷.۵/۱۰۰ محاسبه و بروز می‌شود. سپس، واریانس داده‌ها با احتساب میانگین جدید حساب و بروز می‌شود:

VarianceToNow = ((new_price - MeanToNow)**2 + (VarianceToNow*(CountToNow-1))) / (CountToNow)

حالا میانگین وزن‌دار واریانس را محاسبه می‌کنیم و میانگین جدید را با قبلی مقایسه میکنیم؛ در صورتی که ۴۰ بار میانگین جدید واریانس بیش از ۰.۰۱ درصد تغییرات داشته باشد، یک هشدار برای تغییر توزیع صادر می‌شود(تعداد تغییرات در یک متغیر بنام DetectCount ذخیره می‌شود). در صورتی که کمتر از ۰.۰۱ درصد تغییرات باشد، از DetectCount دو واحد کسر می‌شود و در صورتی که یکبار هشدار صادر شود این متغیر یک پنجم می‌شود و صفر نمی‌شود چرا که ممکن است بصورت متوالی تغییر توزیع رخ دهد. نمودار هشدارها در شکلی که در ادامه آورده می‌شود قابل مشاهده است.

لازم به ذکر است که تمامی محاسبات با ۵ متغیر و بدون استفاده از آرایه و لیست انجام شده است(برای نمودار و نمایش در یک آرایه صرفا ذخیره می‌شود) و این پنج متغیر به ترتیب تعداد داده تا کنون، میانگین وزن‌دار تا کنونی، واریانس تا کنون، میانگین واریانس تا کنون و تعداد تغییرات مثبت می‌باشد.

(۱) طبق توضیحات پیاده سازی فوق که انجام شده است قابل انتظار است که بایستی از میانگین وزن‌دار استفاده کرد چرا که ممکن است توزیع داده‌ها در طول زمان چندین بار تغییر و کلا عوض شود و استفاده از میانگین غیروزن‌دار کلا گمراه کننده و اشتباه باشد لذا بایستی از میانگین وزن‌دار استفاده نمود که در آن روزهای اخیر از وزن بیشتری برخوردار باشد.

(۲) خیر، هر زمانی نمیتوانیم هشدار دهیم و صرفا با آمدن یک داده که با میانگین واریانس تغییر محسوس داشته باشد تغییر در توزیع را تشخیص داده و مبنای هشدار قرار دهیم چرا که میتواند داده‌ی پرت یا نویزی بوده یا حاصل از یک جو روانی و خبری در بازار بیت کوین باشد و نه حاصل از تغییرات توزیع در مدل جریان داده لذا بایستی کمی صبر کنیم.

مکانیزمی که میتواند برای هشدار به کار گرفته شود، شمارش تعداد تغییرات اخیر در میانگین واریانس می‌باشد. این تعداد طبق توضیحات فوق در پیاده‌سازی بنده به این صورت انجام شده است که اگر ۴۰ بار تغییرات داشته باشیم(تقریبا در حد ۱.۵ ماه در چارت)، هشدار داده می‌شود. در صورتی که هشدار صادر شود، تعداد تغییرات یک پنجم تعیین می شود چرا که ممکن است دوباره تغییر توزیع رخ دهد. همچنین در صورتی که تغییراتی در قیمت بعدی وجود نداشته باشد، تعداد را صفر نمیکنیم بلکه ممکن است صرفا یک قیمت گذرا باشد لذا تعداد را دو واحد کم میکنیم تا بدین صورت یک پنالتی اعمال کرده باشیم.