|  |  |  |
| --- | --- | --- |
|  | به نام خدا |  |
| **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر**  **تحلیل داده**  **پروژه پایانی تخمین بازه قیمت خانه در سایت شیپور** | | |

|  |  |
| --- | --- |
| فاطمه زهرا علی نژاد – انوشه سعادتی – محسن آهمند | نام و نام خانوادگی |
| 810100295 | شماره‌ دانشجویی |
|  | تاریخ ارسال گزارش |

­

**فهرست گزارش سوالات**

[بخش دوم – جمع آوری و Crawl کردن 3](#_Toc125455040)

[بخش سوم – روشهای تمییزسازی، پیش پردازش 4](#_Toc125455041)

[بخش چهارم – مصورسازی و تحلیل EDA 5](#_Toc125455042)

[بخش پنجم – انتخاب ویژگی و کاهش ابعاد 6](#_Toc125455043)

[بخش پنجم – روشهای طبقه بندی 7](#_Toc125455044)

# بخش دوم – جمع آوری و Crawl کردن

سلنیوم یک ابزار قدرتمند برای کراول (Crawl) کردن صفحات وب به ویژه صفحات پویا و نیاز به تعامل یا لاگین است. با استفاده از این ابزار سعی در جمع اوری داده های سایت شیپور برای بخش اگهی های خرید و فروش منزل انجام شد.

داده های جمع اوری شده شامل مناطق مختلف شهر تهران برای 23 منطقه ی متفاوت که در مجموع بالغ بر 1200 اگهی میشود.

مناطق استخراج شده به فرم زیر میباشد . البته از ان جایی که تعداد مناطق بسیار زیاد است تعداد محدودی از ان ها انتخاب گردیده است.



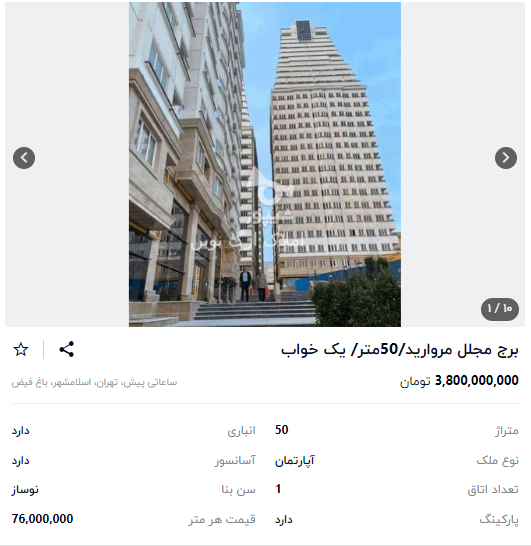
در ابتدا منطقه را انتخاب نموده و به تفکیک هر منطقه به جمع اوری اگهی های آن منطقه میپردازیم.

شرط جمع اوری اگهی برای هر منطقه این است حداقل 20 اگهی از هر منطقه موجود باشد وگرنه داده های آن منطقه جمع اوری نشده است .

پس از انتخاب هر منطقه به اگهی های کلی آن میرسیم که به فرم زیر است .



سپس بر روی هر اگهی کلیک کرده و به جمع اوری داده ها و جزییات هر اگهی پرداخته شده است . Feature های استخراج شده برای هر اگهی به فرم زیر است.



داده های جمع اوری شده در دیتافریم ذخیره و نهایتا در فایل dataframe.csv موجود است.

از جمله چالش های پیش رو این بخش زمان زیاد برای crawl کردن و جمع اوری داده ها و مشکل بزرگتر عدم پاسخگویی سایت شیپور بود . بدلیل request های زیادی که برای اگهی ها زده میشد، بعد از تعداد خاصی دیگر قادر به دریافت و مشاهده ی اگهی ها نبودیم و IP سیستم ار طرف سایت مسدود میشد.

# بخش سوم – روشهای تمییزسازی، پیش پردازش

* حذف nan و یا پرکردن ان ها
* حذف outlier با استفاده از boxplot
* یکسان کردن حروف و اعداد فارسی و انگلیسی
* تغییر نام ستون ها از فارسی به انگلیسی برای راحتی بیشتر
* حذف داده هایی که قیمت توافقی داشتند.

# بخش چهارم – مصورسازی و تحلیل EDA

# بخش پنجم – انتخاب ویژگی و کاهش ابعاد

# بخش پنجم – روشهای طبقه بندی

در این قسمت مدل‌های مختلفی استفاده می‌شود تا طبقه بندی انجام شود.

علاوه بر Accuracy از 3 متریک دیگر برای سنجش عملکرد مدل استفاده شده است.

* Accuracy:

Accuracy معیاری است که نشان می‌دهد یک مدل چقدر به درستی خروجی یا کلاس یک نمونه را پیش بینی می‌کند. اغلب در مسائل طبقه‌بندی استفاده می‌شود و به عنوان تعداد پیش‌بینی‌های صحیح تقسیم بر تعداد کل پیش‌بینی‌ها تعریف می‌شود. به عنوان مثال، اگر یک مدل 100 پیش‌بینی انجام دهد و 90 تای آن درست باشد، Accuracy مدل 90 درصد است. با این حال، Accuracy ممکن است به تنهایی معیار کافی برای عملکرد یک مدل نباشد، زیرا مثبت کاذب یا منفی کاذب را در نظر نمی‌گیرد.

* Precision:

Precision اندازه‌گیری توانایی یک مدل برای شناسایی صحیح نمونه‌های مثبت است و اغلب در مسائل طبقه‌بندی استفاده می‌شود. به عنوان تعداد پیش‌بینی‌های مثبت درست (یعنی موارد مثبت پیش‌بینی‌شده درست) تقسیم بر تعداد کل پیش‌بینی‌های مثبت درست و مثبت کاذب (یعنی همه پیش‌بینی‌های مثبت انجام‌شده توسط مدل) تعریف می‌شود. برای مثال، اگر مدلی 100 پیش‌بینی کند و 80 تای آن‌ها مثبت واقعی باشد، اما 20 مورد مثبت کاذب باشد، Precision مدل 80 درصد است. Precision بالا به این معنی است که یک مدل دارای نرخ مثبت کاذب پایینی است، به این معنی که وقتی مدل مثبت پیش‌بینی می‌کند، احتمال درستی آن بیشتر است. Precision و Recall دو معیار مهم عملکرد یک مدل هستند و اغلب در ارتباط با یکدیگر استفاده می‌شوند. در حالی که Precision نسبت مثبت های واقعی را در بین تمام پیش‌بینی‌های مثبت اندازه‌گیری می‌کند، Recall نسبت مثبت‌های واقعی را در بین تمام موارد مثبت واقعی اندازه‌گیری می‌کند. Precision و Recall با یکدیگر مبادله هستند، به این معنی که وقتی Precision را افزایش می‌دهید، Recall کاهش می‌یابد و بالعکس.

* Recall:

Recall معیاری برای سنجش توانایی یک مدل برای شناسایی صحیح تمام نمونه های مرتبط است و اغلب در مسائل طبقه‌بندی استفاده می‌شود. به عنوان تعداد پیش‌بینی‌های مثبت واقعی (یعنی نمونه‌های مثبت پیش‌بینی‌شده درست) تقسیم بر تعداد کل پیش‌بینی‌های مثبت درست و منفی کاذب (یعنی همه موارد مثبت واقعی) تعریف می‌شود. به عنوان مثال، اگر یک مدل 100 پیش‌بینی کند و 80 تای آن‌ها مثبت واقعی باشد، اما 20 مورد منفی کاذب باشد، Recall مدل 80 درصد است. Recall بالا به این معنی است که یک مدل دارای نرخ منفی کاذب پایینی است، به این معنی که وقتی نمونه های مثبت واقعی وجود داشته باشد، مدل به درستی آنها را بیشتر پیش‌بینی می‌کند.

* F1-Score:

F1-Score معیاری است که Precision و Recall را ترکیب می‌کند و اغلب به عنوان یک معیار واحد برای ارزیابی عملکرد یک مدل در مسائل طبقه‌بندی استفاده می‌شود. به عنوان میانگین هارمونیک Precision و Recall تعریف می‌شود که بهترین مقدار 1 و بدترین مقدار 0 است. F1-Score به صورت

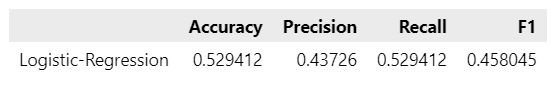
2 \* (Precision \* Recall) / (Precision + Recall) محاسبه می‌شود. این روشی برای ایجاد تعادل بین Precision و Recall است و به ویژه هنگامی که توزیع کلاسی ناهموار دارید مفید است. F1-Score برابر 1 نشان می‌دهد که مدل دارای Precision و Recall کامل است، در حالی که F1-Score برابر 0 نشان می‌دهد که مدل هیچ نقطه مثبت واقعی ندارد. مدلی با F1-Score بالا تعادل خوبی بین Precision و Recall دارد. در برخی موارد، F1-Score ممکن است بیشتر آموزنده باشد تا Precision ، به ویژه زمانی که کلاس ها نامتعادل هستند یا زمانی که علاقه به نرخ مثبت کاذب و منفی کاذب است.

استفاده از مدل‌های مختلف:

* Logistic Regression:

رگرسیون لجستیک یک روش آماری برای تجزیه و تحلیل مجموعه داده‌ای است که در آن یک یا چند متغیر مستقل وجود دارد که یک نتیجه را تعیین می‌کند. نتیجه با یک متغیر دوگانه (که در آن فقط دو نتیجه ممکن وجود دارد) اندازه‌گیری می‌شود. برای پیش‌بینی یک نتیجه باینری (1/0، بله / خیر، درست / نادرست) با توجه به مجموعه‌ای از متغیرهای مستقل استفاده می‌شود. به عبارت ساده، روشی برای برازش یک مدل رگرسیون زمانی است که متغیر پاسخ باینری است. رگرسیون لجستیک احتمال تعلق یک ورودی داده شده به یک دسته یا کلاس خاص را تخمین می‌زند. ورودی می‌تواند یک یا چند متغیر باشد و خروجی احتمال تعلق متغیرهای ورودی به یک کلاس خاص خواهد بود. مدل رگرسیون لجستیک یک مدل خطی برای طبقه‌بندی باینری است که می‌توان آن را به طبقه‌بندی چند طبقه تعمیم داد. احتمال یک پاسخ باینری را به عنوان تابعی از یک یا چند متغیر پیش‌بینی تخمین می‌زند.

با استفاده از داده‌های آموزش، مدل Logistic Regression آموزش داده می‌شود و نتایج به صورت زیر به دست می‌آید.

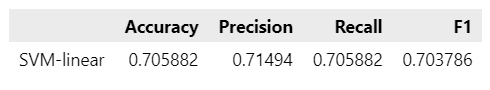


* SVM:

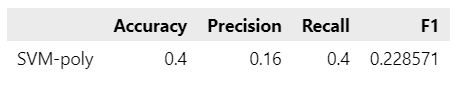
ماشین‌های بردار پشتیبان (SVM) نوعی الگوریتم یادگیری تحت نظارت هستند که می‌توانند برای کارهای طبقه‌بندی و رگرسیون استفاده شوند. هدف یک SVM یافتن بهترین مرز یا سطح تصمیم است که کلاس‌های مختلف در مجموعه داده را از هم جدا می‌کند. این الگوریتم با شناسایی هایپرصفحه ای که حاشیه را به حداکثر می‌رساند، مرز را پیدا می‌کند، که فاصله بین ابر صفحه و نزدیک ترین نقاط داده از هر کلاس است که به عنوان بردارهای پشتیبانی نیز شناخته می‌شود. این بردارهای پشتیبانی، نقاط داده‌ای هستند که به مرز تصمیم نزدیکتر هستند و بیشترین تأثیر را در تعیین موقعیت آن دارند. SVM ها به ویژه زمانی مفید هستند که داده ها ویژگی‌های زیادی داشته باشند، یا زمانی که داده‌ها به صورت خطی قابل تفکیک نیستند، به این معنی که نمی توان از یک خط مستقیم یا یک ابر صفحه برای جداسازی کلاس‌ها استفاده کرد. در این مورد، SVM می‌تواند از تکنیکی به نام ترفند هسته استفاده کند تا داده‌های ورودی را به فضایی با ابعاد بالاتر تبدیل کند که در آن به صورت خطی قابل تفکیک می‌شود. هسته‌های رایج مورد استفاده خطی، چند جمله ای، rbf و سیگموئید هستند. یکی از مزایای اصلی SVMها این است که می‌توان از آنها برای حل مسائل طبقه‌بندی پیچیده و غیرخطی استفاده کرد و با داده‌های با ابعاد بالا به خوبی کار می‌کند. آنها همچنین توانایی مدیریت کارآمد مجموعه داده‌های بزرگ را دارند و می‌توانند حتی با تعداد کمی از نمونه‌های آموزشی به خوبی کار کنند. SVM ها به طور گسترده در زمینه‌های مختلفی مانند تشخیص تصویر، بیوانفورماتیک و پردازش زبان طبیعی استفاده شده‌اند. به خاطر داشته باشید که SVM برای مجموعه داده‌های بزرگ با نمونه‌های زیاد مناسب نیست، همچنین به انتخاب هسته و پارامتر منظم‌سازی حساس است. بنابراین، تنظیم پارامترها و انتخاب هسته‌ای که به بهترین وجه با داده‌های شما مطابقت دارد، مهم است.

با استفاده از داده‌های آموزش، مدل SVM آموزش داده می‌شود و نتایج به صورت زیر به دست می‌آید.

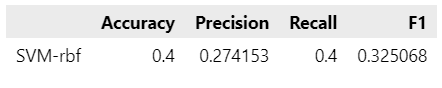
اگر Kernel را برابر Linear قرار داده شود:



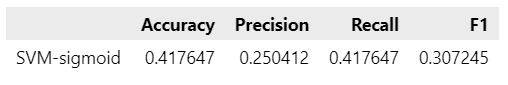
اگر Kernel را برابر Polynomial قرار داده شود:



اگر Kernel را برابر rbf قرار داده شود:



اگر Kernel را برابر Sigmoid قرار داده شود:

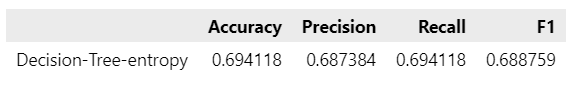


* Decision Tree:

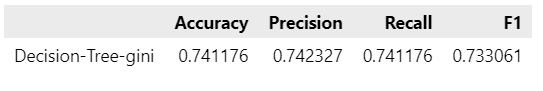
درخت تصمیم یک ساختار درختی فلوچارت مانند است که برای نشان دادن فرآیند تصمیم‌گیری یا مجموعه‌ای از تصمیمات و پیامدهای احتمالی آنها استفاده می‌شود. این یک نوع الگوریتم یادگیری نظارت شده است که می‌تواند برای کارهای طبقه‌بندی و رگرسیون استفاده شود. الگوریتم درخت تصمیم با یک گره ریشه شروع می‌شود که کل مجموعه داده را نشان می‌دهد. سپس گره ریشه به دو یا چند گره فرزند تقسیم می‌شود که نشان دهنده زیرمجموعه‌های داده است. سپس داده‌ها به صورت بازگشتی به زیر مجموعه‌ها بر اساس شرایط خاص تقسیم می‌شوند تا زمانی که به گره‌های برگ برسند، که نشان دهنده نتایج یا پیش‌بینی‌های نهایی است. هر گره داخلی درخت نشان‌دهنده آزمایشی بر روی یک ویژگی، هر شاخه نشان‎‌دهنده نتیجه آزمایش و هر گره برگ نشان‌دهنده یک برچسب کلاس است. بالاترین گره تصمیم‌گیری در یک درخت که با بهترین پیش‌بینی‌کننده به نام گره ریشه مطابقت دارد. الگوریتم درخت تصمیم از یک رویکرد اکتشافی به نام پارتیشن‌بندی بازگشتی برای تقسیم داده‌ها به زیر مجموعه‌ها استفاده می‌کند. رایج‌ترین اکتشافی مورد استفاده برای پارتیشن‌بندی بازگشتی، ناخالصی جینی است که احتمال طبقه‌بندی نادرست یک عنصر تصادفی انتخاب شده را در صورتی که به طور تصادفی بر اساس توزیع کلاس در زیرمجموعه برچسب‌گذاری شده باشد، اندازه‌گیری می‌کند. یکی از مزیت‌های اصلی درخت‌های تصمیم این است که تفسیر و درک آن‌ها آسان است و می‌توانند داده‌های مقوله‌ای و عددی را مدیریت کنند. درخت‌های تصمیم همچنین می‌توانند مجموعه داده‌های بزرگ، مقادیر از دست رفته و مقادیر پرت را به خوبی مدیریت کنند. با این حال، درخت های تصمیم مستعد بیش از حد برازش هستند، که زمانی اتفاق می‌افتد که مدل بیش از حد پیچیده باشد و نویز در داده‌ها را ضبط کند. برای جلوگیری از برازش بیش‌ازحد، می‌توان از تکنیک‌هایی مانند هرس، توقف زودهنگام یا روش‌های گروهی مانند جنگل تصادفی استفاده کرد. توجه به این نکته مهم است که درخت تصمیم برای مجموعه داده‌هایی با ویژگی‌ها یا مجموعه داده‌هایی که ماهیت خطی دارند مناسب نیست. همچنین، درخت تصمیم می‌تواند به سمت ویژگی هایی با سطوح مختلف سوگیری داشته باشد.

با استفاده از داده‌های آموزش، مدل Decision Tree آموزش داده می‌شود و نتایج به صورت زیر به دست می‌آید.

اگر Criterion را برابرEntropy قرار داده شود:



اگر Criterion را برابر Gini قرار داده شود:

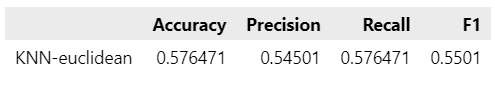


* KNN:

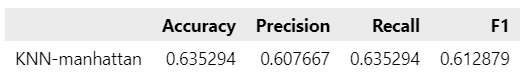
K-Nearest Neighbors (KNN) نوعی الگوریتم یادگیری نظارت شده است که می تواند برای کارهای طبقه بندی و رگرسیون استفاده شود. ایده اصلی پشت الگوریتم KNN این است که ویژگی‌های یک نقطه داده جدید را با ویژگی‌های نقاط داده در مجموعه آموزشی مقایسه می‌کند و برچسب رایج‌ترین کلاس را در میان K نزدیک‌ترین نقاط داده اختصاص می‌دهد. K در الگوریتم KNN نشان دهنده تعداد نزدیکترین همسایه هایی است که الگوریتم هنگام پیش بینی در نظر می گیرد. مقدار K یک فراپارامتر است که توسط کاربر قابل تنظیم است. مقدار کوچکتر K منجر به یک مدل پیچیده تر می شود، در حالی که مقدار K بزرگتر منجر به یک مدل ساده تر می شود. متریک فاصله مورد استفاده برای تعیین نزدیکترین همسایگان نیز یک فراپارامتر است که توسط کاربر قابل تنظیم است. متداول ترین معیارهای فاصله، فاصله اقلیدسی و فاصله منهتن هستند. KNN یک الگوریتم ساده و قدرتمند است که می تواند برای کارهای مختلف استفاده شود. این به ویژه برای مسائلی که مرز تصمیم گیری خطی نیست مفید است و می توان از آن برای کارهای طبقه بندی و رگرسیون استفاده کرد. یکی از مزایای اصلی KNN این است که پیاده سازی و تفسیر آن آسان است و هیچ فرضی در مورد توزیع داده های اساسی نمی کند. با این حال، KNN می تواند از نظر محاسباتی گران باشد، به ویژه برای مجموعه داده های بزرگ، زیرا نیاز به محاسبه فاصله بین یک نقطه داده جدید و تمام نقاط داده در مجموعه آموزشی دارد. توجه به این نکته مهم است که KNN به مقیاس ویژگی ها حساس است، بنابراین توصیه می شود قبل از استفاده از KNN داده ها را عادی کنید. همچنین، KNN زمانی که تعداد ابعاد کم باشد عملکرد خوبی دارد، اما زمانی که تعداد ابعاد افزایش یابد، از نظر محاسباتی گران می شود.

با استفاده از داده‌های آموزش، مدل Decision Tree آموزش داده می‌شود و نتایج به صورت زیر به دست می‌آید.

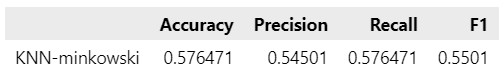
اگر مقدار K را برابر 6 قرار داده و metric را برابر Euclidean قرار داده شود:



اگر مقدار K را برابر 6 قرار داده و metric را برابر Manhattan قرار داده شود:



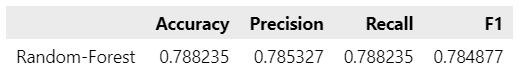
اگر مقدار K را برابر 6 قرار داده و metric را برابر Minkowski قرار داده شود:



* Random Forest:

Random Forest یک الگوریتم یادگیری گروهی است که چندین درخت تصمیم را برای پیش‌بینی ترکیب می‌کند. این یک نوع الگوریتم یادگیری نظارت شده است که می تواند برای کارهای طبقه بندی و رگرسیون استفاده شود. ایده اصلی پشت جنگل تصادفی این است که با انتخاب تصادفی زیرمجموعه‌ای از ویژگی‌ها و زیرمجموعه‌ای از داده‌های آموزشی، چندین درخت تصمیم ایجاد کنیم و پیش‌بینی‌های آن‌ها را با اخذ اکثریت رای برای طبقه‌بندی و میانگین‌گیری برای رگرسیون ترکیب می‌کند. با انجام این کار، بیش از حد برازش را کاهش می دهد و تعمیم را بهبود می بخشد. هنگام آموزش درخت تصمیم، الگوریتم با این مشکل مواجه می شود که در هر مرحله کدام ویژگی را تقسیم کند. در هر مرحله از فرآیند درخت سازی، یک زیرمجموعه تصادفی از ویژگی ها انتخاب می شود و بهترین ویژگی تقسیم از زیر مجموعه استفاده می شود. با میانگین گیری نتایج بسیاری از درختان، می توانیم واریانس مدل را کاهش دهیم. Random Forest مخصوصاً برای مسائلی که مرز تصمیم خطی نیست مفید است و می تواند مجموعه داده های بزرگ، مقادیر از دست رفته و مقادیر پرت را به خوبی مدیریت کند. همچنین اهمیت ویژگی را فراهم می کند که می توان از آن برای انتخاب ویژگی های مهم برای مدل استفاده کرد. با این حال، جنگل تصادفی به منابع محاسباتی بیشتری نسبت به یک درخت تصمیم نیاز دارد، و می‌تواند کمتر قابل تفسیر باشد زیرا یک مدل جعبه سیاه است. توجه به این نکته مهم است که الگوریتم جنگل تصادفی برای مجموعه داده هایی با تعداد زیاد ویژگی های همبسته مناسب نیست. همچنین، می‌تواند به نویز در داده‌ها حساس باشد، بنابراین توصیه می‌شود از تکنیک‌هایی مانند روش‌های هرس یا مجموعه برای جلوگیری از برازش بیش از حد استفاده کنید.

با استفاده از داده‌های آموزش، مدل Random Forest آموزش داده می‌شود و نتایج به صورت زیر به دست می‌آید.



* Naïve Bayes:

Naive Bayes یک الگوریتم احتمالی برای طبقه بندی است. این یک الگوریتم ساده، آسان برای پیاده سازی و کارآمد است که می تواند برای کارهای مختلفی مانند طبقه بندی متن، فیلتر کردن هرزنامه ها و تجزیه و تحلیل احساسات استفاده شود. ایده اصلی پشت بیز ساده این است که از قضیه بیز برای محاسبه احتمال یک نقطه داده متعلق به یک کلاس خاص بر اساس ویژگی های آن نقطه داده استفاده کنیم. قضیه بیز بیان می کند که:

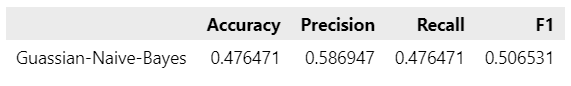
P(A|B) = P(B|A) \* P(A) / P(B)

جایی که A برچسب کلاس و B مجموعه ای از ویژگی ها است. P(A|B) احتمال برچسب کلاس A با توجه به ویژگی های B، P(B|A) احتمال، P(A) احتمال قبلی برچسب کلاس A، و P(B) مدرک است.

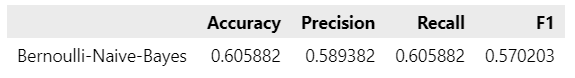
Naive Bayes این فرض را «ساده‌انگیز» می‌کند که ویژگی‌ها با توجه به برچسب کلاس به صورت شرطی مستقل هستند، به این معنی که احتمال می‌تواند به عنوان حاصلضرب احتمالات ویژگی‌های فردی محاسبه شود. این فرض محاسبات را ساده می کند، اما در عمل می تواند محدود کننده باشد. انواع مختلفی از Naive Bayes مانند Gaussian Naive Bayes، Multinomial Naive Bayes و Bernoulli Naive Bayes وجود دارد که هر کدام برای نوع خاصی از داده ها مناسب هستند. به عنوان مثال، Gaussian Naive Bayes برای داده های پیوسته استفاده می شود در حالی که Multinomial Naive Bayes برای داده های گسسته استفاده می شود. Naive Bayes یک الگوریتم سریع و ساده است که می تواند به خوبی با مجموعه داده های با ابعاد بالا کار کند. با این حال، فرض استقلال بین ویژگی‌ها می‌تواند در عمل یک محدودیت باشد و ممکن است به خوبی الگوریتم‌های دیگر مانند Random Forest و SVM عمل نکند.

با استفاده از داده‌های آموزش، مدل Naïve Bayes آموزش داده می‌شود و نتایج به صورت زیر به دست می‌آید.

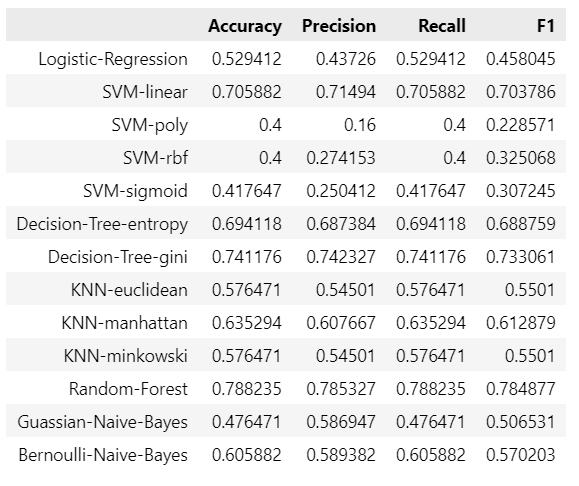
اگر از Guassian-Naive-Bayes استفاده شود:



اگر از Bernoulli-Naive-Bayes استفاده شود:

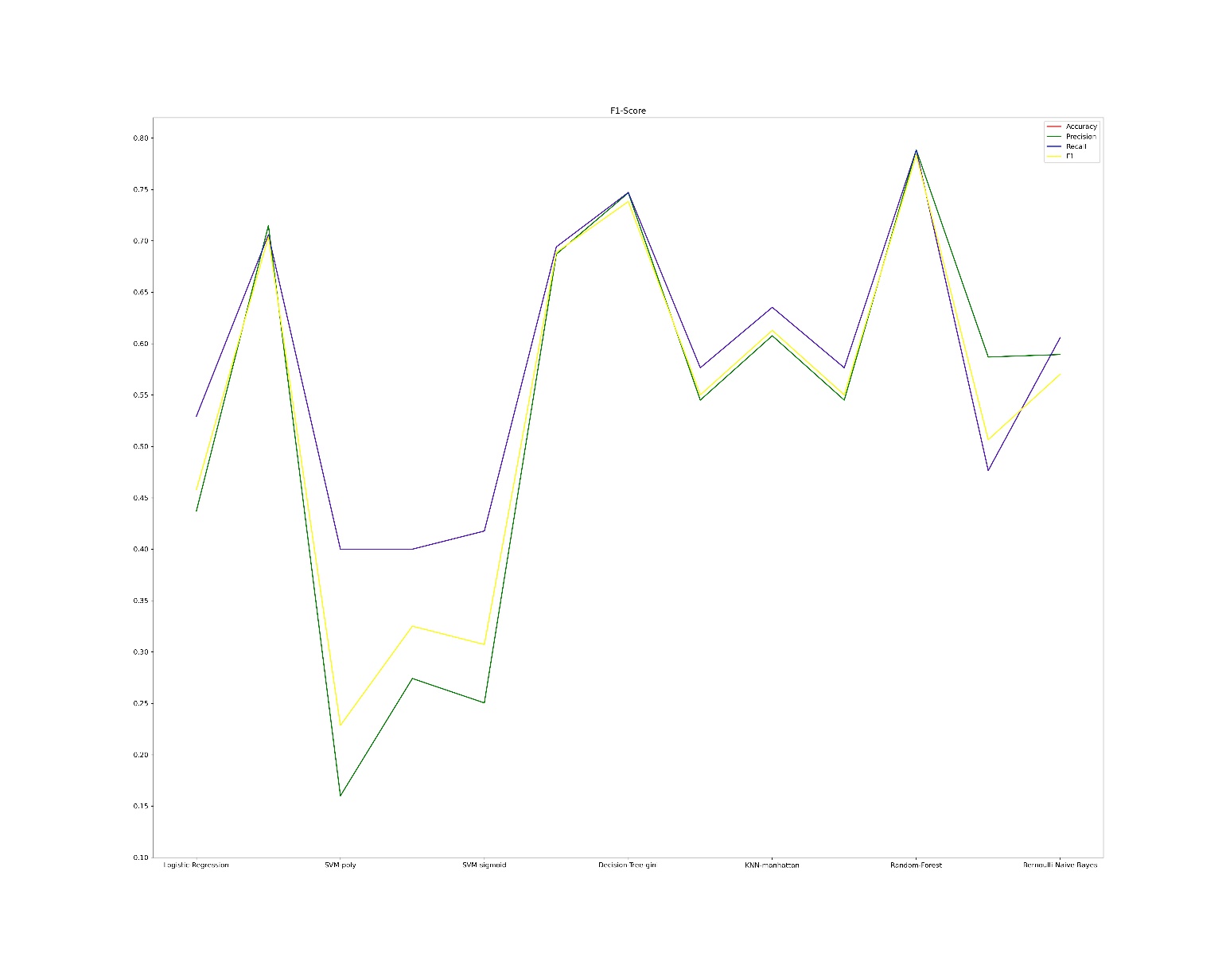


و در انتها نتایج همه‌ی مدل ها را با هم نشان داده می‌شود



همانطور که مشاهده می‌شود در SVM با پارامترهای مختلف، اگر Kernel آن linear باشد، عملکرد آن بهتر از بقیه است. در Decision Tree نیز وقتی Criterion برابر Gini باشد، عملکرد آن بهتر از آن است که Entropy باشد. در KNN با K برابر 6 متریک فاصله‌ی Manhattan بهتر از بقیه است. بین متدهای مختلف Naïve Bayes عملکرد Bernoulli بهتر از Guassian بوده است.

ولی بین همه‌ی متدها بهترین عملکرد را Random Forest داشته است.



همانطور که مشاهده می‌شود Random Forest از نظر Accuracy و Precision و Recall و F1-Score بهتر از بقیه مدل‌ها بوده است و بیشترین عملکرد را داشته است.

برای تمامی مدل‌ها Accuracy و Precision آنها در یک نمودار نشان داده می‌شود:



همانطور که این نمودار نشان می‌دهد، برای بیشتر مدل‌ها Accuracy یا بیشتر از Precision بوده است یا برابر بوده اند. برای Random Forest هم Accuracy و هم Precision بیشتر از بقیه‌ی مدل‌ها بوده است. کمترین Accuracy و Precision را دو مدل SVM با Kernel های Sigmoid و Polynomial داشته اند.

برای تمامی مدل‌ها Accuracy و Recall آنها در یک نمودار نشان داده می‌شود:



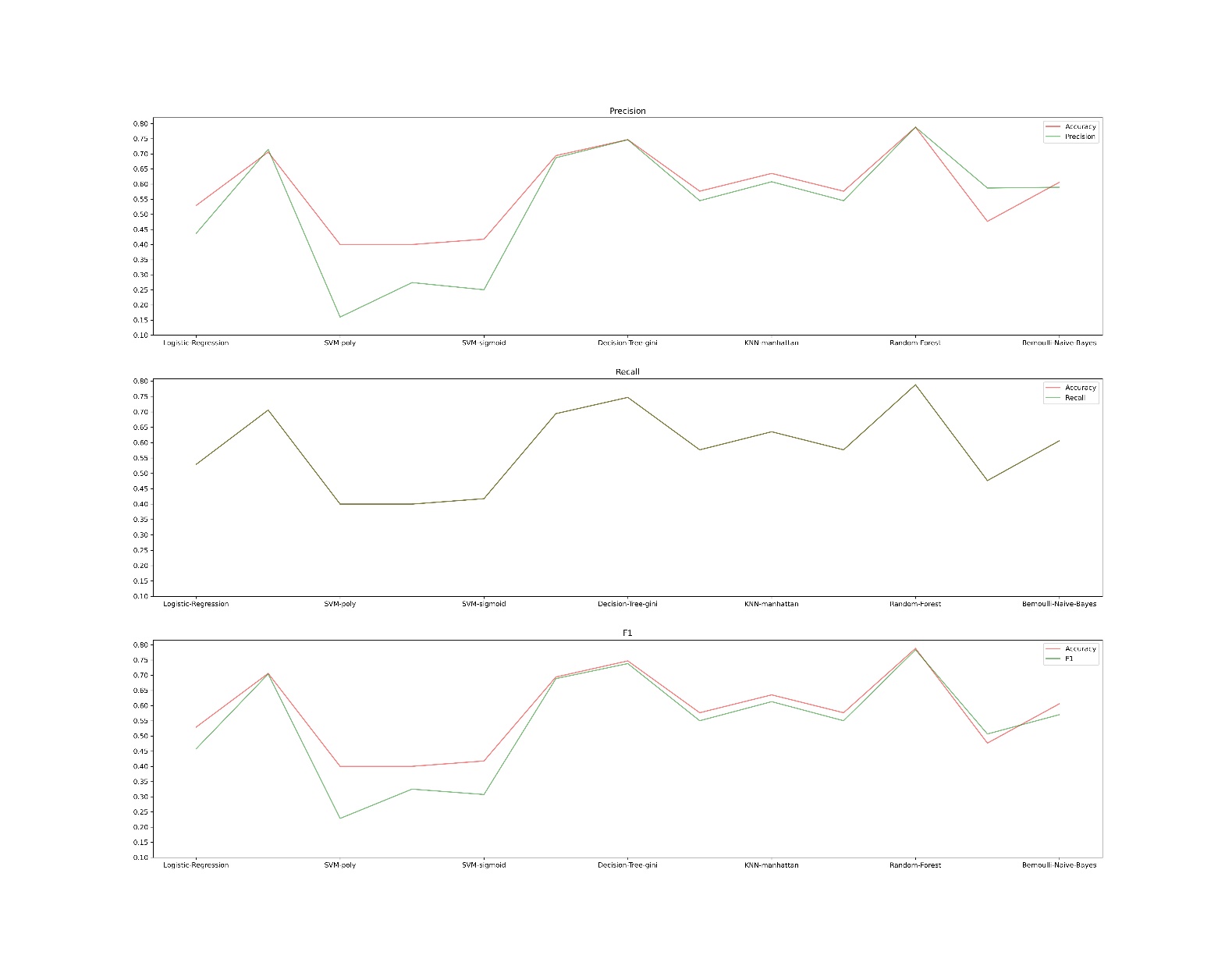
همانطور که این نمودار نشان می‌دهد، برای تمامی مدل‌ها Accuracy با Recall برابر بوده است. Random Forest بیشتر از بقیه‌ی مدل‌ها دارای Accuracy و Recall بوده است. کمترین Accuracy و Recall را دو مدل SVM با Kernel های Sigmoid و Polynomial داشته اند.

برای تمامی مدل‌ها Accuracy و F1-Score آنها در یک نمودار نشان داده می‌شود:



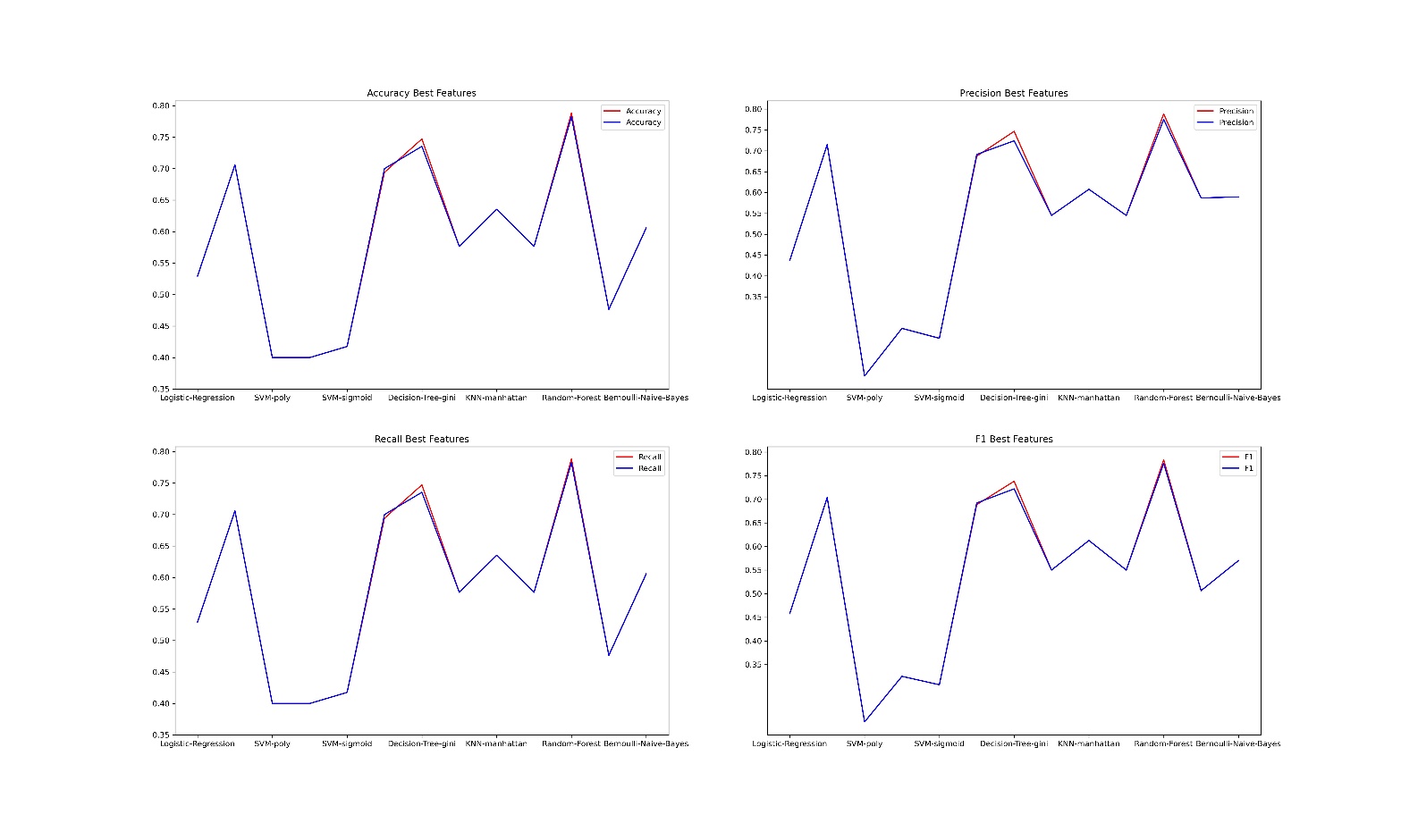
همانطور که این نمودار نشان می‌دهد، برای بیشتر مدل‌ها Accuracy یا بیشتر از F1-Score بوده است یا برابر بوده اند. برای Random Forest هم Accuracy و هم F1-Score بیشتر از بقیه‌ی مدل‌ها بوده است. کمترین Accuracy و F1-Score را دو مدل SVM با Kernel های Sigmoid و Polynomial داشته اند.

در انتها 3 نمودار باهم نشان داده شده است:



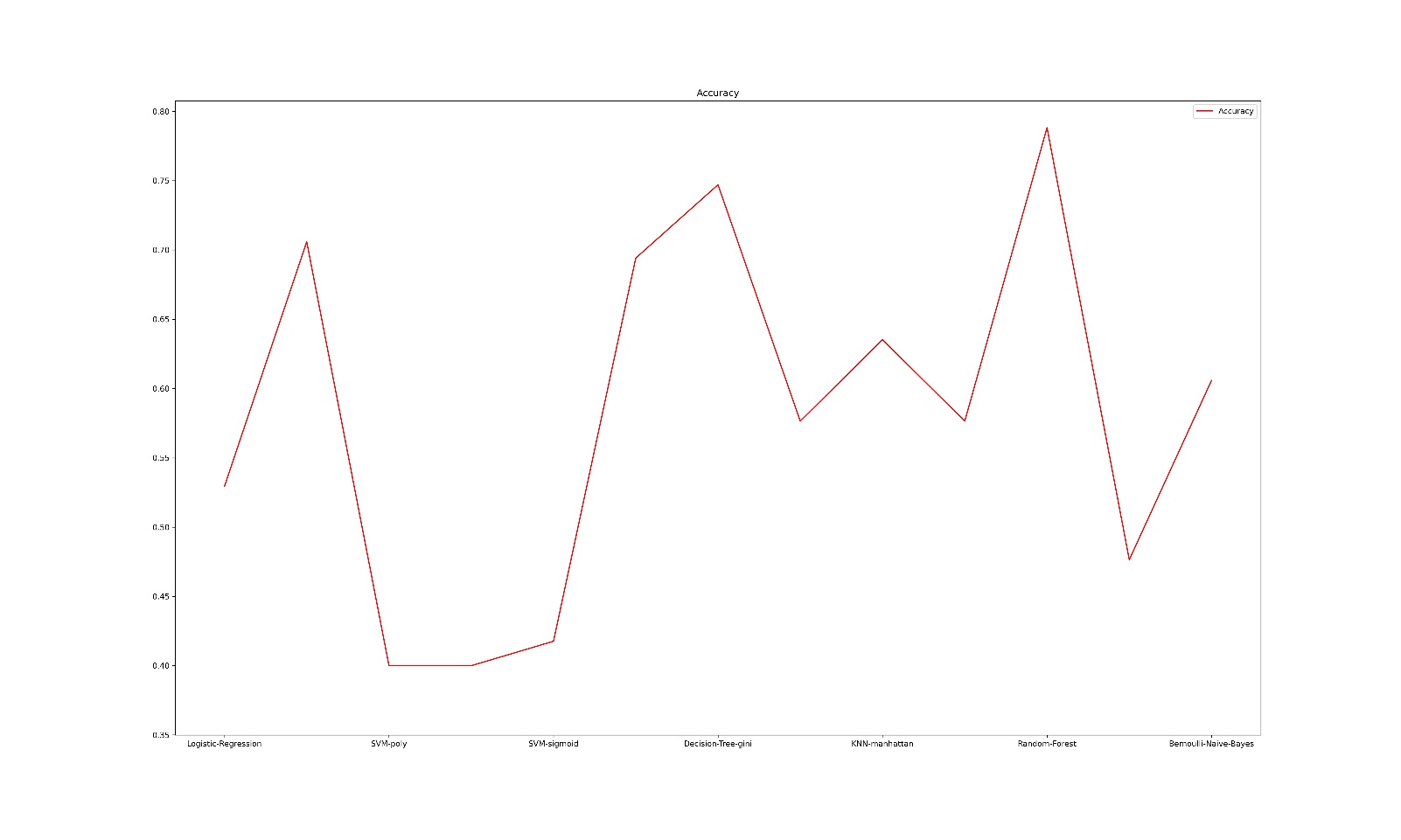
ویژگی‌هایی که از الگوریتم Random Forest Regressor به دست آمده است شامل پارکینگ، آسانسور، متراژ خونه، تعداد اتاق، سن خونه است.

تمامی مدل‌ها را دوباره آموزش داده و متریک‌ها حساب می‌شود. سپس با مقدار متریک‌هایی که برای همه‌ی ویژگی ها به دست آمده است، مقایسه می‌شود.



با توجه به نمودارهای بالا می‌توان به این نتیجه رسید که بیشتر مدل‌ها وقتی فقط از ویژگی‌هایی که الگوریتم RandomForestRegressor انتخاب کرده است، استفاده می‌کنند، متریک های آنها تغییری نمی‌کنند و فقط مدل Decision Tree با Criterion برابر Gini متفاوت عمل کرده است و تمامی متریک‌های آن کمتر شده است.

با توجه به آموزش مدل‌ها مقادیر Accuracy برای تمامی مدل‌ها رسم می‌شود.



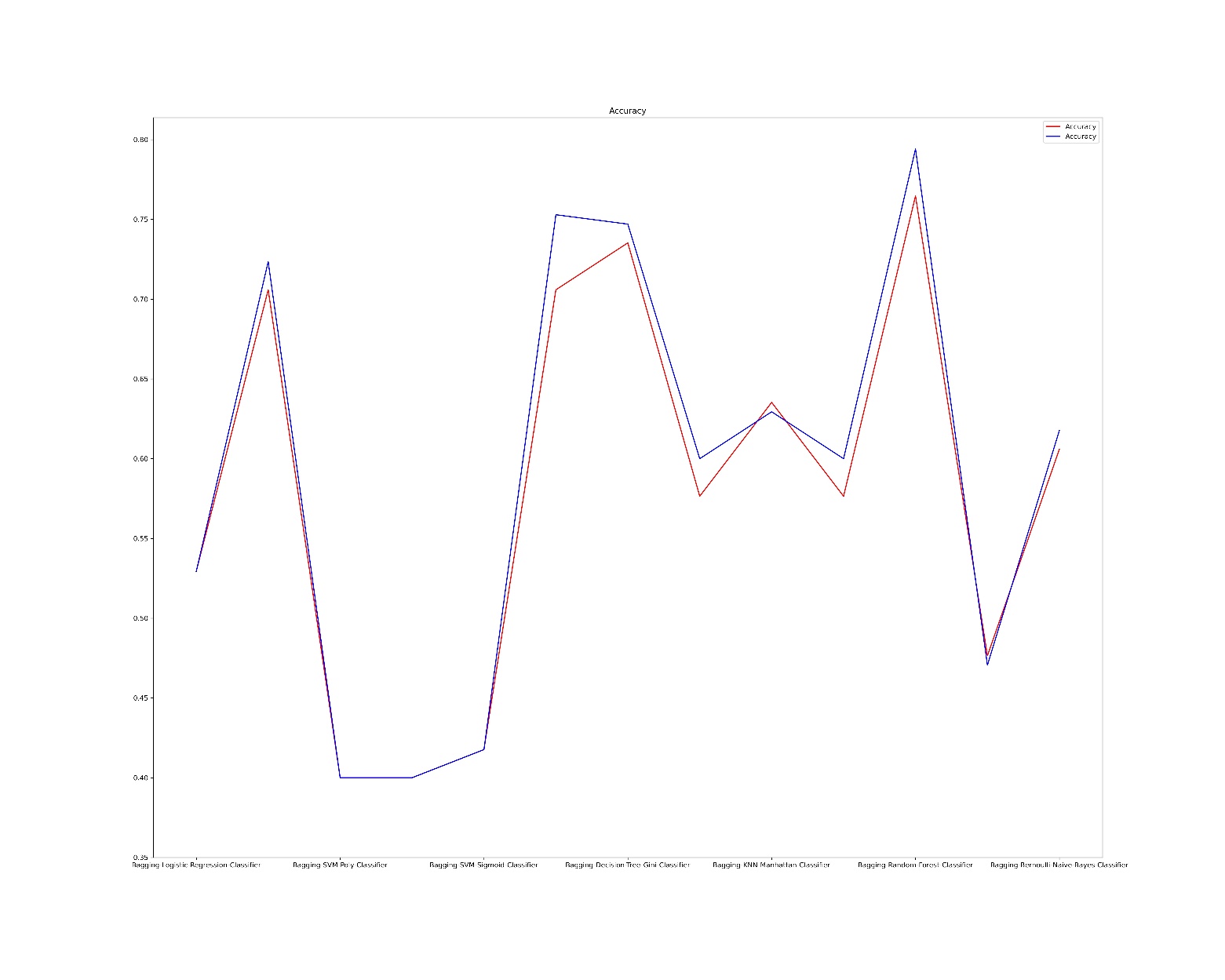
با توجه به تفاسیری که از قسمت‌های قبل به دست آمده است. می‌توان نتیجه گرفت که مقدار Accuracy برای این مسئله می‌تواند متریک خوبی برای سنجش عملکرد مدل‌ها باشد. و با توجه به Accuracy و متریک‌های دیگر مدل Random Forest بهترین عملکرد را داشته است.

مدل‌های دیگری نیز می‌توان استفاده کرد تا دیتاست را آموزش ببینند و پیشبینی کنند.

* Esemble Methods:

روش‌های گروهی تکنیکی در یادگیری ماشینی هستند که در آن چندین مدل برای حل یک مشکل آموزش داده می‌شوند و پیش‌بینی‌های آنها به نوعی برای بهبود عملکرد کلی ترکیب می‌شوند. انواع مختلفی از روش‌های گروه وجود دارد، مانند کیسه‌بندی، تقویت و انباشتن. Bagging شامل آموزش چندین مدل به طور مستقل و میانگین گیری پیش بینی های آنها است. تقویت شامل آموزش مدل های متوالی است که در آن هر مدل سعی می کند اشتباهات مدل قبلی را اصلاح کند. انباشتگی شامل آموزش مدل های متعدد و استفاده از پیش بینی های آنها به عنوان ورودی برای آموزش یک مدل سطح بالاتر است. روش‌های گروهی اغلب در موقعیت‌هایی استفاده می‌شوند که یک مدل واحد برای دستیابی به عملکرد خوب در یک کار کافی نیست و می‌تواند منجر به بهبود قابل توجهی در دقت و استحکام شود.

با استفاده از Bagging مدل‌ها آموزش داده می‌شوند و متریک‌های آنها با متریک هایی که از روش بدون Bagging به دست آمده است، مقایسه می‌شود.

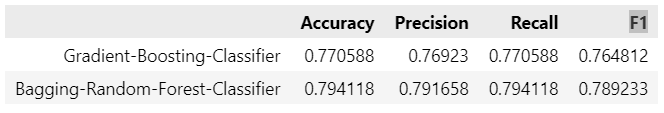


همانطور که مشاهده می‌شود در بیشتر مدل‌ها با استفاده از Bagging به Accuracy برابر یا بیشتر رسیده‌اند. بهترین عملکرد باز هم Random Forest بوده است و با Bagging به Accuracy بهتر نیز رسیده است.

* Gradient Boosting Machine (GBM):

ماشین تقویت گرادیان (GBM) نوعی روش مجموعه ای است که برای وظایف یادگیری نظارت شده مانند طبقه بندی و رگرسیون استفاده می شود. یادگیرندگان ضعیف (مانند درخت تصمیم) را ترکیب می کند تا مدل قوی تری را تشکیل دهد که می تواند پیش بینی های دقیق تری انجام دهد. GBM با آموزش مکرر مدل های ضعیف بر روی خطاهای باقیمانده مدل های قبلی کار می کند. در هر تکرار، یک مدل جدید برای پیش‌بینی باقیمانده‌های مدل‌های قبلی آموزش داده می‌شود. سپس این مدل‌ها با هم ترکیب می‌شوند تا یک مدل نهایی را تشکیل دهند که با جمع کردن پیش‌بینی‌های همه مدل‌های مجموعه، پیش‌بینی می‌کند. یکی از مزایای کلیدی GBM انعطاف پذیری آن است. این می تواند انواع مختلفی از داده ها را مدیریت کند و می تواند برای طبقه بندی خطی و غیر خطی و مسائل رگرسیون استفاده شود. GBM همچنین می‌تواند با آموزش چندین درخت و میانگین‌گیری پیش‌بینی‌های آن‌ها، داده‌های گمشده و پرت را مدیریت کند. GBM همچنین به دلیل توانایی خود در دستیابی به عملکرد پیشرفته در بسیاری از وظایف یادگیری ماشینی، به ویژه هنگامی که با درخت تصمیم به عنوان یادگیرندگان پایه استفاده می شود، شناخته شده است. با این حال، GBM می تواند به بیش از حد برازش حساس باشد، به خصوص زمانی که داده ها نویز دارند یا زمانی که تعداد درختان خیلی زیاد است. برای کاهش آن، استفاده از تکنیک هایی مانند توقف زودهنگام و منظم کردن توصیه می شود. GBM در چندین کتابخانه مانند scikit-learn، XGBoost، LightGBM و CatBoost در دسترس است.

با استفاده از الگوریتم Gradient Boosting Machine مدل آموزش داده می‌شود و متریک‌های آن به دست می‌آید.



همانطور که مشاهده می‌شود متریک‌های الگوریتم GBM کمتر از Bagging Random Forest است.

