گزارش پروژه چهارم علوم داده(بخش دوم)

محمدرضا صیدگر - 97222055

این پروژه راجع به کار با داده های unbalance است و همینطور feature engineering یا همان مهندسی ویژگی است.
داده های unbalance به این معنی است که مثلا در این پروژه که مسئله طبقه بندی کلاس های 0 و 1 است تعداد داده هایی که به کلاس 1 تعلق دارند بسیار بیشتر است از تعداد داده های کلاس 0 که اگر یک مدلی داشته باشیم که فقط کلاس 1 رو پیش بینی کنه میتواند درستی زیادی را در این مسئله داشته باشد.
پس هدف اول درست کردن این مشکل است و هدف بعدی این است که ویژگی های خوبی را پیدا کنیم و به مدل بدیم برای اینکه درستی بهتری داشته باشیم.
برای همه این اهداف مدل های پیشبینی کننده ای مثل logistic regression و svm و knn و ... انتخاب شد که در زیر به نتایج همه آنها اشاره خواهیم کرد.
داده هایی که با آنها کار داریم در این پروژه مربوط به تبلیغاتی است که برای یک شخص نمایش داده میشود و پیش بینی ما روی این است که آیا آن شخص وارد آن صفحه تبلیغاتی میشود یا خیر.
در بخش مهندسی ویژگی برای ستون های ip,app,device,os,channel بجای خود این ستون ها از count encoder آنها استفاده شد چرا که خود اون اعداد اولیه احتمالا معنایی از نظر عددی برای مدل ها نخواهند داشت ولی با استفاده از این encoder از نظر عددی مفهومی وجود دارد که نشان دهنده فراوانی آن داده در دیتاست است.
ستونی با اسم hours اضافه شد به ویژگی ها که نشان دهنده این است که آن شخص وقتی تبلیغات را می بیند چه میزان ساعت از اول آن ماه گذشته است.
ستون دیگری با اسم day_parts اضافه شد که عدد 0 نشان میدهد شخص نصف شب آن تبلیغات را دیده ، عدد 1 یعنی صبح دیده ، 2 یعنی ظهر یا عصر دیده است ، 3 یعنی غروب دیده است و 4 یعنی شب دیده است.
همچنین ستون های دیگری مثل ip*app و ip*device و ... اضافه شد که در واقع ضرب آن ستون ها با هم است که میتواند ویژگی های را به مدل بدهد که مدل خودش ممکن نباشد این ها را به راحتی بدست آورد و در ادامه هم میبینیم که این ویژگی ها برای بعضی از مدل ها بسیار مفید هم بوده اند.
بخش بعدی راجع balance داده ها است که برای این کار میتوان از undersampling و یا oversampling استفاده کرد که undersampling یعنی حذف و کم کردن داده های از کلاس با تعداد بیشتر و oversampling یعنی زیاد کردن داده های از کلاس با تعداد کمتر که ما در این پروژه از undersampling استفاده کردیم که خود متد های فراوانی هم دارند نظیر AllKNN و NearMiss و ... که هر کدام از این ها استفاده شدند اما به علت زیاد بودن حجم داده ها به جواب نمی رسیدند که در نهایت از RandomUnderSampler استفاده شد زیرا محاسبات طولانی و پیچیده ای

ندارند و به صورت تصادفی داده ها را انتخاب میکنند. در کنار این داده ها ، داده های unbalanced هم ذخیره هستند زیرا با آنها هم کار خواهیم داشت.
در ادامه 80 درصد داده ها به عنوان داده های آموزشی انتخاب شدند و 20 درصد هم برای ارزیابی مدل.
از آنجایی که ممکن است بعضی از داده ها پراکندگی زیادی داشته باشند روی داده ها یک transform sqrt انجام شد که متراکم تر شده و در ادامه از standard scaler استفاده شد چرا که میانگین داده ها 0 شود و حدودا اکثر داده های توی بازه -1 تا 1 قرار بگیرند چون مدل ها با این بازه اعداد عملکرد بهتری خواهند داشت.
در زیر به نتایج انواع مدل های مختلف طبقه بندی اشاره می کنیم:

1) Logistic Regression:

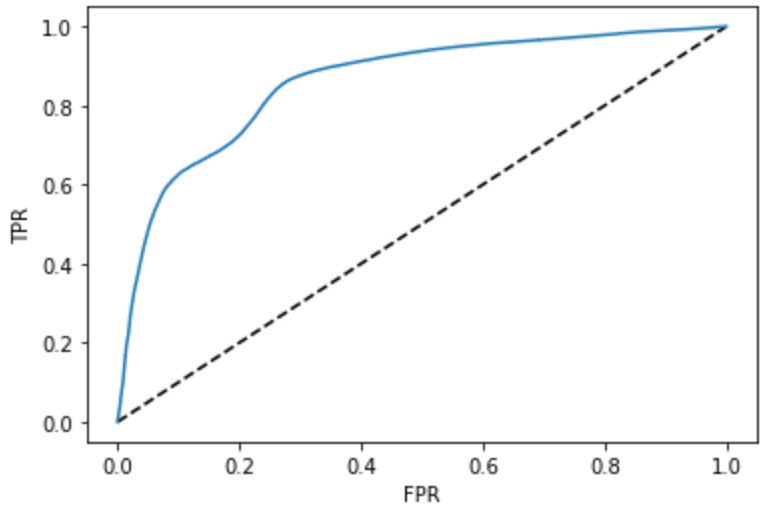
ماتریس confusion آن به شکل زیر شد:

	Predict yes	**Predict no**
Actual yes	61998	29093
Actual no	10466	81182

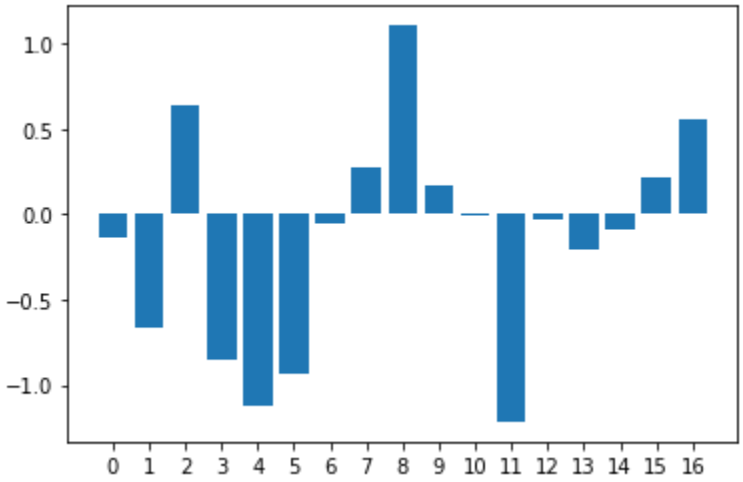
و همینطور بقیه معیار های مهم دیگر مثل accuracy و f1score که در زیر می بینیم:

	precision	recall	f1-score	support
0	0.86	0.68	0.76	91091
1	0.74	0.89	0.8	91648
accuracy			0.78	182739

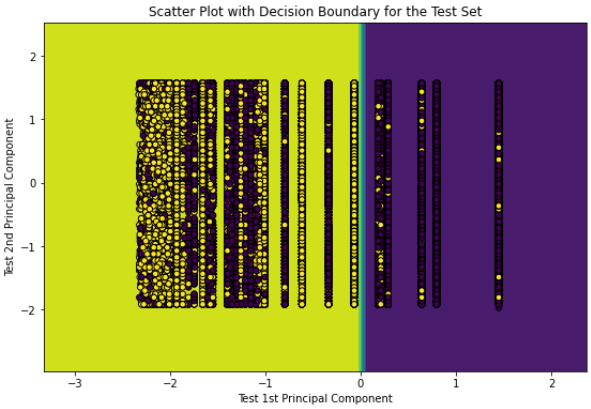
در زیر نمودار ROC curve می بینیم که این منحنی trade off بین TPR و 1 - FPR را نشان می دهد. مدل های طبقه بندی که منحنی‌هایی نزدیک‌تر به گوشه سمت چپ بالا می‌دهند، عملکرد بهتری را نشان می‌دهند. هر چه منحنی به قطر 45 درجه فضای ROC نزدیکتر شود، آزمون دقت کمتری دارد.



اما این یک معیار شهودی است و برای اینکه دقیق بتوانیم که این معیار را بررسی و با بقیه مقایسه کنیم ، برای درک بهتر از مساحت زیر سطح این منحنی استفاده می کنیم که مساحت این منحنی 0.86 شده است که هرچه به 1 نزدیک تر باشد بهتر و هرچه به 0.5 نزدیک تر باشد بدتر است.
برای بخش decision boundary اول باید 2 تا بهترین ستون ها را که برای این مدل بهتر بودند را انتخاب کنیم .



که با بررسی متوجه شدیم که ستون های 4 و 11 بیشترین تاثیر را داشته اند که یعنی ستون های app_count و hours.
مدل را روی این 2 فیچر آموزش داده و مرز تصمیم گیری را روی هم داد های آموزشی و هم ارزیابی نمایش دادیم به شکل زیر:



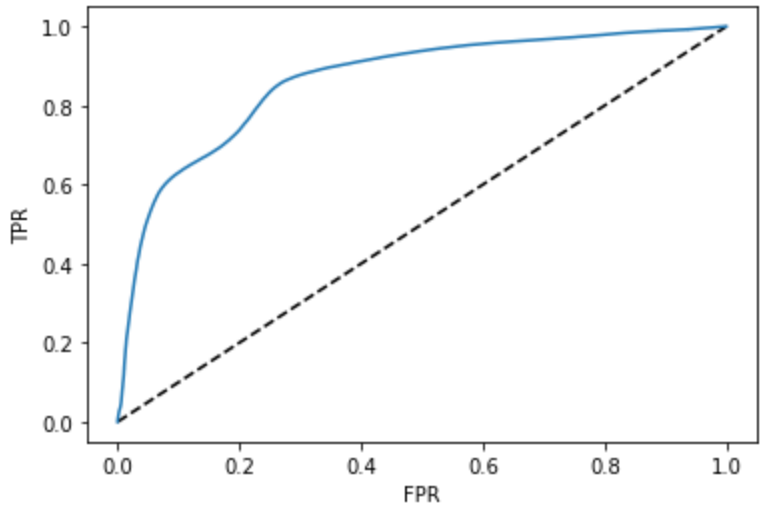
برای این مدل بار دیگر به جای اینکه داده های balance را به آن بدهیم کل داده ها را دادیم ولی از class weight در آن استفاده کردیم که ببینم چه نتایجی را خواهد داشت:
ماتریس confusion آن به شکل زیر شد:

	Predict yes	**Predict no**
Actual yes	283213	85318
Actual no	18304	73278

و همینطور بقیه معیار ها که در زیر می بینیم:

	precision	recall	f1-score	support
0	0.94	0.77	0.85	368531
1	0.46	0.80	0.59	91582
accuracy			0.77	460113

که می بینیم معیار ها برای کلاس 0 افزایش ولی برای کلاس 1 کاهش پیدا کردند که این چیزی نیست که ما از مدل انتظار داریم و به طور کل هم درستی کم شد.
در زیر نمودار ROC curve می بینیم:



که مساحت این منحنی 0.86 شده است که تفاوتی خاصی با قبلی ندارد.

2) SVM:

همانطور که میدانیم سرعت یادگیری مدل های svm بسیار پایین است نسبت به دیگر مدل ها و چون داده های ما زیاد است باید دوباره تعدادی از آنها را انتخاب کنیم برای اینکه به مدل بدهیم . ما از داده ها به شکل تصادفی یک sample ده هزار تایی گرفتیم و svm را روی آنها train کردیم.

ماتریس confusion آن به شکل زیر شد:

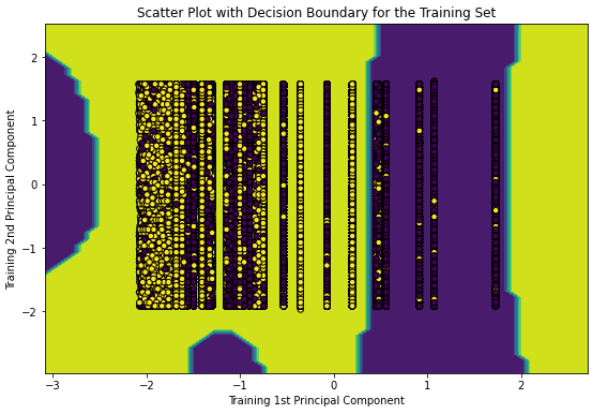
	Predict yes	**Predict no**
Actual yes	67712	23379
Actual no	12307	79341

و همینطور بقیه معیار ها که در زیر می بینیم:

	precision	recall	f1-score	support
0	0.85	0.74	0.79	91091
1	0.77	0.87	0.82	91648
accuracy			0.80	182739

در svm مشخص است که به نسبت عملکرد بهتری را داشتیم در مقایسه با logistic regression.

دوباره داده ها را روی همان 2 فیچر (app_count و hours) آموزش داده و مرز تصمیم را نمایش دادیم:



3) KNN:

برای این مدل هایپر پارامتر n را 5 در نظر گرفتیم به این معنی که برای هر داده جدید 5 داده اطراف آن را می بیند برای اینکه تشخیص دهد جزو کدام کلاس است.

داده های کل را به مدل دادیم برای یادگیری و برای ارزیابی مدل تمام داده های تست را ندادیم چون که سرعت طبقه بندی مدل KNN به شدت پایین است پس بخشی از داده ها را به شکل تصادفی دادیم تا پیشبینی کند.

ماتریس confusion آن به شکل زیر شد:

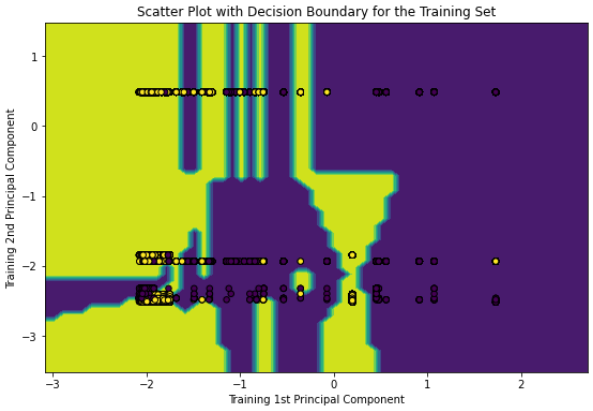
	Predict yes	**Predict no**
Actual yes	4248	732
Actual no	734	4286

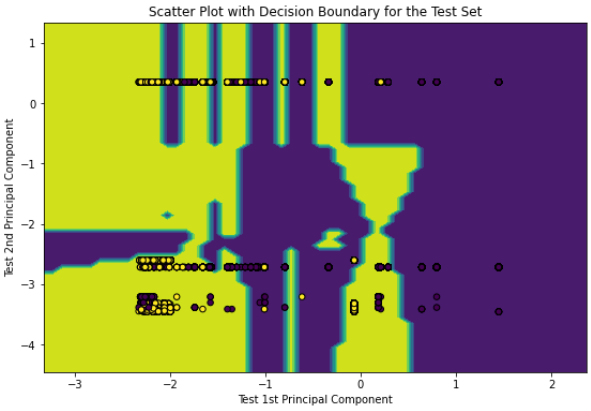
و همینطور بقیه معیار ها که در زیر می بینیم:

	precision	recall	f1-score	support
0	0.85	0.85	0.85	4980
1	0.85	0.85	0.85	5020

accuracy			0.85	10000

مشخص است که این مدل به نسبت مدل های قبلی بهتر بود هم از نظر balance بودن بین 2 کلاس هم از نظر درستی 0.85 .
در ادامه چون نتوانستیم ویژگی های بهتر را پیدا کنیم برای این مدل ، دو ستون app_count و device_count که عملکرد بهتری داشتند را به مدل دادیم برای یادگیری و در آخر هم مرز تصمیم گیری که به شکل زیر است:





4) Decision Tree:
این مدل را روی داده های آموزشی train کردیم که نتایج زیر حاصل شد.
ماتریس confusion آن به شکل زیر شد:

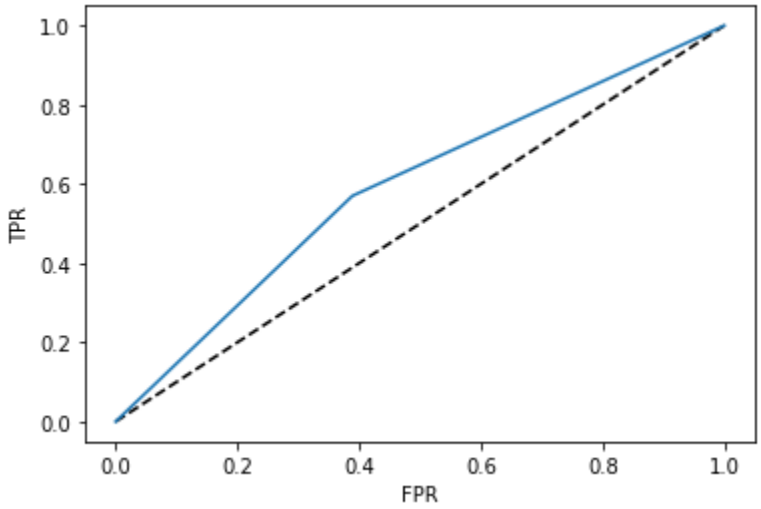
	Predict yes	**Predict no**
Actual yes	55655	35436
Actual no	39413	52235

و همینطور بقیه معیار ها که در زیر می بینیم:

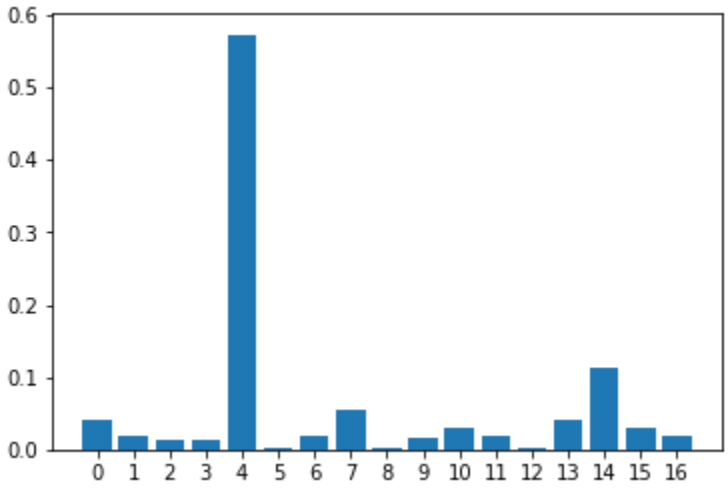
	precision	recall	f1-score	support
0	0.59	0.61	0.60	91091
1	0.60	0.57	0.58	91648

accuracy			0.59	182739

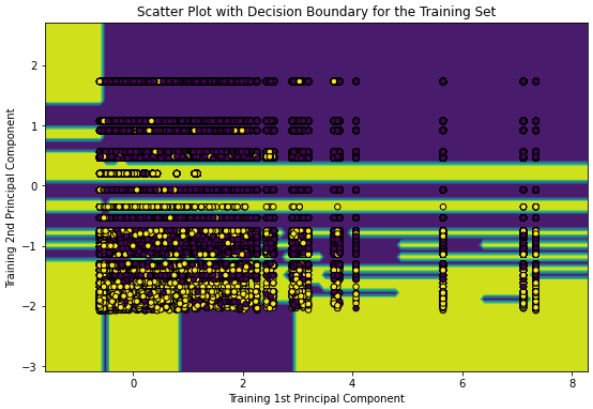
که می بینیم برای این طبقه بندی به شدت نتایج ضعیف شده اند و عملکرد یکم بهتر از random دارد که اصلا خوب نیست.
در زیر نمودار ROC curve می بینیم:

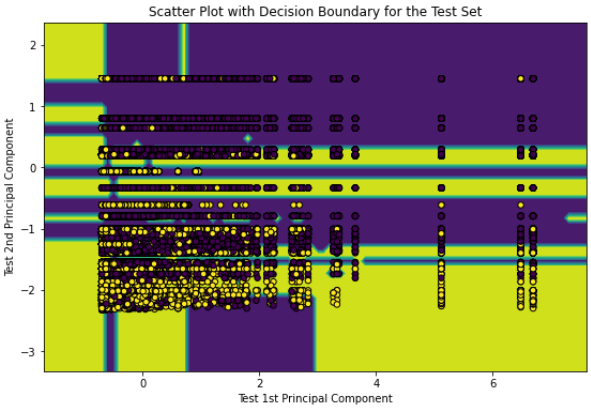


که مشخص است به قطر 45 درجه فضای ROC نزدیک است و مساحت این منحنی 0.59 شده است.
برای decision boundary هم 2 تا بهترین ستون ها را که برای این مدل بهتر بودند را انتخاب کردیم .



که ستون های 4 و 14 یعنی app_count و ip*device بیشترین تاثیر را داشتند و مرز تصمیم روی این 2 ویژگی برای این مدل به شکل زیر شد:





5) Random Forest:

این مدل را با برای هر درخت با ماکزیمم عمق 3 در نظر گرفتیم و داده ها را train کردیم.

ماتریس confusion آن به شکل زیر شد:

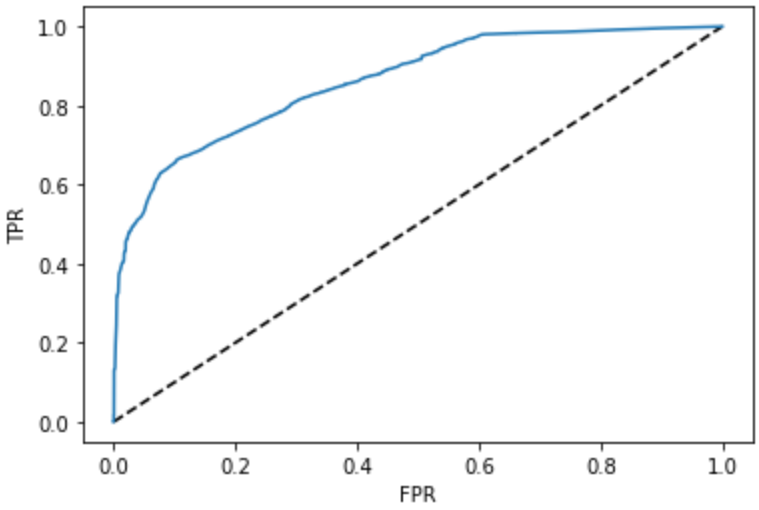
	Predict yes	**Predict no**
Actual yes	68139	22952
Actual no	21186	70462

و همینطور بقیه معیار ها که در زیر می بینیم:

	precision	recall	f1-score	support
0	0.76	0.75	0.76	91091
1	0.75	0.77	0.76	91648
accuracy			0.76	182739

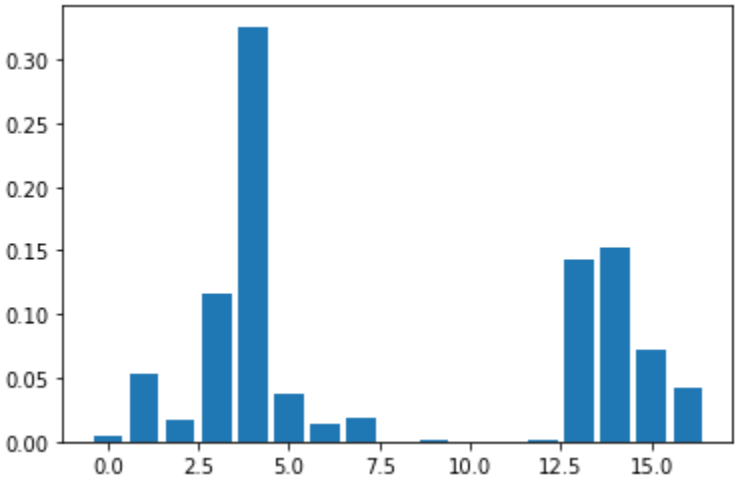
که می بینم نسبت به درخت تصمیم عملکرد بهتری داشته است.

در زیر نمودار ROC curve می بینیم:

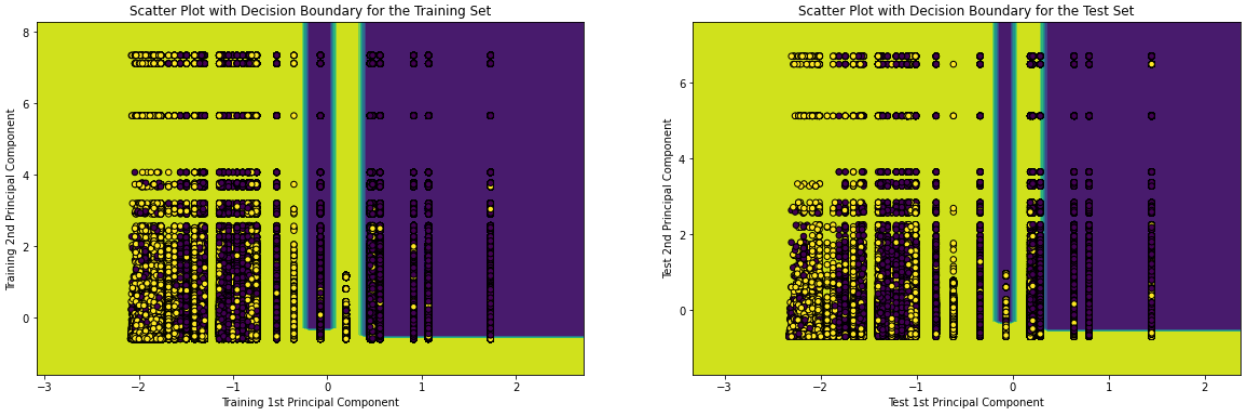


که مساحت این منحنی 0.86 شده است.

برای decision boundary هم 2 تا بهترین ستون ها را که باز هم ستون های app_count و ip*device بودند را انتخاب کردیم .



نمایش مرز تصمیم برای این مدل به شکل زیر شد:



6) Naive Bayes:

برای این مدل از مدل گاوسی استفاده کردیم و داده ها را با آن train کردیم.

ماتریس confusion آن به شکل زیر شد:

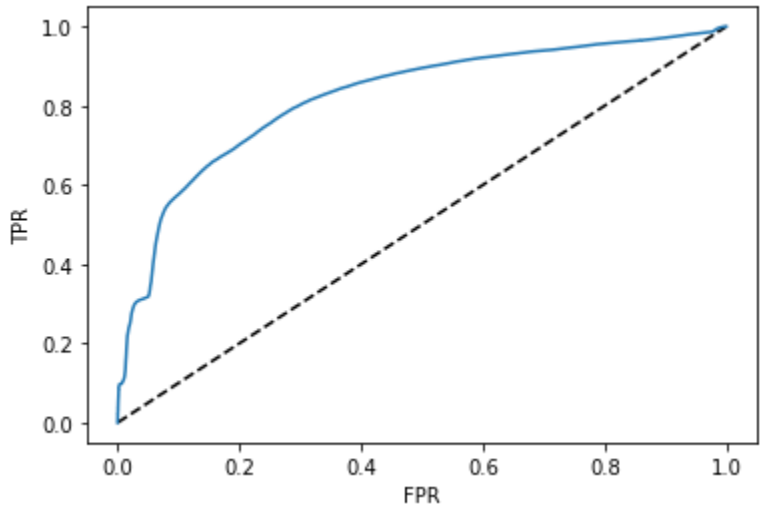
	Predict yes	**Predict no**
Actual yes	30784	60307
Actual no	6098	85550

و همینطور بقیه معیار ها که در زیر می بینیم:

	precision	recall	f1-score	support
0	0.83	0.34	0.48	91091
1	0.59	0.93	0.72	91648
accuracy			0.64	182739

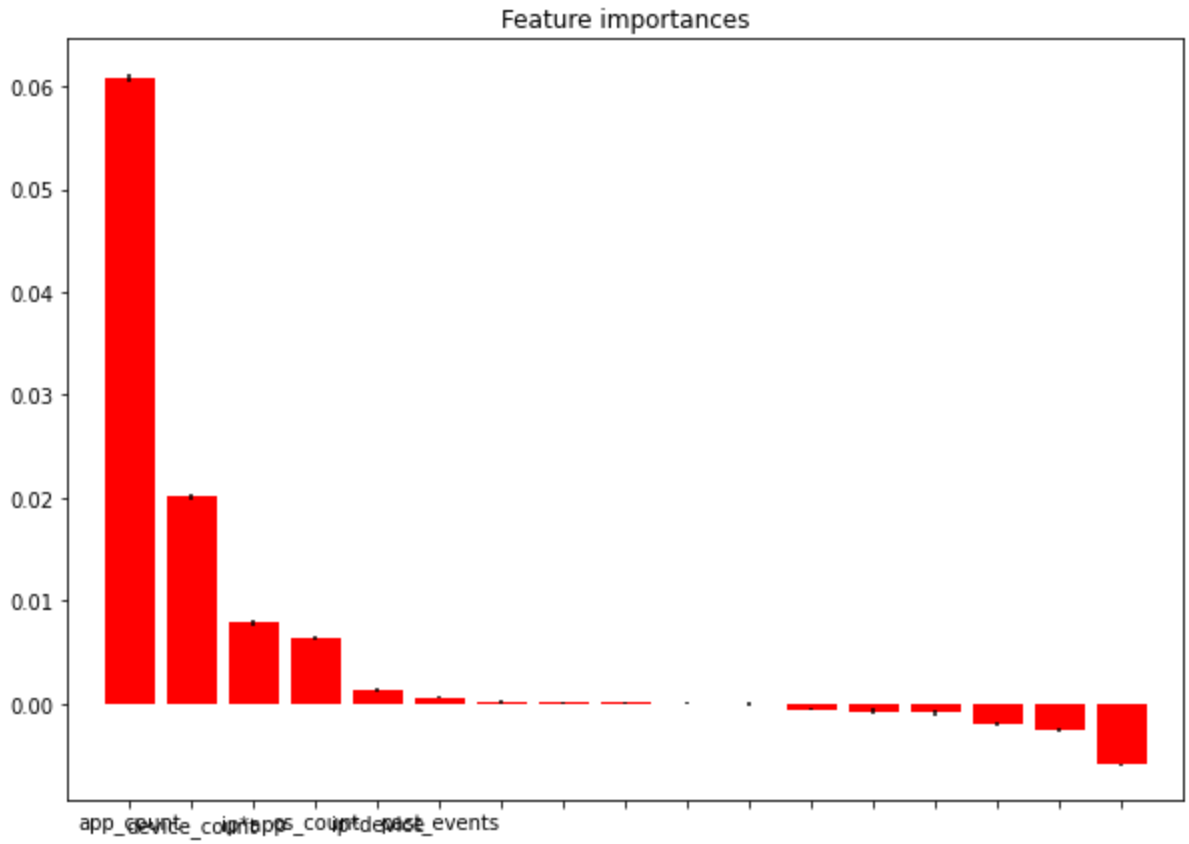
که می بینم عملکرد این مدل بجز از درخت تصمیم از بقیه مدل ها ضعیف تر است.

در زیر نمودار ROC curve می بینیم:

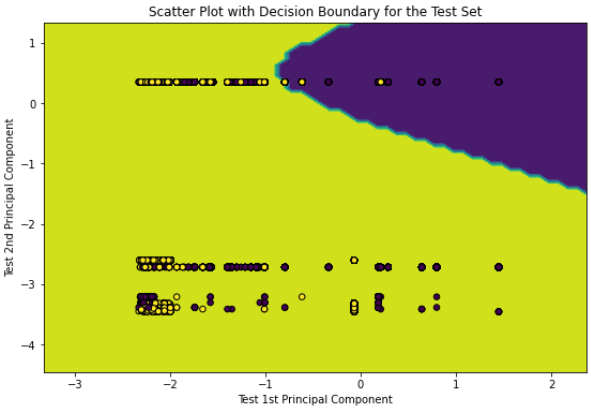


که مساحت این منحنی 0.82 شده است که این مقدار بدی نیست.

برای decision boundary هم 2 ستون app_count و device_count را که بهتر بودند انتخاب کردیم .



نمایش مرز تصمیم برای این مدل به شکل زیر شد:



با توجه به نتایج بالا کامل مشخص است که balance کردن داده ها نقش مهمی در generalization مدل دارد و همینطور انتخاب فیچر های مهم و ساخت فیچر بسیار اهمیت دارد. انتخاب یک مدل درست بر اساس داده ها هم نقش خوبی در نتایج نهایی دارد که همانطور که دیدیم مثلا مدل KNN نتایج خوبی داشت ولی مدل Decision Tree برای پیشبینی این داده ها اصلا خوب نبود.