بسمه تعالی

**دانشگاه شهید بهشتی**

**گزارش تمرین سری چهارم**

**درس مبانی داده کاوی و کاربرد های آن**

**سید حسین محمدی - 96222096**

خرداد 99

فهرست مطالب

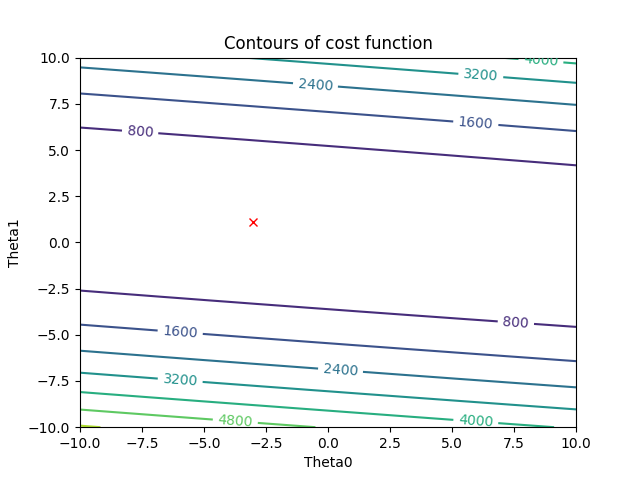
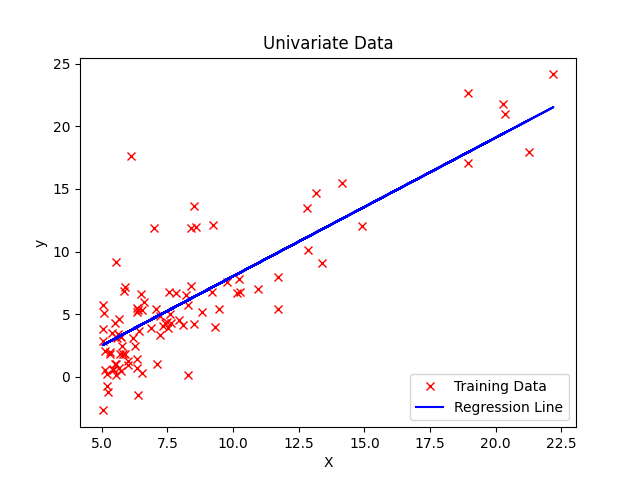
[بخش اول: 1](#_Toc42176547)

[بخش دوم: **Error! Bookmark not defined.**](#_Toc42176548)

## بخش اول:

The places that had a to-do list were filled using the derivative of matrix equation for MSE which is

this are the results for the univariant data using the python file provided by the exercises:



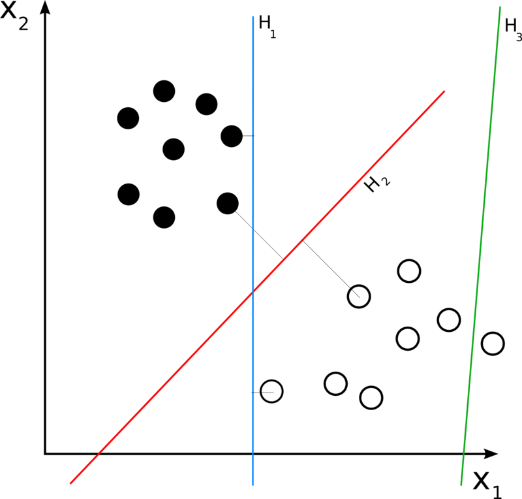
And for the multivariant, no plot is provided but the RSME is 22.5% for the testing dataset.

## بخش دوم:

1. ماشین بردار پشتیبان[[1]](#footnote-1) (به اختصار SVM) الگوریتم نظارت شده ای است که برای دو عمل دسته بندی و رگرسیون استفاده می شود. مبنای کاری این الگوریتم دسته بندی خطی داده ها است و عملکرد آن به صورتی است که خطی انتخاب شود که بیشترین حاشیه اطمینان را دارد. از آنجایی که این الگوریتم نظارت شده است، نیاز به داده های با برچسب دارد و بنابراین برای دیتاست هایی استفاده می شود که دارای برچسب متغیر هدف می باشند. ماشین بردار پشتیبان عملکرد نزدیکی به شبکه های عصبی[[2]](#footnote-2) دارد، به نحوی که یک SVM بدون هسته[[3]](#footnote-3) یک نرون شبکه عصبی با تابع هزینه متفاوت است.

بردار های پشتیبان[[4]](#footnote-4) مجموعه ای از نقاط در فضای چند بعدی داده ها است که مرز بین دسته ها را مشخص می کند. ابعاد این بردار همواره یک واحد کمتر از ابعاد داده ها می باشد. برای مثال در فضای دو بعدی بردار پشتیبان یک خط و در فضای سه بعدی بردار پشتیبان یک صفحه می باشد. هدف اصلی این الگوریتم یافتن بهترین مرزبندی میان داده های ورودی است، به گونه ای که بیشترین حاشیه[[5]](#footnote-5) (فاصله) ممکن را از تمام دسته ها داشته باشد.

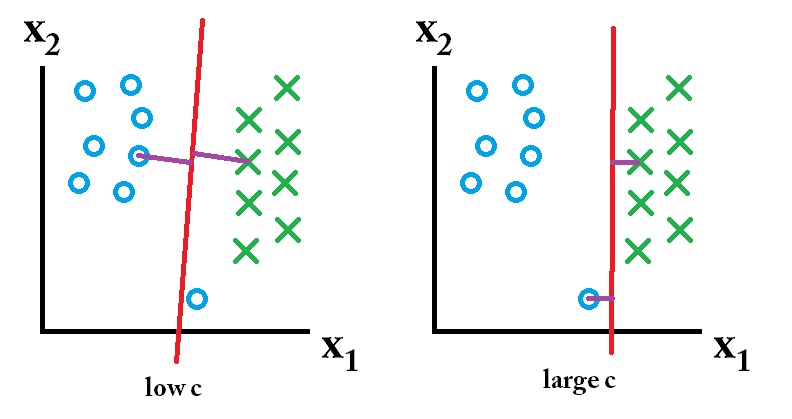
به عنوان مثال در شکل فوق، در صورتی که دو دسته داده های سفید و سیاه را داشته باشیم، خط H3 داده ها را به خوبی تقسیم نمی کند و بنابراین به عنوان یک بردار پشتیبان مورد قبول نیست. دو بردار H2 و H1 بردار های پشتیبان هستند که از بین آنها، H2 عملکرد بهتری دارد چرا که دو دسته را با بیشترین حاشیه ممکن جدا می کند. بردار H1 نیز این دو دسته را با حاشیه بسیار کمی جدا می کند که مطلوب نیست. به طور کلی، از آنجایی که داده های ورودی ممکن است دارای نویز و یا اعوجاج باشند، این الگوریتم تلاش می کند تا بردار پشتیبانی را انتخاب کند که بیشترین حاشیه ممکن را (که به آن حاشیه اطمینان نیز گفته می شود) داشته باشد.



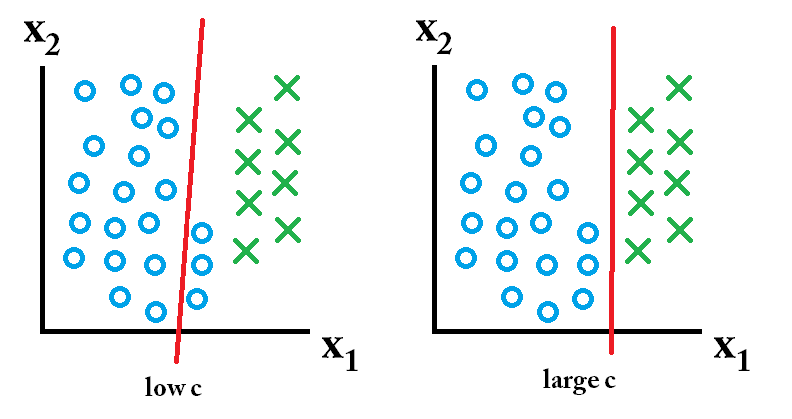
در صورتی که داده های بردار های آموزشی باشند که در دو دسته قرار دارند، هدف SVM پیدا کردن و است، به نحوی که تابع تصمیم گیری زیر برای همه (و یا اکثر) نمونه ها پیش بینی درستی انجام دهد:

پارامتر C که به آن پارامتر رگولاریزاسیون نیز می گویند، ضریبی است که هزینه مربوط به دسته بندی اشتباه داده ها را در فرایند بهینه سازی بیان می کند. در صورتی که مقدار آن افزایش یابد، داده هایی که با خطا دسته بندی شده اند تاثیر بیشتری در تابع خطا[[6]](#footnote-6) خواهد داشت و در صورتی که به بی نهایت میل کند، جواب نهایی به اصطلاح Hard Margin می شود. در صورتی که این پارامتر به صفر میل کند، تعداد دسته بندی های نادرست افزایش می یابد. بنابراین، مقادیر بزرگ C باعث می شود که الگوریتم به دنبال انتخاب بردار ها با حاشیه کمتری باشد، مادامی که تمام داده های ورودی به درستی دسته بندی شوند. مقادیر خیلی کوچک C نیز باعث می شود تا الگوریتم به دنبال انتخاب بردار ها با حاشیه بزرگتری باشد، حتی اگر در این حالت نقاطی با خطا دسته بندی شوند.

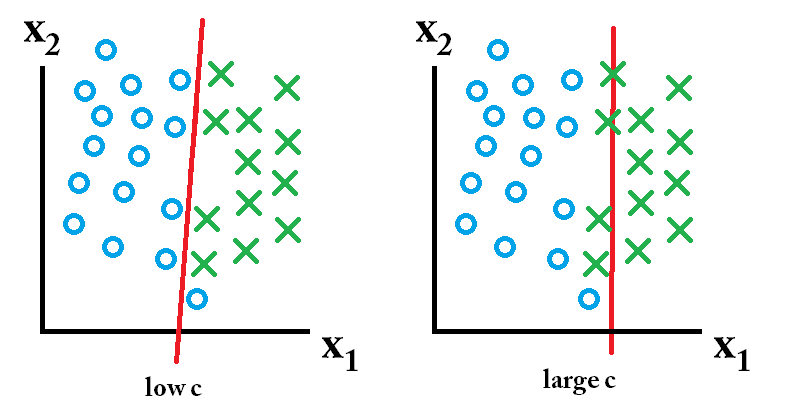
انتخاب این پارامتر به داده های ورودی بستگی دارد. برای مثال در شکل زیر عملکرد یک ماشین بردار پشتیبان در دو حالت با پارامتر C کم و زیاد را مشاهده می کنیم. در صورتی که مقدار پارامتر C کم باشد (شکل چپ)، حاشیه بین داده ها و خط زیاد شده که مطلوب ماست، اما یکی از داده های دسته آبی رنگ که داده پرت محسوب می شود، به اشتباه طبقه بندی می شود. در صورتی که مقدار این پارامتر زیاد باشد، تمامی داده ها به درستی دسته بندی می شوند اما حاشیه بسیار کم می شود.



حال کدام یک از این دو مدل بهتر است؟ این موضوع به داده هایی که قصد پیش بینی و دسته بندی آنها را داریم بستگی دارد. برای مثال در صورتی که داده ها به صورت زیر باشند، مدلی که پارامتر C آن بزرگتر است بهتر عمل می کند.



و در صورتی که داده ها به صورت زیر باشند، مدلی که پارامتر C آن کوچکتر است، عملکرد بهتری دارد.



بنابراین مقدار به خصوص و مشخصی برای این پارامتر، که عملکرد مدل را بهبود دهد وجود نداشته و تاثیر آن بر روی مدل به داده های ورودی و همچنین داده های تست ما بستگی دارد.

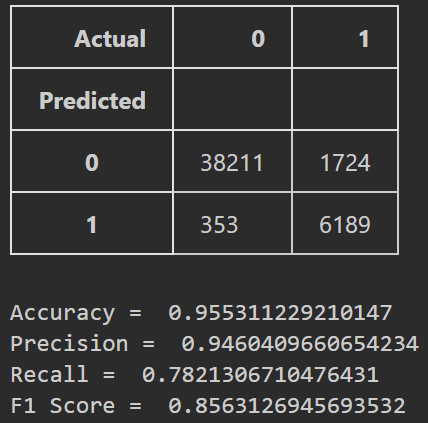
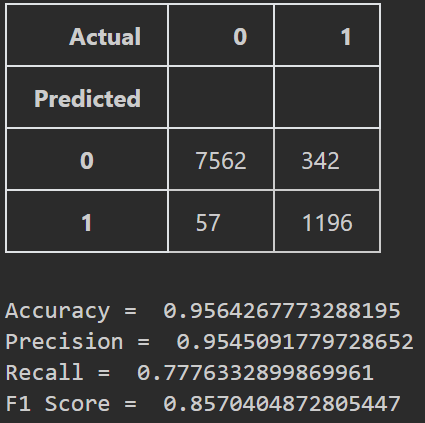
## گام اول:

at first, we remove 2% endpoints of our sampled data (92 columns) which are extreme outliers. Then we curb the next 7% of the data to be less extreme and scale them between 0,1 using min max on the new 2% quintile.

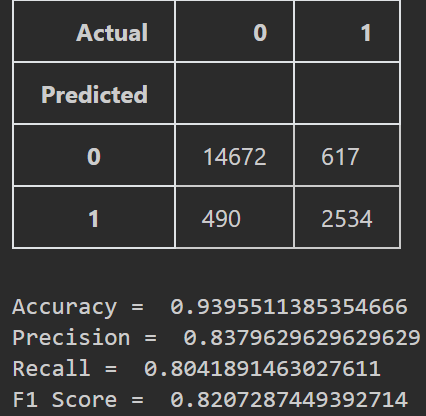
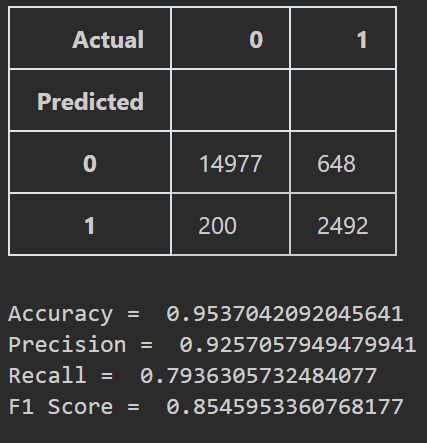
After wards we use PCA to compress our data and use L1 to remove unnessery pca components which removes nearly 40% of the columns, leaving 55 components. And Finaly we use undersmaple out data and remove 10% of the majority class and then uses smote to overample our data. The parameters for both the smote function and the classifier are decided using a grid search which will maximise the F1 score which means it will find the best balance btween precesion and recall.

Theses are the results (sample sizes differ to increase speed):

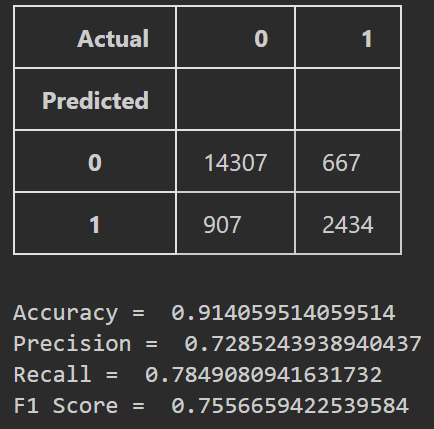
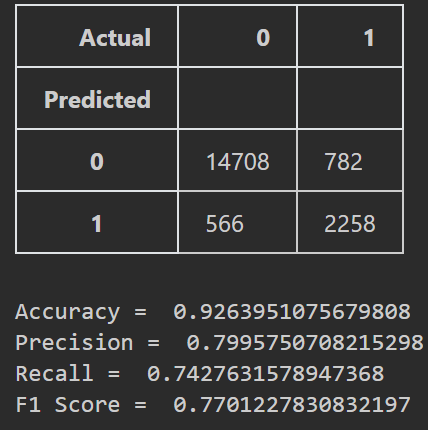
LogiticReg(85.6%): SVC(85.7%):

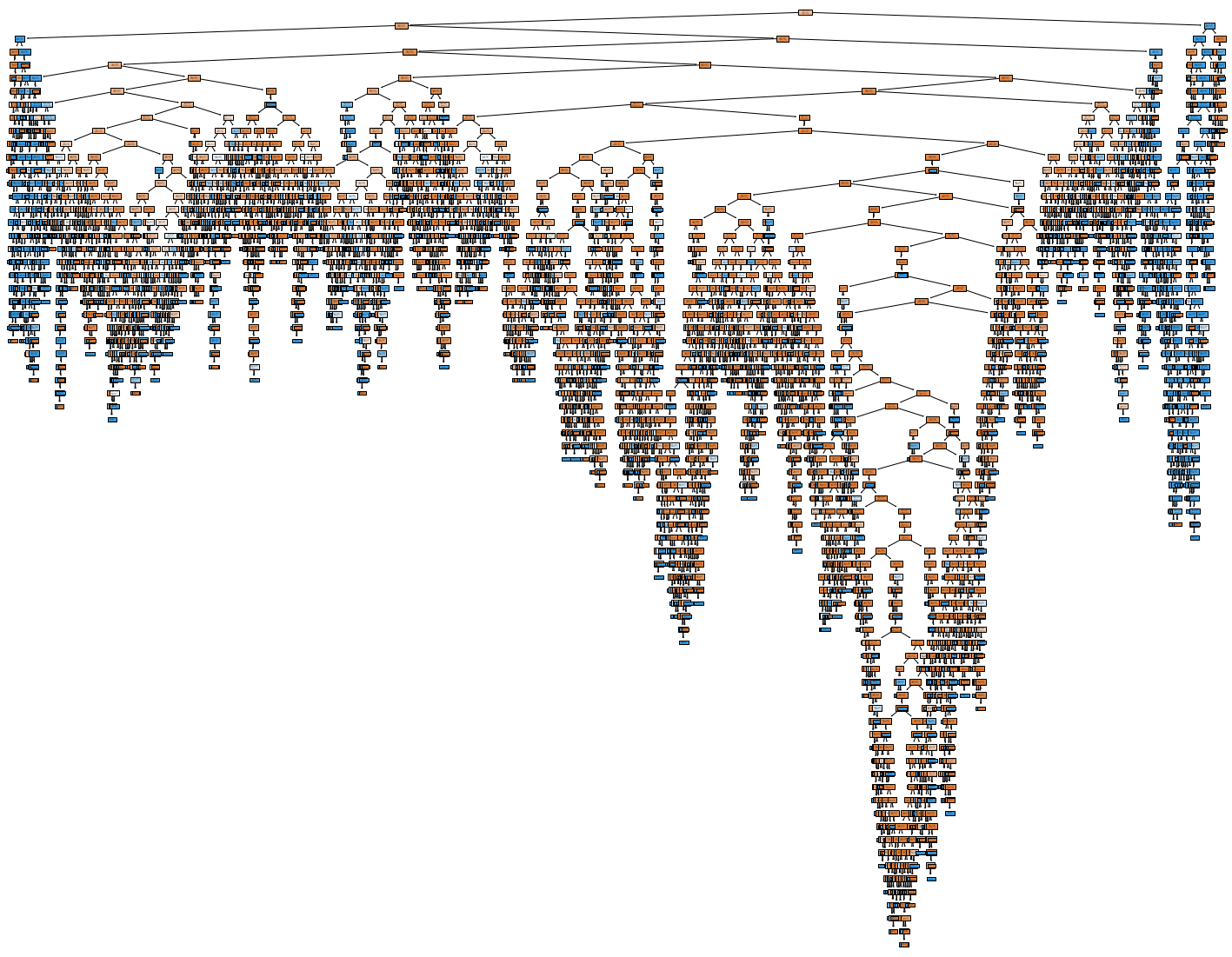
KNeighbors(82.1%): RandomForest(85.4%):

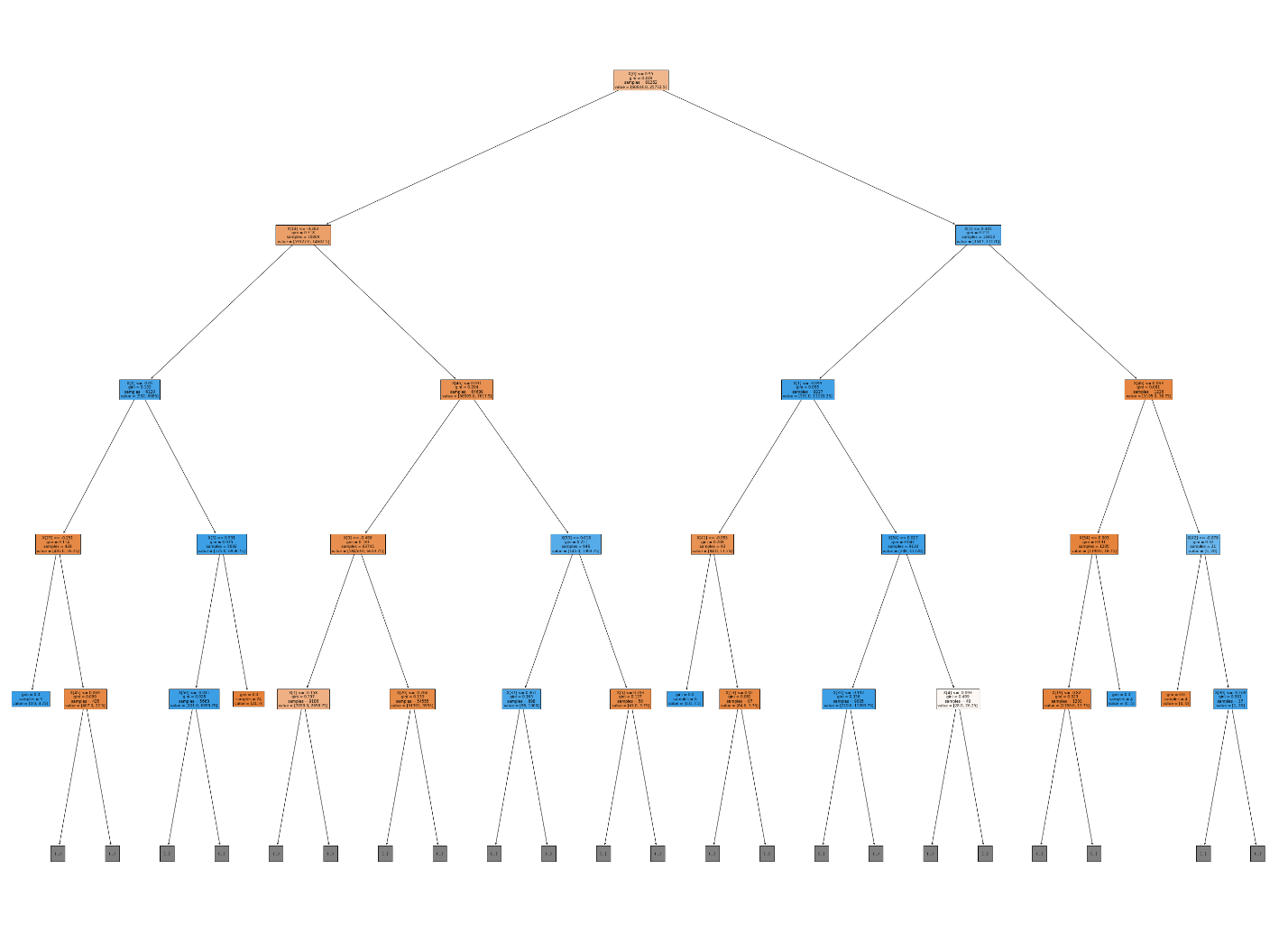
DecisionsTree(75.6%): NB(77%):

Using Decision tree visualization, we find out that X0 and X7 are mosst important componenets:

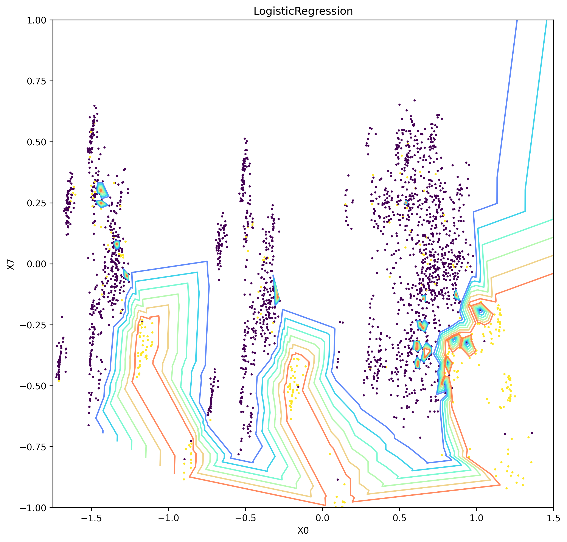
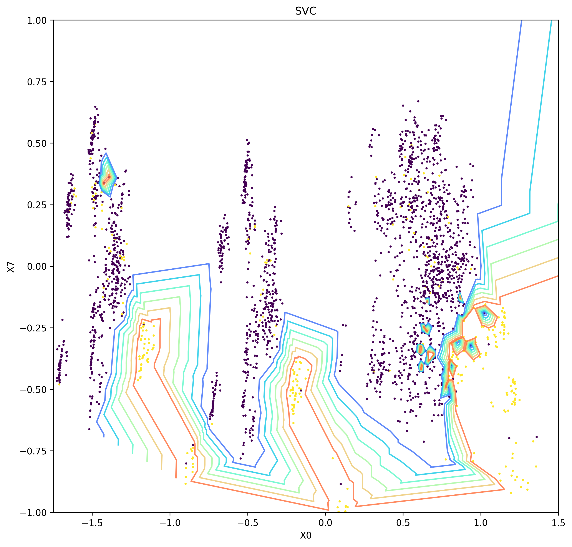


First 4 layers:

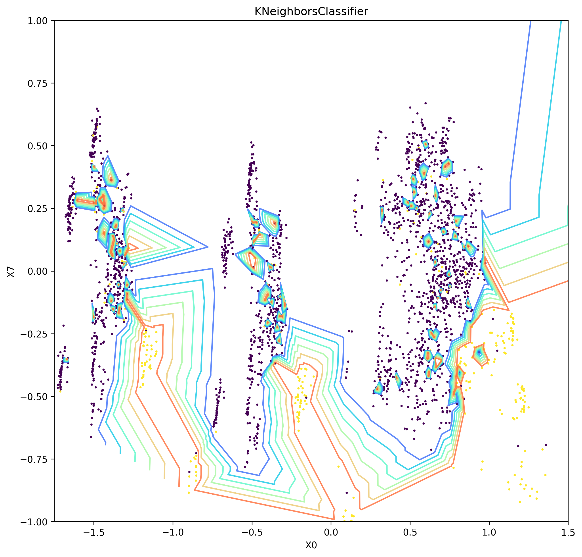
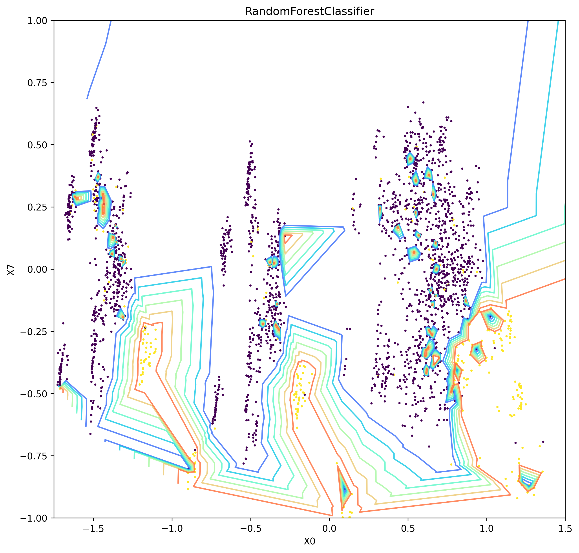


Now uing this 2 axis, we predict using our models and plot contor graph to visualize the boundrys:

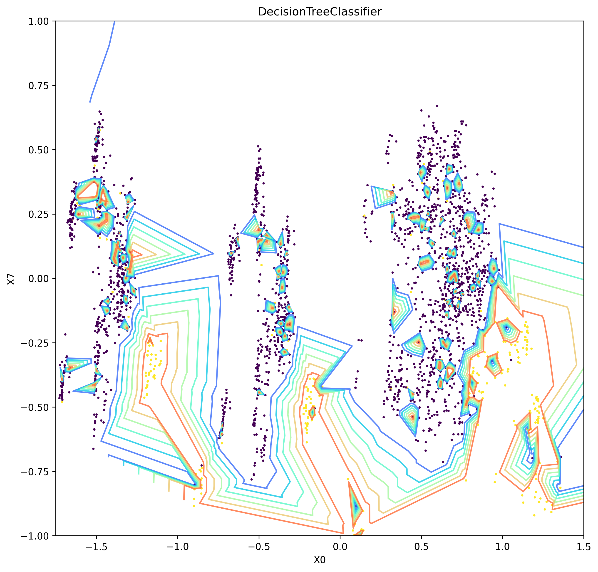
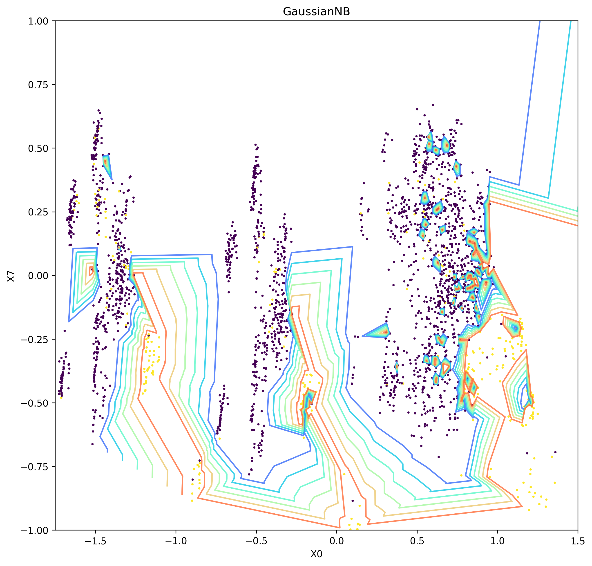
LogiticReg(85.6%): SVC(85.7%):

KNeighbors(82.1%): RandomForest(85.4%):

DecisionsTree(75.6%): NB(77%):

Most of theses models are overfiting and try to draw around outliers. This isss specially visable for NB and DecisionTree.   
One solution would be to reduce the columns used for PCA which is pretty easy to do.   
On other thing is to control epochs, but this can be very hard to do and such a hyperparameter might not even generlise very well if set optimally

SVC seem logisticreg seem fit better than others but SVC is extremly slower than LogiReg so we could say the best model is a logisticreg.

1. Support Vector Machine [↑](#footnote-ref-1)
2. Neural Networks [↑](#footnote-ref-2)
3. Kernel [↑](#footnote-ref-3)
4. Support Vectors [↑](#footnote-ref-4)
5. Margin [↑](#footnote-ref-5)
6. Loss Function [↑](#footnote-ref-6)