Statistical Learning for Big Data

Parsia Basimfar

Classification and Variable Selection

Exercise 1

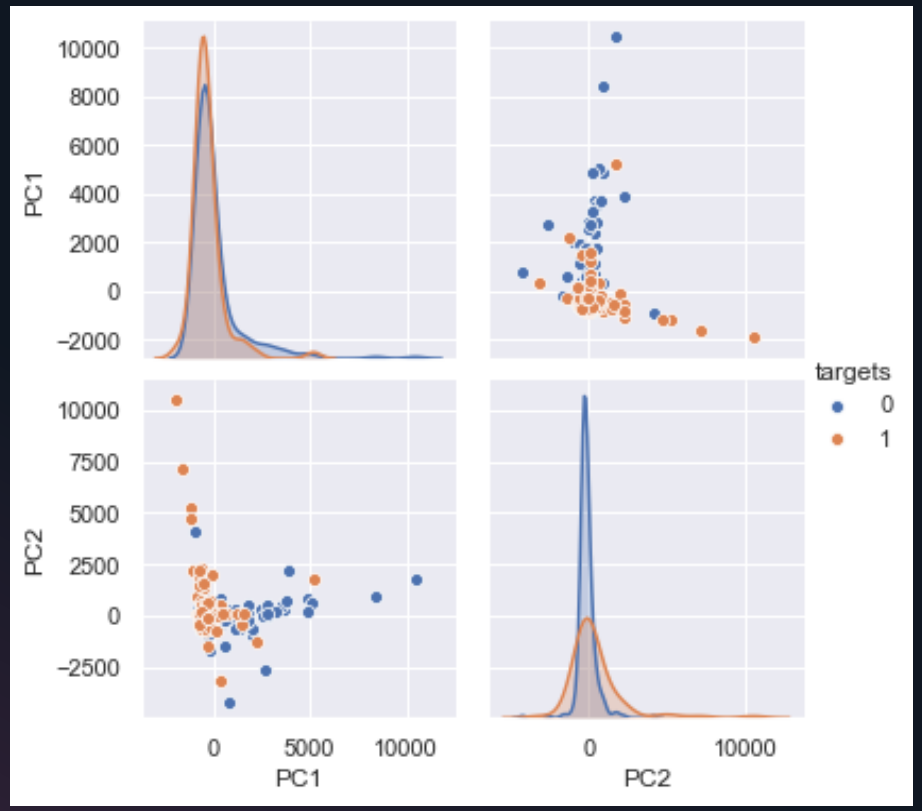
Methods and Exploratory Data Analysis

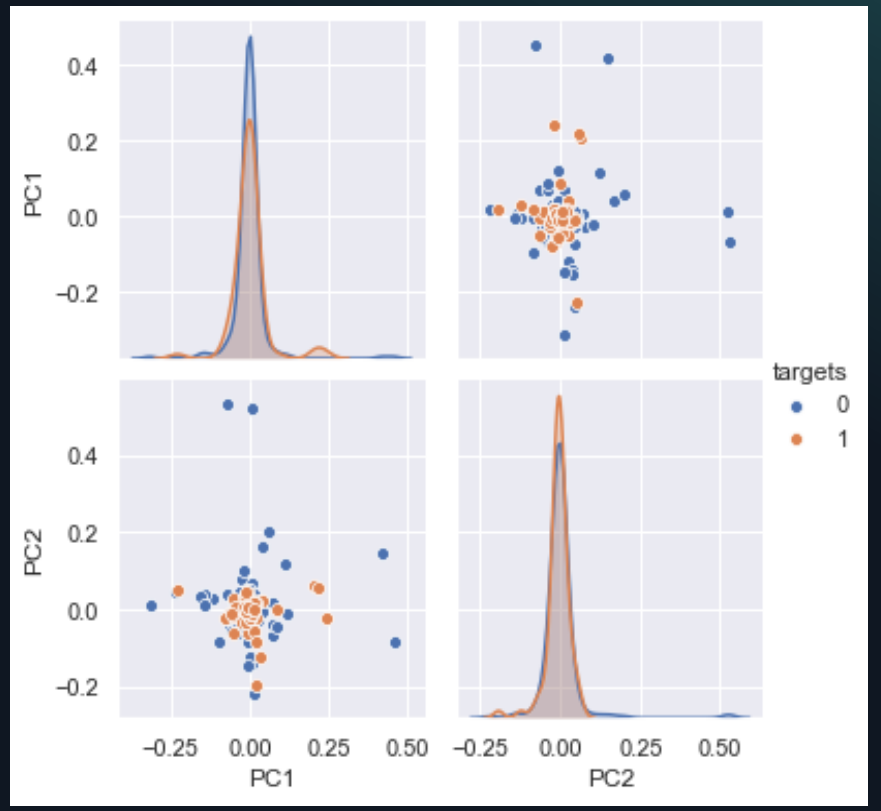
- I run PCA and kernel PCA on the data to be able to visualize the data and check if the data can be rendered linearly separable. Furthermore I run some plots to look at cluster densities (if there are any) and try to fit linear regressions to some of the features to confirm the inseparability of the data.
- Since we want to select the most important features we need methods that enable us. For this task I chose elastic net as it combines L1 and L2 regularizations and thus can be an improvement over the LASSO for example. Particularly, when no prior knowledge exists over the data elastic net is preferred. Moreover, elastic net is better at handling overfitting because it 'sparsifies' the features, and the visualizations suggest that classifiers might suffer it when predicting this particular dataset.
- The other method I choose is the random forests which enables one to use Gini impurities to rank the importance of features. I use the accuracy metric to compare their performances. Specifically I use LogisticRegressionCV and GridSearchCV functions of the scikit learn to tune the hyperparameters, for elastic net and random forests respectively.

- To choose the most important features in elastic net I proceed in two manners. In the first procedure I repeat the result of the regression (with best hyperparameters) 200 times (with resampling so I perform bootstrapping) and see how many times the predictors were estimated non zero over the 200 runs. The coefficients with most number of selection are chosen as the selected variables and I choose their importance based on their magnitude.

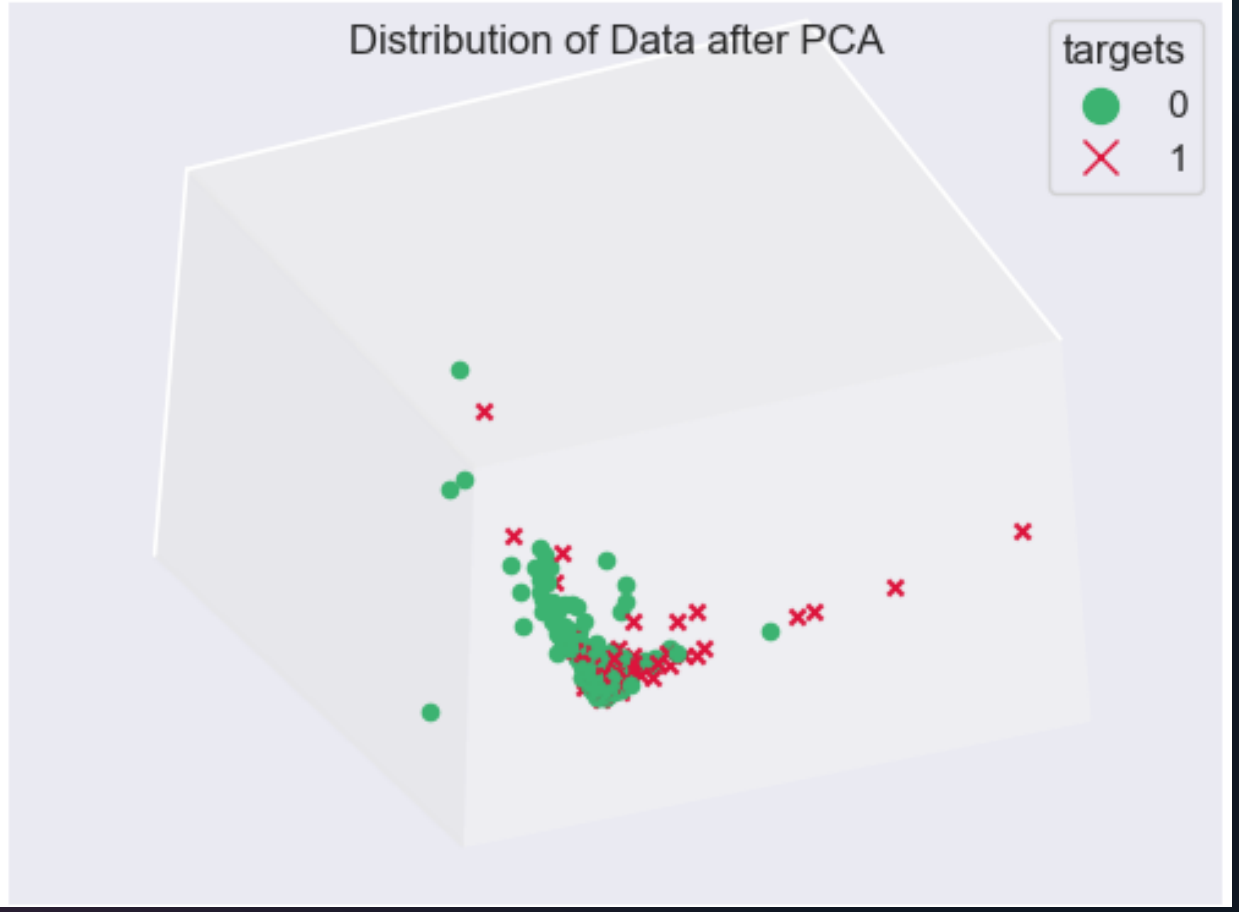
- In the other approach, I decrease C (1/α) to strengthen regularization and the features that persist to be non-zero are the selected features. The more a feature persists (due to high regularization) the more important it is ranked.

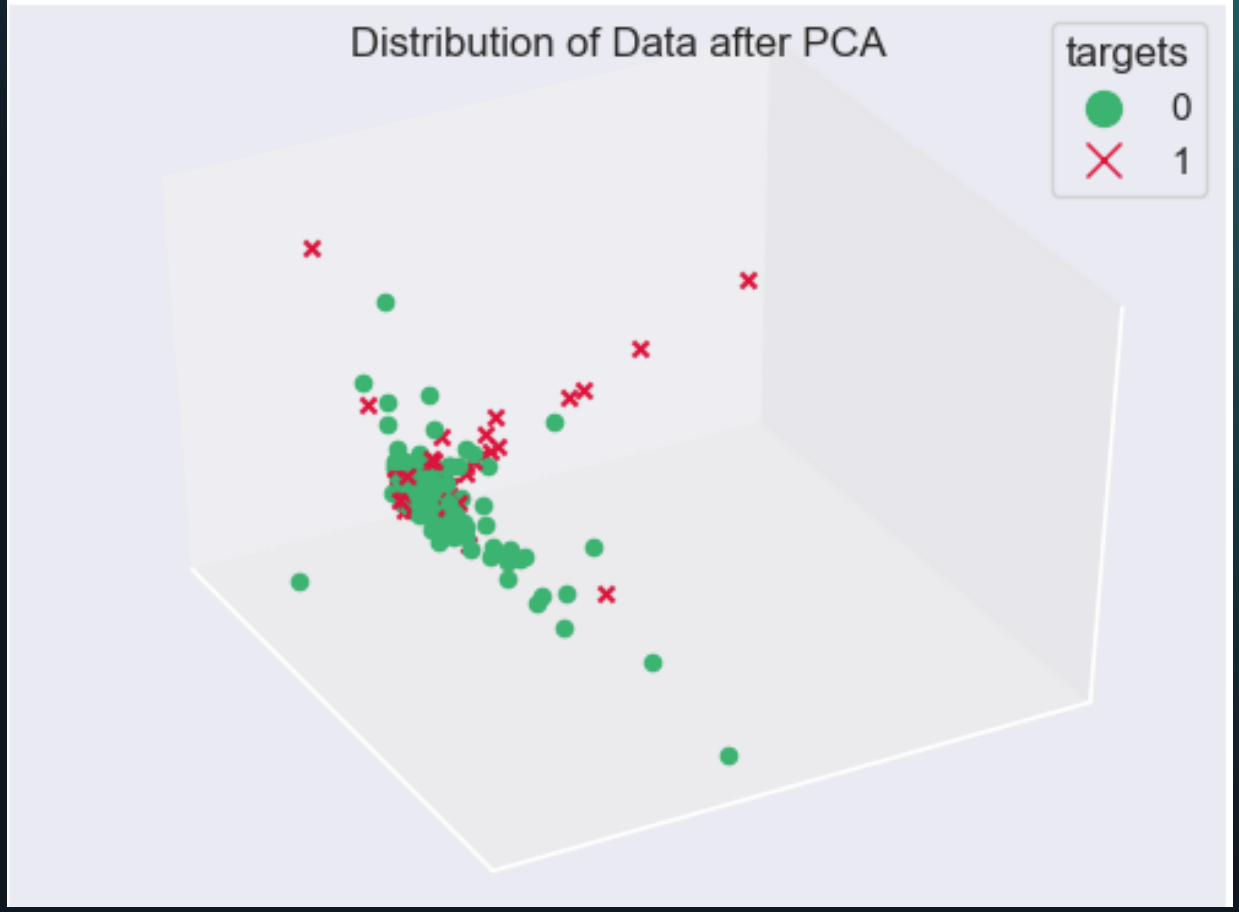
- Choosing the most important features based on the random forest classifier is an easy task as the Gini impurities give us a direct measure of importance of each feature.



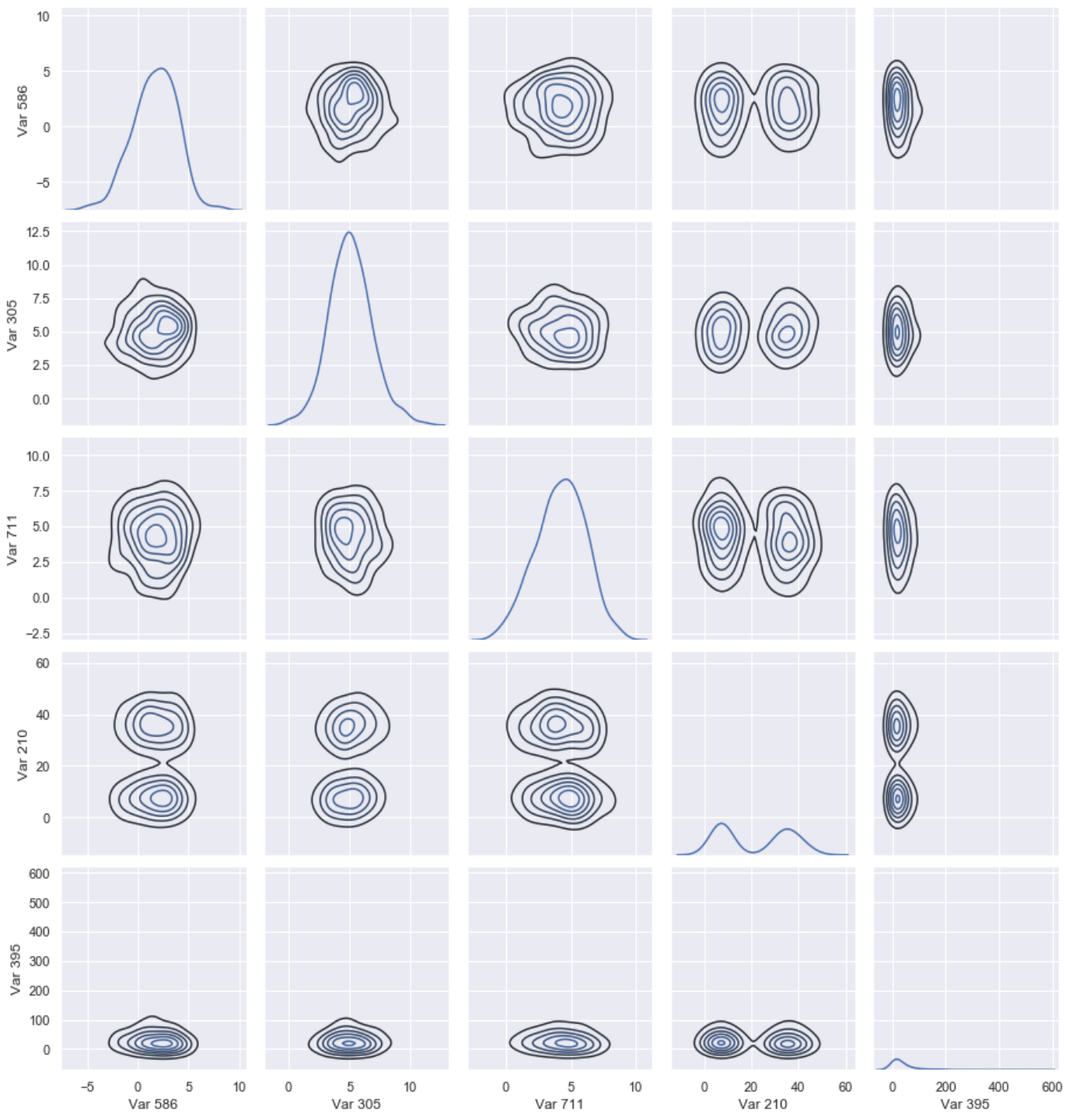


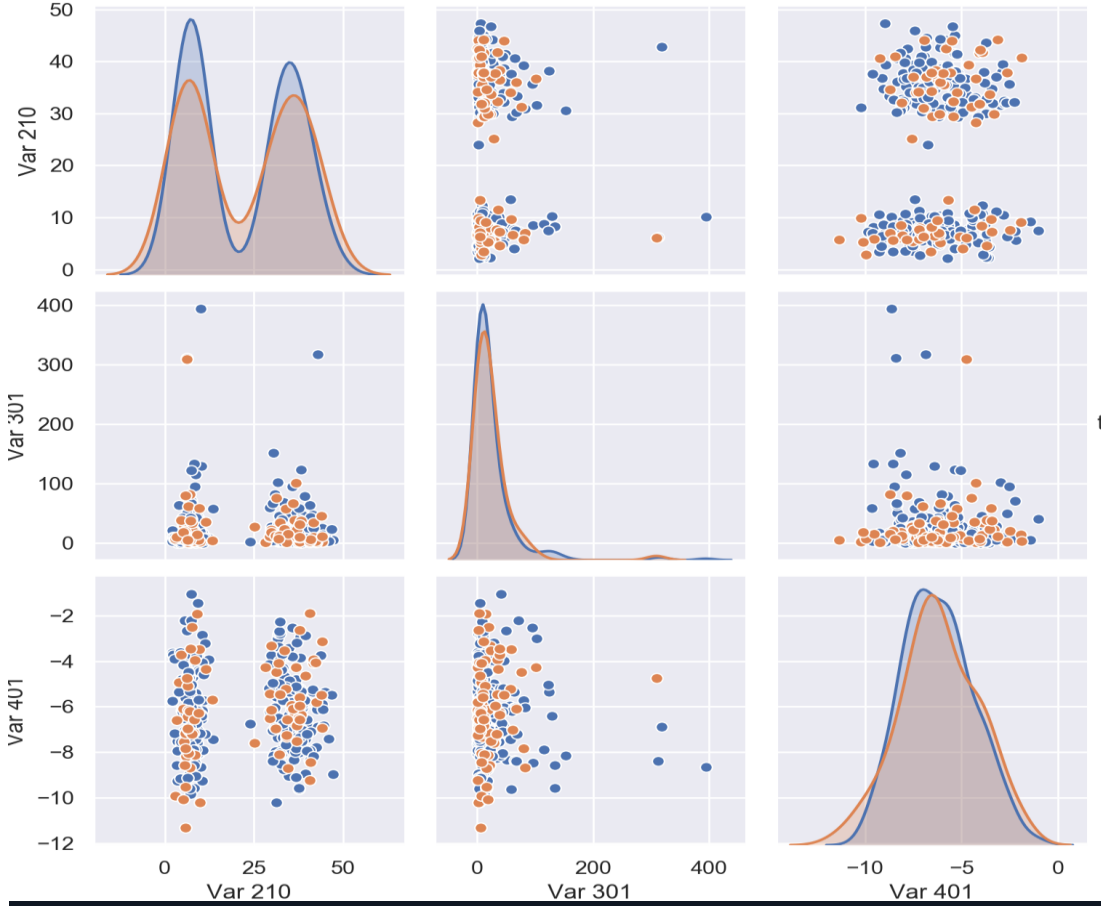
The figure to the left is ordinary PCA and the one to the right is kernel PCA with the RBF kernel (other kernels showed no significant difference). These figures suggest that data is not linearly separable. Moreover there is not a distinct boundary between the classes and some overlap exists. Several outliers can be detected in the data.



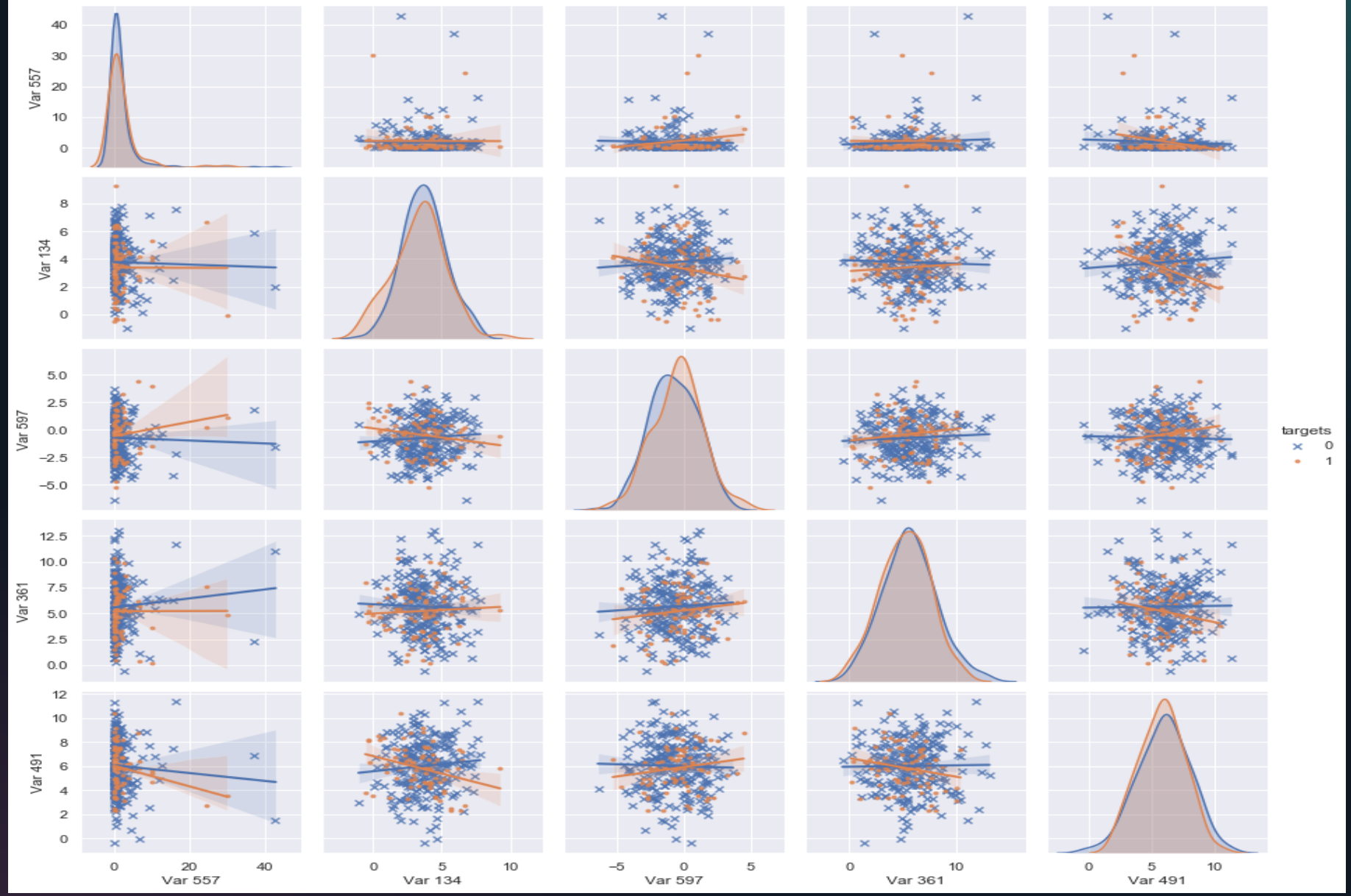


PCA with 3 components somewhat better confirms that the class boundaries overlap and are not clearly separable. The first 3 PCs explain 16.24, 11.12 and 10.36 percent of the data variance respectively. From here on I will use the terms variable and feature interchangeably.

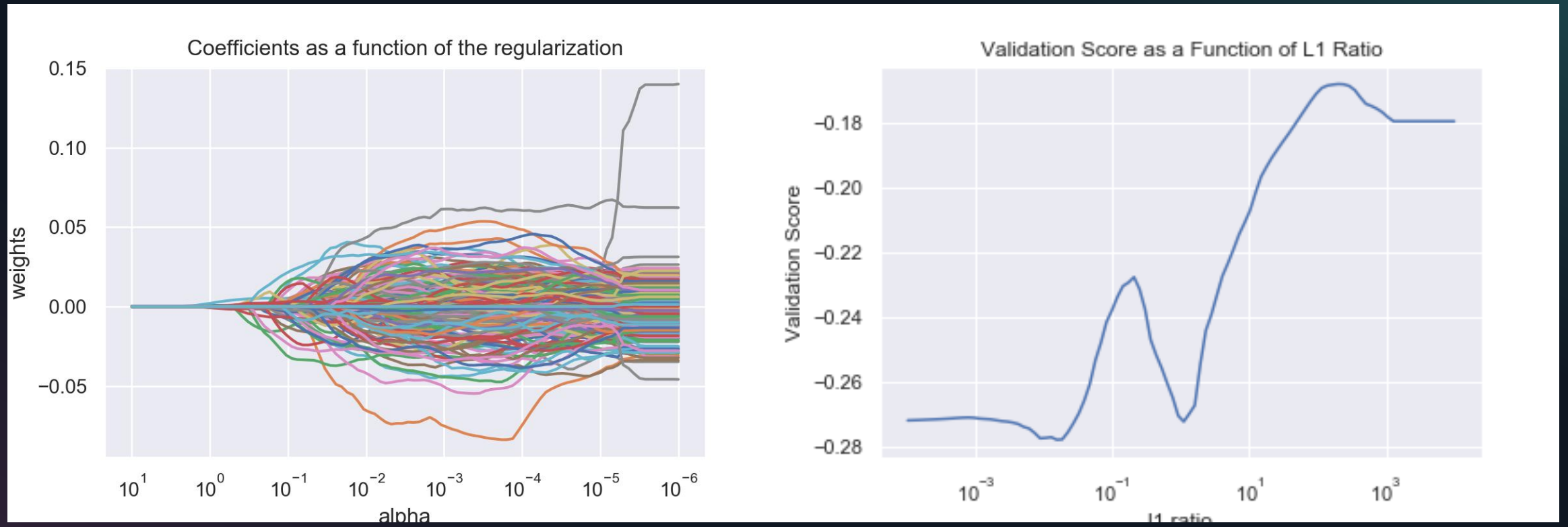




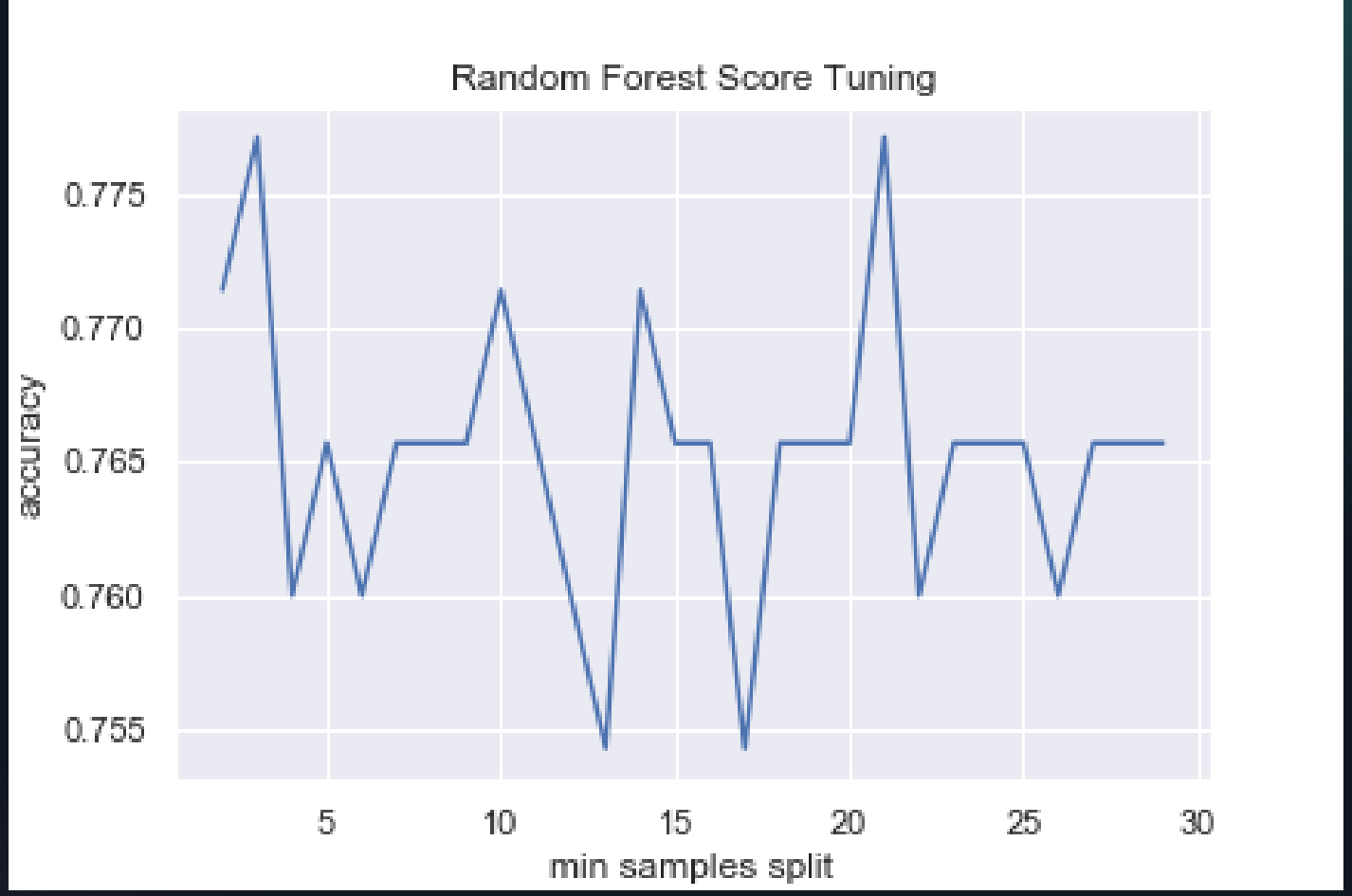
Distribution of some the features in the training dataset. The concentric closed curves show the densities of the data. The plots on the diagonals show the distribution of a feature. Variable 210 is interesting because it shows two clusters but upon further inspection we see that the classes are interspersed between these two clusters. There are a few more features similar to variable 210 in the dataset.



Distributions of another sample of features, this figure and the previous one suggest that most of the variables have a normal distribution. Moreover this figure robustly confirms that the data are not separable by methods such as linear regression.



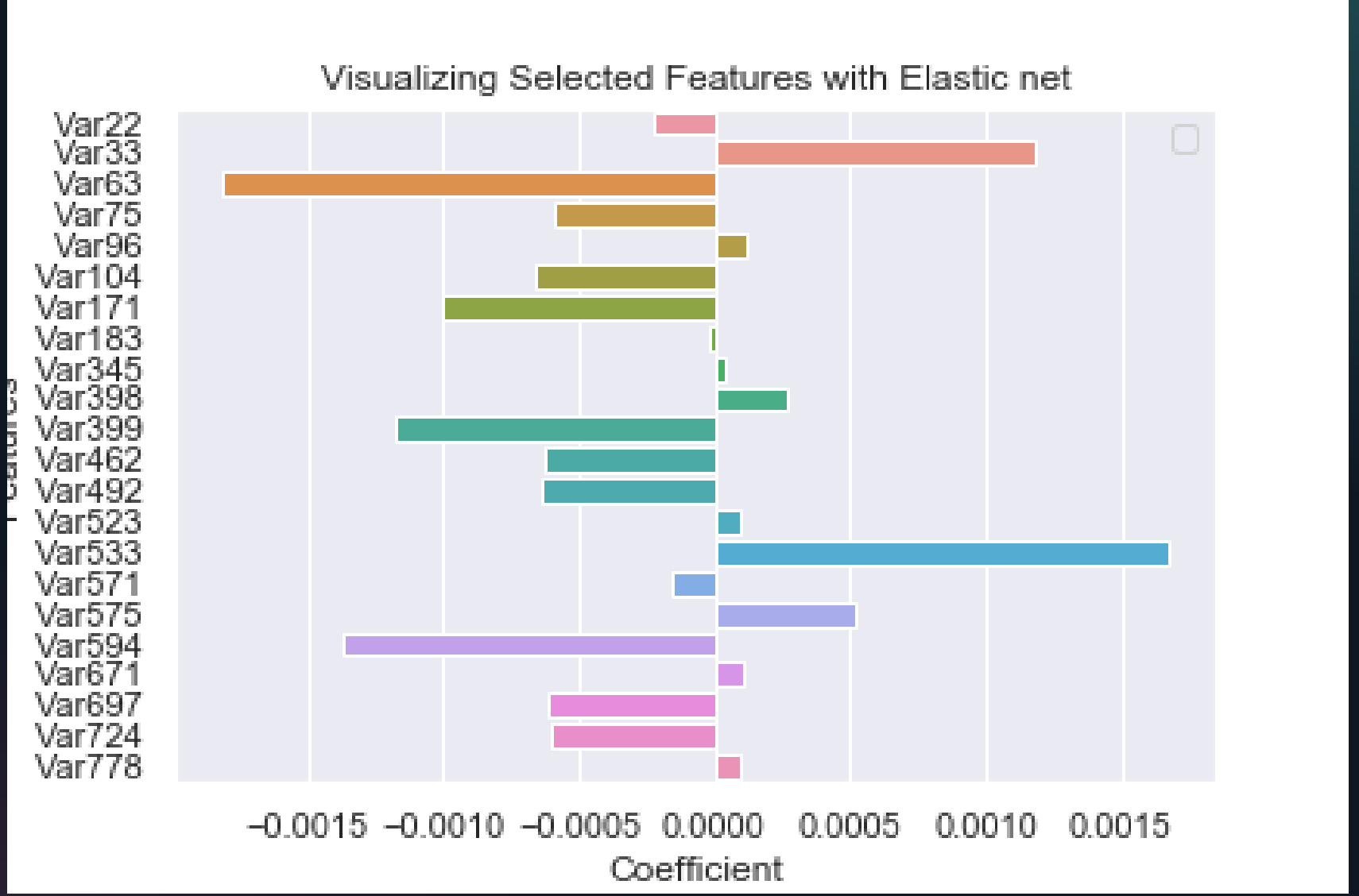
These plots help to choose a suitable range for tuning the hyperparameters; we don't want too many coefficients to be set to zero and also we don't want most of them to be non-zero



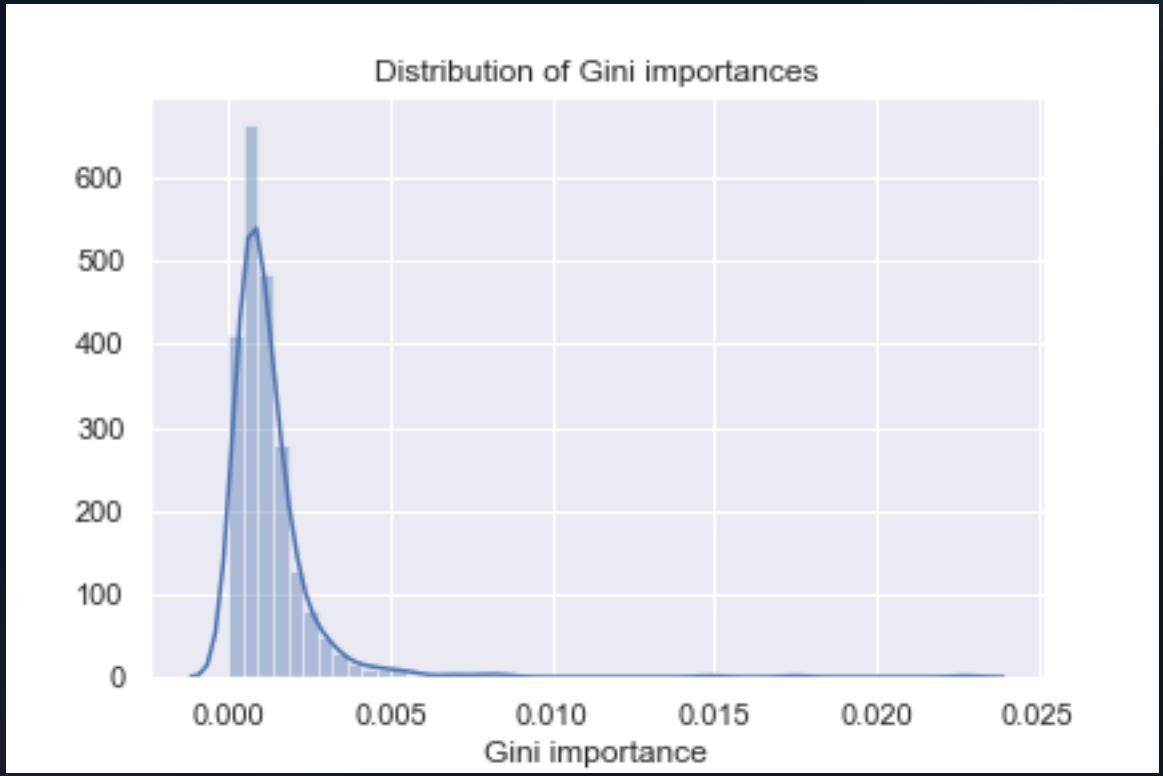
Cross-validation results for hyperparameter tuning in random forests with 200 decision trees. Changing other hyperparameters from default will only worsen the results. The highest accuracy achieved by random forests is 77.7 percent

Results

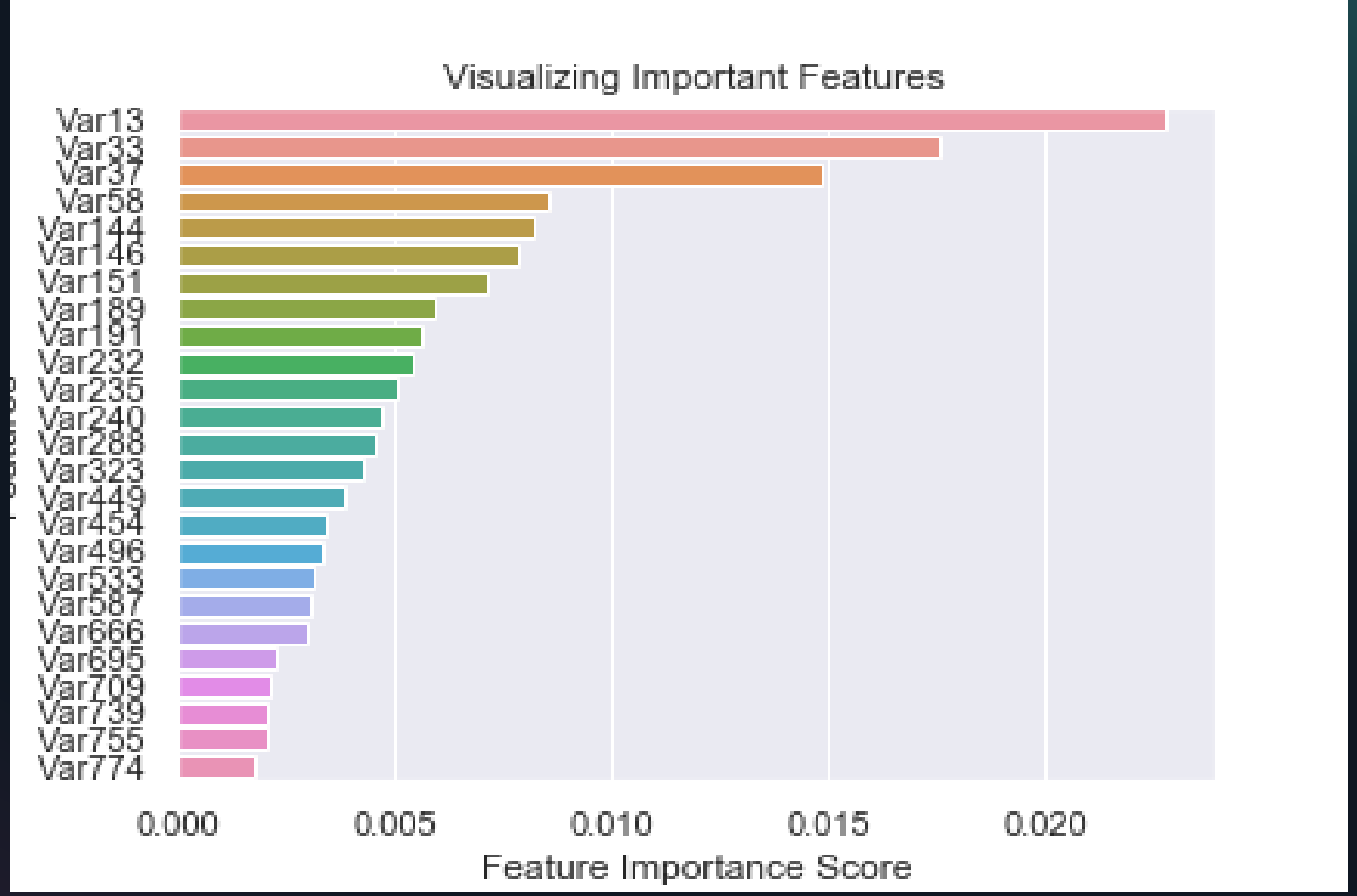
- After searching for hyperparameters the elastic net chooses l1 ratio = 0.47148664 and C (inverse of the regularization strength or alpha) = 0.006, which results in an accuracy of 76% on the validation set and chooses 93 predictors and sets others to zero.
- After bootstrapping these variables (0-indexed) were chosen as the most important ones with 100% confidence over the 200 resamples: [22, 33, 63, 75, 96, 104, 171, 183, 345, 398, 399, 462, 492, 523, 533, 571, 575, 594, 671, 697, 724, 778]
- Using the other ranking method described in the methods section these features were chosen (ranked based on the order of appearance) : [171, 594, 703, 31, 183, 33, 399, 63, 571, 670, 745, 492, 533, 462, 10, 483, 341, 697, 724, 75, 575, 20, 44, 96, 104, 345, 398 ...]. This shows a reasonable amount of overlap (81.81% of the features in the first few estimates of the 1st approach were found in the 2nd approach) between the methods for obtaining most important features in elastic net.


Visualizing Selected Features with Elastic net
Var22
Var33
Var63
Var75
Var96
Var104
Var171
Var183
Var345
Var398
Var399
Var462
Var492
Var523
Var533
Var571
Var575
Var594
Var671
Var697
Var724
Var778
−0.0015
−0.0010
−0.0005
0.0000
0.0005
0.0010
0.0015
Coefficient

- For the random forest classifier I rank the features based on their Gini impurities.



Based on this figure 0.003 or 0.004 would be a good threshold to draw out more relevant features; I choose 0.003 to get more comparable results in terms of numbers to elastic net.



Importance of features based on their Gini importance. The results are the average of 20 resamples (bootstrapping)

Discussion

- The two methods have similar performances on this dataset, with random forests being slightly more accurate (1.7 %) which is not much as random forests are state of the art. This is due to the classes being highly intermingled and as such it is hard to attain a good fit without overfitting. Moreover it is not easy to divide the dataset even by majority of vote because of the ambiguity of the decision boundaries as there is a lot of overlap and it would eventually lead to overfitting. There are some outliers in the data, however removing those who are above 3 standard deviations away from the mean does not significantly change the accuracy, albeit it improved the accuracy of elastic net by 1 percent.

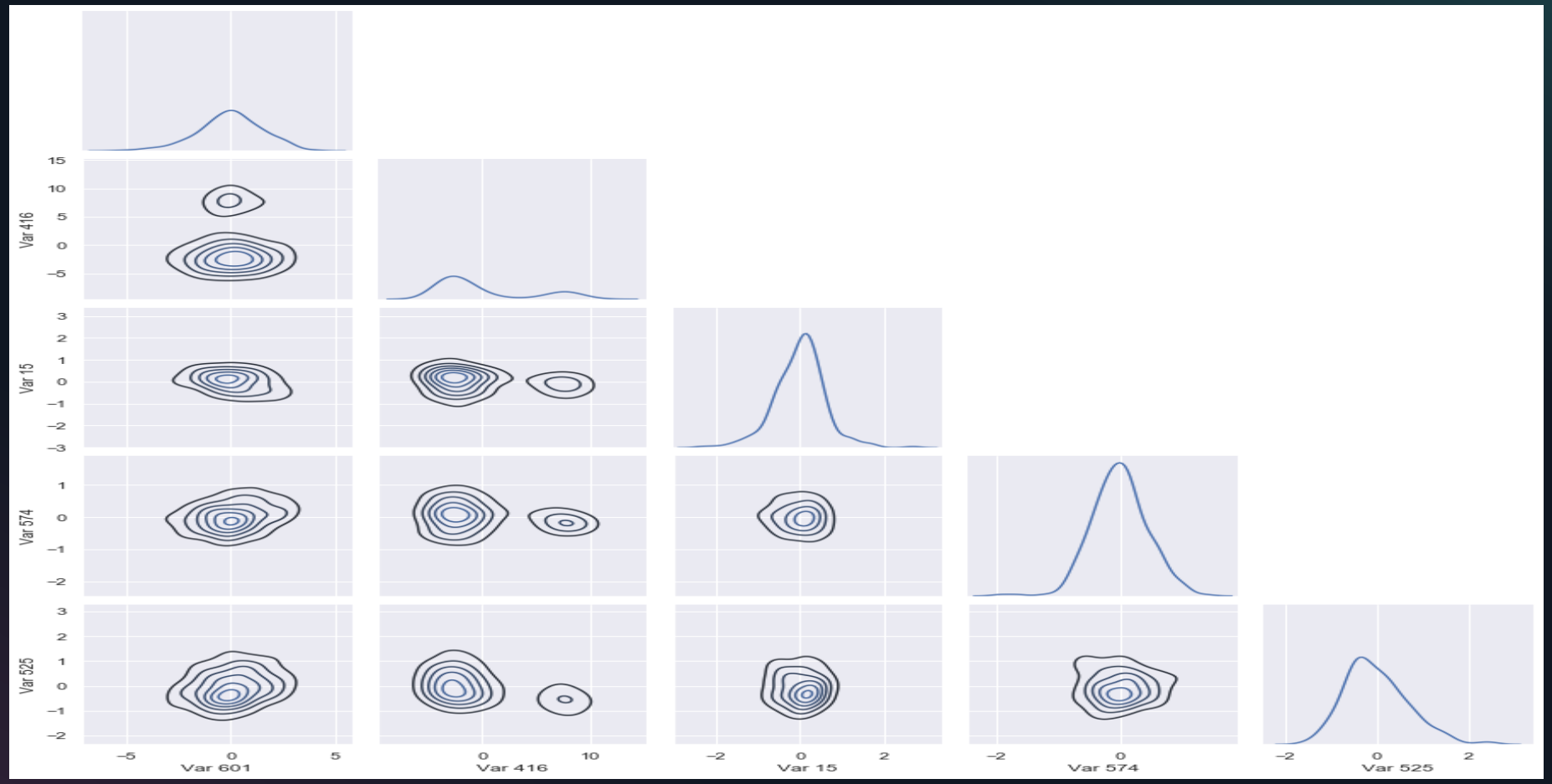
- Feature selection for elastic net was done in a robust manner as two methods were compared and a good overlap was shown among them. I personally prefer the second method (increasing the strength of the regularization) as it gives a direct measure of ranking and shows which coefficients are more important even in the presence of strong regularization. Unfortunately, I could not perform bootstrapping on the second method as it would have taken a lot of runtime.

- The random forest checks the importance of features differently and as such there were only two features [33, 533] shared among both algorithms (at least in the first few ranks). Gini importance or mean decrease in impurity calculates each feature importance as the sum over the number of splits across all tress that include the feature, proportionally to the number of samples it splits. The above described procedure is how sklearn computes it. As a result of this the method is expected to give different features compared to elastic net which is a regression algorithm and tries to find the best group of coefficients that fit the data.

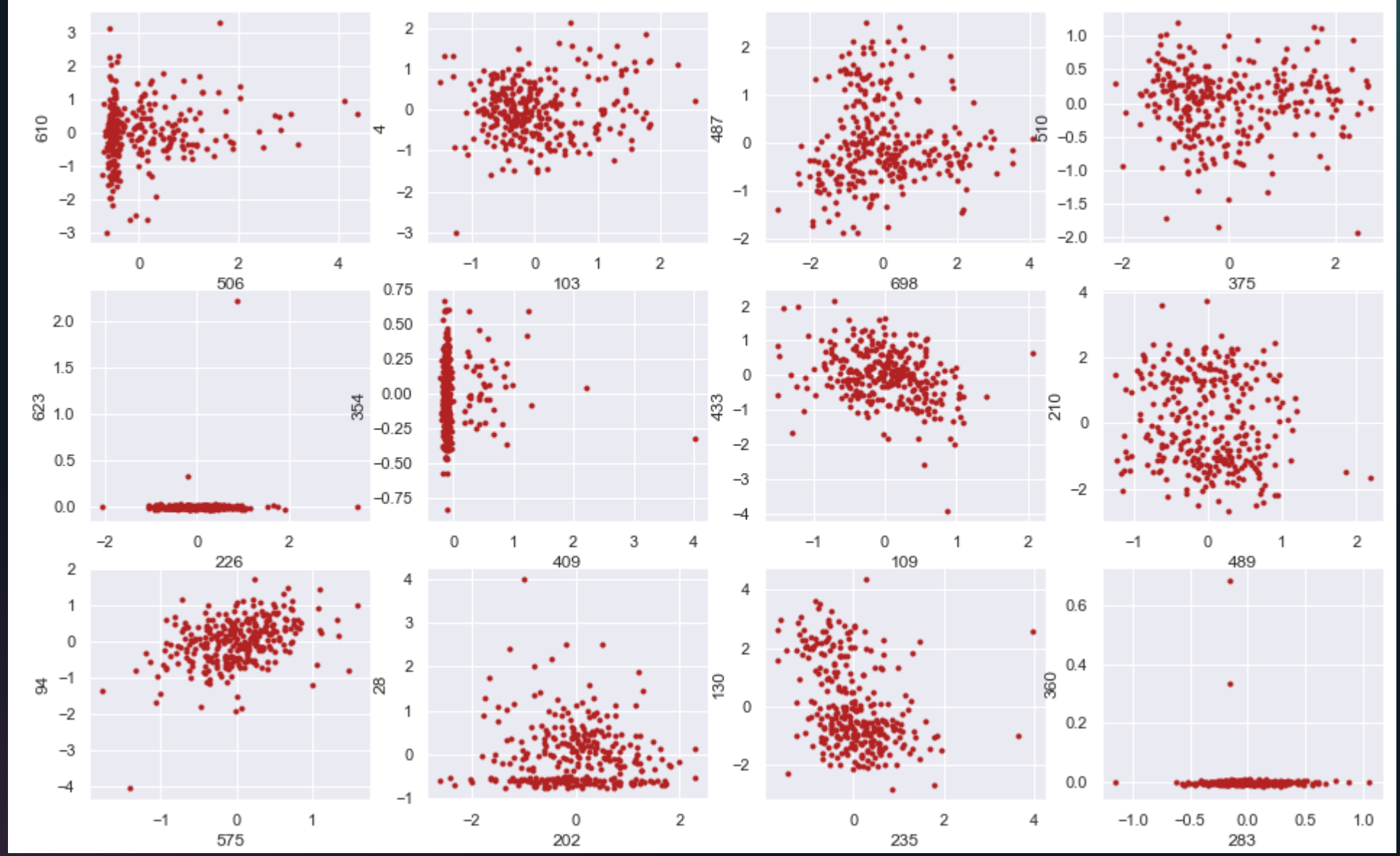
High Dimensional Clustering

Exercise 2

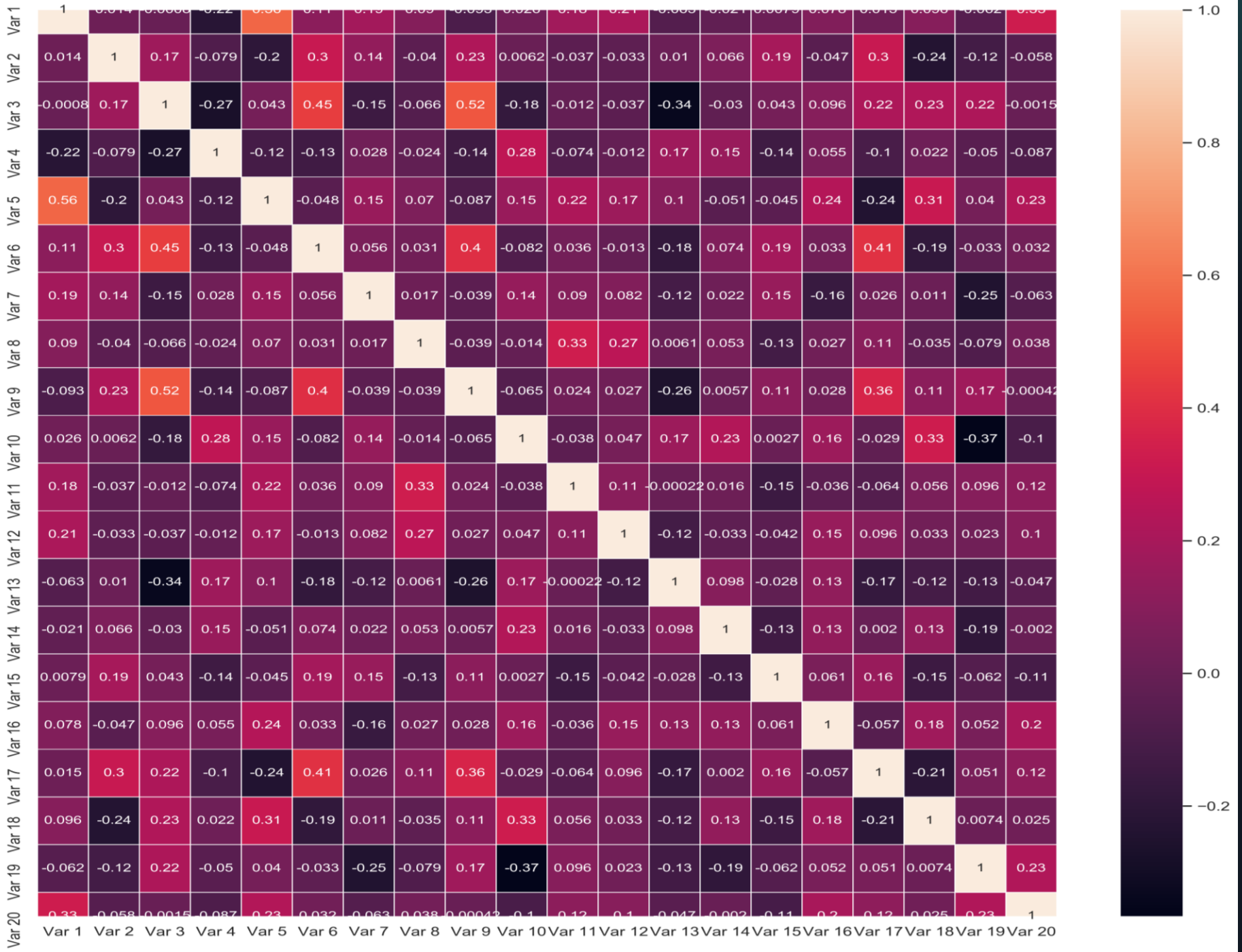
Methods and Exploratory Data Analysis



Some variables seem to be more important as they show some extent of clustering when mapped against other variables, but most don't. The data seem to have roughly a normal distribution

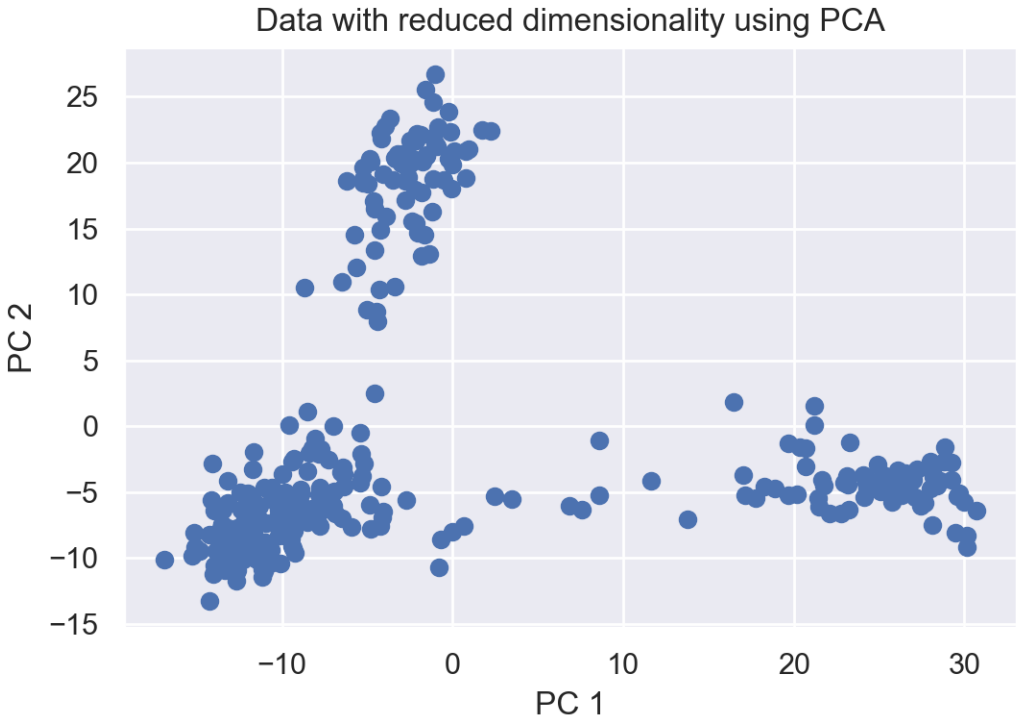


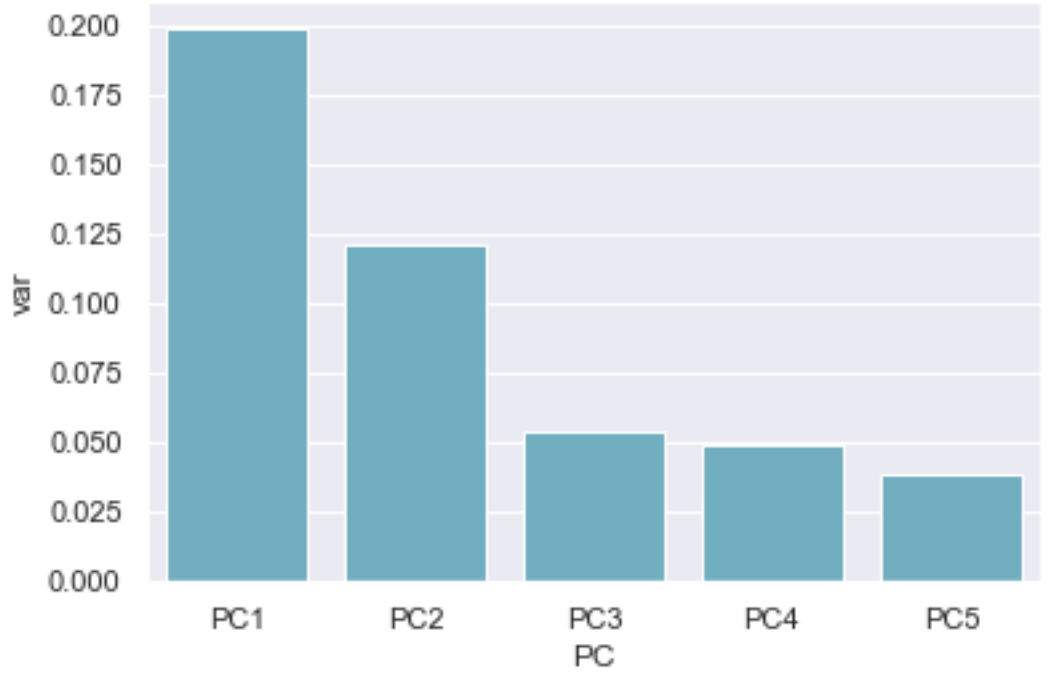
Some randomly selected features mapped against each other. Some variables show almost zero variance which means that they will have no effect in clustering and thus can be removed or projected into a lower-dimensional space without loss of generality.



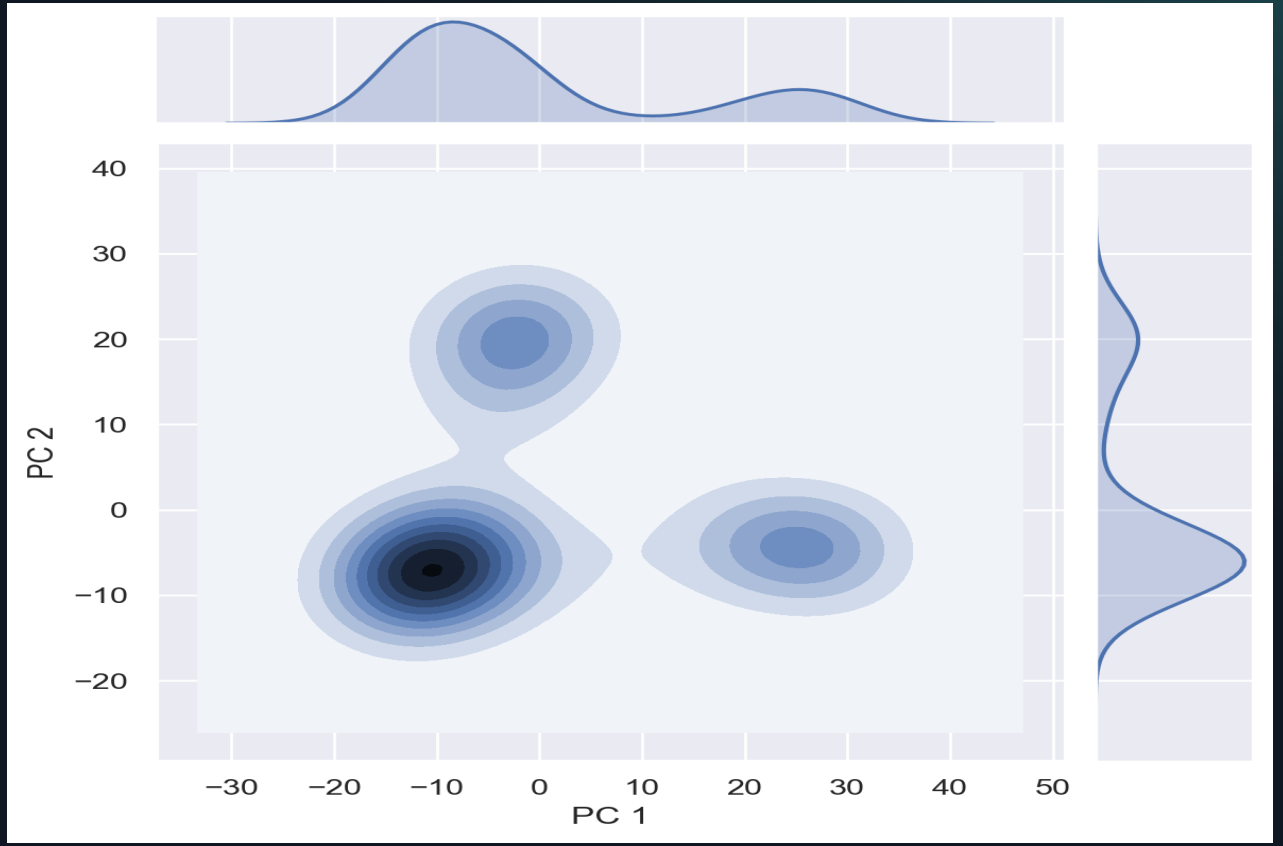
A heatmap showing between variable correlations. Although the correlations are not strong, there is some degree of correlation. This and the previous figure suggest that we can do PCA to be able to visualize and simplify the data set.

MM.DD.20XX

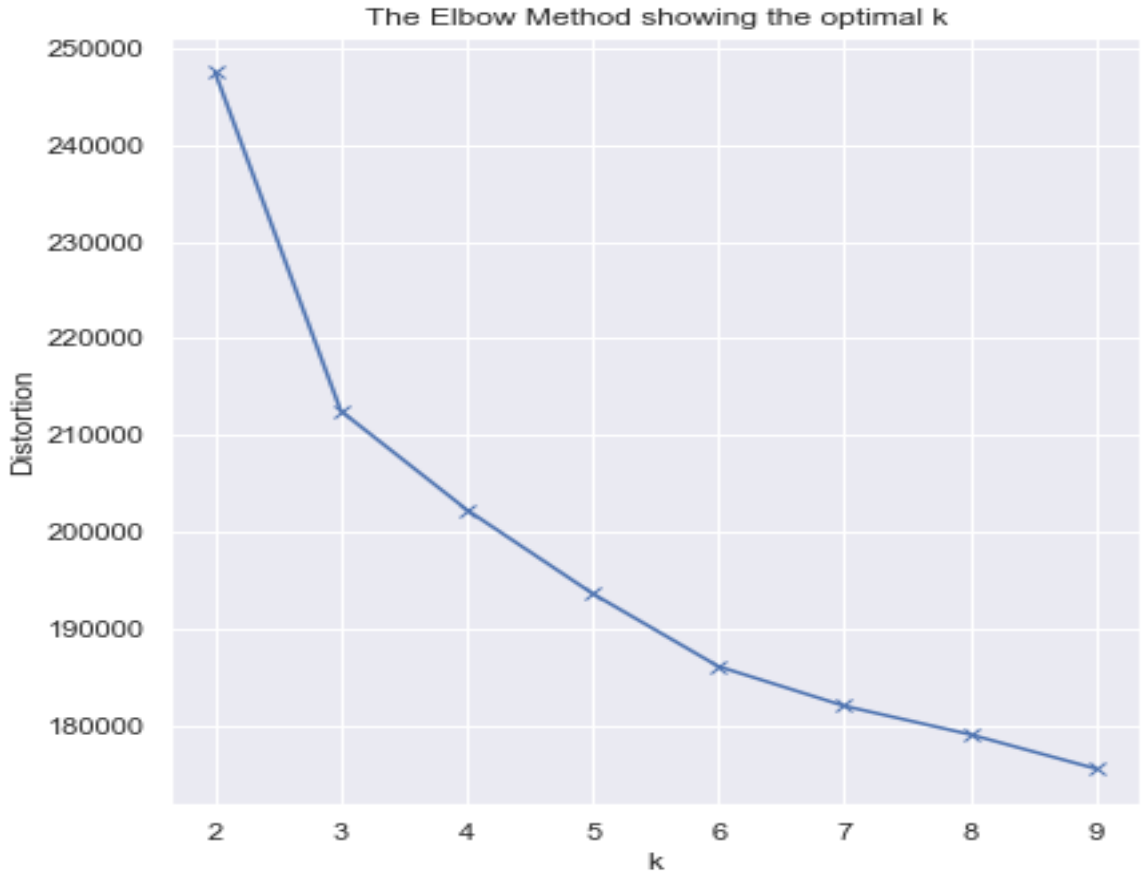


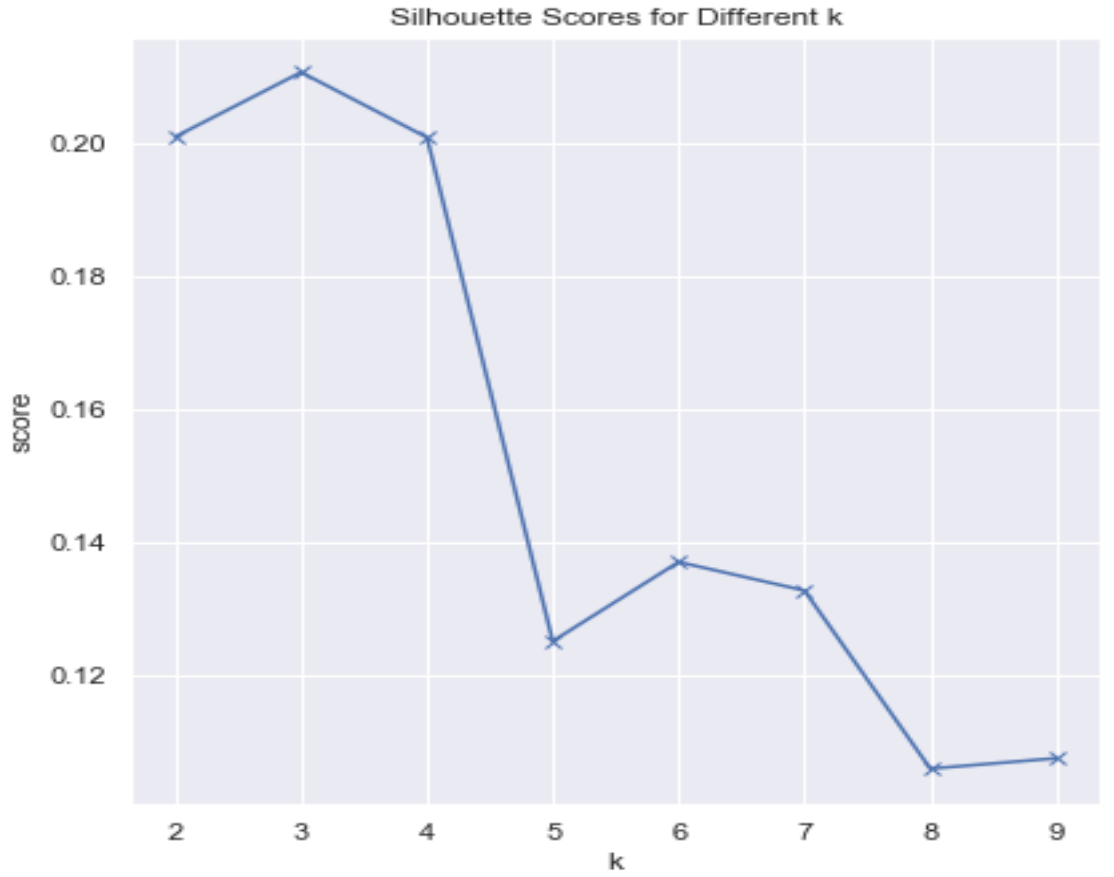


Distribution of the data after PCA which shows 3 clusters. The scree plot shows that the first two PCs capture a reasonable amount of variance in the data.

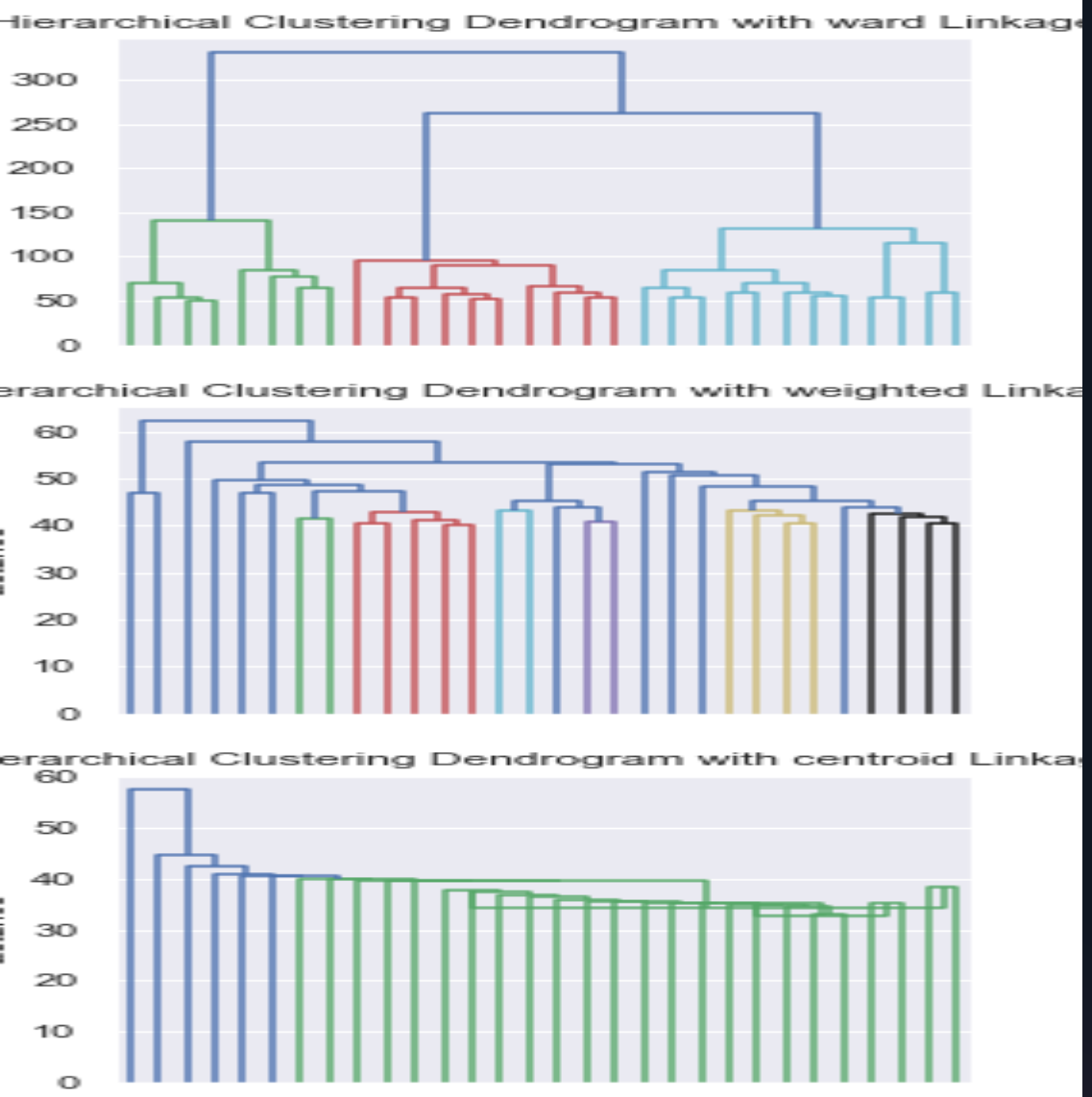


This plot shows the densities of the points in the PCA transformed data, which suggests existence of 3 clusters. Although the lower left cluster has a different density the other two have somewhat the same density which suggests that density based methods such as DBSCAN might not be useful here. Moreover, the clusters do not seem to be noisy with extreme outliers so there is no need to choose a method that accounts for high variance in the data such as K-medoids.





I choose the K means algorithm to accomplish the clustering in this task. The PCA results suggest that the clusters are convex and thus can be grouped as centroids. However, in order to deduce the best number of clusters I use the elbow method based on the distortions (SSE) and the silhouette score on the actual (unreduced) dataset. Both suggest 3 as being a good guess for the number of the clusters (along the visualization in PCA transformed data)

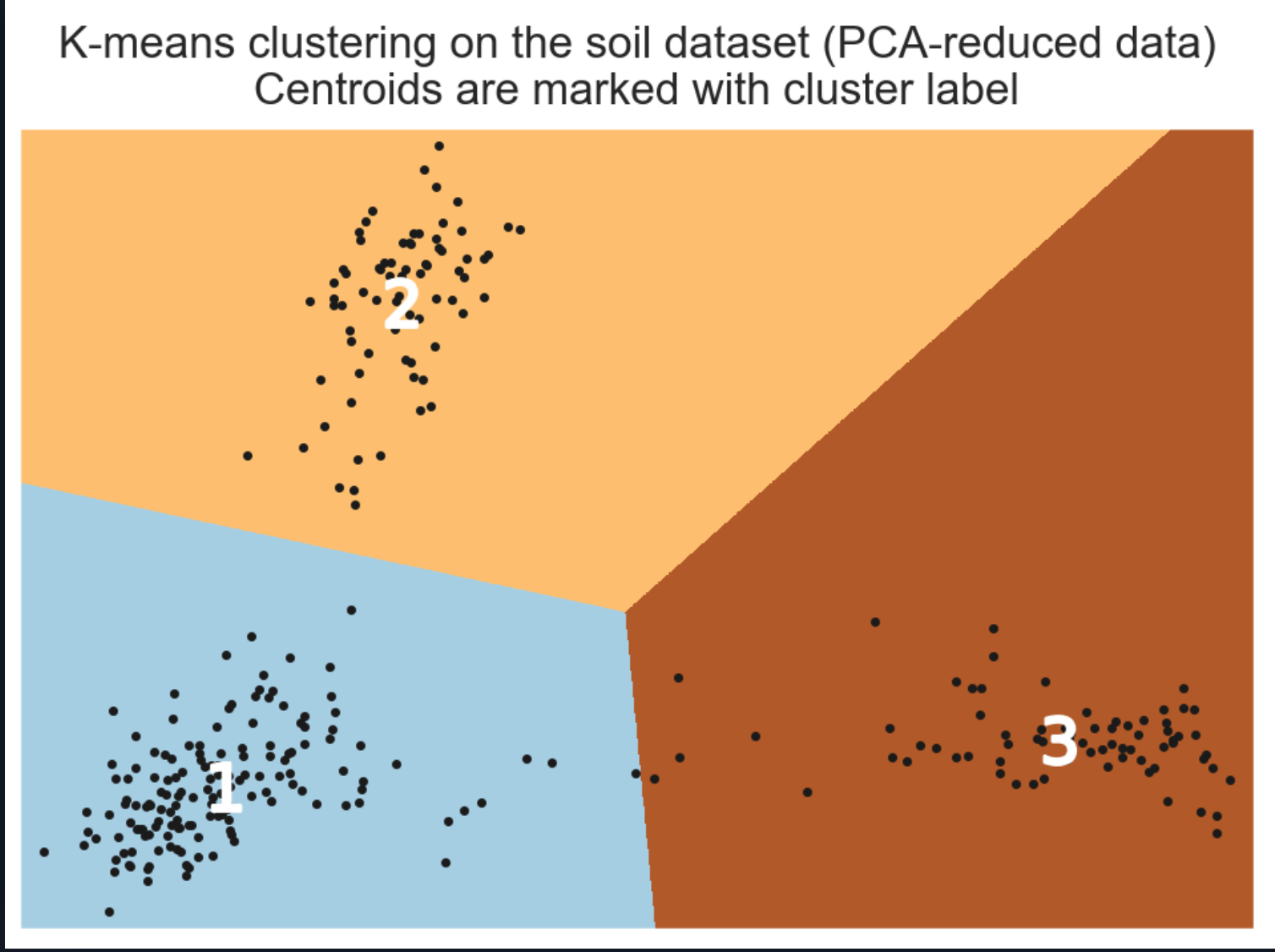


Hierarchical with ward linkage (which is suitable for spherical-like data) yields a better approximation than the other two linkages. Ward linkage is most similar to K means as they both minimize within-cluster sums of squares. However K means being iterative, is a better minimizer and also is more suitable when the clusters have similar shapes (variances) as suggested by the PCA data. After all said and done, I choose 3 clusters to perform the K means algorithm.

Results

In order to check how robust the clustering is on the PCA transformed data I compare the clusters found with k means on the original dataset and on the transformed dataset. The comparison yields a very high similarity with at most 3 points (one point wrong in each cluster) and at least 2 points ‘misclustered’ on the PCA data compared to the original data over several repeats. I choose the best achieved results: clusters 1, 2 and 3 share 99.36 , 100 and 97.30 % of their points respectively, on original and transformed data.

This figure illustrates the boundaries found by the algorithm for each cluster, showing 3 distinct clusters.



Feature selection for each cluster

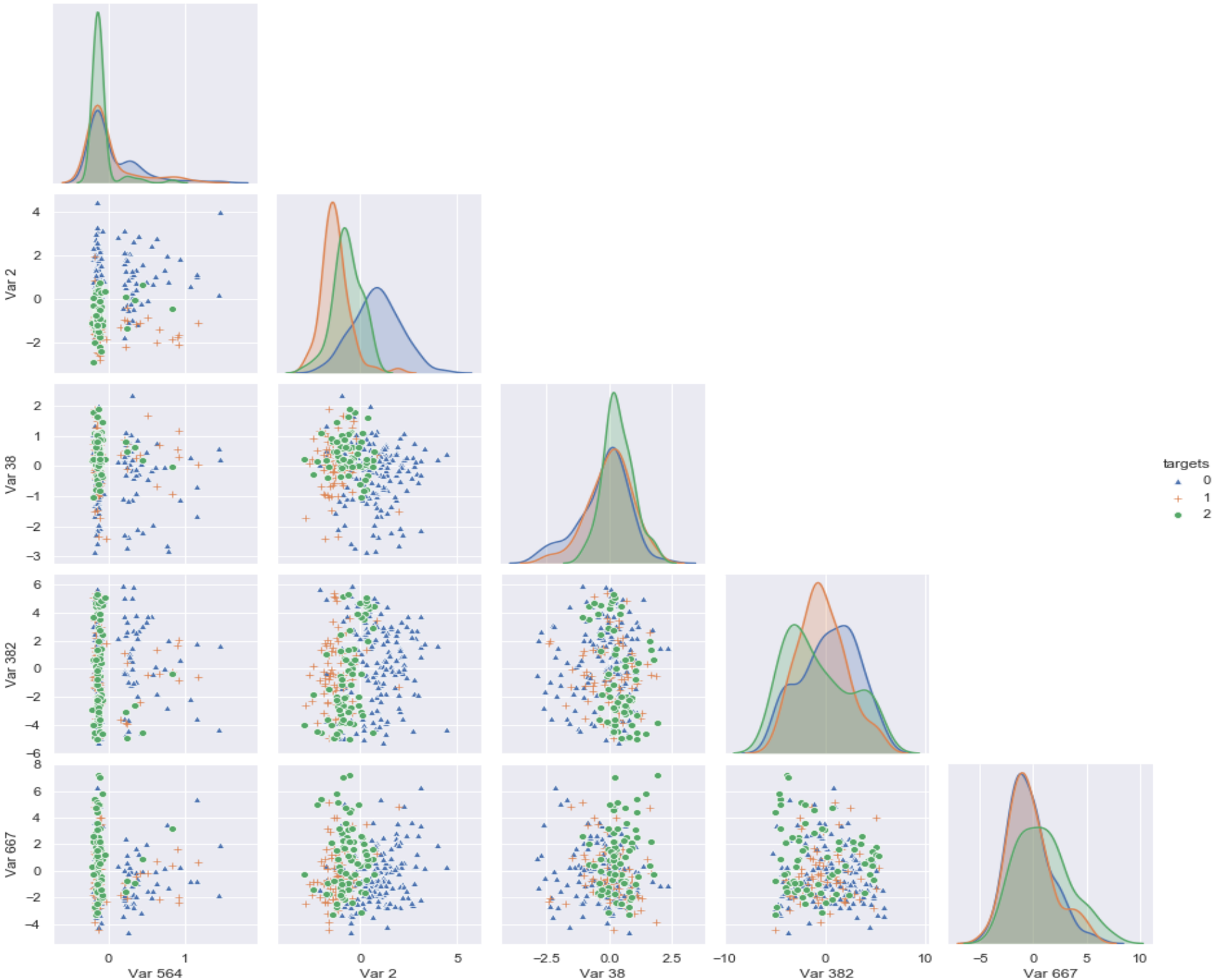
In order to find important features within each cluster, I adopt the following approach: after distinguishing the clusters, I use them as classes to perform a classification task. I use feature ranking with recursive feature elimination (RFE) and cross-validated selection of the best number of features. I do this on 2 clusters at each time (clusters 1 and 2, clusters 1 and 3, and clusters 2 and 3). So the task becomes a binary classification which can be handled by logistic regression. Then I intersect the most important features found for classifying clusters 1 and 3, 1 and 2 to find the suitable features for distinguishing cluster 1; clusters 1 and 2, 2 and 3 for cluster 2; and clusters 1 and 3, 2 and 3 for cluster 3.
I use 10-fold cv on the recursive feature elimination and choose ridge regression as my regularizd classifier, which I in turn fine-tune using a hyperparameter search with 10-fold cv. The search results in a 100% accuracy on most of the validation folds with the optimal regularization factor.

The selected features have ranking 1 according to RFE. The results after intersections are as follows for each cluster (more than 5 features where found however I will only write 5 of them) (Note: since python has 0-based indexing these obtained features should be added by 1 to translate to actual variables):
Cluster 1 : [21, 135, 300, 500, 713]
Cluster 2: [151, 402, 544, 557, 676*]
Cluster 3: [22, 80, 320, 492, 586]

* Feature 676 was shared over all clusters, no more than 4 unique features was found for cluster 2.

- Moreover, features 231, 318, 530, 676 and 719 were shared among all clusters as important

These features (1-indexed) are the best predictors of the clusters and rightfully so. As we can see the means of the distribution of the clusters are not overlapping very much and have some distance from each other. It can be further confirmed that these features can discriminate the clusters to a good extent by looking at the scatter plots. Of these, features 319 and 677 and 720 and 677 seem to be even more robust than others in separating the clusters. I test how accurate is the clustering using these 5 features compared to clustering on the whole feature space. It turns out that the clusters in these two setups share 97.42 , 93.51 and 98.63 percent of their points for clusters 1, 2 and 3 respectively.



This figure uses the same setup as that of the previous page, showing the distributions and scatter plots of some arbitrary features. The distributions show more overlap around their means and seem to have more similar variances.

Discussion

- The K means algorithm was able to establish distinct boundaries between the clusters and gave a good account of the data. As discussed in multiple occasions throughout the previous slides the K means algorithm is sufficient here. There might be some other algorithms who can perform as well as K means on the data but due to K means being more time efficient than most clustering algorithms it is a good choice for this data. Although, this facet is not stressed heavily in this task as the number of samples is not large.
- Besides visualization with PCA I chose two methods (elbow method and silhouette scores) to confirm that there are three clusters and they both confirm three to be the best estimate of the number of clusters in the data. Moreover, I use hierarchical clustering with 3 different linkage methods as this type of clustering can give an independent measure to guess the number of clusters. Since the clusters seem to be spheroids the ward linkage yields the best clustering also with 3 clusters.
- By running the K mean algorithm on the PCA transformed data and the original data and compare the results of clustering I show that the transform does not lead to a loss of generality.
- I also tried the DBSCAN algorithm on the data but it resulted in very bad clustering, detecting most of the points as noise. The reason for this might be the somewhat equal densities of two of the clusters in the dataset which leads to shortcomings for DBSCAN when trying to detect them. Moreover DBSCAN is more suitable when the cluster shapes are complicated and non-convex, which is presumably not the case here.

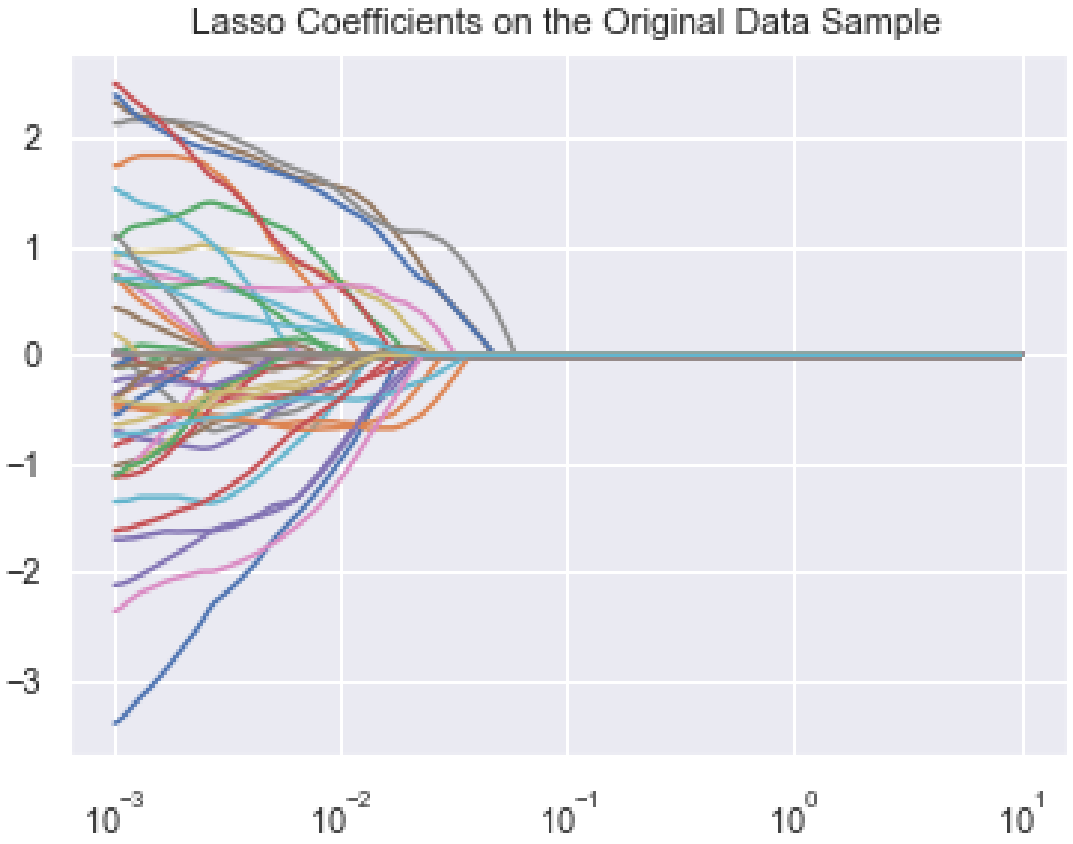
- Since the clusters are linearly separable (based on the PCA) data, logistic regression is more than enough to classify them in order to evaluate feature importance. I use RFE which measures the mean decrease in accuracy after eliminating each feature one by one. The best features are equally ranked by one and their intersections (as described in slide 27) give us the most important features. I use this approach based on this intuition, if some features can best distinguish cluster 1 from clusters 2 and 3, then they must be more indicative of cluster 1 and so on for the other two clusters.
- The figures I show for the most important features confirm that the selected features are best at clustering the dataset. As a further justification the comparison of the results of the clustering with the whole feature space and the top 5 important features chosen gives numerical evidence for this observation.

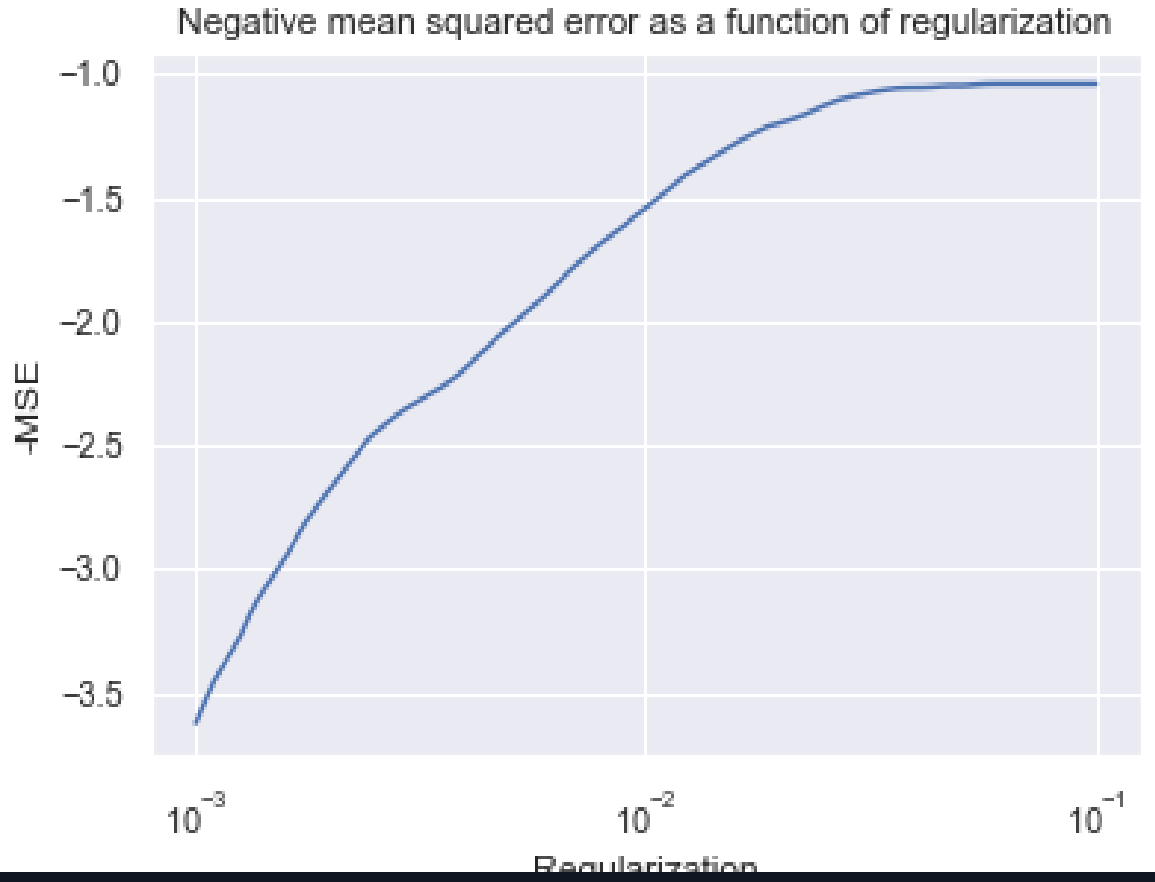
Improving Variable Selection in the Lasso

Exercise 3

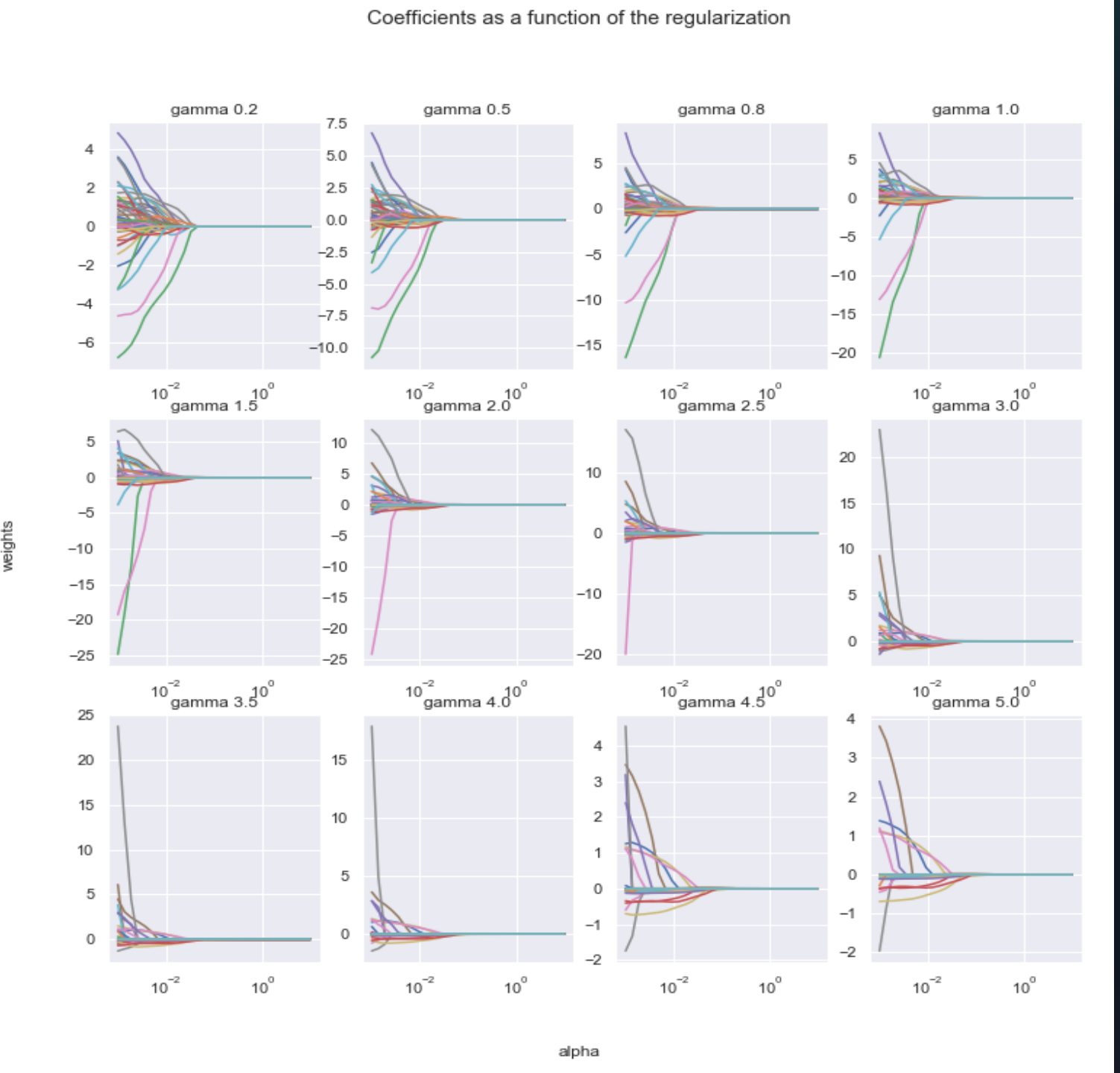
Methods

- Data simulation : For this exercise I sample my coefficients with sparsity of 0.1 from the normal distribution as per the instructions. As for my input I sample them from the beta distribution by setting the parameters $\alpha = 2$ and $\beta = 5$. I add Gaussian noise with variance of 1 to get the output data. I chose my number of coefficients to be 50 and choose the setup n>p.
- Since I am working with python I follow the instructions on how to get the adaptive lasso's objective coefficients from the lasso.
- I cross-validate the lasso with coordinate descent and 10-fold cross-validation to obtain the best hyperparameters and thus coefficients. I use negative mean squared error as my scoring metric. I then obtain the X' as described in the exercise and fit the lasso on X' with again 10 fold cv and obtain the results for different values of gamma. I choose the gamma that can best retrieve the true coefficients.
- I measure the performance of the lasso and adaptive lasso on how many of the original non-zero coefficients they can retrieve. To have more robust results I repeat the procedure on 15 different simulations and take the average of the percentage of found true coefficients over all of them.
- I chose n (number of samples) : [60, 80, 150, 300, 600, 1000]



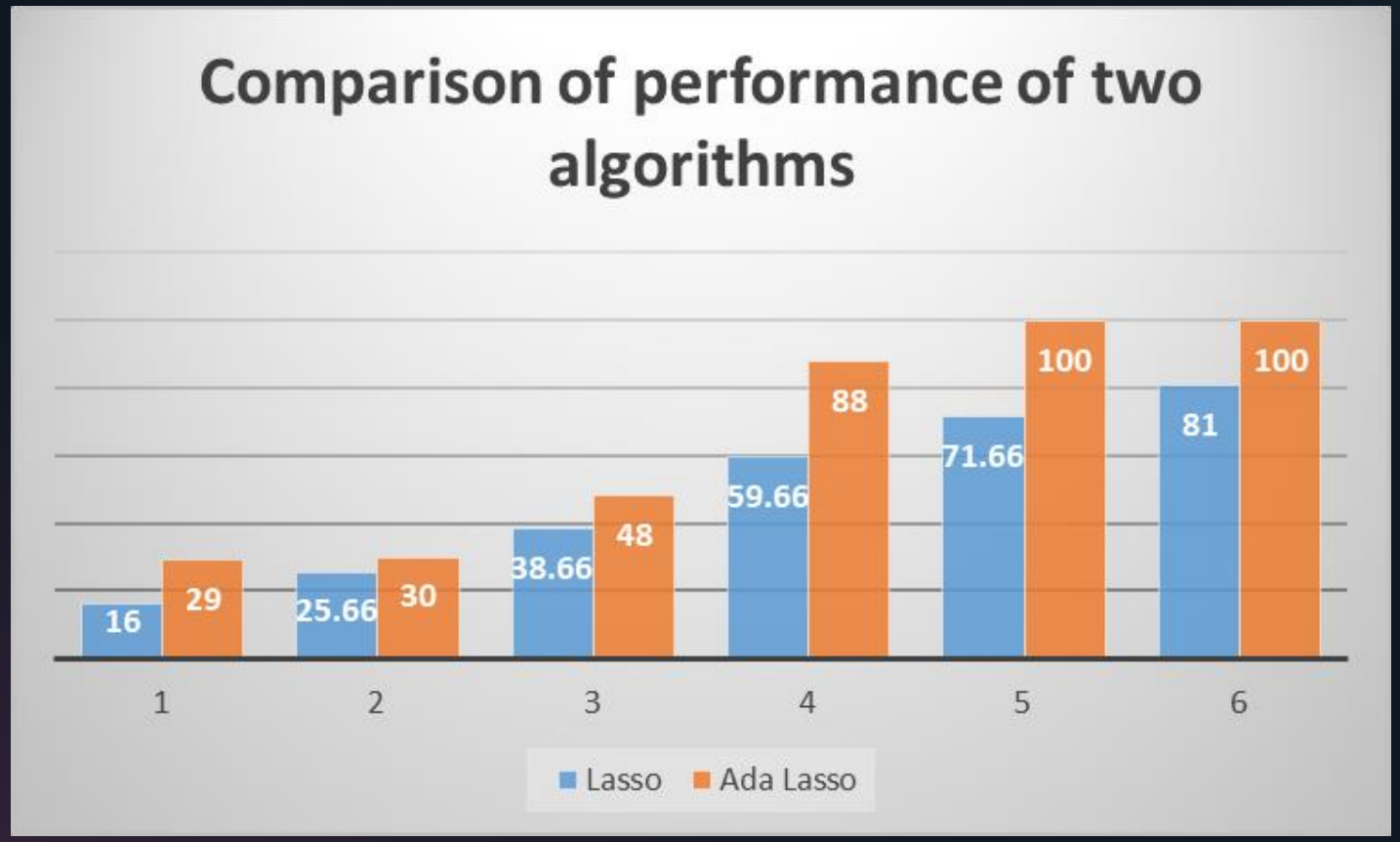


Effect of regularization on the lasso coefficients. We need to avoid high regularization as it tends to set all the coefficients to zero. Based on these graphs I chose the range of my L1 regularization to between 1e-3 and 1 .



This figure shows how the coefficients of adaptive lasso are affected by increasing the exponent gamma as can be seen particularly by the lower right figures corresponding to higher gammas. It seems that increasing gamma causes certain coefficients to become more important ; presumably coefficients with lower values in the linear regression, based on the description of how the weights are obtained. The gammas used in this figures are the ones I chose to cross validate over.

Results



The numbers beneath the bars correspond to the number of samples in each tested group. [60, 80, 150, 300, 600, 1000] respectively. For fewer samples the performance between lasso and adaptive lasso don't differ much but as the number of samples increases, especially in the 300-600 range, adaptive lasso significantly outperforms lasso in retrieving true coefficients. The percentages on the bars show the degree of retrieval averaged over all simulations for each sample size

Discussion

- As expected adaptive lasso is able to perform a better job in retrieving the true coefficients. By adding weights to the regularization in lasso, adaptive lasso tries to counteract the issue of lasso estimates being biased.
- Lasso encounters some problems which adaptive lasso can account for. Small non zero coefficients cannot be detected consistently using the Lasso. Large nonzero coefficients will become small which results in high bias. Adaptive lasso can also deal with correlated data, however we did not address this issue in this exercise.
- Moreover adaptive lasso has oracle properties. Meaning that it is more consistent in variable and parameter selection. Moreover it is more leaned towards the ground truth of the data and can better estimate it. Thus it shows more consistency in retrieving the data. As a matter of fact the obtained performances showed less variance using adaptive lasso compared to lasso