Unsupervised machine learning methods

Partu Ana-Maria

407

1. Task

The task for this project is comparing 2 unsupervised machine learning methods on a specific dataset. The chosen dataset must be unique and it must contain at least 1000 examples and 100 features. A supervised method must also be used in order to be able to compare the results from the unsupervised methods so the chosen dataset must be labeled. The dataset I selected is Gisette and the 2 methods I chose to compare are KMeans and DBSCAN.

2. Dataset

The Gisette dataset is one of the datasets that were used in the NIPS 2003 feature selection challenge. It is a problem of recognition of the confusable digits 4 and 9. It contains a total of 13500 examples.

The features and the labels are located in separate files for each of the sets. The test labels for this dataset were withheld so for this project I only used the training and the validation sets because I needed to be able to check the accuracy of the methods. The digits have been size-normalized and centered in a 28x28 pixels image.

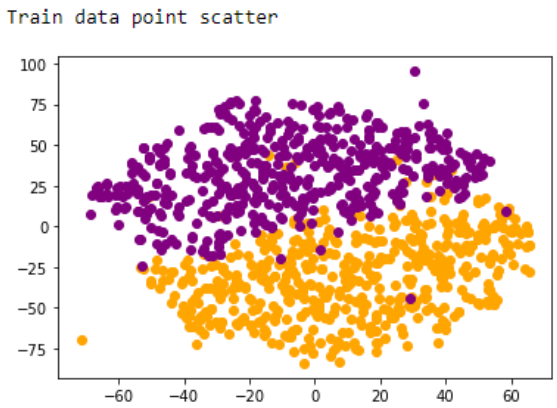
Each of the sets is split into the same number of positive and negative examples. The training set contains 3000 positive and 3000 negative examples, the validation set contains 500 positive and 500 negative examples and the testing set contains 3250 positive and 3250 negative examples. The dataset has 5000 features out of which 2500 are distractor features that have no predictive power. The order of the features and patters was randomized.

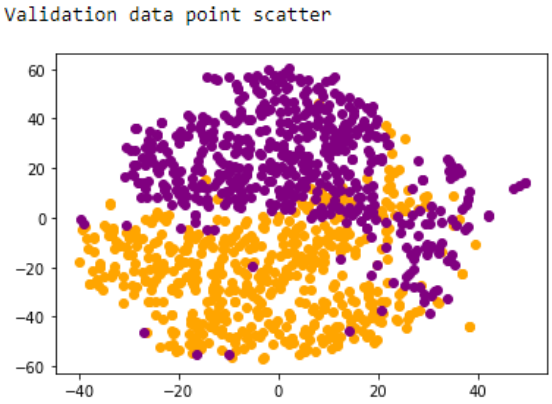
The data was modified for the purpose of the feature selection challenge: pixels were samples at random in the middle top part of the feature containing the information necessary to disambiguate the 2 digits and higher order features were created as product of these pixels thus plunging the problem into a very high dimensional feature space.

3. Processing and Feature Selection

I read and saved the data for each set into an array (X_train and X_valid) and the labels in a different array (y_train and y_valid). All the data was read in string format but since all the data was numerical, I changed it to int. The labels were 1 or -1 and I changed the -1 into a 0 for ease of use when comparing them to the cluster names.

Visual representation of the original dataset after using TSNE to show the data in a bidimensional space:





I tested both unsupervised methods and the supervised method on both the original data and the normalized data. For normalization I used normalize from sklearn.preprocessing.

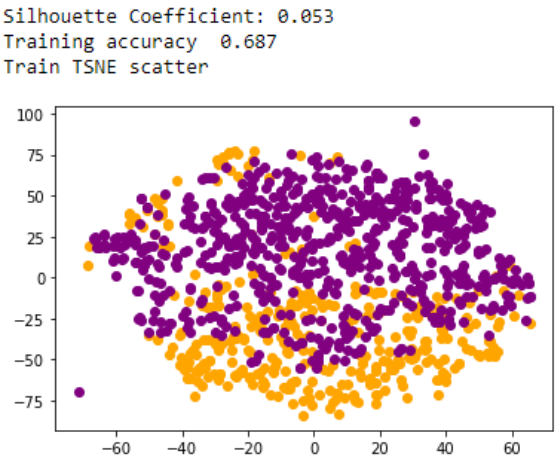
For feature selection we were only allowed to use unsupervised methods so I chose to use an autoencoder. Autoencoders are a type of artificial neural network used for learning feature representation in an unsupervised manner. By using an autoencoder we are forcing the network to create a compressed version of the input data. I used an autoencoder with the 5000 features as input and compressed it to 100 features.

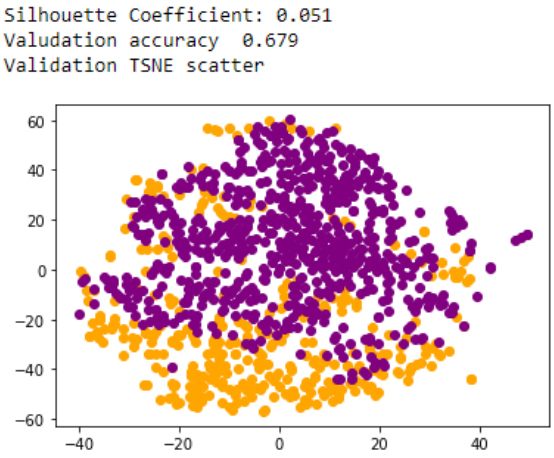
For the 2 unsupervised methods I tested the methods on their own as well as adding either normalization or an autoencoder or both. For comparison I also tested with the same changes to the supervised method. The random chance accuracy for this particular dataset is exactly 0.5 as the data is split into 2 classes with exactly the same number of examples for each one.

4. KMeans

For KMeans I set the number of cluster to 2 as my dataset had 2 classes. I tried combinations of KMeans on original data, normalized data and encoded data. I calculated the accuracy for both sets by comparing the kmean.labels_ with the actual labels of the sets.

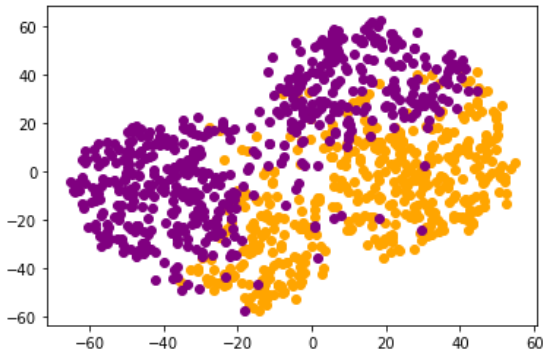
KMeans



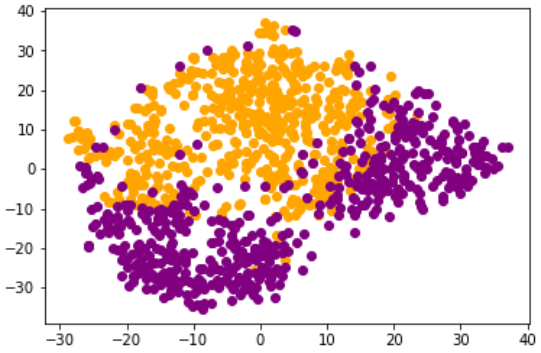


KMeans on normalized data

```
Silhouette Coefficient: 0.028
Training accuracy  0.5811666666666667
Train TSNE scatter
```

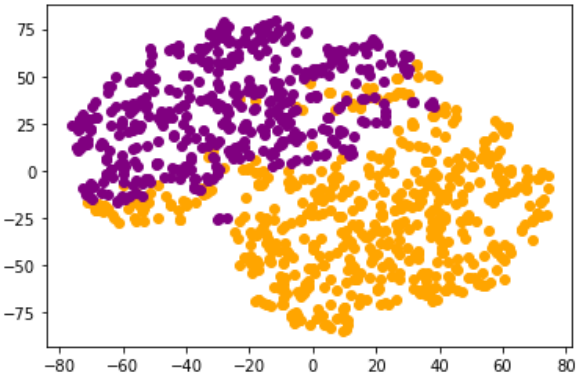


```
Silhouette Coefficient: 0.027
Valudation accuracy  0.564
Validation TSNE scatter
```

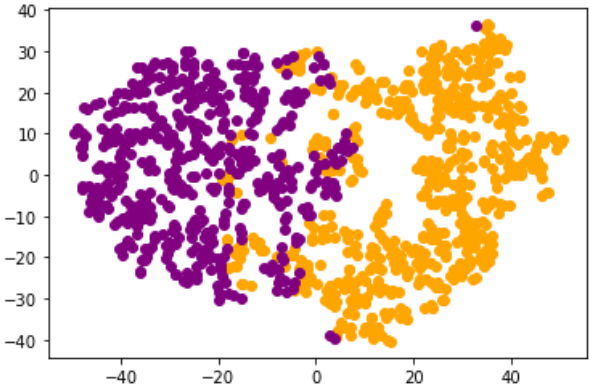


KMeans with autoencoder

```
Silhouette Coefficient: 0.203
Training accuracy  0.879
Train TSNE scatter
```

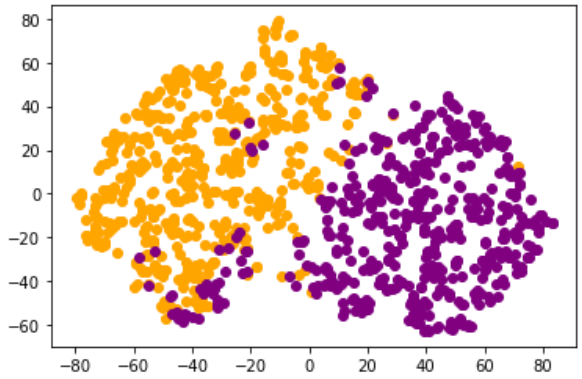


```
Silhouette Coefficient: 0.181
Valudation accuracy  0.899
Validation TSNE scatter
```

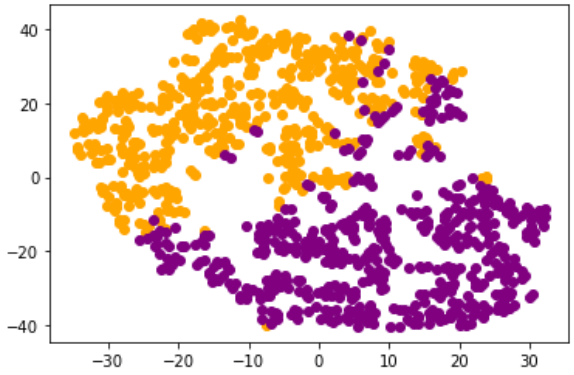


KMeans with autoencoder and normalized data

```
Silhouette Coefficient: 0.206
Training accuracy  0.9048333333333334
Train TSNE scatter
```



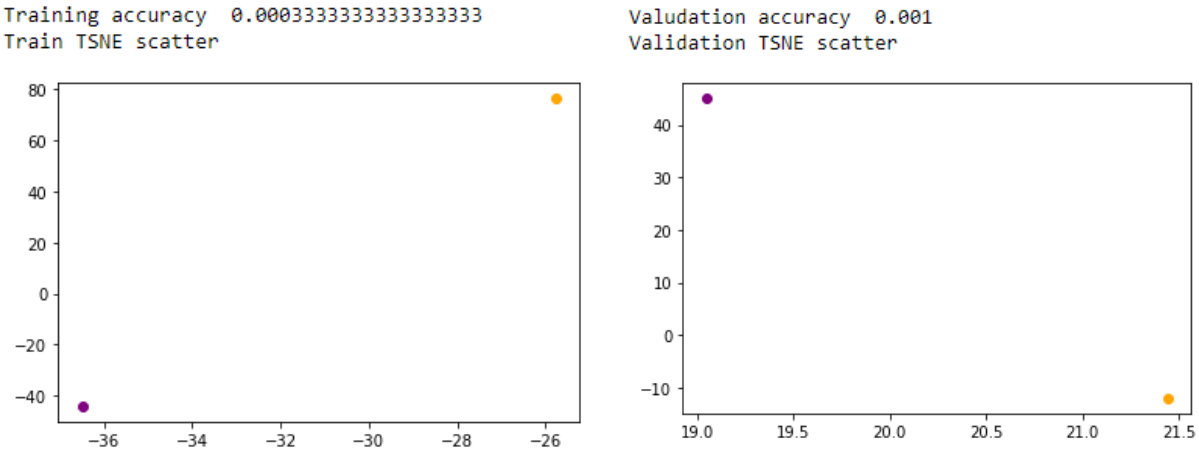
```
Silhouette Coefficient: 0.188
Valudation accuracy  0.904
Validation TSNE scatter
```



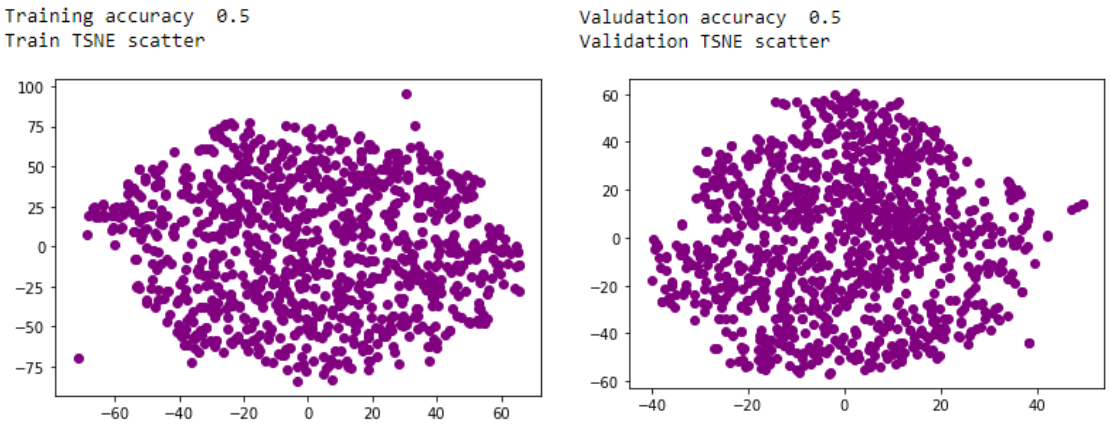
5. DBSCAN

DBSCAN works well when the cluster have similar densities. This seemed to not be the case for my dataset as DBSCAN on its own did not work at all. I decided to try and turn the -1 DBSCAN returned for outliers into a 1 and count it as a cluster to test this case as well since I know there are no actual outliers in the dataset. It turned out that all the examples were clustered into a single cluster. I made a set of functions to calculate the best hyperparameters for the epsilon and min_samples for each of the test cases.

DBSCAN



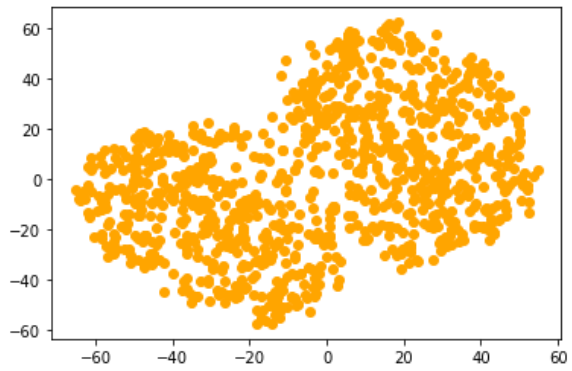
DBSCAN after turning the -1s into 1s



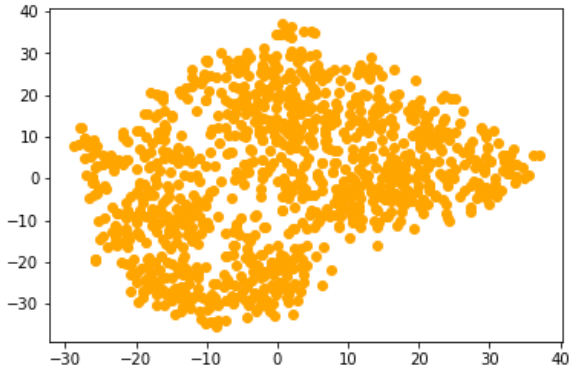
Next I tried normalizing the data before applying DBSCAN. In this case it clustered all the examples into a single cluster. I again tried to turn the outliers into a cluster. It finally showed 2 different clusters with an accuracy of 60% to 65% but the silhouette score was negative.

DBSCAN on normalized data

```
Training accuracy  0.5
Train TSNE scatter
```

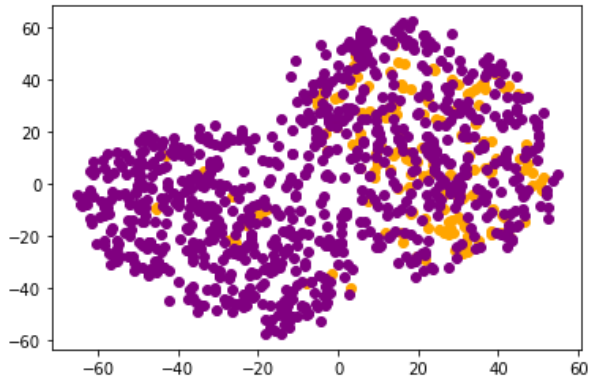


```
Valudation accuracy  0.5
Validation TSNE scatter
```

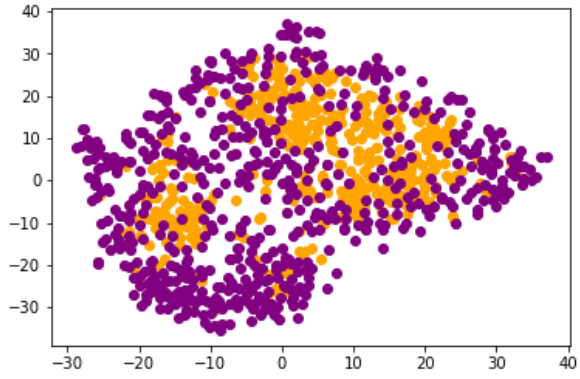


DBSCAN on normalized data after turning the -1s into 1s

```
Silhouette Coefficient: -0.008
Training accuracy  0.6143333333333333
Train TSNE scatter
```



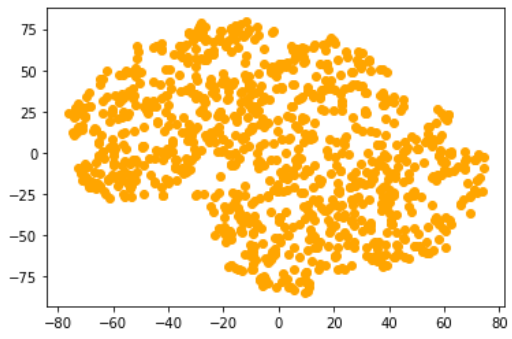
```
Silhouette Coefficient: 0.015
Valudation accuracy  0.652
Validation TSNE scatter
```



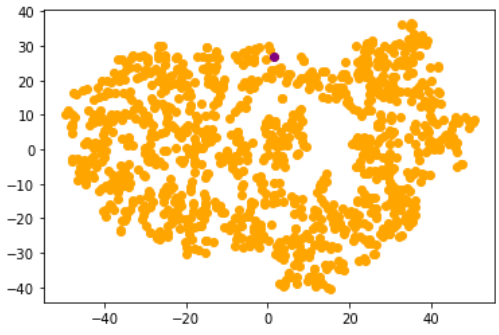
For the next test I applied autoencoder on the data before DBSCAN. It again clustered all the example into a single clustered. In the case of the outliers being classified as a cluster it had an accuracy of about 60% but again the silhouette score was negative.

DBSCAN with autoencoder

```
Training accuracy  0.5001666666666666
Train TSNE scatter
```

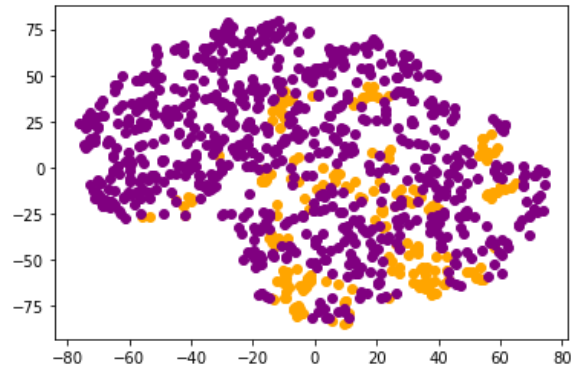


```
Valudation accuracy  0.501
Validation TSNE scatter
```

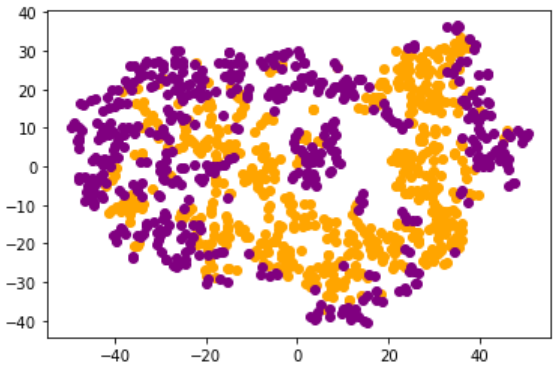


DBSCAN with autoencoder after turning the -1s into 1s

```
Silhouette Coefficient: -0.139
Training accuracy  0.6093333333333333
Train TSNE scatter
```



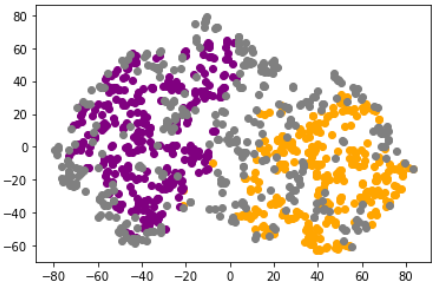
```
Silhouette Coefficient: 0.086
Valudation accuracy  0.59
Validation TSNE scatter
```



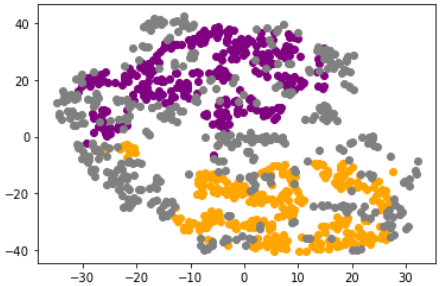
I decided to combine the 2 methods and try to use DBSCAN after applying both the autoencoder and the normalization. The accuracy was between 52% and 58% and the silhouette score was negative. With the outliers as a cluster I obtained the highest accuracy for DBSCAN, 73% to 76% but the silhouette score was negative.

DBSCAN with autoencoder on normalized data

```
Silhouette Coefficient: -0.214
Training accuracy  0.587
Train TSNE scatter
```

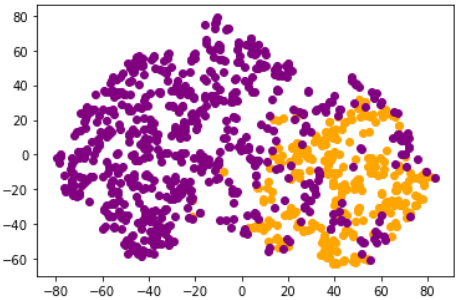


```
Valudation accuracy  0.521
Silhouette Coefficient: -0.109
Validation TSNE scatter
```

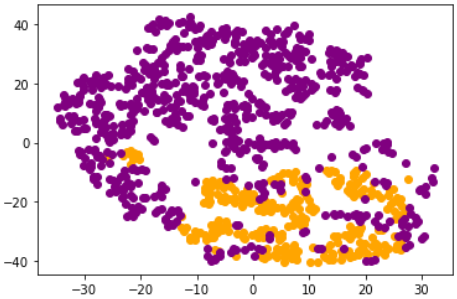


DBSCAN with autoencoder and normalized data after turning the -1s into 1s

```
Silhouette Coefficient: -0.256
Training accuracy  0.764
Train TSNE scatter
```



```
Silhouette Coefficient: -0.142
Valudation accuracy  0.736
Validation TSNE scatter
```

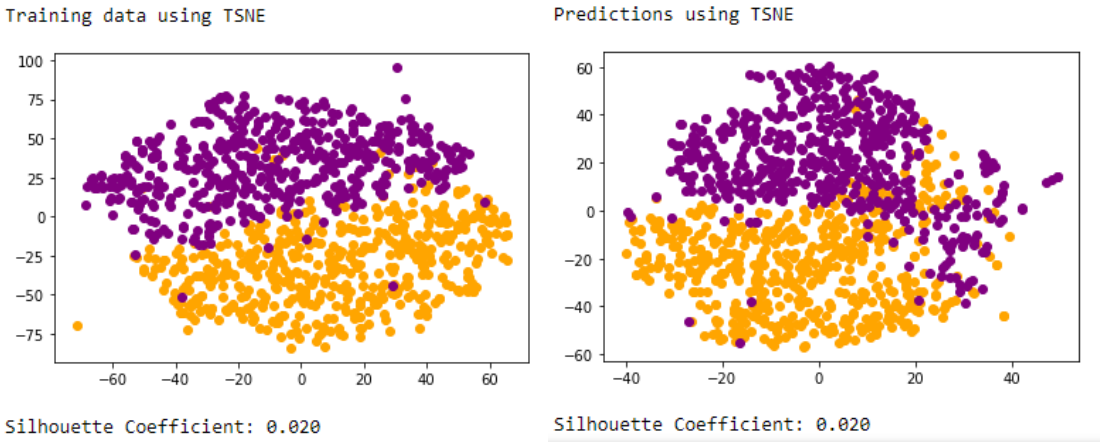


For some cases i couldn't calculate the silhouette score as it requires at least 2 clusters.

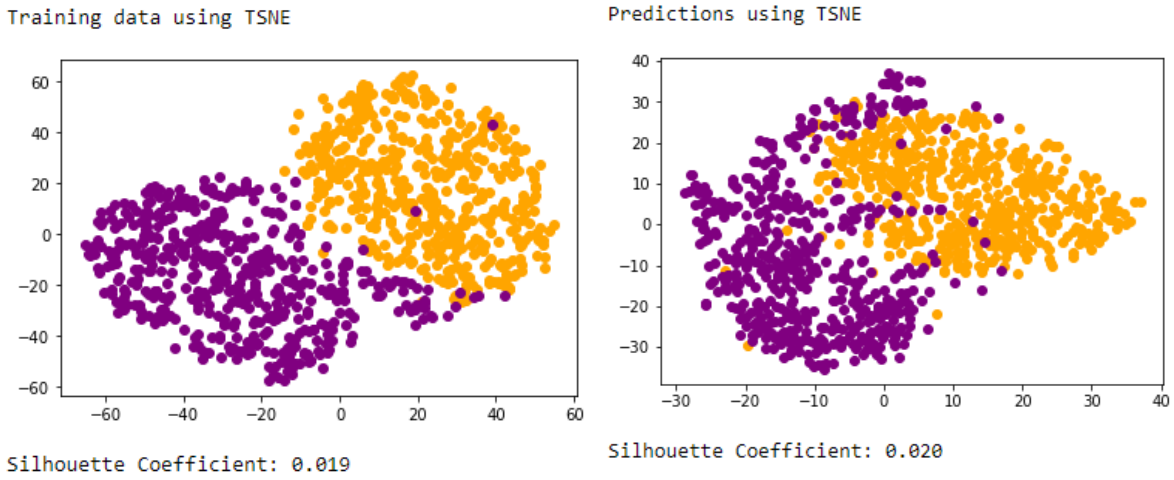
6. Supervised method

I used a simple MLP with a dense layer and varying number of neurons based on how many inputs I had for each test. I used ‘relu’ as the activation function and ‘softmax’ for the output layer. For optimizer I used Adam and for the loss function I used sparse categorical crossentropy. I tested the same changes I applied to the unsupervised methods on the supervised method as well.

MLP on original data

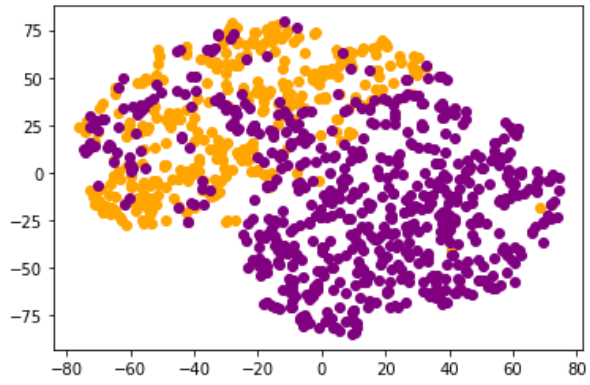


MLP on normalized data

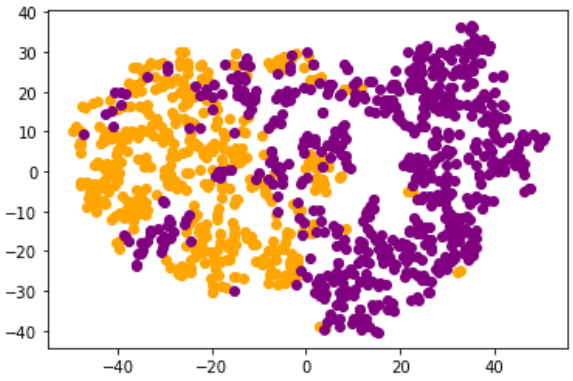


MLP with autoencoder

Training data using TSNE



Predictions using TSNE

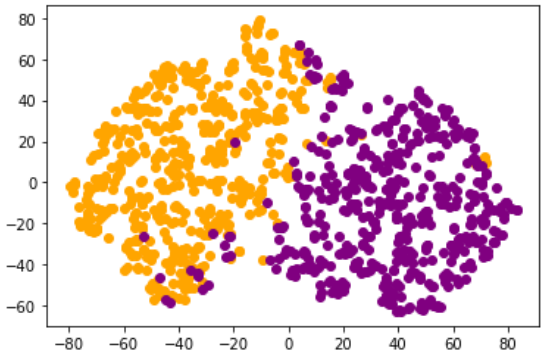


Silhouette Coefficient: 0.116

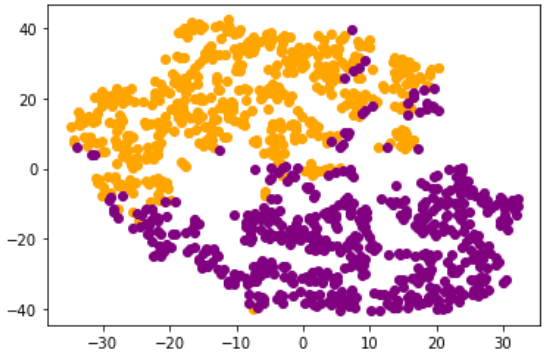
Silhouette Coefficient: 0.128

MLP with autoencoder and normalized data

Training data using TSNE



Predictions using TSNE



Silhouette Coefficient: 0.202

Silhouette Coefficient: 0.188

7. Conclusion

I obtained results over the base line for KMeans with all the methods tried, KMeans on encoded and normalized data yielding the best result with an accuracy of 90% and a positive silhouette score. For DBSCAN I only manage do obtain results over the baseline with 4 out of 8 methods but all had a negative silhouette score which is below the baseline. I got the best result with all the cases where I used MLP but the best unsupervised score came very close to the supervised method. As seen in the table below for this particular dataset KMeans worked very well on encoded and normalized data while DBSCAN failed to yield any results over both baselines (Accuracy and silhouette score).

	Train Accuracy	Train Silhouette Score	Validation Accuracy	Validation Silhouette Score
KMeans	0.687	0.053	0.679	0.051
KMeans + Norm	0.581	0.028	0.564	0.027
KMeans + Autoencoder	0.879	0.203	0.899	0.181
KMeans + Norm + Autoencoder	0.904	0.206	0.904	0.188
DBSCAN	0	N/A	0	N/A
DBSCAN + 1	0.5	N/A	0.5	N/A
DBSCAN + Norm	0.5	N/A	0.5	N/A
DBSCAN + Norm +1	0.614	-0.008	0.652	0.015
DBSCAN + Autoencoder	0.5	N/A	0.5	N/A
DBSCAN + Autoencoder + 1	0.609	-0.139	0.59	0.08
DBSCAN + Norm + Autoencoder	0.587	-0.214	0.521	-0.109
DBSCAN + Norm +Autoencoder +1	0.764	-0.256	0.736	-0.142
MLP	0.997	0.02	0.976	0.02
MLP + Norm	1	0.019	0.979	0.02
MLP + Autoencoder	0.946	0.116	0.858	0.128
MLP + Norm + Autoencoder	0.936	0.202	0.935	0.188
Random Chance	0.5	0	0.5	0