Auxiliary feature learning for small dataset regularization
CE888 Assignment 2 - *Project 1*

Cynthia Masetto, 1802774

***Abstract*— Dimensionality reduction is often used as a pre-processing step in classification. There are some dimensionality reduction techniques which transform the high dimensional data to new lower dimensional data. The autoencoders are now a popular method that are used to reduce the dimensions of the input features. This work performs dimensionality reduction autoencoders in three different databases. The features resulted in the autoencoders were then used to train a Suport Vector Machine classifiers that linearly separates the instances of a class from other classes instances. Feature importance and validation is performed for each dataset reaching accuracies of: 99%, 98.3% and 71.2%, respectively. The conclusion further discusses the results and possible improvements.**

I. Introduction

Nowadays, real-world data has high dimensionality. A lot of data preprocessing techniques have been suggested towards the optimization of the classification process. Dimensionality reduction is one of them and it's often used as a preprocessing step in classification. As the number of dimensions of a data increases, it becomes more difficult to process it. For that reason, classifiers with low-dimensional inputs perform a faster training and are able to learn a better classifier. Moreover, with small datasets overfitting could be avoided.

Dimension reduction methods are based on the assumption that dimension of data is artificially inflated and its intrinsic dimension is much lower. Principal Component Analysis (PCA) is one of the most popular linear dimension reduction, which transforms the high dimension data to lower dimension data. However, this is not the only method that can be used and one example are the autoencoders. The autoencoders are unsupervised Artificial Neural Networks trained so that they can reconstruct their input through an intermediate representation, typically of lower dimension.[12] This work is organised as follows: in the first part, the methods for dimensionality reduction will be explained a long with the autoencoders and simple linear classifiers, in the second part, data exploration and visualisation of 3 datasets obtained from Kaggle are perform, respectively, in the third part, a dimensionality reduction autoencoder that will learn a discriminative classifier is analysed, and finally, discussion and conclusion about the method are summarized.

II. Background

A. Dimensionality Reduction

Dimensionality reduction is the transformation of high-dimensional data into a meaningful representation of reduced dimensionality. Basically, the reduced representation should have a dimensionality that corresponds to the intrinsic dimensionality of the data. This means that the intrinsic dimensionality is the minimum number of parameters needed to account for the observed properties of the data. As a result, dimensionality reduction facilitates classification, visualisation, etc.[2]

Furthermore, dimensionality reduction can remove two types of "noise" from the input. The first noise is the independent random noise, which is uncorrelated with the input and the label, one example of this would be running Principal Components Analysis (PCA) or other unsupervised dimensionality reduction algorithm. The second noise are the unwanted degrees of freedom, which are possibly nonlinear, along which the input changes but the label does not. For that reason a more radical form of denoising requires the dimensionality reduction to be informed by the labels, that is why it is called supervised dimensionality reduction.[6]

On the other hand, dimensionality reduction has being performed using linear techniques such as: PCA, factor analysis and classical scaling[2]. However, the problem arises when complex nonlinear data appears in the picture and this kind of techniques cannot handle it.

For this reason, a number of nonlinear techniques for dimensionality reduction have been proposed. These techniques have the ability to deal with complex nonlinear data. Moreover, previous studies have shown that nonlinear techniques outperform their linear counterparts on complex artificial tasks. Some of those techniques are: Kernel PCA, Isomap, Maximum Variance Unfolding, Difussion Maps, Locally Linear Embedding, Laplacian Eigenmaps, Local Tangent Space Analysis, Sammon Mapping, Multilayer Autoencoders, Locally Linear Coordination and Manifold Charting.[2]

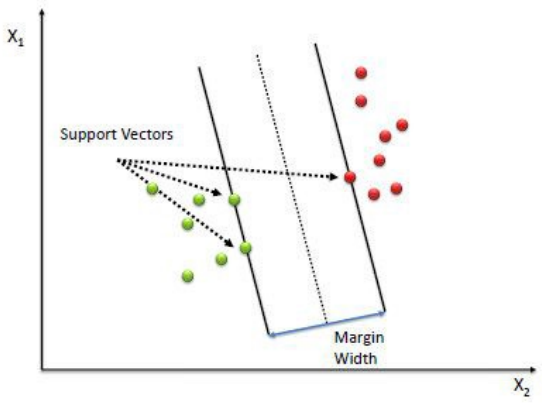
B. Discriminative Classifiers - Support Vector Machines

Discriminative classifiers are a popular approach to solving classification problems, one example of that are the Support vector machines, which are an approximate implementation of structural risk minimisation.[3] Support vector machines are binary classifiers, where the decision boundary is estimated to maximise the margin, the distance from the decision boundary to the closest points from each of the classes. Moreover, they have been found to yield good performance on a wide range of tasks and are suitable for use with data in high dimensional spaces.[3]

Support vector machines when used for classification, they separate a given set of binary labeled training data with a hyper-plane that is maximally distant from them. For cases in which no linear separation is possible, they

can work in combination with the technique of "kernels", that automatically realizes a non-linear mapping to a feature space. The hyper-plane found by the SVM in feature space corresponds to a non-linear decision boundary in the input space.

Fig. 1. Support Vector Machine



Source: Yadav, Ajay,*"Support Vector Machines"*, Towards Data Science, 2018.

As we can see in the *Figure 1*, the points that are closer to the hyper-plane are called "support vector points" an the distance of the vectors from the hyper-plane are called margins.[4]

C. Autoencoders

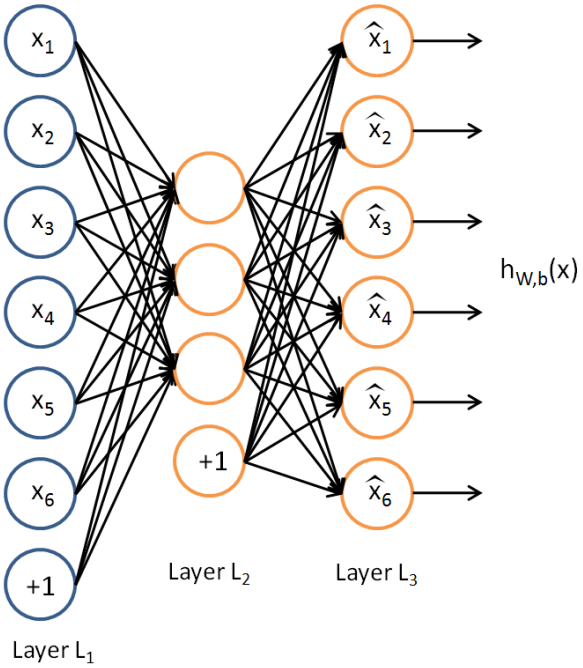
Autoencoder is an unsupervised tool for feature learning. Like other feature learning algorithms, the goal of autoencoder is to produce a good representation of the input data.[1] Autoencoders are widely used in different image classification tasks, speech emotion recognition and distribution estimation.

Specifically, multilayer autoencoders are feed-forward neural networks with an odd number of hidden layers, and share weights between the top and bottom layers.[2] Here, the network is trained to minimize the mean squared error between the input and the output of the network (ideally, the input and the output are equal).

According to Andrew Ng (2012), an autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs ($y^{(i)}$=$x^{(i)}$).

The autoencoder tries to learn an approximation to the identity function, so the output $\hat{x}$ is similar to x. In this case, the identity function is a particularly trivial function that tries to learn by placing constraints on the network such as: limiting the number of hidden units, that could give us some insights about the data. [7] Sometimes the autoencoder often ends up learning a low-dimensional representation very similar to PCAs. This happens when the number of hidden units is small, however when the number of hidden units is large more constraints on the network can be imposed and the autoencoder will still discover interesting structure in the

Fig. 2. Example of an autoencoder



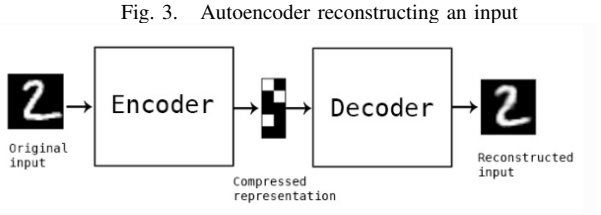
Source: Ng, Andrew, et al,*"Unsupervised Feature Learning and Deep Learning"*, Sandford University, 2012.

data. However, the main weakness of autoencoders is that their training may be tedious.

On the other hand, Chollet (2016) describes the autoencoder as a data compression algorithm where the compression and decompression functions are: data specific, which means that they compress data similar to what they have been trained on, lossy, which means that the decompressed outputs are degraded compared to the original inputs, and finally, they are learned automatically from examples rather than engineered by a human, which means that it is easy to train specialized instances of the algorithm that will perform well on a specific type of input.

The following image shows how an autoencoder reconstructs the input:

Fig. 3. Autoencoder reconstructing an input



Source: Chollet, Francois, *"Building Autoencoders in Keras"*, The Keras Blog, 2016.

An encoding, decoding and distance functions are needed to build an autoencoder. The distance function is the amount of information loss between the compressed representation of your data and the decompressed representation. Moreover, the encoder and decoder will be chosen to be parametric functions and to be differentiable with respect to the dis-

tance function, so the parameters of the encoding/decoding functions can be optimize to minimize the reconstruction loss using the Stochastic Gradient Descent.[5]

Furthermore, autoencoders can learn data projections that are more interesting than other techniques like PCA, where they can perform dimensionality reduction. Although autoencoders can solve the problem of unsupervised learning, they are also a self-supervised technique where they apply supervised learning and the targets are generated from the input data. On the other hand, in order to get self-supervised models to learn interesting features it is necessary to have a synthetic target and a loss function.

There are several types of autoencoders. First, **regularized autoencoders** that use regularization terms in their loss functions to achieve desired properties. Within these autoencoderes, there are the sparse autoencoders which adds a penalty on the sparsity of the hidden layer. Here, regularization forces the hidden layer to activate only some of the hidden units per data sample. This means that the output from a deactivated node to the next layer is zero. This restriction forces the network to condense and store only important features of the data. The other kind of regularized autoencoders are the denoising ones, which a random noise is deliberately added to the input and network is forced to reconstruct the unadulterated input. Here, the decoder function resists small changes in the input and a neural network that resists noise in input results. [8] Moreover, within regularized autoencoders, there also exist the contractive autoencoders, here instead of adding noise to input, contractive autoencoders add a penalty on the large value of derivative of the feature extraction function.

Second, the **variational autoencoders**, which are based on nonlinear latent variable models. These autoencoders consist of two neural networks, the first one for learning the latent variable distribution and the second one, for generating the observables from a random sample obtained from latent variable distribution. These autoencoders, minimize the loss and also the difference between the assumed distribution of latent variables and the distribution resulting from the encoder.

Third, **undercomplete autoencoder**, where the size of the hidden layer is smaller than the input layer in udercomplete autoencoders, here by reducing the hidden layer size we force the network to learn the important features of the dataset. Once the training is over, the decoder part is discarded and the encoder is used to transform a data sample to feature subspace. If the decoder transformation is linear and loss function is MSE the feature subspace is the same as that of PCA. [8]

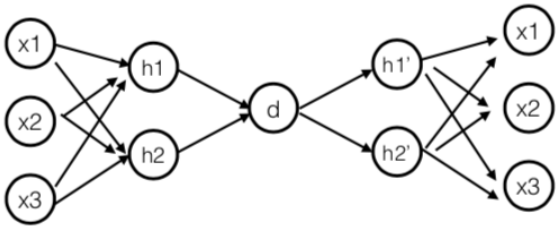
III. Methodology

Following the dimensionality reduction with autoencoders methodology, where the class information is used in order to define new discriminant targets. This approach will use autoencoders to learn the representation of the data and then train a simple linear classifier to classify the dataset into respective classes.

Autoencoders are used to reduce the dimension of the input features. Here, autoencoders are constructed by training the data in a way that the input and output are the same. All the datasets are use to perform this analysis and the features resulted from the autoencoders are used to train the classification algorithms, which in this case is the Support vector machine (SVM). This algorithm corresponds to a linear classifier that tries to identify which class the input belongs to (class = 1 or class=0) by making a decision based on the value of a linear combination of the features.

Autoencoders are trained in a unsupervised manner in order to learn the representations of the input data. This data are then deformed back to project the actual data. This means that an autoencoder is a regression task where the network is asked to predict its input. In order to build an autoencoder, a fullyconnected neural layer as encoder and as decoder are created. Where, as mentioned before, the encoder is the part of the network that compresses the input into a latent space representation. The encoder layer encodes the input data as a compressed representation in a reduced dimension that will later be coded to fed the decoder. On the other hand, the decoder will decode the disorted version of the original input and encode the input back to the original dimension. The decoded input will be a lossy reconstruction of the original input and it is reconstructed from the latent space representation.

Fig. 4. Autoencoder Architecture



Source: Kaggle, *"How autoencoders work? Intro and use cases"*, 2019.

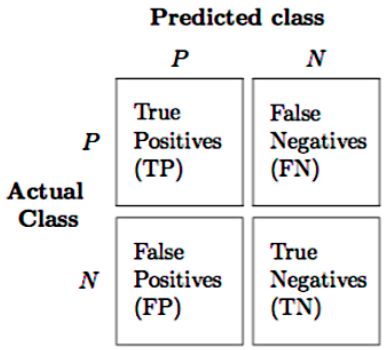
As shown in the image, a highly fine tuned autoencoder model should be able to reconstruct the same input which was passed in the first layer, the key is to use a rule for that, this rule is the equation that can be defined as the learning process.[14]

After training the autoencoder and using the selected features for the SVM classifier, the model performance will be tested trhough accuracy, sensitivity, specificity and ROC metrics. Additionally, the classification metrics can be tabulated by using the confusion matrix which uses the information of actual class labels and the information of predicted class labels. *Figure 5* shows a confusion matrix where the correctly classified labels are defined as True Positives and True Negatives, while the incorrectly classified ones are termed as False positives and False Negatives.

Where:

$$Sensitivity = \frac{tp}{tp + fn} \tag{1}$$

Fig. 5. Confusion Matrix



Source: Confusion Matrix (2018),Towards Data Science.

$$Specificity = \frac{tn}{fp + tn} \tag{2}$$

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{3}$$

As part of the work the following datasets are proposed to the analysis:

1) Credit Card Fraud Detection [9]
2) Cervical Cancer Risk Classification[10]
3) Black Friday[11]

The databases were obtained through Kaggle, which is an online community of data scientist and machine learners that allows users to find and publish data sets, explore and build models in a web-based data-science environment. All datasets and codes are saved in the following **GitHub Repository**: `https://github.com/CynthiaMasetto/Data-Science-Project`

A. *Describing the data*

1) Credit Card Fraud Detection: This dataset contains transactions made by credit cards in September 2013. This dataset presents transactions that occurred in two days, where 492 frauds were found out of 284,807 transactions.[9]

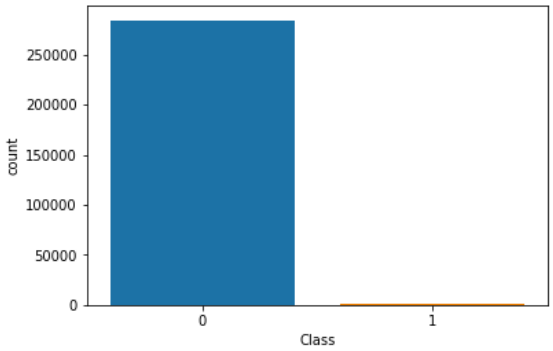
For confidentiality reasons, the dataset contains only numerical input variables which are the result of a PCA transformation. Features V1,V2,...,V28 are the principal components obtained with PCA, the only features of this dataset which have not been transformed with PCA are *Time* and *Amount*. Feature *Time*, contains the seconds elapsed between each transaction and the first transaction in the dataset, on the other hand, feature *Amount* is the transaction amount, finally the feature *Class* is the response variable and it takes value 1 in case of fraud and 0 otherwise.[9]

In *Figure 6*, the class can be seen:

As it was mentioned, the Class = 1 accounts for 0.172% of all transactions which means that the data set is unbalanced.

2) Cervical Cancer Risk Classification: The dataset is obtained from UCI repository, which is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms and contains a list of risk factors for cervical cancer. According to the World Health Organization (WHO), almost all cases of cervical cancer are caused by HPV. HPV is a very common virus that can be passed on through any type of sexual contact with a man or a woman. Cervical cancer occurs when the cells of the cervix grow abnormally and invade other tissues and organs of the body.[4] Worldwide, cervical cancer is the fourth most frequent cancer in women.

Fig. 6. Class "Fraud"

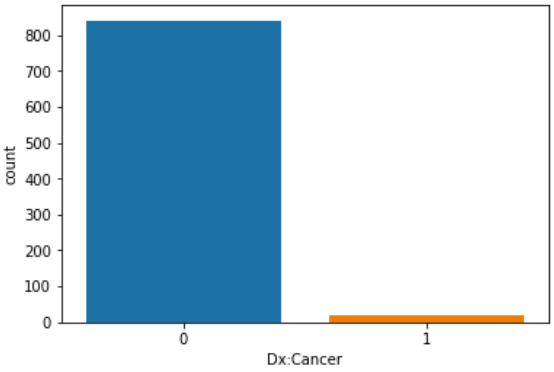


Source: Own analysis with data from kaggle.

The following variables are contained in the dataset: age, number of sexual partners, first sexual intercourse, number of pregnancies, smokes, smokes (years and packs), hormonal contraceptives, hormonal contraceptives(years), IUD (with years), STDs, diagnosis, hinselmann, shciller, citology and biopsy. The class of diagnosis is 1 if cancer is positive and 0 otherwise.[10]

As part of the analysis, the graphs for class = 1 (positive cancer) and the age distribution is showed:

Fig. 7. Class cancer

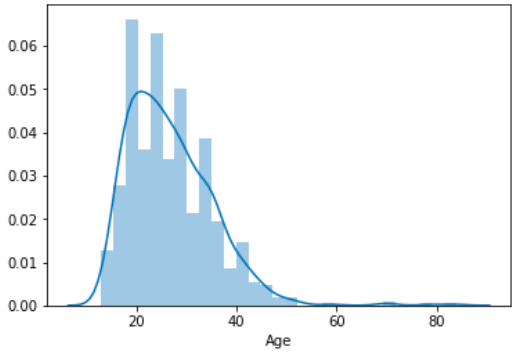


Source: Own analysis with data from kaggle.

As we can see, in *Figure 7*, the distribution is centered between 20-40 years.

3) Black Friday: The Black Friday is an informal name for the Friday following Thanksgiving Day in the United States. The day after Thanksgiving has been regarded as the beginning of America's Christmas shopping season since 1952. Many stores offer highly promoted sales on Black Friday, such are the volume of costumers in this date that during 2017 over $59.57 billion dollars were spent. The dataset is a

Fig. 8. Distribution of age

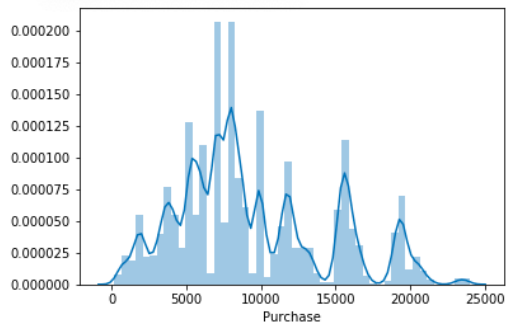


Source: Own analysis with data from kaggle.

sample of 555,000 observations about the transactions made in a retail store during Black Friday. The variables contained in the dataset are: user id, product id, gender, age, occupation, city, marital status, product category and purchase amount in dollars.[11]

The following graph shows the distribution of the purchases made:

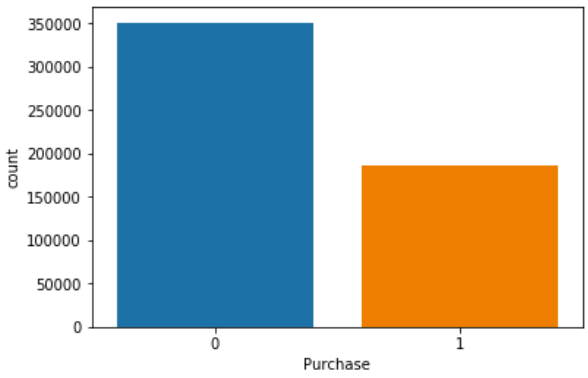
Fig. 9. Purchases Distribution



Source: Own analysis with data from kaggle.

As we can see there there's a right tale. For this dataset the prediction of purchases greater than $10,000 USD dollars will be predicted as the target class. After creating the class, the distribution is seen in the *Figure 10.*

Fig. 10. Class purchase



Source: Own analysis with data from kaggle.

For this dataset the prediction of purchases greater than $10,000 USD dollars will be predicted as the target class.

IV. Experiments

In this section, all the experiments and analyses that were performed are described for each dataset.

A. Credit Card Fraud Detection

As part of the analysis, an autoencoder was performed for dimensionality reduction. When exploring the data, the main highlights of the data analysis were that there were no missing values, so no imputation was needed.

1) Sample splitting: To estimate the autoencoder, the data was split into two groups: a training and a validation set, consisting of 80% and 20% of the data respectively. The partition of the original dataset was done using random selection with a fixed seed. Validation was done over the partition to corroborate that there are similar distribution of the two classes in each of the datasets. The results of this validation are presented in *Figure 11.*

Fig. 11. Class distribution in original and splitted datasets

	ALL DATA 284,807 obs. (100%)		TRAINING SPLIT 227,845 obs. (80%)		VALIDATION SPLIT 56,962 obs. (20%)	
CLASS	Number of obs.	Percentage	Number of obs.	Percentage	Number of obs.	Percentage
0	284,315	99.82%	227,450	99.82%	56,865	99.82%
1	492	0.0017%	395	0.0017%	97	0.0017%
Total	284,807	100%	227,845	100%	56,962	100%

Source: Own analysis with data from kaggle.

2) Model definition and performance scores: The autoencoder was trained with 2 encoder layers that fed the decoder, and then take the rectified linear unit and the hyperbolic tangent as activation functions to produce the outputs. The autoencoder was trained for 10 epochs and the Mean Squared Error (MSE), that is the average squared difference between the estimated values and what is estimated, was taked to measure the loss.[15] Additionally, the optimizer used was *Adadelta*, which is a optimizer that adapts learning rates based on a moving window of gradient updates and the initial learning rate and decay factor are not necessary to be set.[16]

Figure 12 shows the change in the validation loss while training for 10 epochs.

After finishing the training 20 features were selected out of 30 with a Validation Final Loss of 0.1141.

Once the model was trained, the representation of the input learned by the model was obtained. This was done accessed by the weights of the trained model creating another network containing sequential layers that helped predicted the new classes that were also trained on the dataset with the SVM classifier.

To test the performance of the SVM classifier, firstly the confusion matrix is shown and analysed in *Figure 13.*

On the confusion matrix it can be seen that 56,854 are true negative predicted observations, 8 are false negative, 69 are false positive predicted observations and 31 true positive observations.

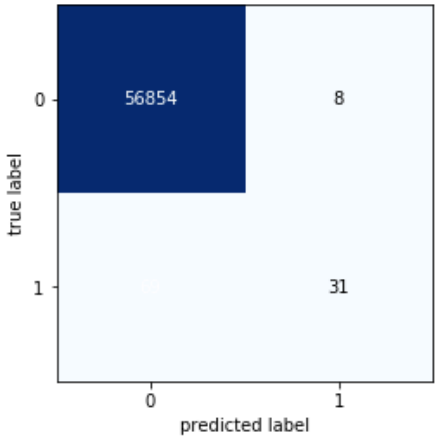
The percentage of correct predictions (accuracy) over the training set is 99%. When analysing the sensitivity

Fig. 12. Epochs at each stage with validation scores

EPOCH	VAL LOSS
Epoch 1/10	val_loss: 0.1102
Epoch 2/10	val_loss: 0.1113
Epoch 3/10	val_loss: 0.1113
Epoch 4/10	val_loss: 0.1123
Epoch 5/10	val_loss: 0.1123
Epoch 6/10	val_loss: 0.1132
Epoch 7/10	val_loss: 0.1134
Epoch 8/10	val_loss: 0.1137
Epoch 9/10	val_loss: 0.1140
Epoch 10/10	val_loss: 0.1141

Source: Own analysis with data from kaggle.

Fig. 13. Confusion Matrix for Credit Card Fraud Detection



Source: Own analysis with data from kaggle.

and specificity of the predictions, the first score is 79.5% which means that the actual 1 values are correctly predicted, whereas the second score is high with 99% of the actual 0 values are correctly predicted. This is because the model predicts a much higher percentage of *zeros*. On the other hand, the ROC, that is the sensitivity and specificity ratio is 65.49%.

B. Cervical Cancer Risk Classification

1) Sample splitting: To estimate the autoencoder, the data was split into two groups: a training and a validation set, consisting of 80% and 20% of the data respectively. The partition of the original dataset was done using random selection with a fixed seed. Validation was done over the partition to corroborate that there are similar distribution of the two classes in each of the datasets. The results of this validation are presented in *Figure 14*.

2) Model definition and performance scores: As the previous dataset, the autoencoder was trained with 2 encoder layers that fed the decoder, and took the rectified linear unit and the hyperbolic tangent as activation functions to produce the outputs. The autoencoder was trained for 10 epochs and

Fig. 14. Class distribution in original and splitted datasets

	ALL DATA 858 obs. (100%)		TRAINING SPLIT 686 obs. (80%)		VALIDATION SPLIT 172 obs. (20%)	
CLASS	Number of obs.	Percentage	Number of obs.	Percentage	Number of obs.	Percentage
0	840	97.90%	674	98.20%	166	96.51%
1	18	0.020%	12	0.017%	6	0.034%
Total	858	98%	686	98%	172	97%

Source: Own analysis with data from kaggle.

the Mean Squared Error (MSE) was taked to measure the loss. Additionally, the optimizer used was *Adadelta*.

Figure 15 shows the change in the validation loss while training for 10 epochs.

Fig. 15. Epochs at each stage with validation scores

EPOCH	VAL LOSS
Epoch 1/10	val_loss: 0.1106
Epoch 2/10	val_loss: 0.1067
Epoch 3/10	val_loss: 0.1031
Epoch 4/10	val_loss: 0.1000
Epoch 5/10	val_loss: 0.0972
Epoch 6/10	val_loss: 0.0945
Epoch 7/10	val_loss: 0.0921
Epoch 8/10	val_loss: 0.0898
Epoch 9/10	val_loss: 0.0876
Epoch 10/10	val_loss: 0.0856

Source: Own analysis with data from kaggle.

After finishing the training 20 features were selected out of 33 with a Validation Final Loss of 0.0792.

Just like before, once the model was trained, the representation of the input learned by the model was obtained. This was done accessed by the weights of the trained model creating another network containing sequential layers that helped predicted the new classes that were also trained on the dataset with the SVM classifier.

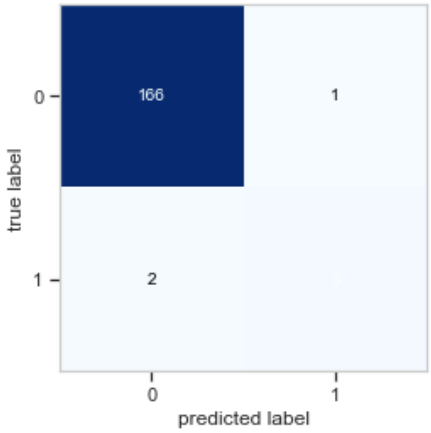
To test the performance of the SVM classifier, firstly the confusion matrix is shown and analysed in *Figure 16*.

On the confusion matrix it can be seen that 166 are true negative predicted observations, 2 are false negative, 3 are false positive predicted observations and 1 true positive observations. The percentage of correct predictions (accuracy) over the training set is 98.3%. When analysing the sensitivity and specificity of the predictions, the first score is 75% which means that the actual 1 values are correctly predicted, whereas the second score is high with 98.8% of the actual 0 values are correctly predicted. This is because the model predicts a much higher percentage of *zeros*. On the other hand, the ROC, that is the sensitivity and specificity ratio is 79.7%.

C. Black Friday

1) Sample splitting: To estimate the autoencoder, the data was split into two groups: a training and a validation set, consisting of 80% and 20% of the data respectively. The

Fig. 16. Confusion Matrix for Cervical Cancer Classification



Source: Own analysis with data from kaggle.

partition of the original dataset was done using random selection with a fixed seed. Validation was done over the partition to corroborate that there are similar distribution of the two classes in each of the datasets. The results of this validation are presented in *Figure 17*.

Fig. 17. Class distribution in original and splitted datasets

	ALL DATA 537,577 obs. (100%)		TRAINING SPLIT 430,061 obs. (80%)		VALIDATION SPLIT 107,516 obs. (20%)	
CLASS	Number of obs.	Percentage	Number of obs.	Percentage	Number of obs.	Percentage
0	350,920	65.30%	280,865	65.30%	70,055	65.30%
1	186,657	34.7%	149,196	34.7%	37,461	34.7%
Total	537,577	100%	430,061	100%	107,516	100%

Source: Own analysis with data from kaggle.

2) Model definition and performance scores: As the previous dataset, the autoencoder was trained with 2 encoder layers that fed the decoder, and took the rectified linear unit and the hyperbolic tangent as activation functions to produce the outputs. The autoencoder was trained for 10 epochs and the Mean Squared Error (MSE) was taked to measure the loss. Additionally, the optimizer used was *Adadelta*.

Figure 18 shows the change in the validation loss while training for 10 epochs.

After finishing the training 20 features were selected out of 33 with a Validation Final Loss of 0.0240.

Just like before, once the model was trained, the representation of the input learned by the model was obtained. This was done accessed by the weights of the trained model creating another network containing sequential layers that helped predicted the new classes that were also trained on the dataset with the SVM classifier.

To test the performance of the SVM classifier, firstly the confusion matrix is shown and analysed in *Figure 19*.

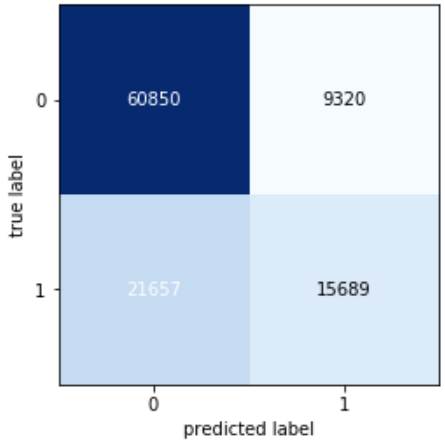
On the confusion matrix it can be seen that 60,850 are true negative predicted observations, 9,320 are false negative, 21,657 are false positive predicted observations and 15,689 true positive observations. The percentage of correct predictions (accuracy) over the training set is 71.2%. When analysing the sensitivity and specificity of the predictions, the first score is 62.7% which means that the actual 1 values

Fig. 18. Epochs at each stage with validation scores

EPOCH	VAL LOSS
Epoch 1/10	val_loss: 0.0527
Epoch 2/10	val_loss: 0.0353
Epoch 3/10	val_loss: 0.0468
Epoch 4/10	val_loss: 0.0289
Epoch 5/10	val_loss: 0.0284
Epoch 6/10	val_loss: 0.0262
Epoch 7/10	val_loss: 0.0271
Epoch 8/10	val_loss: 0.0267
Epoch 9/10	val_loss: 0.0265
Epoch 10/10	val_loss: 0.0240

Source: Own analysis with data from kaggle.

Fig. 19. Confusion Matrix for Purchases prediction on Black Friday



Source: Own analysis with data from kaggle.

are correctly predicted, whereas the second score is high with 73.8% of the actual 0 values are correctly predicted. On the other hand, the ROC, that is the sensitivity and specificity ratio is 73.8%.

V. Discussion

In previous sections, an approach of dimensionality reduction techniques through autoencoders were presented. It was observed that dimensionality reduction is the transformation of highdimensional data into a meaningful representation of reduce dimensionality that is able to facilitate classification and visualisation.

The approach was reproduced for the three selected datasets. As the goal of autoencoders is to produce a good representation of the input data, the autoencoders were trained until their MSE were small enough and the dimensionality reduction was achieved. For that, the output label from the data or class was removed for the training and the constructed encoder and decoder performed the training searching for the minimum reconstruction error. Additionally, the features resulted from each experiment were again trained for another network that helped provide

the new dataset that then was used to train the Support Vector Machine algorithm, which identified which class the input belong to *class = 1* or *class = 0* by making a decision based on the value of a linear combination of the features. The model performance was measure at each stage with the validation scores.

For each experiment, accuracy, sensitivity and specificity were estimated. In the first experiment, an accuracy of 99% is observed. On the other hand, ROC is approximately 65.49%. In the second experiment, the classifier performed an accuracy of 98.3%, predicting a large percentage of *zeros*, just like in the first experiment. Finally, the third experiment performed an accuracy of 71.2%, which is the lowest accuracy of all experiments. The observed results vary because of the nature of each data set.

As it was mentioned before, the performance depends on the hyper parameters, learning rate, activation function and auxiliary features. These results indicates that changing the number of features or learning rate would lead to a change on the classifier performance. Moreover, further work on heavy data preprocessing, changing the number of auxiliary features would be done to improve the performance on the autoencoder and on the classifier.

VI. Conclusion

In conclusion, supervised learners are trained over regression or classification algorithms with target label and input data features. However when optimising the classification process a lot of data preprocessing techniques have been suggested specially for highdimensionality data. As the number of dimension increases it become more difficult to process it. For that reason, this work performed autoencoders for dimensionality reduction that helped identified the pattern involved in the data to after be used into a linear classifier.

Autoencoders are an unsupervised tool that is used for feature learning. It tries to learn an approximation to the identity function that later is able to give information from the data. The dimensionality reduction achieved with autoencoders was used to defining new discriminant targets, learning representation of the data and then used it to train a SVM classifier in three different databases.

In this work, autoencoders were created to reduce dimensionality and train a linear classifier, the results showed that high accuracy was achieved but further work must be done including data preprocessing and varying the encoderdecoder architecture such as: learning rate, number of epochs, activation function and auxiliary features.

References

[1] Angshuman, Paul, et al., *"Discriminative Autoencoder"*, 2018.

[2] Van der Maaten, Laurens, et al., *"Dimensionality Reduction: Acomparative Review"*, Tiburg University, The Netherlands, 2009.

[3] Gales, M.J.F., et al.,*"Discriminative classifiers with adaptive kernes for noise robust speech recognition*, Cambridge University Engineering Department. 2009.

[4] Yadav, Ajay*"Support Vector Machines"*, Towards Data Science, 2018. Online resource: `https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589`

[5] Chollet, Francois, *"Building Autoencoders in Keras"*, The Keras Blog, 2016. Online resource: `https://blog.keras.io/building-autoencoders-in-keras.html`

[6] Wang, Weiran, et al., *"The role of Dimensionality Reduction in Classification*, Association for the Advancement of Artificial Intelligence, University of California, 2014.

[7] Ng, Andrew, et al., *"Unsupervised Feature Learning and Deep Learning"*, Sandford University, 2012. Online resource: `http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial`

[8] Kumar Pal, Ashwini*"Dimension Reduction Autoencoders"*, Machine Learning Blog, 2018. Online resource: `https://blog.paperspace.com/dimension-reduction-with-autoencoders/`

[9] Credit Card Fraud Detection. Anonymized credit card transactions labeled as fraudulent or genuine. 2018. Online resource: `https://www.kaggle.com/mlg-ulb/creditcardfraud`

[10] Cervical Cancer Risk Classification. Prediction of cancer indicators. Online resource: `https://www.kaggle.com/loveall/cervical-cancer-risk-classification`

[11] Black Friday. A study of sales through consumer behaviours. Online resource: `https://www.kaggle.com/mehdidag/black-friday`

[12] Nousi, Paraskevi, et al.,*"Deep learning algorithms for discriminant autoencoding"*, Department of Informatics, Aristotle University of Thessaloniki, Greece, 2017.

[13] WHO, World Health Organization, *"Cervical cancer"*, 2019. Online resource: `https://www.who.int/cancer/prevention/diagnosis-screening/cervical-cancer/en/`

[14] Kaggle, *"How autoencoders work? Intro and use cases"*, 2019. Online resource: `https://www.kaggle.com/shivamb/how-autoencoders-work-intro-and-usecases`

[15] Lehmann, E. L, et al., *"Theory of Point Estimation"*, New York: Springer, 1998.

[16] Keras Documentation, *"Optimizers"*, 2019. Online resource: `https://keras.io/optimizers/?fbclid=IwAR1ZfUDsFuWagHhXF3sH_ZOav6odTh4EBxaeSyGHFZjSV7VJJMcwUQezlG4`