Data Classification

Introduction

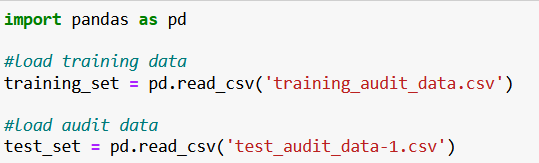
In this experimental report, the goal is to classify the training and test data extracted from the Audit dataset to determine which firm is fraudulent based on the risk factors that are present. To do this, we use an unsupervised machine learning algorithm called Principal Component Analysis (PCA) to reduce the dimensionality (number of features) within the dataset while still retaining as much information as possible (AWS, n.d.). PCA is useful for classification tasks such as this one where there are many input features present in the form of the tabular data (AWS, n.d.). After obtaining the principal components, the test-transformed PCA data is used to calculate the confusion matrix needed for building the Support Vector Machine (SVM) classifier. The data is assumed to be of satisfactory quality and requires only normalization before being used in modeling.

Task 1 – Data Exploration

The given Audit dataset consists of two files, audit\_training\_data.csv for training and audit\_data\_test-1.csv for testing. The training set contains 391 instances, each representing a firm and ten input features. The last column of the file is the class label, which indicates whether the firm is considered high risk (indicated by label 1) or low/no risk (indicated by label 0). While observing the files, it is determined that there are 202 high risk and 189 low risk/no risk instances present.

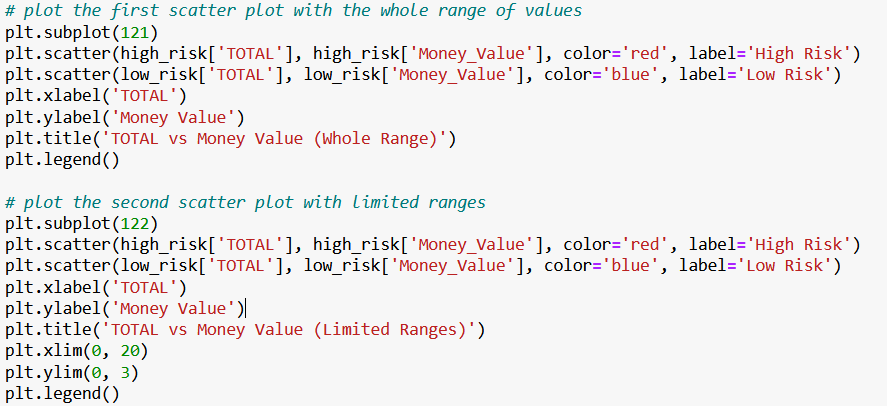
1. To perform Principal Component Analysis (PCA), the python code used above imports the Pandas library using “pd” alias. The “read\_csv” function from Pandas in both lines is used to read the files into a data frame called training\_set and test\_set respectively (Li, 2023).

The original training set is denoted as training set (I).

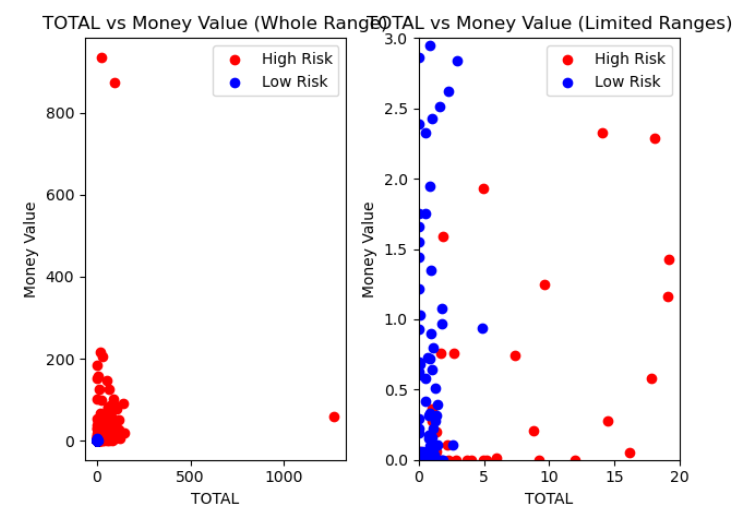


1. Separation of input features is the next step in Principal Component Analysis (PCA).

The key python code is included here while rest of it can be found in the appendix section of this report.

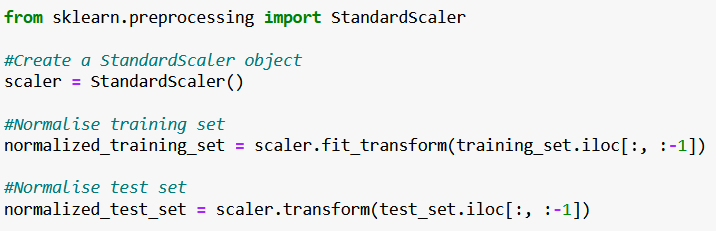


The ‘subplot’ function is used to create two subplots within one figure where the number 121 indicates the number of rows, columns and subplot number respectively (John Hunter, Darren Dale, Eric Firing, Michael Droettboom, Matplotlib Development Team, 2023). The ‘xlim’ and ‘ylim’ functions are used to set the x and y limits for the second subplot (Matplotlib Development Team, 2023).



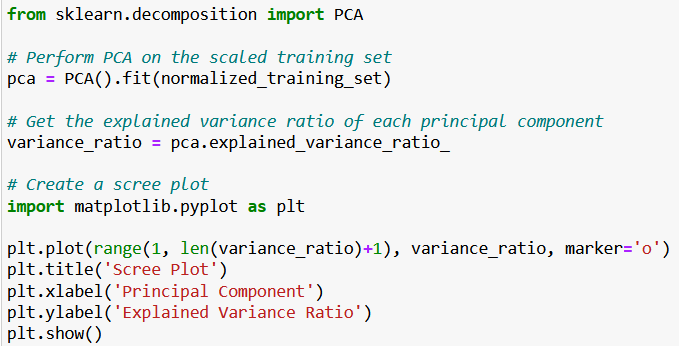
The first scatter plot shows the entire range of values for both ‘TOTAL’ and ‘Money Value’ while the second scatter plot only shows values for both ‘TOTAL’ in the range [0, 20] and ‘Money Value’ in the range [0, 3] to assess the relationship between the two variables at different scales (Jake VanderPlas, November 2016). From the generated plots, it can be observed that the high-risk points (in red) tend to have higher ‘TOTAL’ and ‘Money Value’ values than the low-risk points (in blue).

1. Moving on, the input features should be normalised in both the datasets to ensure that they are on the same scale.

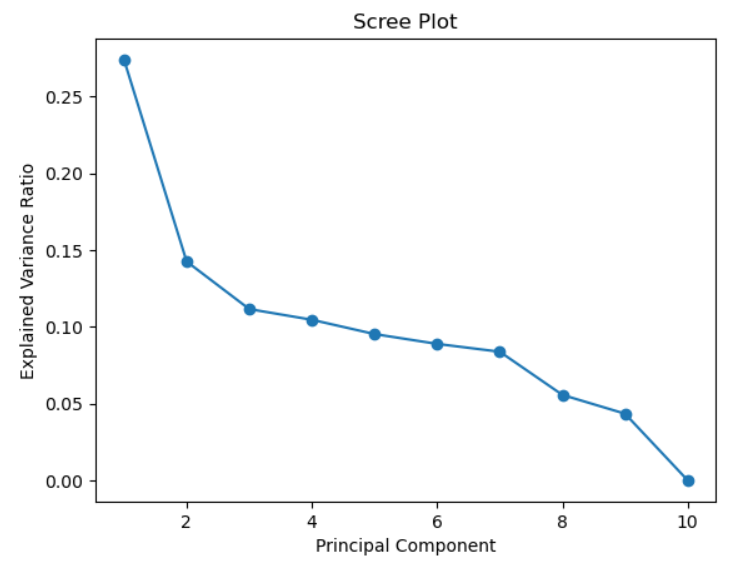


The python code used creates a ‘StandardScaler’ object and the fit\_transform() method is used to transform the scale of the data points in the ‘training\_set’ except for the last column which is the target variable (Badole, 2021). The transformed data is then stored in the normalized\_training\_set. The same applies to the normalized\_test\_set.

1. PCA is then used to reduce the dimensionality of the input features in both datasets.

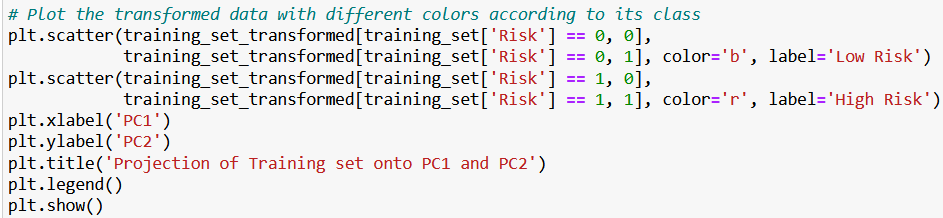


This code performs a Principal Component Analysis (PCA) on the normalized training set and then plots the scree plot to report the variances captured by each principal component. The PCA function is used without any parameter to fit the normalized training set.

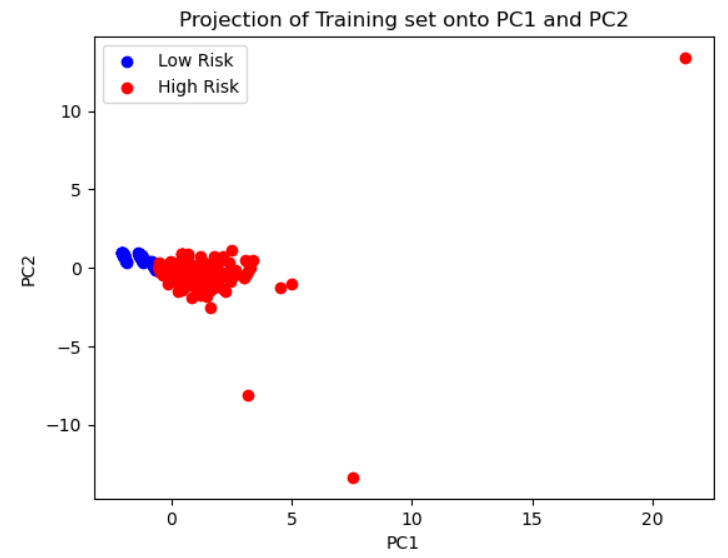


The figure shows the scree plot with cumulative explained variance ratio on the y-axis and the number of principal components on the x-axis. The scree plot shows the decreasing amount of variance explained by each principal component and can be used to determine the optimal number of principal components to use in the next task.

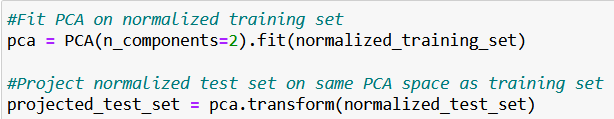
1. The full code used for this task can be found in the appendix section of this report.



This code performs PCA on the scaled training set and transforms the data onto the PC1 and PC2 axes. The transformed data is then plotted with different colors according to its class label (low or high risk) on the x-axis and y-axis separately.

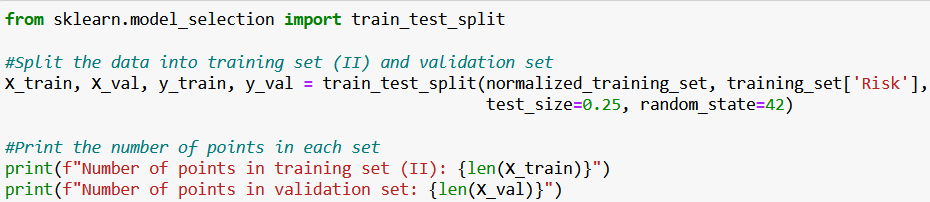


From the figure, a scatter plot is obtained showing the projection of the training set onto PC1 and PC2 with data points colored by class (low risk - blue, high risk - red). We can see that the data points from the two classes are separated from each other, indicating that PC1 and PC2 are both useful to differentiate between low-risk and high-risk samples.



1. The code uses the transform() method to project the normalized test set on the same PCA space as the training set and stores it in a variable called projected\_test\_set.

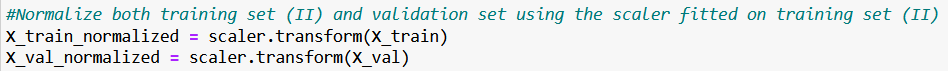
Task 2



1. The python code above uses the ‘train\_test\_split’ function from Scikit-learn’s ‘model\_selection’ module to split the normalized training set into a new set denoted by ‘X\_train’ and ‘y\_train’ as well as a validation set, ‘X\_val’ and ‘y\_val’ (Li, 2023). 25% of the data is used for validation and 75% is used for training. The ‘random\_state’ function is a random seed set to 42 that ensures the split is reproducible (Bansal, 2020).



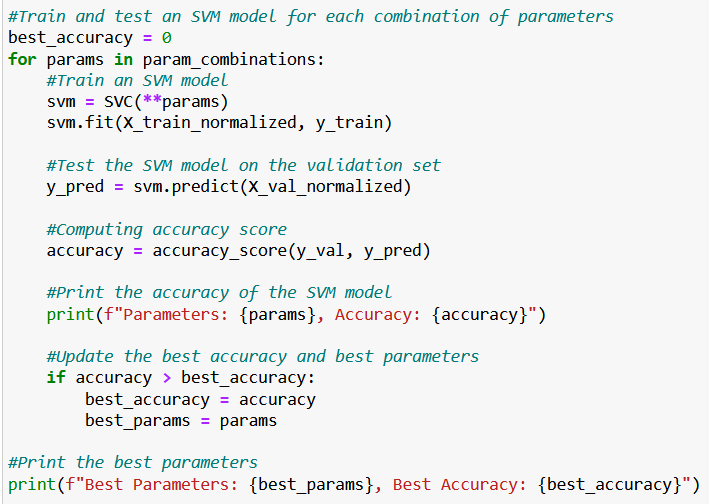
The output generated by the above code contains 293 points in training set (II) and 98 points in validation set. It is important to note that the model does not use this dataset to learn but to estimate the generalization of new data.



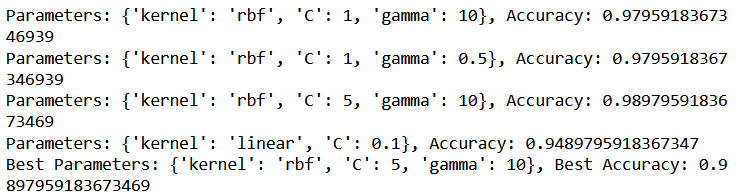
1. The python code above uses the StandardScaler() object fitted on training set to normalize the training set (II) and validation set.

Task 3 - SVM Classification

1. This task requires us to train an SVM model on the PCA-transformed training data using both Gaussian RBF and linear kernels predicting output labels for normalization and training set and calculate the accuracy score for each parameter on the validation set.

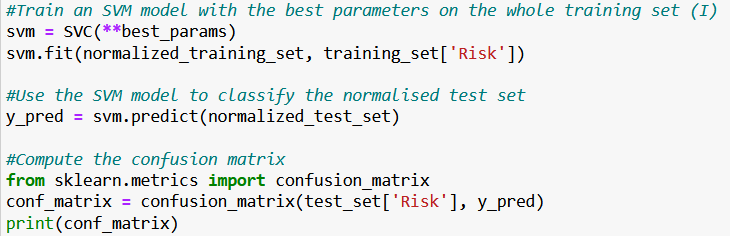


1. The full code for this task can be found in the appendix section of this report. The above code shows us that the parameters have been defined and fit to the normalized training set to create the SVM model using SVC function. ‘best\_accuracy’ is initialized to 0 to keep track of the best accuracy achieved by the SVM model and ‘best\_params’ are set to none. There is a loop through each set of parameters in param\_ combinations.



From the output code, the best parameters is {'kernel': 'rbf', 'C': 5, 'gamma': 10} with a validation accuracy score of 0.987959183673469. This means that the SVM model with the RBF kernel of C = 5 and gamma = 10 yields a higher score compared to linear SVM model and is the best among the four models (Sowndarya Krishnamoorthy, 2018).

1. This task requires us to do further testing on the SVM model and evaluate its performance to produce a confusion matrix.



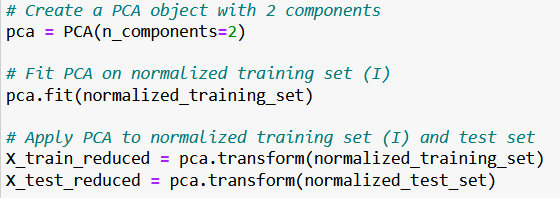
The best parameters are selected from the validation and used to train the SVM model on the whole training set (I) using the SVC function. The trained model is used to predict class labels and a confusion matrix is generated to give a summary of the classification results. The matrix shows number of true positives, false positives , true negatives and false negatives (Bhandari, 2023).

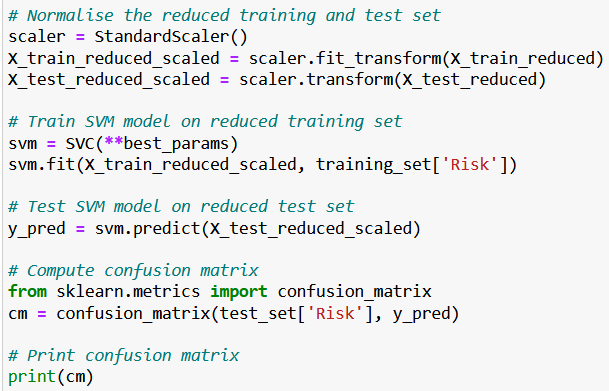


The confusion matrix shows the SVM model correctly predicted 78 instances of low risk and 241 instances of high risk but incorrectly predicted 14 instances of low risk as high risk and 43 instances of high risk as low risk.

Advanced Task 3 (b)

1. From looking at the scree plot generated in Task 1 (d), we can see the elbow point occurs after the number of principal components is 2. The explained variance ratio begins to decrease and adding more principal components does not result in it being explained by the model. Based on the plot, it seems that 2 principal components are enough to explain a large percentage of the total variance (Mangale, 2020).
2. The code below shows the features being reduced for both normalized training set (I) and test set with the number of selected principal components being 2.



1. The final task needs us to fine tune the SVM parameters and improve the SVM model by testing it with the reduced features.

After predicting the reduced test set, the confusion matrix is computed to compare values for each class.



In this case, the SVM model predicted 92 high risks and 89 low risk instances. There were 0 false positives (predicted as high risk but is low risk) and 195 false negatives (predicted as low risk but actually is high risk).

Task 4 - Conclusion

From the analysis performed, it can be concluded that the SVM model with parameter values C = 5 and gamma = 10 trained on the original features received the highest accuracy on the validation set and also performed well on the test set with an accuracy of 98.97%.

After the features were reduced using PCA, the SVM model trained on the reduced features received an accuracy of 94.97% on the linear kernel which is slightly lower than the model trained on the original features.

The scree plot showed that the variance could be explained by the first two principal components. Hence, reducing the number of principal did not reduce the ability of the model.

Overall, the SVM model performed well on both sets, original and reduced. Reducing the model was a time consuming process but made it better in classification.

The results obtained suggest that the SVM model with the selected reduced parameters values provided a more accurate classification of the dataset with zero false negatives reported.

References

AWS, n.d. *Principal Component Analysis (PCA) Algorithm.* [Online]   
Available at: https://docs.aws.amazon.com/sagemaker/latest/dg/pca.html  
[Accessed 20 March 2023].

Badole, M., 2021. *Difference Between fit(), transform(), and fit\_transform() Methods in Scikit-Learn.* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2021/04/difference-between-fit-transform-fit\_transform-methods-in-scikit-learn-with-python-code/  
[Accessed 27 March 2023].

Bansal, J., 2020. *How to Use Random Seeds Effectively.* [Online]   
Available at: https://towardsdatascience.com/how-to-use-random-seeds-effectively-54a4cd855a79  
[Accessed 28 March 2023].

Bhandari, A., 2023. *Understanding & Interpreting Confusion Matrices for Machine Learning (Updated 2023).* [Online]   
Available at: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/  
[Accessed 02 April 2023].

Jake VanderPlas, November 2016. *Python Data Science Handbook.* s.l.:O'Reilly Media, Inc..

John Hunter, Darren Dale, Eric Firing, Michael Droettboom, Matplotlib Development Team, 2023. *matplotlib.pyplot.subplot.* [Online]   
Available at: https://matplotlib.org/stable/api/\_as\_gen/matplotlib.pyplot.subplot.html  
[Accessed 25 March 2023].

Li, J., 2023. *Generate classification report and confusion matrix in Python.* [Online]   
Available at: https://www.projectpro.io/recipes/generate-classification-report-and-confusion-matrix-in-python  
[Accessed 23 March 2023].

Mangale, S., 2020. *Scree Plot.* [Online]   
Available at: https://sanchitamangale12.medium.com/scree-plot-733ed72c8608  
[Accessed 28 August 2023].

Matplotlib Development Team, 2023. *matplotlib.pyplot.ylim.* [Online]   
Available at: https://matplotlib.org/stable/api/\_as\_gen/matplotlib.pyplot.ylim.html  
[Accessed 25 March 2023].

Sowndarya Krishnamoorthy, L. R. S. S. H. E., 2018. *Identification of User Behavioral Biometrics for Authentication Using Keystroke Dynamics and Machine Learning.* [Online]   
Available at: https://www.researchgate.net/figure/Comparision-of-classification-accuracy-for-linear-svm-and-RBF-svm-with-grid-search-having\_fig4\_327026450  
[Accessed 30 March 2023].

APPENDIX

