Research Outline

1. Statistical Analysis & Dimensional Reduction Algorithms
    1. Principal Component Analysis (PCA) [x]
    2. Multiple Factor Analysis (MFA) [TBD] [ ]
    3. Autoencoders [x]

2. Classification Algorithms
    1. Random Forest [x]
    2. Isolation Forest [x]
    3. Local Outlier Factor (LOF) [x]
    4. Deep Neural Network [x]

3. Advanced Anomaly Detection Algorithms
    1. Autoencoder Forest [x]

4. Datasets
    1. Iris Dataset
    2. Credit Card Fraud Dataset
    3. Spotify Music Dataset

Statistical Analysis & Dimensional Reduction Algorithms

(1.1) Principal Component Analysis

Goals
Review how PCA works, create working PCA program in python.

Background
Principal component analysis can be used for dimensional reduction by finding correlations between features in a dataset. We can find the PC-lines as follows:

Given a matrix A, with N samples and M measurements

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

First, we get the covariance matrix:

$$B = A^T A$$

Next, we get the Eigen values and Eigen vectors for the covariance matrix: W, λ
Finally, we get the scores using the Eigen vectors (loadings) and the original data

$$T = XW$$

We can use the Eigen values to determine which PC-lines are the most important (higher values denote higher importance). We can choose between 1-N dimensions.

Findings
PCA can be very useful for dimension reduction while still keeping the majority of the variance of the original dataset. PCA only works on continuous data. As long as we keep track of the loadings (Eigen vectors) generated in PCA, we can easily convert between the original data and the modified PCA data (scores).
Github: https://github.com/BrendanCReidy/DataScienceResearch/blob/main/pca.py

PCA on Spotify Dataset

Introduction

The Spotify Dataset has over 28,000 songs, each with 16 metrics for the music including genre(s), artist(s), tempo, valence …etc. The dataset was trimmed down into only the 11 variables that are continuous to perform PCA (acousticness, danceability, duration (ms), energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, popularity).

Findings

Based on the results (table 1.1.1), PCA does not significantly reduce the number of dimensions for this dataset (we can eliminate at most 2 dimensions if we want to retain the majority of the information in the dataset). In most cases, PCA can reduce the number of dimensions in a dataset at the cost of losing some information. If we want to keep a high percentage of the information (>=99%), PCA may not be able to help unless the data is highly correlated (redundant features). Based on this information, the Spotify Dataset is likely a poor candidate for PCA, since all of the dimensions are highly independent.

Results

Table 1.1.1: Spotify Dataset: %Variance Covered VS Number of PC-Lines

Number of PC-Lines	Percentage of Variance Covered
1	48.5%
2	65.2%
3	77.6%
4	84.8%
5	89.2%
6	92.9%
7	95.4%
8	97.6%
9	99.4%
10	99.9%
11	100%

References

- Dimensionality Reduction: Principal Components Analysis, Part 1: https://www.youtube.com/watch?v=ZqXnPcyIAL8
- Data Analysis 6: Principal Component Analysis (PCA) - Computerphile: https://www.youtube.com/watch?v=TJdH6rPA-TI
- What does it mean when PCA does not produce a reduction in dimensionality?: https://stats.stackexchange.com/questions/451619/what-does-it-mean-when-pca-does-not-produce-a-reduction-in-dimensionality

(1.3) Auto Encoder

Goals
Investigate using auto encoders and compare them to PCA. Create an auto encoder using the Spotify Dataset and compare to PCA.

Background
Auto encoders can be used as a dimensional reduction tool for high dimensional data. Unlike PCA, auto encoders can learn non-linear relationships in data. The auto encoder is an hour-glass shaped Deep Neural Network, where the input and output dimensions are the same. The goal of the auto encoder is to “squeeze” the data into as few dimensions as possible, and then accurately reconstruct the data at the output. Once the network is trained, it can be split into two networks: the encoder network (the descending part of the network), and the decoder network (the ascending part of the network).

Auto Encoder on Spotify Dataset

Introduction
The auto encoder uses the same 11 features that were used in PCA. The auto encoder has a topology of 11x256x128xNx128x256x11, where N is the number of encoding neurons. The network uses the Sigmoid activation function for the output layers, and RELU for the rest of the layers.

Findings
The auto encoder has surprisingly high accuracy for small numbers of encoding neurons (table 1.3.1). For small numbers of features, the auto encoder out performs PCA however, the auto encoder should only be used in the most extreme cases of dimensional reduction due to the significant loss in information. Even with 11 encoding neurons (0 dimensional reduction), the auto encoder loses 5.2% of information from input to output. For a higher number of encoding neurons, the auto encoder performs significantly worse than PCA. Auto encoders also have another significant drawback; the amount of time it takes to train. If you already have high dimensional data, an auto encoder might be redundant due to the amount of time it takes to train. However, unlike PCA, auto encoders can be optimized to yield higher accuracy with enough tuning; although it may still be too inaccurate to be used.

Results

Table 1.3.1: Spotify Dataset: Auto Encoder Accuracy VS Number of Encoding Neurons

Number of Encoding Neurons	Accuracy
1	73.1%
2	80.6%
3	79.2%
4	86.6%
5	88.2%
6	87.6%
7	88.8%
8	92.9%
9	93.9%
10	95.3%
11	94.8%

Table 1.3.1: The number of encoding neurons is the number of neurons in the middle-most layer of the neural network. The accuracy is a measurement of how close the reconstructed output values are to the original data.

Further Reading

Autoencoders vs PCA: when to use?: https://towardsdatascience.com/autoencoders-vs-pca-when-to-use-which-73de063f5d7

References

- Building Autoencoders in Keras: https://blog.keras.io/building-autoencoders-in-keras.html

Classification Algorithms

(2.1) Random Forest

Goals
Review how Random Forest works. Get Random Forest working on credit card fraud dataset using scikit-learn in python.

Background
Random forest is based on decision trees. Random Forest generates a “forest” of decision trees. For prediction, each decision tree gets a vote on the classification. The classification with the most votes is taken to be the predicted classification.

Findings
Random Forest works relatively well on unbalanced data. Random Forest can maintain accuracy when large portions of the data are missing. Although the model has good accuracy, accuracy cannot be used as a meaningful metric when evaluating unbalanced data. However, if you balance the data; the classification metrics are much better.

Results

Table 2.1.1: Credit Card Fraud Detection Metrics for Random Forest

	Precision	Recall	F1-score	Support
Not Fraud	1.00	1.00	1.00	93828
Fraud	0.96	0.74	0.83	159
Accuracy	-	-	1.00	93987
Macro avg	0.98	0.87	0.92	93987
Weighted avg	1.00	1.00	1.00	93987

Table 2.1.1: Classification metrics for Random Forest on Credit Card Fraud Dataset (Test size of 33% and random state of 21)

Table 2.1.2: Balanced Credit Card Fraud Detection Metrics for Random Forest

	Precision	Recall	F1-score	Support
Not Fraud	0.93	0.95	0.94	165
Fraud	0.95	0.93	0.94	160
Accuracy			0.94	325
Macro avg	0.94	0.94	0.94	325
Weighted avg	0.94	0.94	0.94	325

Table 2.1.2: Classification metrics for random forest on Balanced Credit Card Fraud Dataset (Test size of 33% and random state of 21)

References

- Accuracy, Precision, Recall or F1?: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9
- Credit Card Fraud Detection using Random Forest: https://www.kaggle.com/hassanamin/credit-card-fraud-detection-using-random-forest
- Fraud Detection | SVM, Random Forest and CNN https://www.kaggle.com/kayademirs/fraud-detection-svm-random-forest-and-cnn#Random-Forest
- Credit Card Fraud Detection (dataset): https://www.kaggle.com/mlg-ulb/creditcardfraud/tasks?taskId=77

(2.2) Isolation Forest

Goals
Research Isolation Forest algorithm. Get Isolation Forest working on credit card fraud dataset using scikit-learn in python.

Background
Isolation forest is an unsupervised learning algorithm based on decision trees. Isolation forest is designed to find outliers in high dimensional space. Similar to random forest, isolation forest creates a “forest” of decision trees. Isolation forest creates decision trees by splitting the data randomly into two sections in different dimensions. The process is repeated many times until only one data point is left in the within the splits; once the data is “isolated” (or until the depth reaches a defined stopping point). This process is repeated many times to create a “forest”. A data point is determined to be an outlier based on the average number of splits to isolate that data point (higher numbers of splits correspond to inliers whereas lower numbers of splits correspond to outliers).

Findings
Isolation Forest performs better on unbalanced data (the balanced data had 0 for precision and recall for the fraud class) than on balanced data however, the performance is much worse than expected when compared to Random Forest.

Results

Table 2.2.1: Credit Card Fraud Detection Metrics for Isolation Forest

	Precision	Recall	F1-score	Support
Not Fraud	1.00	1.00	1.00	284031
Fraud	0.31	0.31	0.31	491
Accuracy	-	-	1.00	284522
Macro avg	0.67	0.65	0.65	284522
Weighted avg	1.00	1.00	1.00	284522

Table 2.2.1: Classification metrics for Isolation Forest on Credit Card Fraud Dataset

References:

- Accuracy, Precision, Recall or F1?: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9
- Credit Card Fraud Detection using Random Forest: https://www.kaggle.com/hassanamin/credit-card-fraud-detection-using-random-forest

- Credit Card Fraud Detection (dataset): https://www.kaggle.com/mlg-ulb/creditcardfraud/tasks?taskId=77
- Jan van der Vegt: A walk through the isolation forest | PyData Amsterdam 2019: https://www.youtube.com/watch?v=RyFQXQf4w4w

(2.3) Local Outlier Factor

Goals
Research Local Outlier Factor (LOF) algorithm. Get LOF working on credit card fraud dataset using scikit-learn in python.

Background
Local outlier factor uses

Findings

Results

Table 2.3.1: Balanced Credit Card Fraud Detection Metrics for LOF

	Precision	Recall	F1-score	Support
Not Fraud	0.90	0.90	0.90	492
Fraud	0.90	0.90	0.90	491
Accuracy	-	-	0.90	983
Macro avg	0.90	0.90	0.90	983
Weighted avg	0.90	0.90	0.90	983

Table 2.3.1: Classification metrics for Isolation Forest on Credit Card Fraud Dataset (n_neighbors=492, contamination = 0.5)

References:

- Accuracy, Precision, Recall or F1?: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9
- LOF: Identifying Density-Based Local Outliers: https://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf

(2.4) Deep Neural Network

Goals
Design a simple neural network that performs as good or better than the random forest model

Background
Deep Neural Networks can be used to map input to output using back propagation and gradient descent. Deep Neural Networks find non-linear patterns between input data and output data

Introduction
The neural network has a topology of 30x200x1. The network uses hyperbolic tangent for the middle layer activation and sigmoid for the output activation. A dropout of 0.2 is used for the first and second layer. The inputs to the network are the PC-lines in the Balanced Credit Card Fraud Dataset with no normalization applied.

Findings
The neural network performed better than Random Forest for the balanced dataset. The results in the tables are for the best networks. Although an accuracy of 0.94 seems reasonable, when this accuracy is applied to the normal dataset this equates to 15,000 false negatives.

Results

Table 2.4.1: Credit Card Fraud Detection Metrics for Deep Neural Network

	Precision	Recall	F1-score	Support
Not Fraud	1.00	1.00	1.00	93834
Fraud	0.88	0.84	0.86	153
Accuracy	-	-	1.00	93987
Macro avg	0.94	0.92	0.93	93987
Weighted avg	1.00	1.00	1.00	93987

Table 2.4.1: Classification metrics for Deep Neural Network on Credit Card Fraud Dataset (Batch size 20, epochs 10)

Table 2.4.2: Balanced Credit Card Fraud Detection Metrics for Deep Neural Network

	Precision	Recall	F1-score	Support
Not Fraud	0.92	0.96	0.94	165
Fraud	0.96	0.91	0.94	160
Accuracy	-	-	0.96	325
Macro avg	0.94	0.94	0.94	325
Weighted avg	0.94	0.94	0.94	325

Table 2.4.2: Classification metrics for Deep Neural Network on Credit Card Fraud Dataset (Batch size 10, epochs 20)

Table 2.4.2: Off Balanced Credit Card Fraud Detection Metrics for Deep Neural Network (10X)

	Precision	Recall	F1-score		Support
Not Fraud	0.99	1.00	0.99		1624
Fraud	0.98	0.95	0.95		162
Accuracy	-	-	0.99		1786
Macro avg	0.99	0.96	0.97		1786
Weighted avg	0.99	0.99	0.99		1786

Table 2.4.2: Classification metrics for Deep Neural Network on Credit Card Fraud Dataset where the number of non-fraud examples is 10x that of the fraud examples (Batch size 10, epochs 20)

References

- Accuracy, Precision, Recall or F1?: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9
- Credit Card Fraud Detection (dataset): https://www.kaggle.com/mlg-ulb/creditcardfraud/tasks?taskId=77

Advanced Anomaly Detection Algorithms

(3.1) Autoencoder Forest

Goals
Research autoencoder forest algorithm and get working on Credit Card Fraud dataset

Background
Autoencoder forest is an unsupervised learning technique that creates a “forest” of autoencoders, trained on subsets of the data. Once the autoencoder reaches a certain accuracy or number of epochs, the model is saved to a list of models, and another autoencoder is trained. You can have as many autoencoders as you like. Once the autoencoders are all trained, prediction of anomalies can be performed by getting the average accuracy on the dataset for all autoencoders. This is used as a baseline. Depending on how sensitive you want the autoencoder to be, you can set a threshold for when the autoencoder considers a particular datapoint to be an anomaly. Each auto encoder gets a vote. If the number of votes passes a certain threshold, the data point is considered to be an anomaly. Autoencoder forest can be ran in real time.

Findings

Results

Table 3.1.1: Autoencoder Forest on Credit Card Fraud Dataset

	Precision	Recall	F1-score	Support
Not Fraud	1	0.99	0.99	284315
Fraud	0.10	0.82	0.18	492
Accuracy	-	-	0.99	284807
Macro avg	0.55	0.90	0.59	284807
Weighted avg	1	0.99	0.99	284807

Table 3.1.1:

(4.1) Iris Dataset

(4.2) Credit Card Fraud Dataset

(4.3) Spotify Dataset