Summer 8-7-2024

Rapid Identification of Antibiotic Resistance in Staphylococcus Using FTIR and Machine Learning

Wilbur Hudson

Rapid Identification of Antibiotic Resistance in Staphylococcus Using FTIR and Machine Learning

by

Wilbur G Hudson IV

Under the Direction of Yi Jiang, PhD

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2024

ABSTRACT

Antimicrobial resistance (AMR) poses a significant threat to global health, with Methicillin-resistant *Staphylococcus aureus (MRSA)* contributing substantially to morbidity. Rapid identification of MRSA versus Methicillin-susceptible *S. aureus (MRSA)* is critical for timely and appropriate antibiotic use. This study explores the use of Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) on Fourier Transform Infrared (FTIR) spectroscopy data to quickly distinguish MRSA from MSSA. By analyzing the growth patterns and spectral data of SA6538 (MSSA) and SA43300 (MRSA) *under antibiotic stress*, we demonstrate the feasibility of separation, highlighting spectral differences and their likely biological causes. LDA, when applied to primary, secondary, and tertiary FTIR datasets, achieves high classification accuracy, particularly when initially processed with PCA. This combined approach suggests a rapid and reliable diagnostic method to improve clinical outcomes and curb the spread of AMR.

INDEX WORDS: Antibiotic, Resistance Staphylococcus aureus, FTIR, PCA, LDA

Rapid Identification of Antibiotic Resistance in Staphylococcus Using FTIR and Machine Learning

by

Wilbur G Hudson IV

DEDICATION

I dedicate this thesis to my friends and chosen family that have supported me over the past 6 months. Their support was unending even when I was not able to give them much of my time and even as *Staphylococcus aureus* infected my casual conversation. Special thanks to A.B. Wilds for his tremendous support.

ACKNOWLEDGEMENTS

The greatest acknowledgments go to Micheal Nelson, Caroline C. Taylor, and Alex Marchesani. Caroline and Alex prepared all the SA samples from December 2022 to June 2024; additionally, their advice and intuition was invaluable to this paper. Micheal Nelson scanned all of the FTIR samples from December 2022 to June 2024. I learned an immense amount from him, and his careful consistency has ensured impressive, high-quality data. This paper could not have been produced without all three of them, and their impressive skill sets.

Next, I have to thank Eric Gilbert and Gary Hastings for their advice. Even though it has not been published, their paper Time-resolved Infrared Difference Spectroscopy for Probing Bacterial Cell Growth was extremely insightful and provided much biological context that I was missing.

Finally, I want to express my appreciation for my committee chair and project advisor Yi Jiang. When I first came to her, I had no experience in research. It is only thanks to her guidance and care that I was able to draft this thesis. Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
AMR	Antimicrobial resistance
ATCC	American Type Culture Collection
ATR-FTIR	Attenuated total reflectance Fourier transform infrared spectroscopy
ddH2O	double distilled water
LB	Luria-Bertani
LDA	Linear Discriminant analysis
MIC	minimum inhibitory concentration
MRSA	*Methicillin-resistant Staphylococcus aureus*
MSSA	Methicillin-susceptible Staphylococcus aureus
OD	Optical Density
PC	principal component
PCA	Principal component analysis
PCR	polymerase chain reaction
RNA	Ribonucleic acid
rpm	rotations per minute
SA	Staphylococcus aureus

1 INTRODUCTION

Antimicrobial resistance (AMR) is widely recognized as a formidable challenge to global health. Global deaths attributable to AMR in 2019 alone rose to 1.27 million; 100,000 of which were directly caused by methicillin-resistant s*taphylococcus aureus*, or MRSA (Murray C, 2022). One of the critical drivers of AMR is the overuse of antibiotics, which accelerates the selection for resistant bacterial strains (Llor 2014). The bacterium *Staphylococcus aureus* ranks second to *Escherichia coli* in the number of AMR-related fatalities. Because of these reasons, MRSA is of particular concern, especially in healthcare environments (Liu F, 2022).

MRSA infections are linked with significantly higher mortality rates and poorer clinical outcomes compared to their Methicillin-susceptible counterparts (MSSA) (Xing, 2022). Studies have demonstrated that intervention within the first 24 hours, is crucial for favorable patient outcomes. However, the conventional antibiotic susceptibility testing methods, which require culturing the organism in the presence of an antibiotic, typically yield results in 48-72 hours (Hassoun 2017). In the last five years polymerase chain reaction (PCR) has become popular due to its high sensitivity and rapidity. This method identifies S. aureus from a single-base-pair mismatch in SA's 16S ribosomal RNA gene sequence. While this test is the golden standard due to its speed and 100% sensitivity, it still takes 24 – 48 hours for results. This delay can lead to the empirical over-prescription of antibiotics by healthcare professionals, aiming to mitigate immediate health risks to patients (Xing, 2022).

The need for faster diagnostic techniques to distinguish MRSA from MSSA is evident. Such advancements could enable more precise antibiotic prescriptions, potentially curbing the spread of resistant bacteria. Additionally, A rapid, reliable testing method would support clinicians in making informed decisions, ensuring that patients receive the most effective treatment as quickly as possible.

1.1 ATR-FTIR

Spectroscopy has been a widely utilized analytical technique that measures the emission and absorption properties of materials across various points of the electromagnetic spectrum. Different materials exhibit unique emission and absorption spectra, enabling their identification based on these spectral characteristics. In 1969 the potential uses of this technology exploded with the first commercial mid-infrared Fourier transform spectrometer with a $2cm^{-1}$ resolution, opening the possibility of biological and biomedical spectroscopy. This non-destructive method is frequently employed to fingerprint, explore, and compare biological structures, offering valuable insights into their composition and function (Faix, 1992). Unlike spectroscopy that uses scanning monochromators Attenuated Total Reflectance Fourier Transform Infrared Spectroscopy (ATR-FTIR) measures all the resolution elements at the same time. This shift to taking multiplexed measurements improves the signal-to-noise ratio of the readings. This simplification also eliminates the needs for slits, allowing for higher overall throughput. After only one cycle of mirror movement the complete infrared spectrum can be reconstructed, however additional scans improve the signal-to-noise ratio as noise is inversely related to the square root of the number of scans (Griffiths, 2007). These advancements make spectroscopy a rapid and available option for routine lab work.

1.2 Analysis

Thanks to these advancements in spectroscopy, differentiating similar spectral fingerprints in real-world diagnostic settings is becoming more realistic. While this work is not the first study to address differentiating resistant and susceptible strains of Staphylococcus aureus. However, previous works often focus on differentiating by genotype and phenotype. Although such identification can be extremely rapid, it has been shown to require strict control of growth medium as well as standardization of temperature and time (Amiali, 2011). Consequently, these tests can be hard to reproduce in real laboratory settings. In this study, we explore differences after bacteria have been exposed to antibiotics over a various times and antibacterial dosages. Differentiation is achieved using a combination of Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). We employ this approach with the aim of paving the way for more flexible, rapid testing methods.

2 DATA COLLECTION AND METHODOLOGY

2.1 Sample Preparation

Primary and secondary datasets were collected by Alex Marchesani and Caroline Taylor in December 2022 and January 2023, respectively. Two strains of *Staphylococcus aureus* were selected. S. aureus ATCC 6538 was selected as the susceptible strain and S. aureus ATCC 43300 was selected for the resistant strain.

Table 1 Primary dataset of MSSA and MRSA at various ampicillin dosages and times.

ATCC 6538 (MSSA)		
Ampicillin Dosage (μg)	Time of Growth Before Sample Collection (min)	Number of Samples
0 μg	0, 30, 60, 120, 240, 360 minutes	7
0.25 μg	0, 30, 60, 120, 240, 360 minutes	7
0.5 μg	0, 30, 60, 120, 240, 360 minutes	7
1.0 μg	0, 30, 60, 120, 240, 360 minutes	7
ATCC 43300 (MRSA)		
Ampicillin Dosage (μg)	Time of Growth Before Sample Collection (min)	Number of Samples
0 μg	0, 30, 60, 120, 240, 360 minutes	7
2.5 μg	0, 30, 60, 120, 240, 360 minutes	7
5.0 μg	0, 30, 60, 120, 240, 360 minutes	7
10.0 μg	0, 30, 60, 120, 240, 360 minutes	7

For the primary dataset cultures of each were inoculated in Luria-Bertani (LB) broth before being incubated aerobically at 37°C over night. The mature cells were then diluted in sterile LB broth to achieve an initial optical density (OD_{600}) of 0.35 with standard error of 0.15. Concentrations of ampicillin were added to flasks in sets of 3 to be averaged later. Differences in levels of antibiotics between MSSA and MRSA samples are due to the significantly higher minimum inhibitory concentration (MIC) of ampicillin interacting with ATCC 43300 (Eucast 2003).

The cultures were incubated aerobically at 37°C and 200 rpm for a total of 6 hours. Samples were collected at 0, 30, 60, 90, 120, 240, and 360 minute, and growth was monitored using OD_{600} measurements. The samples were centrifuged at 13,000 rpm, then the cell pellets were washed with ddH_2O. The supernatants were decanted, and finally the pellets were re-suspended in 10 µL of ddH_2O for spectroscopic analysis.

The secondary dataset followed a similar process, however only samples of S. aureus ATCC 6538 were grown. Additionally, the sample given a dosage of 0.5 µg and grown for 30 minutes was thrown out due to low data quality.

Table 4 Secondary dataset of MSSA and MRSA at various ampicillin dosages

ATCC 6538 (MSSA)		
Ampicillin Dosage (µg)	Time of Growth Before Sample Collection (min)	Number of Samples
0 µg	0, 30, 60, 120, 240, 360 minutes	7
0.25 µg	0, 30, 60, 120, 240, 360 minutes	7
0.5 µg	0, 60, 120, 240, 360 minutes	6
1.0 µg	0, 30, 60, 120, 240, 360 minutes	7

The tertiary dataset was collected in June 2024 by Alex Marchesani and Caroline Taylor. The same two strains as the primary dataset were used and a largely similar process followed. The most notable difference in this dataset was the focus on the early time periods. Samples were taken with 10-minute resolution for up to 1 hour. Additionally, samples were diluted in sterile LB broth to achieve an initial optical density (OD_{600}) of to 0.2. As the goal of this data set was for differentiating MSSA and MRSA, fewer antibiotic dosages were used. Additionally, the dosage categories were selected for more direct comparison between MSSA and MRSA. These samples were *not* grown and scanned in triplicate due to time restraints. Therefore they are potentially of lower quality than the original dataset. This important preliminary dataset serves two purposes, 1) validating, the constructed algorithm, and 2 for the proof of concepts for the earliest possible time to differentiate MRSA and MSSA.

Table 7 Tertiary dataset of MSSA and MRSA at various ampicillin dosages

ATCC 6538 (MSSA)		
Ampicillin Dosage (μg)	Time of Growth Before Sample Collection (min)	Number of Samples
0 μg	0, 10, 20, 30, 40, 50, 60 minutes	7
0.5 μg	0, 10, 20, 30, 40, 50, 60 minutes	7
ATCC 43300 (MRSA)		
Ampicillin Dosage (μg)	Time of Growth Before Sample Collection (min)	Number of Samples
0 μg	0, 10, 20, 30, 40, 50, 60 minutes	7
0.5 μg	0, 10, 20, 30, 40, 50, 60 minutes	7
5.0 μg	0, 10, 20, 30, 40, 50, 60 minutes	7

2.2 FTIR measurements

The primary and secondary datasets were collected by Micheal Nelson in December 2022 and January 2023, respectively. The tertiary data set was collected in June 2024, led by Michael Nelson and performed by Michael Nelson and Wilbur Hudson. Before each measurement, the sampling window (diamond ATR crystal) was cleaned with methanol and lens paper to remove all traces of previous sample. A new $4cm^{-1}$ background spectrum was collected before depositing the next set of cells for measurement. After background collection, 2µL of pelleted cells were pipetted from a microcentrifuge tube and deposited on to the diamond window. Care was taken to ensure no bubbles formed on the surface as a result of pipetting. This is done so that the layers of cells dry as uniformly as possible. Since the absorption bands of liquid water overlap with the bands of interest in this study, it is necessary to carefully monitor and control the levels of hydration present in the samples. This is done spectroscopically. As the samples dry on the ATR crystal, the absorbance spectrum stabilizes. See figure 22 in appendix A. In the infrared, this loss of water can be seen as a gradual suppression of the water bands and increasing intensity of the amides and other prominent features. As the deviation between readings falls below a certain level the sample is ready. 8 scans were used per measurement. For tertiary samples this was after 60 scans or 15 minutes of drying. This process was followed for each sample to reduce noise and improve comparability between samples.

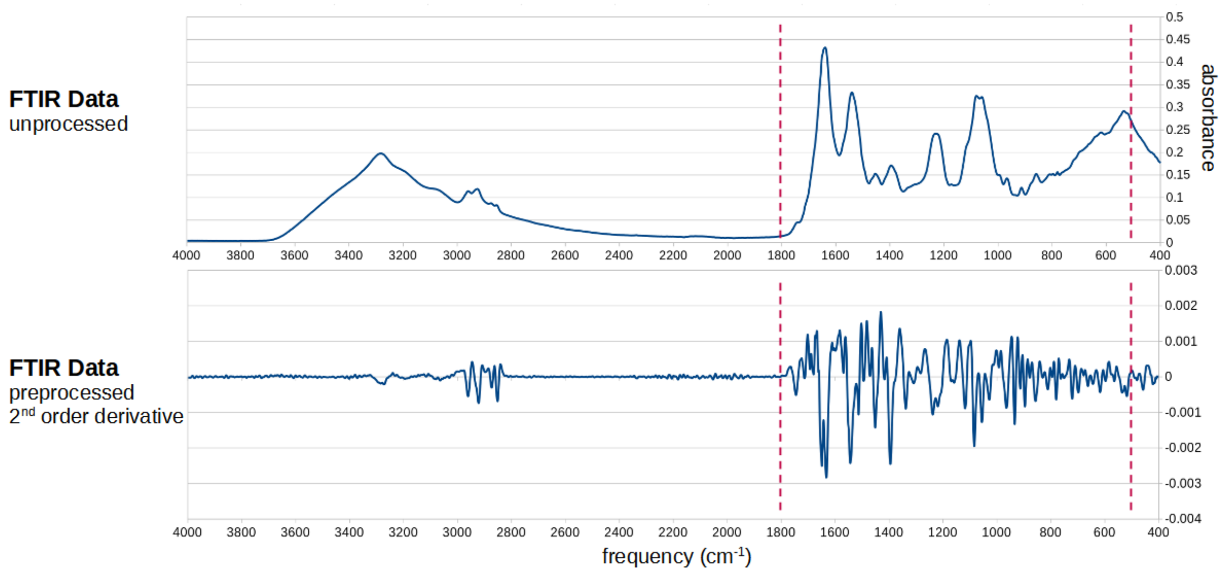


Figure 1 Example of unprocessed spectra with red dashed lines indicating selected cut off points on top with 2nd order derivative spectra with cut off points on bottom.

2.3 Pre-processing through OPUS

The absorbance curves were read through Bruker OPUS software. Primary absorbance curves were min-max normalized from wavenumber 1800 cm^{-1} to 1000 cm^{-1}. This is done to reduce variation caused by inconsistencies in sampling.

Often direct absorbance can obscure overlapping peaks. A well-known solution for this is to apply a smoothing filter such as Savitzky-Golay and using the smoothed data to create a 2nd derivative curve. The 2nd derivative curve can tease apart the peaks allowing for better differentiation between similar absorbance curves. In this study we will focus on this preprocessed data. We will also focus on the mid-infrared frequency region, i.e. the "fingerprinting region," from 500 cm^{-1} to 1800 cm^{-1} as this region yields critical information on key protein structures. We use the OPUS software to first smooth the curves using the Savitzky-Golay filter with a window size of 9, then generate the 2nd derivative curve.

2.4 Machine Learning Analysis

All machine learning analyses was performed using Python. Before any analysis, all 634 features were standardized to ensure each feature contributed equally. This was done using the StandardScaler function from the scikit library, which applies the standardization formula $X_{\text{standardized}} = (X-\mu)/\sigma$, where X represents the original frequency values, μ denotes the mean, and σ stands for the standard deviation.

Principal Component Analysis (PCA) was employed on the 56 primary, 27 secondary, and 35 tertiary samples. Additional preprocessing, including smoothing and calculating the second derivative, was required for the secondary and tertiary datasets. This was achieved using the Savitzky-Golay filter with a window size of 9 and a polynomial degree of 2, matching the preprocessing of the primary dataset previously processed in Bruker's OPUS software.

PCA was implemented for 10 principal components, and the scree plot was analyzed. As the variance per principal component sharply decreases past the third principal component, a three-dimensional PCA analysis was selected. The analysis was performed using the PCA functions from the scikit library. Due to the large number of parameters and relatively small number of features, the full SVD method was selected. To calculate loadings at each frequency the function pca.components_.T was used. As loadings are simply the elements of the eigenvectors calculated by the earlier PCA function, pca.components_.T simply returns these values in order. Each value was assigned back to the frequencies and plotted as a line plot on top of a bar chart with each principal component assigned to a different color.

When applying the Linear Discriminant Algorithm (LDA) to the primary dataset, training and testing subsets were selected. The primary dataset was grouped based on the minimum inhibitory concentration and time. Each group was removed from the training data before being projected and classified to assess the algorithm. For the secondary and tertiary datasets, this step was not required. Instead, the complete primary dataset was used for training before projecting the secondary and tertiary datasets. For PCA-fed LDA, PCA was applied first to reduce the dimensionality of the data, followed by the LDA algorithm as described.

All processing was done using OPUS, LibreOffice Calc, and Python. Data processing libraries included numpy, pandas, scipy.signal, and scikit-learn. Data visualization libraries included matplotlib.pyplot, mpl_toolkits.mplot3d, and seaborn. Scipy.signal was used for the Savitzky-Golay filter, smoothing, and calculating the second derivative of the secondary and tertiary data. Notably, scikit-learn was used for standardizing, PCA, and LDA. All data was standardized before processing.

3 ANALYSIS AND RESULTS

3.1 Analysis of Primary data

The normal growth curve of bacteria can be determined by inoculating a small number of bacterial cells into a suitable culture medium and counting the bacteria in aliquot samples at regular intervals. When the number of the viable cells are plotted against time, it gives a typical curve called as bacterial growth curve or growth cycle of bacteria. The resulting curve has four distinct phases (Figure 2).

As bacterial cells encounter a new environment, growth is not the first concern. Instead, cells spend their energy on adapting. This initial phase of slow growth is commonly referred to as the lag phase. Once a culture has adapted, it enters its exponential growth (log) phase before environmental limitations force the bacteria into their last phases: stationary and eventually logarithmic decline (Fankhauser D, 2016). The length of time over which these phases occur differs from strain to strain and environment to environment. It is a well-known result that antibiotic perturbation can shorten or lengthen lag and log time for cells with differing antibiotic resistance (Bertrand R, 2019).

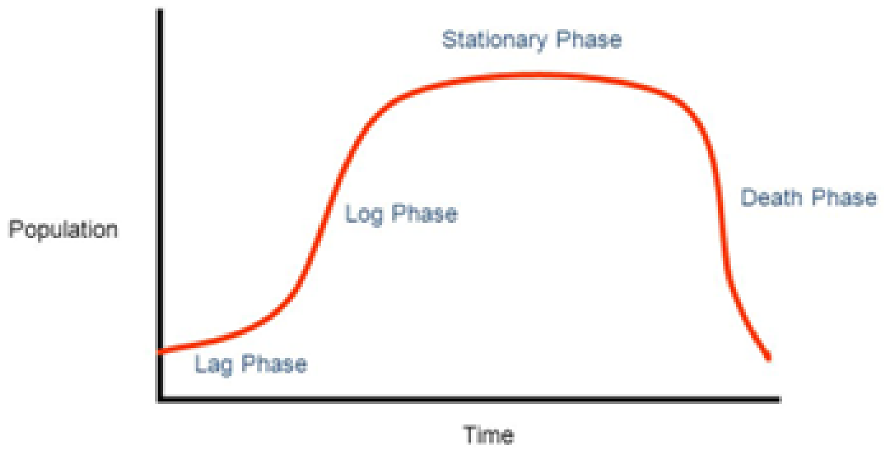


Figure 2 Four distinct phases of bacterial growth (Yourassowsky E, 2021)

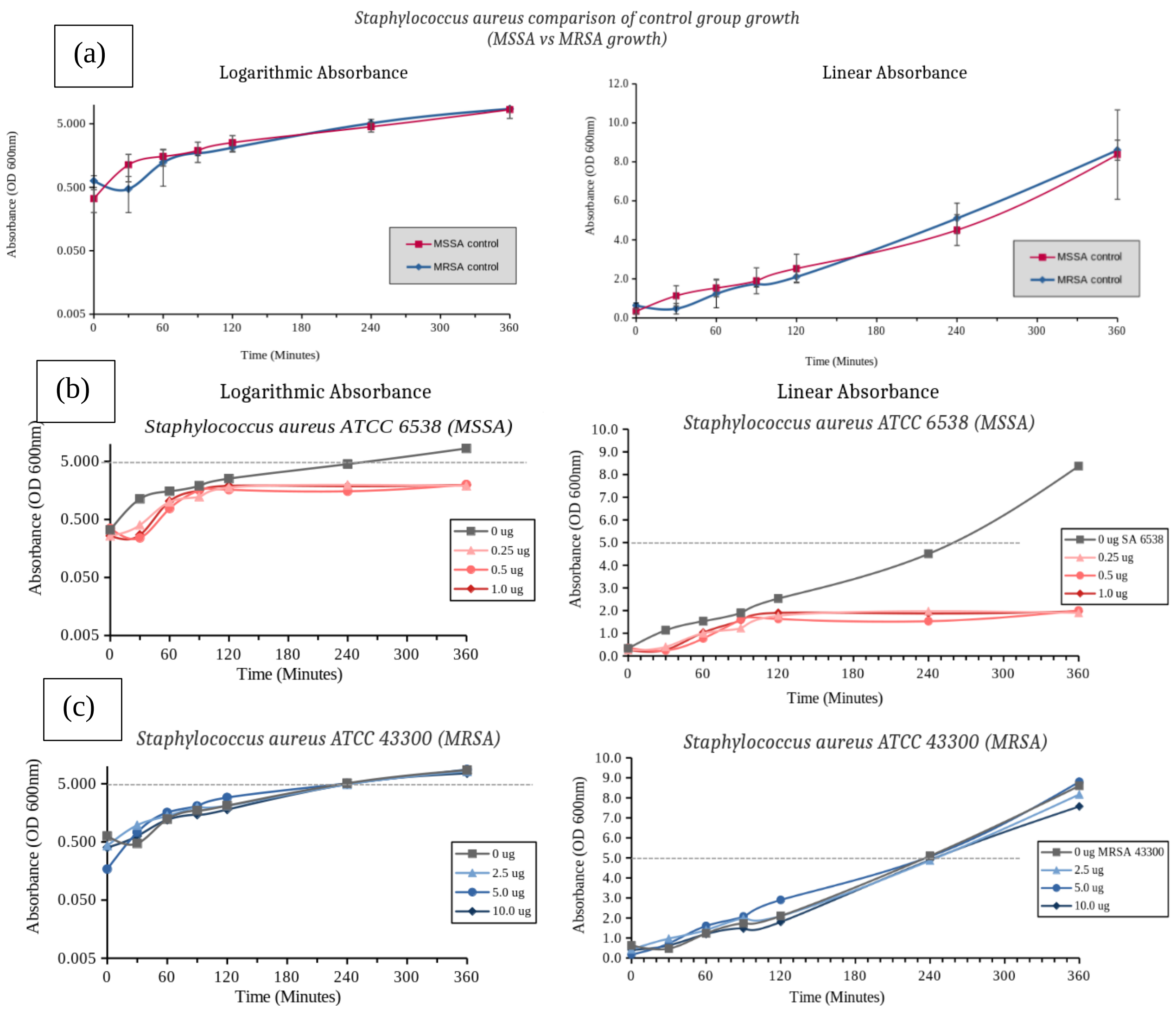


Figure 3 Growth curves for primary data split into logarithmic (left) and linear (right) absorbance (a) shows the growth curves measured by optical density at 600 nm with error for ATCC 6538 and ATCC 43300 with no antibiotic treatment. (b & c) Shows growth of the control groups with samples of differing antibiotic treatment for both MSSA and MRSA. Gray plots show the control group while red and blue show growth of treated samples. Darker colors are used for increasing antibacterial dosage.

Figure 3(a) shows the growth curves of SA6538 (MSSA) and SA43300 (MRSA). While there is a clear lag phase during the first 30 minutes, the growth does not seem to stabilize into its log phase until the 90 to 120-minute mark. This is more clear on the linear absorbance plots to the right. The 120-minute mark would be consistent with previous works but may be altered due to experimental conditions (Kochan, 2020). Additionally, we see that MSSA and MRSA show similar growth under no antibiotic stress. As expected, under antibiotic stress, MRSA remains largely unchanged (figure 3(c)) while MSSA enters its log and stationary phase significantly faster (figure 3(b)). In addition, all dosed samples of MSSA enter the stationary phase between 60 to 90 minutes. As shown in previous works, these phase differences are detectable by FTIR and will likely play the biggest role in forming a rapid method of differentiating between resistant and nonresistant strains of staphylococcus aureus (Hastings G, 2024).

3.1.1 PCA and the separation of MSSA and MRSA

Each FTIR spectrum has 634 frequencies, but only a handful of these play a major role in differentiating between the MSSA and MRSA. PCA creates a linear combination of frequencies that captures the largest variation in our dataset. More so, we can compute the correlations between the original frequencies and each principal component. This allows us to trace our variance back to specific portions of our spectrum (Smith, 2002). In doing so a connection between variation can be connected to its biological cause. We will demonstrate this possibility without going into the detailed spectral interpretation, as the latter falls outside the scope of this thesis.

The primary set of Savitzky-Golay filtered second derivative data with 634 frequencies was standardized and used to calculate a covariance matrix. The eigenvectors associated with the greatest eigenvalues are selected as our principal components in descending order. The number of eigenvalues selected is the dimensionality of your reduced space. Finally, our dataset was projected onto this new orthogonal basis, allowing us to visualize our data in a reduced dimensional space. Importantly, this also helps filter out potential noise that does not correlate well with the variability of our data points.

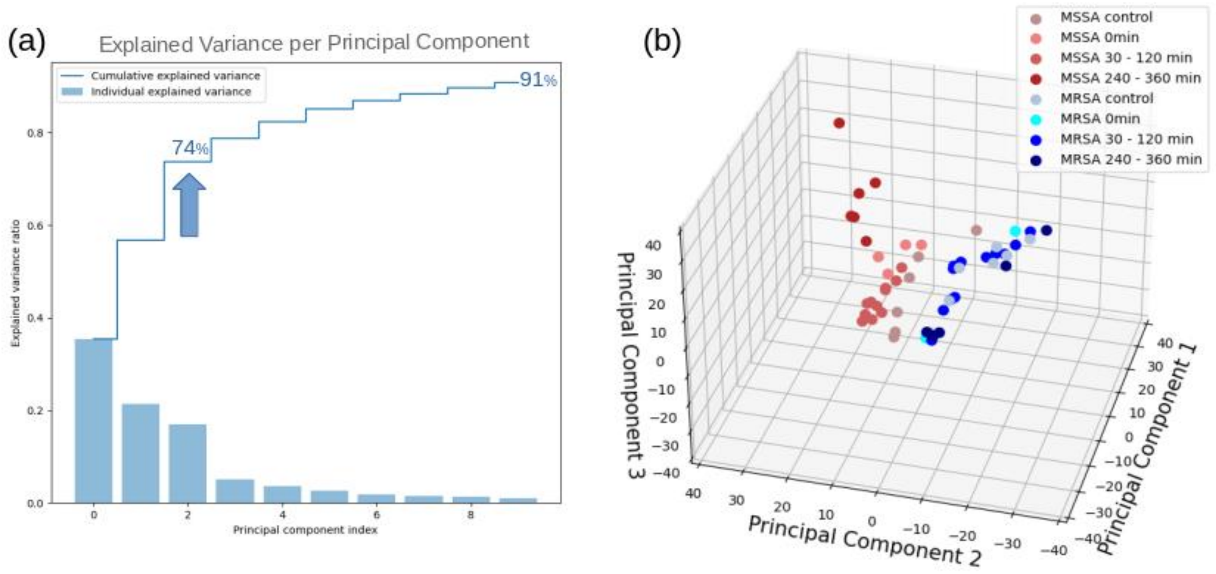


Figure 4 Scree plot and 3 dimensional PCA of primary data (a)Explained variance graph shows in bars how much variance is contained in each principal component (PC). The step chart displays cumulative variance. As shown, the first 3 PC contain 74% of total variance of the set. This continues up to PC 10 which cumulatively contains 91% total variance. (b) Each data-point (28 MRSA and 28 MSSA) is projected onto the first 3 principal components. Red and blue points refer to MSSA and MRSA respectively. The data points are grouped by growth time in darkening colors.

A scree plot or explained variance in PCA represents the proportion of total variance attributed by each principal component, essentially showing how each principal component differentiates each of our samples. As shown in figure 4(a), the first three principal components contain 74% of the total variance of the primary data set. After that there is a sharp reduction in individual explained variance. When the 56 primary data points are plotted along these first three principal components we can already visually identify clustering of MRSA versus MSSA. This separation primarily occurs along PC2, with MRSA having negative

PC2 values and MSSA having positive PC2 values. Only two points fail to separate along this line: MSSA 0 µg at 240 minutes and MSSA 0 µg at 360-minutes. This follows the expectation that untreated MSSA will grow similarly to treated and untreated MRSA. Interestingly, the rest of the MSSA control group grown for shorter periods are close to the MRSA cluster, but they remain separable.

While the MRSA data points cluster well overall, they do not seem to show any obvious *internal* separation by growth time. The plotted MSSA data, on the other hand does not cluster as tightly, but similar data points show clear internal clustering. First, treated (dosage > 0) MSSA of times 0 minutes to 120 minutes measure an average PC2 value of 9 with a standard deviation of 1.8. The final two time points of 240 and 360 minutes scatter further along PC2 with an average of 19 and standard deviation of 5.7. Interestingly, regardless of dosage, we see a similar story along PC3. Early data points from time 0-120 minutes have a low PC3 with an average of -4 with standard deviation of 4.5 while later time points at 240-360 show a clear jump in PC3 up to an average of 15.9 with standard deviation of 8.5. This effect is more pronounced for treated samples.

.

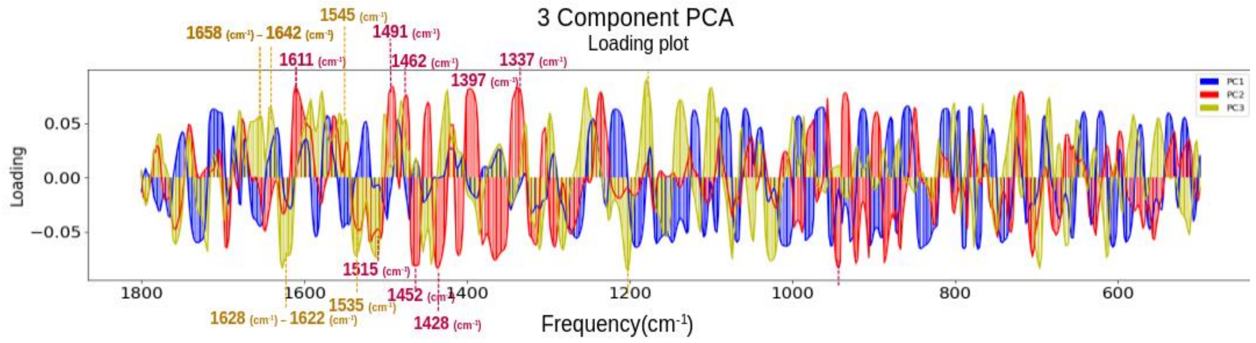


Figure 5 PCA Loading plot for each frequency. Spike magnitude tells us which frequencies impact (positively or negatively) each principal component. With each principal component separated into colors. Blue, red and yellow depict PC's 1, 2, and 3 respectively. Specific peaks are highlighted either due to their magnitude or known influence.

To understand the spectral correspondence of the principal components (eigenvectors), we performed PCA loadings analysis. To do this we simply visualize the elements of each eigen vector associated with each frequency. This allows us to begin to connect the separability of MSSA and MRSA to individual frequencies. Figure 5 shows these loadings for the first three PCs, giving us the spectral representation of each PC.

Frequencies 1200cm^{-1} – 500cm^{-1} are noticeably dominated by PC1. However, as shown in figure 4(b), PC1 does not meaningfully cluster the data in a way that matters for MSSA and MRSA groupings. Additionally, it is known that the ALPHA II Compact FT-IR Spectrometer by Bruker produces more noise on lower frequencies. Therefore the Amide I, II, and III bands at frequencies ranging from the 1800cm^{-1} – 1200cm^{-1} play the most vital role in differentiating the samples based on figures 4 and 5. This is important for both looking for algorithmic differentiation as well as identifying material biological differences in how MSSA and MRSA react under antibiotic conditions. This lines up well with previous works such as Kochan et al. 2020.

Previous works suggest that within the Amide I region some bands of importance include 1658 cm^{-1} and 1642 cm^{-1} that are associated with the alpha-helix secondary structure while features at 1628 cm^{-1} and 1622 cm^{-1} are associated with the protein beta-sheet secondary structures (Kochan, 2020). They appear as major peaks, associated positively and negatively respectively, for PC3. In the Amide II region a positive peak at 1545 cm^{-1} and negative peak at 1535 cm^{-1} has been associated with changes in the alpha-helix subunits in the lag phase and beta-sheet subunits in the log phase in previous works (Hastings G., 2024) (Table A.2). Figure 5 shows the first set of positive and negative peaks appear solely for PC3 while the second set appears for both PC2 and PC3. This might explain the shift in PC3 that occurs between the lag and log phases, regardless of dosage. It also suggests that the first peaks may be less useful in differentiating between MRSA and MSSA while the second set of peaks do in fact provide this separation. Additionally features of PC2 are observed at 1515, 1462, 1428, 1397, and 1337 cm^{-1} which are all associated with alpha-helical protein segments. They may suggest, a connection between these alpha helical protein segments and the differentiation of MSSA and MRSA.

Some further peaks of interest occur at 1452 cm^{-1} that is associated with proteins, lipids and polysaccharides as well as 1491 cm^{-1} and 1611 cm^{-1}. However, more work is needed to connect these to their molecular origins and biological functions.

3.1.2 Linear Discriminate Analysis

Linear discriminant analysis (LDA) finds the linear combinations of features that best separate two or more classes. In our case only two classes are needed (MSSA and MRSA), LDA will project our data to a one-dimensional space with a linear boundary. Linear boundaries are already clearly visible from the three dimensional PCA performed in 3.1.1. We expect that PCA-filtered LDA will provide a stable method of differentiating our dataset.

We first applied LDA directly to the primary dataset. The data was split into testing and training sets, based on both MIC dosage and growth time. We trained the LDA model with 42 samples and tested with the remaining 14. This was repeated with each partition.

Figure 6 shows the one-dimensional projection from testing the 0.25 μg MSSA and 2.5 μg MRSA samples. In the associated confusion matrix to the right we see that the 60 minute and 240-minute MRSA samples have been predicted incorrectly as MSSA. The rest have all been predicted correctly.

Figure 7 shows the confusion matrices for all dosage-based testing training categories, which indicates that, the model struggles to make predictions on low dosage samples. This can explain our missed predictions in figures 6 and 7.

When we split the data into testing and training sets by growth time, the model is then trained off of 48 samples and 8 are used for testing. By itself this change will likely improve the efficacy of the algorithm. The projection in figure 8 shows the model assessing the 8 samples with 30 minutes of growth time. This is of particular interest as we would like the algorithm to make accurate predictions at as low of a time point as possible. In figure 8, we see our 30-minute points have been predicted with 100% accuracy.

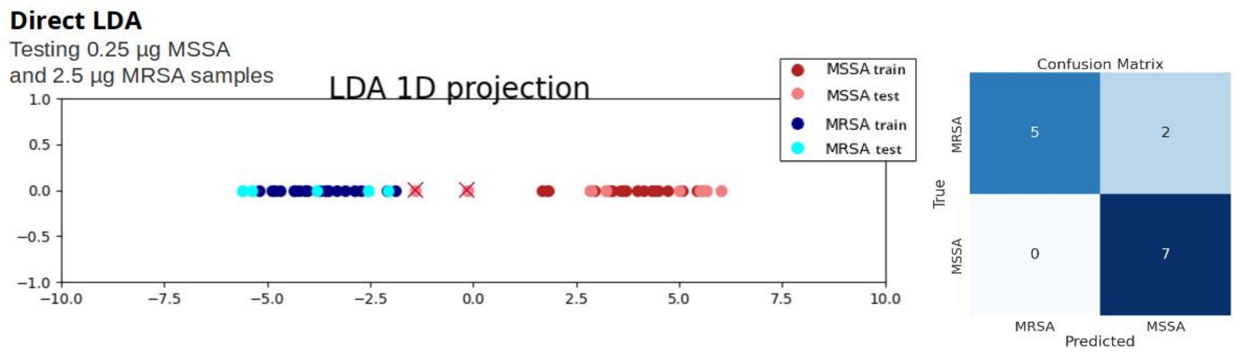


Figure 6 discriminates MRSA from MSSA ***by dosage.*** *To the left: an example projection 0.25µg MSSA and 2.5 MRSA samples taken out of the training data and used for testing. Dark red and blue points refer to the training MSSA and MRSA data while light red and blue points refer to the testing data projected and predicted. Red X's mark incorrectly predicted data points. The associated confusion matrix for the example is shown to the right. Values along the primary axis show correctly predicted samples while the off axis shows inaccurately predicted data. This shows both how the samples should be categorized and what the algorithm predicted.*

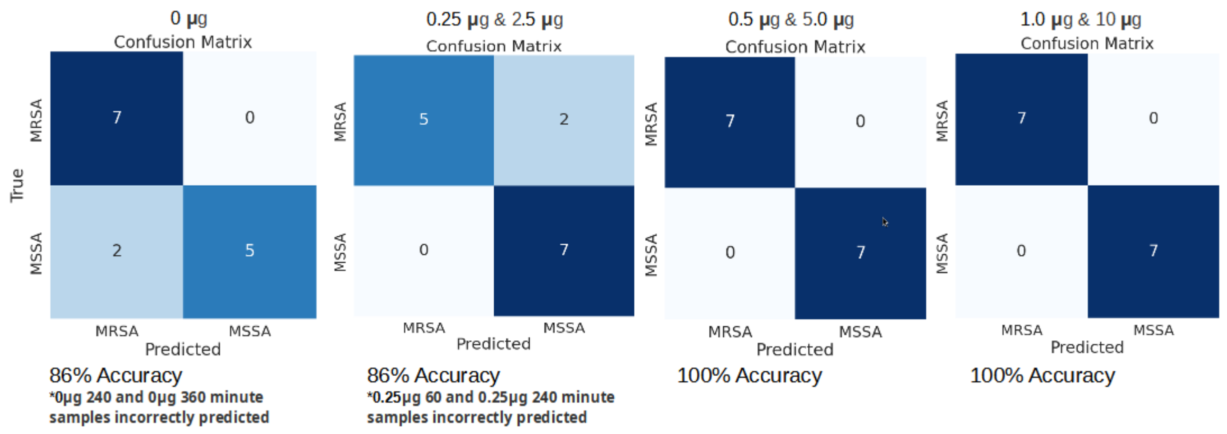


Figure 7 confusion matrix and algorithm accuracy for each dosage-based partition. Values along the primary axis show correctly predicted samples while the off axis shows inaccurately predicted data. This shows both how the samples should be categorized and what the algorithm predicted. Find inaccurately predicted samples below accuracy.

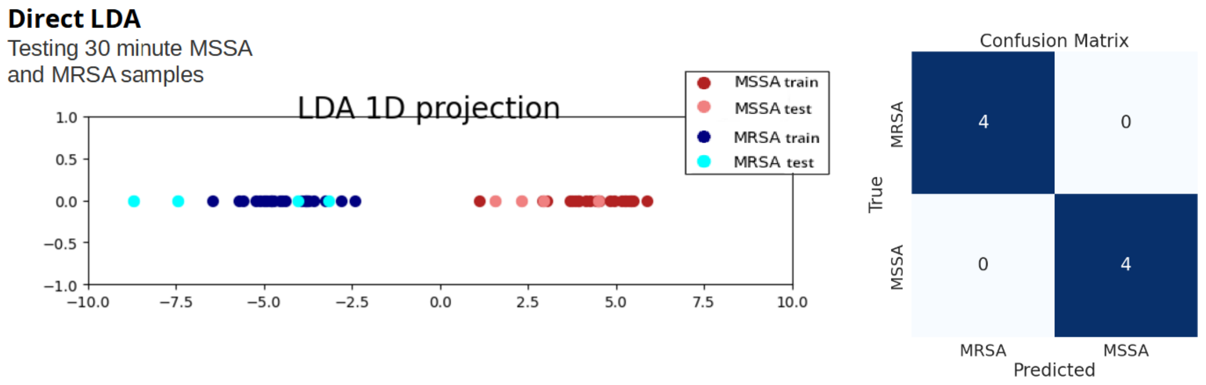


Figure 8 LDA discriminates MRSA from MSSA by dosage ***by growth time.*** *To the left: an example projection 0.25µg MSSA and 2.5 MRSA samples taken out of the training data and used for testing. Dark red and blue points refer to the training MSSA and MRSA data while light red and blue points refer to the testing data projected and predicted. Red X's mark incorrectly predicted data points. The associated confusion matrix for the example is shown to the right. Values along the primary axis show correctly predicted samples while the off axis shows inaccurately predicted data. This shows both how the samples should be categorized and what the algorithm predicted.*

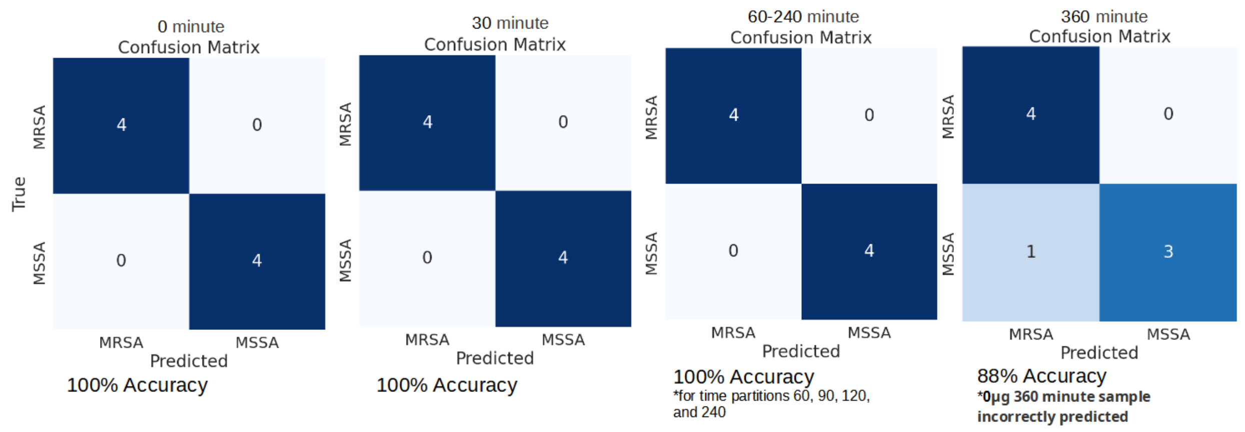


Figure 9 confusion matrix and algorithm accuracy for each growth time-based partition. Values along the primary axis show correctly predicted samples while the off axis shows inaccurately predicted data. This shows both how the samples should be categorized and what the algorithm predicted. Find inaccurately predicted samples below accuracy. Additionally note that all time-based partitions from 60, 90, 120, and 240 minutes all had 100% accuracy.

Figure 9 shows the confusion matrices for all time-based testing/training categories. With more points used for training we *do* see better overall accuracy. One point is inaccurately predicted. The 0 μg 360-minute MSSA is confused with MRSA. While these preliminary results are promising, the clear clustering of our earlier principal component analysis suggests that this could still be improved.

3.1.3 LDA with PCA

A natural question arises: can the potential noise reduction of PCA improve the linear discriminant analysis algorithm's ability to predict MSSA versus MRSA even for lower-dosage data points? Of course we are not limited to 3 principal components. From the earlier explained variance plot, 10 principal components contain 91% of the data variability. These first 10 principal components were calculated, then this 10-dimensional data set was split and used to train and test the LDA model. Next this was repeated for both the dosage and time training-testing sets.

For the dosage sets, 0.25 and 2.5 μg MSSA and MRSA samples are predicted at 100% accuracy. The control group is also improved but again the 0 μg 360-minute MSSA sample is mistaken for MRSA. For the time-based sets, accuracy of all tests improves to 100%.

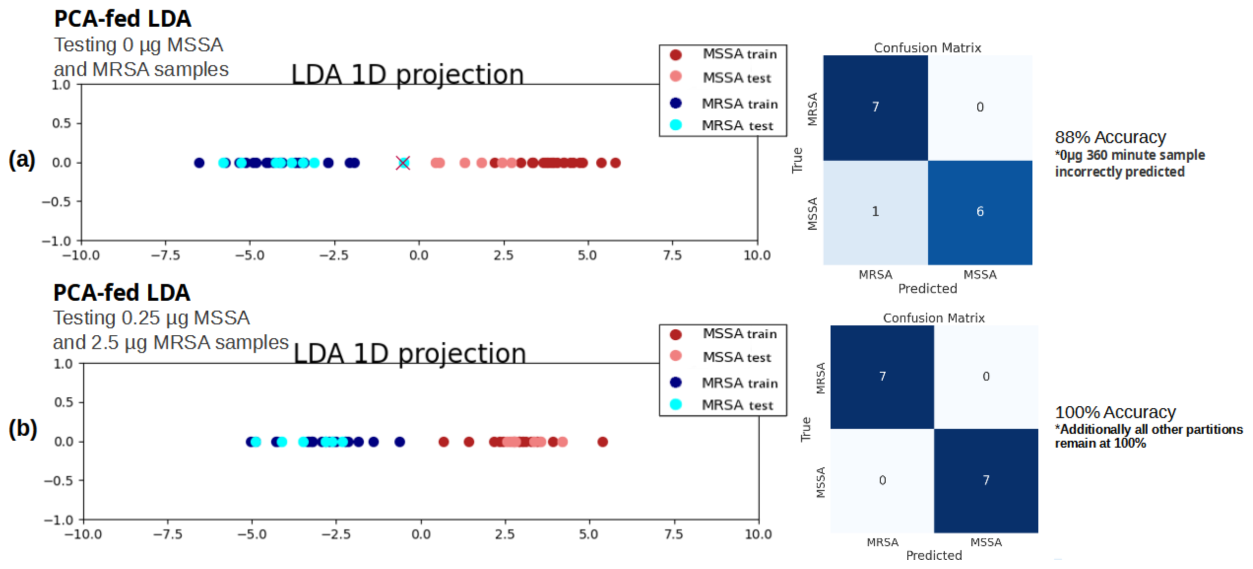


*Figure 10 LDA with PCA using 10 PC's Trial testing and training **partitioned by dosage** (a) Shows LDA projection of 0 µg testing samples after 10 component filter and associated confusion matrix. Dark red and blue points refer to the training MSSA and MRSA data while light red and blue points refer to the testing data projected and predicted. Red X's mark incorrectly predicted data points. The associated confusion matrix shows correctly predicted samples on the primary diagonal while the off axis shows inaccurately predicted data. (b) PCA fed LDA projection of 0.25ug MSSA and 2.5ug MRSA samples after 10 component filter and associated confusion matrix. Note the improvement in accuracy in comparison to the directly fed LDA algorithm.*

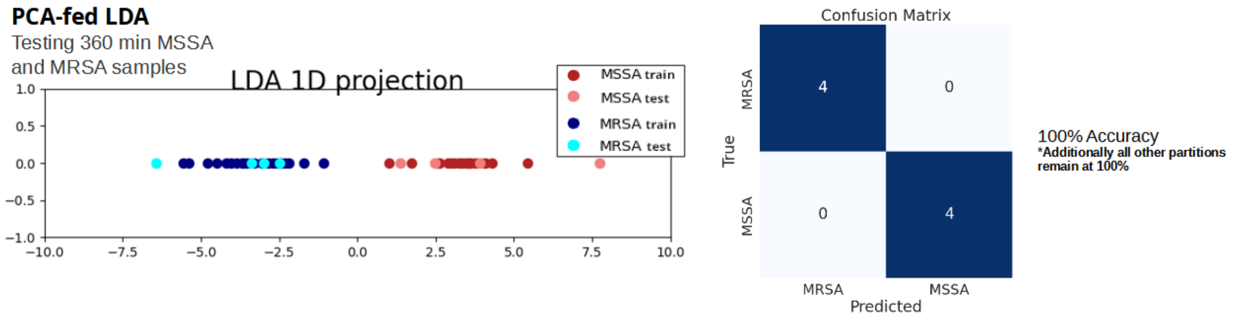


*Figure 11 PCA filtered with 10 principal components then fed into LDA. Trial testing and training **partitioned by time.** The figure shows LDA projection of 360-minute testing samples after 10 component filter and associated confusion matrix. Dark red and blue points refer to the training MSSA and MRSA data while light red and blue points refer to the testing data projected and predicted. Red X's mark incorrectly predicted data points. The associated confusion matrix shows correctly predicted samples on the primary diagonal while the off axis shows inaccurately predicted data. Note the improvement from 88% to 100%*

3.2 Analysis with Secondary data

While this looks promising, due to the large initial dimensionality of the dataset and the relatively small number of training data-points, there is a danger of over-fitting in this LDA algorithm. The most direct way to check this is to test our model on new data to see if our model works as expected. This was done using the secondary dataset. 27 existing (triplicate averaged) absorbance spectra of MSSA dosed at the same levels as the primary dataset was taken in January 2023. This data was prepossessed using the Savitzky-Golay filter with a window size of 9 and polynomial degree 2. The 2nd derivative was calculated after this to match the pre-processing of the primary data set. See the initial example absorbance curves and the preprocessed 2nd derivative curves in figure 12.

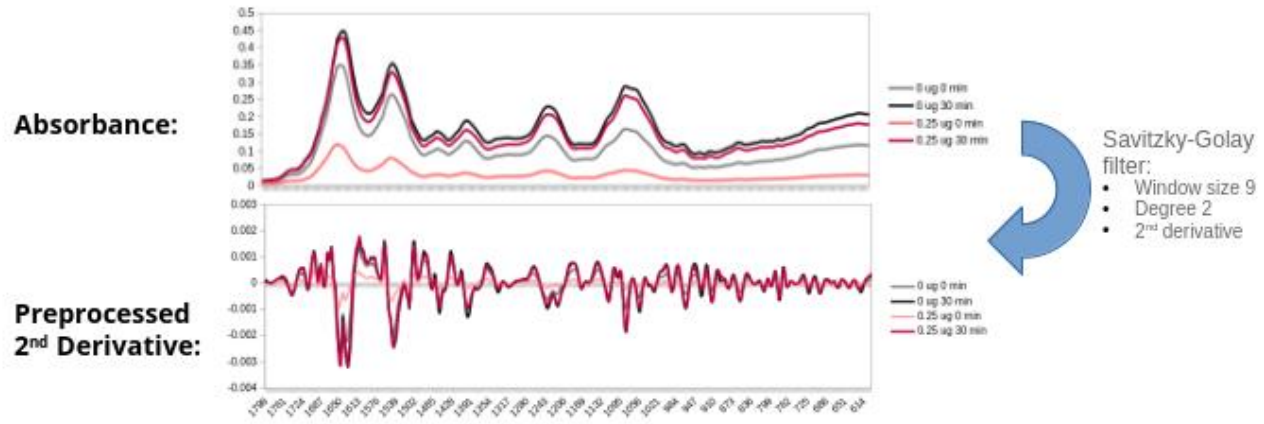


Figure 12 Example of preprocessing used to match the secondary dataset with the primary dataset before it could be used in LDA. Plots show examples form the control group and low dosed groups.

This data was selected due to availability. Only testing MSSA data is not ideal, but it still provides a useful test for the algorithm trained on the initial primary data set.

When the algorithm is applied directly to our 27 new samples of MSSA, we get the projection and associated confusion matrix seen in figures 13 and 14. This shows a 67% accuracy but many of the improperly predicted points were either treated with 0 µg or were measured after 0 minutes. Once these control points are taken out, we get an accuracy of 88%. The 1 µg 360-minute point and the 1 µg 30-minute points are still predicted incorrectly. The poor performance of the control group is a good sign that the algorithm is detecting and differentiating based on real biological differences due to the MSSA's reaction to ampicillin over time as opposed to MRSA. In this way the predictability of the algorithm, is supporting evidence

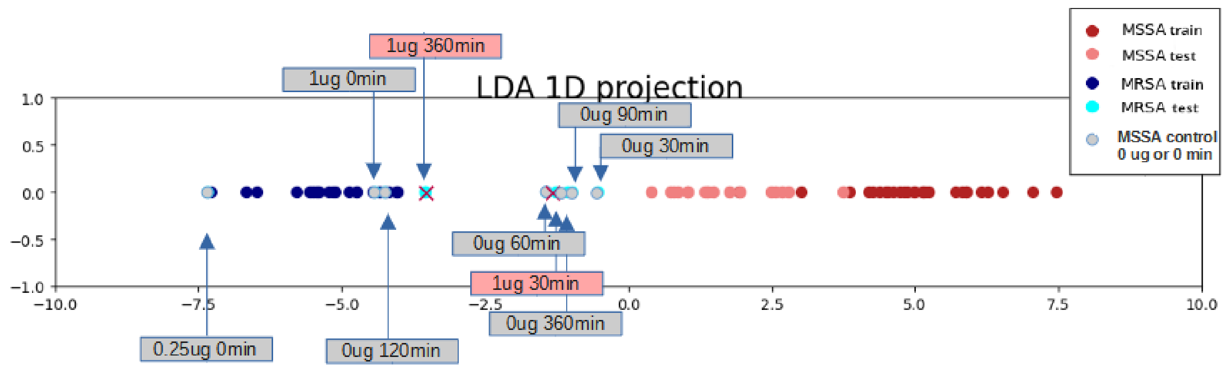
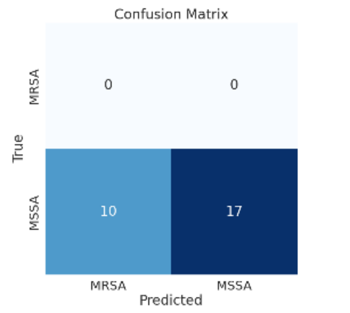


Figure 14 the LDA projection of the secondary dataset (only MSSA) trained directly on all 56 samples of the primary dataset. Dark red and blue points refer to the training MSSA and MRSA data while light red and blue points refer to the testing data projected and predicted. Data that falls into the control group, either by 0ug dosage or 0 minutes are marked in gray. Red X's mark incorrectly predicted data points.



63% Accuracy
***Note that when the control group is removed accuracy increases to 88%**

Figure 13 The associated confusion matrix shows correctly predicted samples on the primary diagonal while the off axis shows inaccurately predicted data.

that the model is not simply over-fitted. Even so, perhaps filtering out noise through PCA can improve our model once again.

3.2.1 LDA with PCA on the secondary dataset

As we saw previously, dimensional reduction through PCA can improve the algorithm's accuracy. Starting with the 3 principal components in figure 15 that were visualized earlier, we get an improved accuracy of 96% figure 16(a). One point from our secondary dataset was mistaken. Unsurprisingly this point was our familiar 0 μg 360-minute MSSA.

Increasing the number of principal components improves up to PC 15 in figure 16(c). After this we see the model degrade back towards the performance of direct LDA.

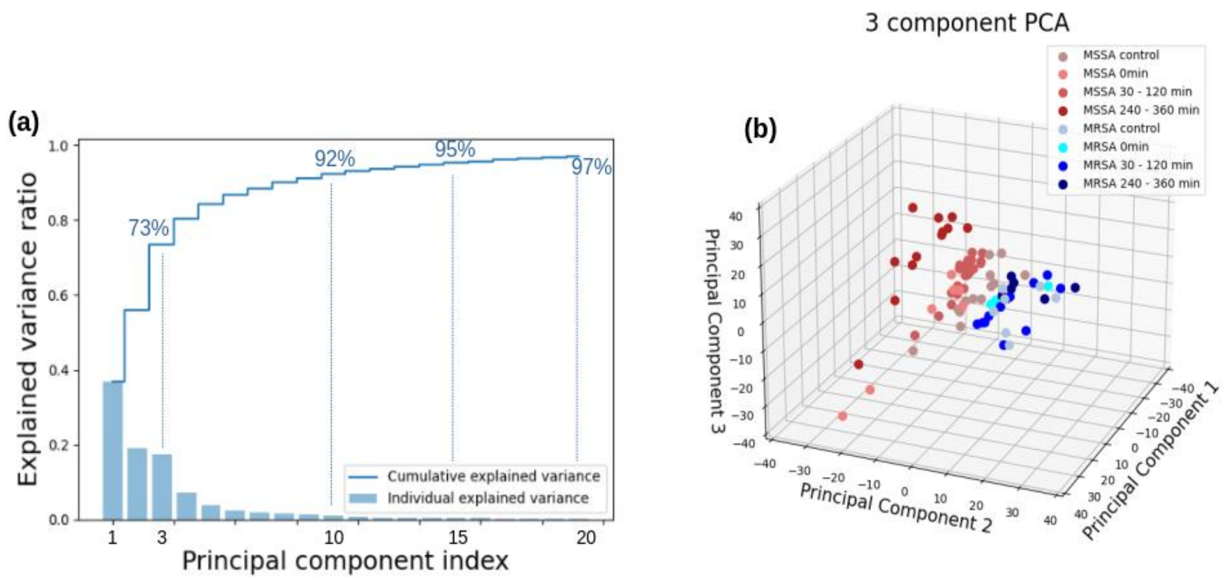


Figure 15 Scree plot and 3 dimensional PCA of primary data (a)Explained variance graph shows in bars how much variance is contained in each principal component (PC). The step chart displays cumulative variance. As shown, the first 3 PC contain 73% of total variance of the set. This continues up to PC 20 which cumulatively contains 97% total variance. (b) Each data-point of the primary and secondary datasets (28 MRSA and 55 MSSA) is projected onto the first 3 principal components. Red and blue points refer to MSSA and MRSA respectively. The data points are grouped by growth time in darkening colors.

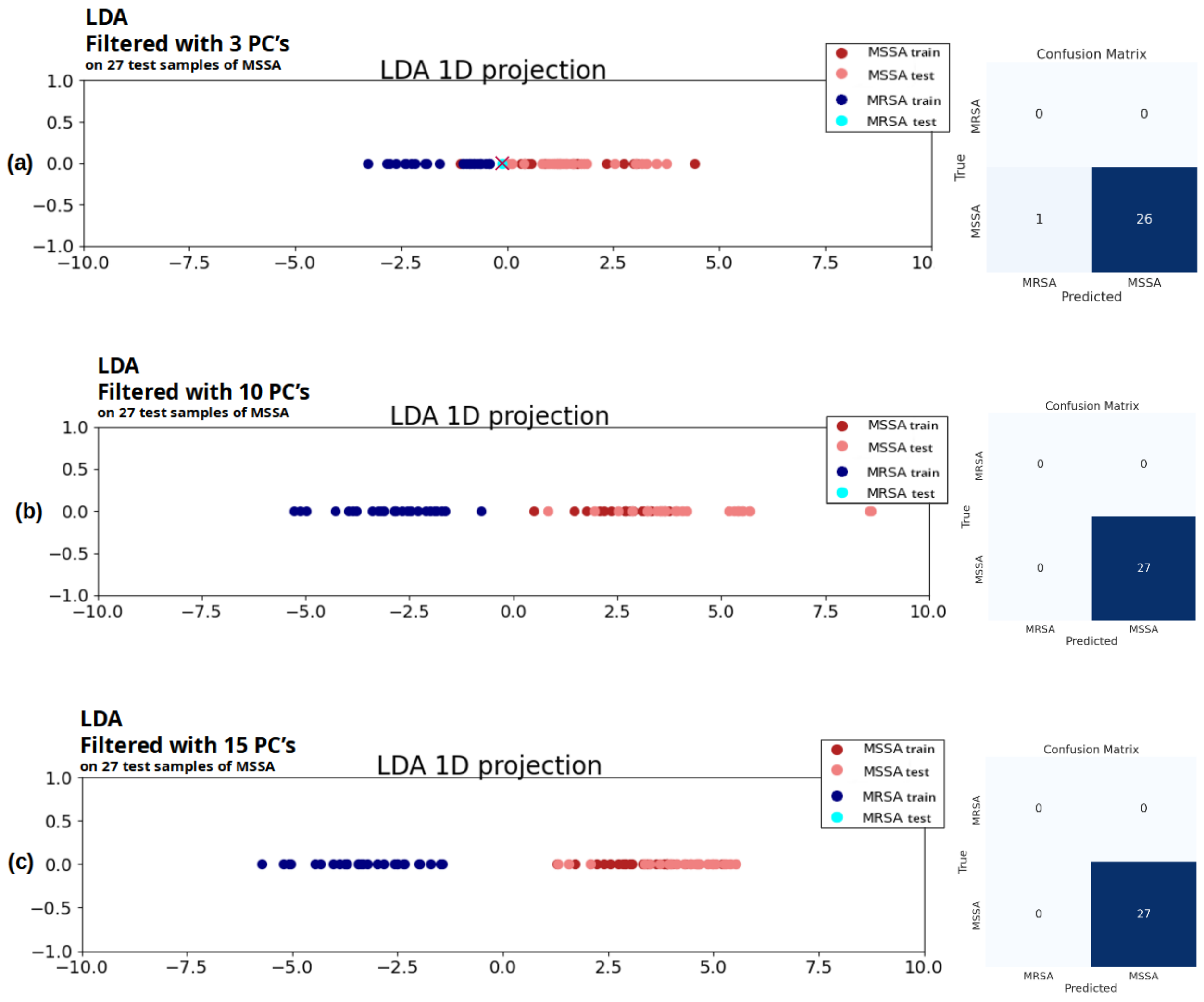


Figure 16. LDA projection of the secondary data. Dark red and blue points refer to the training MSSA and MRSA data while light red and blue points refer to the testing data projected and predicted. Red X's mark incorrectly predicted data points. (a) when filtered through the first 3 PC's for 96% accuracy. (b) when filtered through the first 10 PC's for 100% accuracy. (c) when filtered through 20 PC's. Here we see degradation of the model giving 78% accuracy.

3.3 Analysis with tertiary data

A tertiary set of data was collected both for MSSA and MRSA with a 10-minute resolution up to 60 minutes, the goals of which are testing how separable this data remains at shorter time scales, as well as exploring potential different causes of separation specifically during the early lag period. Unlike previous data, this data was not averaged in triplicate which may cause some difficult-to-predict data-points.

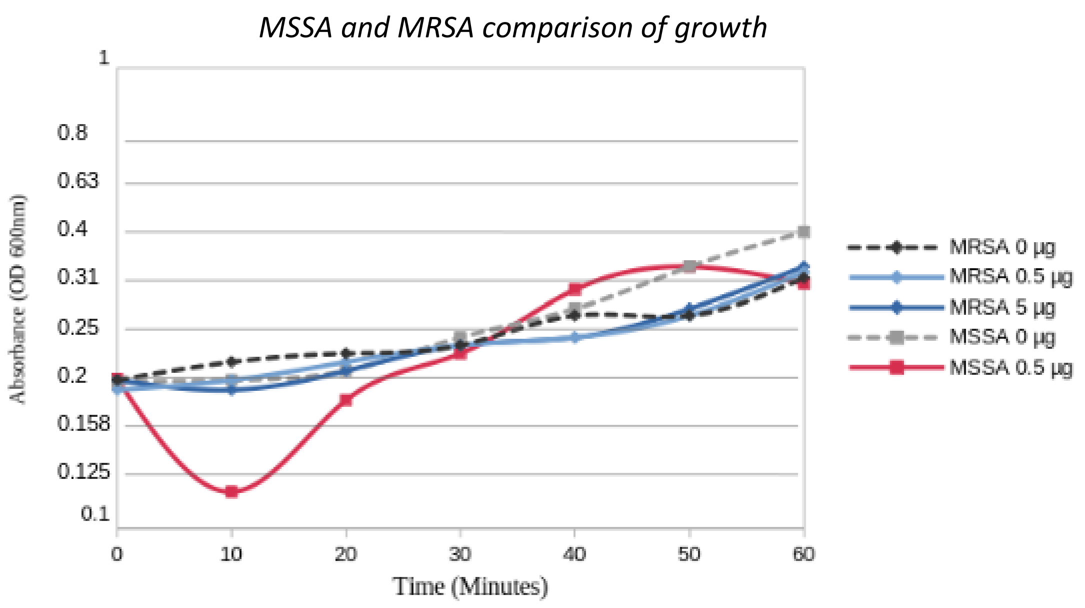


Figure 17 growth curves of control as well as dosed groups of MSSA and MRSA. Dotted grey lines display the growth curves of control samples. Red and blue curves display the growth curves of MSSA and MRSA respectively. Darker colors imply greater dosages. Note the extreme drop in absorbance at 10 minutes.

As supported by earlier growth data, we see in figure 17 that the dosed MSSA sample's process of transitioning from lag to log to stationery and death occur at a much faster rate than control and MRSA groups (Figure 17). However, this is happening at a significantly faster rate than in the primary data. As mentioned before, this difference is likely the most useful mechanism in differentiating MRSA and MSSA with this process at an early time. With that in mind, a more refined algorithm would benefit from excluding non-dosed MSSA data from the training set.

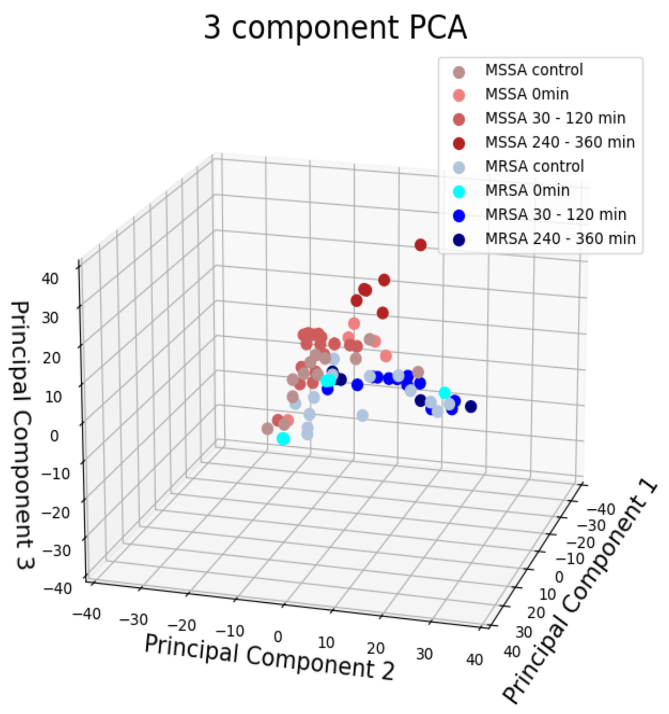


Figure 18 Three dimensional PCA of the primary and tertiary dataset. Each data-point (49 MRSA and 42 MSSA) is projected onto the first 3 principal components. Red and blue points refer to MSSA and MRSA respectively. The datapoints are grouped by growth time in darkening colors.

Figure 18 shows a PCA plot of the primary and tertiary data. As expected from the earlier analysis, data is noticeably less separable at lower time points. Even so, clear clusters are still seen both between MRSA and MSSA but also internally between time partitions. These clusters of tertiary data remain in line with clusters shown for the primary dataset. This suggests solid reproducibility even with lower quality samples. Also make note: While our first PCA plot showed solid separability of MRSA and MSSA from just PC2, this plot suggests that the best separation will come from a linear combination of PC2 and PC3.

Figure 19 shows the loading plot generated from these data-points with shorter growth times, we see new aspects of our spectrum playing a major role. From 1000 – 800 cm^{-1} new major peaks emerge. This suggests different underlying mechanisms may be at play in early lag phases compared to later lag phases. Additionally, this suggests PCA-fed LDA may under-perform unless it is trained more heavily on early time points.

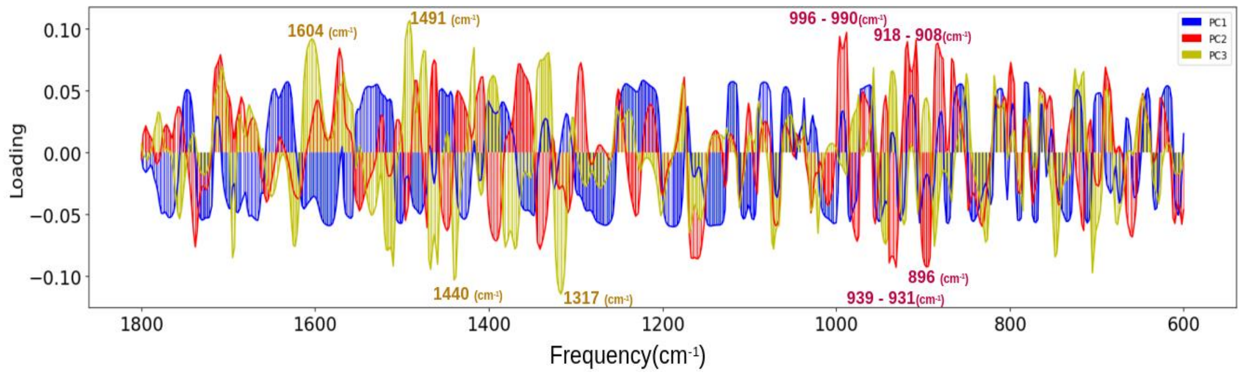


Figure 19 loading plot of primary and tertiary data. spike magnitude tells us which frequencies impact (positively or negatively) each principal component. With each principal component separated into colors. Blue red and yellow depict PC's 1, 2, and 3 respectively. Specific peaks are highlighted either due to their magnitude or known influence. Notice the newly highlighted peaks from wavenumbers 1000 cm^{-1} to 800 cm^{-1}.

As our differentiation primarily stems from principal components 2 and 3, these will be the areas focused on. Comparing this to our first loading plot, we notice some changes. Significantly more differentiation is stemming from frequencies from 1000 cm^{-1}– 800 cm^{-1}. The segment from 996 cm^{-1} – 990 cm^{-1} generates the biggest peak in this range and is associated with polysac charides and ribose (Kochan, 2020). Next, the peak at 1611 cm^{-1} has shifted to 1604 cm^{-1}. This is unlikely an actual shift but instead a result of how the principal components are split up differently in both graphs. The peaks at 1440 cm^{-1} and 1317 cm^{-1} have become more prevalent than previously shown, suggesting these areas of our spectrum play a bigger role in differentiating between samples grown for shorter periods. More work is needed for identifying underlying causes for rest of the peaks in these regions. These changes are likely to impact the differentiating algorithm when prefiltered through PCA.

Next, the original algorithm trained on the primary dataset was tested on the high-resolution, tertiary data. This was done both directly (figure 21) and by pre-filtering through PCA (figure 20) as done before. From the above growth data and PCA analysis, the expectation is for the model to perform worse due to the tighter clustering but still differentiate the data. Both figures show an LDA projection of the tertiary FTIR data projected on top of the primary FTIR data.

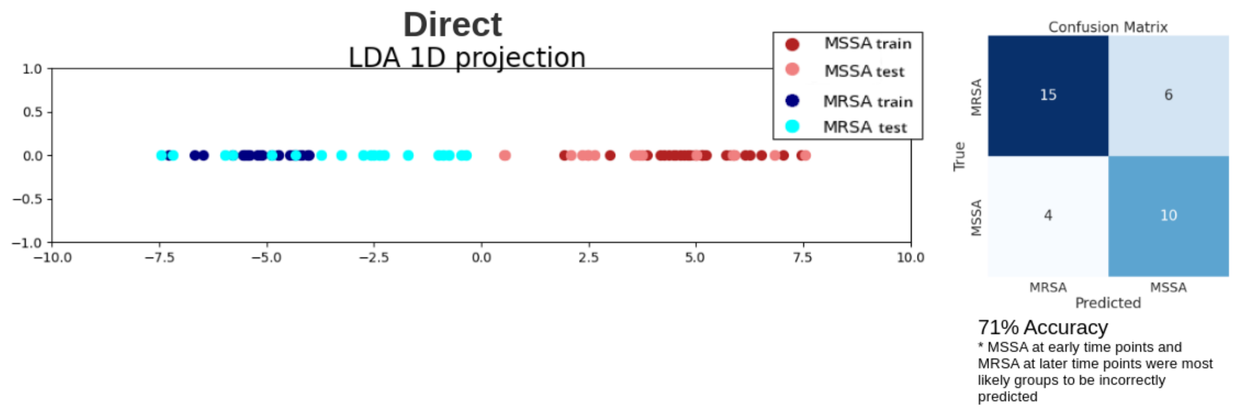


Figure 20 LDA projection of direct (non PCA filtered) tertiary data, trained on primary dataset. Dark red and blue points refer to the training MSSA and MRSA data while light red and blue points refer to the testing data projected and predicted. The confusion matrix for the example is shown to the right. Values along the primary axis show correctly predicted samples while the off axis shows inaccurately predicted data. This shows both how the samples should be categorized and what the algorithm predicted.

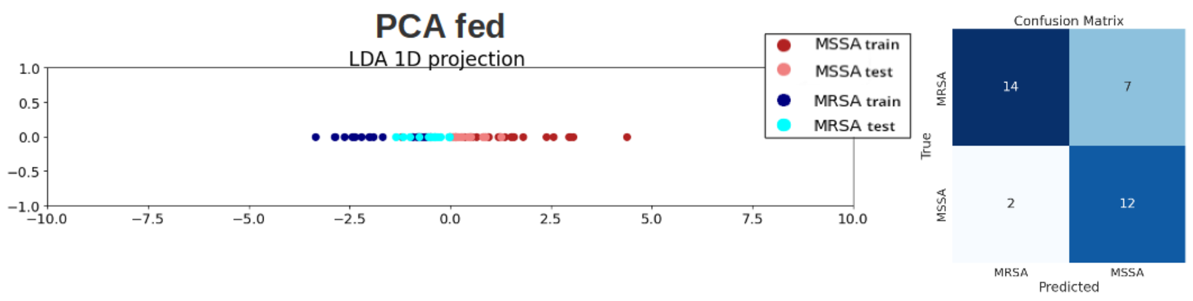


Figure 21 LDA projection of PCA filtered tertiary data, trained on the PCA filtered primary dataset. Dark red and blue points refer to the training MSSA and MRSA data while light red and blue points refer to the testing data projected and predicted. The confusion matrix for the example is shown to the right. Values along the primary axis show correctly predicted samples while the off axis shows inaccurately predicted data. This shows both how the samples should be categorized and what the algorithm predicted. Notice the very small improvement in accuracy.

3.3.1 Discussion

The LDA algorithm that was fed data directly (figure 21) performed with 71% accuracy. Specifically, it struggled categorizing MSSA samples with a shorter growth period, 0-minute samples and 10-minute samples. This was made up of two control samples (either dosed at 0 μg or grown for 0 minutes) and one dosed 10-minute sample. This latter sample could be a sign that 10 minutes is not enough time for differentiation with direct LDA or could simply call for higher resolution training data. Dosed and non-dosed MRSA also showed poor accuracy at later (40, 50, and 60-minute) growth periods. This is a more concerning result.

Next, PCA filtered data was projected (figure 20) and differentiated by the LDA algorithm. This showed a slight accuracy improvement from 71% to 74%. Accuracy on MSSA samples with short growth times improved dramatically, with the only mislabeled data-point occurring for the 0 μg sample grown for 30 minutes. This sample had an unusual artifact in the carbohydrate region at 1033 cm^{-1} which likely caused the issue. See an image of this spectra in the appendix A.2. Otherwise, just as with the previous directly fed algorithm, we still see poor differentiation for dosed and non-dosed MRSA grown for 40 to 60 minutes. This could suggest that biological and chemical changes that occur at earlier time periods may be reversed later in the lag phase.

4 CONCLUSION

These findings demonstrate that the spectral variation observed between MRSA and MSSA under differing antibiotic stress conditions combined with PCA-fed LDA offer a promising approach for the rapid identification of antibiotic resistance in Staphylococcus aureus. Overall high accuracy was achieved, however better training at earlier time points is necessary. Variations in the Amide I, II, and III regions of the spectrum, suggest potential biomarkers that could improve rapid diagnostic tests. By focusing on these regions, future studies can refine the diagnostic method, improving precision, even under non-ideal circumstances. In addition, training and testing with differing strains could help narrow in on bands that could allow for differentiation irrespective to strain. Overall, this approach shows potential for improving clinical outcomes by improving rapidity and appropriate antibiotic use, contributing to the global effort to curb the spread of AMR. However better training at earlier time points is necessary.

REFERENCES

Amiali, N. (2011). Rapid identification of community-associated methicillin-resistant Staphylococcus aureus by Fourier transform infrared spectroscopy. *Diagnostic Microbiology and Infectious Disease*. https://doi.org/10.1016/j.diagmicrobio.2010.12.016

Baker, Matthew J, et al (2019). “Using Fourier Transform IR Spectroscopy to Analyze Biological Materials.” *Nature Protocols*, vol. 9, no. 8, 3 July 2014, pp. 1771–1791, www.ncbi.nlm.nih.gov/pmc/articles/PMC4480339/, https://doi.org/10.1038/nprot.2014.110. Accessed 9 Nov. 2019.

Bertrand, Robert L (2019). “Lag Phase Is a Dynamic, Organized, Adaptive, and Evolvable Period That Prepares Bacteria for Cell Division.” *Journal of Bacteriology*, vol. 201, no. 7, 14 Jan. 2019, https://doi.org/10.1128/jb.00697-18.

Bridgeman, Adam (2024). “Spectroscopy in Organic Chemistry: Infrared.” *Sydney.edu.au*, 2024, scilearn.sydney.edu.au/OrganicSpectroscopy/index.cfm?type=Infrared&page=Fingerprint%20Region.

EUCAST (2003). “Determination of Minimum Inhibitory Concentrations (MICs) of Antibacterial Agents by Broth Dilution.” *Clinical Microbiology and Infection*, vol. 9, no. 8, Aug. 2003, pp. ix–xv, onlinelibrary.wiley.com/doi/full/10.1046/j.1469-0691.2003.00790.x, https://doi.org/10.1046/j.1469-0691.2003.00790.x.

Faix, O. (1992). Fourier transform infrared spectroscopy. In S. Y. Lin & C. W. Dence (Eds.), *Methods in lignin chemistry* (pp. 83-109). Springer. https://doi.org/10.1007/978-3-642-74065-7_7

Griffiths, P., & de Haseth, J. (2007). *Fourier transform infrared spectrometry* (2nd ed.). John Wiley & Sons. https://books.google.com/books?id=C_c0GVe8MX0C&lpg=PA75&pg=PP1#v=onepage&q&f=false

Hassoun, A., Linden, P. K., & Friedman, B. (2017). Incidence, prevalence, and management of MRSA bacteremia across patient populations—a review of recent developments in MRSA management and treatment. *Critical Care*, *21*(1). https://doi.org/10.1186/s13054-017-1801-3

Hastings, G., Nelson, M., Taylor, C. C., Marchesani, A., & Gilbert, E. (2024). Time-resolved Infrared Difference Spectroscopy for Probing Bacterial Cell Growth. *Department of Physics and Astronomy and Department of Biology, Georgia State University*.

Kochan, K., et al. (2020). Vibrational spectroscopy as a sensitive probe for the chemistry of intra-phase bacterial growth. *Sensors*, *20*(12), 3452. https://doi.org/10.3390/s20123452

Liu, F., Rajabi, S., Shi, C., et al. (2022). Antibacterial activity of recently approved antibiotics against methicillin-resistant Staphylococcus aureus (MRSA) strains: A systematic review and meta-analysis. *Annals of Clinical Microbiology and Antimicrobials*, *21*(1), 37. https://doi.org/10.1186/s12941-022-00529-z

Llor, Carl, and Lars Bjerrum (2014). "Antimicrobial Resistance: Risk Associated with Antibiotic Overuse and Initiatives to Reduce the Problem." *Therapeutic Advances in Drug Safety*, U.S. National Library of Medicine, Dec. 2014, www.ncbi.nlm.nih.gov/pmc/articles/PMC4232501/.

Murray C. (2022). Global burden of bacterial antimicrobial resistance in 2019: a systematic analysis. Lancet Planetary Health, 6(2), E75-E75.https://doi.org/10.1016/S0140-6736(21)02724-0

Smith, B. (2002). Principal components analysis. Retrieved from http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

Talsky, Gerhard, et al (1978). "High-Resolution, Higher-Order UV/VIS Derivative Spectrophotometry." *Angewandte Chemie International Edition in English*, vol. 17, no. 11, Nov. 1978, pp. 785–799, https://doi.org/10.1002/anie.197807853.

Xing, S. Y., Wei, L. Q., Abushelaibi, A., Lai, K. S., Lim, S. H. E., & Maran, S. (2023). Current molecular approach for diagnosis of MRSA: A meta-narrative review. *National Center for Biotechnology Information*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9906022/

Yourassowsky , E (2021). "Bacterial Growth Curve Antibiotics." *Jpabs.org*, 2021, jpabs.org/misc/bacterial-growth-curve-antibiotics.html.

APPENDICES

Appendix A

Appendix A.1 Example Wet/Dry samples on ATR diamond:

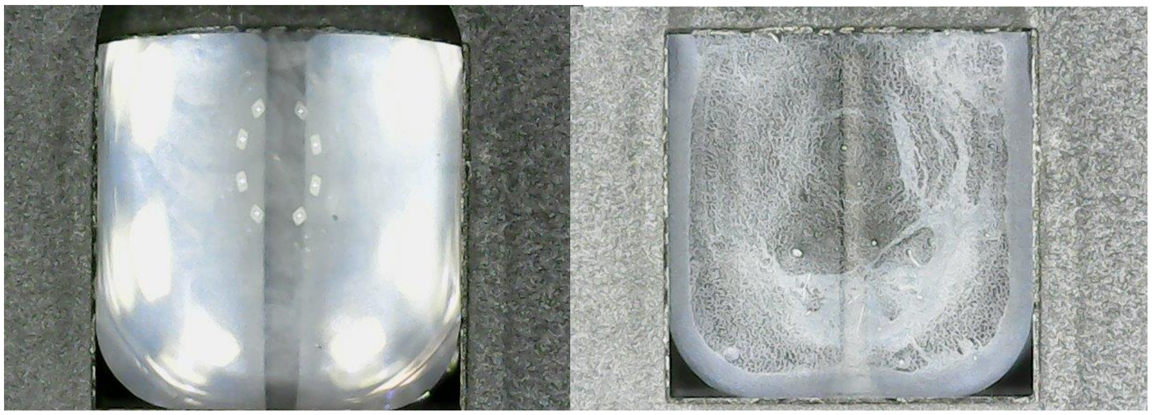


Figure 23 Anomaly at wavenumber 1033, causing inaccuracy when predicting 0 µg 30-minute MSSA sample from the tertiary dataset.

4.1.1 Appendix A.2 Anomaly

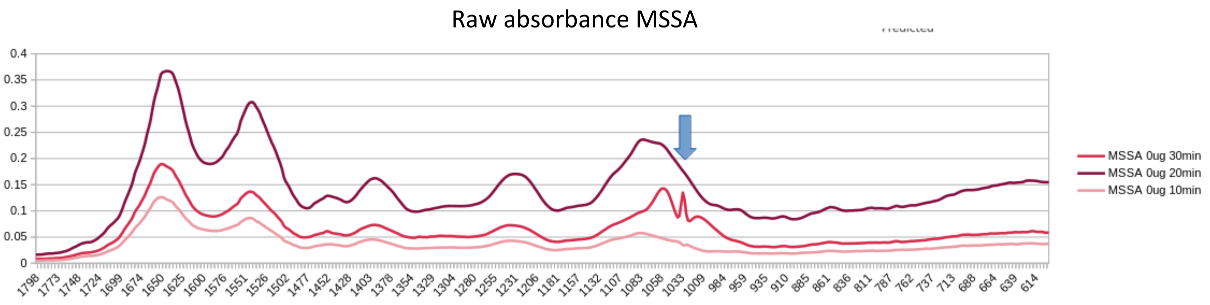


Figure 22 Example of wet and dry sample droplets on the ATR crystal

4.1.2 Appendix A.3 Band assignments table (Hastings, G 2024)

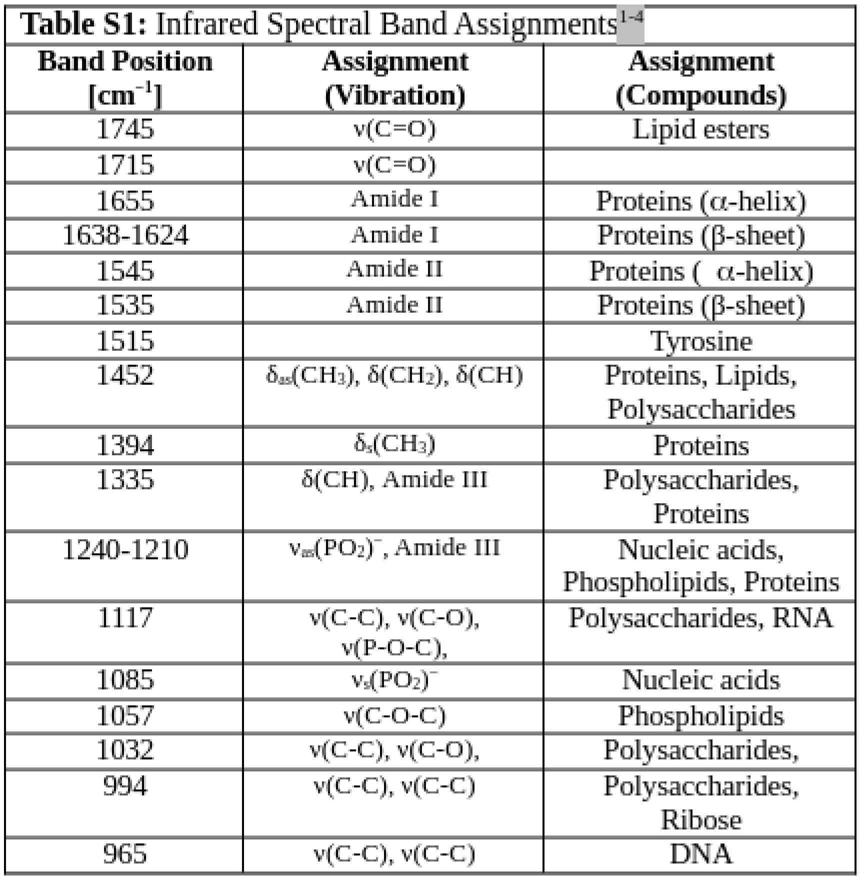


Table S1: Infrared Spectral Band Assignments[1-4]

Band Position [cm^{-1}]	Assignment (Vibration)	Assignment (Compounds)
1745	ν(C=O)	Lipid esters
1715	ν(C=O)	
1655	Amide I	Proteins (α-helix)
1638-1624	Amide I	Proteins (β-sheet)
1545	Amide II	Proteins (α-helix)
1535	Amide II	Proteins (β-sheet)
1515		Tyrosine
1452	$\delta_{as}(CH_3)$, $\delta(CH_2)$, δ(CH)	Proteins, Lipids, Polysaccharides
1394	$\delta_s(CH_3)$	Proteins
1335	δ(CH), Amide III	Polysaccharides, Proteins
1240-1210	$\nu_{as}(PO_2)^-$, Amide III	Nucleic acids, Phospholipids, Proteins
1117	ν(C-C), ν(C-O), ν(P-O-C),	Polysaccharides, RNA
1085	$\nu_s(PO_2)^-$	Nucleic acids
1057	ν(C-O-C)	Phospholipids
1032	ν(C-C), ν(C-O),	Polysaccharides,
994	ν(C-C), ν(C-C)	Polysaccharides, Ribose
965	ν(C-C), ν(C-C)	DNA

4.1.3 Appendix A.3 PCA table for primary data

	principal component 1	principal component 2	principal component 3	target
0.25ug 0min	7.352917	-17.740656	14.086074	MRSA 0min
0.25ug 30min	27.202826	-10.615076	-11.669964	MRSA 30 - 120 min
0.25ug 60min	4.025471	-2.453686	-15.158507	MRSA 30 - 120 min
0.25ug 90min	-3.983032	-16.206577	14.856929	MRSA 30 - 120 min
0.25ug 120min	-2.780521	-6.834274	-3.701390	MRSA 30 - 120 min
0.25ug 240min	-15.057350	-4.981695	-7.476533	MRSA 240 - 360 min
0.25ug 360min	-23.532542	-6.158746	1.209535	MRSA 240 - 360 min
0.5ug 0min	-13.217722	-3.211772	-10.130139	MRSA 0min
0.5ug 30min	17.451604	-1.161853	-11.279704	MRSA 30 - 120 min
0.5ug 60min	15.723695	-1.721015	-8.729767	MRSA 30 - 120 min
0.5ug 90min	24.375910	-13.800007	-5.408651	MRSA 30 - 120 min
0.5ug 120min	-2.327347	-13.710708	11.731183	MRSA 30 - 120 min
0.5ug 240min	-18.660629	-7.617693	-2.792981	MRSA 240 - 360 min
0.5ug 360min	6.710413	-24.861443	16.409108	MRSA 240 - 360 min
1ug 0min	-14.424470	-3.459996	-9.327538	MRSA 0min
1ug 30min	-3.052704	-6.014491	-4.578217	MRSA 30 - 120 min
1ug 60min	6.566952	-21.299792	15.290373	MRSA 30 - 120 min
1ug 90min	12.977814	-4.084683	-4.555770	MRSA 30 - 120 min
1ug 120min	-6.736169	-2.699206	-16.944681	MRSA 30 - 120 min
1ug 240min	2.011351	-17.062122	5.848693	MRSA 240 - 360 min
1ug 360min	-21.887479	-7.072084	-1.437103	MRSA 240 - 360 min
0ug 0min	25.068458	-9.228692	-7.648254	MRSA control
0ug 30min	9.080266	-20.678459	10.436216	MRSA control
0ug 60min	12.743305	-3.795226	-6.576035	MRSA control
0ug 90min	-18.833492	-5.610438	-3.105606	MRSA control
0ug 120min	-2.887866	-15.439338	10.511470	MRSA control
0ug 240min	26.826997	-11.254652	-11.816374	MRSA control

0ug 360min	-3.017539	-5.647838	-5.047816	MRSA control
0.5ug 0min	13.238528	8.782124	-1.343002	MSSA 0min
0.25ug 0min	5.517002	10.726203	-6.088362	MSSA 0min
1ug 0min	4.915181	12.726458	0.145364	MSSA 0min
0ug 0min	18.788427	6.649088	-5.340092	MSSA 0min
0.5ug 30min	1.945425	7.794088	-4.975821	MSSA 30 - 120 min
0.25ug 30min	0.694484	9.940443	-7.762830	MSSA 30 - 120 min
1ug 30min	7.921869	8.212556	-5.208320	MSSA 30 - 120 min
0ug 30min	10.327961	7.122662	-10.614506	MSSA control
0.5ug 60min	-20.141102	4.767704	2.715017	MSSA 30 - 120 min
0.25ug 60min	-6.769291	9.774242	-7.321995	MSSA 30 - 120 min
1ug 60min	1.942982	10.259968	-8.207487	MSSA 30 - 120 min
0ug 60min	-4.353119	5.822957	-10.678443	MSSA control
0.5ug 90min	-16.273507	8.071838	-3.789150	MSSA 30 - 120 min
0.25ug 90min	-22.329517	7.453987	2.231904	MSSA 30 - 120 min
1ug 90min	-13.033374	9.221733	-0.531566	MSSA 30 - 120 min
0ug 90min	-13.237063	4.089722	-11.809620	MSSA control
0.5ug 120min	-20.276243	8.426308	1.273772	MSSA 30 - 120 min
0.25ug 120min	-21.575742	8.711504	-0.074070	MSSA 30 - 120 min
1ug 120min	-13.354085	10.035902	-0.979855	MSSA 30 - 120 min
0ug 120min	-22.183624	1.200550	-1.579037	MSSA control
0.5ug 240min	3.925195	17.894628	14.097058	MSSA 240 - 360 min
0.25ug 240min	-9.092705	11.303473	17.304793	MSSA 240 - 360 min
1ug 240min	2.643231	18.189538	15.424134	MSSA 240 - 360 min
0ug 240min	-9.335066	-0.403582	14.620808	MSSA control
0.5ug 360min	26.920713	27.992352	28.043379	MSSA 240 - 360 min
0.25ug 360min	3.423331	16.538119	23.146345	MSSA 240 - 360 min
1ug 360min	38.512521	23.079458	-1.434544	MSSA 240 - 360 min
0ug 360min	3.518471	-9.961808	15.741574	MSSA control

Appendix B - Code

Appendix B.1 Primary dataset:

3 Dimensional PCA Analysis on primary dataset

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn_pandas import DataFrameMapper
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # Importing Axes3D module from mpl_toolkits.mplot3d

# Load the dataset
df = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA-PCA.ods', index_col=0, engine='odf')

# Extract frequency columns and target column
freq = list(df.columns[0:634].values)
x = df.loc[:, freq].values
y = df.loc[:, ['target']].values

# Scale the features
mapper = DataFrameMapper([(freq, StandardScaler())])  # Use freq instead of df.columns
scaled_features = mapper.fit_transform(df)  # Pass df directly to fit_transform
scaled_features_df = pd.DataFrame(scaled_features, index=df.index, columns=freq)

# Apply PCA
pca = PCA(n_components=3)
principal_components = pca.fit_transform(scaled_features_df)
principal_df = pd.DataFrame(data=principal_components, columns=['principal component 1', 'principal component 2', 'principal
component 3'], index=scaled_features_df.index)

# Compute loadings
loadings = pd.DataFrame(data=pca.components_.T, columns=['PCA1', 'PCA2', 'PCA3'], index=freq)

# Combine PCA results with target column
final_df = pd.concat([principal_df, df[['target']]], axis=1)
print(final_df)

# Explained variance
exp_var_pca = pca.explained_variance_ratio_
cum_sum_eigenvalues = np.cumsum(exp_var_pca)

# 3D Plot
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')
ax.set_xlabel('Principal Component 1', fontsize=15)
ax.set_ylabel('Principal Component 2', fontsize=15)
ax.set_zlabel('Principal Component 3', fontsize=15)
ax.set_title('3 component PCA', fontsize=20)
max_axis_size = 40  # Set your desired maximum axis size here
ax.set_xlim([-max_axis_size, max_axis_size])
ax.set_ylim([-max_axis_size, max_axis_size])
ax.set_zlim([-max_axis_size, max_axis_size])

# Define targets and colors
targets = ['MSSA control', 'MSSA 0min', 'MSSA 30 - 120 min', 'MSSA 240 - 360 min', 'MRSA control', 'MRSA 0min', 'MRSA 30 -
120 min', 'MRSA 240 - 360 min']
colors = ['rosybrown', 'lightcoral', 'indianred', 'firebrick', 'lightsteelblue', 'cyan', 'blue', 'navy']

# Scatter plot for each target
for target, color in zip(targets, colors):
    indices_to_keep = final_df['target'] == target
    ax.scatter(final_df.loc[indices_to_keep, 'principal component 1'],
               final_df.loc[indices_to_keep, 'principal component 2'],
               final_df.loc[indices_to_keep, 'principal component 3'],
            c=color,
            s=50, alpha=1)
ax.legend(targets)
ax.grid()
plt.show()

# Print sorted loadings
print(loadings.sort_values("PCA2"))
print(loadings.sort_values("PCA3"))

# Plot loadings
fig, ax = plt.subplots(figsize=(8, 8))
plt.bar(loadings.index, loadings.iloc[:, 0], label='PC1', color='b')
plt.plot(loadings.index, loadings.iloc[:, 0], color='b')
plt.bar(loadings.index + 0.2, loadings.iloc[:, 1], label='PC2', color='r')
plt.plot(loadings.index, loadings.iloc[:, 1], color='r')
plt.bar(loadings.index + 0.4, loadings.iloc[:, 2], label='PC3', color='y')
plt.plot(loadings.index, loadings.iloc[:, 2], color='y')

plt.tick_params(axis='both', which='major', labelsize=18)
plt.ylabel('Loading', fontsize=18)
plt.xlabel('Frequency', fontsize=18)
plt.gca().invert_xaxis()
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

LDA tested on primary dataset partitioned by dosage:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn_pandas import DataFrameMapper
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Load the dataset
df = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA.ods', index_col=0, engine='odf')

# Standardize the features
scaler = StandardScaler()
scaler.fit(df.iloc[:, 0:634].values)

# Splitting data into testing and training sets
groupie = df.groupby(df.index)
drug = input("Enter value to test: ")

# Define test and train datasets based on the drug input
test = pd.concat([
    groupie.get_group(f"{drug}ug 0min"),
    groupie.get_group(f"{drug}ug 30min"),
    groupie.get_group(f"{drug}ug 60min"),
    groupie.get_group(f"{drug}ug 90min"),
    groupie.get_group(f"{drug}ug 120min"),
    groupie.get_group(f"{drug}ug 240min"),
    groupie.get_group(f"{drug}ug 360min")
])

train = df.drop([
    f"{drug}ug 0min",
    f"{drug}ug 30min",
    f"{drug}ug 60min",
    f"{drug}ug 90min",
    f"{drug}ug 120min",
    f"{drug}ug 240min",
    f"{drug}ug 360min"
])

# Extract frequency columns for test and train sets
freq_test = list(test.columns[0:634].values)
freq_train = list(train.columns[0:634].values)

# Prepare training and testing data
x_test = test.loc[:, freq_test].values
x_train = train.loc[:, freq_train].values
y_test = np.ravel(test.loc[:, ['target']].values)
y_train = np.ravel(train.loc[:, ['target']].values)

# Standardize the features
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_test)

# Apply Linear Discriminant Analysis (LDA)
lda = LinearDiscriminantAnalysis()
lda.fit(x_train_scaled, y_train)

# Transform training data using LDA
x_train_lda = lda.fit_transform(x_train, y_train)
y_pred = lda.predict(x_test_scaled)
accuracy = lda.score(x_test_scaled, y_test)

print("Accuracy: {:.2f}%".format(accuracy * 100))

# Plot the LDA-transformed feature space
```

```
x_train_df = pd.DataFrame(x_train_lda, columns=['LDA feature 1'])
y_train_df = pd.DataFrame(y_train, columns=['target'])
final_df = pd.concat([x_train_df, y_train_df], axis=1)

df_test = pd.concat([
    pd.DataFrame(lda.transform(x_test_scaled), columns=['LDA1']),
    pd.DataFrame(y_pred, columns=['target'])
], axis=1)
print(df_test)

# Plot training and test data in LDA space
targets = ['MSSA', 'MRSA']
colors = ['firebrick', 'navy']
test_colors = ['lightcoral', 'cyan']

for target, color in zip(targets, colors):
    indices_to_keep = final_df['target'] == target
    plt.scatter(
        final_df.loc[indices_to_keep, 'LDA feature 1'],
        [0] * final_df.loc[indices_to_keep, 'LDA feature 1'].shape[0],
        c=color, s=50, label=f'Train {target}'
    )

for target, color in zip(targets, test_colors):
    indices_to_keep = df_test['target'] == target
    plt.scatter(
        df_test.loc[indices_to_keep, 'LDA1'],
        [0] * df_test.loc[indices_to_keep, 'LDA1'].shape[0],
        c=color, s=50, label=f'Test {target}'
    )

plt.legend()
plt.xlabel('LDA projection', fontsize=18)
plt.ylabel('', fontsize=18)
plt.suptitle('LDA 1D projection', fontsize=20)
plt.xlim([-10, 10])
plt.ylim([-1, 1])
plt.show()

# Display the predicted vs actual values
oracle = pd.concat([pd.DataFrame(x_test_scaled), pd.Series(y_pred), pd.Series(y_test)], axis=1)
print(oracle)

# Confusion matrix
conf_m = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Display the confusion matrix as a heatmap
labels = ["MRSA", "MSSA"]
plt.figure(figsize=(6, 6))
sns.set(font_scale=2)
sns.heatmap(conf_m, annot=True, fmt="d", cmap="Blues", cbar=False, square=True,
            xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

LDA tested on primary dataset partitioned by time:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn_pandas import DataFrameMapper
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Load the dataset
df = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA.ods', index_col=0, engine='odf')

# Extract frequency columns and absorbance values
freq = list(df.columns[0:634].values)
df_absorbance = df.loc[:, freq].values

# Splitting data into testing and training sets
groupie = df.groupby(df.index)
drug = input("Enter time value to test: ")

test = pd.concat([groupie.get_group(f"0ug {drug}"),
                  groupie.get_group(f"0.25ug {drug}"),
                  groupie.get_group(f"0.5ug {drug}"),
                  groupie.get_group(f"1ug {drug}")])

train = df.drop(["0ug " + drug, "0.25ug " + drug, "0.5ug " + drug, "1ug " + drug])

# Extracting frequency columns for test and train sets
freq_test = list(test.columns[0:634].values)
freq_train = list(train.columns[0:634].values)

# Prepare training and testing data
x_test = test.loc[:, freq_test].values
x_train = train.loc[:, freq_train].values
y_test = np.ravel(test.loc[:, ['target']].values)
y_train = np.ravel(train.loc[:, ['target']].values)

# Standardize the features
scaler = StandardScaler()
scaler.fit(df_absorbance)
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_test)

# Apply Linear Discriminant Analysis (LDA)
lda = LinearDiscriminantAnalysis()
lda.fit(x_train_scaled, y_train)
x_train_lda = lda.fit_transform(x_train_scaled, y_train)
y_pred = lda.predict(x_test_scaled)
accuracy = lda.score(x_test_scaled, y_test)

print("Accuracy: {:.2f}%".format(accuracy * 100))

# Plot the LDA-transformed feature space
x_train_df = pd.DataFrame(x_train_lda, columns=['LDA feature 1'])
y_train_df = pd.DataFrame(y_train, columns=['target'])
final_df = pd.concat([x_train_df, y_train_df], axis=1)

df_test = pd.concat([pd.DataFrame(lda.transform(x_test_scaled), columns=['LDA1']),
                     pd.DataFrame(y_pred, columns=['target'])], axis=1)
print(df_test)

# Plot training and test data in LDA space
targets = ['MSSA', 'MRSA']
colors = ['firebrick', 'navy']
test_colors = ['lightcoral', 'cyan']

for target, color in zip(targets, colors):
    indices_to_keep = final_df['target'] == target
    plt.scatter(final_df.loc[indices_to_keep, 'LDA feature 1'],
                [0] * final_df.loc[indices_to_keep, 'LDA feature 1'].shape[0],
                c=color, s=50, label=f'Train {target}')

for target, color in zip(targets, test_colors):
    indices_to_keep = df_test['target'] == target
    plt.scatter(df_test.loc[indices_to_keep, 'LDA1'],
                [0] * df_test.loc[indices_to_keep, 'LDA1'].shape[0],
                c=color, s=50, label=f'Test {target}')

plt.legend()
plt.xlabel('LDA projection', fontsize=18)
plt.suptitle('LDA 1D projection', fontsize=20)
plt.xlim([-10, 10])
plt.ylim([-1, 1])
plt.show()

# Display the predicted vs actual values
oracle = pd.concat([pd.DataFrame(x_test), pd.Series(y_pred), pd.Series(y_test)], axis=1)
print(oracle)

# Confusion matrix
conf_m = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Display the confusion matrix as a heatmap
labels = ["MRSA", "MSSA"]
plt.figure(figsize=(6, 6))
sns.set(font_scale=2)
sns.heatmap(conf_m, annot=True, fmt="d", cmap="Blues", cbar=False, square=True,
            xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

PCA fed LDA tested on primary dataset partitioned by dosage:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn_pandas import DataFrameMapper
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Load the dataset
df = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA.ods', index_col=0, engine='odf')
freq = list(df.columns[0:634].values)

# Extract features and target
x = df.loc[:, freq].values
y = df.loc[:, ['target']].values

# Standardize the features
mapper = DataFrameMapper([(freq, StandardScaler())])
scaled_features = mapper.fit_transform(df)
scaled_features_df = pd.DataFrame(scaled_features, index=df.index, columns=freq)

# Apply PCA
pca = PCA(n_components=10)
principalComponents = pca.fit_transform(scaled_features_df)
principalDf = pd.DataFrame(data=principalComponents, columns=[str(i) for i in range(1, 11)], index=scaled_features_df.index)
finalDf = pd.concat([principalDf, df[['target']]], axis=1)

# Prepare data for LDA
df = finalDf
print(df)

# Group the data by index and prompt user for a drug value
groupie = df.groupby(df.index)
drug = input("Enter value to test: ")

# Split data into train and test sets
test = pd.concat([groupie.get_group(f"{drug}ug {time}min") for time in [0, 30, 60, 90, 120, 240, 360]])
train = df.drop([f"{drug}ug {time}min" for time in [0, 30, 60, 90, 120, 240, 360]])

# Extract features and target for train and test sets
freq_test = list(test.columns[0:10].values)
freq_train = list(train.columns[0:10].values)

x_test = test.loc[:, freq_test].values
x_train = train.loc[:, freq_train].values
y_test = np.ravel(test.loc[:, ['target']].values)
y_train = np.ravel(train.loc[:, ['target']].values)

# Scale features (optional, commented out)
# scaler = StandardScaler()
# x_train_scaled = scaler.fit_transform(x_train)
# x_test_scaled = scaler.transform(x_test)

x_train_scaled = x_train  # Temporary
x_test_scaled = x_test    # Temporary

# Apply LDA
lda = LinearDiscriminantAnalysis()
lda.fit(x_train_scaled, y_train)

x_train_lda = lda.fit_transform(x_train_scaled, y_train)
y_pred = lda.predict(x_test_scaled)
accuracy = lda.score(x_test_scaled, y_test)
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
# Plot the LDA-transformed feature space
x_train_df = pd.DataFrame(x_train_lda, columns=['LDA feature 1'])
y_train_df = pd.DataFrame(y_train, columns=['target'])
final_df = pd.concat([x_train_df, y_train_df], axis=1)
df_test = pd.concat([pd.DataFrame(lda.transform(x_test_scaled), columns=['LDA1']), pd.DataFrame(y_pred, columns=['target'])], 
axis=1)
print(df_test)

targets = ['MSSA', 'MRSA']
colors = ['firebrick', 'navy']
targets2 = ['MSSA', 'MRSA']
colors2 = ['lightcoral', 'cyan']

for target, color in zip(targets, colors):
    indicesToKeep = final_df['target'] == target
    plt.scatter(final_df.loc[indicesToKeep, 'LDA feature 1'], [0] * final_df.loc[indicesToKeep, 'LDA feature 1'].shape[0], c=color, 
s=50)

for target, color in zip(targets2, colors2):
    indicesToKeep = df_test['target'] == target
    plt.scatter(df_test.loc[indicesToKeep, 'LDA1'], [0] * df_test.loc[indicesToKeep, 'LDA1'].shape[0], c=color, s=50)

plt.legend(targets)
plt.xlabel('LDA projection', fontsize=18)
plt.ylabel('', fontsize=18)
plt.suptitle('LDA 1D projection', fontsize=20)
plt.xlim([-10, 10])
plt.ylim([-1, 1])
plt.show()

# Concatenate and print test results
oracle = pd.concat([pd.DataFrame(x_test_scaled), pd.Series(y_pred), pd.Series(y_test)], axis=1)
print(oracle)

# Compute and display the confusion matrix
conf_m = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

labels = ["MRSA", "MSSA"]
plt.figure(figsize=(6, 6))
sns.set(font_scale=2)
sns.heatmap(conf_m, annot=True, fmt="d", cmap="Blues", cbar=False, square=True, xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

PCA fed LDA tested on primary dataset partitioned by time:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn_pandas import DataFrameMapper
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import confusion_matrix

# Load and preprocess the dataset
df = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA.ods', index_col=0, engine='odf')
freq = df.columns[0:634].values.tolist()

x = df.loc[:, freq].values
y = df.loc[:, ['target']].values

# Standardize the features
mapper = DataFrameMapper([(freq, StandardScaler())])
scaled_features = mapper.fit_transform(df)
scaled_features_df = pd.DataFrame(scaled_features, index=df.index, columns=freq)

# Apply PCA
pca = PCA(n_components=10)
principalComponents = pca.fit_transform(scaled_features_df)
principalDf = pd.DataFrame(data=principalComponents, columns=[str(i) for i in range(1, 11)], index=scaled_features_df.index)
finalDf = pd.concat([principalDf, df[['target']]], axis=1)

# Prepare data for LDA
df = finalDf
print(df)

# Group the data by index and prompt user for a drug value
groupie = df.groupby(df.index)
drug = input("Enter value to test: ")

# Split data into train and test sets
test = pd.concat([groupie.get_group(f"{time}ug {drug}") for time in ["0", "0.25", "0.5", "1"]])
train = df.drop([f"{time}ug {drug}" for time in ["0", "0.25", "0.5", "1"]])

# Extract features and target for train and test sets
freq_test = test.columns[0:10].values.tolist()
freq_train = train.columns[0:10].values.tolist()

x_test = test.loc[:, freq_test].values
x_train = train.loc[:, freq_train].values
y_test = np.ravel(test.loc[:, ['target']].values)
y_train = np.ravel(train.loc[:, ['target']].values)

# Apply LDA
lda = LinearDiscriminantAnalysis()
lda.fit(x_train, y_train)

x_train_lda = lda.fit_transform(x_train, y_train)
y_pred = lda.predict(x_test)
accuracy = lda.score(x_test, y_test)
print(f"Accuracy: {accuracy:.2f}%")

# Plot the LDA-transformed feature space
x_train_df = pd.DataFrame(x_train_lda, columns=['LDA feature 1'])
y_train_df = pd.DataFrame(y_train, columns=['target'])
final_df = pd.concat([x_train_df, y_train_df], axis=1)
df_test = pd.concat([pd.DataFrame(lda.transform(x_test), columns=['LDA1']), pd.DataFrame(y_pred, columns=['target'])], axis=1)
print(df_test)

# Visualize the LDA results
targets = ['MSSA', 'MRSA']
```

```
colors = ['firebrick', 'navy']
targets2 = ['MSSA', 'MRSA']
colors2 = ['lightcoral', 'cyan']

for target, color in zip(targets, colors):
    indicesToKeep = final_df['target'] == target
    plt.scatter(final_df.loc[indicesToKeep, 'LDA feature 1'], [0] * final_df.loc[indicesToKeep, 'LDA feature 1'].shape[0], c=color, 
s=50)

for target, color in zip(targets2, colors2):
    indicesToKeep = df_test['target'] == target
    plt.scatter(df_test.loc[indicesToKeep, 'LDA1'], [0] * df_test.loc[indicesToKeep, 'LDA1'].shape[0], c=color, s=50)

plt.legend(targets)
plt.xlabel('LDA projection', fontsize=18)
plt.ylabel('', fontsize=18)
plt.suptitle('LDA 1D projection', fontsize=20)
plt.xlim([-10, 10])
plt.ylim([-1, 1])
plt.show()

# Print test results
oracle = pd.concat([pd.DataFrame(x_test), pd.Series(y_pred), pd.Series(y_test)], axis=1)
print(oracle)

# Compute and display the confusion matrix
conf_m = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

labels = ["MRSA", "MSSA"]
plt.figure(figsize=(6, 6))
sns.set(font_scale=2)
sns.heatmap(conf_m, annot=True, fmt="d", cmap="Blues", cbar=False, square=True, xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

3 Dimensional PCA Analysis on secondary dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # Importing Axes3D for 3D plotting
from scipy.signal import savgol_filter
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn_pandas import DataFrameMapper

# Load initial dataset
df_init = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA-PCA.ods', index_col=0, engine='odf')
freq_init = df_init.columns[0:585].values.tolist()

# Load test dataset
test_raw = pd.read_excel('~/Documents/Project/Clump/test_data_PCA.ods', index_col=0, engine='odf')
freq_test = test_raw.columns[0:585].values.tolist()

np.set_printoptions(precision=2)  # Set numpy print options for compact display

# Apply Savitzky-Golay filter to smooth the test data
test = pd.DataFrame()
for sample in test_raw.index:
    d = test_raw.loc[sample]
    d_filtered = savgol_filter(d.iloc[0:585], 9, 2, deriv=2)
    d_filtered_series = pd.Series(d_filtered, index=freq_test, name=sample)
    test = pd.concat([test, d_filtered_series], axis=1)

# Transpose test data and add target column
test = test.transpose()
target_series = pd.Series(test_raw['target'].values.flatten(), index=test_raw.index, name='target')
test = pd.concat([test, target_series], axis=1)

# Save processed test data to Excel
test.to_excel("2D-ABS", engine='odf')

# Combine test data with initial dataset
df = pd.concat([test, df_init])
df_absorbance = df.loc[:, freq_init].values

# Standardize the features
mapper = DataFrameMapper([(freq_init, StandardScaler())])
scaled_features = mapper.fit_transform(df)
scaled_features_df = pd.DataFrame(scaled_features, index=df.index, columns=freq_init)

# Apply PCA
pca = PCA(n_components=20)
principalComponents = pca.fit_transform(scaled_features_df)
principalDf = pd.DataFrame(data=principalComponents, columns=[f'principal component {i}' for i in range(1, 21)],
index=scaled_features_df.index)

# Get PCA loadings
loadings = pd.DataFrame(data=pca.components_.T, columns=[f'PCA{i}' for i in range(1, 21)], index=freq_init)

# Add target column to principal components DataFrame
finalDf = pd.concat([principalDf, df[['target']]], axis=1)

# Print the final DataFrame
pd.set_option("display.max_rows", 100)
print(finalDf)

# Calculate explained variance
exp_var_pca = pca.explained_variance_ratio_
cum_sum_eigenvalues = np.cumsum(exp_var_pca)

# Plot 3D PCA
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')
```

```
ax.set_xlabel('Principal Component 1', fontsize=15)
ax.set_ylabel('Principal Component 2', fontsize=15)
ax.set_zlabel('Principal Component 3', fontsize=15)
ax.set_title('3 component PCA', fontsize=20)
max_axis_size = 40
ax.set_xlim([-max_axis_size, max_axis_size])
ax.set_ylim([-max_axis_size, max_axis_size])
ax.set_zlim([-max_axis_size, max_axis_size])

# Define targets and colors for plotting
targets = ['MSSA control', 'MSSA 0min', 'MSSA 30 - 120 min', 'MSSA 240 - 360 min', 'MRSA control', 'MRSA 0min', 'MRSA 30 -
120 min', 'MRSA 240 - 360 min']
colors = ['rosybrown', 'lightcoral', 'indianred', 'firebrick', 'lightsteelblue', 'cyan', 'blue', 'navy']

# Plot each target group in different colors
for target, color in zip(targets, colors):
    indicesToKeep = finalDf['target'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1'],
               finalDf.loc[indicesToKeep, 'principal component 2'],
               finalDf.loc[indicesToKeep, 'principal component 3'],
               c=color, s=50, alpha=1)
ax.legend(targets)
ax.grid()
plt.show()

# Plot PCA loadings
fig, ax = plt.subplots(figsize=(8, 8))
ax.bar(loadings.index, loadings.iloc[:, 0], width=0.2, label='PC1', color='b')
ax.bar(loadings.index + 0.2, loadings.iloc[:, 1], width=0.2, label='PC2', color='r')
ax.bar(loadings.index + 0.4, loadings.iloc[:, 2], width=0.2, label='PC3', color='y')
ax.set_ylabel('Loading')
ax.set_xlabel('Frequency')
ax.legend(loc='best')
plt.tight_layout()
plt.show()

# Plot explained variance
plt.figure(figsize=(8, 8))
plt.bar(range(len(exp_var_pca)), exp_var_pca, alpha=0.5, align='center', label='Individual explained variance')
plt.step(range(len(cum_sum_eigenvalues)), cum_sum_eigenvalues, where='mid', label='Cumulative explained variance')
plt.ylabel('Explained variance ratio', fontsize=18)
plt.xlabel('Principal component index', fontsize=18)
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

Direct Linear discriminant analysis on secondary data with SG filter:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn_pandas import DataFrameMapper
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import confusion_matrix
from scipy.signal import savgol_filter
import seaborn as sns

# Load data
df = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA.ods', index_col=0, engine='odf')
test_raw = pd.read_excel('~/Dsocuments/Project/Clump/test_data.ods', index_col=0, engine='odf')

# Extract frequency columns
freq_test = test_raw.columns[0:585].values.tolist()
np.set_printoptions(precision=2)  # Set numpy print options for compact display

# Display sample data
print(test_raw.iloc[0, 0:585])
print(test_raw.index)

# Apply Savitzky-Golay filter to smooth the test data
test = pd.DataFrame()
for sample in test_raw.index:
    d = test_raw.loc[sample]
    d_filtered = savgol_filter(d.iloc[0:585], 9, 2, deriv=2)
    d_filtered_series = pd.Series(d_filtered, index=freq_test, name=sample)
    test = pd.concat([test, d_filtered_series], axis=1)

# Transpose and add target column to test data
test = test.transpose()
target_series = pd.Series(test_raw['target'].values.flatten(), index=test_raw.index, name='target')
test = pd.concat([test, target_series], axis=1)
print(test)

# Combine test data with training data
train = df

# Extract frequency columns
freq_train = train.columns[0:585].values.tolist()

# Prepare training and test data
x_train = train.loc[:, freq_train].values
y_train = np.ravel(train.loc[:, ['target']].values)
x_test = test.loc[:, freq_test].values
y_test = np.ravel(test.loc[:, ['target']].values)

# Standardize the features
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.fit_transform(x_test)

# Apply Linear Discriminant Analysis (LDA)
lda = LinearDiscriminantAnalysis()
lda.fit(x_train_scaled, y_train)

# Transform data using LDA
x_train_lda = lda.fit_transform(x_train_scaled, y_train)
y_pred = lda.predict(x_test_scaled)
accuracy = lda.score(x_test_scaled, y_test)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Plot the LDA-transformed feature space
x_train_df = pd.DataFrame(x_train_lda, columns=['LDA feature 1'])
y_train_df = pd.DataFrame(y_train, columns=['target'])
final_df = pd.concat([x_train_df, y_train_df], axis=1)
```

```
df_test = pd.concat([pd.DataFrame(lda.transform(x_test_scaled), columns=['LDA1']), pd.DataFrame(y_pred, columns=['target'])],
axis=1)
print(df_test)

# Plot the results
targets = ['MSSA', 'MRSA']
colors = ['firebrick', 'navy']
colors2 = ['lightcoral', 'cyan']

for target, color in zip(targets, colors):
    indices_to_keep = final_df['target'] == target
    plt.scatter(final_df.loc[indices_to_keep, 'LDA feature 1'], [0] * final_df.loc[indices_to_keep, 'LDA feature 1'].shape[0], c=color,
s=50)

for target, color in zip(targets, colors2):
    indices_to_keep = df_test['target'] == target
    plt.scatter(df_test.loc[indices_to_keep, 'LDA1'], [0] * df_test.loc[indices_to_keep, 'LDA1'].shape[0], c=color, s=50)

plt.legend(targets)
plt.xlabel('LDA projection', fontsize=15)
plt.ylabel('', fontsize=15)
plt.suptitle('LDA 1D projection', fontsize=20)
plt.xlim([-10, 10])
plt.ylim([-1, 1])
plt.show()

# Display oracle dataframe
oracle = pd.concat([pd.DataFrame(x_test), pd.Series(y_pred), pd.Series(y_test)], axis=1)
print(oracle)

# Compute and display the confusion matrix
conf_m = confusion_matrix(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Display the confusion matrix as a heatmap
labels = ["MRSA", "MSSA"]
plt.figure(figsize=(6, 6))
sns.set(font_scale=2)
sns.heatmap(conf_m, annot=True, fmt="d", cmap="Blues", cbar=False, square=True,
            xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

PCA fed Linear discriminant analysis on secondary data with SG filter:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn_pandas import DataFrameMapper
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix
from scipy.signal import savgol_filter

# Load initial and test data
df_init = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA.ods', index_col=0, engine='odf')
test_raw = pd.read_excel('~/Documents/Project/Clump/test_data.ods', index_col=0, engine='odf')

# Extract frequency columns
freq_init = df_init.columns[0:585].values.tolist()
freq_test = test_raw.columns[0:585].values.tolist()

# Set numpy print options
np.set_printoptions(precision=2)

# Apply Savitzky-Golay filter to test data
test = pd.DataFrame()
for sample in test_raw.index:
    d = test_raw.loc[sample]
    d_filtered = savgol_filter(d.iloc[0:585], 9, 2, deriv=2)
    d_filtered_series = pd.Series(d_filtered, index=freq_test, name=sample)
    test = pd.concat([test, d_filtered_series], axis=1)

# Transpose and add target column to test data
test = test.transpose()
target_series = pd.Series(test_raw['target'].values.flatten(), index=test_raw.index, name='target')
test = pd.concat([test, target_series], axis=1)

# Combine test and initial data
df = pd.concat([test, df_init])

# Standardize features
mapper = DataFrameMapper([(freq_init, StandardScaler())])
scaled_features = mapper.fit_transform(df)
scaled_features_df = pd.DataFrame(scaled_features, index=df.index, columns=freq_init)

# Apply PCA
pca = PCA(n_components=20)
principal_components = pca.fit_transform(scaled_features_df)
principal_df = pd.DataFrame(data=principal_components, columns=[f'principal component {i+1}' for i in range(20)],
index=scaled_features_df.index)

# Combine PCA results with target column
final_df = pd.concat([principal_df, df[['target']]], axis=1)

# Split data into training and test sets
test = final_df.iloc[:27, :]
train = final_df.iloc[27:, :]

# Prepare training and test data
x_train = train.iloc[:, :-1].values
y_train = np.ravel(train.iloc[:, [-1]].values)
x_test = test.iloc[:, :-1].values
y_test = np.ravel(test.iloc[:, [-1]].values)

# Apply Linear Discriminant Analysis (LDA)
lda = LinearDiscriminantAnalysis()
lda.fit(x_train, y_train)
x_train_lda = lda.fit_transform(x_train, y_train)
y_pred = lda.predict(x_test)
```

```
accuracy = lda.score(x_test, y_test)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Plot the LDA-transformed feature space
x_train_df = pd.DataFrame(x_train_lda, columns=['LDA feature 1'])
y_train_df = pd.DataFrame(y_train, columns=['target'])
final_train_df = pd.concat([x_train_df, y_train_df], axis=1)

x_test_lda = lda.transform(x_test)
x_test_df = pd.DataFrame(x_test_lda, columns=['LDA1'])
y_test_df = pd.DataFrame(y_pred, columns=['target'])
final_test_df = pd.concat([x_test_df, y_test_df], axis=1)

# Plot the training and test data in LDA feature space
targets = ['MSSA', 'MRSA']
colors_train = ['firebrick', 'navy']
colors_test = ['lightcoral', 'cyan']

plt.figure(figsize=(10, 6))
for target, color in zip(targets, colors_train):
    indices = final_train_df['target'] == target
    plt.scatter(final_train_df.loc[indices, 'LDA feature 1'], [0] * sum(indices), c=color, label=target + ' Train', s=50)

for target, color in zip(targets, colors_test):
    indices = final_test_df['target'] == target
    plt.scatter(final_test_df.loc[indices, 'LDA1'], [0] * sum(indices), c=color, label=target + ' Test', s=50)

plt.legend()
plt.tick_params(axis='both', which='major', labelsize=14)
plt.xlabel('LDA projection', fontsize=18)
plt.ylabel('', fontsize=18)
plt.suptitle('LDA 1D projection', fontsize=20)
plt.xlim([-10, 10])
plt.ylim([-1, 1])
plt.show()

# Display the confusion matrix as a heatmap
conf_m = confusion_matrix(y_test, y_pred, labels=["MRSA", "MSSA"])
print(f'Accuracy: {accuracy:.2f}')

plt.figure(figsize=(6, 6))
sns.set(font_scale=2)
sns.heatmap(conf_m, annot=True, fmt="d", cmap="Blues", cbar=False, square=True,
            xticklabels=targets, yticklabels=targets)

plt.tick_params(axis='both', which='major')
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

PCA analysis on tertiary data with SG filter

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  # Importing Axes3D for 3D plotting
from scipy.signal import savgol_filter
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn_pandas import DataFrameMapper

# Load initial dataset
df_init = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA-PCA.ods', index_col=0, engine='odf')
freq_init = df_init.columns[0:585].values.tolist()

# Load test dataset
test_raw = pd.read_excel('~/Documents/Project/Clump/test_data_PCA.ods', index_col=0, engine='odf')
freq_test = test_raw.columns[0:585].values.tolist()

np.set_printoptions(precision=2)  # Set numpy print options for compact display

# Apply Savitzky-Golay filter to smooth the test data
test = pd.DataFrame()
for sample in test_raw.index:
    d = test_raw.loc[sample]
    d_filtered = savgol_filter(d.iloc[0:585], 9, 2, deriv=2)
    d_filtered_series = pd.Series(d_filtered, index=freq_test, name=sample)
    test = pd.concat([test, d_filtered_series], axis=1)

# Transpose test data and add target column
test = test.transpose()
target_series = pd.Series(test_raw['target'].values.flatten(), index=test_raw.index, name='target')
test = pd.concat([test, target_series], axis=1)

# Save processed test data to Excel
test.to_excel("2D-ABS", engine='odf')

# Combine test data with initial dataset
df = pd.concat([test, df_init])
df_absorbance = df.loc[:, freq_init].values

# Standardize the features
mapper = DataFrameMapper([(freq_init, StandardScaler())])
scaled_features = mapper.fit_transform(df)
scaled_features_df = pd.DataFrame(scaled_features, index=df.index, columns=freq_init)

# Apply PCA
pca = PCA(n_components=20)
principalComponents = pca.fit_transform(scaled_features_df)
principalDf = pd.DataFrame(data=principalComponents, columns=[f'principal component {i}' for i in range(1, 21)],
index=scaled_features_df.index)

# Get PCA loadings
loadings = pd.DataFrame(data=pca.components_.T, columns=[f'PCA{i}' for i in range(1, 21)], index=freq_init)

# Add target column to principal components DataFrame
finalDf = pd.concat([principalDf, df[['target']]], axis=1)

# Print the final DataFrame
pd.set_option("display.max_rows", 100)
print(finalDf)

# Calculate explained variance
exp_var_pca = pca.explained_variance_ratio_
cum_sum_eigenvalues = np.cumsum(exp_var_pca)

# Plot 3D PCA
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')
```

```
ax.set_xlabel('Principal Component 1', fontsize=15)
ax.set_ylabel('Principal Component 2', fontsize=15)
ax.set_zlabel('Principal Component 3', fontsize=15)
ax.set_title('3 component PCA', fontsize=20)
max_axis_size = 40
ax.set_xlim([-max_axis_size, max_axis_size])
ax.set_ylim([-max_axis_size, max_axis_size])
ax.set_zlim([-max_axis_size, max_axis_size])

# Define targets and colors for plotting
targets = ['MSSA control', 'MSSA 0min', 'MSSA 30 - 120 min', 'MSSA 240 - 360 min', 'MRSA control', 'MRSA 0min', 'MRSA 30 -
120 min', 'MRSA 240 - 360 min']
colors = ['rosybrown', 'lightcoral', 'indianred', 'firebrick', 'lightsteelblue', 'cyan', 'blue', 'navy']

# Plot each target group in different colors
for target, color in zip(targets, colors):
    indicesToKeep = finalDf['target'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1'],
               finalDf.loc[indicesToKeep, 'principal component 2'],
               finalDf.loc[indicesToKeep, 'principal component 3'],
               c=color, s=50, alpha=1)
ax.legend(targets)
ax.grid()
plt.show()

# Plot PCA loadings
fig, ax = plt.subplots(figsize=(8, 8))
ax.bar(loadings.index, loadings.iloc[:, 0], width=0.2, label='PC1', color='b')
ax.bar(loadings.index + 0.2, loadings.iloc[:, 1], width=0.2, label='PC2', color='r')
ax.bar(loadings.index + 0.4, loadings.iloc[:, 2], width=0.2, label='PC3', color='y')
ax.set_ylabel('Loading')
ax.set_xlabel('Frequency')
ax.legend(loc='best')
plt.tight_layout()
plt.show()

# Plot explained variance
plt.figure(figsize=(8, 8))
plt.bar(range(len(exp_var_pca)), exp_var_pca, alpha=0.5, align='center', label='Individual explained variance')
plt.step(range(len(cum_sum_eigenvalues)), cum_sum_eigenvalues, where='mid', label='Cumulative explained variance')
plt.ylabel('Explained variance ratio', fontsize=18)
plt.xlabel('Principal component index', fontsize=18)
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```

Direct Linear discriminant analysis on tertiary data with SG filter:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from scipy.signal import savgol_filter
import seaborn as sns

# Load data
df = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA.ods', index_col=0, engine='odf')
test_raw = pd.read_excel('~/Documents/Project/Clump/tertiary_nn.ods', index_col=0, engine='odf')

# Extract frequency columns from test data
freq_test = list(test_raw.columns[0:585].values)

# Apply Savitzky-Golay filter to test data
test = pd.DataFrame()
for sample in test_raw.index:
    d = test_raw.loc[sample]
    d_filtered = savgol_filter(d.iloc[0:585], 9, 2, deriv=2)  # Apply Savitzky-Golay filter
    d_filtered_series = pd.Series(d_filtered, index=freq_test, name=sample)
    test = pd.concat([test, d_filtered_series], axis=1)

test = test.transpose()

# Add target column to filtered test data
target_series = pd.Series(test_raw['target'].values.flatten(), index=test_raw.index, name='target')
test = pd.concat([test, target_series], axis=1)

train = df

# Prepare training and test sets
freq_train = list(train.columns[0:585].values)
x_test = test.loc[:, freq_test].values
x_train = train.loc[:, freq_train].values
y_test = np.ravel(test.loc[:,['target']].values)
y_train = np.ravel(train.loc[:,['target']].values)

# Standardize data
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.fit_transform(x_test)

# Perform Linear Discriminant Analysis (LDA)
lda = LinearDiscriminantAnalysis()
lda.fit(x_train_scaled, y_train)

# Transform data using LDA
x_train_lda = lda.fit_transform(x_train_scaled, y_train)
x_test_lda = lda.transform(x_test_scaled)

# Predictions and accuracy
y_pred = lda.predict(x_test_scaled)
accuracy = lda.score(x_test_scaled, y_test)
print("Accuracy: {:.2f}%".format(accuracy * 100))

# Display LDA-transformed data
x_train_df = pd.DataFrame(x_train_lda, columns=['LDA feature 1'])
y_train_df = pd.DataFrame(y_train, columns=['target'])
final_df = pd.concat([x_train_df, y_train_df], axis=1)
```

```
df_test = pd.concat([pd.DataFrame(lda.transform(x_test_scaled), columns=['LDA1']),
                     pd.DataFrame(y_pred, columns=['target'])], axis=1)
print(df_test)

# Plot LDA 1D projection
targets = ['MSSA', 'MRSA']
colors = ['firebrick', 'navy']
targets2 = ['MSSA', 'MRSA']
colors2 = ['lightcoral', 'cyan']

for target, color in zip(targets, colors):
    indicesToKeep = final_df['target'] == target
    plt.scatter(final_df.loc[indicesToKeep, 'LDA feature 1'], [0] * final_df.loc[indicesToKeep, 'LDA feature 1'].shape[0],
                c=color, s=50)
    for target, color in zip(targets2, colors2):
        indicesToKeep = df_test['target'] == target
        plt.scatter(df_test.loc[indicesToKeep, 'LDA1'], [0] * df_test.loc[indicesToKeep, 'LDA1'].shape[0],
                    c=color, s=50)

plt.legend(targets)
plt.xlabel('LDA projection', fontsize=15)
plt.ylabel('', fontsize=15)
plt.suptitle('LDA 1D projection', fontsize=20)
plt.xlim([-10, 10])
plt.ylim([-1, 1])
plt.show()

# Display test data with predictions
oracle = pd.concat([pd.DataFrame(x_test), pd.Series(y_pred), pd.Series(y_test)], axis=1)
print(oracle)

# Compute confusion matrix
conf_m = confusion_matrix(y_test, y_pred)

# Display accuracy
print(f'Accuracy: {accuracy:.2f}')

# Display confusion matrix as heatmap
labels = ["MRSA", "MSSA"]
plt.figure(figsize=(6, 6))
sns.set(font_scale=2)
sns.heatmap(conf_m, annot=True, fmt="d", cmap="Blues", cbar=False, square=True,
            xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

PCA filtered Linear discriminant analysis on tertiary data with SG filter:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn_pandas import DataFrameMapper
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix
from scipy.signal import savgol_filter
import seaborn as sns

# Load initial data
df_init = pd.read_excel('~/Documents/Project/Clump/MSSA.vs.MRSA.ods', index_col=0, engine='odf')
freq_init = list(df_init.columns[0:585].values)

# Load test data
test_raw = pd.read_excel('~/Documents/Project/Clump/tertiary_nn.ods', index_col=0, engine='odf')
freq_test = list(test_raw.columns[0:585].values)
np.set_printoptions(precision=2)  # For compact display.

# Apply Savitzky-Golay filter and concatenate results
test = pd.DataFrame()
for sample in test_raw.index:
    d = test_raw.loc[sample]
    d_filtered = savgol_filter(d.iloc[0:585], 9, 2, deriv=2)  # Apply Savitzky-Golay filter
    d_filtered_series = pd.Series(d_filtered, index=freq_test, name=sample)
    test = pd.concat([test, d_filtered_series], axis=1)

test = test.transpose()

# Add target column to filtered test data
target_series = pd.Series(test_raw['target'].values.flatten(), index=test_raw.index, name='target')
test = pd.concat([test, target_series], axis=1)

# Combine initial and filtered test data
df = pd.concat([test, df_init])
df_absorbance = df.loc[:, freq_init].values

# Standardize data using DataFrameMapper
mapper = DataFrameMapper([(freq_init, StandardScaler())])
scaled_features = mapper.fit_transform(df)
scaled_features_df = pd.DataFrame(scaled_features, index=df.index, columns=freq_init)

# Perform PCA
pca = PCA(n_components=20)
principalComponents = pca.fit_transform(scaled_features_df)
principalDf = pd.DataFrame(data=principalComponents,
                           columns=['principal component 1', 'principal component 2', 'principal component 3',
                                    '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20'],
                           index=scaled_features_df.index)
finalDf = pd.concat([principalDf, df[['target']]], axis=1)

# Split data into test and train
test = finalDf.iloc[:35, :]
train = finalDf.iloc[35:, :]
freq_test = list(test.columns[0:3].values)
freq_train = list(train.columns[0:3].values)

x_test = test.loc[:, freq_test].values
x_train = train.loc[:, freq_train].values
y_test = np.ravel(test.loc[:, ['target']].values)
y_train = np.ravel(train.loc[:, ['target']].values)

# Perform Linear Discriminant Analysis (LDA)
lda = LinearDiscriminantAnalysis()
lda.fit(x_train, y_train)
```

```
# Transform data using LDA
x_train_lda = lda.fit_transform(x_train, y_train)
target_names = ["MRSA", "MSSA"]

# Predictions and accuracy
y_pred = lda.predict(x_test)
accuracy = lda.score(x_test, y_test)
print("Accuracy: {:.2f}%".format(accuracy * 100))

# Display LDA-transformed data
x_train_df = pd.DataFrame(x_train_lda, columns=['LDA feature 1'])
y_train_df = pd.DataFrame(y_train, columns=['target'])
final_df = pd.concat([x_train_df, y_train_df], axis=1)
df_test = pd.concat([pd.DataFrame(lda.transform(x_test), columns=['LDA1']),
                     pd.DataFrame(y_pred, columns=['target'])], axis=1)
print(df_test)
print(final_df)

# Plot LDA 1D projection
targets = ['MSSA', 'MRSA']
targets2 = ['MSSA', 'MRSA']
colors = ['firebrick', 'navy']
colors2 = ['lightcoral', 'cyan']

for target, color in zip(targets, colors):
    indicesToKeep = final_df['target'] == target
    plt.scatter(final_df.loc[indicesToKeep, 'LDA feature 1'], [0] * final_df.loc[indicesToKeep, 'LDA feature 1'].shape[0],
                c=color, s=50)
    for target, color in zip(targets2, colors2):
        indicesToKeep = df_test['target'] == target
        plt.scatter(df_test.loc[indicesToKeep, 'LDA1'], [0] * df_test.loc[indicesToKeep, 'LDA1'].shape[0],
                    c=color, s=50)

plt.legend(targets)
plt.tick_params(axis='both', which='major', labelsize=14)
plt.xlabel('LDA projection', fontsize=18)
plt.ylabel('', fontsize=18)
plt.suptitle('LDA 1D projection', fontsize=20)
plt.xlim([-10, 10])
plt.ylim([-1, 1])
plt.show()

# Display test data with predictions
oracle = pd.concat([pd.DataFrame(x_test), pd.Series(y_pred), pd.Series(y_test)], axis=1)
print(oracle)

# Compute confusion matrix
conf_m = confusion_matrix(y_test, y_pred, labels=["MRSA", "MSSA"])

# Display accuracy and confusion matrix as heatmap
print(f'Accuracy: {accuracy:.2f}')
labels = ["MRSA", "MSSA"]
plt.figure(figsize=(6, 6))
sns.set(font_scale=2)
sns.heatmap(conf_m, annot=True, fmt="d", cmap="Blues", cbar=False, square=True,
            xticklabels=labels, yticklabels=labels)
plt.tick_params(axis='both', which='major')
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```