COMP6208 Group Project Data Exploration Report

Identifying Trends in Popular Music from 1970 to 2019 Using Advanced Machine Learning Techniques

Rohan Bungre - rsb1g17 - 29466423

Carl Richardson - cr2g16 - 28544285

Charlie Steptoe - cs5n17 - 29801389

Christian Hall - ch4g17 - 29340942

A report presented for the module of
Advanced Machine Learning

Department of Electronics and Computer Science
University of Southampton
29/03/2021

1 Introduction

This report will cover the data exploration techniques used to identify trends in popular music from 1970 to 2019. The mathematical study of musical metrics yields a wealth of data on which to perform an immense number of possible regression, classification and data analysis tasks. These mathematical techniques can be used to extract unique features that underpin every song. Exploring these features can help identify why certain songs are popular in certain eras and detect any trends as time moves on. To do this a dataset was created from the Spotify database, which includes popular songs for each year and the extracted mathematical music features, defined in Appendix A.

2 Feature Trends

This section explores how the features changed between 1970 and 2019. Tempo, mode, loudness and time signature were all normalised to between 0 and 1, the rest of the features already had this structure. Most of the features had trends that changed over the decades, whether it was generally decreasing or increasing or a cyclical pattern. Tempo and time signature were exceptions to this, offering little in-sight as there was very little change or no pattern that we could discern. We have chosen to display the discuss the most interesting features in this section. The full feature trends can be seen in Appendix F.

Valence is the emotion of a song, where a higher value indicates positive emotions and a lower value indicates negative emotions. The valence of songs has been decreasing significantly since the 1970s by ~ 0.20, furthermore there are large fluctuations with before early 1980s and early 2000s having a greater level of valence than the others. This could be due in-part to the positivity associated with a new millennium, and the negativity since the early 2000s could be due to the culture of negativity that we are currently in.

Figure 1: To ensure that the graph is easily view able, the features have been adjusted so their smallest value is zero, the following values are the features' true smallest value. Average valence adjustment is: 0.47, standard deviation valence adjustment is: 0.21, average danceability adjustment is: 0.48, standard deviation danceability adjustment is: 0.13.

Acousticness is the level of confidence in whether the song has used primarily acoustic instruments. A song with an acousticness of 1, suggests that the song might contain violins, guitars or pianos, with minimal post-processing on them. The average acousticness of a song has reduced from 0.35 to 0.19, with the lowest about 0.1 between 2005 and 2010. The standard deviation of acoustiness, floats around 0.27 and 0.15, generally getting lower throughout the years. This suggests that the number of songs using electronic sounds or post-processing has increased throughout the decades, and that hit tracks are more consistently using a similar level of it.

Danceability is the suitability of a track for dancing, with 1 being a high level of suitability and 0 being a low level of suitability. The danceability of tracks has been increasing throughout the years with starting with ~ 0.48 in the 1970s and increasing to ~ 0.65 by the late 2010s. However this increase was not linear, there were spikes in the early 1980s, early 2000s and late 2010s. This would suggest that there is a cyclical nature to the hit tracks, and danceability could be a feature that people care about but can get bored of.

Speechniess is the confidence level and type of speech in a track. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent instrumental music and other non-speech-like tracks. The amount of singing/talking in tracks has increased since the 1970s, this could be caused by the increased presence of

rap and hip hop in popular culture. The early 2000s had a increased level of speechiness, which might have been due to the popularity of rap artists, like Eminem and Jay-Z.

3 Principal Component Analysis & Linear Separability

An intuitive method by which to analyse the data distributions through the decades is to calculate the directions of maximum variance of the features in the dataset. By finding these principal components, it is possible to project subsets of each decade's data onto these vectors and investigate them mathematically to appreciate how each direction contributes to linear separability. An 'out-of-the-box' PCA implementation was applied to the well-distributed numeric features of the dataset (Appendix B): 'acousticness', 'danceability','energy', 'instrumentalness', 'liveness', 'speechiness', 'tempo' & 'valence'. The principal components have been shown in Figure 2 as a color map for interpretability.

Figure 2 demonstrates that the directions of maximum variance are unaligned with the features chosen. The magnitudes of the values in the first two principal components are of particular note. In principal component 0, acousticness, energy and loudness values contribute most to the direction, aligning well with trends identified when understanding the variation over time. Furthermore, principal component 1 has a very strong value present under danceability, another feature that was identified as varying a significant amount over time.

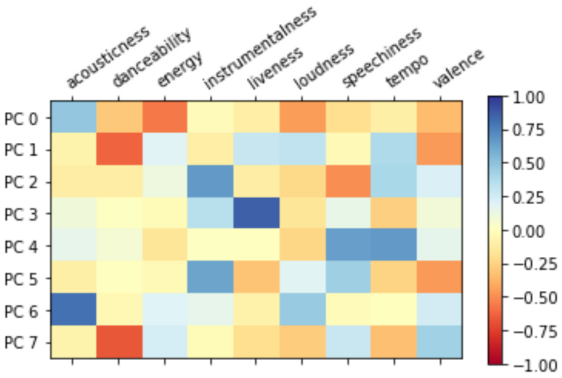


Figure 2: Principal Components Colour Map

Following the initial PCA, projecting the data of each decade to the components, assuming a normal distribution holds in each subset, we can fit a distribution to each subset, obtaining a new mean and standard deviation for each. First, consider this in the one-dimensional case, corresponding to the mapping onto each principal component individually. Figure 3 shows the normal distributions by decade, above their respective Bhattacharyya distance matrix, for the set of principal components where mean pairwise Bhattacharyya distance was maximised.

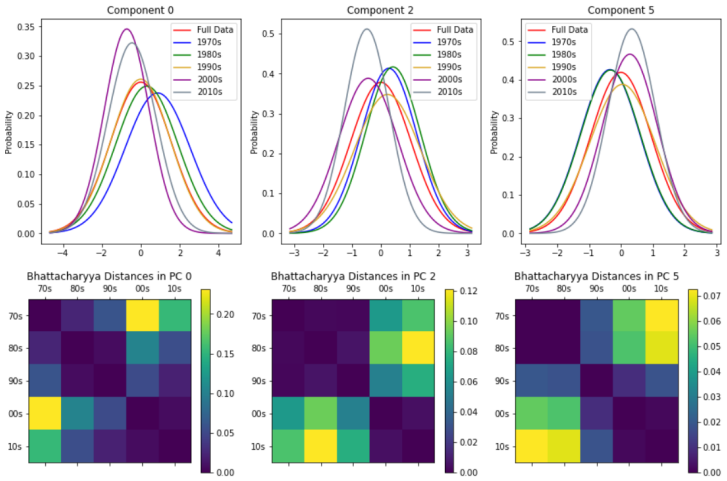


Figure 3: Projected Distributions & D_B Distance Matrices

From this figure, it is evident that some principal components' directions are more linearly separable between pairs of decade classes than others. For example, principal components 2 and 5 clearly increase separability more for decade pairs including the 2010s, whereas component 0 more successfully increases separability between the 2000s and other decades. An additional finding is that separability between the 70s and 80s, as well as the 00s and 10s decades tend to be significantly lower in all component projections.

Finally, it is possible to map each decade's dataset to a transformed 'principal' space, by projecting the set onto all principal components simultaneously. As, after using PCA, it is possible to specify the resultant multivariate normal distributions' covariance matrices are diagonal, these covariance matrices are calculated and the metric of Mahalanobis distance is applied to each decade pair, resulting in the D_M distance matrix of Appendix C. As hoped, through the use of these principal components, there is clear linear separability between pairs of decades, improving as the time separation increases.

4 Feature Correlation

It is often useful to investigate correlation between the features of a dataset and the target values. This helps to identify strongly correlated features, allowing for feature reduction; find features to help in predicting the target value and gain qualitative insight into the structure of the dataset. A correlation matrix can be used to show the Pearson Correlation Coefficients between each input feature of the dataset. The correlation coefficient has values between -1 to 1. A value closer to 0 implies weaker correlation (exact 0 implying no correlation). A value closer to 1 implies stronger positive correlation. A value closer to -1 implies stronger negative correlation.

Figure 4 shows the correlation matrix with a reduced feature set that only identify relatively strong correlations. The full correlation matrix including all features can be seen in Appendix D. It can be seen that year and decade features have extremely similar correlations with respect to the other music features. The loudness feature had the strongest positive correlation of 0.54. This shows that the loudness of songs increases as the years move on. The speechiness feature also has a positive correlation of 0.28, implying that there is more spoken word as the years move on. The acousticness feature shows a negative correlation of -0.27 as the years move on. These features would all be useful to help classify which year and decade a song is from.

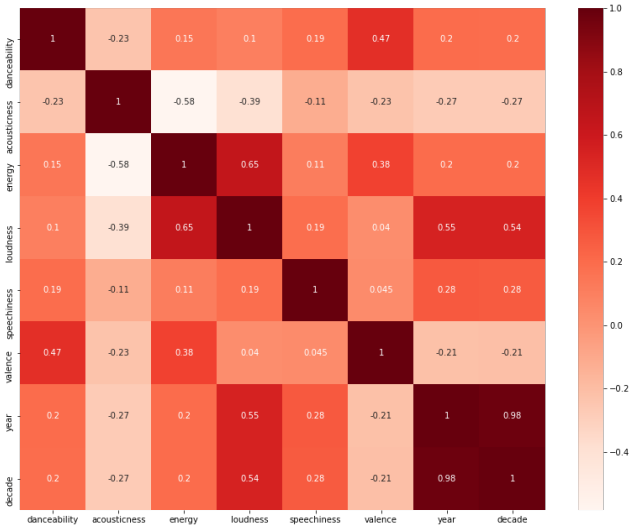


Figure 4: Reduced Feature Correlation Matrix

Comparing the correlation between the music features allows for an interesting understanding of their relationships to one another. There is a strong positive correlation between danceability and valance; and energy and loudness which makes sense because these features are similar in definition. There is a strong negative correlation between energy and acousticness; loudness and acousticness; and danceability and acousticness which also makes sense as these features are opposite in definition. Doing this analysis increases confidence that dataset includes accurate music feature definitions.

5 Feature Selection

To gain an insight into which input variables play an important role for decade classification, a decision tree was used to partition the input space. Each branch of the tree represents a decision boundary in the input space which optimally separates the data with respect to the Gini index (a measure of a nodes impurity). Its value ranges from zero, when the node contains only one class, to $1/C$, when the classes are evenly distributed within the node (C represents the number of classes within the node). To decide how to split a node, a weighted sum of the Gini index, of the two sub-nodes, is treated as the loss function to be minimised with respect to each feature and a corresponding threshold. The combination of feature and threshold which minimises the loss function is used to partition the input space. This is a recursive procedure which terminates when all the end nodes (leaves) are pure or some early stopping condition is met, such as the maximum tree depth.

Before running the decision tree, some minor preprocessing steps were applied to the data set. All features which did not quantify attributes of the track were removed (i.e name, release date, popularity, year, etc.). This was an important step as the goal is to classify the decade of a track based on the musical and duration descriptors. A decade column was also added to provide the classification labels. For decision trees, the coding scheme of the labels does not matter, so integer labels were assigned. For the purpose of data exploration, as the layers of the tree are traversed, the decision boundaries

appear in descending order of importance, when starting at the root node. The decision boundary at the root node, is the optimal boundary for maximising the purity of the sub-nodes, when acting on all the data. For this reason, when identifying the important features, only the first two layers of the tree were analysed.

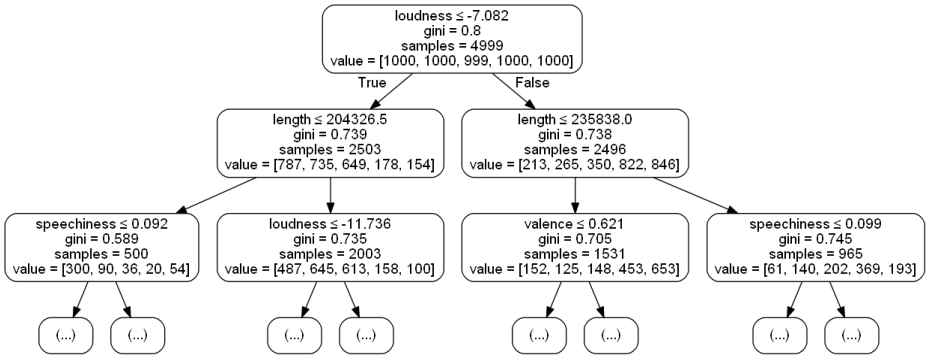


Figure 5: First 2 Layers of Decision Tree

Figure 5 shows the resulting decision tree. The value vector, from left to right, represents the count for 70's through to 10's, in ascending order. The rule at the root of the tree partitioned tracks at the -7.082 dB loudness boundary. This simple rule shows that if a track was louder than this value, it had a 67% probability of being from the 00's and 10's and if it was quieter, it had an 87% probability of being from one of the three other decades. Thus, loudness was an important feature for separating the 00's and 10's from the other classes. The second layer shone light on some further simple rules. The purest node had a Gini index of 0.589 and occurred when a track was quieter than -7.082 dB and had a shorter duration than 204 s. Under these conditions, there was a 60% chance the track was from the 70's.

The decision tree was also used to see which features were least important for classification of the decades. Table 1 shows the numbered set of features (mapping of feature names to numbers is included in Appendix E) and the first layer the feature was used as a decision boundary. For example, feature 7 corresponds to loudness and was first used at the root of the tree (depth 1). This shows features 8 and 11, which correspond to mode and time signature, were particularly insignificant when trying to classify the decade of the track.

Feature Number	0	1	2	3	4	5	6	7	8	9	10	11	12
First Depth	2	4	2	2	4	5	4	1	6	2	3	6	2

Table 1: Features and Depth First Used as a Decision Boundary

As the goal of the learning machine is to classify this data set by decade, removing mode and time signature may help to eliminate some of the spurious rules the machine might find, with minimal impact on its classification performance. This would help to provide better generalisation and interpretability of the machine.

6 Conclusion

The data appears to be separable and the patterns emerging in the decision tree appear to align with trend analysis and cultural shift. Furthermore, the principal component analysis demonstrates effectiveness for increasing linear separability by augmenting the dataset. Finally, feature correlation highlights that a number of the features are strongly correlated. In conclusion, the data exploration techniques covered prove that when using a select combination of features, our data is suitable for machine learning classification tasks.

Appendices

Appendix A Feature Definitions

Danceability: describes the suitability of a track for dancing. This is based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

Energy: a measure from 0.0 to 1.0, and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.

Key: the key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on.

Loudness: the overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing the relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.

Mode: indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

Speechiness: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent instrumental music and other non-speech-like tracks.

Acousticness: a confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence that the track is acoustic.

Instrumentalness: predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.

Liveness: detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

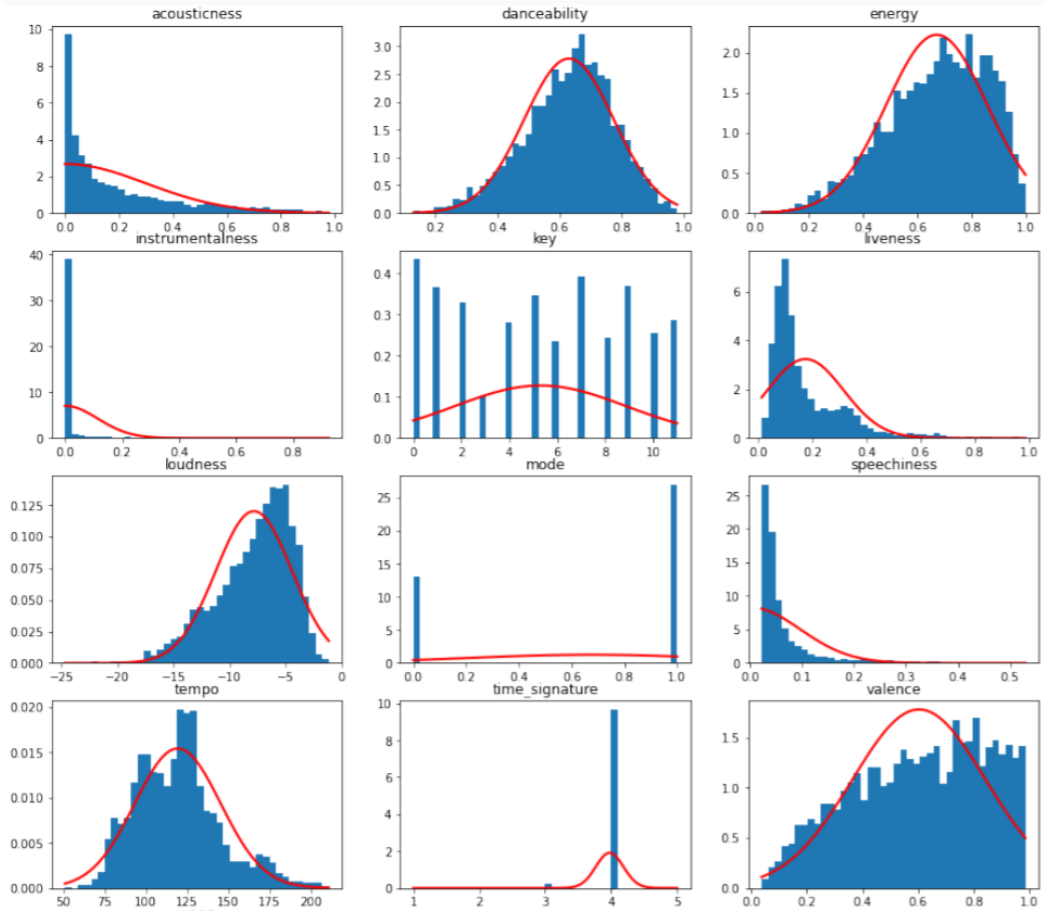
Valence: a measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (for example happy, cheerful, euphoric), while tracks with low valence sound more negative (for example sad, depressed, angry).

Tempo: the overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece, and derives directly from the average beat duration.

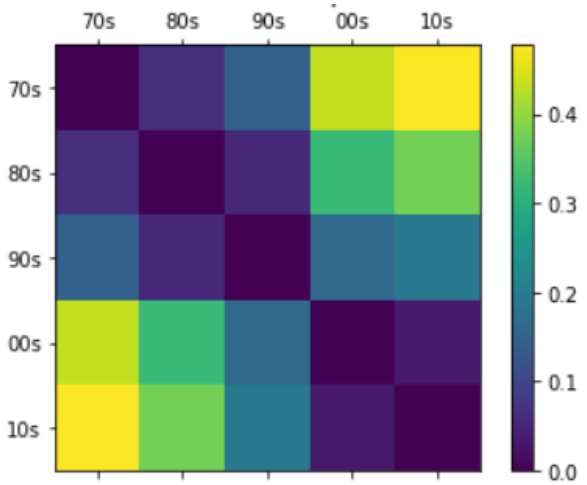
Duration_ms: the duration of the track in milliseconds.

Time_Signature: An estimated overall time signature of a track. The time signature (metre) is a notational convention to specify how many beats are in each bar (or measure).

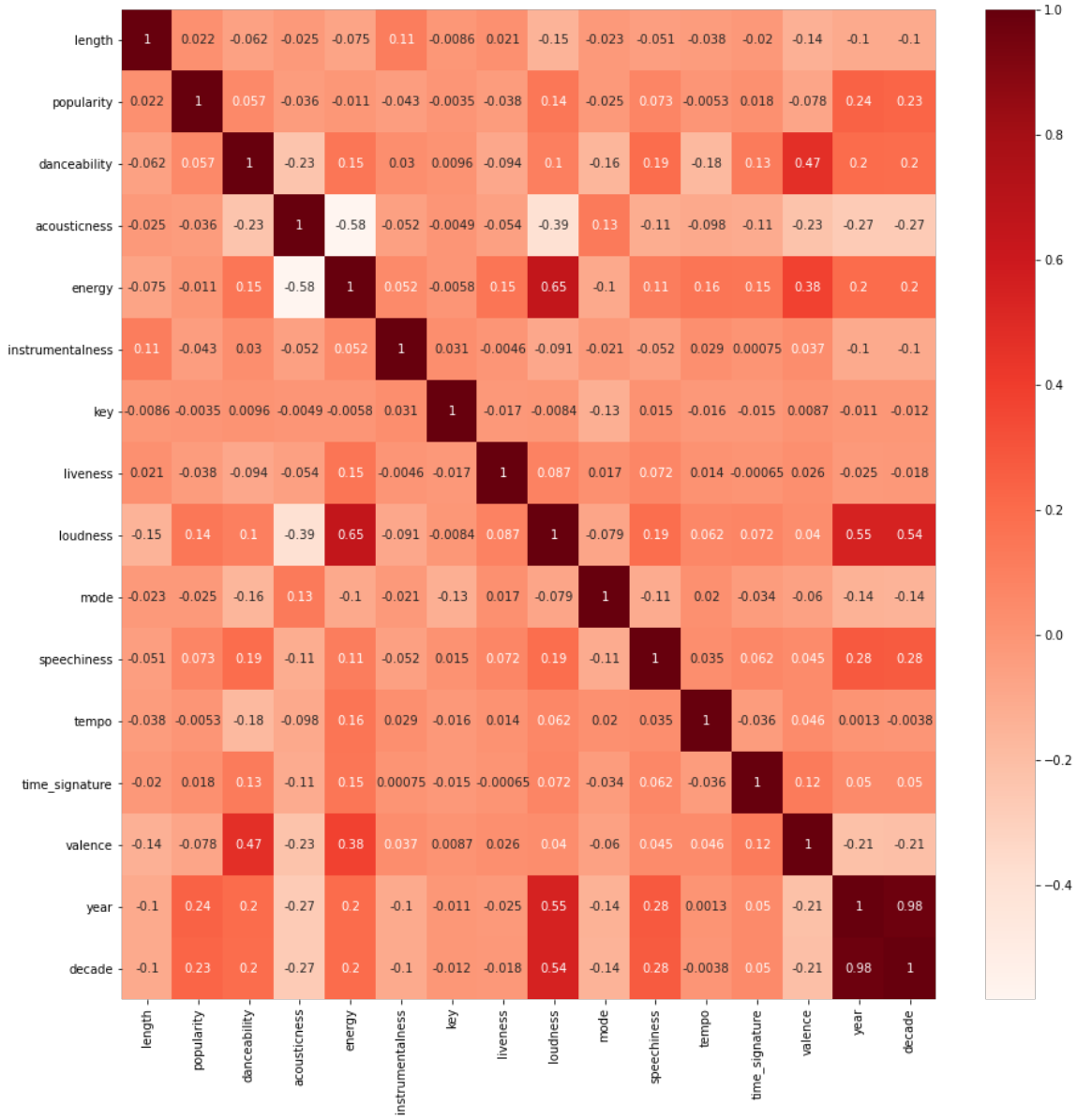
Appendix B Truncated Normal Distributions of Features



Appendix C PCA Mahalanobis Distance Matrix



Appendix D Full Correlation Matrix



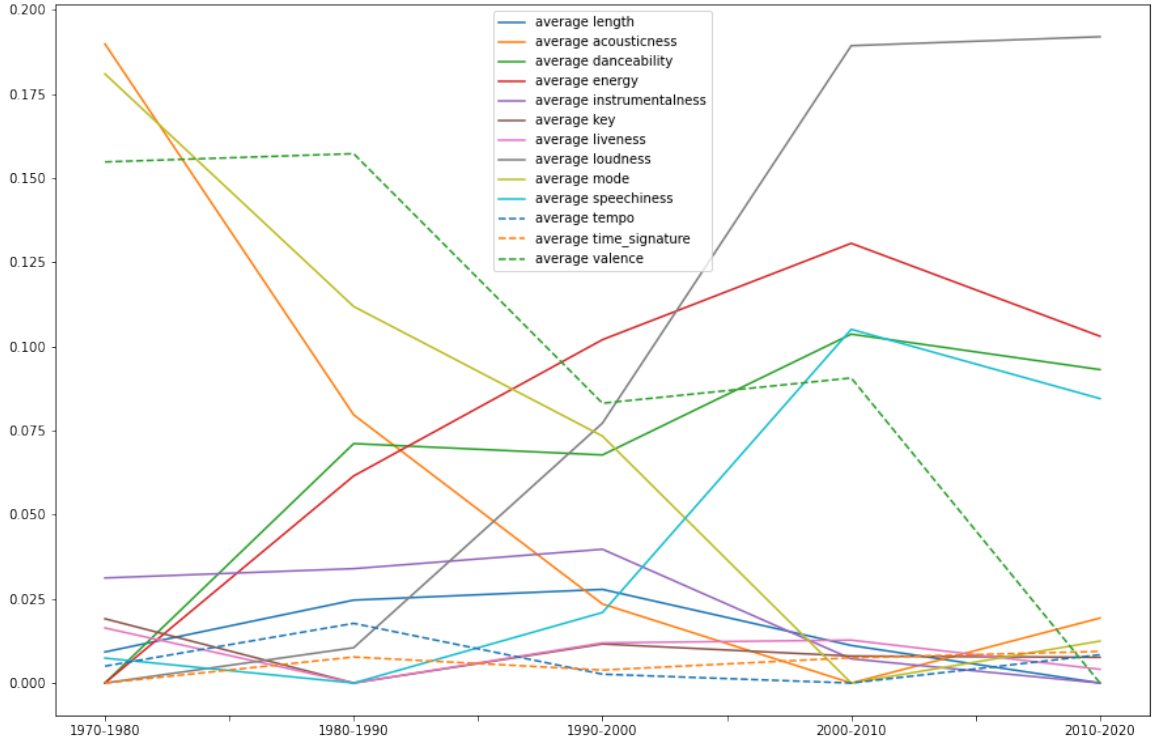
Appendix E Feature Number to Feature Name Mapping

Feature Number	Feature Name
0	Length
1	Acousticness
2	Danceability
3	Energy
4	Instrumentalness
5	Key
6	Liveness
7	Loudness
8	Mode
9	Speechiness
10	Tempo
11	Time Signature
12	Valence

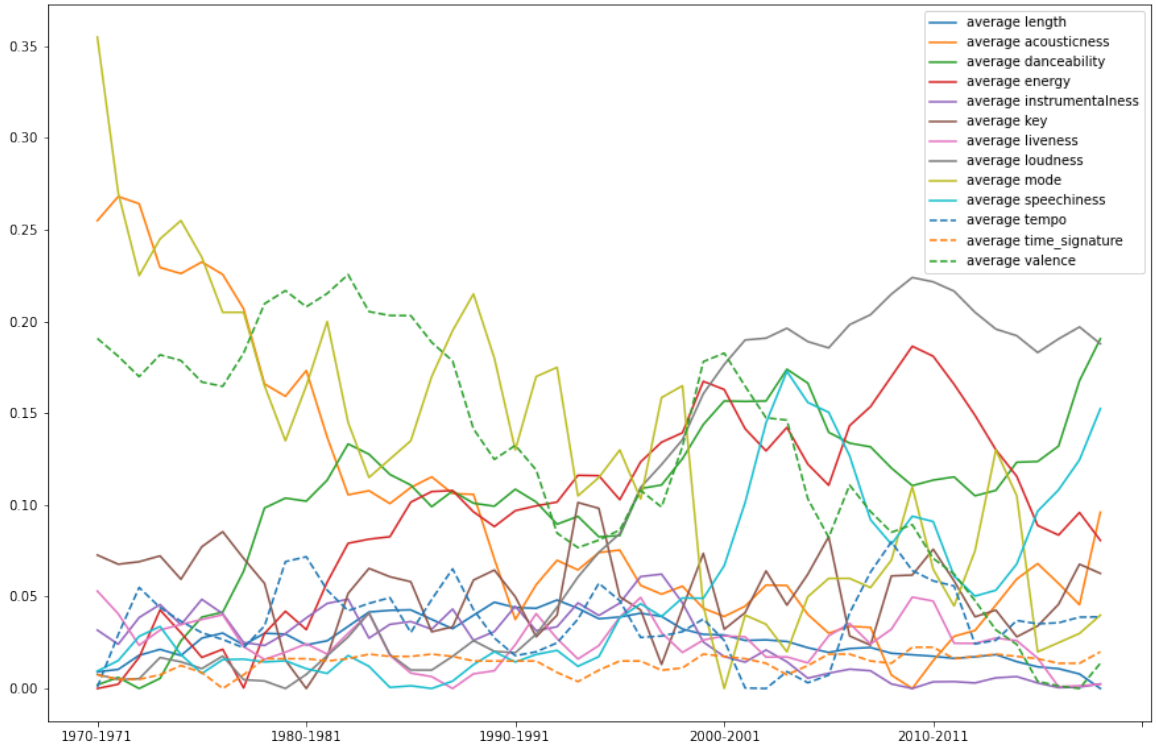
Appendix F Feature Trends

Note: The following graphs have been normalised with their minimum value set to zero.

Appendix F.a Feature Trends Decades



Appendix F.b Feature Trends Years



COMP6208 Group Project Machine Learning Report

Identifying the Release Decade of Popular Music from 1970 to 2019
Using Advanced Machine Learning Techniques

Rohan Bungre - rsb1g17 - 29466423

Carl Richardson - cr2g16 - 28544285

Charlie Steptoe - cs5n17 - 29801389

Christian Hall - ch4g17 - 29340942

A report presented for the module of
Advanced Machine Learning

Department of Electronics and Computer Science
University of Southampton
07/05/2021

1 Introduction

This report will cover the advanced machine learning techniques used to predict the release decade of popular music from 1970 to 2019. The idea is that music from different decades would have distinct feature distributions that a learning machine could identify. Using analysis from the data exploration report, a range of machine learning techniques were applied to the dataset, identifying which initial models performed the best. The best models were further analysed and their hyper-parameters tuned to find an optimal model which allowed for the most accurate decade prediction. To avoid data-leakage, all methods followed a process of having a separate test, train and validation set.

2 Machine Learning Techniques

2.1 Ensemble Methods

Ensemble methods are a machine learning technique where multiple machines called weak learners are trained to solve the same problem and combined to get better results. These weak learners can be combined to obtain a final machine that is often accurate and robust. This section covers two methods of ensembling- bagging and boosting. Scikit-Learn provides a number of models to use for this process. The first step was to perform a test-train split on the dataset. This was kept at 80% for training and 20% for testing. The data was then scaled using a standard scaler fit on the training data. For each of these techniques the default parameters were used to gain an initial insight into each method's performance for predicting a song's decade.

Model	Initial Accuracy
Random Forest	0.578
Extra Trees	0.544
AdaBoost	0.499
Gradient Boosting	0.585
XGBoost	0.58

Table 1: Initial Accuracy

From the initial results, the Gradient Boosting and XGBoost algorithms performed best with their default parameters. The accuracies were still relatively low, being around 58%. Improvements could be made by tuning the hyper-parameters. It is important to note that these accuracies come from having all features along with the generated PCA data. All accuracies decreased when a reduced feature set was used.

2.1.1 Gradient Boosting

Following preliminary tests, Gradient Boosting achieved an accuracy of 58.5%. Using the Scikit-learn, GridSearchCV function, the model's parameters would be fine-tuned to increase its accuracy. The main parameters to optimise within the Gradient Boosting model, were the 'n_estimators', 'max_depth', 'min_samples_split', 'min_samples_leaf' and 'max_features'. Each parameter was varied and the average accuracy across 5-folds was taken leaving the best parameter. This was then used in the next grid search to find the optimal new parameter. This process was repeated until an optimal accuracy was found.

Unfortunately this method of tuning the hyper-parameters only increased accuracy by 0.2% to 58.7%. This involved decreasing the 'n_estimators' parameter. The default parameters were found to be good for this problem. Figure 1 shows the confusion matrix for this model. It is clear to see that neighbouring decades are mistaken for each other, so perhaps the decade labels needed to be more granular.

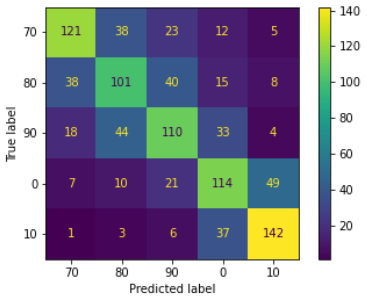


Figure 1: GB Confusion Matrix

2.1.2 XGBoost

Following the preliminary tests, XGBoost was shown to have an accuracy of 58%. In an attempt to achieve a greater accuracy, the hyper parameters and data set were tuned.

To combat the random element of the test training split method the results were averaged over a selection of static test training splits. In total 5 were used for the hyper parameter tuning and 64 in the final "showdown" phase of the testing. The testing was completed on a normalised version of the data set. To normalise the data, all the data was scaled between 0 and 1 as this achieved better results than using the standard scalar method available from Scikit.

Model	Norm. Dataset Acc.%	PCA Dataset	Combined Dataset
Untuned Model	56.760	50.340	57.200
Initial Tuning	58.580	49.420	58.800
Improved Tuning	58.240	50.060	58.519
3 sig. fig. Tuning	58.160	50.240	58.600

Table 2: XGBoost Model Comparison

For each hyper-parameter, an appropriate range of values were tested and an accuracy value measured. This data was graphed for each hyper-parameter and the best performing value was taken. This model is called the initial tuning model. After the initial tuning model was determined, further optimisation took place. For each hyper-parameter a tight range was tested around the initial tuning model's parameters. These new parameter values were stored in the improved metric model. Finally, the parameters from the improved metric model were reduced to 3 significant figures in case these parameters were over fitting, this was called the improved metric to 3 significant figure tuning model.

In conclusion, an accuracy of 58% is not unreasonable, given the gradual changes between decades, as music does not abruptly change every 10 years, there is a slow change over each year. The hyper parameter tuning improved accuracy, however the margin was only 2%, which points to the effectiveness of the default values Scikit offers. Given more time, a data set could have been generated that gave better classification possibilities, for example, a song is from the the late 90s or early 00s. This classification would allow for a higher accuracy in this case and more useful classifications as well.

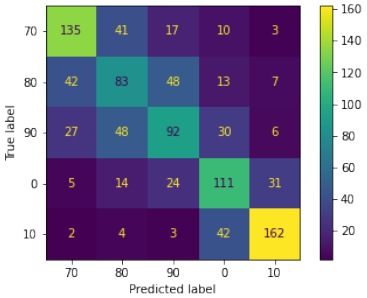


Figure 2: XGB Confusion Matrix

2.2 Support Vector Machines

The SVM was one learning machine considered for the task of predicting the decade a track was released. Based on evidence provided in the data exploration report, the data set was trimmed to remove the features 'time signature', 'mode' and 'key' as these were demonstrated to be uninformative. The data was split into training, testing and validating sets using a 70/20/10 split, then four different SVM kernel functions were evaluated and tuned in order to find the optimal model for this classification problem.

As shown by figure 3 the polynomial kernel had an optimal order of three; higher order functions showed significant over-fitting on the validation set. Furthermore, the parameter C, used to penalise the slack terms, was varied between 2^{-5} and 2^{5} for each kernel, where the optimal polynomial order was used. As can be seen, the radial basis function with a penalty of two provided the optimal performance. With the exception of the linear kernel, small penalty terms on the slack variables did not generalise well to the unseen data; this suggests that allowing the slack variables too much freedom did not help to understand the underlying trends of the decades. However, as shown by the drop off in performance of the sigmoid, polynomial and radial basis kernels, penalising the slack variables too much prevented the hyper-planes from accommodating each other, in order to maximise overall performance, and instead encouraged them to maximise their one versus all margin in isolation.

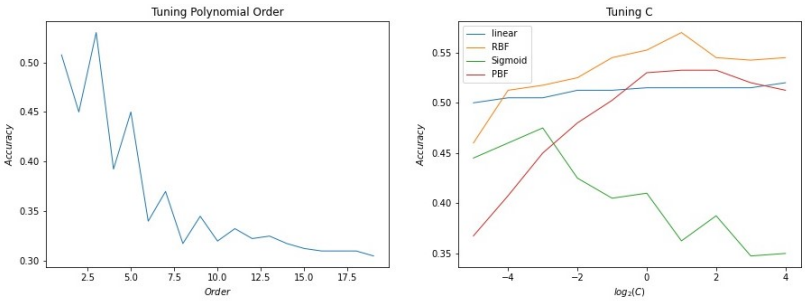


Figure 3: Tuning hyper-parameters

After training the SVM with the radial basis kernel and a penalty term of two, the predictive performance of the model was tested. Figure 4 shows the confusion matrix, where each row represents the distribution of predictions for tracks from a given decade. On average, the machine correctly classified each decade with an accuracy of 55.6% whilst the variance of correct classification across the decades was 5.08%. This shows the classification accuracy of each decade was fairly similar,

supporting the earlier suggestion that the penalty was optimised such that the set of margins prioritised overall classification performance over their individual one versus all. Furthermore, on average, each track was classified to within one decade of its true release 82% of the time. This suggests the model captured the underlying features of the data well as trends in music continuously change throughout and across decades as opposed to on discrete boundaries, as modelled by this classification task.

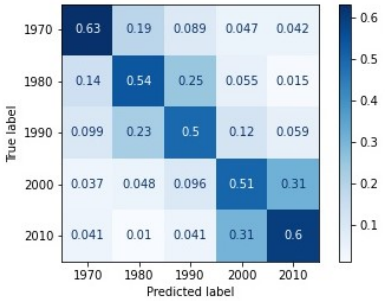


Figure 4: SVM confusion matrix

In conclusion, the SVM achieved an average classification accuracy of 55.6% illustrating that even after transforming into the non-linear radial basis feature space, finding linearly separable planes between the five decades wasn't possible. This low classification accuracy appears to be caused by the framing of the task rather than poor performance of the SVM. The evidence suggests the SVM captured the underlying musical trends well, as supported by the 82% of tracks correctly classified to within one decade. Since musical trends change continuously over time, rather than on decade boundaries, this result carries significant weight in terms of comparing how well the model understood the data.

2.3 Clustering

To keep the amount of time spent on hyperparameter tuning to a minimum, only clustering algorithms available from sklearn were tested. These included: Affinity Propagation Clustering, K-Means, Mean-Shift, Hierarchical Clustering, DBSCAN, OPTICS, and Birch. Scikit also offers Spectral Clustering, however the time to generate results was approximately 1 hour with no perceivable advantage, so it was dropped from the testing process. To evaluate the techniques, the accuracy and the number of clusters were used primarily. Other metrics were available, but were likely less relevant for this problem.

Only Mean-Shift and K-Means were able to estimate the correct number of clusters while using the PCA data set, and given that one of K-Means' hyper parameters is the number of clusters, it is unremarkable that it calculated this. The accuracy of the K-Means model was approximately 5%, this was due to it crushing inputs into a single cluster and not clustering the remaining items, resulting in a worse accuracy than Mean-Shift or random assignment. Mean-Shift achieved an accuracy of 20%, without tuning any hyper parameters. Observing the confusion matrix of this algorithm highlighted that all input data had been group into a single cluster.

Mean-Shift was decided to be investigated further, as tuning may have alleviated these initial problems. The first step was to vary the bandwidth to work with different data sets rather than just the PCA data set. However, this did not improve the model, and tuning hyperparameters seemed to be fruitless.

Clustering has its advantages, with testing being quick to implement and quick to return results averaging less than a minute for most techniques. However, the unsupervised nature of the learning process caused most of the techniques to incorrectly measure the number of clusters. The data set had some very clear trends throughout the years, as discussed in the previous report, however the variance of that "average" in each year and decade is extremely high, resulting in a high dimensional, noisy dataset that is difficult to cluster. Given more time, experimenting with techniques not offered by Scikit could have been useful, however, it was chosen to spend this time on more hopeful avenues.

2.4 Neural Networks

Investigation into the use of neural networks on the constructed dataset was divided into three main phases: Initial testing, optimised classification and regression. Initial tests were conducted using the Scikit-learn package's built-in MLP implementation. Having normalised the input features, testing a small number of hidden layer and loss function configurations yielded an early accuracy of 51.3%. Comparing this accuracy against other tested methods demonstrated the potential strength of using neural networks for decade classification. The full initial confusion matrix can be seen in Appendix A.

2.4.1 Decade Classification

To expand functionality, the more extensive PyTorch library was utilised to further specify network construction and decrease computation time. In particular, the ability to vary activation functions by layer and the inclusion of dropout assisted greatly in improving accuracy and generalisation capabilities. The classification network was tested using a collection of layer sizes and adjusting dropout layer probabilities. The optimal network layout was designed in an additive fashion, testing various forms of added layers to improve upon simpler networks. A subset of tested configurations is given below.

Network Configuration	Val. Acc.
$\rightarrow 100 \rightarrow 100 \rightarrow 100 \rightarrow$	49.9%
$\rightarrow 50 \rightarrow 50 \rightarrow Dropout(= 0.3) \rightarrow 50 \rightarrow$	53.3%
$\rightarrow 100 \rightarrow 100 \rightarrow Dropout(= 0.3) \rightarrow 100 \rightarrow$	55.6%
$\rightarrow 100 \rightarrow 100 \rightarrow Dropout(= 0.5) \rightarrow 100 \rightarrow$	53.2%

Through extensive fine-tuning of model parameters, it was found that the optimal model is as shown, with a dropout layer and hidden layer sizes of 100, producing an average correct classification rate of 55.6%.

In order to avoid any overfitting of the model, the loss on a validation set was calculated and plotted to ensure there was no requirement for early stopping. These plots can be observed in Appendix B. To visualise the final accuracy, a confusion matrix for data classification by decade of release was produced, which can be seen in Figure 5.

This confusion matrix has a similar appearance to the linear separability of songs by their respective decades; songs released in the '00s' and '10s' were more likely to be mistaken for each other and another cluster appears among the '70s', '80s' & '90s' decades. The highest accuracy was obtained for the extreme decades, '70s' and '10s'.

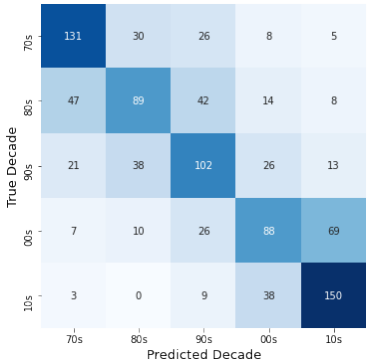


Figure 5: NN Confusion Matrix

2.4.2 Release Year Regression

When considering the date of release of songs in the dataset, it is important to consider that the year of release is more granular than decades and, as such, a basic accuracy measure considers those songs released at the beginning and end of a decade to be of similar semantic distance from the adjacent decades as those released in the middle of a decade. Accordingly, a regression model was also implemented with the target of predicting the year of release. A very similar procedure was followed for constructing the optimal network architecture as in the classification problem, resulting in the configuration ($\rightarrow 50 \rightarrow 50 \rightarrow Dropout(= 0.6) \rightarrow 50 \rightarrow$).

Having trained the network on release years standardised between 0 and 1, the average validation error reached was 6.9 years, corresponding to 78% of songs predicted as being within 10 years of their true release year. A plot of prediction by true year is provided in Appendix C, with mean predicted year and σ by year, showing two distinct features. As expected, a shallower gradient appeared at both extrema of the true years, with a much steeper gradient between the years 1990 and 2000. Additionally, the mean predictions become 'squashed' towards the average year of the dataset, potentially due to lack of training data in the surrounding years. This feature was a problem not foreseen in data exploration.

3 Conclusion

This report applied four machine learning methodologies to the music release date classification problem: support vector machines, ensemble learning, clustering & deep learning. Overall, a marked similarity was found in the optimised accuracies obtained by three os these techniques, reaching between 55-59% test set prediction accuracy. This consistency may indicate that only approximately 60% of songs in the constructed dataset are truly representative of their release decade, with an amount of noise and higher-frequency fluctuation expected of 'human-driven' data.

In future work, time could be spent creating a more representative classification model that encapsulates the fluid nature of music rather than trying to classify arbitrary boundaries (e.g. by decade). This could be achieved by using the year of release as the classification boundary, which would allow diffuse classifications, such as indicating "late 00s / early 10s", or further investigating regression techniques.

Appendices

Appendix A Additional Confusion Matrices

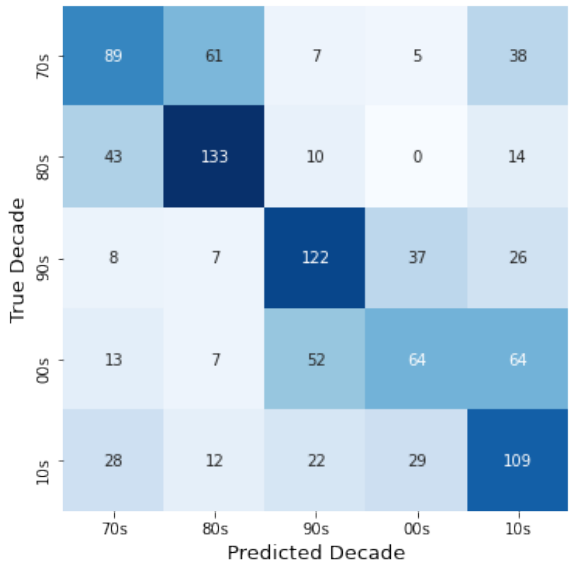


Figure 6: Initial Scikit-Learn MLP Confusion Matrix

Appendix B Neural Network Loss Plots

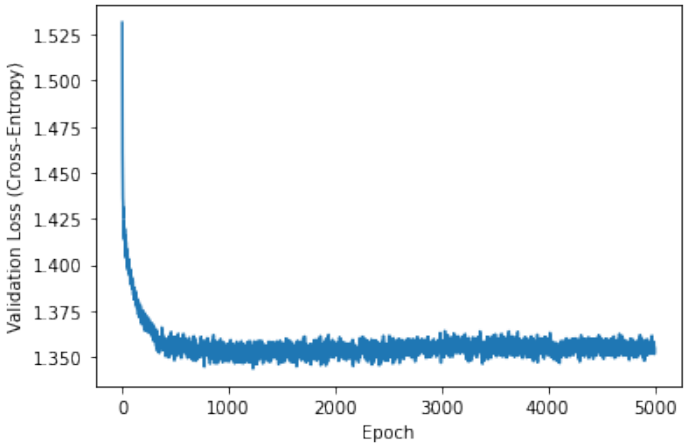


Figure 7: PyTorch Classification Cross-Entropy Loss Plot

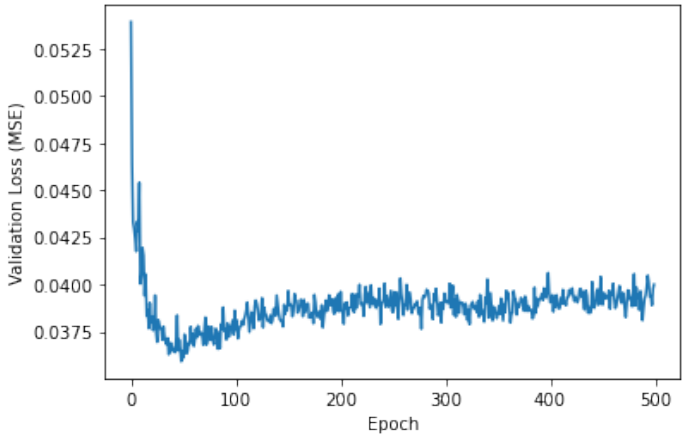


Figure 8: PyTorch Regression Mean-Squared-Error Loss Plot

Appendix C Neural Network Regression Results

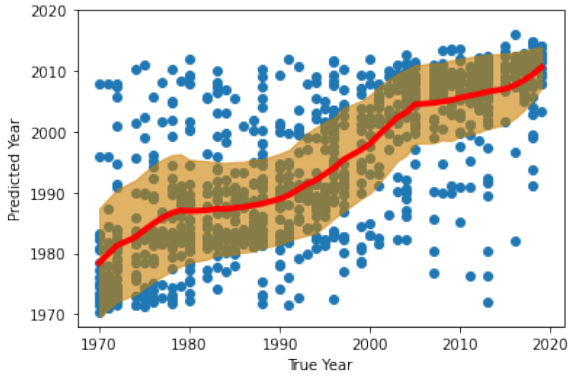


Figure 9: Neural Network Regression Results (Mean prediction in red with $\pm 1\sigma$ highlighted)

REPRODUCIBILITY CHALLENGE: ADJUSTABLE REAL-TIME STYLE TRANSFER

Rohan Bungre, Carl Richardson, Charlie Steptoe & Alexander Thomas

ABSTRACT

'Adjustable Real-Time Style Transfer' (ICLR 2020) addresses the inclusion of a secondary conditioner network to adjust hyperparameters without retraining a style transfer network, otherwise requiring hours of computation. This document attempts to recreate the paper's findings, by reproducing figures and describing the difficulties and nuances in reproducing the described network. Relevant code can be found at the Real-Time-Adjustable-Style-Transfer *repository* (git.io/JsT1E)

1 INTRODUCTION

This report evaluates the reproducibility of Babaeizadeh & Ghiasi (2018). Neural Style Transfer traditionally requires retraining of networks whilst varying hyperparameters over many hours to produce sensible results. This recent work optimises a new network that extends the input to include these hyperparameters, thus allowing for real-time tuning to produce variable and more suitable results without retraining. Result reproduction utilised an existing code base for similar problems.

2 REPRODUCTION OF CODE

2.1 ADAPTING EXISTING CODE

To understand the established structure of code for style transfer problems, it was useful to find an existing implementation for previous, similar papers. Investigation yielded an implementation using the PyTorch library, from Nikita Prudnikov. During this work, it was realised that although the basic implementation follows the intentions of the original paper, minute implementation details were overlooked. Incorrect VGG-19 layers were used, with inconsistent use of instance normalisation and exponential moving average compensation on content loss, so manual scale correction was required, the removal of which is the intention of this paper.

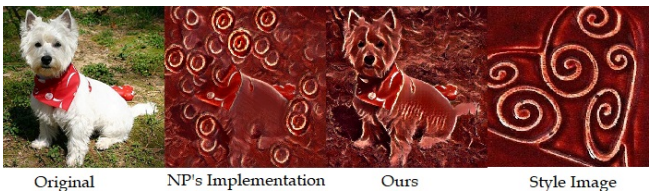


Figure 1: Initial Results Comparison

Overall, the original code was found to prioritise style loss and the conditioner network responsible for hyperparameter replacement was less effective than expected, meaning the content image became lost. Regardless, this implementation provided a base to amend and more accurately follow the intended implementation. Example stylised images can be seen in Figure 1, for comparison against the amended version.

2.2 IMPLEMENTATION DETAILS

All references to equations and figures made in this section are with respect to Babaeizadeh & Ghiasi (2018). Details of the network architectures were provided in the appendix. The conditioner was

comprised of 11 dense layers; however, the output function of each layer was not explicitly stated (like it was for the stylizing network). After our first attempt of replicating the model failed, we assumed ReLU activations were used on all but the output layer, as standard for dense layers. The full architecture of the stylizing network was explicitly stated. The conditional instance normalisation, as given by equation 6, was applied to the output of each activation function following all convolutions.

Equation 1 shows the loss was computed by passing images into a pre-trained network and extracting specific features. There were several discrepancies in this section. From how the loss was detailed, it seemed like equation 1 included a mistake and the content loss at layer l should be between the activation of the stylized and content images. Furthermore, figure 2 indicates VGG-16 was utilised whereas section 5.1 stated VGG-19. The implementation details were followed and VGG-19 was implemented. Finally, the implementation details describes the features extracted as 'the last feature set of conv2, conv3 and conv4 layers'. After downloading the pre-trained VGG-19 network from Pytorch, we found these labels didn't map to those used in the module description and we were unable to confidently identify the correct layers to which 'the last feature set' corresponded. We opted to use the convolutional layers $conv2\_2$, $conv3\_3$ & $conv4\_3$.

The model was now ready to be trained using the loss and optimisation details specified throughout the paper and in the appendix. However, before this, the training images/data had to be prepared. The paper provided no details of how the images were pre-processed so we decided to re-use the transformations from the existing code. The content images, before input to the stylizing network were resized to 256x256, centre cropped and each pixel value was scaled by 255. This ensured all content images were formatted identically. The style image, before being standardised, had each pixel value scaled by 255. Before passing any image into VGG-19, all pixel values were divided by 255 and each input channel was standardised using the ImageNet means and variances. Regarding the training data, section 5.1 and the appendix provided the information, where the appendix details were interpreted as training over a single epoch using 200,000 batches of size 8.

2.3 Training, Initial Results & Loss Values

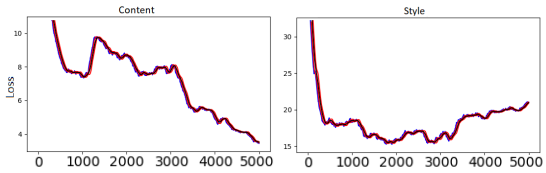


Figure 2: Loss Function Saddle Point at Batch $\approx$ 3000

In contrast to the existing code, when re-implemented, the networks trained steadily, with a decreasing loss value for both style and content. However, at approximately batch 3000 of training, the loss function appeared to reach a saddle point. Beyond this, the content loss dropped dramatically and the style loss increased, as seen in Figure 2. This may be due to sharing of VGG-19 layers between content and style. As a result, supervision was required to ensure this did not occur.

The trained revised network produced considerably more convincing results, though it is arguable that it learned a content loss that was too low, resulting in less variety. The two implementations' outputs can be seen in Figure 1. Each new style image used requires 200,000 epochs to adhere strictly to the ICLR paper ($\sim$ 25 hours), so it was decided to use only the most common style.

3 Figure Reproducibility

3.1 Layer Loss

Figure 7 in Babaeizadeh & Ghiasi (2018) encapsulates the main motivation of the original ICLR paper. It is clear that by adjusting the input vector $\boldsymbol{\alpha}$, the respective loss of that layer of the VGG-19 network should decrease when compared against a style image. This behaviour facilitates the *real-time adjustable* nature of the network. The network was passed an image and the elements of the input vector $\boldsymbol{\alpha}$ varied individually between zero and one, whilst keeping the other elements constant

at zero or one. The reproduced plot using our implementation can be seen in Figure 3. Note that, unexpectedly some layer losses, in fact, increase with their respective α value. It is not clear why this is the case, but it may correlate with findings of Section 3.3.

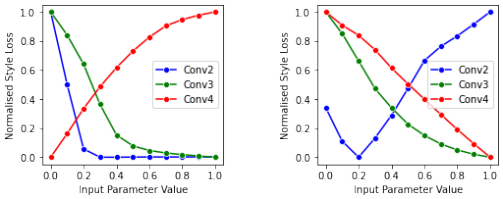


Figure 3: Layer Loss Plots w/ Remaining α_s kept at 0 (Left) and 1 (Right)

3.2 Varying the Image Style

When reproducing Figures 3 and 6 from Babaeizadeh & Ghiasi (2018), several issues occurred as no source images were defined, nor their scale. Testing found that the size of the input image affected the styling of the output image. Using reverse image search, the original test images were found and scaled down to a size of 300x300 for optimal results. The first experiment was to reproduce the effect of adjusting input parameters on stylisation (Figure 3). The stylised output was observed with a given layer weight increasing from 0 to 1, with the rest fixed to 0. The details of each stylised image should vary with deeper layers representing larger features of the style image.

Using Prudnikov's model, the input image was stylised, and intensity of stylisation did increase as with increased weights, shown in Figure 4. However, it is unclear from this reproduction if deeper layers used larger features of the style image to stylise the content. A major issue with this model was that when all style weights were 0, content image reconstruction was imperfect. Using the group's model, the input image was stylised and again, stylisation did increase with increased weights as shown in Figure 4. Unlike the previous model, the deeper layer $conv\_3$ does use larger features to style the image, when compared to $conv\_2$. The problem with the group's model was that the deepest layer $conv\_4$ was not contributing any style to the image.

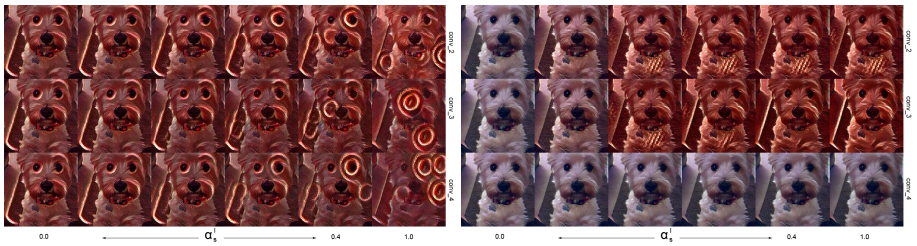


Figure 4: Truncated reproduction of Figure 3 using NP's (left) and our own implementation (right)

The second experiment was to reproduce the effects of combining the styling from several convolution layers (Figure 6). There is no specific result here, however the styling should look different after combining the effects from different layers. Using both Prudnikov's model and the group's model, the input image was stylised based on a combination of layers as shown in Figure 5. Both models did produce distinct stylised images, with Prudnikov's model opting to favour circular style features, whilst the group's model favoured linear features.

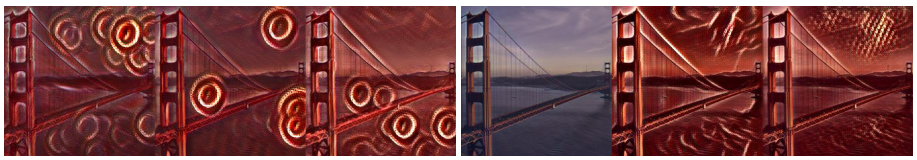


Figure 5: Truncated reproduction of Figure 6 using NP's (left) and our own implementation (right)

3.3 EFFECT OF NOISE

Figure 5 from Babaeizadeh & Ghiasi (2018) demonstrates the effect on the images produced by adjusting parameters of the model, adding a few points of Gaussian noise and doing both of these things. The paper specifies that the noise should be produced by multiplying the input image with a white mask, in which fewer than 10 randomly-chosen pixels have been changed to 0 (black). An inverse Gaussian filter is then applied to this mask. Efforts to reproduce the figure using Prudnikov's code were unsuccessful. The tiny amounts of added noise did not affect the final result as much as the paper suggests, and all produced images were identical. The most likely explanation is that the transformations using Prudnikov's code are not sensitive enough for such small changes to make a difference. Experimenting by randomly adding larger noise spots did produce different results, as shown in Figure 6. Reproduction of the figure using this technique demonstrates that the addition of noise to the input image does affect the image produced, though the images produced are not particularly similar to those produced by the original author and the method is different. The same process was carried out using the new model, as shown on the right in Figure 7. While the inverse Gaussian method of adding noise again didn't make any noticeable difference, there was a substantial difference when adding the black spots as described above. The effect of adding the randomisation is generally less visible using the new model, due to its avoidance of the other model's tendency to obscure details by adding swirls.

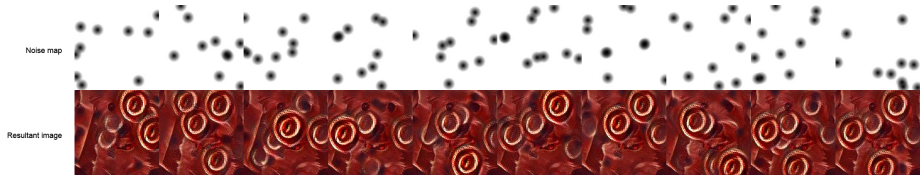


Figure 6: Effect of adding noise to the input image

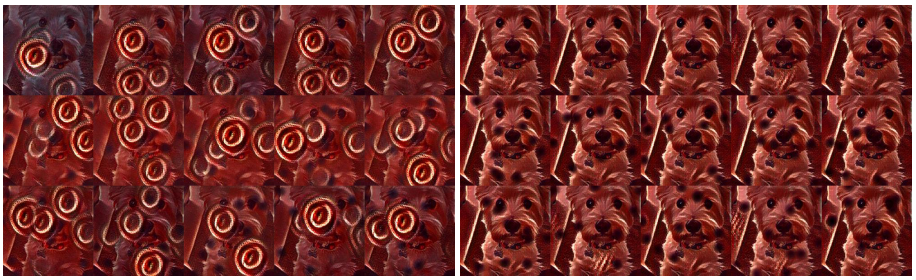


Figure 7: Truncated reproduction of Figure 5 using NP's (left) and our own implementation (right)

4 CONCLUSION

Using the methodology provided by Babaeizadeh & Ghiasi (2018), an Adjustable Real-Time Style Transfer model was implemented, reproducing results to an acceptable level. Likely due to the under-specification in the paper, the reproductions were not completely representative of the original, however they still showed that stylisation of an image can be varied using weights, without retraining the network. Due to the time needed to train one style network, testing capabilities were limited when compared to the original paper's.

REFERENCES

Mohammad Babaeizadeh and Golnaz Ghiasi. Adjustable real-time style transfer. *arXiv preprint arXiv:1811.08560*, 2018.

Student ID: 29466423
Username: rsb1g17
Full Name: Rohan Bungre

COMP6212 Deep Learning 2020/21

Lab 1 Exercise - Playing with gradients and matrices in PyTorch (5%)

1 Implement a matrix factorisation using gradient descent

```
def sgd_factorise(A,r,N,lr):

  U = torch.rand(m, r)
  V = torch.rand(n, r)

  for epoch in range(N):
    for r in range(m):
      for c in range(n):

        e = A[r, c] - U[r] @ V[c].t()
        U[r] = U[r] + lr*e*V[c]
        V[c] = V[c] + lr*e*U[r]

  return U, V
```

The low-rank matrix factorisation can be achieved by considering the following optimisation problem $min(\left\|\boldsymbol{A}-\hat{\boldsymbol{U}}\hat{\boldsymbol{V}}^{\top}\right\|_{\mathrm{F}}^{2})$. This was implemented by the sgd_factorise() function. Given an input

$$A: \begin{bmatrix} 0.3374 & 0.6005 & 0.1735 \\ 3.3359 & 0.0492 & 1.8374 \\ 2.9407 & 0.5301 & 2.2620 \end{bmatrix},$$

a rank 2 factorisation was calculated as

$$U: \begin{bmatrix} -0.2620 & 0.5964 \\ 1.5184 & 0.6313 \\ 1.0029 & 1.2218 \end{bmatrix} V: \begin{bmatrix} 1.6768 & 1.1121 \\ -0.3149 & 0.7453 \\ 0.8922 & 0.9881 \end{bmatrix}.$$

The reconstruction loss of this rank-2 factorisation was 0.1220.

2 Compare your result to truncated SVD

Eckart and Young gave an elegant constructive solution to finding the approximation of one matrix by another of lower rank or to minimize over $\widehat{D} \quad \|\boldsymbol{A}-\widehat{\boldsymbol{A}}\|_{\mathrm{F}} \quad$ subject to $\quad \operatorname{rank}(\widehat{\boldsymbol{A}}) \leq r$. They found that the rank-r matrix, obtained from the truncated singular value decomposition $\tilde{\boldsymbol{A}}=\boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^{\top}$, is such that $\left\|\boldsymbol{A}-\tilde{\boldsymbol{A}}^{*}\right\|_{\mathrm{F}}=\min_{\operatorname{rank}(\widehat{\boldsymbol{A}})\leq r}\|\boldsymbol{A}-\widehat{\boldsymbol{A}}\|_{\mathrm{F}}=\sqrt{\sigma_{r+1}^2+\cdots+\sigma_m^2}$. Due to this, the results from the gradient-based factorisation should be similar to the truncated SVD. Calculating the SVD of A and setting the last singular value to 0 gave a rank-2 approximation of $\widehat{A}$. The reconstruction loss of the SVD rank-2 truncation was 0.1219. $A: \begin{bmatrix} 0.3374 & 0.6005 & 0.1735 \\ 3.3359 & 0.0492 & 1.8374 \\ 2.9407 & 0.5301 & 2.2620 \end{bmatrix}, \widehat{A}: \begin{bmatrix} 0.2245 & 0.5212 & 0.3592 \\ 3.2530 & -0.0090 & 1.9737 \\ 3.0378 & 0.5983 & 2.1023 \end{bmatrix}$

3 Matrix completion

```
def sgd_factorise_masked(A,M,r,N,lr):

  U = torch.rand(m, r)
  V = torch.rand(n, r)

  for epoch in range(N):
    for r in range(m):
      for c in range(n):
        if M[r,c]=='1':

          e = A[r, c] - U[r] @ V[c].t()
          U[r] = U[r] + lr*e*V[c]
          V[c] = V[c] + lr*e*U[r]

  return U, V
```

A gradient based approach for matrix factorisation can be used for matrix completion. This is when missing data in the matrix can be inferred from some kind of assumption about the processes that produced the data. Using the binary mask matrix to only compute updates using the valid values defined in the question gives an estimate:

$$\widehat{\boldsymbol{A}} = \begin{bmatrix} 0.2257 & 0.5345 & 0.3519 \\ 3.2448 & -0.0046 & 1.9830] \\ 3.0433 & 0.5908 & 2.0980 \end{bmatrix},$$

which gives a reduction loss of 0.2221. This shows us that we can perform very accurate matrix completion using a gradient-based approach to matrix factorisation. This may confirm the assumption that the process is controlled by a small number of latent variables.

Student ID: 29466423
Username: rsb1g17
Full Name: Rohan Bungre

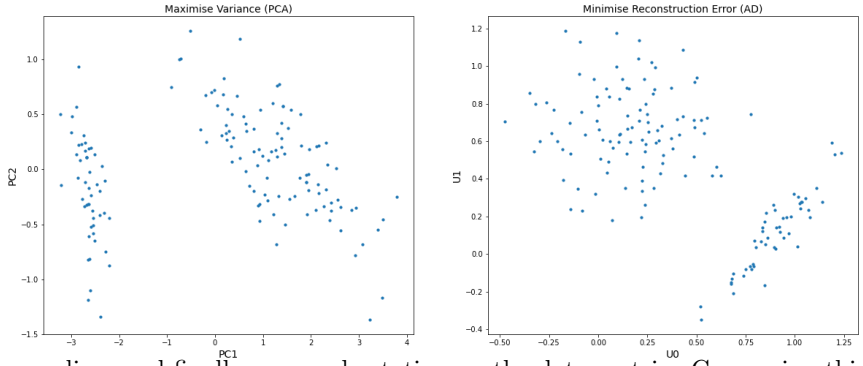
COMP6212 Deep Learning 2020/21

Lab 2 Exercise - PyTorch Autograd (5%)

1 Implement matrix factorisation using gradient descent

```
def sgd_factorise_ad(A, r, N, lr):
  m, n = A.shape
  U = Variable(torch.rand(m, r),requires_grad=True)
  V = Variable(torch.rand(n, r),requires_grad=True)
  for epoch in range(N):
    loss_fn = torch.nn.MSELoss(reduction = 'sum')
    e = loss_fn(A, U@V.t())
    e.backward()
    U.data = U.data - lr*U.grad.data
    V.data = V.data - lr*V.grad.data
    U.grad.data.zero_()
    V.grad.data.zero_()
  return U, V
```

Unlike the first lab, the sgd_factorise_ad() function implements the PyTorch AD framework. Each element is not updated one at a time, but rather all the gradients of $\widehat{U}$ and $\widehat{V}$ are computed and applied once per epoch. This function was used to compute the rank-2 factorisation of the iris data matrix and compare the reconstruction loss to that of a rank-2 reconstruction computed using a truncated Singular Value Decomposition (SVD) of the same data. These methods achieving reconstruction losses of 15.2289 and 15.2288 respectively, showing they are similar in practice.



When comparing the scatter plot of the data projected onto the first two principle axes computed by SVD to a second scatter plot from the data in matrix $\hat{\boldsymbol{U}}$, we see that there is clearly a relationship with two clear clusters in both plots. SVD applies a first rotation, then a scaling, and finally a second rotation on the data matrix. Comparing this to just the initial data matrix, there will be a rotation and scaling factor of difference between the two. This can be seen in the data plots.

2 A simple MLP

```
W1 = Variable(torch.randn(4,12),requires_grad=True)
W2 = Variable(torch.randn(12,3),requires_grad=True)
b1 = Variable(torch.tensor(0.0),requires_grad=True)
b2 = Variable(torch.tensor(0.0),requires_grad=True)

for epoch in range(N):
  logits = torch.relu(data_tr @ W1 + b1) @ W2 +b2
  loss = torch.nn.functional.cross_entropy(logits,tr)
  loss.backward()
  with torch.no_grad():
    W1.data = W1.data - lr*W1.grad.data
    W2.data = W2.data - lr*W2.grad.data
    b1.data = b1.data - lr*b1.grad.data
    b2.data = b2.data - lr*b2.grad.data

  W1.grad.data.zero_()
  W2.grad.data.zero_()
  b1.grad.data.zero_()
  b2.grad.data.zero_()
```

When running this MLP on the training data, and computing both the training and validation accuracies at the end of training, the observation was that the training and validation accuracies vary over different repeats. Sometimes the validation accuracy is higher than that of the training accuracy.

Student ID: 29466423
Username: rsb1g17
Full Name: Rohan Bungre

COMP6212 Deep Learning 2020/21

Lab 3 Exercise - Optimise it! (5%)

1 Exploring optimisation of analytic functions

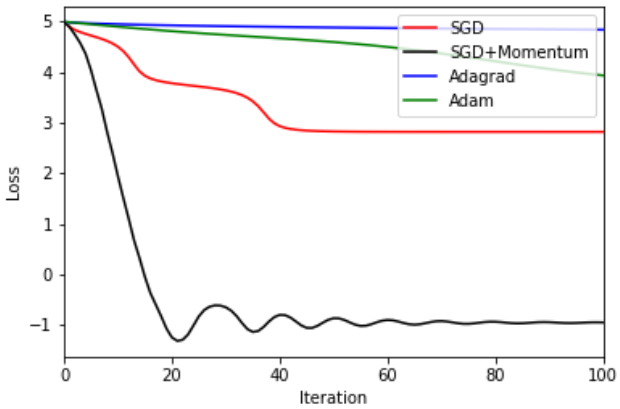
The 2D Rastrigin function can be defined as:

$$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos\left(2\pi x_i\right)\right], \tag{1}$$

Finding the minimum of this function is a fairly difficult problem due to its large search space and its large number of local minima. To make the problem easier A will be set to a value of 1. Starting at a point (5, 5) and computing the point where a number optimisers arrive at after 100 epochs. PyTorch implementations of the optimisers with the default values for unspecified parameters were used with learning rates of 0.01. The points of convergence were:

SGD: (2.8224, 2.8224), **SGD+Mom:** (-0.9470, -0.9470), **Adagrad:** (4.8423, 4.8423) **Adam:** (3.9386, 3.9386).

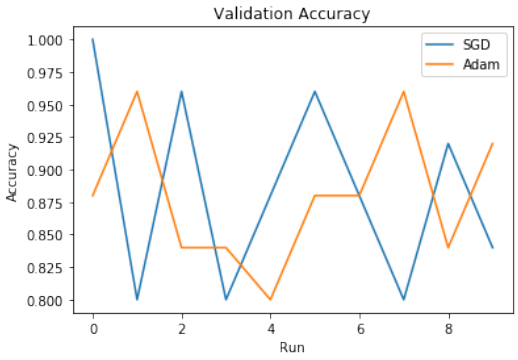
The loss plots show that SGD + 0.9 Momentum far outperforms the other optimisers in this scenario, with Adagrad getting stuck at a local minimum very early. Perhaps the poor performing optimisers would be able to get out of the local minima if non-default values were used.



2 Optimisation of a SVM on real data

```
def hinge_loss(y_pred, y_true):
    loss = torch.mean(torch.clamp(1- y_pred.t()*
    y_true,min=0))
    return loss

def svm(x, w, b):
    h = (w*x).sum(1) + b
    return h

SGD = optim.SGD([w,b],lr=0.01,weight_decay=0.0001)
Adam = optim.Adam([w,b],lr=0.01,weight_decay=0.0001)
```

A Soft-margin Linear SVM was trained with different optimisers, using a batch size of 25 and trained over 100 epochs. This was used to solve the two-class classification problem of the Iris dataset. The two optimisers were SDG and Adam both using a learning rate of 0.01 and weight decay of 0.0001. The average validation accuracy and variance over multiple different random initialisations were:



SGD: [Average accuracy: 0.9116, Variance: 0.0048] and **Adam: [Average accuracy: 0.8952, Variance: 0.0026]**. We also see that for different random initialisations, the validation accuracy strongly varies. This suggests that depending from where the algorithm initially starts, its performance will be affected. SGD has the highest accuracy, but is the most temperamental with random initialisations. I did not notice anything counter-intuitive but did not expect such large variations in validation accuracy.

Student ID: 29466423
Username: rsb1g17
Full Name: Rohan Bungre

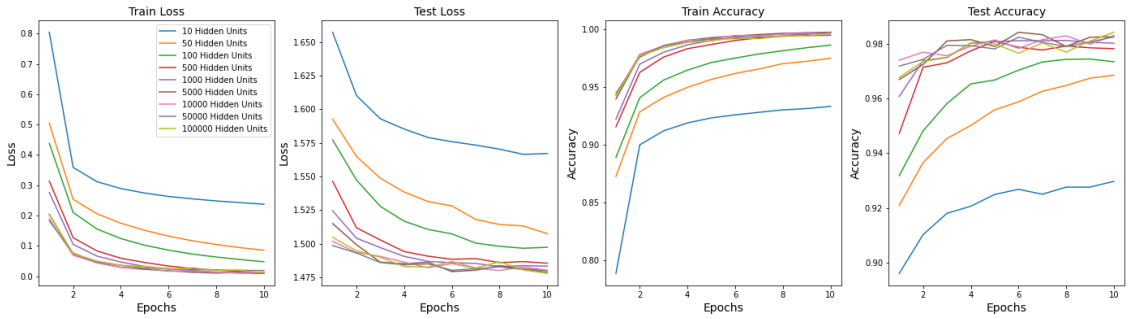
COMP6212 Deep Learning 2020/21
Lab 4 Exercise - Fun with MLPs MNIST (5%)

1 Wide MLPs on MNIST

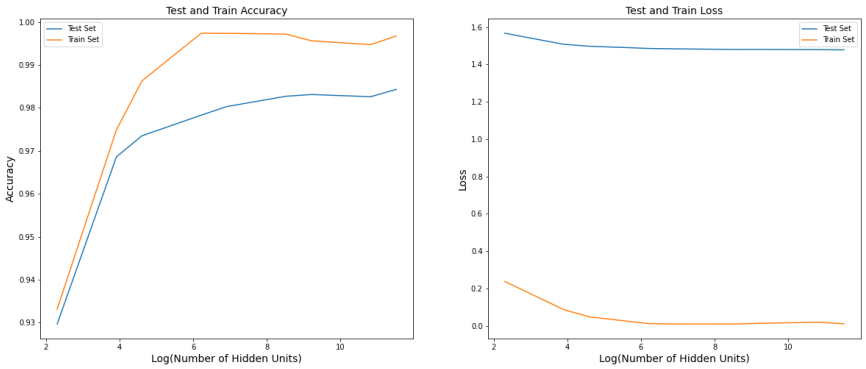
```
class BaselineModel(nn.Module):
    def __init__(self,input_size,hidden_size,num_classes):
        super(BaselineModel, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.fc2 = nn.Linear(hidden_size, num_classes)
    def forward(self, x):
        out = self.fc1(x)
        out = F.relu(out)
        out = self.fc2(out)
        if not self.training:
            out = F.softmax(out, dim=1)
        return out
```

A baseline MLP model was created with one hidden layer. The aim of this lab was to explore the effects on the training and validation accuracy and loss when changing the width (number of nodes in the hidden layer) in the MPL. This helped to determine if over-fitting was occuring on the MNIST dataset. Over-fitting occurs when the model fails to generalise reasonably to the MNIST test set.

The number of learnable parameters in the MLP can be calculated by the number of input nodes, hidden nodes and output nodes. It is sensible to assume that with the more learnable parameters in the model, the quicker(in terms of epochs) it over-fits. Intuitvely, when the number of learnable parameters is equal to the number of distinct data points, the model will start to overfit. As there are 60,000 training images in the MNIST dataset, the model should theoretically overfit with 60,000 hidden units.



When plotting the loss and accuracies against epochs for different size models, it was shown that the accuracy of test and train results remained pretty stable. The wider networks also have better accuracy and loss values. At 10 epochs, the wider the network, the better the training and test accuracies were. When plotting the training and test accuracies and loss against the number of hidden units, it was shown that both accuracies continues to increase even when the number of hidden units was 100,000. There is no evidence to suggest that the network is over-fitting as the model generalises well to the MNIST test set. The test accuracy is still very high, even at a large number of hidden units, which could suggest that the data is easily separable.



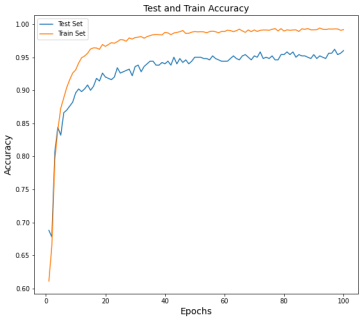
Student ID: 29466423
Username: rsb1g17
Full Name: Rohan Bungre

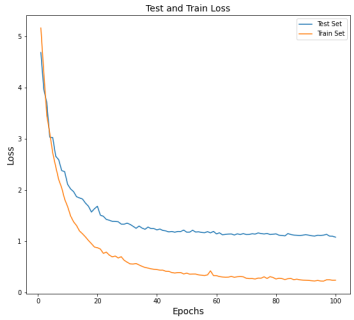
COMP6212 Deep Learning 2020/21

Lab 5 Exercise - A little Linear Regression (5%)

Due to the 1 page limit the code for each CNN model has been omitted.

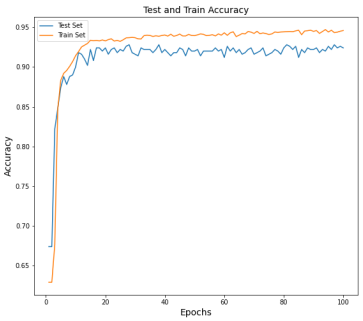
1 A simple CNN baseline

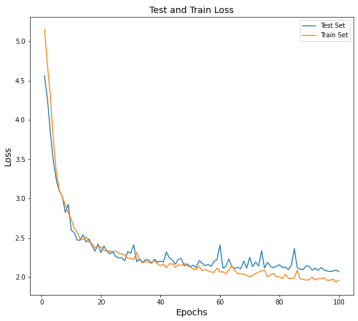




This lab explored the performance of a CNN as the complexity of the model was increased. This simple model uses a convolution layer and a fully connected layer. The loss was calculated using the nn.L1Loss() function. The test loss was 1.29 and the test accuracy was 0.94. The loss and accuracy curves show that model is overfitting.

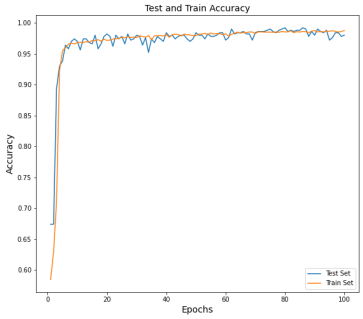
2 A simple CNN with global pooling

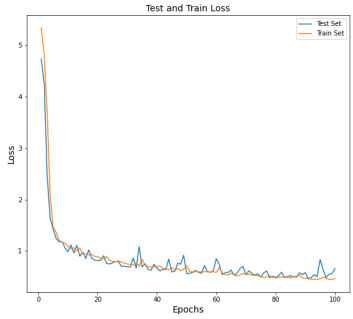




Overfitting is most likely to happen within the fully connected layer. Therefore reducing the number of nodes in it would reduce the amount of overfitting. To acheive this an extra convolution layer was added along with a global max-pooling layer. The global pooling layer calculates the maximum or average from a group of pixels directly from the input data. This meant that the number of nodes decreased so the model should not overfit as much. Dropout could be used as an alternative to help reduce the number of nodes. The test loss was 2.25 and the test accuracy was 0.92. From the accuracy and loss plot this technique helped to improve the generalisation of the model, along with increasing the convergence speed. This model however had a decreased accuracy and an increased loss.

3 Something that actually works?





To further improve the performance of the CNN model, a pair of interleaved layers was introduced to transform the input data. The first channel remains to be a 40x40 tensor with zeros and ones, representing the line. The second and third channel represent the x and y grid. Note that this is the same for every data item. It uses 48 channels, each with a different filter. By encoding the grid in x, you retain the spatial information. Doing this aimed to increase the accuracy of the model, whilst improving the generalisation ability. The test loss was 0.71 and the test accuracy was 0.97. From the accuracy and loss plots this technique was able to have an increased accuracy whilst being generalised.

Student ID: 29466423
Username: rsb1g17
Full Name: Rohan Bungre

COMP6212 Deep Learning 2020/21

Lab 6 Exercise - Reflections on transfer learning (5%)

1 Transfer Learning

Transfer learning is a machine learning technique where a model trained for one task is re-purposed for a second related task. The task was focused on the classification of boats in the Venice canals, using a rather small and poorly balanced dataset comprising of 4774 images from 24 different classes. Training a simple CNN model on this dataset provided a validation accuracy of 0.71, however performs extremely poorly for classes with a small amount of training examples. Ideally more data would be collected and a more complex model would be built for this model, however this can be expansive and impractical. Transfer learning aims to help by using a model such as ResNet50, to classify the Venice canal boats.

1.1 Finetuning

The first method of finetuning involved modifying the ResNet50's architecture to work with the boat dataset. An adaptive average pooling layer was added to keep the training images at a high resolution without cropping. The model's wide classification layer was replaced by new one for the 24 classes in the dataset. To tune the model with the new data, previously trained weights were frozen whilst initial weights were learnt for the new layers added to avoid over-fitting. This was motivated by the observation that the earlier features of ResNet50 contain more generic features that were useful for this classification task. It is possible to fine-tune all the layers of the modified ResNet50 however, a smaller learning rate would have to be used for ResNet50 weights that are being fine-tuned, in comparison to the (randomly-initialised) weights for the new linear classifier that computes the class scores of your new dataset. This is because it is expected that the ResNet50 weights are relatively good, so don't wish to distort them too quickly and by too much. Practically this method would take far too long to train.

1.2 Feature Extraction

The second method involved extracting features from the original ResNet50 model and training a machine learning classifier to find correlations between these features and the boat classification outputs. Doing this involved removing the last few fully connected layers of the pre-trained network and connect it with a Support Vector Machine that correspond to the canal boat classification task.

1.3 Comparison

	precision	recall	f1-score	support
Alilaguna	0.67	0.32	0.43	19
Ambulanza	0.67	0.36	0.47	22
Barchino	0.60	0.06	0.11	51
Gondola	0.00	0.00	0.00	3
Lanciafinol0m	0.00	0.00	0.00	7
Motobarca	0.17	0.03	0.06	59
Motopontonerettangolare	0.00	0.00	0.00	3
MotoscafoACTV	0.00	0.00	0.00	1
Mototopo	0.58	0.89	0.70	274
Patanella	0.40	0.45	0.42	74
Polizia	0.00	0.00	0.00	15
Raccoltarifiuti	0.80	0.42	0.55	19
Sandoloaremi	0.00	0.00	0.00	3
Topa	0.00	0.00	0.00	29
VaporettoACTV	0.94	1.00	0.97	325
Water	0.90	0.92	0.91	420
accuracy			0.77	1324
macro avg	0.36	0.28	0.29	1324
weighted avg	0.72	0.77	0.72	1324

Figure 1: Finetuning

	precision	recall	f1-score	support
Alilaguna	0.95	1.00	0.97	19
Ambulanza	0.87	0.91	0.89	22
Barchino	0.73	0.53	0.61	51
Gondola	0.75	1.00	0.86	3
Lanciafinol0m	0.00	0.00	0.00	7
Motobarca	0.62	0.61	0.62	59
Motopontonerettangolare	1.00	1.00	1.00	3
MotoscafoACTV	1.00	1.00	1.00	1
Mototopo	0.94	0.94	0.94	274
Patanella	0.60	0.84	0.70	74
Polizia	0.67	0.80	0.73	15
Raccoltarifiuti	0.84	0.84	0.84	19
Sandoloaremi	1.00	0.67	0.80	3
Topa	0.62	0.52	0.57	29
VaporettoACTV	0.99	1.00	1.00	325
Water	1.00	0.96	0.98	420
accuracy			0.91	1324
macro avg	0.79	0.79	0.78	1324
weighted avg	0.91	0.91	0.91	1324

Figure 2: Feature Extraction

The first method, where the classification head of the network was modified and finetuning was performed, produced a validation accuracy of 0.71. From Figure 1 the metrics report shows that the model was not able to accurately classify across all classes. This can be seen by weighted average values. This was probably due to the lack of training data used to train the newly added layers. The second method, where network features were used to train SVM classifiers, produced a validation accuracy of 0.91 after being optimised with the hyper-parameter C = 31. From Figure 2 the metrics report shows that the model had a fairly good accuracy across all classes. The second method performs well even when there was a lack of training data and was the faster of the two models.

Student ID: 29466423
Username: rsb1g17
Full Name: Rohan Bungre

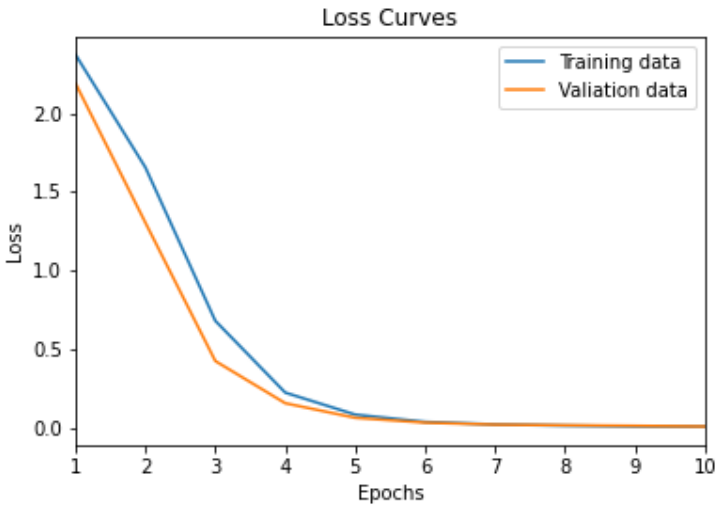
COMP6212 Deep Learning 2020/21
Lab 7 Exercise - Transforming Sequences (5%)

1 Sequence-to-Sequence Modelling

```
class Encoder(nn.Module):
    def __init__(self, input_dim, emb_dim, hid_dim):
        super().__init__()
        self.hid_dim = hid_dim
        self.embedding = nn.Embedding(input_dim, emb_dim)
        self.rnn = nn.LSTM(emb_dim, hid_dim)
    def forward(self, src):
        embedded = self.embedding(src)
        output, (hidden, cell) = self.rnn(embedded)
        return hidden, cell
```

Sequence to sequence modelling allows a variable length input sequence to be translated to an output sequence. This is the backbone of language translation models. The model used LSTMs to encode and decode the sequences as vectors, trained on a dataset of example sequence translations. The code on the left shows the completed Encoder class, with a completed forward function.

Training the model over 10 epochs produced a loss plot. Interestingly the validation loss was lower than the training loss. Using the decode function provided, the codes given were decoded into English. This first code was: "answer the following". The second code was: "why is the order of the output reversed". The third code was: "what is the point of teacher forcing".



Current research states "we do not have a complete explanation to this phenomenon" of output reversing. However the theory is that by reversing the sentence, it is easier to map the input to the output sequence. By reversing the words in the source sentence, the average distance between corresponding words in the source and target language was unchanged. However, the first few words in the source language are now very close to the first few words in the target language, so the problem's minimal time lag is greatly reduced. Thus, back propagation has an easier time "establishing communication" between the source sentence and the target sentence, which in turn results in substantially improved overall performance.

Teacher forcing is a method for quickly and efficiently training recurrent neural network models that use the ground truth from a prior time step as input. It is a network training method critical to the development of deep learning language models used in machine translation, text summarising, and image captioning, among many other applications. The teacher forcing method addresses slow convergence and instability when training recurrent neural networks that use output from prior time steps as input.

After giving the model a few examples of longer chunks, it was found that the model does not work very well with these longer chunks. This may be directly related to the fact that the training data consists primarily of smaller chunks. The model may find it hard to infer meaning from and translate the longer chunks.

Student ID: 29466423
Username: rsb1g17
Full Name: Rohan Bungre

COMP6212 Deep Learning 2020/21
Lab 8 Exercise - Exploring Latent Spaces (5%)

1 Exploring the latent space of a VAE and the code space of a standard Auto-Encoder

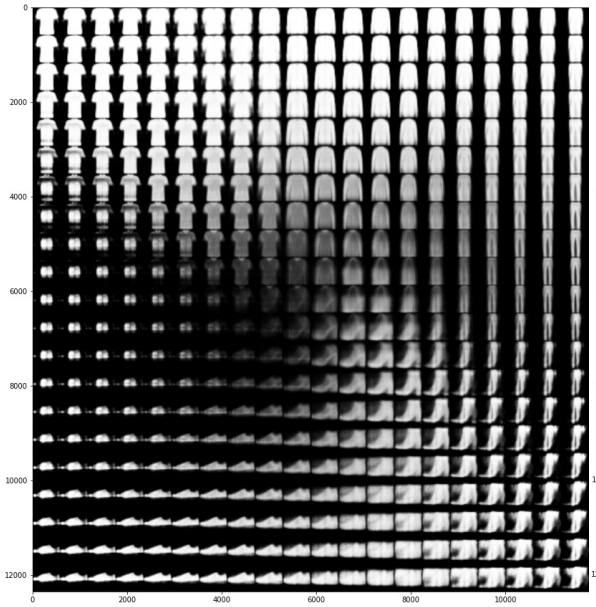


Figure 1: Variational Auto-Encoder

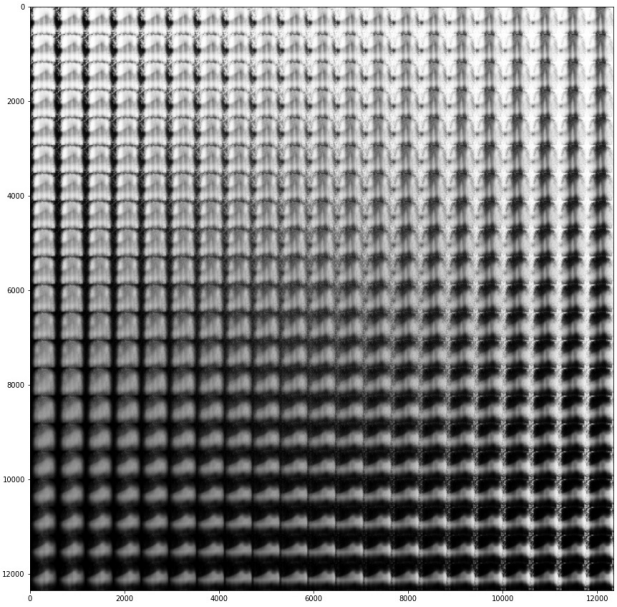


Figure 2: Standard Auto-Encoder

Using the Fashion MNIST dataset, both a VAE and AE models were created and trained. The decoder networks were used to generate images by uniformly sampling the prior from 4σ to $+4\sigma$ in both dimensions. 21 samples per dimension were created, leaving a total of 441 2-d z vectors. Each reconstruction was drawn into a large 588 $\times$ 588 image according to the z vector that generated it. The image generated from z = [-4, 4] is in the top left, and the image from z = [4, -4] is in the bottom right. For the Auto-Encoder the latent space was changed to 2-d rather than 64-d. The generated images can be seen in Figure 1 and Figure 2.

2 Compare the latent spaces of the VAE and Auto-Encoder

The images generated from the VAE latent space are much clearer and representative of the orignial Fashion MNIST dataset compared to the images generated from the AE latent space. The top left and bottom right images from the VAE are the best generated images. None of the AE generated images looked good.

An autoencoder takes an input, compresses it, and then recreates the original input. The aim of an autoencoder is to learn an encoding for a set of data, by training the network to ignore signal noise. The two main uses of an autoencoder are to compress data to lower dimensions so it can be graphed, and to compress and decompress images or documents. It is not able to capture the underlying distribution of the source data, so fails to generate good images, representative of the Fashion MNIST dataset.

However, a variational autoencoder assumes that the source data has some sort of underlying probability distribution and then attempts to find the parameters of the distribution. The one main use of a variational autoencoder is to generate new data that's related to the original source data. A variational autoencoder is a generative system, and serves a similar purpose as a generative adversarial network so will generate good images, representative of the Fashion MNIST dataset.

Group 13: Airbnb New User Bookings

Abdulaziz Alkhalefah
asaa1g18@soton.ac.uk
University of Southampton
Southampton, UK

Rohan Bungre
rsb1g17@soton.ac.uk
University of Southampton
Southampton, UK

Harry Goatman
hjg1g17@soton.ac.uk
University of Southampton
Southampton, UK

Jack Pennington
jcp3g17@soton.ac.uk
University of Southampton
Southampton, UK

Andrea Sanchez Fresneda
asf1g17@soton.ac.uk
University of Southampton
Southampton, UK

William Savage
ws1g17@soton.ac.uk
University of Southampton
Southampton, UK

ABSTRACT

This report outlines the steps taken to predict new Airbnb users' first booking destination, with justifications and conclusions made throughout the different processes. We investigated a range of data mining techniques to; explore the dataset, including feature instance distributions and correlations between features; prepare the dataset through feature engineering, scaling and encoding; and predict the classes using supervised algorithms able to handle multiple classes. Ultimately, an $F1 - score$ of 0.83 and an $nDCG$ score of 0.86 on the official Kaggle competition was achieved, beating our initial baseline model.

CCS CONCEPTS

• **Computing methodologies** → **Classification and regression trees**; *Supervised learning*; **Machine learning**; **Ensemble methods**; **Classification and regression trees**; **Supervised learning by classification**.

KEYWORDS

data mining, machine learning, classification, supervised learning, kaggle, airbnb

ACM Reference Format:
Abdulaziz Alkhalefah, Rohan Bungre, Harry Goatman, Jack Pennington, Andrea Sanchez Fresneda, and William Savage. 2021. Group 13: Airbnb New User Bookings. In *Proceedings of COMP6237: Data Mining (COMP6237).* ACM, UoS, Southampton, UK, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The problem we chose to tackle was Airbnb's Kaggle competition: New Users Bookings [1]. The objective was to predict which country new Airbnb users will book as their first destination. Accurately predicting a user's booking, Airbnb can help their users tackle the overwhelming task of exploring stays in over 34,000 cities, ensure personalised advertisements reach the appropriate audience, and overall further understand their users.

This is a predictive data mining problem complemented with rich enough data to allow us to explore a range of user habits and apply a variety of techniques and models to evaluate and compare results.

1.1 Data Description

Airbnb provided 5 datasets, including user demographics, web session records and some general summary statistics about the destination countries (details on these are provided in section 2). However we needed not to use all the datasets to build the predictive model. Thus our first challenge was to decide what data was useful to the problem (based on analysis). Note: all users in the datasets are from the USA.

1.2 Baseline Model

A preliminary baseline model was built prior to any exploration or model-building to help understand the task/data, for use during the evaluation stage, and to set an initial value to improve from. It was built following these steps:

(1) Find top 5 most common destination countries amongst all users in train dataset
(2) For each user in test, predict their destination as the top 5, to calculate nDCG with $k = 5$ (refer to subsubsection 1.3.2).

The model was submitted to the Kaggle competition and obtained an nDCG score of 0.85 (refer to subsubsection 1.3.2) and a cross-validation score of 0.807.

1.3 Evaluation Strategy

1.3.1 Cross-validation. To validate our model, we opted for k-fold cross validation (with $k = 5$), allowing us to assess how the results of the models will generalise to an independent dataset and flag problems such as overfitting or selection bias. In k-fold cross validation, the training data is randomly partitioned in k equal sizes (non-overlapping), wherein a single partition is held out for validation and $k - 1$ remaining partitions are used for training. This process is repeated k times (for each of the partitions is to be used as testing data once), the results are then averaged to produce a single score.

[1] https://www.kaggle.com/c/airbnb-recruiting-new-user-bookings/data

COMP6237, May, 2021, Southampton, UK

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1.3.2 Metrics. Our key metric to evaluate model performance was **normalised discounted cumulative gain** (nDCG). It is a measure of ranking quality and it is used by the official competition (with $k = 5$), thus it was set as our key performance metric:

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{log_2(i+1)}$$

$$nDCG_k = \frac{DCG_k}{IDCG_k}$$

where:

rel_i = relevance of result at position i

$IDCG_k$ = maximum possible (ideal) DCG for a given set of queries

Additionally, F1 score was used to measure classification accuracy of our models and compare them.

2 DATA INSPECTION

This section aims to provide a description into our data exploration techniques with justification, gained insights from the data and overall conclusions to get data ready for the next stage.

2.1 Countries & Age-gender buckets datasets

These two provided datasets supplied with general, summary statistics on the destination countries. The former dataset included details on latitude, longitude, language and total area per country. While the latter provided with population distribution as broken down by demographics (age and gender wise) of the destination countries.

2.2 Train dataset

This dataset was the main dataset containing the potential features to be used and the target column (destination country), i.e. the examples to be mainly used during the learning stage of the model. Hence, we proceeded to explore the dataset in a rigorous manner. We found there to be 213,451 entries (no duplicate user id's nor rows) and 15 features (not including the target column), with both categorical and continuous data types.

2.2.1 Univariate. For this analysis, we looked at individual features, such as counts of the different options for every feature, percentage distribution of the different options for every feature and histograms of age distribution.

Our main results and conclusions from this were how the data was highly imbalanced, with 'NDF' dominating with almost 60% of presence over the other 14 countries (refer to Figure 1). With regards to age, there were outliers which were to be dealt with such as extreme values over 100 (2000+) and values under 18, which are illegal as you must be at least 18 to create an account and book on Airbnb. Additionally, the majority (45%) of users chose not disclose their gender and Apple devices are the most popular amongst users. Regarding dates, June was the most popular month to create an account, perhaps users wait until the start of summer to book their holidays, as seen in Figure 2. Finally, there was an extensive amount of `first_browser` instances (52) with the majority appearing < 100 times, thus this will be looked into during feature engineering.

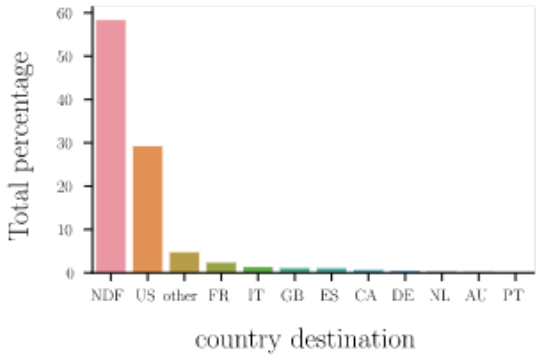


Figure 1: Target column options distribution

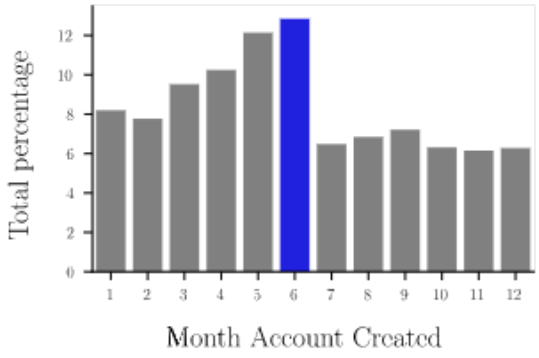


Figure 2: Accounts created per month distribution

2.2.2 Bivariate. The main aim of this was identify any correlations or patterns between different features and the target column, for feature selection and/or feature engineering.

Our findings were as follows. During the winter months, Australia gets booked the most, perhaps as it summer time there. There is a high correlation between 'NDF' and non-disclosed user information on their gender and/or age (plots for these seen in Figure 3 and Figure 4).

2.2.3 Multivariate. We used `t-SNE` with two dimensions to visualise multivariate analysis [2]. Ultimately, no clear clustering was observed as seen in Figure 5. It is to note there might be some clustering found in higher dimensions, however for visualisation purposes, we performed it with only two dimensions.

2.3 Sessions dataset

Contains a log of the users' (in train and test dataset) actions from 2014. Provides details on every click, search, page visited and time between actions. The amount of sessions details per user varies vastly (between 1000 – 6.5million rows per user), therefore we had

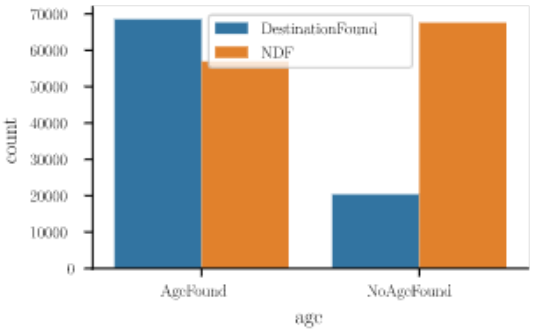


Figure 3: Age Disclosed and Country destination found

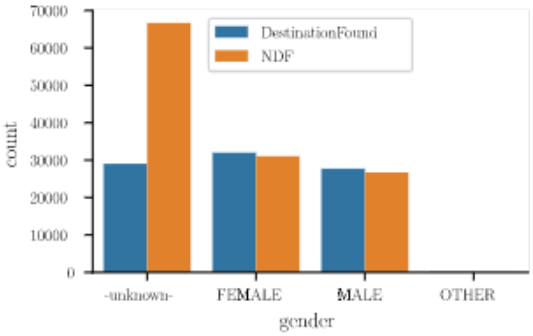


Figure 4: Gender options for destination found or not

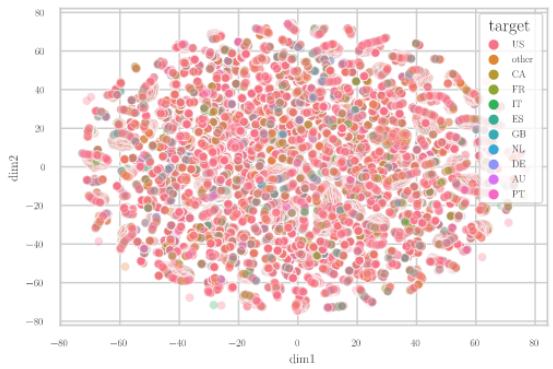


Figure 5: t-SNE on train dataset

to consider how to approach this dataset, and how to condense to a single entry per user (for efficiency purposes and joining with the train dataset). Furthermore, only around 30% of the users in the training dataset were also found in this dataset, entailing we would have to either drop the remaining entries, or extrapolate/fill in the data on sessions for the remaining training subset without such data. We considered having a clustering algorithm to artificially generate this data based on their assigned cluster, however this was deemed as too time consuming and the potential improvements not rewarding enough.

2.4 Conclusions

Ultimately, the training dataset (`train_users_2.csv`) was the only in use during the learning phase of the models, discarding the remaining datasets as they were deemed as data dumps (countries' demographics) or too time consuming (sessions) to clean and prepare, the problems it brought outweighed the possible improvements gained. Within the train set, we would look into engineering features from the devices, browsers and dates. We were also to drop `date_first_booking` feature as the problem involved predicting users who have not yet booked, thus could lead to overfitting (this column will most likely be empty for testing sets, or not present at all).

3 DATA PREPARATION

This section outlines the pre-processing techniques applied to the train set. The overview of our pipeline can be seen in Figure 6.

3.1 Data Cleaning

The dataset was relatively clean, no duplicate id's entries were found and empty values were only found in 3 of the columns. This lead to us ensuring those missing values were all of the same type: `np.nan`, as they were marked as '-unknown-' or 'NaN'.

3.2 Feature engineering

One-hot encoding raw features produced an extremely sparse training set, mainly from the larger number of values in `first_browser` feature (52 options, some used significantly infrequently). There was found to be values containing the mobile version of the browser, which was excess data as user's hardware type was provided in another feature. Casting the browser with a relative frequency of $< 0.5\%$ to `other` and mobile browsers to the desktop version (e.g. `Chrome_mobile` to `Chrome`), we reduced the values from 52 to 5.

We also engineered features from `first_device_type` by splitting it into two columns: hardware type and manufacturer. As the original contained both in one string, making it hard for the machine to understand, as well as providing more useful information (e.g.: `Mac_Desktop` split into `Mac` and `Desktop`).

Regarding date features, they were engineered into 4 other features:

- `year`: year of the date e.g. 2020, 2021
- `month`: numerical representation of the month e.g Janurary = 01, August = 08
- `day`: numerical representation of the day of the month e.g 04/05/2021 = 04
- `dayofweek`: numerical representation of the day of the week e.g. Monday = 01, Friday = 05

Encoding the date features in this way will allow, for potential patterns or seasonality to be inferred from the models.

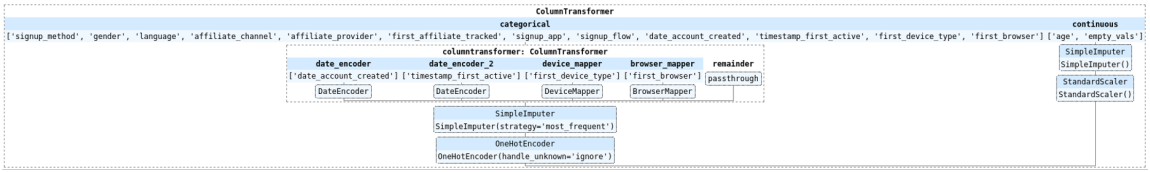


Figure 6: Pre-processing Pipeline

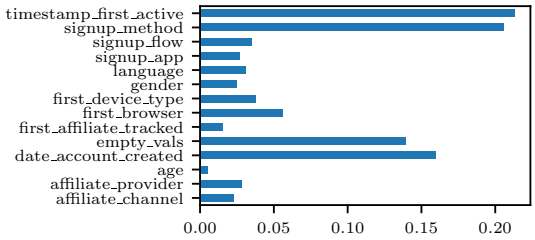


Figure 7: Feature importance plot

The final engineered feature was a simple column which counted the number of missing values per entry, as it was found there was high correlation between missing values and the most popular class 'NDF', during the exploration phase.

3.3 Feature Selection & Importance

As previously mentioned in subsection 2.4, the timestamp of first booking feature was dropped, additionally to this, the `id` feature was also dropped as it was redundant (only used to identify users, no substantial information obtained from it).

To identify which features were supplying the most information, we plotted a feature importance chart: Figure 7 (using our best model `XGBClassifier`). The engineered date features were amongst the most important, as it is a time-sensitive problem, this was expected (e.g. different months of the year influence due to seasons, weekends v weekdays, only create account to book...). Based on our analysis, 81% of users with 'Google' as `signup_method` had an 'NDF' country destination, whereas for the other two options, it was much more balanced. Therefore, as 'NDF' is the largest class, it is relatively easy to predict for this value from this feature. Finally, the feature `missing_value_counts` was also highly important, likely because users may fill in less information when they are just browsing ('NDF') or only Airbnb requires certain information to book.

3.4 Data Transformations

Depending on the data type, the features were dealt with accordingly. Most features were found to be categorical, with only `age` and the empty values counter (`empty_vals`) deemed as continuous.

3.4.1 Categorical. For features of this type, we applied a `Column Transformer` with the following steps:

(1) `Simple Imputer` for imputing missing values using the mode - cannot produce a mean or median as they are discrete classes
(2) `One Hot Encoding` to encode values, so no relationship between values can be inferred by the model (as compared to using ordinal encoding)

3.4.2 Continuous. For this type, we applied a `Column Transformer` with the following steps:

(1) `Simple Imputer` for imputing missing values using the mean
(2) `Standard Scaler` to scale the features - standardised features by removing the mean and scaling unit variance:

$$z = \frac{x - \mu}{\sigma}$$

where:

m = mean
σ = standard deviation

3.5 Dimensionality Reduction

We looked into investigating how performing PCA on our cleaned dataset would affect our most effective models' performance. However, as most of the data is categorial and `One Hot Encoded`, we opted to use `Truncated SVD` as it works efficiently with sparse matrices. We aimed to keep 50% most effective dimensions. The results of this are found in section 5.

3.6 Balancing Data

After analysing the dataset it was clear the classes were imbalanced, as seen in Figure 1. Thus we explored running our best performing model on a rebalanced dataset. The two ways we rebalanced our dataset were as follows:

- **Random under-sampling:** Random under-sampling removes samples from the majority class, with or without replacement. However, it can lead to increase in variance of the classifier and can degrade results due to discarding potentially useful or important data.
- **Random over-sampling:** Random over-sampling involves supplementing the training data with multiple copies of some of the minority classes. However, this can lead to dramatically increased dataset size.

The results can be found in section 5.

4 PREDICTIVE MODELS

As the problem in hand is a supervised (labelled data to train on), multi-class classification problem, thus we considered models appropriate for this. Furthermore, it required to obtain the top 5 predictions (for the metric), so models able to to predict probabilities for each class was another requirement. Following are the different model types we tried:

Linear - Used as a starting point to gain familiarity with the data and how to work with the pipeline. Scikit-learn's `LogisticRegression` [6] model was used with the parameters set to work with a multi-class classification problem. Performed poorly, only marginally beating the baseline. This is expected as the data is not linearly separable, thus kernels would have been needed (rather, we decided to move on to different models).

Decision Trees - Supervised learning models able to work with multiple classes, good starting point before approaching more complex models, met our requirements for a model. Although easy to understand, they required a substantial amount of fiddling parameters to train, which did not seem like a good approach. Additionally, we found them to be overfitting based on the difference between training and validation scores. However the scores were more promising, therefore the decision was made to investigate Random Forests.

Forests - Basic model of a Random Forest provided in Python library scikit-learn [6], with `entropy` as splitting criterion (this decision was trivial, as choosing between the two options has no significant difference in the results [7]). The idea of building a collection of trees to decide the outcome predicted class reduces the overfitting observed in Decision Trees [4], an issue we came across with. Advanced Forest models were explored given the encouraging results obtained from the basic model:

- LightGBM - based on decision tree algorithms using gradient boosting, used for ranking and classification, thus an appropriate option. Popular choice for Kaggle data science competitions.
- AdaBoost - most popular boosting algorithm [7], used for classification rather than regression. Although it was originally for binary problems, is now able to work with multi-class problems; we used scikit-learn's model [6].
- XGBoost - an optimised algorithm based on parallel tree boosting [3] (rather than bagging like Random Forest algorithms), boosting is a standard when dealing with data mining problems, particularly classification problems [7].

Artificial Neural Network - Particularly, we used a feedforward one (i.e. information only moves forward): an MLP (Multilayer Perceptron classifier). This was due to its ability to distinguish non-linearly separable data [1], something we discovered our data was from analysis. As choosing a learning rate is an intricate task, we used an adaptive learning rate for its general outperformance over configured ones [5].

5 RESULTS & EVALUATION

The models' results can be found in Table 1.

With dimensionality reduction, as described in subsection 3.5, we ran a `RandomForest` model and obtained a lower `nDCG` value of 0.844, compared to without: 0.882. We had under 20 features, thus we did not advantage from performing dimensionality reduction. Nonetheless, it did reduce time taken for training by over 20 minutes, which is a notable benefit.

With the balancing of the dataset as described in subsection 3.6, we ran our best performing model on the under-sampled dataset, which had a low `nDCG` score of 0.327. We suspected this to be due to the large decrease in the size of our dataset. When ran on the over-sampled data we got a `nDCG` score of 0.979. This led us to run our top 3 best performing models on the over-sampled dataset. Refer to Table 2 for the results. The high values for the tree methods we suspected to be due to over-fitting, due to the large duplication of data from over-sampling, as the difference between the size of the majority class and the smallest minority class is extremely large.

We also explored building a more complex ensemble model from our top 3 models: `XGBoost`, `MLP` and `RandomForest` (the first and last model are themselves types of ensemble methods). It performed worst than our best model (0.88) and took almost an hour to run, in addition to it being harder to tune parameters. Nonetheless, we suspected it to be less overfitting than some of the previously used models due to the ensemble's robust nature, and so for future works, further investigation into complex ensemble methods should undoubtedly continue. Due to the increase in computational cost and complexity and decrease in available time left for project completion, any further ensemble methods explorations were halted and we decided to concentrate on evaluating and discussing our achieved results.

Table 1: Table of results

MODEL	nDCG (validation)	F1 Score (validation)
Baseline	0.807	-
DecisionTree	0.825	0.754
LightGBM	0.821	0.605
LogisticRegression	0.814	0.533
MLPClassifier	0.900	0.816
RandomForest	0.882	0.800
XGBClassifier	0.903	0.828

Table 2: Table of top 3 performing models on over-sampled dataset

MODEL	nDCG (validation)	F1 Score (validation)
MLPClassifier	0.484	0.664
RandomForest	0.996	0.99
XGBClassifier	0.979	0.955

5.1 Reflection

Overall, our tree-based models performed better than other models. These models are popular amongst the Kaggle community and its competitions, due to the high scores they obtain. Ensembles of trees often work well due to the combination of weak learners derived from the dataset, to make a good strong learner that is able to

generalise efficiently and well. The dataset provided allowed us to build these required highly uncorrelated weak learners, thus performing so satisfactorily. `XGBoost` and `RandomForest` work on this principle so performed the best as well as provided with the added benefit that they tend not to overfit. Yet our final submission to Kaggle was lower than the cross-validation scores we obtained. We suspect either the dataset to train on was not representative enough, or our implementation of `nDCG` may have not been entirely correct. This would have to be re-verified in future works. Similarly, the `F1 Score` on certain models was considerably higher than the `nDCG` one. We concluded this to be due to the former being a better metric when imbalanced class distribution exists.

Debatably, using the sessions dataset in addition may have improved our final scores. However based on our analysis and discussions, we concluded potential improvements achieved using this latter dataset too would have not been worth the amount of time it would have taken to get the data ready. This is because the sessions data needed to be condensed into single rows per user and perhaps use methods such as `TF-IDF` to deal with the action details, as well as necessary data cleaning and feature engineering. With the added problem of how only a small percentage of the sessions users are also present in the training set. Further investigation into the sessions dataset undoubtedly should be performed in future works.

We refrained from exhausting parameter tuning as we decided exploring and trying a range of techniques would be more beneficial, nonetheless, in future, techniques such as `GridSearch` would be an approach to try for this. Additionally, we considered experimenting with more complex Deep Learning models, but due to limited computation availability, we were unfortunately not able to. This is something else to be researched into in future works.

Ultimately, from a business-perspective, Airbnb do not lose much if any users are misclassified, as the user themself would easily be able to correct where they would like to book first, without Airbnb's guidance. Moreover, most users were predicted to be 'NDF', in both training and testing, making it hard for Airbnb to accommodate the website to the user, so techniques such as gathering additional information of the users, whether directly or indirectly would highly benefit the company with this first user booking problem. Another approach to these cases could be simply providing information on the most popular destinations, and thus using 'wisdom of the crowd'. Nevertheless, even if not used for predicting users' first booking destination, the datasets provided are rich enough to gain other insights into Airbnb users, potentially helping the company. As an example, most users' affiliate channel was `basic`, suggesting how Airbnb should improve their marketing strategies in other channels such as `semi-brand` and perhaps remove themselves from some which are significantly less present amongst users' data.

6 CONCLUSION

Rigorous data inspection unveiled interesting patterns amongst Airbnb users and highlighted weaknesses in the data quality. We were able to clean and engineer features, aiming to improve the data, as well as using state of the art techniques to pre-process the data. Exploring a range of models enabled us to learn a range of machine learning approaches used in industry and ultimately achieved an `nDCG` score of 0.86 with the XGBClassifier in the Kaggle competition, beating our initial baseline model. Given the time scope and experience, this is a rewarding result and most importantly, the exposure gained to a range of techniques will prove valuable for future works.

REFERENCES

[1] Mutasem Alsmadi, Khairuddin Omar, Shahrul Azman Mohd Noah, and Ibrahim Almarashdah. 2009. Performance Comparison of Multi-layer Perceptron (Back Propagation, Delta Rule and Perceptron) algorithms in Neural Networks. *2009 IEEE International Advance Computing Conference, IACC 2009*, 296 – 299. https://doi.org/10.1109/IADCC.2009.4809024

[2] Renesh Bedre. 2021. t-SNE in Python. https://www.reneshbedre.com/blog/tsne.html.

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug 2016). https://doi.org/10.1145/2939672.2939785

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, California.

[5] Russell D Reed; Robert J Marks. 1999. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks.* Bradford Book, Cambridge, Mass.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[7] Laura Raileanu and Kilian Stoffel. 2004. Theoretical Comparison between the Gini Index and Information Gain Criteria. *Annals of Mathematics and Artificial Intelligence* 41 (05 2004), 77–93. https://doi.org/10.1023/B:AMAI.0000018580.96245.c6

Student ID: 29466423
Username: rsb1g17
Full Name: Rohan Bungre

COMP6212 Computational Finance 2020/21

Solutions to the Bitcoin Protocol Assignment (25%)

1 What is the maximum number of Bitcoins? How is this calculated?

The maximum number of Bitcoins hardwired into the Bitcoin protocol is 21 million. Assuming that an average block is mined every 10 minutes and the initial 50BTC reward is halved every 4 years; the total number of Bitcoins can be calculated as the sum of a geometric series.

$$\sum_{n=0}^{\infty} \frac{blocks\_per\_4\_years \times initial\_reward}{2^n} = \sum_{n=0}^{\infty} \frac{210000 \times 50}{2^n} = \frac{210000 \times 50}{1 - \frac{1}{2}} = 21000000 \tag{1}$$

2 If, on average, it takes 10 minutes to mine a block, when will the last Bitcoin be created? When will 98% of Bitcoin be mined? How is this calculated?

Assuming it takes 10 minutes to mine a Bitcoin and the reward halves every 4 years starting at 50BTC, the last Bitcoin will be created in the year 2140, approximately 3rd January 2140. 98% of Bitcoin will be mined by the year 2032, approximately in 2030. These dates can be calculated using an exponential halving formula.

$$\frac{start\_value}{2^n} = end\_value \rightarrow n = \log_2 \frac{start\_value}{end\_value} \tag{2}$$

The last Bitcoin will be created in the year when the block reward falls below 1 Satoshi (1×10^{-8} BTC). This is because 1 Satoshi is the smallest currency in the Bitcoin Protocol. This will happen after $32.2 \approx 33$ halves, 132 years after the first block was mined in 2009, which is 2140.

$$n = \log_2 \frac{50}{1 \times 10^{-8}} = 32.2 \approx 33 \tag{3}$$

98% of Bitcoin will have been mined when the block reward falls to 2% of its original value. This will happen after $5.64 \approx 6$ halves, 24 years after the first block was mined in 2009, which is 2032.

$$n = \log_2 \frac{50}{50 \times 0.02} = 5.64 \approx 6 \tag{4}$$

3 Are the transactions on the Bitcoin network completely anonymous? Why?

Transactions on the Bitcoin network are not completely anonymous, they are in fact pseudo-anonymous meaning that transactions are not linked to real-world entities but rather bitcoin addresses. Owners of bitcoin addresses are not explicitly identified, but all transactions on the blockchain are public. Although the user's wallet address is generated randomly and can be changed per transaction, some retailers and wallet vendors will require personal information. This means that there may be at least one user address with a personal link. This personal link could also be in the form of an IP address. It is possible to cluster these different addresses used by a single user, meaning if just one address in the cluster is compromised anonymity is lost. Although transaction data is fed through a random set of network nodes, if several nodes within the network can be analysed, the combined data collected from these different nodes might be enough to determine where a transaction originated from. There have been attempts to anonymise the network using VPNs and Mixers, however the public ledger and network structures often prevent true anonymous behaviour.

4 Who governs Bitcoin? In other words, who defines the rules and writes the code? Briefly explain their roles and power.

Bitcoin is a decentralised system meaning there is no central authority controlling it. With no central servers and no central storage, the Bitcoin ledger is publicly distributed and maintained by a network of equally privileged miners. The Bitcoin source code and initial protocol was created as an open source project by Satoshi Nakamoto. Due to its open source nature anyone can create their own Bitcoin implementation that interacts with the network. There is a core team that maintain the most popular implementation 'Bitcoin Core' which runs on 96% of the network's nodes. This core team updates the software and removes bugs found within the code. These developers can improve the software but they can not force a change in the rules of the Bitcoin protocol. This is because if the team implement rules and features that are unpopular, users will move to another Bitcoin implementation. In order to stay compatible with each other, all users need to use software complying with the same rules. Bitcoin can only work well if there is a complete agreement on the software standard between all users. Therefore, all users and developers are encouraged to use and protect this standard. These maintainers don't actually have the power to make decisions that run against the consensus of the users. If there is a governing power over Bitcoin, it would be the Bitcoin users themselves.

5 What is double-spending? How does the Bitcoin network achieve consensus?

Double spending is a flaw within digital currencies where a digital token has the potential to be used in multiple transactions, meaning the same token could be spent multiple times by a single owner. This problem is prevented in cryptocurrencies such as Bitcoin that use a blockchain, using a consensus technique known as proof of work. This is done with a decentralised network of miners who validate the integrity of transactions on the blockchain, which in turn detects and prevent double spending. For a transaction to be valid and accepted onto the blockchain, a miner must collect it into a block, and calculate the corresponding nonce value. This is proof of work as it requires compute power to guess the nonce, verified by other users in the network. If a user attempts to double spend, it will be stopped as the original transaction will be on the public blockchain. To ensure the transaction is valid, it is advised to wait for 6 new blocks to be added to the blockchain so the transaction isn't lost. There are other vulnerabilities which could allow double spend attacks to take place. If an attacker is able to control at least 51% of the network, they can commit double spending as they could reverse transactions and create a separate blockchain which includes the double spend transaction.

6 What are SegWit2x and Lightning network? Explain their similarities and differences.

Both SegWit2x and Lightning network are attempts to fix the scalability issues of the Bitcoin protocol. SegWit2x was an unsuccessful hard fork, aimed to upgrade the bitcoin protocol in two ways. The first was to implement an increase to the volume of transactions that fit into each block without increasing the block size. The second involves updating the Bitcoin protocol rules to allow for 2MB blocks. Lightning network is a payment protocol that operates on top of existing blockchain protocols like Bitcoin. It successfully aims to increase the transaction speed on the Bitcoin blockchain by introducing off-ledger transactions. Because of this not every transaction on the channel has to be stored on the blockchain, only the start and ending payments. The similarities between SegWit2x and Lightning are that both methods aim to improve the transaction speed of the Bitcoin network; both aim to reduce the transaction fee; both are not created by the Bitcoin Core team; and both aim to fix the scalability issues of Bitcoin. The differences between SegWit2x and Lightning are that SegWit2x is a protocol rule change that involves a hard fork of the network whilst Lightning is a layer 2 protocol that works with Bitcoin Core; SegWit2x was a failed implementation whilst Lightning is operational and successful; and SegWit2x writes all transactions to the blockchain whilst Lightning only stores the initial and final transactions.

COMP3212 Major Assignment 2020

Using Machine Learning Techniques to Understand and Classify Lung, Breast and Skin Cancers

Rohan Bungre - rsb1g17 - 29466423

Carl Richardson - cr2g16 - 28544285

Lewis O'Shaughnessy - lo2g17 - 21939864

(Three Persons Project)

A report presented for the module of
Computational Biology

Department of Electronics and Computer Science
University of Southampton
28/05/2020

1 Introduction

The purpose of this project was to find a subset of genes for classifying which cancer- known to be skin, lung or breast, a sample cell was from. The identified genes were used to establish a more computationally efficient classifier; as well as focus the scope of study for cancer researchers. To achieve this goal, three data sets were identified which contained samples of each cancer type. These were used to train clustering, feature selection, dimensionality reduction and classification models.

The clustering model was used to prove the data was sufficiently separated to allow for good classification. The feature selection model filtered out the genes which contained redundant information whereas, the dimensionality reduction model transformed the data set into an optimal, lower dimensional, space for separating the classes. Feature selection and dimensionality reduction models were used as pre-processing steps for improving efficiency and performance of the classification model. A variety of classification algorithms were used for validating the performance. The classification performance with and without the pre-processing steps were compared against each other. From this, a good classification algorithm was found, and the effectiveness of each pre-processing step was evaluated.

2 The Data

The datasets were selected from the UCSC archive (1). Each dataset took the form of a gene expression matrix with the same 21,816 genes. Each column represented the gene expression levels from a single experiment (cancer cell sample) and each row represented the expression of a gene across all experiments. Each element of the matrix was a base 2 log of the gene expression ratio denoted by equation 1.

$$\begin{aligned} y &= \log_2 (\text{Gene Expression Ratio}) \\ \text{Gene Expression Ratio} &= \frac{T}{R} \end{aligned} \tag{1}$$

The numerator, T, is the gene expression level of the testing sample (cancer cell) and the denominator, R, is the gene expression level of the reference sample (normal cell). The value of y was interpreted as equation 2 shows. A gene expression can be regulated by a wide range of mechanisms used, by cells, to increase or decrease the production of specific gene products (protein or RNA). When a gene was up-regulated, its production increased, with respect to the reference, while the opposite was true for down-regulation.

$$\begin{aligned} y = 0 &: \text{represented no change i.e. } T = R \\ y > 0 &: \text{represented upregulation i.e. } T > R \\ y < 0 &: \text{represented downregulation i.e. } T < R \end{aligned} \tag{2}$$

To form the input data matrix, 95 skin, 130 lung and 51 breast cancer samples were obtained from (2) (3) (4). Each sample was labelled by its cancer type and contained the log of the gene expression ratio for 21,816 genes. Hence, the input data matrix had 276 rows and 21,816 columns. When set up in this way, the genes were the features of the data matrix as opposed to the samples.

3 Clustering

Clustering is an unsupervised process that allows for patterns to be found within a dataset. The aim was to be able to find a distinction between lung, skin and breast cancer. Collating each cancer dataset into one, a range of clustering algorithms (5) can be used to see if there was a separation between the three types of cancers. Table 1 shows the clustering analysis.

Clustering algorithms that use a euclidean approach such as Mini Batch K Means, Agglomerative and Birch were able to find three clusters, each relating to one of the cancers. Algorithms that use a hierarchical approach such as DBSCAN and OPTICS, struggle to find a separation. This shows that the dataset will allow for classification of a cancerous cell into one of three classes.

Clustering Algorithm	**Number of Clusters Found**
Mean Shift	1
Mini Batch K Means	3
Agglomerative	3
DBSCAN	1
OPTICS	1
Affinity Propagation	24
Birch	3

Table 1: Number of clusters found using each clustering algorithm

Even though the dataset shows separation between the cancers, feature selection and feature reduction performed on the dataset would further aid the classification of cancerous cells.

4 Feature Selection

A linear support vector classification (SVC) model (5), penalised by its L1- norm, was used as the feature selection model. This classification model penalises weight vectors with larger norms; hence, it tends to find sparse solutions. For feature selection, the features corresponding to the non-zero elements of the solution were interpreted as the features important for classifying the labelled data whereas the features corresponding to zero elements of the solution were considered redundant.

The linear SVC was trained for a range of regularisation parameter values. The number of features selected by each model is shown in figure 1. This figure shows that as the regularisation parameter was increased, so was the number of features selected.

The mean classification accuracy was then computed for each model. This is shown by figure 1, where the number of features was plotted against the mean classification accuracy. The minimum number of components, which achieved the maximum classification accuracy of 0.82, was 64. The mean accuracy decreased gradually between 64 and 14 features where the classification accuracy was 0.72. When the number of features was less than 14, the classification accuracy decreased rapidly.

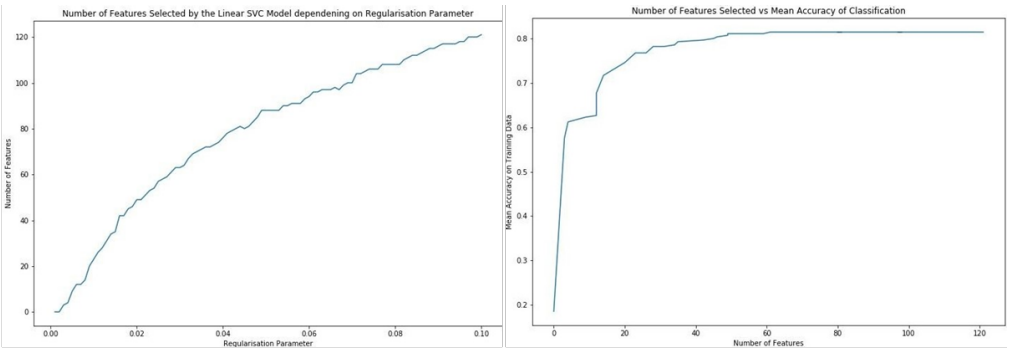


Figure 1: Number of features selected dependant on the regularisation parameter, Number of features selected vs classification accuracy

The 23 most informative genes corresponding to the solution found using a regularisation parameter value of 0.01 was used to reduce the number of features used during classification of the cancers. This model had a classification accuracy of 0.77. The names of the 23 most informative genes found by feature selection were:

201820_at, 206153_at, 206276_at, 206376_at, 209125_at, 209278_s_at, 209810_at, 210437_at, 21173_x_at, 213240_s_at, 214877_at, 215145_s_at, 215621_s_at, 216623_x_at, 217626_at, 219140_s_at, 219554_at, 220057_at, 220414_at, 220779_at, 221728_x_at, 221854_at, 37004_at.

5 Feature Reduction

Unlike feature selection, where the best N features are selected, dimensional reduction aims to reduce the size of the feature set whilst preserving the information within data. Reducing the number of features has many benefits. These include, removing redundant correlated features that do not help describe the data and speeding up computations.

Principal Component Analysis (PCA) is a technique where features, are transformed into a new set of features, which are linear combination of original features. These new set of features are known as principle components. These become the new set of axis for the data in a lower dimension.

PCA uses the idea that by maximising the variance of the new features, the amount of information retained from the original data set is maximised. Principle components are obtained in such a way that first principle component accounts for most of the possible variation of original data after which each succeeding component has the next highest possible variance. Figure 2 shows that the first 250 principle components can explain 99% of the original 21816 features.

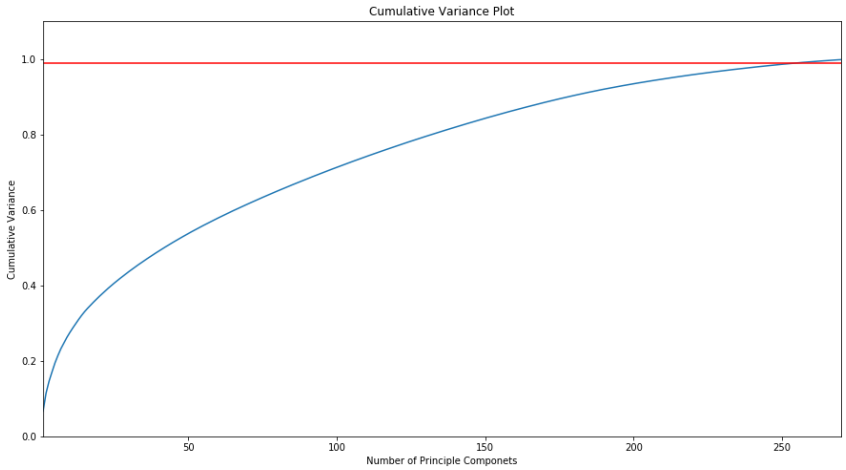


Figure 2: Number of principle components selected vs data accuracy

6 Classification

Classification is necessary to be able to assign labels to unknown samples. The goal of the classification is to assign labels of the type of cancer they most closely describe, hence the labels will be “skin”, “breast” and “lung”.

By splitting the dataset into training and testing sets, the classifiers can be initially trained to provide a more accurate result. The percentage of training data is set to 40% of the total dataset, this can be changed however was found to be a good middle ground between good results and a good number of usable results, as too much training tends to lead to less results.

To this end, a good supervised classifier was needed. A suite of classifiers (5) and their respective accuracy scores was used to determine the most successful classifier for this task. Each classifier returns a percentage score of how accurate it classifies the samples. The suite was run on the entire original dataset of all three cancers, the feature selection dataset and the feature reduced dataset, providing the accuracy table 2.

Classifier Algorithm	**Original Dataset Accuracy**	**Feature Selection Accuracy**	**Feature Reduction Accuracy**
Nearest Neighbor	54.1%	84.7%	32.4%
Linear SVM	38.7%	80.2%	19.8%
RBF SVM	44.1%	62.2%	45.1%
Gaussian Process	44.1%	82.9%	36.0%
Decision Tree	91.0%	85.6%	68.5%
Random Forest	82.9%	96.4%	68.5%
Neural Net	47.7%	91.0%	44.1%
AdaBoost	81.1%	94.6%	72.1%
Naive Bayes	99.1%	98.2%	92.8%
QDA	29.7%	86.5%	21.6%

Table 2: The accuracy of each classifier algorithm using different dataset formations

Much like the clustering predicted, the original dataset could be used to classify a cancerous cell. It performed averagely, with some classifier algorithms such as AdaBoost, Decision Tree, Random Forest and Naive Bayes performing admirably.

The feature selection dataset, which only included the 23 informative genes, performed extremely well when being used by all but one classifier algorithms with the highest consistent accuracy rates. This shows that there are features within the original dataset that hinder the classification of cancerous cells.

The feature reduction dataset, consisting of the 250 largest principle components, performed much worse than expected. Using PCA was supposed to remove unwanted and useless features, however performed worse than the original dataset.

One possible reason for a poor result could be the assumption that features with high variance equals more information is wrong. Another reason could be the assumption of the statistical importance of the mean and covariance of the data. PCA uses the eigenvectors of the covariance matrix which finds the independent axes of the data under a Gaussian assumption. In the case of non-Gaussian data, PCA simply de-correlates the axes. There is no guarantee that the directions of maximum variance will contain good features for the separation of cancers (6).

The best classifier algorithm was found to be Naive Bayes. It had an accuracy within the 90% range for each dataset. Naive Bayes classifiers have worked quite well in many complex real-world situations. “An analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of Naive Bayes classifiers” (7). An advantage of Naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification (8).

7 Conclusion

In conclusion, the informative genes important for classifying lung, skin and breast cancer were found using feature selection techniques. The identified genes were successfully used to establish a more computationally efficient classifier; as well as focus the scope of study for cancer researchers. Before continuing with this line of research, these results should be evaluated from a biological perspective. This will determine whether the interpretation of the results fits with established work in the field.

The best classifier algorithm for the cancer dataset was found to be Naive Bayes. It had an accuracy of 96.7%, due to the small amount of training it requires. If more training data was available the other classifiers such as Random Forest, Decision Trees and AdaBoost may have been alternative options. Future work could look into how the classifier treats other cancerous cells, outside its scope. This could expose the weaknesses of Naive Bayes.

Using PCA as the feature reduction technique was found to a poor decision. This was due to its reliance on statistical assumptions, that might not manifest themselves in biological systems. Future work could look into using supervised approaches such as Fishers LDA, that do not rely so much on statistical inference.

References

[1] "Cancer database," 2020. [Online]. Available: https://xenabrowser.net/datapages/

[2] "Breast cancer dataset," 2020. [Online]. Available: https://xenabrowser.net/datapages/?dataset=ucsfNeve_public%2FucsfNeveExp_genomicMatrix&host=https%3A%2F%2Fucscpublic.xenahubs.net&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443

[3] "Skin cancer dataset," 2020. [Online]. Available: https://xenabrowser.net/datapages/?dataset=lin2008_public%2Flin2008Exp_genomicMatrix&host=https%3A%2F%2Fucscpublic.xenahubs.net&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443

[4] "Lung cancer dataset," 2020. [Online]. Available: https://xenabrowser.net/datapages/?dataset=raponi2006_public%2Fraponi2006_genomicMatrix&host=https%3A%2F%2Fucscpublic.xenahubs.net&removeHub=https%3A%2F%2Fxena.treehouse.gi.ucsc.edu%3A443

[5] "Sklearn library," 2020. [Online]. Available: https://scikit-learn.org/stable/index.html

[6] "Why is principal component analysis a bust for genetics?" 2020. [Online]. Available: https://amethix.com/why-is-principal-component-analysis-a-bust-for-genetics/?fbclid=IwAR1Nv-17cy2PrB2ZKHeK8AMQG90DJT8DvINQz9UW7-qT2QqHQCEWoBnHfJ4

[7] "The optimality of naive bayes," 2020. [Online]. Available: http://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf

[8] "Naive bayes classifier 2020," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_classifier

8 Appendix

This appendix contains the relevant code and results for proof of the work in this report.

Tutorial 1 - Rohan Bungre - rsb1g17 - 29466423

Week 1

Task 1

Import the cell cycle dataset excel spreadsheet (using Pandas). You may need to do some tidying of the data such as dropping rows with missing NaN values.

In [1]:

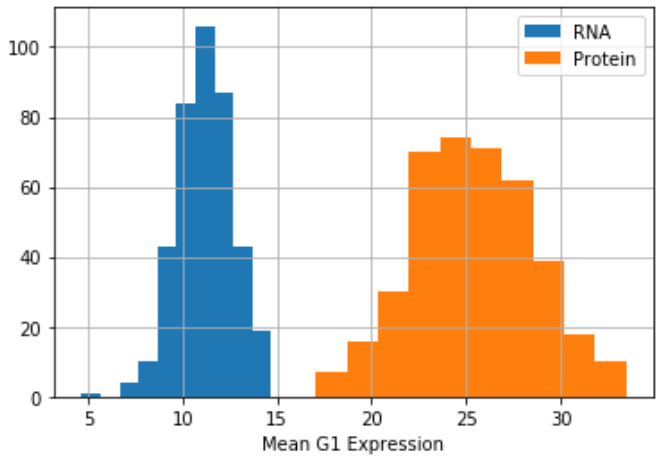
```
import pandas as pd

path = "C:/Users/rohan/Documents/UNI/COMPBIO/"
df = pd.read_excel(path + 'Cell-Cycle-Set.xlsx')
df.dropna(inplace=True)
```

Task 2

Generate a histogram of one of the cell cycle stages of the RNA and protein distribution. Do you notice anything interesting with regards to the mean/variance of the distribution?

In [2]:

```
import matplotlib.pyplot as plt

df.mean_RNA_G1.hist(label='RNA')
df.mean_protein_G1.hist(label='Protein')

plt.legend()
plt.xlabel('Mean G1 Expression')
plt.show()
```



In the G1 stage the cell grows in size and synthesizes mRNA and proteins that are required for DNA synthesis. It is interesting that the mean concentraion of the RNA is less than the protein. As mRNA codes for prtein I would expect them to be equal. This must be due to each RNA producing more than one protein during the cycle.

Task 3

Look at the pairwise correlations between each of the RNA/protein columns (this can be achieved using the corr() function). Does the change in timestep have much effect on the relationship(s) between RNA and protein?

In [3]:

```
df.corr()
```

Out[3]:

	mean_RNA_G1	mean_RNA_S	mean_RNA_G2	mean_protein_G1	mean_pro
mean_RNA_G1	1.000000	0.991063	0.992023	0.522658	0.5
mean_RNA_S	0.991063	1.000000	0.986836	0.514705	0.5
mean_RNA_G2	0.992023	0.986836	1.000000	0.510364	0.5
mean_protein_G1	0.522658	0.514705	0.510364	1.000000	0.9
mean_protein_S	0.541428	0.536190	0.529690	0.970289	1.0
mean_protein_G2	0.544206	0.534322	0.532565	0.977016	0.9

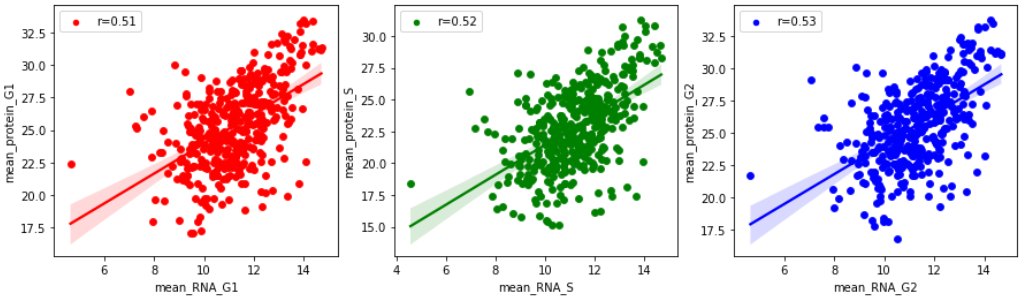
The change in timestep does not have much effect in the relations between RNA and Proteins. The values remain around 0.52-0.54

Task 4

Generate a scatterplot of the RNA versus. protein for each cell cycle stage. Fit a linear model to the data, can we infer protein concentration from RNA concentration?

In [4]:

```
from scipy.stats import spearmanr
import seaborn as sns

fig3,ax3 = plt.subplots(ncols=3, figsize=(15,4))

df.plot.scatter('mean_RNA_G1', 'mean_protein_G1', ax=ax3[0],color='r', label='r={:0.2f}
'.format(
            spearmanr(df.mean_RNA_G1.values, df.mean_protein_G1.values)[0]))

sns.regplot(df['mean_RNA_G1'],df['mean_protein_G1'], ax=ax3[0], color='r')

df.plot.scatter('mean_RNA_S', 'mean_protein_S', ax=ax3[1], color='g', label='r={:0.2f}'
.format(
            spearmanr(df.mean_RNA_S.values, df.mean_protein_S.values)[0]))

sns.regplot(df['mean_RNA_S'],df['mean_protein_S'], ax=ax3[1],color='g')

df.plot.scatter('mean_RNA_G2', 'mean_protein_G2', ax=ax3[2], color='b', label='r={:0.2
f}'.format(
            spearmanr(df.mean_RNA_G2.values, df.mean_protein_G2.values)[0]))

sns.regplot(df['mean_RNA_G2'],df['mean_protein_G2'],ax=ax3[2],color='b')

plt.show()
```



From the diagrams we can imfer protein concentration due to the correlation value of around 0.5. Correlation coefficients whose magnitude are between 0.5 and 0.7 indicate variables which can be considered moderately correlated. We can infer a moderate positive correlation between RNA and Protein concentrations.

Week 2

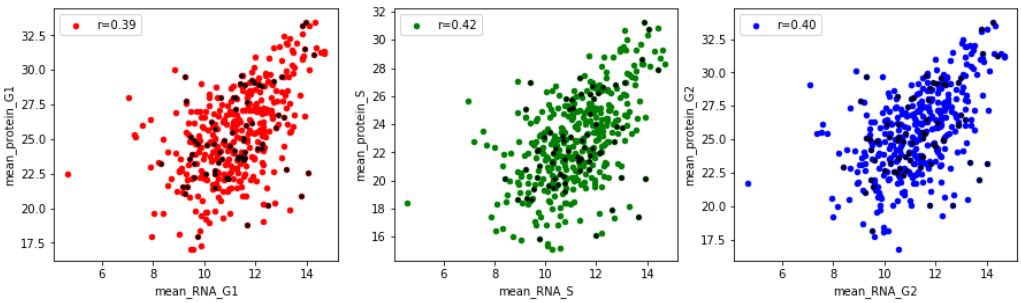
Task 1

Find all genes that contain 'cell cycle' in their GOBP term and plot them as a scatterplot (with different colour) overlaid across all genes for each cell cycle phase. Is there a stronger/weaker correlation?

In [5]:

```
gobp = df[df.GOBP.str.contains('cell cycle')]

fig4,ax4 = plt.subplots(ncols=3, figsize=(15,4))

df.plot.scatter('mean_RNA_G1', 'mean_protein_G1', ax=ax4[0], color='r',label='r={:0.2f}
'.format(
            spearmanr(gobp.mean_RNA_G1.values, gobp.mean_protein_G1.values)[0]))

ax4[0].scatter(gobp.mean_RNA_G1, gobp.mean_protein_G1, color='k', s=10.)

df.plot.scatter('mean_RNA_S', 'mean_protein_S', ax=ax4[1], color='g',label='r={:0.2f}'.
format(
            spearmanr(gobp.mean_RNA_S.values, gobp.mean_protein_S.values)[0]))

ax4[1].scatter(gobp.mean_RNA_S, gobp.mean_protein_S, color='k', s=10.)

df.plot.scatter('mean_RNA_G2', 'mean_protein_G2', ax=ax4[2], color='b',label='r={:0.2f}
'.format(
            spearmanr(gobp.mean_RNA_G2.values, gobp.mean_protein_G2.values)[0]))

ax4[2].scatter(gobp.mean_RNA_G2, gobp.mean_protein_G2, color='k', s=10.)

print(len(gobp))
print("Ignore colour of legend. The correlation refers to the black GOBP data points")
```

```
71
Ignore colour of legend. The correlation refers to the black GOBP data poi
nts
```



There is a weaker correlation of around 0.4 when the cells with the GOBP terms are plotted.

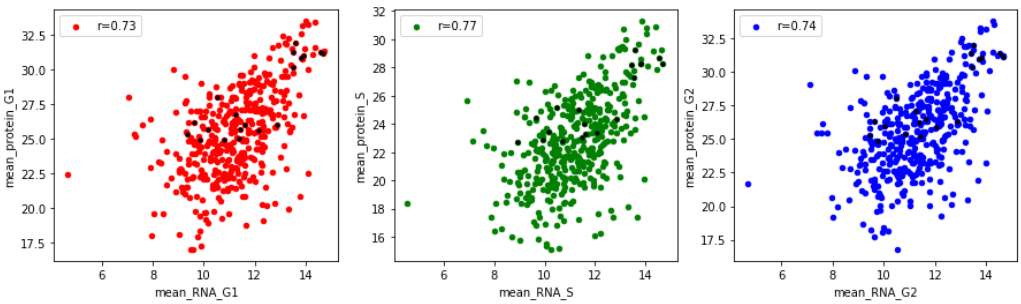
Task 2

Repeat task 1 by finding genes that contain 'ribosome' in their GOCC term.

In [6]:

```
gocc = df[df.GOCC.str.contains('ribosome')]

fig5,ax5 = plt.subplots(ncols=3, figsize=(15,4))

df.plot.scatter('mean_RNA_G1', 'mean_protein_G1', ax=ax5[0], color='r',label='r={:0.2f}
'.format(
            spearmanr(gocc.mean_RNA_G1.values, gocc.mean_protein_G1.values)[0]))

ax5[0].scatter(gocc.mean_RNA_G1, gocc.mean_protein_G1, color='k', s=10.)

df.plot.scatter('mean_RNA_S', 'mean_protein_S', ax=ax5[1], color='g',label='r={:0.2f}'.
format(
            spearmanr(gocc.mean_RNA_S.values, gocc.mean_protein_S.values)[0]))

ax5[1].scatter(gocc.mean_RNA_S, gocc.mean_protein_S, color='k', s=10.)

df.plot.scatter('mean_RNA_G2', 'mean_protein_G2', ax=ax5[2], color='b',label='r={:0.2f}
'.format(
            spearmanr(gocc.mean_RNA_G2.values, gocc.mean_protein_G2.values)[0]))

ax5[2].scatter(gocc.mean_RNA_G2, gocc.mean_protein_G2, color='k', s=10.)

print(len(gocc))
print("Ignore colour of legend. The correlation refers to the black GOCC data points")
```

```
19
Ignore colour of legend. The correlation refers to the black GOCC data poi
nts
```



There is a stronger correlation of around 0.75 when the cells with the GOCC terms are plotted. Hoever this coulf be due to the lower number of terms plotted. This could be a false postive correlation. There needs to be a larger sample size to be sure.

Task 3

Count the number of occurrences of every GOBP term across all genes, what are some of the difficulties that arise when using these terms?

In [7]:

```
print(df.GOBP.str.split(';',expand=True).stack().value_counts())
```

```
cellular process                                                 377
metabolic process                                                273
cellular metabolic process                                       260
primary metabolic process                                        255
biological regulation                                            236
                                                                ...
bile acid biosynthetic process                                     1
positive regulation of estrogen receptor signaling pathway         1
carnitine metabolic process                                        1
pyridoxal 5'-phosphate salvage                                     1
low-density lipoprotein particle receptor catabolic process        1
Length: 2854, dtype: int64
```

The GO terms are human defined they do not serve as an accurate cell process. There are also so many terms, finding correlation would require a very large data set where the number of data points as much higher than the number of GO features.

Task 4

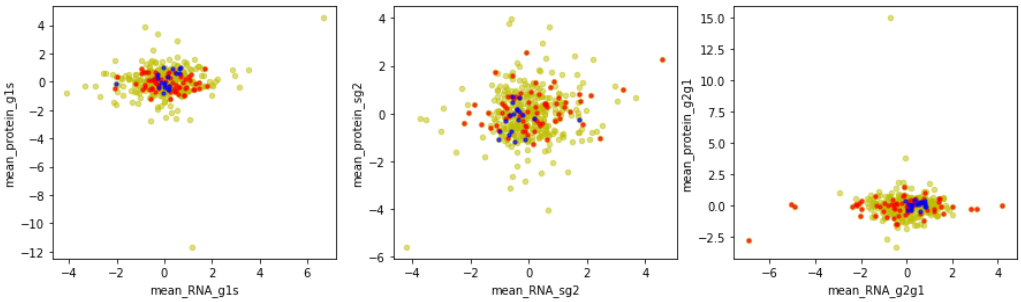
Calculate the change in mRNA/protein level across the cell cycle by taking the difference at each stage (G1-S, S-G2, G2-G1), and standardize the differences by mean-centering and variance scaling. Repeat tasks 1 and 2 by plotting the changes in levels with GOBP/GOCC labelling. What do we notice about changes in the cell cycle? Is there any apparent clustering of GO terms?

In [8]:

```
df['mean_RNA_g1s'] = (df.mean_RNA_S - df.mean_RNA_G1)
df['mean_RNA_sg2'] = (df.mean_RNA_G2 - df.mean_RNA_S)
df['mean_RNA_g2g1'] = (df.mean_RNA_G1 - df.mean_RNA_G2)
df['mean_protein_g1s'] = (df.mean_protein_S - df.mean_protein_G1)
df['mean_protein_sg2'] = (df.mean_protein_G2 - df.mean_protein_S)
df['mean_protein_g2g1'] = (df.mean_protein_G1 - df.mean_protein_G2)

# standardise
df.iloc[:,-6:] = (df.iloc[:,-6:] - df.iloc[:,-6:].mean(axis=0)) / df.iloc[:,-6:].std(ax
is=0)

gobp = df[df.GOBP.str.contains('cell cycle')]
gocc = df[df.GOCC.str.contains('ribosome')]
fig6,ax6 = plt.subplots(ncols=3, figsize=(15,4))

df.plot.scatter('mean_RNA_g1s', 'mean_protein_g1s', ax=ax6[0], color='y', alpha=.5)
df.plot.scatter('mean_RNA_sg2', 'mean_protein_sg2', ax=ax6[1], color='y',alpha=.5)
df.plot.scatter('mean_RNA_g2g1', 'mean_protein_g2g1', ax=ax6[2], color='y', alpha=.5)

ax6[0].scatter(gobp.mean_RNA_g1s, gobp.mean_protein_g1s, color='r', s=10., alpha=.7)
ax6[1].scatter(gobp.mean_RNA_sg2, gobp.mean_protein_sg2, color='r', s=10., alpha=.7)
ax6[2].scatter(gobp.mean_RNA_g2g1, gobp.mean_protein_g2g1, color='r', s=10., alpha=.7)
ax6[0].scatter(gocc.mean_RNA_g1s, gocc.mean_protein_g1s, color='b', s=10., alpha=.7)
ax6[1].scatter(gocc.mean_RNA_sg2, gocc.mean_protein_sg2, color='b', s=10., alpha=.7)
ax6[2].scatter(gocc.mean_RNA_g2g1, gocc.mean_protein_g2g1, color='b', s=10., alpha=.7)
```

Out[8]:

```
<matplotlib.collections.PathCollection at 0x225c48ab288>
```



The GOCC terms are always clustered in the center and do not change value. This is because they are cellular components and do not change through the cell cycle. GOBP representing biological processes are more spread out but are still clustered. This alludes to the conclusion that genes with certain terms will behave in an expected manner when it comes to RNA and Protein concentration levels.

COMP3212 Tutorial 2: Weeks 3-5 2019/20

Rohan Bungre

29466423

1 Needleman-Wunsch Implementation for Protein Base Matching

Using the Needleman-Wunsch algorithm with the Blosum50 scoring matrix, protein sequences could be analysed and global comparisons could be made between different proteins.

Matching the protein sequence **HEAGAWGHEE** with **PAWHEAE** yielded the result shown in table 1.

	*	H	E	A	G	A	W	G	H	E	E
*	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
P	-8	-2	-9	-17	-25	-33	-41	-49	-57	-65	-73
A	-16	-10	-3	-4	-12	-20	-28	-36	-44	-52	-60
W	-24	-18	-11	-6	-7	-15	-5	-13	-21	-29	-37
H	-32	-14	-18	-13	-8	-9	-13	-7	-3	-11	-19
E	-40	-22	-8	-16	-16	-9	-12	-15	-7	3	-5
A	-48	-30	-16	-3	-11	-11	-12	-12	-15	-5	2
E	-56	-38	-24	-11	-6	-12	-14	-15	-12	-9	1

Table 1: Needleman-Wunsch matching matrix

Using a back-tracking algorithm the optimal score was found to be **1** with a matched sequence of:

HEAGAWGHE_E
_PA__W_HEAE

Although this is not the same as the answer found on page 23 of lecture 5, it is still an optimal solution as the are several optimal paths for the backtracking algorithm to take.

Match the protein sequence **SALPQPTTPVSSFTSGSMLGRTDTALTNTYSAL** with **PSPTMEAVTSVEASTASHPHSTSSYFATTYYHLY** yielded the result shown partially in table 2

	*	S	A	L	P	Q	P	T	T
*	0	-8	-16	-24	-32	-40	-48	-56	-64
P	-8	-1	-9	-17	-14	-22	-30	-38	-46
S	-16	-3	0	-8	-16	-14	-22	-28	-36
P	-24	-11	-4	-4	2	-6	-4	-12	-20
T	-32	-19	-11	-5	-5	1	-7	1	-7
M	-40	-27	-19	-8	-8	-5	-2	-7	0
E	-48	-35	-27	-16	-9	-6	-6	-3	-8
A	-56	-43	-30	-24	-17	-10	-7	-6	-3
V	-64	-51	-38	-29	-25	-18	-13	-7	-6
T	-72	-59	-46	-37	-30	-26	-19	-8	-2

Table 2: Partial Needleman-Wunsch matching matrix

Using a back-tracking algorithm the optimal score was found to be **7** with a matched sequence of:

SALPQPTTPVSSFTSGSMLGRTDTALTNTYSAL
PSPTMEAVTSVEA_STASHPHSTSSYFATTYYHLY

2 Smith-Waterman Implementation for Protein Base Matching

Similarly, using the Smith-Waterman algorithm with the Blosum50 scoring matrix, protein sequences could be analysed and local comparisons could be made between different proteins.

Matching the protein sequence **HEAGAWGHEE** with **PAWHEAE** yielded the result shown in table 3.

	*	H	E	A	G	A	W	G	H	E	E
*	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
P	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	5	0	5	0	0	0	0	0
W	0	0	0	0	2	0	20	12	4	0	0
H	0	10	2	0	0	0	12	18	22	14	6
E	0	2	16	8	0	0	4	10	18	28	20
A	0	0	8	21	13	5	0	4	10	20	27
E	0	0	6	13	18	12	4	0	4	16	26

Table 3: Smith-Waterman matching matrix

Using a back-tracking algorithm the optimal score was found to be **28** with a matched sequence of:

AWGHE
AW_HE

This matches the result found on page 5 of lecture 6.

Matching the protein sequence **MQNSHSGVNQLGGVFVNGRPLPDSTRQKIVE-LAHSGARPCDISRILQVSNGCVSKILGRY** with **TDDECHSGVNQLGGVFVG-GRPLPDSTRQKIVELAHSGARPCDISRI** yielded the results shown partially in table 4.

	*	M	Q	N	S	H	S	G	V
*	0	-8	-16	-24	-32	-40	-48	-56	-64
T	0	0	0	0	0	0	0	0	0
D	0	0	0	2	0	0	0	0	0
D	0	0	0	2	2	0	0	0	0
E	0	0	2	0	1	2	0	0	0
C	0	0	0	0	0	0	1	0	0
H	0	0	1	1	0	10	2	0	0
S	0	0	0	2	6	2	15	7	0
G	0	0	0	0	2	4	7	23	15
V	0	1	0	0	0	0	2	15	28
N	0	0	1	7	1	1	1	7	20

Table 4: Partial Smith-Waterman matching matrix

Using a back-tracking algorithm the optimal score was found to be **273** with a matched sequence of:

HSGVNQLGGVFVNGRPLPDSTRQKIVELAHSGARPCDISRI
HSGVNQLGGVFVNGRPLPDSTRQKIVELAHSGARPCDISRI

3 Testing the BLAST algorithm

The BLAST algorithm increases the speed of alignment by decreasing the search space or number of comparisons it makes when compared to other dynamic programming techniques.

Using the BLAST algorithm to compare the Pax6 protein for the mouse with the eyeless protein for the fruit fly, found four sections of potential sequence matches. Section 1 showed the largest matching sequence.

Section 1: 61 to 101, Identities: 40/41(98%), Positives: 40/41(97%), Gaps: 0/41(0%)

Mouse: **HSGVNQLGGVFVNGRPLPDSTRQKIVELAHSGARPCDISRI**
Fly: **HSGVNQLGGVFVGGRPLPDSTRQKIVELAHSGARPCDISRI**
Match: **HSGVNQLGGVFV_GRPLPDSTRQKIVELAHSGARPCDISRI**

Section 2: 26 to 55, Identities: 8/30(27%), Positives: 11/30(36%), Gaps: 0/30(0%)

Mouse: **PQPTTPVSSFTSGSMLGRTDTALTNTYSAL**
Fly: **PSPTMEAVEASTASHPHSTSSYFATTYYHL**
Match: **P_PT___++_S____T_+____TY__L**

Section 3: 35 to 54, Identities: 8/20(40%), Positives: 13/20(65%), Gaps: 1/20(5%)

Mouse: **SNTPSHIPISSS-FSTSVYQ**
Fly: **ASTASHPHSTSSYFATTYYHL**
Match: **++T_SH___+SS_F+T+_Y**

Section 4: 27 to 42, Identities: 5/16(31%), Positives: 8/16(50%), Gaps: 0/16(0%)

Mouse: **SPSVNGRSYDTYTPPH**
Fly: **SPTMEAVEASTASHPH**
Match: **SP++___T_+_PH**

4 Programming a Hidden Markov Model to Generate CG Rich Regions

A Hidden Markov Model (HMM) as shown in figure 1 can be used to generate DNA sequences. The sequence will consist of several regions, in this case two. The model works on the probabilistic nature of having a certain base within a specific region.

The probabilities found within the model are often selected after carefully examining specific DNA, and training the model off this data.

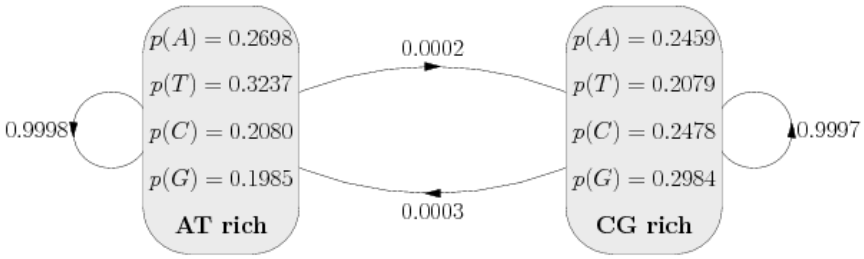


Figure 1: HMM to generate CG rich regions

There were three states the model could be in. The Start State(S), the CG Rich State(H) and the AT Rich State(L). The Start state had an equal 0.5 probability of leading to either of the H or L states, however could not be returned to.

Due to the probability of changing states being so low, around 5000 bases would have to be produced to see a change in state. A example CG Rich region produced by the HMM was:

HHHHHHHHHHHHHHHHHHHHHHHHHHHHH
CCGAGGTTGTTGAAACGGTCCCAACTTT

5 Writing a Viterbi Algorithm to Find the Most Likely CG Regions

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states, called the Viterbi path, that results in a sequence of observed events. The Viterbi algorithm works by producing the maximum likelihood estimates of the successive states of a finite-state machine (FSM) from the sequence of its outputs.

Figure 2 shows the Viterbi algorithm identifying the most likely sequence for each state, the red representing CG rich, and the black representing AT rich. This sequence was a subsection of the sequence generated by the HMM in the previous section.

Figure 2: Most likely CG rich regions from HMM sequence

6 Using the Viterbi Algorithm on the Phase Lambda Genome

The method in the previous section was applied to the Phase Lambda Genome, and the most likely CG regions were found in a colour coded manner. The results can be seen in the Jupiter Notebooks attached.

7 Appendix

Below are the Jupiter Notebooks that contain the code and relevant results for proof of work done.

COMP3212 Tutorial 4: Weeks 10-12 2019/20

Rohan Bungre

29466423

1 Differential Equations Describing a System of Chemical Equations

Chemical systems, such as gene regulation networks can be modelled by chemical equations, shown by equation 1.

$$\begin{aligned} \Theta &\xrightarrow{1} X \\ X &\xrightarrow{2} Y \\ 2X + Y &\xrightarrow{0.02} 3X \\ X &\xrightarrow{0.04} \Theta \end{aligned} \tag{1}$$

These chemical equations can be modelled by differential equations, shown by equation 2. These explain the change in the concentration of X and Y over the time t.

$$\begin{aligned} \frac{\mathrm{d}[X]}{\mathrm{d}t} &= 1 - 2.04[X] + 0.04[X]^2[Y] \\ \frac{\mathrm{d}[Y]}{\mathrm{d}t} &= 2[X] + 0.02[X]^2[Y] \end{aligned} \tag{2}$$

There are many methods of solving differential equations, from integrating both sides and rearranging or by using numerical methods.

2 Ordinary Differential Equation Approach

Using Scipy's odeint function, the differential equations were able to be solved in a deterministic manner. A deterministic model regards the time evolution as a continuous, fully predictable process.

Figure 1 shows that as X is being created, it immediately turns into Y. However, once Y passes a threshold point it starts combining with X, due to the X^2 factor. X dramatically jumps up, rapidly converting all Y to X. Once Y runs out, X slowly begins to degrade to an equilibrium position.

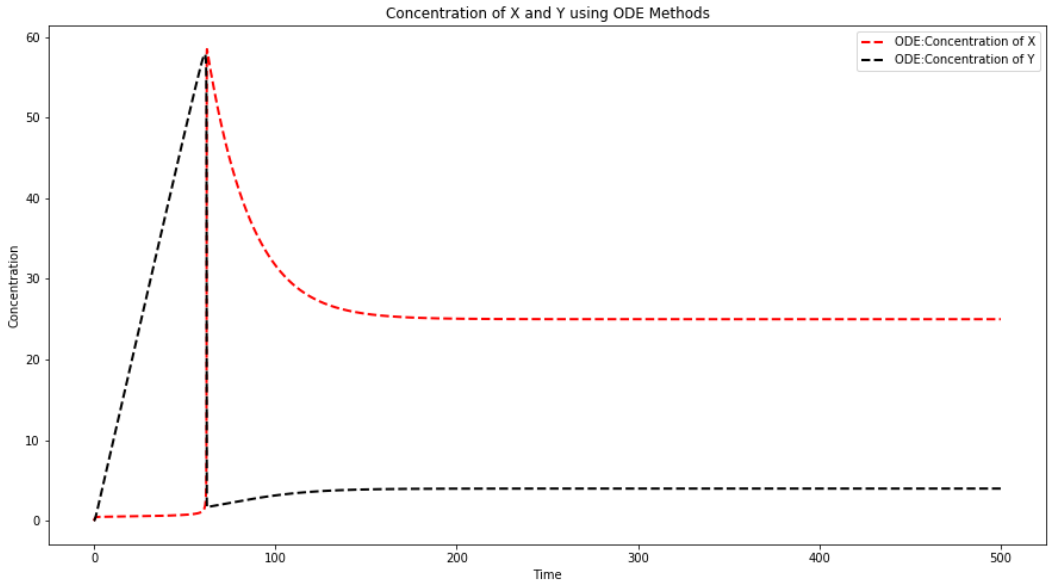


Figure 1: Concentration of X and Y using an ODE method

3 Gillespie Algorithm Approach

Another way to look at the chemical system, is to consider it being stochastic. The time evolution is regarded as a kind of random-walk process. A chemically reacting system is not a continuous process, because molecular population levels can only change by discrete integer amounts.

The Gillespie algorithm is one way to model this process. Figure 2 shows the results from the programmed Gillespie algorithm, that modelled the chemical system shown in equation 1.

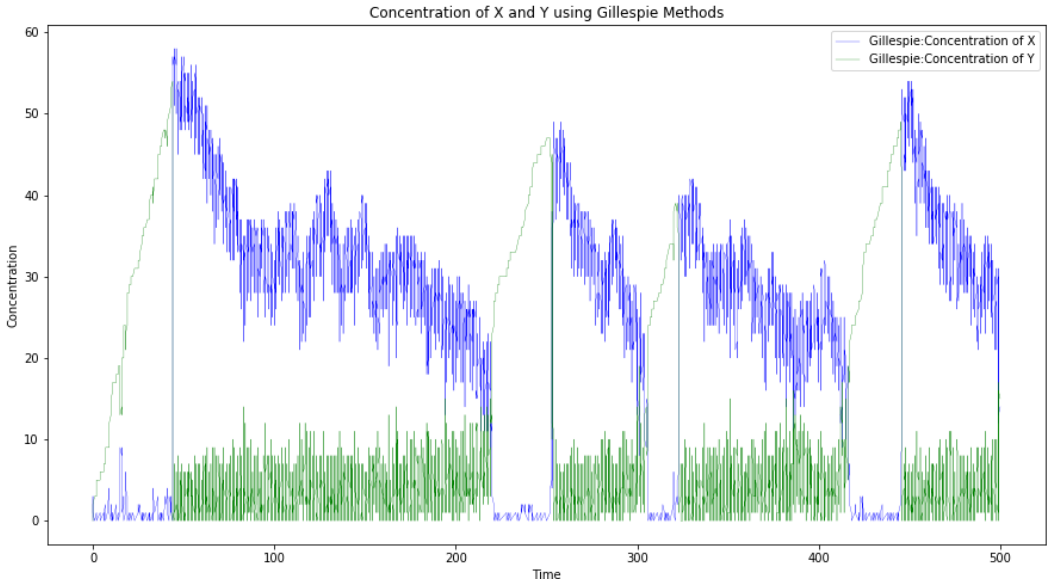


Figure 2: Concentration of X and Y using the Gillespie method

4 Comparing the Deterministic ODE and Stochastic Gillespie Approaches

Figure 3 shows both approaches used to derive the concentration of X and Y over 500 time-periods. They both assume initial conditions of $[X](0) = [Y](0) = 0$. Both graphs follow a similar pattern. There is a steep rise and cut off, in both the concentrations of X and Y. The ODE shows a smooth response and reaches equilibrium after 200 time-periods, however the Gillespie continues to rise and fall. This is due to the random processes that happen within the algorithm.

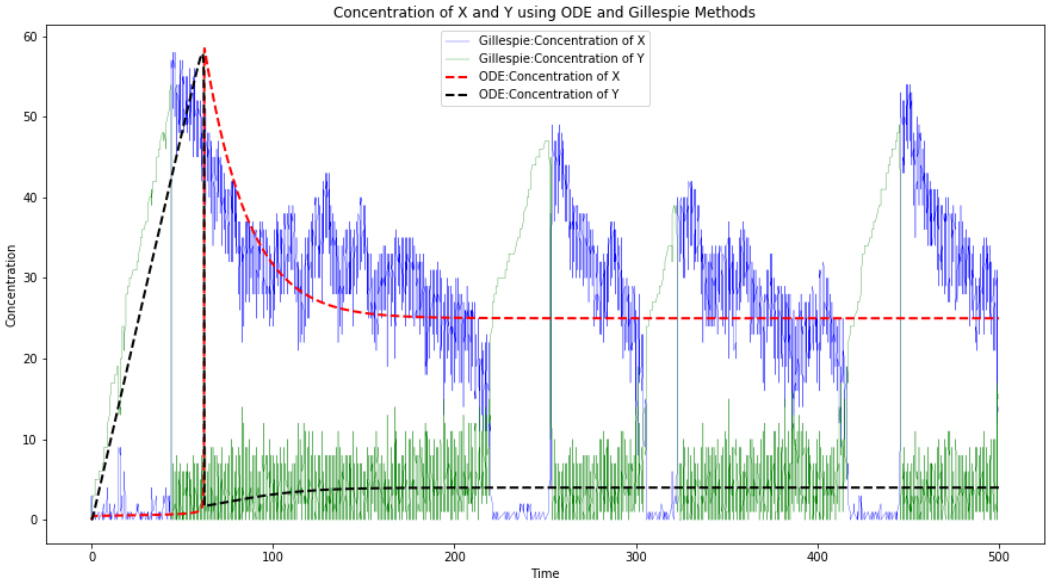


Figure 3: Concentration of X and Y using ODE and Gillespie methods

5 Appendix

Below are the Jupiter Notebooks that contain the code and relevant results for proof of work done.

COMP3223: Coursework 2019

Rohan Bungre

rsb1g17@soton.ac.uk

29466423

4. Linear Regression with non-linear functions

To fit polynomial functions to y(x) = sin(4πx) + ϵ where ϵ is a noise term. Must generate a number N of points chosen randomly from 0 ⩽ x ⩽ 1. Training and test sets will be taken from these (xn, yn) pairs, for n = 1, . . . , N.

4.1 Performing linear regression with polynomials and radial basis functions

To create the training set, the function f() *(Appendix A, lines 5-10)* was used. It would take in a random uniform vector x *(Appendix A, lines 58-59)* and return the function sin(4πx) + ϵ. The noise term ϵ was defined using a normal distribution with mean = 0 and variance = 0.01. This allowed for a noise term that oscillated above and below the true value, without distorting the sinusoidal shape. To create the test data set, a new random uniform vector x_new was created *(Appendix A, lines 71-72).*

4.1.a Linear regression using a polynomial basis function

The polynomial basis function was defined as: $\phi j(x_n) = x^j n$, where x_n was an input value. This can be explained by *Equation 1* and *Equation 2*:

$$y_N = \omega_0 + \omega_1 x_N + \omega_2 x_N^2 + \cdots \omega_N x_N^j \quad \textbf{[1]}$$

In matrix form:

$$Y = \begin{bmatrix} \omega_0 & \omega_1 & \cdots & \omega_N \end{bmatrix} \blacksquare \begin{bmatrix} 1 & 1 & 1 \\ x_1 & \cdots & x_N \\ \vdots & \ddots & \vdots \\ x_1^j & \cdots & x_N^j \end{bmatrix} \quad \textbf{[2]}$$

Equation 3 shows the design matrix X, which is the transpose of the input data matrix. Using the function create_poly(), (Appendix A, lines 18-25) the polynomial design matrix was created. The extra column of ones were added on using the function make_design(), (Appendix A , lines 36-42). The polynomial order was kept at 10 after experimentation to allow for a good model.

$$A = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^j \\ 1 & x_2 & x_2^2 & \cdots & x_2^j \\ 1 & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^j \end{bmatrix} \quad \textbf{[3]}$$

To train the model, the weight vector needed to be calculated. *Equation 4* shows the process that was followed to calculate the weights.

$$\boldsymbol{w} = (\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A} + \boldsymbol{\lambda}\boldsymbol{I}_{\boldsymbol{P+1}})^{-\boldsymbol{1}}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{y} \quad \textbf{[4]}$$

The function calc_weights(), (Appendix A, lines 44-52) shows the methodology behind this algorithm. Using the features of numpy, AtA was calculated alongside the identity matrix. The value of lamda was kept at 0 to not affect the polynomial linear regression features. These were summed together, then inversed using the function np.linalg.inv(), providing a suedo inverse. AtY was calculated and product was found, leaving the model's weight vector.

Using the principles from *Equation 2*, the test data was input to the model. This output the Y predictions. Using matplotlib, the original model, training data and test data was plotted on graphs for analysis. The number of test data points was always a quarter the number of training data set points.

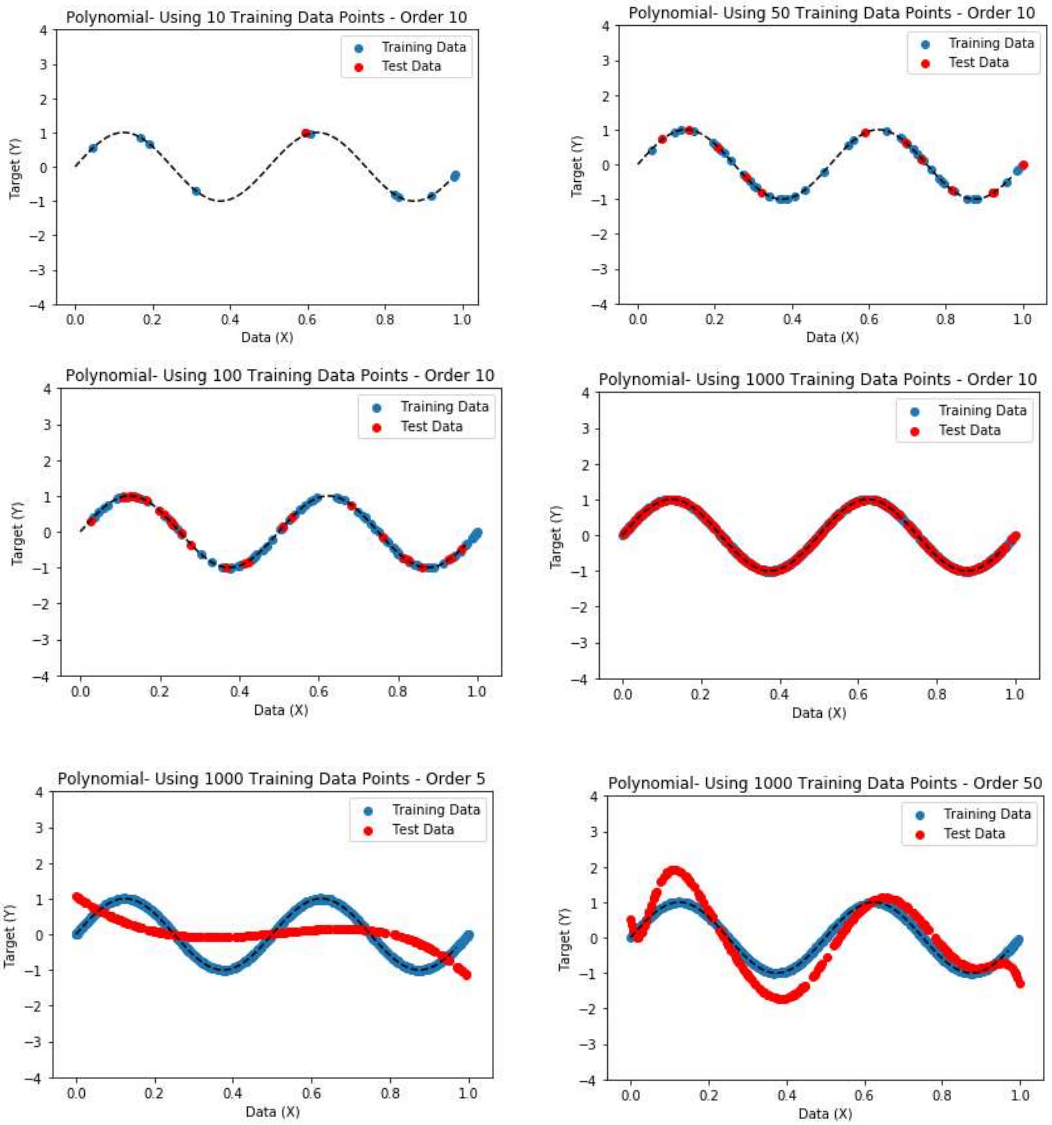


Figure 1 - Polynomial Fitting Using 10,50,100,500,1000 Training Data Sets

From *Figure 1,* the polynomial fitting works well when an order of 10 is used. It works well with even a small amount of training data. When the order is less than 10 the model will underfit the data. When above the order is above 10 the model will overfit.

4.1.b Linear regression using a radial basis function

The gaussian RBF was defined as $\Phi(\mathrm{x};\ \mathrm{x}_n) = \exp\left(-\frac{(\mathrm{x}-\mathrm{x}_n)}{2S^2}\right)$ where x_n was an input value. This can be explained by *Equation 5* and *Equation 6*:

$$\boldsymbol{y_N = \omega_0 + \omega_1 exp\left(-\frac{(x_n - x_1)^2}{2S^2}\right) + \omega_2 exp\left(-\frac{(x_n - x_2)^2}{2S^2}\right) + \cdots \omega_N exp\left(-\frac{(x_n - x_n)^2}{2S^2}\right)} \quad \textbf{[5]}$$

In the matrix form:

$$\boldsymbol{Y = [\omega_0 \quad \omega_1 \quad \cdots \quad \omega_N] \blacksquare \begin{bmatrix} 1 & 1 & 1 \\ exp\left(-\frac{(x_1 - x_1)^2}{2S^2}\right) & \cdots & exp\left(-\frac{(x_n - x_1)^2}{2S^2}\right) \\ \vdots & \ddots & \vdots \\ exp\left(-\frac{(x_1 - x_n)^2}{2S^2}\right) & \cdots & exp\left(-\frac{(x_n - x_n)^2}{2S^2}\right) \end{bmatrix}} \quad \textbf{[6]}$$

Equation 7 shows the design matrix when the transpose of the input training data. Using the function create_gauss(), (Appendix A lines 27-34), the gauss design matrix was created. The function itself concatenates the extra column of ones.

$$\boldsymbol{A = \begin{bmatrix} 1 & exp\left(-\frac{(x_1 - x_1)^2}{2S^2}\right) & \cdots & exp\left(-\frac{(x_1 - x_n)^2}{2S^2}\right) \\ 1 & exp\left(-\frac{(x_2 - x_1)^2}{2S^2}\right) & \cdots & exp\left(-\frac{(x_2 - x_n)^2}{2S^2}\right) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & exp\left(-\frac{(x_n - x_n)^2}{2S^2}\right) & \cdots & exp\left(-\frac{(x_n - x_n)^2}{2S^2}\right) \end{bmatrix}} \quad \textbf{[7]}$$

To train the model, the weight vector needed to be calculated. *Equation 8* shows the process that was followed to calculate the weights.

$$\boldsymbol{w = (A^T A + \lambda I_{P+1})^{-1} A^T y} \quad \textbf{[8]}$$

The function calc_weights(), (Appendix A, lines 44-52) shows the methodology behind this algorithm. Using the features of numpy, AtA was calculated alongside the identity matrix. The value of lamda was kept at 0 to not affect the gaussian linear regression features. These were summed together, then inversed using the function np.linalg.inv(), providing a suedo inverse. AtY was calculated and product was found, leaving the model's weight vector.

Using the principles from *Equation 6*, the test data was input to the model. This output the Y predictions. Using matplotlib, the original model, training data and test data was plotted on graphs for analysis.

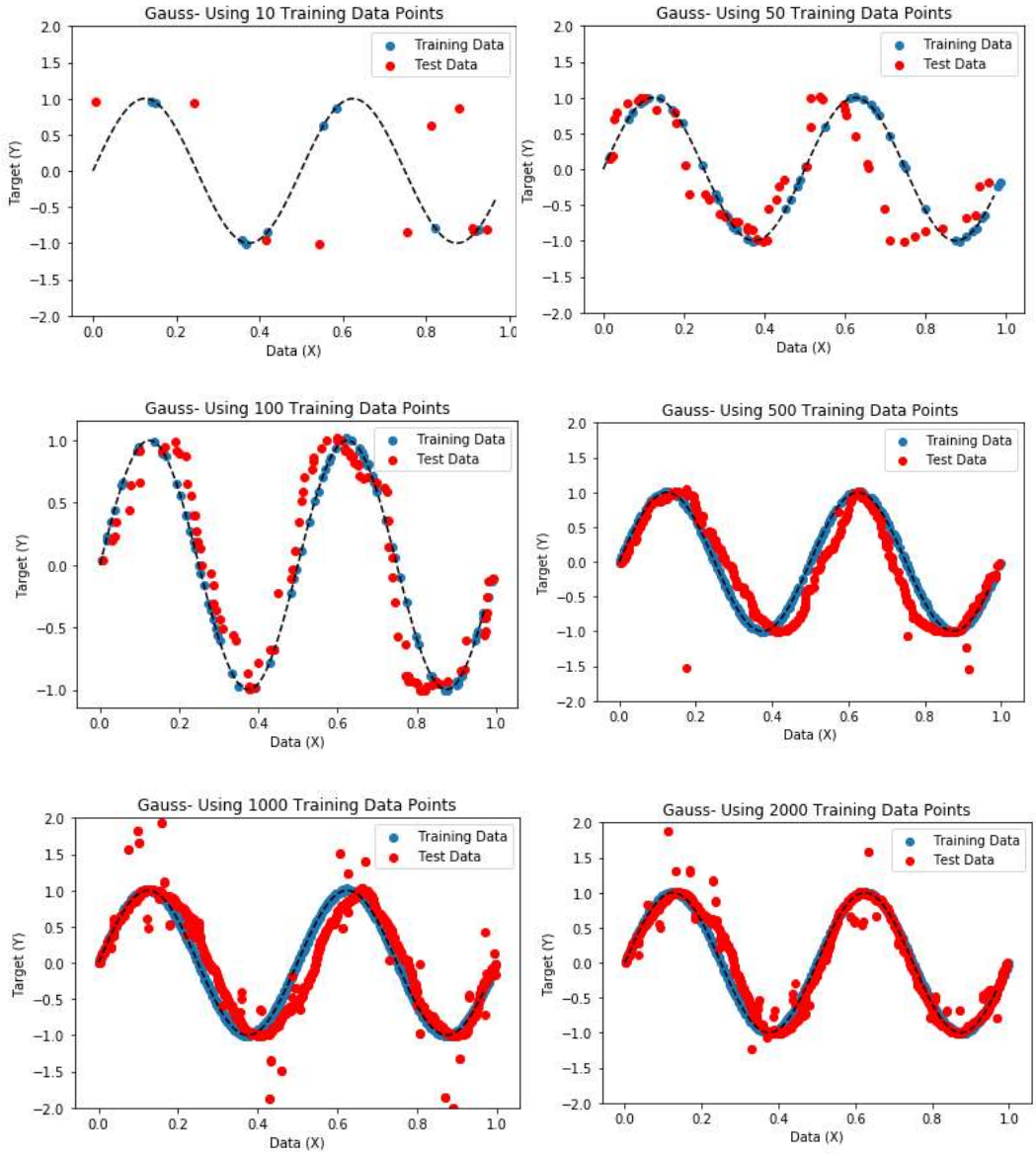


Figure 2- Gaussian Fitting Using 10,50,100,500,1000,2000 Training Data Sets

From *Figure 2*, the Gaussian RBF fitting works very well for large training data sets. From training data sets more than 50, the model can map to the original sinusoidal function. With 10 data points, the model didn't have enough data to produce accurate weights. When a very large training set is used, there appears to be more outlier predictions in the test data. This is because the data set input number is related to the design matrix size. More inputs, therefore more Φ values allow for a more accurate model.

4.2 How does regression generalise? Computing bias and variance

When there is not enough test data to verify the performance of the model, the input data is split into training and test data at a ratio commonly of 3:1. The weights are calculated using the training data and the model is evaluated using the sudo test data. Using scikit learn's train_test_split() function (Appendix B line 67), the input data was split up accordingly.

4.2.a Effect of the number of basis components on the model accuracy

The function poly_basis_dep (Appendix B lines 175-201) was used to observe the dependency between the number of basis components in the polynomial model and the accuracy of said model. The accuracy was determined by calculating the mean squared error (Appendix B line 189).

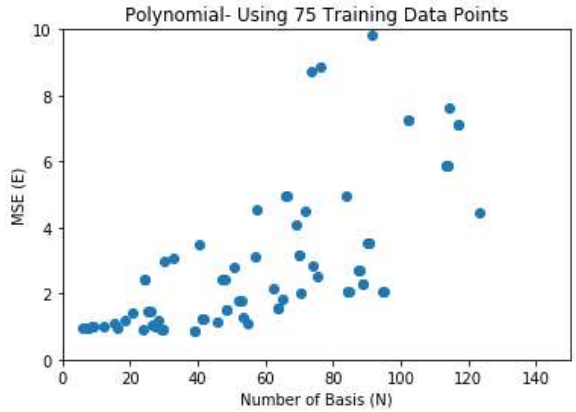


Figure 3 – MSE based on the number of polynomial basis components

Figure 3 shows the dependency when 75 training points are used. The results show that there is a general trend where the more basis components used, the larger the MSE. With a higher polynomial model, there seems to be a larger amount of overfitting causing a larger distance from the target value increasing the MSE. This alludes to a higher variance. At a lower polynomial order there is a large amount of underfitting due to the high bias so the mean squared error will be small. There is a trade-off between the two, with the best experimental value being a polynomial order of 10.

The function gauss_basis_dep (Appendix B lines 201-219) was used to observe the dependency between the number of basis components in the gaussian model and the accuracy of said model. Again, the MSE was used to evaluate the accuracy.

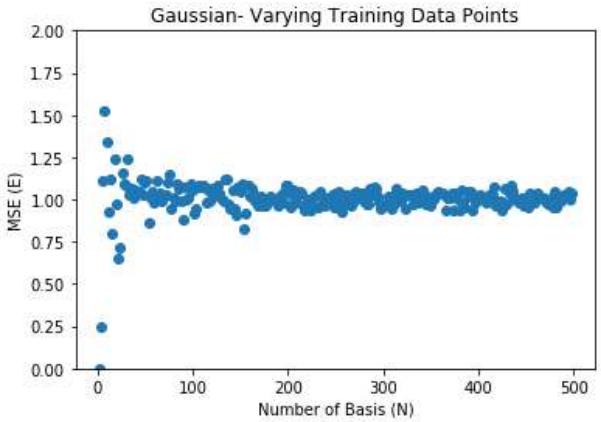


Figure 4 - MSE based on the number of gaussian basis components

Figure 4 shows the dependency when the gaussian model is used. The results show that at a low number of basis components the MSE is larger than when the number of components is increased. The model outputs a relatively fixed MSE value as the number of components is increased. This shows that the variance of the models is low, and for each new test the MSE remains the same. This appears to be a much better model than the polynomial one as the MSE remains constant.

4.2.b Effect of the regularisation coefficient on the model accuracy

The functions poly_lamda_dep() and gauss_lamds_dep() (Appendix B lines 123-172) was used to observe the dependency between the strength of the regularisation coefficient and the accuracy of said models. The accuracy was determined using the mean squared error.

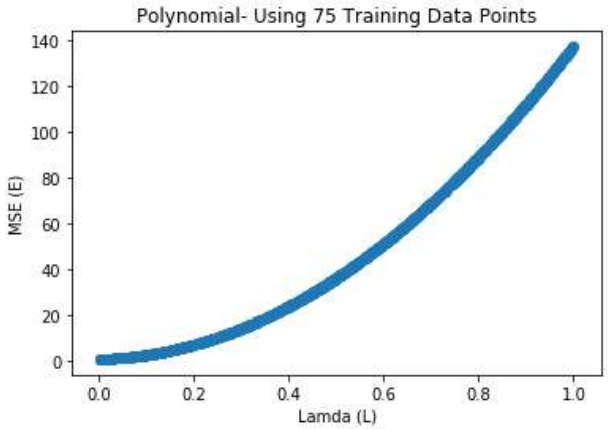


Figure 5 - MSE vs Regularisation Coefficient for Polynomial

At a fixed polynomial order of 10, *figure 5* shows the results of varying the regularisation coefficient. When increasing the regularisation coefficient, the MSE value increased at a quadratic rate. This is because increasing the coefficient increases the bias, generalising the model more. This would increase the MSE by a squared amount, due the quadratic relationship between bias and MSE.

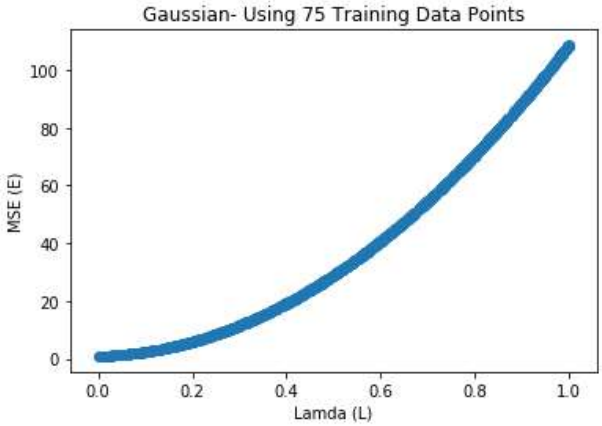


Figure 6 - MSE vs Regularisation Coefficient for Gaussian

At a fixed gaussian that used 75 training points, *figure 6* shows the results of varying the regularisation coefficient. Much like the polynomial, the MSE increased with an quadratic rate as the coefficient was increased. This follows the same logic that increasing the coefficient increases the bias, generalising the model.

These results show that the bias in both models didn't need to be increased to improve the accuracy of the models. Setting lambda as 0 was ideal to obtain the most accurate model.

4.2.c Observing the bias and variance of the models

By splitting the training data into smaller cardinal training sets the bias and variance could be evaluated by observing and comparing the MSE values for each smaller set.

The functions poly_train_split() and gauss_train_split() (Appendix B, lines 244-329) were created to split training data set into its smaller cardinal ones. It was able to create weightings for each data set and model the sinusoidal functions. 5 smaller training sets were created for both models and the MSE was calculated for each.

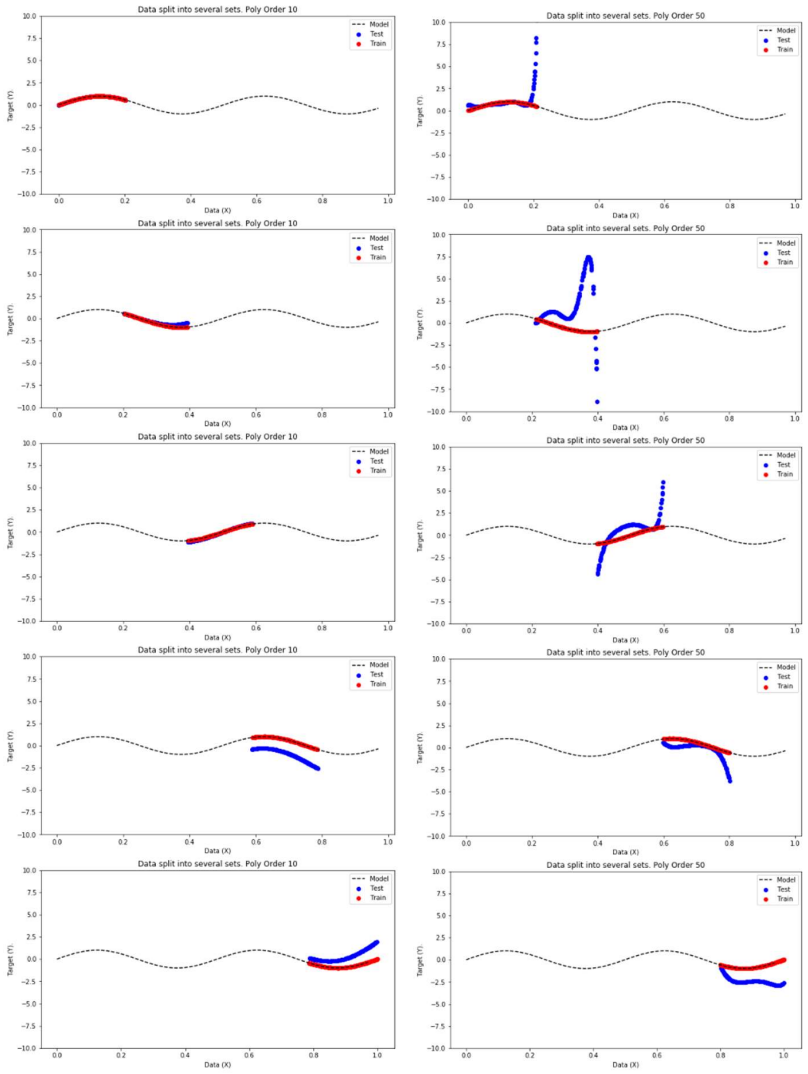


Figure 7 - Polynomial Model trained from 5 smaller sets using order 10 and 50

Figure 7 shows the polynomial model having been trained in sections using 5 smaller training sets. Two tests were done, one with a polynomial order of 10 and another with an order of 50. From the results a higher order polynomial produces a much worse fit with a small bias and large variation. This causes the overfitting as shown. The polynomial of order 10 produces a good model with the best trade-off between the bias and variance.

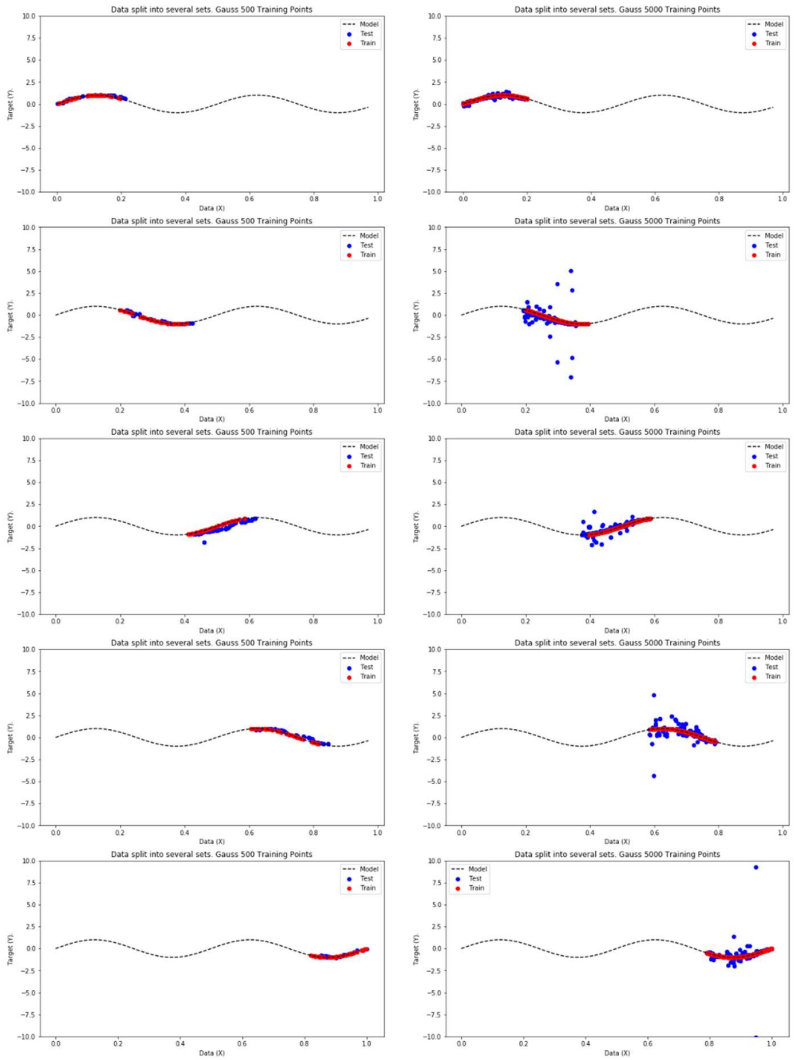


Figure 8 - Gaussian Model trained from 5 smaller sets

Figure 8 shows the gaussian model having been trained from the 5 smaller training sets. Two tests were done, varying the number of basis components. From the results when using more basis components, the variance increases as the MSE doesn't remain constant. However, the bias does increase allowing for the model to become more generalised. The lower order model does produce a more accurate model for the data provided so has the best bias – variance trade-off.

Data Set	Order 10 MSE	Order 50 MSE
1	0.1601	1.654
2	0.419	10.399
3	0.8331	2.98
4	3.1373	1.473
5	1.577	2.9915

Figure 9 - MSE for polynomial model

Data Set	Small Basis MSE	Large Basis MSE
1	0.1445	0.167
2	0.493	4.789
3	0.9163	0.8123
4	0.8079	0.626
5	0.1832	0.5618

Figure 10 - MSE for gaussian model

$$MSE = Bias^2 + Variance \quad \textbf{[9]}$$

Equation 9 show the Bias-Variance decomposition. For the smallest MSE the bias and variance must be small. This means that the model is very accurate for all possible test data sets provided. *Figure 9* shows the MSE of each data set for the polynomial model. For a 10^{th} order polynomial each data set produced a varied MSE value with a range from 0.160-3.137. I believe that the bias was small enough not to cause error, however the model did cause a fair amount of variance. For the 50^{th} order polynomial the range of MSE error value was 1.65-10.399. This shows an extremely large variance, caused by overfitting due to a model with small bias that was too precise.

Figure 10 shows the MSE of each data set for the gaussian model. The small basis model provided the smallest variance in data with a range of 0.1445-0.9163. The MSE values were consistently low meaning that that model had the optimal trade-off between the bias and variance. With the large basis model, the range of MSE values as 0.167-4.789, which means it had a larger variance than the small basis model. The model appears to be too general when looking at *figure 8*, therefore will have an increased bias value. The trade-off was not as optimal as the small basis model using the given data set.

Overall the small basis gaussian model was the best model in terms of the bias-variance trade off, giving the least MSE values. In a practical sense it might be too specific for a real-life scenario, so the 10^{th} order polynomial or the large basis gaussian may be better.

5. Classification

To explore methods of data transformations and projections. To use Fishers LDA to classify two datasets.

5.1 Data 1: separate 2 Gaussians

Two multivariate gaussian distributions were set up, where the differences of the means were of the order of magnitude of the standard deviations (Appendix C, lines 5-19). *Figure 11* shows this data, where Class A and Class B are clearly distinct, however are close enough to show some overlap.

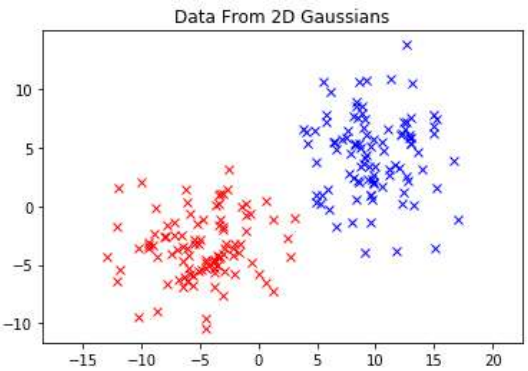


Figure 11 - Data with Mean A = [10,5] Mean B = [-5,-5]

To visualise the aspect of dimensional reduction, random direction vectors were chosen to display the class separation. *Figure 12* shows the histograms after the data had been projected onto each random vector. This shows that there will be an optimal vector that allows for maximum class separation.

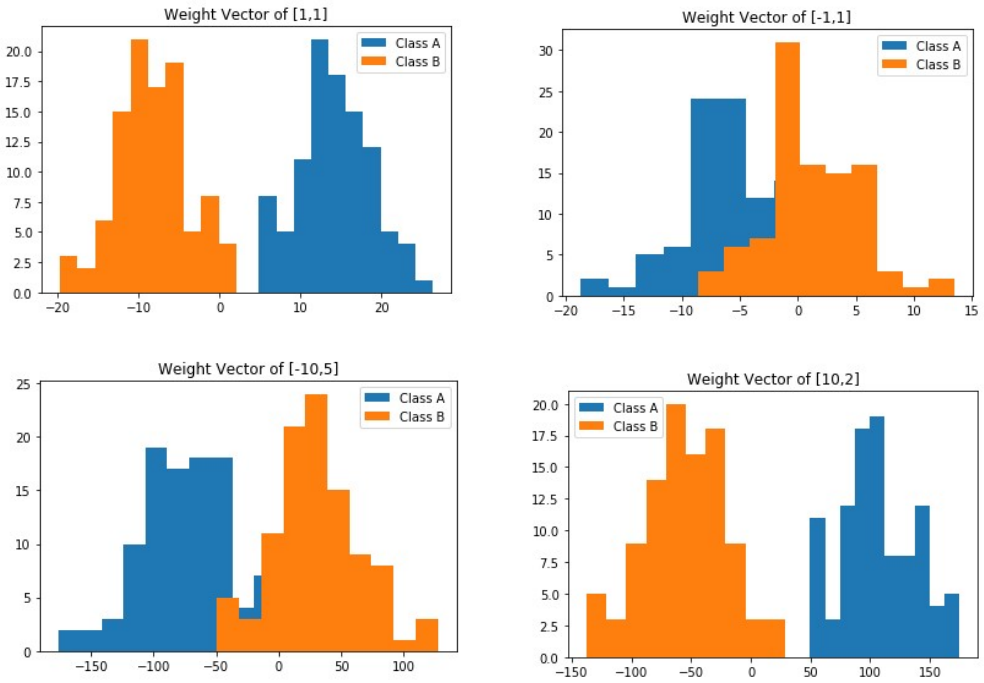


Figure 12 - Data projection for random direction vectors

To find the optimal value of this weight vector, a random starting vector was rotated through a full rotation, with the separation being measured using the Fishers Ratio (Appendix C, lines 37-58). The rotation was done using a rotation matrix that used a value of theta. The relationship of theta and the Fisher ratio was taken, then using the np.argmax() function (Appendix C, lines 64-66), the maximum Fisher ratio and its corresponding theta value was found. This would mean the optimal weight vector was given by the random starting vector dot product with the rotation matrix using the optimal theta value.

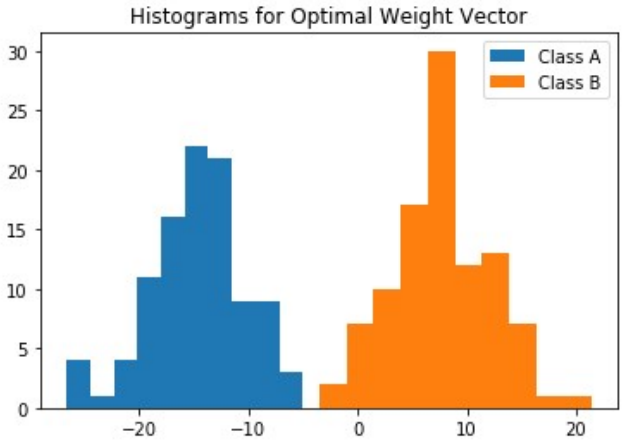


Figure 13 - Histogram using optimal weight vector

Figure 13 shows the data and class separation after projection onto the optimal weight vector. There is no overlap between classes in this data, and each class itself is more compact meaning input data is more likely to fall within its correct class.

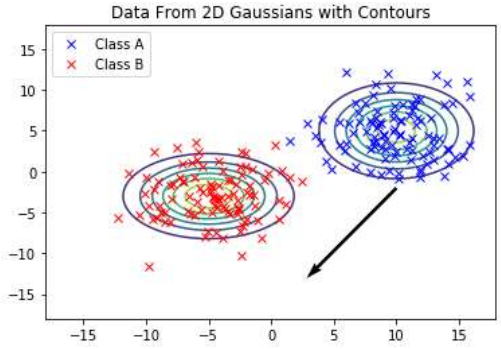


Figure 14 - Contour Plot and Direction Unequal Covariance

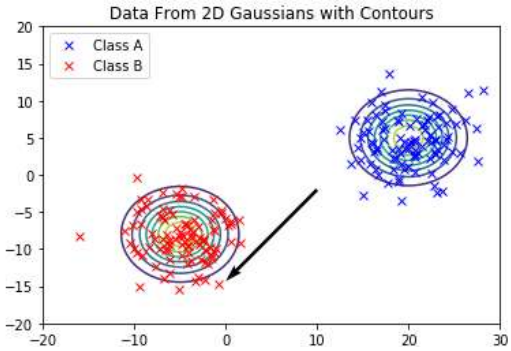


Figure 15 - Contour Plot and Direction Equal Covariance

Figure 14 and *Figure 15* show data points with the equi-probable contour lines for each class and the direction of the optimal choice vector w. *Figure 14* shows the original data plot when the covariance is unequal. This shows an ellipse shape for the contours as the variances of each axis are not the same. *Figure 15* shows new data points where the covariance is equal, so the contours form a circular shape. The optimal vector w shows the direction which will be orthogonal to the decision boundary, so it appears to be in the correct direction.

To plot the contours Scipy was used to create a multivariate gaussian (Appendix C lines 80-92). A mesh grid of input points could be created, then the output would show the probability contours. To plot the direction arrow the plt.quiver function was used.

$$\log\frac{P(c=a|x^n)}{P(c=b|x^n)} = -\frac{1}{2}(X^T(\varepsilon_a^{-1}-\varepsilon_b^{-1})X) - (\mu_a^T\varepsilon_a^{-1}-\mu_b^T\varepsilon_b^{-1})X + const \qquad \textbf{[10]}$$

To find the decision boundary the log odds ratio shown in *equation 10* needs to be found. Values of X that set the log-odds ratio to zero, will be points that lie along the decision boundary. To do this, the python library Sympy was used to solve this equation for X (Appendix C, lines 127-142). This would return a line equation that would represent the decision boundary.

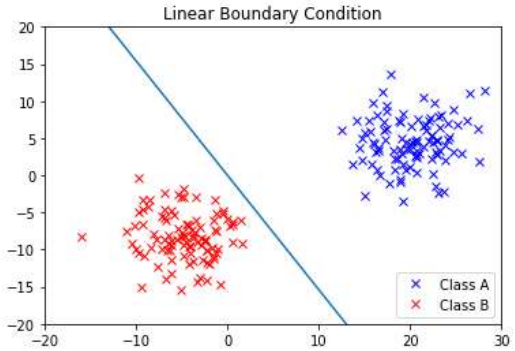


Figure 16 – Linear decision boundary for equal covariance

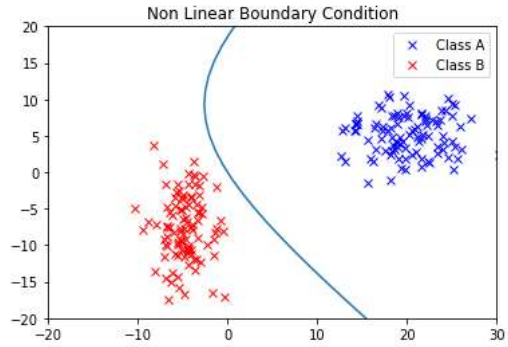


Figure 17 – Quadratic decision boundary for unequal covariance

Figure 16 shows the data points for each class with an equal covariance. When this happens the quadratic term in *equation 10* becomes zero, so the remaining term is linear in X. This means that the decision boundary is a linear function of the class features in X. *Figure 17* shows the data points when there is an unequal covariance. The quadratic term remains in *equation 10,* so the decision boundary is quadratic function of the class features in X.

5.2 Data 2: Iris data

The Iris dataset contains 3 different flower classes of 50 data points, with each data point having 4 description features.

$$S_W W = \lambda S_B W \quad \textbf{[11]}$$

To maximise the Fisher Ratio the optimal w direction vector must satisfy *equation 11*. This is known as a generalised eigenvalue problem and can be solved by calculating the S_W (Within-Class Scatter matrix) and the S_B (Between-Class Scatter Matrix). After loading the Iris dataset into python, the first step was to calculate the 4-dimentional feature mean vector for each of the 3 flower classes (Appendix D, lines 16-32) as it would be needed in subsequent calculations.

$$S_W = \sum_{i=1}^{c} \sum_{x \in D_i}^{n} (X - M_i)(X - M_i)^T \quad \textbf{[12]}$$

$$S_{\boldsymbol{B}} = \sum_{i=1}^{c} (M_i - \bar{\boldsymbol{M}})(M_i - \bar{\boldsymbol{M}})^T \quad \textbf{[13]}$$

Equation 12 shows the method for calculating the within-class scatter matrix. This was implemented in code (Appendix D, lines 34-57) by subtracting the mean vector from each data point in one class and dot product with the same but transposed. The all the class scatter matrices were summed together. This gives the effect of squaring the data points when working with matrices. When considered as the square of each data point from the mean, it can be thought of as the covariance of each class. Therefore S_W is equivalent to the sum of the class covariance matrices, without a factor of $\frac{1}{N-1}$.

Equation 13 shows the method for calculating the between-class scatter matrix. This was implemented in code (Appendix D, lines 59-67) by creating an overall mean vector, and subtracting it from each individual class mean vector. This was then dot product with the same but transposed and summed up for each different class.

$$S_W{}^{-1} S_B \quad \textbf{[14]}$$

To solve *equation 11*, the eigenvectors and eigenvalues of *equation 14* needs to be found. This can be done by using the np.linalg.eig() function (Appendix D, line 69). Although it was recommended to use the np.linalg.eigh() function, this caused the program to stop working so the former function was used.

$$Eigen\ 1: \begin{bmatrix} -0.2049 \\ -0.3871 \\ 0.5465 \\ 0.7138 \end{bmatrix} 32.3 \quad Eigen\ 2: \begin{bmatrix} -0.009 \\ -0.589 \\ 0.2543 \\ -0.767 \end{bmatrix} 0.278 \quad Eigen\ 3: \begin{bmatrix} 0.179 \\ -0.3178 \\ -0.3658 \\ 0.6011 \end{bmatrix} 0 \quad Eigen\ 4: \begin{bmatrix} 0.179 \\ -0.3178 \\ -0.3658 \\ 0.6011 \end{bmatrix} 0$$

Figure 18 – Eigen Vectors and Values from function

The eigen vectors and values from *figure 18* provide information about the linear transformation. The vectors show the direction of the new axis formed when reducing dimensions with the values being the scaling factors for the vector.

To check if the new vectors and values satisfy *equation 11*, the np.testing.assert_array_almost_equal() function can be used to test if *equation 15* is satisfied (Appendix D, lines 73-74).

$$S_W{}^{-1} S_B W = \lambda \boldsymbol{W} \quad \textbf{[15]}$$

Figure 18 shows that each eigen vector has a corresponding eigen value. The aim of LDA is to find the greatest class separation and to project data into a lower dimensional space. The optimal weight can be found using any of the eigen vectors, however, to reduce the dimensional space, several eigen vectors need to be ignored. The eigen value dictates how much information will be preserved at a lower dimension, so the larger eigen value the more useful the eigen vector is in a lower dimension. This means that the eigen values that are 0 can be ignored as the provide no information at a lower dimension.

It can also be shown that in LDA the number of non-zero eigen values will be at most c-1, where c is the number of class labels. This is because S_B is the sum of c matrices with a rank 1 or less, so the rank of S_B will be limited [2]. Using code found on the internet [1] (Appendix D, lines 76-80) the useful eigen vectors were combined into a 4x2 transformation matrix. The original data could then be projected into a lower dimensional space.

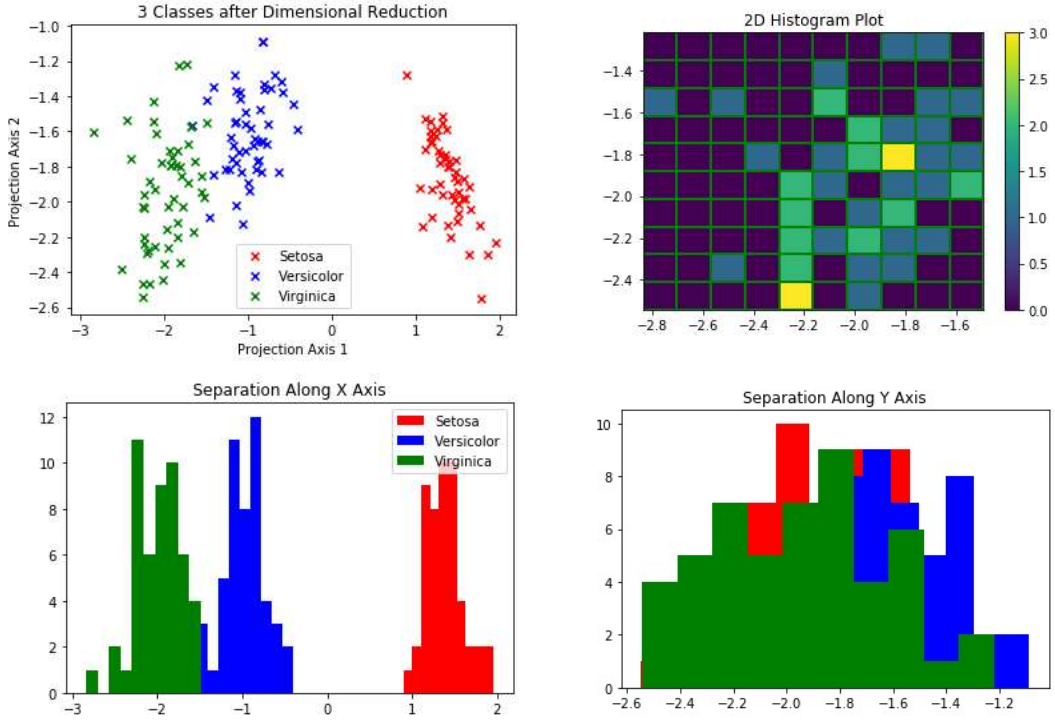


Figure 19 – Data projections and histograms

Figure 19 shows the data projection onto a lower dimension. Along axis 1 the Setosa class is well separated, however along axis 2 the data is not well separate at all. This can be explained by the two different eigen values. Axis 1 was the eigen vector with the eigen value of 32.3, that allows for the data to remain separated as the dimension is reduced. Axis 2 was the eigen vector with an eigen value of 0.278, so the class separation is lost.

In 5.1 the process of finding the optimal direction vector is in an iterative process, whereas in 5.2 a rigorous mathematical approach is taken. Both exercises require an optimal direction to be found so that the data can be projected into a lower dimension. In 5.1 this allowed for a good class separation and decision boundary to be found. In 5.2, only one axis of the lower dimension allowed for a good class separation. Having two classes with only 2 feature sets is much easier to project onto a 1D line when it comes from a gaussian data set.

LDA was not the best method to classify the Iris dataset as it doesn't reduce dimensions very well. A mixture of PCA and LDA could be beneficial to stop the issues of the information loss at lower dimensions.

References

[1]https://medium.com/journey-2-artificial-intelligence/lda-linear-discriminant-analysis-using-python-2155cf5b6398

[2] https://www.apsl.net/blog/2017/07/18/using-linear-discriminant-analysis-lda-data-explore-step-step/?fbclid=IwAR1c8xIGHgdph7aKXawrBX9OIXRSFuR22KP-xWb7IEeqUHAjSiJmPlSxuJ8

[3] https://www.youtube.com/watch?v=9K5NL4nfi0Y

[4] https://sebastianraschka.com/Articles/2014_python_lda.html

MATH3082 Optimisation Coursework 2019-20

Rohan Bungre

29466423

1 Integer Linear Programming

Linear programming is a method to maximise or minimise a mathematical model represented by a linear relationship. More formally, linear programming is a technique for the optimisation of a linear objective function, subject to linear equality and linear inequality constraints.

$$\begin{aligned} \max z &= 5x_1 + 6x_2 \\ x_1 + x_2 &\leq 5 \\ 4x_1 + 7x_2 &\leq 28 \\ x_1, x_2 &\geq 0 \end{aligned} \tag{1}$$

Equation 1 shows an example model, where the optimal values of x_1, x_2 can be found to maximise the objective function. Using a simplex method the solution to equation 1 can be found to be $x_1 = 7/3$ and $x_2 = 8/3$, giving an objective function maximum value of $z^* = 27.7$. Figure 1 shows the graphing method of solving the optimisation problem.

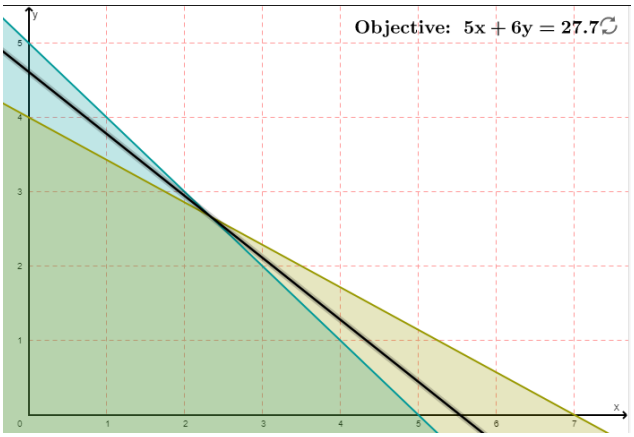


Figure 1: Graphing method of linear programming

In equation 1, the variables x_1, x_2 are free to be any number ≥ 0. However, in some cases the variables must be constrained to an integer value because they represent a discrete unit, such as the number of staff in a workplace. Solving the problem when

$x_1, x_2 \geq 0$ is simple using the simplex method, however, when x_1, x_2 are required to be integer values additional steps are required.

The trivial solution would be to try every integer pair within the feasible region but this is not scalable with a large number of variables. Another solution could be to solve the optimisation problem normally, obtain non integer results and then round them down, however this does not guarantee to produce the optimal solution. As figure 2 demonstrates, the rounded integer values of $x_1 = 2$, $x_2 = 2$ gives $z = 22$ however, the optimal integer values of $x_1 = 3$, $x_2 = 2$ gives $z^* = 27$.

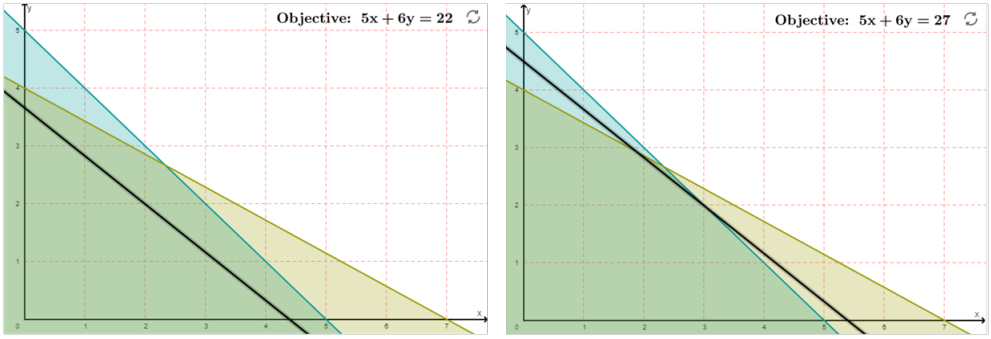


Figure 2: Rounding down vs optimal integer solution

$$\begin{aligned} \max z &= 5x_1 + 6x_2 \\ x_1 + x_2 &\leq 5 \\ 4x_1 + 7x_2 &\leq 28 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\text{ are integers} \end{aligned} \tag{2}$$

Equation 2 shows the updated formulation of the integer linear programming problem. To solve integer linear programming problems there are two methods available.

1. Branch and Bound Methods
2. Cutting Plane Methods

2 Branch and Bound Methods

The branch and bound method aims to split the feasible region into several sub-regions hoping to find an integer solution within one of them. Sometimes, the sub-regions must be split further if no integer solution is found.

To solve equation 2, the first step is to drop the requirement that all variables must be integers. This is known as the linear programming relaxation and forms the original equation 1. The linear programming relaxation will be used to estimate the optimal solution of the integer programming problem.

As shown in chapter 1, the solution to the linear relaxation is the solution to equation 1, which is $x_1 = 2.33$ and $x_2 = 2.67$, giving an objective function maximum value of $z^* = 27.7$. Choosing the variable to split the feasible region on is up for debate amongst academics. For this example x_2 is selected as the variable to split the feasible region on as is contains the larger decimal proportion.

As $x_2 = 2.67$, it can be bound by $x_2 \leq 3$ and $x_2 \geq 2$ for an integer solution, this causes two possible branches in the feasible region. Recalculating the linear relaxation problem using the simplex method with the extra bound constraints, the optimal solution can be calculated for each new sub region. Figure 3 shows this process using a graphical method.

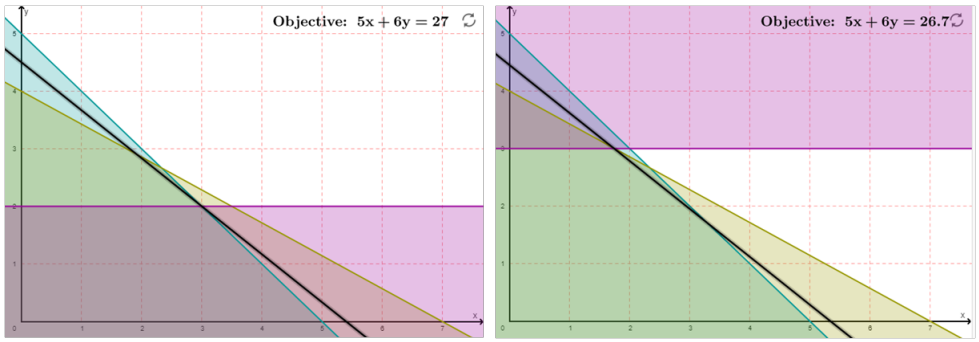


Figure 3: Comparing sub-region 2 with sub-region 3

Figure 4 shows the branching and bounding, stored in an enumeration tree. Region 2 has found an integer solution so it would not branch again. To ensure the optimal integer solution was found, Region 3 would branch and the process would iterate until no branching could happen, therefore all integer solutions had been found. Region 2 found the optimal integer programming solution of $x_1 = 3$, $x_2 = 2$ and $z^* = 27$.

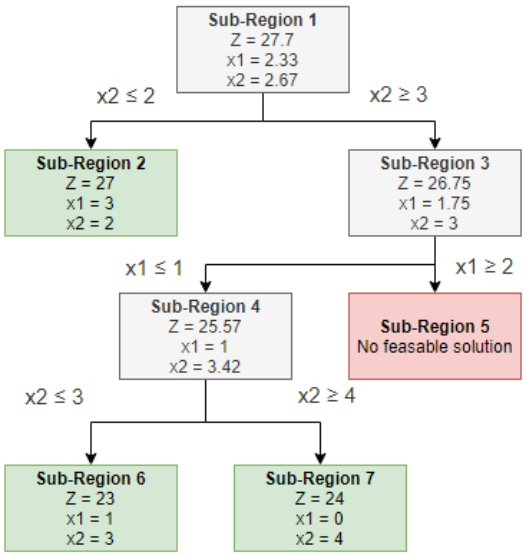


Figure 4: Enumeration tree for branch and bound methods

3 Cutting Plane Methods

The cutting plane method aims to solves integer linear programming problems by modifying the linear relaxation solutions until an integer solution is found. Unlike the branch and bound approach, it does not split the feasible region into sub-regions. It instead tries to refine a single linear problem, by adding new constraints.

These new constraints act as plane, cutting off parts of the feasible region that do not contain an integer solution. This successively reduces the feasible region until an integer optimal solution is found, demonstrated in figure 5. In practice, the branch-and-bound method always outperforms the cutting-plane algorithm.

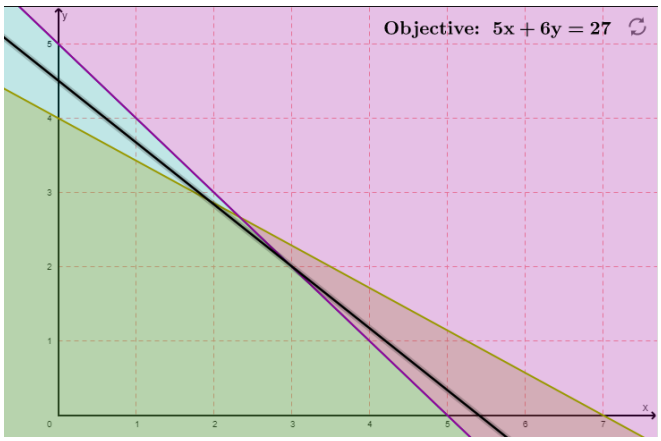


Figure 5: Cutting plane method shown graphically

An integer linear problem shown in equation 2 can be solved using the cutting plane method by first solving its linear relaxation formulation as shown in equation 1 using a simplex method. This produces a tabular shown in table 1.

BV	**x1**	**x2**	**s1**	**s2**	**RHS**
x1	1	0	7/3	-1/3	7/3
x2	0	1	-4/3	1/3	8/3
Z	0	0	11/3	1/3	83/3

Table 1: Original simplex tabular

The next step is to pick one of the basic variables that has a non-integer solution in the RHS. In the case of equation 3, the x_1 row was chosen. The equation from the row needs to be written out in a form so that the fractional values are written as the summation of an integer value and a fractional portion. This is demonstrated by equation 3.

$$\begin{aligned} x_1 + \frac{7}{3}s_1 - \frac{1}{3}s_2 &= \frac{7}{3} \\ x_1 + (2 + \frac{1}{3})s_1 + (-1 + \frac{2}{3})s_2 &= 2 + \frac{7}{3} \end{aligned} \tag{3}$$

To create the cutting plane equation, equation 3 needs to be rearranged so that anything with an integer coefficient is moved to the LHS and anything with fractional coefficients is moved to the RHS. This is demonstrated by equation 4.

$$x_1 + 2s_1 - s_2 - 2 = -\frac{1}{3}s_1 - \frac{2}{3}s_2 + \frac{1}{3} \tag{4}$$

As the slack variables s_1 and s_2 are always ≥ 0, this means that the RHS of equation 4 will always be $\leq \frac{1}{3}$. As the LHS variables will only take integer values the $\leq \frac{1}{3}$ will be rounded to its nearest integer value of ≤ 0. This leaves the cutting plane equation, shown in equation 5.

$$-\frac{1}{3}s_1 - \frac{2}{3}s_2 + \frac{1}{3} \leq 0 \tag{5}$$

This cutting plane equation can now become a new constraint for the optimal tabular in table 1. Placing the constraint in tabular produces table 2.

BV	**x1**	**x2**	**s1**	**s2**	**s3**	**RHS**
x1	1	0	7/3	-1/3	0	7/3
x2	0	1	-4/3	1/3	0	8/3
s3	0	0	-1/3	-2/3	1	-1/3
Z	0	0	11/3	1/3	0	83/3

Table 2: Simplex tabular with new cutting plane constraint

The dual simplex method can now solve table 2 and the previous steps can iterate until integer solutions are found. Table 3 shows the final tabular producing the same optimal integer solution as the graphical and branch and bound methods.

BV	**x1**	**x2**	**s1**	**s2**	**s3**	**s4**	**RHS**
x1	1	0	3	0	0	-1	3
x2	0	1	-2	0	0	1	2
s1	0	0	2	1	0	-3	2
s2	0	0	1	0	1	-2	1
Z	0	0	3	0	0	1	**27**

Table 3: Simplex tabular with optimal cutting plane constraints

The final tabular found the optimal integer programming solution of $x_1 = 3$, $x_2 = 2$ and $z^* = 27$. The cutting plane method can be used in conjunction with the branch and bound method, to help reduce the number of branches required. The cutting plane will remove non-integer solutions before branching and bounding needs to be performed, reducing the number of interactions required to find an optimal solution.

4 Staff Scheduling Task

The aim of this task is to minimise the labour cost of a cafeteria, whilst ensuring there are enough staff to maintain customer satisfaction.

The important information and assumptions made:

- The cafe is open from 9am-5pm which includes 8 hour long periods.
- A full time employee works for 7 hour long periods plus an additional hour long break.
- A part time employee works for 4 consecutive hour long periods with no breaks.
- A full time employee is paid £12 per hour and is paid for their break totaling £96 per day
- A part time employee is paid £7.5 per hour totaling £30 per day.
- There must be at least 4 full time employees hired per day.

To form a model that would produce the optimal results, the full time workers would have to be able take their break at any point during the day. The same applies for the 4 hour blocks that part time workers would have to work.

As there were 8 different periods that full time workers could have their break in, there would be 8 different types of full time workers $F_1 \rightarrow F_8$. As there were 5 different ways of working 4 consecutive hours, there were 5 different types of part time employees $P_1 \rightarrow P_5$.

This could be formulated as an integer linear programming problem with a model shown in equation 6, where A is the decision matrix; x is the vector of basic variables $F_1 \rightarrow F_8$ and $P_1 \rightarrow P_5$; and b is the vector of constraints.

$$\begin{aligned} \min Z = 96 \sum_{n=1}^{8} F_n + 30 \sum_{n=1}^{5} P_n \\ \text{subject to } Ax \leq b \\ x \geq 0 \text{ as integers} \\ \sum_{n=1}^{8} F_n \geq 4 \end{aligned} \tag{6}$$

Equation 6 is able to model the problem of minimising the labor cost per day, whilst having enough staff per hour period; having at least 4 full time staff, ensuring the optimal result.

Using Microsoft Excel and its linear programming solver, the optimal integer solution to equation 6 was found. Microsoft Excel was used because of the University closures and not having reliable access to the Xpress-IVE software.

Figure 6 shows the Excel model, which allows for a visualisation of the decision matrix A (shown in white), the constraints vector b (shown in yellow) and the basic variables vector x (shown in green).

Firstly, the solution found that there was no over-staffing issues with the current schedule. The number of staff required per period is equal to the staff hired per period, meaning customers will remain content.

Secondly, the solution found that four full time workers must be hired, with two taking their lunch break from 3pm-4pm and the other two taking their lunch break from 10am-11am. Although these are not ideal lunch hours, it does provide the optimal solution.

Finally, the solution found that 6 part time worker must be hired, with two working from 9am-1pm; one working from 10am-2pm; another one working from 12noon-4pm and the final two working from 1pm-5pm.

	Type of Staff	F1	F2	F3	F4	F5	F6	F7	F8	P1	P2	P3	P4	P5			
Period	# Per Type	0	2	0	0	0	0	2	0	2	1	0	1	2	# Per Period		# Required
9am-10am		1	1	1	1	1	1	1	0	1	0	0	0	0	6	>=	6
10am-11am		1	1	1	1	1	1	0	1	1	1	0	0	0	5	>=	5
11am-12noon		1	1	1	1	1	0	1	1	1	1	1	0	0	7	>=	7
12noon-1pm		1	1	1	1	0	1	1	1	1	1	1	1	0	8	>=	8
1pm-2pm		1	1	1	0	1	1	1	1	0	1	1	1	1	8	>=	8
2pm-3pm		1	1	0	1	1	1	1	1	0	0	1	1	1	7	>=	7
3pm-4pm		1	0	1	1	1	1	1	1	0	0	0	1	1	5	>=	5
4pm-5pm		0	1	1	1	1	1	1	1	0	0	0	0	1	6	>=	6
	Cost Per Day	96	96	96	96	96	96	96	96	30	30	30	30	30			

Figure 6: Staff scheduling model in Excel

Figure 7 shows the minimised labour cost to be £564 per day, when hiring the 10 staff in the proposed manner. Due to the flexibility of being able to decided when each employee could work and take breaks, it allowed for the most optimal integer solution given the provided constraints.

Type of Staff	# Hours	Salery/Hour
Full Time (F1-F8)	8	12
Part Time (P1-P5)	4	7.5

Labour Cost/Day	# Full Time	#Part Time
564	4	6

Figure 7: Minimum labour cost and number of staff hired per day

Most full time employees would probably prefer to have their lunch break during the hours of 12noon-1pm or 1pm-2pm. By constructing a new model in Excel, following the same constraints as equation 6, with an updated decision matrix A, and basic variable vector x, this can be investigated.

Figure 8 shows the updated Excel model, which has a much simpler formation due to the smaller number of basic variables. Now that the employees can only have a lunch break during the typical lunch hours of 12noon-1pm or 1pm-2pm, there is an over-staffing problem.

During those lunch break hour, more part time employees must be hired to maintain customer satisfaction. This has caused 3 of the hour periods, 10am-11am, 11am-12noon and 3pm-4pm to be overstaffed.

	Type of Staff	F1	F2	P1	P2	P3	P4	P5			
Period	# Per Type	2	2	2	3	1	0	2	# Per Period		# Required
9am-10am		1	1	1	0	0	0	0	6	>=	6
10am-11am		1	1	1	1	0	0	0	9	>=	5
11am-12noon		1	1	1	1	1	0	0	10	>=	7
12noon-1pm		0	1	1	1	1	1	0	8	>=	8
1pm-2pm		1	0	0	1	1	1	1	8	>=	8
2pm-3pm		1	1	0	0	1	1	1	7	>=	7
3pm-4pm		1	1	0	0	0	1	1	6	>=	5
4pm-5pm		1	1	0	0	0	0	1	6	>=	6
	Cost Per Day	96	96	30	30	30	30	30			

Figure 8: Updated staff scheduling model in Excel

Figure 9 shows that the updated minimum labour cost per day has increased to £624. This is caused by the additional 2 part time workers having to be hired to cover the full time staff whilst they were on a lunch break.

These results show the trade off that many businesses have to make, between minimising labour costs and maintaining employee satisfaction.

Type of Staff	# Hours	Salery/Hour
Full Time (F1-F2)	8	12
Part Time (P1-P5)	4	7.5

Labour Cost/Day	# Full Time	#Part Time
624	4	8

Figure 9: Updated minimum labour cost and number of staff hired per day

Further investigations could look into part time employees having flexible working hours with potentially some being shorter, as that would have helped reduce labour costs. Another investigation could look at reducing the number of full-time staff required as the part-time staff gain experience. This would save a lot more money for the cafe.

COMP3219 ETC PROPOSAL

2019

Rohan Bungre, rsbg17, 29466423
David Court, dwc1g17, 29195152
William Hazelden, wth1g17, 29270847
Thomas Mead, tcm1g17, 29460778
Etienne Peach, ep4g17, 29142947
Rahul Ponda, rnp1n17, 29605377

Word Count: 4994

Executive Summary

PER-FIT is a feasible product that will revolutionise the online fashion industry, improving on an outdated market. Using modern technology such as Augmented Reality and Machine Learning, a unique and novel experience can be had by the user. PER-FIT aims to fix the current issues experienced with the industry, redefining the standard of online fashion. This will allow PER-FIT to have a monopoly on the market and continue to grow and maintain profit.

The existing market Per-Fit aims to enter is vast and shows signs of steady growth all the way up to 2023 and beyond. The target market is expected to be predominantly within the ages 16 to 35 with an equal gender split, and the app must therefore cater to this demographic. While there are products attempting to solve the same sort of problem as Per-fit, which shall be close competition, there is nothing available as effective. Per-fit effectively houses a multitude of features all in one place which will give an edge over its closest competitors such as ASOS.

PER-FIT will operate a multi-sided business model which deals with fashion brands and consumers as an E-commerce company. A 'Freemium' business strategy will be adopted meaning that not all the features of PER-FIT are made available to non-premium customers. PER-FIT ensures an expected monthly revenue which is independent to the amount of clothes sold due to the subscriptions made by customers and fashion brands. Pricing tactics such as 'penetration pricing' will be used to encourage fashion brands to adopt PER-FIT's services. The target market is young adults, and the advertising is adapted for this market using Instagram and not using intrusive advertising techniques.

The proposed schedule of development for PER-FIT is 14 weeks for a specialist team of application developers to deploy. PER-FIT has a very promising financial forecast in correlation with the increasing number of monthly users. With an initial funding of £410,000, this company will become profitable after 1.2 years, with a payback period of 2.2 years and an internal rate of return at 30.58% after 3 years. A thorough risk assessment explains the potential risks, likelihood and impacts, with the most considerable risk being unable to gain support from clothing retailers.

Intellectual property (IP) is an intangible asset that is the result of creations of the mind. IP provides PER-FIT with ownership rights offering protection by giving PER-FIT the right to control the use of its work and use it to gain financial reward, preventing others from illegally using PER-FIT's designs. User data protection (GDPR) and intellectual property assurance are at the forefront of PER-FIT's core values. Maximising consumer satisfaction by promoting sustainable comfort with minimal environmental impact in efforts to better the lives of consumers by revolutionising the online fashion industry.

Contents

1. Product Description

What is PER-FIT?

AT PER-FIT the vision is simple: “To lead the next generation of online fashion.”

PER-FIT is a state-of-the-art service and application that will revolutionise the online fashion industry benefiting the younger generation who shop online. PER-FIT will be a 3rd party vendor of popular fashion brands that uses modern technology to help sell their inventory, **Figure 1.1**.

PER-FIT aims to find clothing items that fit the user perfectly, without having to physically try them on, avoiding the hassle of returning ill-fitting items. PER-FIT also aims to recommend the user new clothing items to encourage more sales.

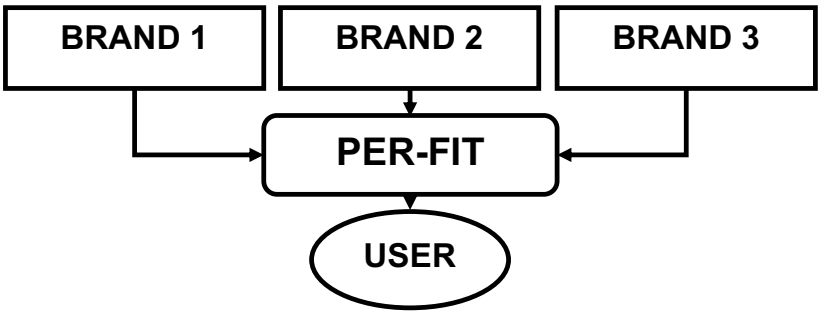


Figure 1.1: PER-FIT Service Model

PER-FIT has an application that can be installed onto the user’s phone. This application will make use of the cameras and sensors to scan the user’s body, **Figure 1.2**. Augmented reality (AR) will show how a clothing item would fit the user in real time. Any garment or whole outfits can also be shown on the user, to gauge a correct fit, **Figure 1.3**. This will replace the worry of ordering clothes online due to the confidence of the AR fit.

The AR will constantly be improving its ability due to machine learning techniques implemented by PER-FIT. Users can also enter their measurements into the application to aid the AR technology. A virtual avatar of the user’s body can be stored on the application, allowing the user ease of use on the go.

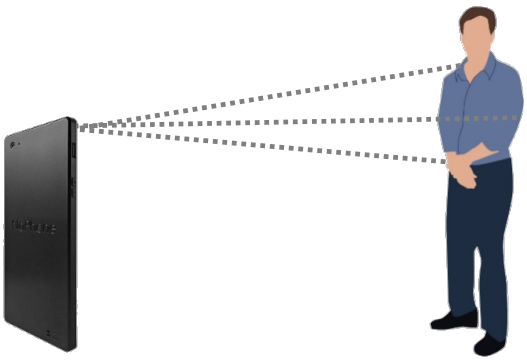


Figure 1.2: Augmented Reality Scanning. Created using: Phone Template. (2019), Man Standing. (2019).

Figure 1.3: Augmented Reality Application. Created using: Coloured Tees. (2019), Phone Template. (2019), Casual Man Standing. (2019).

PER-FIT includes an online shopping experience, accessed from the website or the mobile application. This service will have access to the entire database of brands and the items sold through PER-FIT. It will contain all variants of each item, and whether the item is in stock with the supplier, **Figure 1.4**.

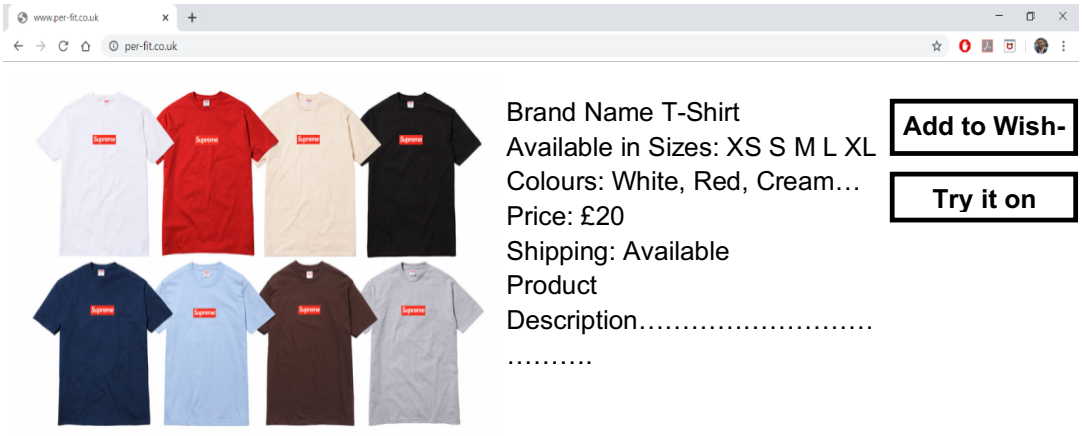


Figure 1.4: PER-FIT Website Design. Created using: Supreme Tee. (2019).

When the user likes an item of clothing, it can be added to a wish-list or can be virtually tried on using the AR mobile application, viewing how each item of clothing fits themselves or their virtual avatar. The size and colour can be varied to find the preferred fit and style. PER-FIT will recommend a size based on previous choices made by the user and other users.

PER-FIT will use Artificial Intelligence (AI) to learn the user's preferred fit will recommend similar items. The preferred colours and brands will also be recommended to encourage sales. Unique to PER-FIT are machine learning algorithms which find cheaper alternatives to an expensive item that share a similar fit and style. This will show the user that we care about them and their money. PER-FIT will recommend whole outfits, based on what items the user likes, **Figure 1.5**.

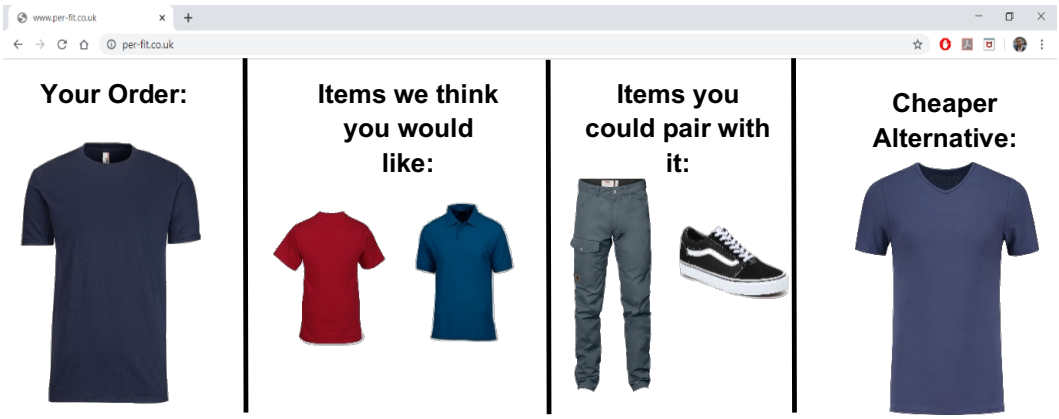


Figure 1.5: PER-FIT AI Recommendations. Created using: Black Shoes. (2019), Dark Blue Tee. (2019), Coloured Tees. (2019).

PER-FIT will also send out a personalised newsletter that will contain 3D models of the user's stored avatar wearing clothing items recommended by PER-FIT. This will further increase sales and take the trouble out of choosing what to wear. PER-FIT aims to make online fashion simple and easier to use, therefore having a monopoly on the online market. Shipping will be handled by the brand itself because as a start-up, PER-FIT does not have access to a large warehouse and contracts with large shipping companies.

Novelty

PER-FIT combines many existing technologies into one novel service to improve a large online market, **Figure 1.6**. Companies such ASOS sell branded clothes online, however the user must risk receiving ill-fitting clothing, thus go through a laborious returning process if so. PER-FIT solves this with its AR application.

Snapchat and Instagram have made large strides in AR filter technology yet have not used it to sell products unlike PER-FIT. PER-FIT's closest rivals are companies like M-Tailor who measure the users body using a phone camera and create custom clothes. This, however, offers no real time visualisation or avatar virtualisation.

PER-FIT offers a much cheaper and exciting solution to these tailors. PER-FIT is a unique company that will use a novel AR technique to sell more products than any other online clothing retailer.

Figure 1.6: PER-FIT's Unique and Novel Features

Feasibility

PER-FIT is a solution that hinges on already realised technologies. The main technological aspect of PER-FIT is the proposed AR technology. Snapchat, Instagram and Facebook have already shown that AR can be performed on the human body, using the front facing camera and facial pattern recognition techniques, **Figure 1.7**.

Using cameras and additional sensors if available, mapping clothing onto a human body will be possible. Companies like M-Taylor use simple trigonometry to calculate body measurements, so adding the AR element to this will be feasible.

Apple and many other phone manufacturers have implemented virtualised facial recognition, so storing a virtualised avatar is feasible. PER-FIT will collect measurements from items of clothing sold, allowing for accurate AR fits.

Creating a website is extremely feasible, with Wix and Square Space offering simple interfaces to build a reliable website that can integrate payment and back end services such as Amazon Web Services or Google Databases. When starting up, cloud services can host all PER-FIT's user data and stock data. This will reduce the complexity of needing to set up local servers.

Figure 1.7: Instagram, Snapchat and Facebook AR Mapping Technology

2. Market Analysis

Industry Overview

Since 2008 the number of internet users ordering goods online has increased steadily by ~2% per year, with no signs of slowing, **Figure 2.1**. Examining the online shopping market shows that out of the vast range of products available online, the largest field by far is Clothing and sports goods, with an average of 65% of online shoppers purchasing from this category (Ec.europa.eu, 2019) and it is predicted to increase by as much as 64.15% from total sales of $534 billion in 2018 to $873 billion in 2023, **Figure 2.2**.

The online fashion industry is clearly a massive market, but one of its biggest issues is the returns rate. From BBC news' investigation into returns, a major issue leading to returns is that clothes don't fit or look the same as on the website. One interviewee, standing at 5 foot, stated it is hard to find items to fit so often orders the same thing in three sizes, while another claimed clothes often look "fantastic" on the models but don't look the same on her (Hope, 2019). This suggests a clear need for Per-fit, allowing users to see clothes on themselves and find the correct size, removing any disappointment when the item arrives and reducing the need to return items.

This doesn't just benefit the customer however, but retailers as well. A Barclaycard survey found 57% of retailers said handling returns has a negative impact on the day-to-day running of their business (Charlton, 2019), whilst **Figure 2.3** shows how a bad returns experience can cause a business to lose customers. By using Per-fit shoppers will be less likely to need to return an item, reducing the disruption to businesses, whilst removing the risk of a negative impact on customers.

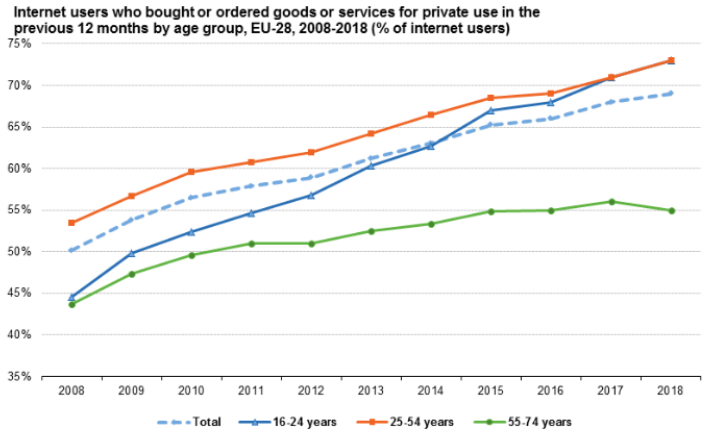


Figure 2.1: Percentage of internet users who ordered goods for private use, taken from (Ec.europa.eu, 2019)

Figure 2.2: Global online fashion sales (2018-2023) (Charlton, 2019)

Figure 2.3: Effect of negative returns experience taken from (Charlton, 2019)

Potential market

Looking at **Figure 2.1** once more, toward the start of the timeframe most orders came from people aged 25-54, however by 2018 this age group was matched by those aged 16-24 at approximately 73% each. This puts the target market predominantly in the enormous range of people aged 16-54 however a large proportion of people aged 55-74 may be expected. This split is best seen in **Figure 2.4**, showing those who purchased clothing or sports goods online in 2019. Again, this supports that the potential customer base is likely to fall predominantly within the age group of 16-34, whilst having a reasonable number from the older age groups, up to 65+. Looking at gender now, it is seen 56% of men shopped online compared to a slightly higher figure of 64% for women, however this is still a fairly even split and Per-fit should therefore cater equally to both genders.

Smartphone users in the UK, and thus those who may potentially download Per-fit, has seen rise since 2012, **Figure 2.5**. Here, as you might expect, those of younger age groups have always been high in percentage, initially being led by those aged 16-24 years with the group 25-34 years taking the lead in 2018 with a huge percentage of 98% (the reverse trend of **Figure 2.1**). What is surprising here is the 55-64 age group, initially being just 9% smartphone users in 2012 shows a huge rise up to 71% in 2018 (+10.3% each year).

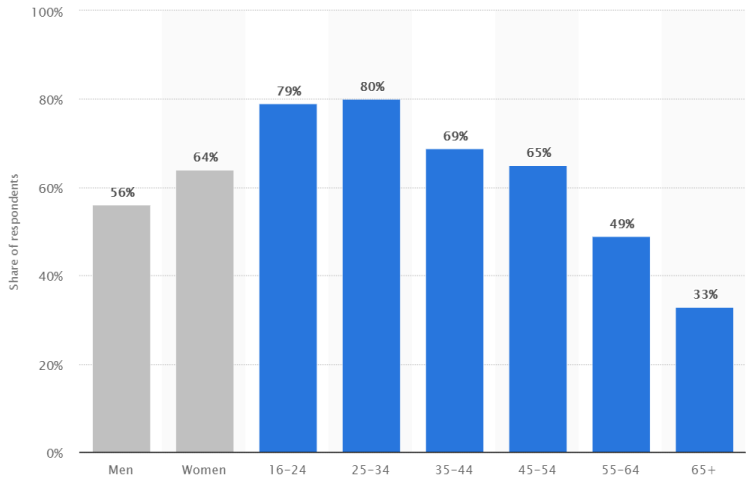


Figure 2.4: Individuals who purchased clothes or sports good online in 2019, taken from (Sabunoglu, 2019)

From the data discussed in this section one would expect to see an even split between genders with perhaps slightly more female users, and for users to be of a wide age range up to 65 years and above with the majority being 16-35 years old. It should be noted that this 16-35 age range where most customers are expected is within those groups most affected by negative returns experience in **Figure 2.3**, and therefore may benefit most from a service like Per-fit.

Competition

There are other products on the market trying to tackle the fitting problem each with a different take, the most sophisticated being 'Acustom Apparel' which offers clients the chance to have a full 3D body scan at their head office and custom make clothing from your measurements; essentially a high tech tailor and clearly a much more expensive solution. 'Mysizeid' offers the chance to take measurements on your phone and then offers size recommendations for a large variety of clothing brands, however doesn't show what they would look like on you.

'Stylewhile' and 'Fits.me' attempt to solve this problem by allowing users to create an avatar to try on the clothes for them while 'Fitbay' allows users to follow their 'body doubles' to see what fits well on others (Brooke, 2019). Per-fit is unique as through the use of augmented reality you will actually be able to see the clothes on yourself in real time, as opposed to an avatar or someone similar, and experiment with different sizes and combinations to find the correct fit for you.

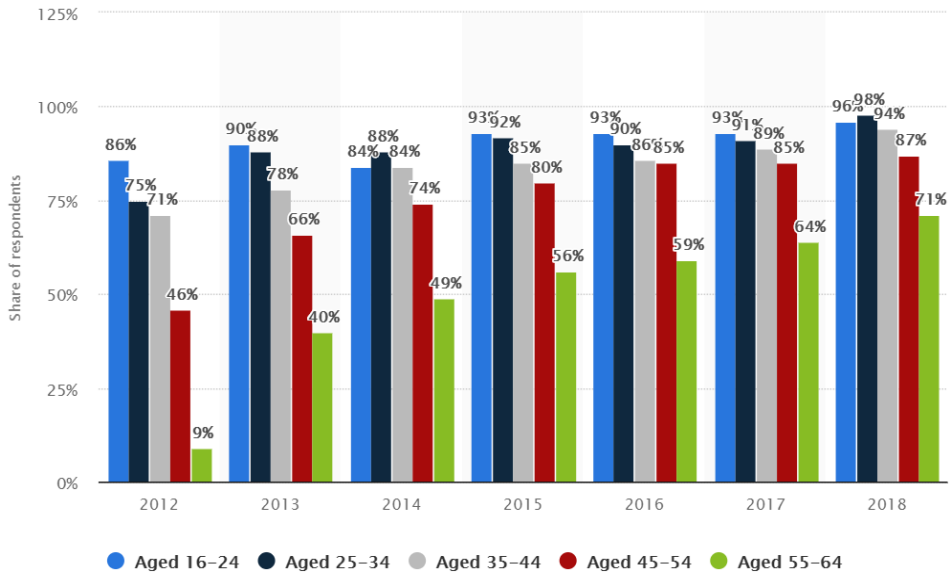


Figure 2.5: Smart phone users in the UK taken from (O'Dea, 2019)

While not trying to solve the fitting problem themselves another major competitor we are likely to face is ASOS, which hosts many brands to be purchased through their page. The biggest problem here would be ASOS' reputation and potential loyalties, which we would have to overcome through unique features and by building Per-fit's brand in order to compete.

Table I shows several different features that may be considered for a product such as Per-fit compared to its closest competitors more concisely. While many share features or a similar feature, Per-fit offers almost all these features in a much more effective and collaborative way.

Table I: Direct competitor's features

	Per-fit	ASOS	Acustom Apparel	Stylewhile	Fits.me	Thirdlove	Fitbay	Virtusize	MysizeID	Virtual fitting room
Takes measurements	✓	×	✓	×	✓	✓	×	×	✓	×
Style quiz/ preferences	✓	×	✓	×	✓	×	✓	×	×	×
Suggests sizes	✓	×	✓	×	✓	✓	×		✓	×
Suggests alternatives	✓	×	×	×	×	×	×	✓	×	×
Gives new suggestions	✓	✓	×	×	×	×	×	×	×	×
Features different brands	✓	✓	×	✓	✓	×	×	✓	✓	✓
Custom made	×	×	✓	×	×	×	×	×	×	×
Gives visual representation	✓	×	×	✓	✓	×	✓	×	×	✓
On app/site purchase	✓	✓	✓	×	×	×	×	×	×	×

3. Marketing Strategy

Freemium Strategy

All customers have full access to the clothing range and can use the augmented reality to see if clothes fit them. This attracts customers to PER-FIT's website so that they can get used to the service provided without making any paid commitments. A premium service is offered because subscribers are more likely to buy more and buy more frequently according to Weinstein, M. (2019). A premium service to users would involve a subscription of £9.99 a month which will ensure that PER-FIT acquires an expected amount of monthly revenue. The premium membership aims to give customers a tailored experience of the service that provided, and it keeps the website and app free from adverts, thereby making it a clean user-experience. A summary of what is offered by the free and premium plans can be seen in **Table II**.

Table II: Summary of what is offered by free and premium plans

Feature	Free Features	Premium Features
Use of the augmented reality technology	• Can be used for one item of clothing	• Can be used for multiple items of clothing so that a whole outfit can be viewed
Sharing of clothes	• Can share what one item of clothing looks like on the user to friends. Can also recommend items of clothing to friends	• Can share what multiple items of clothing look like on the user to a friend. Can also recommend items of clothing to friends
Newsletter	• Can sign up to a newsletter, but all recommended clothes will be showcased by a generic model. The newsletter will also contain adverts from clothing brands	• Can sign up to a newsletter and all recommended clothes will be showcased using the user's avatar. The newsletter will not contain adverts from clothing brands.
Mobile Wardrobe	• Can view all previously bought items of clothing	• Can view all previously bought items of clothing • Can pick outfits to wear while on-the-go by selecting previously bought items of clothing and placing them on their avatar
Delivery	• Free 3-5 working day delivery on all orders over £30 • No priority shipping	• Free next-day delivery on any order • Priority shipping around Christmas time
Exclusive deals	• Standard deals that are available to all customers	• Exclusive deals that are tailored to the user

Future Strategy

When PER-FIT is able to store clothes in its own warehouses, a subscription service will be offered where customers pay a £10 styling charge and clothes are sent to them based on the recommendation of PER-FIT's styling algorithm which is based on a questionnaire that they fill out when signing up. They can return any clothes that they do not want for free and they will only pay for the clothes that they keep. The customer can choose how often they wish to receive the clothes but typically this would be monthly. This will add to PER-FIT's consistent monthly revenue.

Attracting Clothing Brands

Clothing brands are attracted to use PER-FIT because it is guaranteed that less clothes will be returned by customers. PER-FIT will charge clothing brands £15 a month to display their clothes on the website, this is £5 a month less than advertised on the ASOS Marketplace. A 15% commission for all clothes bought through our website will also be charged, this number is 5% less than advertised on the ASOS Marketplace and the same price as Amazon says Weinstein, M. (2019).

Clothing brands will also be offered a subscription of £15 a month in order to get their products to be high in the recommended clothes offered to users. There is also an option for the clothing brands to pay £200 for a "Homepage Takeover" for a day.

Use of Subscriptions

The income strategy primarily involves the use of subscriptions. This is due to a guaranteed net monthly revenue independent to the amount of clothes sold.

This means that revenue will come from 3 sources:

- Usage Fee: monthly rental fees and subscriptions
- Advertising Revenues: fees charged to clothing brands for advertising on the website
- Transaction Revenues: generated from commission fees

Company Image

Social media advertising is done in elegant and comical ways in order to engage with the target market and to be seen as a trustworthy company (Forbes.com (2019)). This is further emphasised by the logo **Figure 3.1** and slogans **Figure 3.2** which contain the word "PER" which is being used as a pun and makes the brand name easy to remember.

Figure 3.1: PER-FIT logo

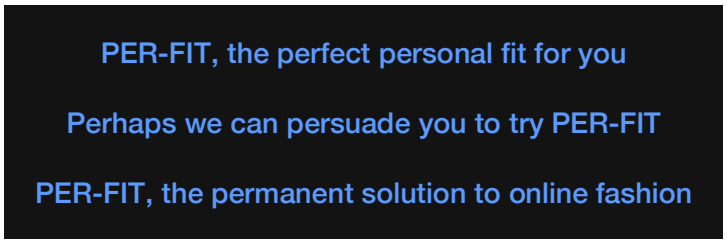


Figure 3.2: PER-FIT slogans

Advertising

Millennials "don't trust advertisements" says Grodesky, N. (2019), therefore, we will not pay for advertising online, but will instead spread our name in less intrusive ways. This method can be shown by the fact that ASOS has reduced its advertising spending from 6% to 4%, and is now focusing more on AI and machine learning to draw customers to their site says The Drum. (2019).

Customers will receive monthly newsletters and offers in order to keep them excited by the services provided, which is a proven way of keeping clients according to FourWeekMBA. (2019). The newsletter will propel the "Email Marketing Sequences" Adams, R. (2019) which helps PER-FIT to understand its customers and therefore provide them with a better service.

Magazines and Instagram pages will be encouraged to create folders with clothes that they like for customers to view. This follows in a similar style to Spotify, which allows companies to create their own playlists on their platform. These playlists boost the company's advertising and increases the content that the platform offers, according to Grodesky, N. (2019).

Instagram Advertising

Instagram will be our preferred social media platform to advertise on as it is one of the largest with "1 billion monthly active users", with "80% of users following a business account" and "64% of 18-29 year-olds" using Instagram, as seen in **Figure 3.3**. Despite Instagram only being the social media platform with the third highest active users, it has a 58% higher engagement rate than the largest social media platform Facebook, as seen in **Figure 3.4**, while still keeping the cost per sponsored post per

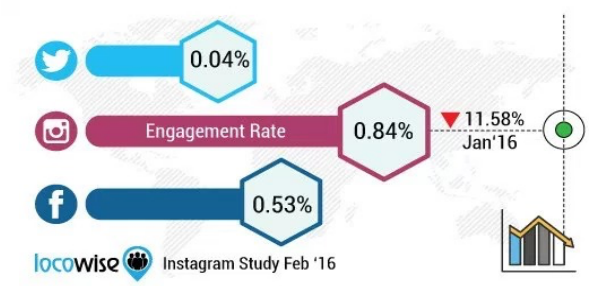


Figure 3.3: Social media platform engagement rate taken from Jolly. W.

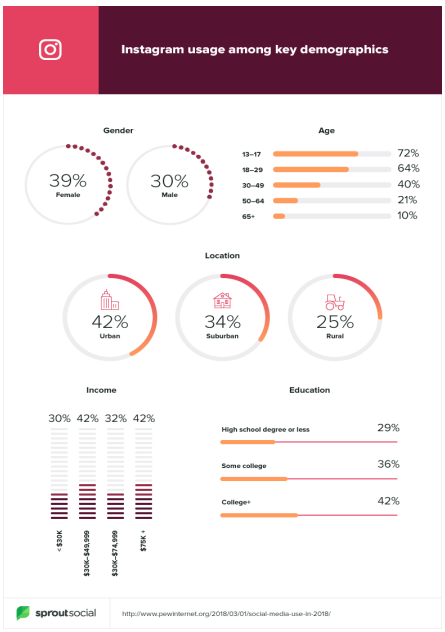


Figure 3.4: Instagram usage among key demographics taken from Sprout Social (2019)

1000 followers at $10 which is the same as Facebook and half that of Youtube, as stated by Webfx.com. Our advertising money will be spent on getting 'micro-influencers (1000 to 100,000 followers) to recommend our product and tag our page in their post. The cost of paying the micro-influencers is significantly lower to high profile celebrities but they still have the potential to spread PER-FIT's name effectively. Payment of the 'micro-influencers' will be done on a pay-per-view method, as it is the most popular and is the method most preferred by influencers and businesses according to Webfx.com. (2019).

The content on PER-FIT's Instagram page will be predominantly video based as it receives "38% more engagement than image posts". When the Instagram page has reached 1000 followers, which is the number of followers which businesses and 'micro-influencers' would typically have according to WorkMacro. (2019), then competitions will be introduced to further spread PER-FIT's name. Competitions would include users trying to find the funniest outfits on our website and uploading a picture of them with the item of clothing shown on them using the augmented reality technology. The user will then share the picture to social media and use PER-FIT's hashtag. This sort of competition has a comedic effect which will not seem like advertising, while at the same time showcasing PER-FIT's augmented reality technology and encourage customers to browse the selection of clothing on the website. An example of the how PER-FIT's Instagram page will look like can be seen in **Figure 3.5**.

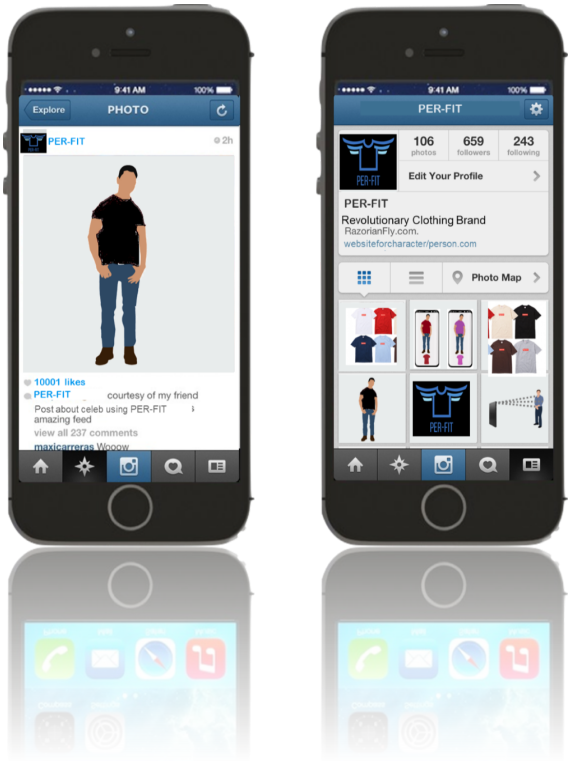


Figure 3.5: Example of PER-FIT Instagram page. Taken from Mr. Akans Online (2019)

4. Fiscal Matters

The required start-up monthly expenditure is £53,752, this includes the total expenses and cost of developing the application. Capital expenditure of £12,000 is required to purchase equipment such as computers in the first month. Marketing expenses are set to increase in line with net profit, in year 2 a second marketing salesperson will be brought in. The predicted monthly expenditure is set to rise by 2% each year in line with inflation.

Activity-based costing estimates the cost of developing the application. The cost of each essential feature to produce a minimum viable product is calculated by analysing application development surveys (Craigmile, 2015), and professional application development estimators (Howmuchtobuildanapp.io, 2019), (Waspmobile.com, 2019). This determined a total driver cost of 3110 application developer hours, this indicates 14 (37.5 hours) weeks for 6 application developer to complete. After initial development, the team will begin working on application maintenance and future features.

Table III: Application Development Cost

Stage	Activity Cost Pool	Cost Drivers (Developer Hours)
Design	User Experience	200
	Visual Design	210
Features	Secure User Login Authentication	50
	Payment	220
	Native Device Features	50
	Use of Location Data	100
	AI Recommendation	500
	User Engagement (email, SMS)	150
	Camera	40
	Augmented Reality	700
Infrastructure	Third-party API integration (social media, online marketplaces)	130
	Data storage	200
	Data Encryption	60
	Performance Analytics and Management	70
Testing	Functionality Testing	400
Deployment	Upload to Market	30
Total Hours		3110

The financial forecast for the company's first 3 years of monthly active users and revenue has been carefully estimated against the market analysis and finical analysis of similar firms. The market strategy PER-FIT has opted for suggests a similar active user growth to (Spotify Goodwater Capital, 2018), this is justified through the similar premium subscription business model. This growth is expected to increase during traditional fashion season transitions from the spring/summer collection (Jan/Jun) and the autumn/winter collection (Jul/Dec). The predicted spending per consumer this estimated by initial ASOS retail sales (Sabanoglu, T, 2019) and the number of ASOS consumers (Media, 2019). The average active user spending per month increases from £7.43 in the first year of roll out to £8.10 by the third. The estimated revenue generated through advertisements per ad-supported monthly active user of PER-FIT is £0.30. This value has been estimated against the revenue generated per user by the advertisement-supported version of Spotify (Spotify Goodwater Capital, 2018), because Spotify implement a similar market strategy to the free version of PER-FIT.

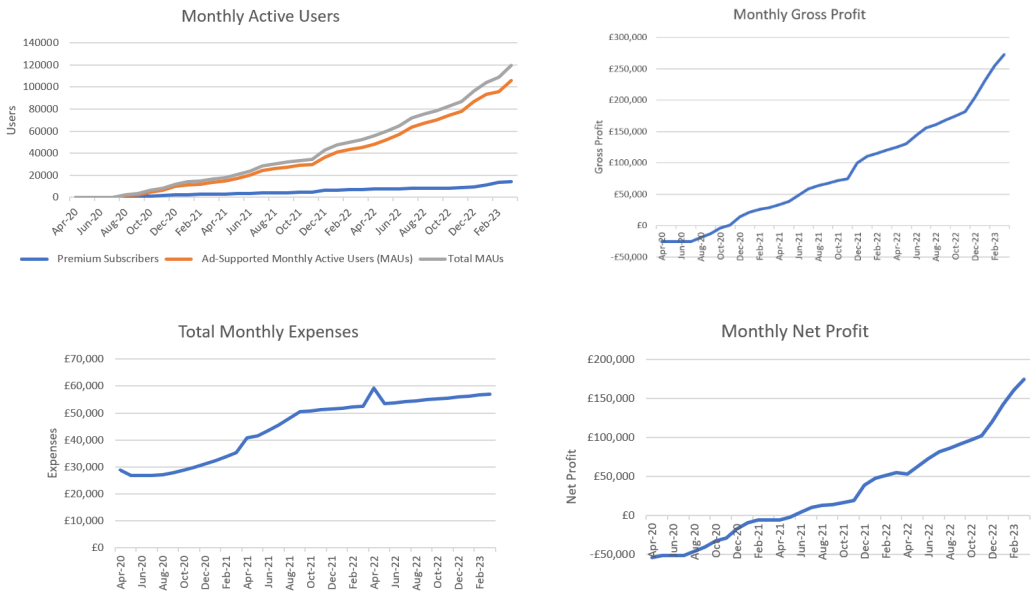


Figure 4.1: Monthly active users, monthly gross profit, total monthly expenses and monthly net profit.

Table IV: ASOS average customer spending per month

Year	Retail Sales (thousands)	Number of consumers (thousands GBP)	Average Spending per Consumer per Month (GBP)
2013	276027	3094	7.43
2014	372241	3917	7.92
2015	473884	4874	8.10

The investors will hold a 20% stake in the company, their dividends will equal 15% of the net profit of the company in that working year. An initial funding of £410,000 will fund this company to 1.2 years, at which point the company becomes profitable. The payback period for this investment will be 2.2 years. (**See Appendix B**).

Table V: Net profit, IRR and NPV

Net Profit	£1,112,581
Internal Rate of Return	30.58%
Net Present Value	£465,363.98

After 3 years PER-FIT has been forecasted to gain over 100,000 monthly active users, including 14,000 premium subscribers, and set to earn over £1,000,000 in net profit. Reviewing the statistics of mobile application usage, downloads and retention an appropriate discount rate estimation would be 7%, (Iqbal, 2019). This concludes PER-FIT to hold an evidential supported financial forecast with promising figures including an encouraging internal rate of return and a tremendous potential net present value.

5. Risk Analysis

Perceived Risk

A significant risk for PER-FIT is the consumer's need for it, as this will determine the success of this product. Market analysis indicates the online clothes marketplace holds a considerable proportion of the clothing market. By applying our market strategy, PER-FIT will develop a strong consumer audience, taking on consumer feedback to further develop our product.

Funding and Competition

A major risk of this company is gaining enough capital through venture capitalists to take our business idea from concept stage to implementation stage. Especially in this competitive market, with other companies gaining funding for the development of similar products. However, developing the innovative augmented reality sizing for a mobile application separates PER-FIT from the competitors to both gain funding and sales.

Clothing Retailer Support

Gaining support from other online clothes retailers to sell their products on the proposed platform is another risk that will determine the success of this company. Strategies to entice suppliers will be offered, such as advertising their clothes in people's recommendations, and 'homepage takeover' to promote their clothing for reasonable prices.

Augmented Reality Clothes Sizing

Developing the innovative augmented reality clothes sizing feature proposes a significant challenge. However, this technology is feasible because mobile applications have been developed that accurately measure an object using the mobile camera. To ensure this essential feature is complete the development team will be specially selected to provide on the necessary skills to complete it. Also, management techniques such as Agile and Gantt Charts will ensure the team stay on schedule.

Table VI: Risk, probability and impact assessment

Risks		Likelihood (1-5)	Impact (1-5)	Overall Risk (1-25)	Risk Level
R1	Consumer Need	1	4	4	Low
R2	Funding	2	4	8	Moderate
R3	Competition	2	2	4	Low
R4	Online Clothes Retailer Support	3	5	15	High
R5	Augment Reality Sizing	2	4	8	Moderate
R6	Deadline Completion	2	3	6	Low

Risk Level	Overall Risk
Low	0-6
Moderate	7-14
High	15-25

6. Intellectual Property

To prevent loss of revenue via IP theft several protection methods will be adopted (Government, accessed 11/2018a).

Trademark and Copyright Legal Issues

Trademark & Design Rights

For exclusive use of PER-FIT's name, slogan and logo registration will be made to the Trade Marks Registry under the *Trade Marks Act 1994* which 'protects the owner of any sign capable of being represented graphically which is capable of distinguishing goods or services of one undertaking from those of another.' Once registered the duration of each of the Trade Marks is 10 years, renewable for additional periods of 10 years indefinitely. Any additional material developed by PER-FIT's employees will be registered to the company itself, preventing competitors from copying PER-FIT and passing off as their own (Government, accessed 11/2018b).

To protect PER-Fit's visual look new original designs having individual character will be registered at the Patent Office (Design Registry) under the *Registered Designs Act 1949*. The interface design will be specifically trademarked for its colours, layout and its overall complete design. Registered designs will be protected for a duration of 25 years. Note: There is an automatic right to unregistered 3D designs for 15 years.

Copyright

The code that runs PER-FIT's platform must be and will be kept private. Under the Copyright law governed by the *Copyright Designs and Patents Act (CDPA) 1988*, 'all literary works are an automatic right to the owner given that proof of ownership is established,' hence all code will be recorded. All PER-FIT employees and any independent freelancer working on code for PER-FIT will only be contracted after signing a legal document that binds them to *passing over* any copyright to PER-FIT.

Copyright protection adheres for the life of the author in addition to 70 years post death of author.

Additional Legal Protection

To further protect PER-FIT there will be constant aggressive pursuit of developers intending to infringe on the company. Machine learning algorithms will be compiled to machine code for support on various operating systems and the code used will be made difficult, if not impossible, to comprehend using sophisticated encryption techniques. All employees working on PER-FIT's code will have to sign a confidentiality agreement legal contract in order to protect the company against code leakages. Example of confidential information case: *Fraser V Thames Television (1983).*

The aim is to prevent clone applications appearing and immediately take down such applications that do appear.

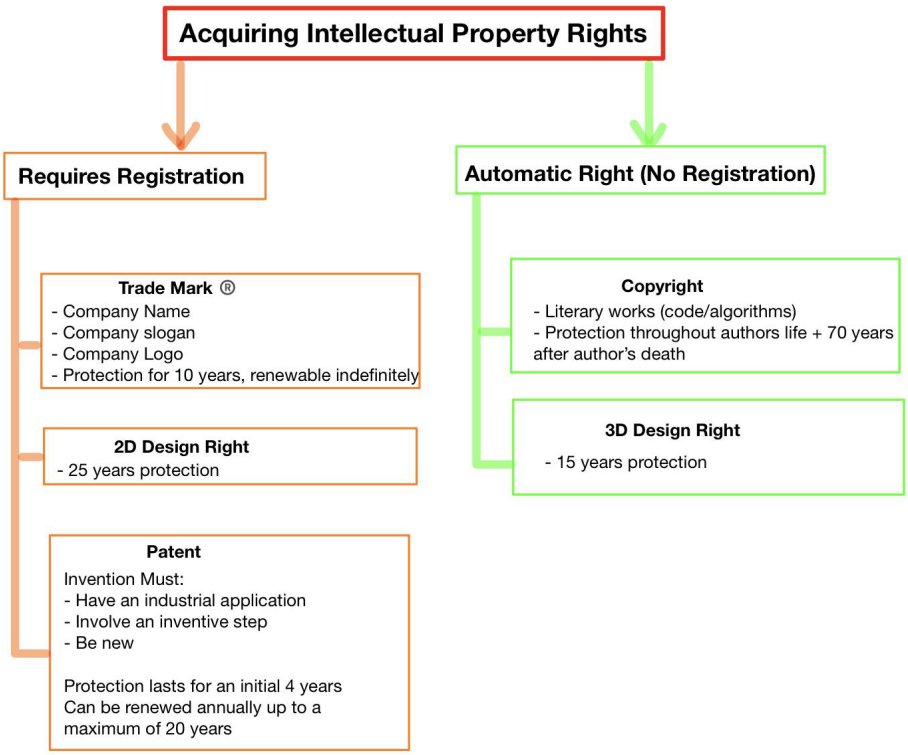


Figure 6.1: Summarised Intellectual Property Acquisition

General Data Protection Regulation (GDPR)

To ensure lawful user data protection PER-FIT will abide by all regulations stated under the *new GDPR, May 2018*. Users' will be prompted with 'Terms & Conditions' to which they must accept in order to use the application, allowing PER-FIT to store their data.

Sophisticated data encryption methods will be enforced to securely store users' data to prevent intercepted information from being understood.

Monopolisation by Patent Outreach

Patent protection will be sought for complete security of the entire concept aiming to revolutionise the fashion industry. To be granted a patent an invention must have an industrial application, involve an inventive step and be new

If granted, the patent protection lasts an initial 4 years after which it can be renewed annually to a maximum of 20 years.

7. Sustainability and Environmental Impact

The nature of PER-FIT, being an application, makes it relatively environmentally friendly in terms of the raw product itself. User's will simply be downloading the application through an app store that they can access on their mobile phones.

PER-FIT is designed to promote sustainability and reduce environmental issues by increasing the quality of the outreach of fashion consumer goods to individuals and reducing the overall carbon footprint of both the consumers and vendors, as seen in **Table VII**.

Table VII: Weighing Sustainable and Environmental Impact of PER-FIT

Sustainable & Environmental Impact	
Benefits	**Drawbacks**
Online fashion purchasing made more convenient	Increase in overall delivery of goods, increasing carbon footprint
Individuals will be more confident in purchasing fashion products online	Individuals with less sophisticated cameras/sensors will be at a disadvantage
Increased confidence will also increase the overall purchasing of products for individuals that are simply too far to commute to	
Increased online sales will reduce the volume of individuals commuting to stores, reducing carbon footprint	
Broader product choice for individuals, offering variety for the consumer	
Less commercial building space wasted by vendors as more sustainable warehouses would be ideal for storing items, improving space per unit item stored	

8. Evaluation

Team style and structure

Our team decided to choose a team structure that has a leading figure (project manager - Agile Project Management), allowing for decisions to be made definitively if conflict arose and keeping each member on track with their allocated role. We kept in communication via an online group chat to allow for quick response to queries regarding the proposal. Alongside this, we met up in weekly face-to-face meetings at a set time allowing for progress checks, making sure everyone was still clear on the product and if anything was unclear, questions could be asked. We chose to have face-to-face meetings as well as online group chat as studies have proven that face-to-face meetings are more effective in group communication - "overall impact of computer-mediated communication indicates that its use is associated with more negative work outcomes than occur in face-to-face groups" (Baltes et al., 2002); as well as being a principle in Agile Project Management. Social-bonding activities (e.g.pub meet up) were implemented to increase chemistry within the team and lead to better morale and standard of work.
If members were unable to make the weekly meetings due to other commitments, we would pass on meeting notes to the member who was not present, allowing the meeting to go ahead. This occurred once and passing on meeting notes proved to be an effective project management technique.

Skills and task allocation

Firstly, as a team we discussed and split our report/proposal into areas: Market Analysis, Market Strategy, Product Development, Law, Accounting & Finance and Team Management/Evaluation. Everyone had a clear end goal for our proposal, effective Agile project planning. We then decided to use a skills matrix (seen in **Figure 8.1**), "a grid or table that clearly and visibly illustrates the skills and competence held by individuals within a team" (Skillsmatrix.info, n.d.) to be able to find out each other's strengths and weaknesses. By adding relevant skills under each role and using a score weighting to define each member's competency in that skill, we calculated which member would fit the role the best by their total role score.

Team Members	Accounting & Finance	Accounting maths	Budgeting	Organisation	Product Development	Creativity	Idea Clarity	Methodical	Market Analysis	Working with data	Researching	Good with statistics	Market Strategy	Data Analysis	Communication	Multi-tasking	Law	Analytical & Logical Reasoning	Knowledge of Substantive Law & Legal Procedure	Commercial Awareness	Team/Management Analysis	Analysing performance	Evaluating performance	Report writing
David		1	1	2		3	2	2		3	2	3		2	1	2		1	0	0		2	1	1
Etienne		1	1	3		1	2	2		3	1	1		2	3	2		0	2	2		2	3	1
Rahul		1	1	2		1	2	2		1	2	2		2	2	1		2	3	3		2	0	3
Rohan		2	1	2		3	3	2		2	1	2		2	1	1		0	0	0		1	1	1
Tom		3	3	3		1	2	2		3	2	3		3	1	1		2	1	0		2	1	1
Will		2	1	2		1	1	1		2	1	2		2	1	1		1	1	1		2	3	2

Score Definitions
0 - No capability
1 - Basic capability
2 - Intermediate capability
3 - Advanced capability

Figure 8.1: Skills Matrix, each role has relevant skills which each member filled out their capability in those skills using Score

In **Figure 8.1** above, green highlight represents the member with the highest score in that role. For Market Analysis, both Tom and David scored the highest (score = 8) meaning they are of similar capability to carry out the role. However, as Tom scored highest in Accounting & Finance, we all decided that Tom would be more suitable to carry out that section which Tom was happy about, and therefore resolving the conflict. Upon review, using a skills matrix to delegate jobs to each member was very effective within our team as members carried out their roles to a very good standard. To improve, we could have added a larger skill pool to each role as some members were proficient in the same proposal area, meaning there wasn't a distinct reason to allocate these members to that specific area. As well as this, adding a "Role Interest" weighting to the scores would have allowed for people not being allocated a role in which they have no interest in, reducing productivity. However, lack of interest in proposal area was not apparent within our team.

Figure 8.2: Gantt Chart of proposed workload over coursework period

To manage time and workload, we implemented a Gantt Chart allowing each member to visually schedule their work within a chosen time frame; in effect being the roadmap for the development of the proposal (Agile project management). The time frame ended short of the report hand-in to allow for team review of each section as well as report structure. Each member would fill in the Gantt chart for themselves, estimating the effort required for each sub-task (unit of weeks). This was used as a tool at meetings to check if people had carried out their task for the week or if anyone was over-burdened, they could collaborate with someone who was comfortable with the added workload.

How conflicts were resolved

Within our team, some conflicts did arise however we had effective techniques to resolve them. When choosing the name of our product Rohan wanted PER-FIT but David wanted E-FIT; to resolve this conflict David and Rohan both presented the reasoning behind their name for the product and a decision was made by the remaining members of the group by a vote (a fair, reasonable resolution).

Another example of conflict resolution was the design of the logo as this would be a key part of our marketing strategy. Will and Etienne wanted a logo which represented the business and not the service as this would narrow expansion opportunities in the future. Rohan and Tom wanted the logo to represent the service and clothing industry (a logo with clothing in it). The solution to this was a compromise of a logo which contains clothing but doesn't limit future business endeavours.

Overall reflection

Upon reflection, having Agile project management was very beneficial. It allowed us to split our workload effectively, improve team communication and individual accountability (face-to-face meetings) and have a common final goal which is the key measure of progress for the team. In contrast, the Agile technique we didn't choose to have was daily meetings due to other commitments, we chose to have waterfall management approach to final integration of the report (one date for final integration).

Overall, our team worked extremely well together to complete our task of creating a start-up business proposal PER-FIT. We effectively implemented team and project management techniques to ensure successful delivery of the proposal to a high business standard.

9. References

Black Shoes. (2019). [image] Available at: https://www.dsw.com/en/us/product/vans-ward-lo-suede-sneaker---womens/404995 [**Accessed 15 Nov. 2019**].

Brandcrowd.com. (2019). Flying Shirt Logo | BrandCrowd Logo Maker. [online] Available at: https://www.brandcrowd.com/maker/logo/cb1b6a67-b8e4-43c1-ab6d-5a67ae87331b?text=PERFIT [**Accessed 15 Nov. 2019**].

Casual Man Standing. (2019). [image] Available at: https://www.digimadmedia.com/vector-Casual+Man+Standing [Accessed 15 Nov. 2019].

Dark Blue Tee. (2019). [image] Available at: https://www.vistaprint.in/clothing-bags/mens-t-shirts/anvil-ring-spun-ink-printed-lightweight-mens-t-shirt?GP=11%2f15%2f2019+09%3a03%3a07&GPS=5533166027&GNF=0 [**Accessed 15 Nov. 2019**].

Man Standing. (2019). [image] Available at: https://www.digimadmedia.com/vector-Man+Standing+Illustration [**Accessed 15 Nov. 2019**].

Phone Template. (2019). [image] Available at: https://i5.walmartimages.com/asr/36db44e7-8ba9-4a6f-87fa-81317e923916_1.416fcdd938fb7f63573d3d937a2c8283.jpeg?odnHeight=450&odnWidth=450&odnBg=ffffff [**Accessed 15 Nov. 2019**].

Coloured Tees. (2019). [image] Available at: https://www.4imprint.com/product/6729-S-C-A/Hanes-Tagless-T-Shirt-Screen-Colors [Accessed 15 Nov. 2019].

Supreme Tee. (2019). [image] Available at: http://streetwearvilla.com/supreme-box-logo-t-shirt-white [**Accessed 15 Nov. 2019**].

Ec.europa.eu. (2019). *E-commerce statistics for individuals - Statistics Explained*. [online] Available at: https://ec.europa.eu/eurostat/statistics-explained/index.php/E-commerce_statistics_for_individuals#Most_popular_online_purchases [**Accessed 12 Nov. 2019**].

Charlton, G. (2019). *Online Fashion Retail: 11 Essential Statistics - SaleCycle*. [online] SaleCycle. Available at: https://blog.salecycle.com/featured/online-fashion-retail-11-essential-statistics/ [**Accessed 12 Nov. 2019**].

Hope, K. (2019). *The people who return most of what they buy*. [online] BBC News. Available at: https://www.bbc.co.uk/news/business-46279638 [**Accessed 12 Nov. 2019**].

Charlton, G. (2019). *Ecommerce Returns: 2018 Stats and Trends - SaleCycle*. [online] SaleCycle. Available at: https://blog.salecycle.com/featured/ecommerce-returns-2018-stats-trends/ [**Accessed 12 Nov. 2019**].

Sabunoglu, T. (2019). *Online clothing purchasing by demographic Great Britain 2019 | Statista*. [online] Statista. Available at: https://www.statista.com/statistics/286096/clothes-and-sports-goods-online-purchasing-in-great-britain-by-demographic/ [**Accessed 12 Nov. 2019**].

O'Dea, S. (2019). *Smartphone usage in the UK 2011-2018 | Statista*. [online] Statista. Available at: https://www.statista.com/statistics/300398/smartphone-usage-in-the-united-kingdom/ [**Accessed 12 Nov. 2019**].

Brooke, E. (2019). *8 Startups Trying To Help You Find Clothing That Fits*. [online] Fashionista. Available at: https://fashionista.com/2014/07/8-tech-startups-tackling-clothing-fit [**Accessed 12 Nov. 2019**].

Weinstein, M. (2019). *How to Sell on Amazon: Everything You Need to Know for 2019 | Tinuiti*. [online] Tinuiti. Available at: https://tinuiti.com/blog/amazon/sell-on-amazon/ [**Accessed 15 Nov. 2019**].

FourWeekMBA. (2019). *How To Scale Your Business With A Subscription Business Model - FourWeekMBA*. [online] Available at: https://fourweekmba.com/subscription-business-mdel/ [**Accessed 15 Nov. 2019**].

Smart Insights. (2019). *Spotify Case Study - Marketing*. [online] Available at: https://www.smartinsights.com/digital-marketing-strategy/online-business-revenue-models/spotify-case-study/ [**Accessed 15 Nov. 2019**].

Medium. (2019). *Spotify's Innovative marketing and revenue model*. [online] Available at: https://medium.com/@sho0by/spotifys-innovative-marketing-and-revenue-model-b3351bbc968d [**Accessed 15 Nov. 2019**].

Grodesky, N. (2019). *How to Advertise on Spotify | Power Digital*. [online] Power Digital. Available at: https://powerdigitalmarketing.com/blog/how-to-advertise-on-spotify/ [**Accessed 15 Nov. 2019**].

Digiday. (2019). *What Online Ads Really Cost - Digiday*. [online] Available at: https://digiday.com/media/what-online-ads-really-cost/ [Accessed 15 Nov. 2019].

The Drum. (2019). *Asos reduces ad spend – here's where it's investing instead*. [online] Available at: https://www.thedrum.com/news/2018/10/18/asos-reduces-ad-spend-here-s-where-it-s-investing-instead [**Accessed 15 Nov. 2019**].

Webfx.com. (2019). *Influencer Marketing Pricing: What Does It Cost in 2019?*. [online] Available at: https://www.webfx.com/influencer-marketing-pricing.html [**Accessed 15 Nov. 2019**].

WorkMacro. (2019). *What Your Follower/Following Ratio Say About Your Instagram Account*. [online] Available at: https://workmacro.com/instagram/follower-following-ratio-say-instagram-account/ [**Accessed 15 Nov. 2019**].

Adams, R. (2019). *10 Marketing Strategies to Fuel Your Business Growth*. [online] Entrepreneur. Available at: https://www.entrepreneur.com/article/299335 [**Accessed 15 Nov. 2019**].

Forbes.com. (2019). *The CMO's Guide To Using Humor In Marketing*. [online] Available at: https://www.forbes.com/sites/steveolenski/2018/06/15/the-cmos-guide-to-using-humor-in-marketing/#6b7af0ae62bf [**Accessed 16 Nov. 2019**].

Mr. Akans Online. (2019). *Templates for Fake Twitter/Facebook/Instagram pages*. [online] Available at: http://www.mrakansonline.com/20132014/templates-for-fake-twitterfacebookinstagram-pages [**Accessed 15 Nov. 2019**].

Sprout Social (2019). *17 Instagram stats marketers need to know for 2019*. [online] Sprout Social. Available at: https://sproutsocial.com/insights/instagram-stats/ [**Accessed 16 Nov. 2019**].

Jolly, W. (2019). *The 6 Most Effective Types of Social Media Advertising in 2019*. [online] The BigCommerce Blog. Available at: https://www.bigcommerce.co.uk/blog/social-media-advertising/#2-instagram-advertising [Accessed 16 Nov. 2019].

Craigmile, N. (2015). *Cost to Build a Mobile App: A Survey*. [online] Clutch.co. Available at: https://clutch.co/app-developers/resources/cost-build-mobile-app-survey-2015 [**Accessed 13 Nov. 2019**].

Howmuchtobuildanapp.io. (2019). *How much does it cost to make an app?* [online] Available at: http://www.howmuchtobuildanapp.io/ [Accessed 11 Nov. 2019].

Waspmobile.com. (2019). *Wasp Mobile - App Cost Calculator*. [online] Available at: https://waspmobile.com/app/ [**Accessed 10 Nov. 2019**].

Goodwater Capital. (2018). *Understanding Spotify: Making Music Through Innovation*. [online] Available at: https://www.goodwatercap.com/thesis/understanding-spotify [**Accessed 15 Nov. 2019**].

Sabanoglu, T. (2019). *ASOS retail sales in the UK 2013-2019 | Statista*. [online] Statista. Available at: https://www.statista.com/statistics/500728/asos-retail-sales-in-the-united-kingdom-uk/ [**Accessed 17 Nov. 2019**].

Media, K. (2019). *Asos products purchased in the UK, by clothing type 2013-2018 survey | Statista*. [online] Statista. Available at: https://www.statista.com/statistics/312667/asos-products-purchased-uk-by-clothing-product-type/ [**Accessed 15 Nov. 2019**].

Iqbal, M. (2019). App Download and Usage Statistics (2019). [online] Business of Apps. Available at: https://www.businessofapps.com/data/app-statistics/ [**Accessed 14 Nov. 2019**].

Bond, K. (2019). Copyright Law Basics For UK Software Developers — Smashing Magazine. [online] Smashing Magazine. Available at: https://www.smashingmagazine.com/2018/03/copyright-law-basics-for-uk-software-developers/[**Accessed 25 Oct. 2019**].

Anon, (2019). [online] Available at: http://blog.thetrademarkhub.com/intellectual-property/7-ways-to-legally-protect-your-mobile-app/[Accessed 29 Nov. 2019].

Resourcecentre.org.uk. (2019). Data protection for community groups | Resource Centre. [online] Available at: https://www.resourcecentre.org.uk/information/data-protection-for-community-groups/[**Accessed 3 Nov. 2019**].

Gdpr.co.uk. (2019). GDPR.co.uk - Privacy Policy. [online] Available at: https://gdpr.co.uk/privacy-policy/[Accessed 8 Nov. 2019].

Baltes, B., Dickson, M., Sherman, M., Bauer, C. and LaGanke, J. (2002). Computer-Mediated Communication and Group Decision Making: A Meta-Analysis. Organizational Behavior and Human Decision Processes, 87(1), pp.156-179.

Skillsmatrix.info. (n.d.). What is a Skills Matrix and Effective Skills Management.. [online] Available at: https://skillsmatrix.info/ [**Accessed 9 Nov. 2019**]

Appendix A

Internal Rate of Return

$$0 = NPV = \sum_{n=0}^{3} \frac{Cash\ Flow_n}{(1+IRR)^n} = Cash\ Flow_0 + \frac{Cash\ Flow_1}{(1+IRR)^1} + \frac{Cash\ Flow_2}{(1+IRR)^2} + \frac{Cash\ Flow_3}{(1+IRR)^3}$$

Where,

$Cash\ Flow_0 = Initial\ Investment$

$Cash\ Flow_n = Cash\ Flow$

$n = Yearly\ Period$

$NPV = Net\ Profit\ Value$

$IRR = Internal\ Rate\ of\ Return$

Table VIII: Cash Flow

Year	Cash Flow
0	(£410,000)
1	(£395,278)
2	262,099
3	1,244,484

$$0 = -410000 + \frac{-395278}{(1+IRR)} + \frac{262099}{(1+IRR)^2} + \frac{1244484}{(1+IRR)^3}$$

$$0 = -410000(1+IRR)^3 - 395278(I+IRR)^2 + 262099(1+IRR) + 1244484$$

$$0 = -410000(IRR^3 + 3IRR^2 + 3IRR + 1) - 395278(IRR^2 + 2IRR + 1) + 262099(1+IRR) + 1244484$$

$$0 = -410000IRR^3 - 1625278IRR^2 - 1758457 + 701305$$

$$IRR = 0.30575 = 30.575\%$$

Net Present Value

Table VIIII: Net Present Value

Year	Discount Rate	Cash Flow	Present Value
0	1.0000	(£410,000)	(£410,000)
1	0.9346	(£395,278)	(£369,426.81)
2	0.8734	£262,099	£228,917.92
3	0.8163	£1,244,484	£1,015,872.87
Net Present Value			**£465,363.98**

3 Year Financial Forecast

	Apr-20	May-20	Jun-20	Jul-20	Aug-20	Sep-20	Oct-20	Nov-20	Dec-20	Jan-21	Feb-21	Mar-21	Apr-21	May-21	Jun-21	Jul-21	Aug-21
Users																	
Premium Subscribers	0	0	0	0	500	1050	1360	1510	2330	2590	2780	2890	3010	3340	3690	3950	4200
Ad-Supported Monthly Active Users (MAUs)	0	0	0	0	1600	2300	5000	6500	9800	11500	12000	13500	15000	17400	20100	24500	26000
Total MAUs	**0**	**0**	**0**	**0**	**2100**	**3350**	**6360**	**8010**	**12130**	**14090**	**14780**	**16390**	**18010**	**20740**	**23790**	**28450**	**30200**
Revenue																	
Retail Sale Commission (%15)	£0.00	£0.00	£0.00	£0.00	£2,028.39	£3,235.77	£6,143.12	£7,736.86	£11,716.37	£13,609.53	£14,276.00	£15,831.10	£17,395.86	£20,032.77	£22,978.76	£27,479.86	£31,093.92
Brand Promotions	£0.00	£0.00	£0.00	£0.00	£100.00	£1,100.00	£4,950.00	£7,425.00	£11,137.50	£15,000.00	£18,450.00	£18,634.50	£18,820.85	£19,009.05	£23,500.00	£26,400.00	£26,928.00
Premium User	£0.00	£0.00	£0.00	£0.00	£4,995.00	£10,489.50	£13,586.40	£15,084.90	£23,276.70	£25,874.10	£27,772.20	£28,871.10	£30,069.90	£33,366.60	£36,863.10	£39,460.50	£41,958.00
Ad-Supported MAU	£0.00	£0.00	£0.00	£0.00	£480.00	£690.00	£1,500.00	£1,950.00	£2,940.00	£3,450.00	£3,600.00	£4,050.00	£4,500.00	£5,220.00	£6,030.00	£7,350.00	£7,800.00
Total Revenue Ex VAT	**£0**	**£0**	**£0**	**£0**	**£7,603**	**£15,515**	**£26,180**	**£32,197**	**£49,071**	**£57,934**	**£64,098**	**£67,387**	**£70,787**	**£77,628**	**£89,372**	**£100,690**	**£107,780**
Total Revenue In VAT	**£0**	**£0**	**£0**	**£0**	**£6,336**	**£12,929**	**£21,816**	**£26,831**	**£40,892**	**£48,278**	**£53,415**	**£56,156**	**£58,989**	**£64,690**	**£74,477**	**£83,909**	**£89,817**
Cost Sales																	
Staff Costs	£21,000	£21,000	£21,000	£21,000	£21,000	£21,000	£21,000	£21,000	£21,000	£21,000	£21,000	£21,000	£21,420	£21,420	£21,420	£21,420	£21,420
Employer's National Insurance (13% of Staff Costs)	£2,730	£2,730	£2,730	£2,730	£2,730	£2,730	£2,730	£2,730	£2,730	£2,730	£2,730	£2,730	£2,785	£2,785	£2,785	£2,785	£2,785
Employer's Pension Contributions (5% of Staff Costs)	£1,050	£1,050	£1,050	£1,050	£1,050	£1,050	£1,050	£1,050	£1,050	£1,050	£1,050	£1,050	£1,071	£1,071	£1,071	£1,071	£1,071
Software licences	£47	£47	£47	£47	£47	£47	£47	£47	£47	£47	£47	£47	£48	£48	£48	£48	£48
Cloud Service	£30	£30	£30	£30	£32	£50	£95	£120	£182	£211	£222	£246	£270	£311	£357	£427	£453
Equipment rental	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0
Bank fees per sale (2% of sales)	£0	£0	£0	£0	£312	£498	£945	£1,190	£1,803	£2,094	£2,196	£2,436	£2,676	£3,082	£3,535	£4,228	£4,488
Total Cost of Sales	**£24,857**	**£24,857**	**£24,857**	**£24,857**	**£25,171**	**£25,375**	**£25,868**	**£26,138**	**£26,812**	**£27,133**	**£27,245**	**£27,509**	**£25,594**	**£25,635**	**£25,681**	**£25,751**	**£25,777**
Gross Profit	**-£24,857**	**-£24,857**	**-£24,857**	**-£24,857**	**-£18,835**	**-£12,446**	**-£4,052**	**£693**	**£14,080**	**£21,146**	**£26,170**	**£28,647**	**£33,395**	**£39,055**	**£48,796**	**£58,158**	**£64,040**
Gross Margin	**0%**	**0%**	**0%**	**0%**	**-297%**	**-96%**	**-19%**	**3%**	**34%**	**44%**	**49%**	**51%**	**57%**	**60%**	**66%**	**69%**	**71%**
Expenses																	
Rent (Serviced Office)	£400	£400	£400	£400	£400	£400	£400	£400	£400	£400	£400	£400	£408	£408	£408	£408	£408
Business Rates	£200	£200	£200	£200	£200	£200	£200	£200	£200	£200	£200	£200	£204	£204	£204	£204	£204
Utility Bills	£150	£150	£150	£150	£150	£150	£150	£150	£150	£150	£150	£150	£153	£153	£153	£153	£153
Employer' Liability Insurance	£25	£25	£25	£25	£25	£25	£25	£25	£25	£25	£25	£25	£26	£26	£26	£26	£26
Travel Costs	£500	£500	£500	£500	£500	£500	£500	£500	£500	£500	£500	£500	£510	£510	£510	£510	£510
Capital Expenditure	£12,000	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£1,000	£0	£0	£0	£0
Accountants fees	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,100	£5,100	£5,100	£5,100	£5,100
Marketing & Advertising	£0	£10,000	£10,000	£10,000	£10,100	£11,009	£12,000	£13,080	£14,257	£15,540	£16,939	£18,463	£20,125	£21,936	£23,910	£26,062	£28,408
Overhead staff:-																	
Sales & Marketing Manager	4000	£4,000	£4,000	£4,000	£4,000	£4,000	£4,000	£4,000	£4,000	£4,000	£4,000	£4,000	£4,080	£4,080	£4,080	£4,080	£4,080
Sales Person	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£2,000	£2,000	£2,000	£2,000	£2,000
Managing Director	5000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,000	£5,100	£5,100	£5,100	£5,100	£5,100
Employer's National Insurance (13% of Staff Overhead)	£1,170	£1,170	£1,170	£1,170	£1,170	£1,170	£1,170	£1,170	£1,170	£1,170	£1,170	£1,170	£1,453	£1,453	£1,453	£1,453	£1,453
Employer's Pension Contributions (5% of Staff Overhead)	£450	£450	£450	£450	£450	£450	£450	£450	£450	£450	£450	£450	£559	£559	£559	£559	£559
Total Expenses	**£28,895**	**£26,895**	**£26,895**	**£26,895**	**£26,995**	**£27,904**	**£28,895**	**£29,975**	**£31,152**	**£32,435**	**£33,834**	**£35,358**	**£40,718**	**£41,529**	**£43,503**	**£45,655**	**£48,001**
Profit before Tax	**-£53,752**	**-£51,752**	**-£51,752**	**-£51,752**	**-£45,830**	**-£40,350**	**-£32,946**	**-£29,282**	**-£17,072**	**-£11,290**	**-£7,664**	**-£6,711**	**-£7,323**	**-£2,474**	**£5,292**	**£12,503**	**£16,039**
Corporation Tax (19% of Profit)	£0.00	£0.00	£0.00	£0.00	£0.00	£0.00	£0.00	£0.00	£0.00	-£2,145.02	-£1,456.15	-£1,275.17	-£1,391.38	-£470.01	£1,005.57	£2,375.52	£3,047.38
Net Profit	**-£53,752**	**-£51,752**	**-£51,752**	**-£51,752**	**-£45,830**	**-£40,350**	**-£32,946**	**-£29,282**	**-£17,072**	**-£9,145**	**-£6,208**	**-£5,436**	**-£5,932**	**-£2,004**	**£4,287**	**£10,127**	**£12,991**

Appendix B

Sep-21	Oct-21	Nov-21	Dec-21	Jan-22	Feb-22	Mar-22	Apr-22	May-22	Jun-22	Jul-22	Aug-22	Sep-22	Oct-22	Nov-22	Dec-22	Jan-23	Feb-23	Mar-23
4350	4570	4800	6400	6780	6990	7230	7450	7540	7850	8100	8230	8390	8540	8760	9510	11300	13400	14000
27500	28900	29500	36400	41000	43200	45300	48200	52300	57000	63900	67400	70300	74300	77900	86700	93102	95600	105670
31850	**33470**	**34300**	**42800**	**47780**	**50190**	**52530**	**55650**	**59840**	**64850**	**72000**	**75630**	**78690**	**82840**	**86660**	**96210**	**104402**	**109000**	**119670**
£32,792.76	£34,460.71	£35,315.28	£44,066.88	£49,194.29	£51,675.62	£54,084.89	£57,297.24	£61,611.26	£66,769.56	£74,131.20	£77,868.65	£82,860.57	£87,230.52	£91,252.98	£101,309.13	£109,935.31	£114,777.00	£126,012.51
£27,466.56	£28,015.89	£28,576.21	£32,000.00	£35,010.00	£35,360.10	£35,713.70	£36,070.84	£36,431.55	£42,300.00	£44,560.00	£45,451.20	£46,360.22	£47,287.43	£48,233.18	£55,000.00	£60,100.00	£61,302.00	£62,528.04
£43,456.50	£45,654.30	£47,952.00	£63,936.00	£67,732.20	£69,830.10	£72,227.70	£74,425.50	£75,324.60	£78,421.50	£80,919.00	£82,217.70	£83,816.10	£85,314.60	£87,512.40	£95,004.90	£112,887.00	£133,866.00	£139,860.00
£8,250.00	£8,670.00	£8,850.00	£10,920.00	£12,300.00	£12,960.00	£13,590.00	£14,460.00	£15,690.00	£17,100.00	£19,170.00	£20,220.00	£21,090.00	£22,290.00	£23,370.00	£26,010.00	£27,930.60	£28,680.00	£31,701.00
£111,966	**£116,801**	**£120,693**	**£150,923**	**£164,236**	**£169,826**	**£175,616**	**£182,254**	**£189,057**	**£204,591**	**£218,780**	**£225,758**	**£234,127**	**£242,123**	**£250,369**	**£277,324**	**£310,853**	**£338,625**	**£360,102**
£93,305	**£97,334**	**£100,578**	**£125,769**	**£136,864**	**£141,522**	**£146,347**	**£151,878**	**£157,548**	**£170,493**	**£182,317**	**£188,131**	**£195,106**	**£201,769**	**£208,640**	**£231,103**	**£259,044**	**£282,188**	**£300,085**
£21,420	£21,420	£21,420	£21,420	£21,420	£21,420	£21,420	£21,848	£21,848	£21,848	£21,848	£21,848	£21,848	£21,848	£21,848	£21,848	£21,848	£21,848	£21,848
£2,785	£2,785	£2,785	£2,785	£2,785	£2,785	£2,785	£2,840	£2,840	£2,840	£2,840	£2,840	£2,840	£2,840	£2,840	£2,840	£2,840	£2,840	£2,840
£1,071	£1,071	£1,071	£1,071	£1,071	£1,071	£1,071	£1,092	£1,092	£1,092	£1,092	£1,092	£1,092	£1,092	£1,092	£1,092	£1,092	£1,092	£1,092
£48	£48	£48	£48	£48	£48	£48	£49	£49	£49	£49	£49	£49	£49	£49	£49	£49	£49	£49
£478	£502	£515	£642	£717	£753	£788	£835	£898	£973	£1,080	£1,134	£1,180	£1,243	£1,300	£1,443	£1,566	£1,635	£1,795
£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0
£4,733	£4,974	£5,097	£6,360	£7,100	£7,458	£7,806	£8,270	£8,892	£9,637	£10,699	£11,239	£11,693	£12,310	£12,878	£14,297	£15,514	£16,197	£17,783
£25,802	**£25,826**	**£25,838**	**£25,966**	**£26,041**	**£26,077**	**£26,112**	**£26,665**	**£26,728**	**£26,803**	**£26,910**	**£26,965**	**£27,011**	**£27,073**	**£27,130**	**£27,274**	**£27,396**	**£27,465**	**£27,625**
£67,503	**£71,508**	**£74,739**	**£99,803**	**£110,823**	**£115,445**	**£120,235**	**£125,213**	**£130,820**	**£143,689**	**£155,406**	**£161,166**	**£168,095**	**£174,696**	**£181,510**	**£203,830**	**£231,648**	**£254,722**	**£272,459**
72%	**73%**	**74%**	**79%**	**81%**	**82%**	**82%**	**82%**	**83%**	**84%**	**85%**	**86%**	**86%**	**87%**	**87%**	**88%**	**89%**	**90%**	**91%**
£408	£408	£408	£408	£408	£408	£408	£416	£416	£416	£416	£416	£416	£416	£416	£416	£416	£416	£416
£204	£204	£204	£204	£204	£204	£204	£208	£208	£208	£208	£208	£208	£208	£208	£208	£208	£208	£208
£153	£153	£153	£153	£153	£153	£153	£156	£156	£156	£156	£156	£156	£156	£156	£156	£156	£156	£156
£26	£26	£26	£26	£26	£26	£26	£26	£26	£26	£26	£26	£26	£26	£26	£26	£26	£26	£26
£510	£510	£510	£510	£510	£510	£510	£520	£520	£520	£520	£520	£520	£520	£520	£520	£520	£520	£520
£0	£0	£0	£0	£0	£0	£0	£6,000	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0	£0
£5,100	£5,100	£5,100	£5,100	£5,100	£5,100	£5,100	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202
£30,965	£31,274	£31,587	£31,903	£32,222	£32,544	£32,870	£33,198	£33,530	£33,866	£34,204	£34,546	£34,892	£35,241	£35,593	£35,949	£36,308	£36,672	£37,038
£4,080	£4,080	£4,080	£4,080	£4,080	£4,080	£4,080	£4,162	£4,162	£4,162	£4,162	£4,162	£4,162	£4,162	£4,162	£4,162	£4,162	£4,162	£4,162
£2,000	£2,000	£2,000	£2,000	£2,000	£2,000	£2,000	£2,040	£2,040	£2,040	£2,040	£2,040	£2,040	£2,040	£2,040	£2,040	£2,040	£2,040	£2,040
£5,100	£5,100	£5,100	£5,100	£5,100	£5,100	£5,100	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202	£5,202
£1,453	£1,453	£1,453	£1,453	£1,453	£1,453	£1,453	£1,482	£1,482	£1,482	£1,482	£1,482	£1,482	£1,482	£1,482	£1,482	£1,482	£1,482	£1,482
£559	£559	£559	£559	£559	£559	£559	£570	£570	£570	£570	£570	£570	£570	£570	£570	£570	£570	£570
£50,558	**£50,867**	**£51,180**	**£51,496**	**£51,815**	**£52,137**	**£52,462**	**£59,183**	**£53,515**	**£53,850**	**£54,189**	**£54,531**	**£54,876**	**£55,225**	**£55,578**	**£55,934**	**£56,293**	**£56,656**	**£57,023**
£16,946	**£20,641**	**£23,560**	**£48,307**	**£59,008**	**£63,308**	**£67,773**	**£66,030**	**£77,305**	**£89,839**	**£101,217**	**£106,635**	**£113,218**	**£119,470**	**£125,932**	**£147,896**	**£175,354**	**£198,066**	**£215,436**
£3,219.67	£3,921.77	£4,476.31	£9,178.39	£11,211.57	£12,028.46	£12,876.78	£12,545.66	£14,687.91	£17,069.42	£19,231.31	£20,260.73	£21,511.51	£22,699.37	£23,927.14	£28,100.25	£33,317.34	£37,632.50	£40,932.86
£13,726	**£16,719**	**£19,083**	**£39,129**	**£47,797**	**£51,279**	**£54,896**	**£53,484**	**£62,617**	**£72,770**	**£81,986**	**£86,375**	**£91,707**	**£96,771**	**£102,005**	**£119,796**	**£142,037**	**£160,433**	**£174,503**

Appendix C

(Timesheet in hours)

Team Members	David	Etienne	Rahul	Rohan	Tom	Will
14/10/19	0	0	0	0	0	1
15/10/19	2	1	1	2	0	0
16/10/19	0	0	1	0	3	2
17/10/19	2	1	2	3	2.5	0
18/10/19	1.5	0	0	5	1	2
19/10/19	0	1	2	1	0	0
20/10/19	1.5	1	0	1	2	0
21/10/19	1.5	2	0	1	2	3
22/10/19	2	1	3	0	2	0
23/10/19	0	0	2	0	0	3
24/10/19	2.5	1	0	1	1	0
25/10/19	1	0	2	5	0	0
26/10/19	2.5	3	1	0	4	1.5
27/10/19	3	2	3.5	1	1	1
28/10/19	1.5	3	2.5	1	1	0
29/10/19	4	1.5	1	1	1	1.5
30/10/19	1.5	0	0	0	0	3
31/10/19	3	4	1.5	0	0	2
1/11/19	0	1.5	1	0	5	1.5
2/11/19	3	4	2	2	0	1
3/11/19	0	3	0	1	3	0
4/11/19	1	1	4	3.5	3	1
5/11/19	1.5	3	3	3	0	0
6/11/19	0	0	3	4	3	3
7/11/19	2	1.5	0	0	4	0
8/11/19	1.5	0	1.5	0	2	2
9/11/19	2	3	1	1	0	2
10/11/19	0	2	0.5	2	0	2
11/11/19	2.5	3.5	2	5	0	0
12/11/19	2	3	2	0	5	4
13/11/19	0	0	1	0	5	3
14/11/19	1.5	4	0	1	0	2
15/11/19	2	0	4.5	4	1	0
16/11/19	2	2.5	3	0	0	3
17/11/19	1	4	2	2	1	5
18/11/19	2	2	4	4	3	2
19/11/19	1	2	3	3	1	3
20/11/19	0	0	1	0	0	3
Total	54.5	61.5	61	57.5	56.5	57.5