Classifying Compound Dose Response Curves

Team Curveball
Sinéad Dunphy, Rohit Duvadie, and Alyson Fay

MADS Capstone Project
April 2023

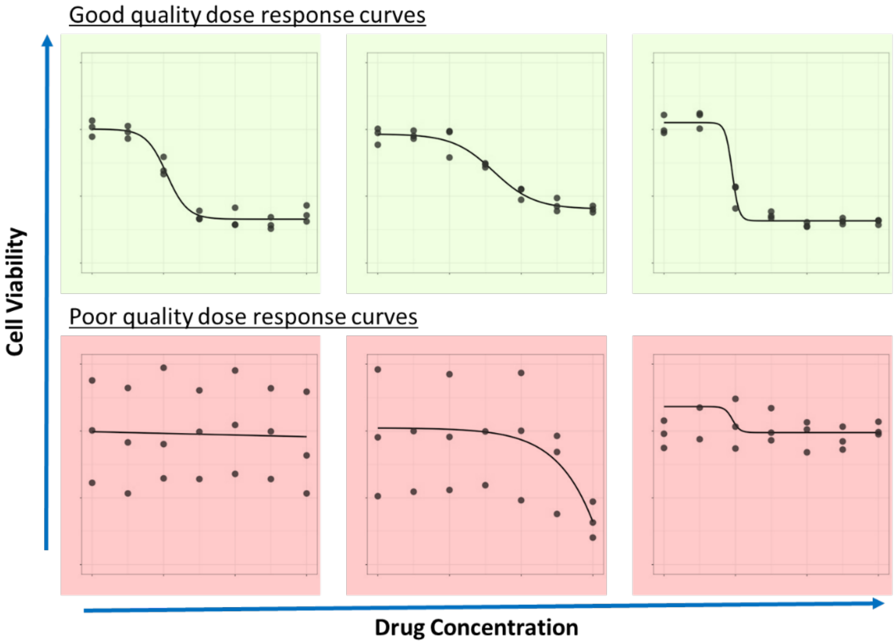
Project Statement

As part of the process of cancer drug discovery, chemical compounds are tested *in vitro* on cancer cell lines at varying dose concentrations to determine if the cell lines show sensitivity to a compound and what dosage of the compound is optimal to inhibit the viability of the cancer cells. The response of a cell line to increasing doses of a compound can be visualized in what is known as a **dose response curve,** by plotting the cell viability readout along with standard error against the compound dose.

Researchers may work to identify promising drug candidates by screening large numbers of compounds against a variety of cell lines and then anaylsing the resulting curves to determine the effect of each compound on each individual cell line, choosing the compounds which produce good quality dose response curves for further investigation. A large-scale screen such as this can result in millions of datapoints and hundreds of thousands curves that need to be evaluated. This can mean examining the metrics of each curve such as maximum effect of the compound (Amax), the point at which there is 50% of the maximum effect (IC50), and the inflection point of the curve, but it also involves visually inspecting the curve images manually to ensure that the computationally drawn curve is done appropriately as well as identify any outlying datapoints. This manual review can be understandably burdensome on the researcher.

The goal of our project is to develop an image-based machine learning algorithm that can classify the quality of dose response curves. In this way, curves with poor quality (e.g. high variability between replicate measures or random fluctuations in response that cannot be modelled easily with a curve) can be easily eliminated from further analysis, allowing scientists to focus future validation and research on those compounds that provide the most useful information. **Figure 1** displays some examples of good and poor quality curve images that are routinely generated.

Figure 1

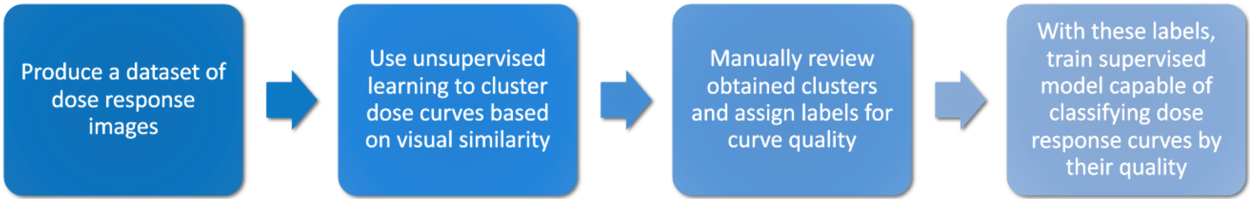


Methodology and Results

Overview

To train our machine learning curve classifier algorithm, we needed a dataset of dose response curve images with labels for their quality. As there is no such existing dataset, we first needed to produce a large set of images and assign labels. To overcome the need to manually assign labels to individual curve images, we used unsupervised learning methods to cluster the curves into visually similar groups and then labelled on a cluster-level. We then moved forward with these labels to train supervised models to classify dose response curves by their quality. The workflow is outlined in **Figure 2**.

Figure 2



Produce a dataset of dose response images

Use unsupervised learning to cluster dose curves based on visual similarity

Manually review obtained clusters and assign labels for curve quality

With these labels, train supervised model capable of classifying dose response curves by their quality

Dataset Description

The Genomics of Drug Sensitivity in Cancer (GDSC) Project is an initiative in the field of cancer drug discovery to characterise potential cancer therapeutics by assessing their effect on a wide range of human cancer cell lines (1). As of July 2022, the GDSC has screened 621 compounds at various concentrations for the effect on the cell viability of 1000 human cancer cell lines of various lineages and has made the data available for research. We used this as the raw data to produce our dose response curve images.

The two datasets available, GDSC1-raw-data and GDSC2-raw-data, were accessed directly from the website (2). Both datasets are extremely large (1GB and 2GB, respectively). They contain the results of drug screening experiments performed by applying various doses of compounds to different cell lines in welled plates (3). Each plate contains a single cell line and the individual wells receive different treatments of compounds at a range of concentrations or are used as controls (e.g. no drug added, blank wells with no cells for background readout, vehicle only for drugs which are diluted in a specific vehicle solution). The combined datasets contain the results of 40252 individual plate experiments. The compounds used target many different pathways important to cell functions to try to trigger the death of the cancer cells.

Data cleaning

The data contained in the datasets is quite complex, with a variety of experimental designs and plate layouts used. Approximately 50% of the experiments were conducted using 384-well plates, 40% using 1536-well plates, and the remaining using 96-well plates. An example of a layout for a 384-well plate is shown in **Figure 3**, taken from Garnett, et. al. (4). A breakdown of the number of individual plates and cell lines by lineage is shown in Figure 4, right. A breakdown of the compounds by pathway targeted and number of cell lines screened is shown in Figure 4, left.

Figure 3

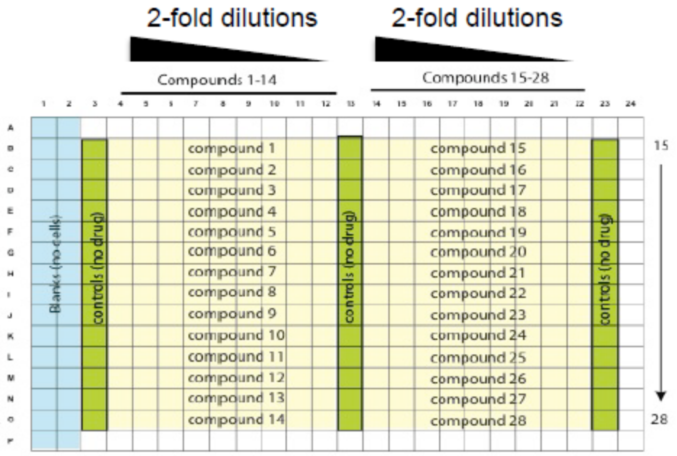
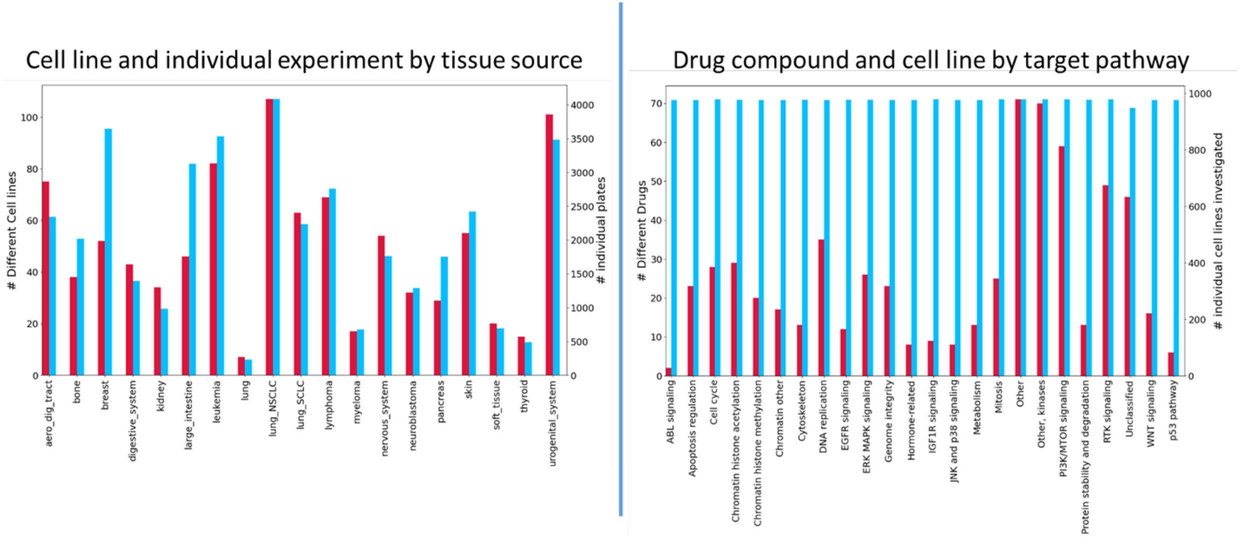


Figure 4



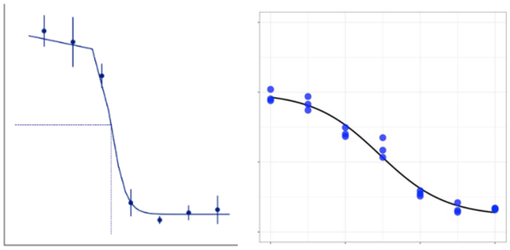
To separate the data needed to plot each individual curve, we used the unique combination of values in the following columns of the dataframe: DRUGSET_ID (plate ID number), DRUG_ID (compound ID number), CELL_ID (cell line ID number), ASSAY (type of viability readout), SEEDING_DENSITY (number of cells plated into each well at the beginning of the experiment), and DURATION (length of the experiment). We ensured that there were at least five concentrations (CONC) present so that a curve could be generated. The INTENSITY column was used as the measure of cell viability. Additionally, we eliminated any data points that were not in triplicate, as we wished to focus on assays with replicates as good quality dose response curves would be expected to have low deviation between repeat datapoints. For those that were in higher replicates than three, we batched them into groups of three so that we could generate the greatest number of curves from the data as possible. Finally, we normalized the INTENSITY values by the untreated (NC-0) control for each plate and cell line.

Curve production

We researched several methods of generating the dose-response curve images in both Python and R. For Python, we first evaluated the Thunor Core library (5). This library produces dose-response data and curve fit parameters using the Pandas library. Unfortunately, the library did not work for the GDSC1 and GDSC2 datasets due to a formatting issue. We then evaluated the simplydrug library available for Python (6). It is a set of open-source based analysis workflows, which contains high throughput screening and generation of dose-response curves. The library uses classical Hill equation, a four parameters sigmoidal function, to generate the curves.

In R, we were able to generate dose-response curves using the dr4pl package (7). This package generates four parameter logistic model curves as well as important metrics like IC50, slope of the curve, and the upper and lower limits of the curve. This package uses ggplot2 to generate images of the curves and individual data points. We collected metrics from the raw data and curves in a table to be used in future supervised learning. These metrics included plate ID, cell line ID, compound ID, curve IC50, curve slope, curve lower and upper limits, minimum and maximum cell viability, convergence on a best fit model, and mean standard deviation of cell viability.

Figure 5



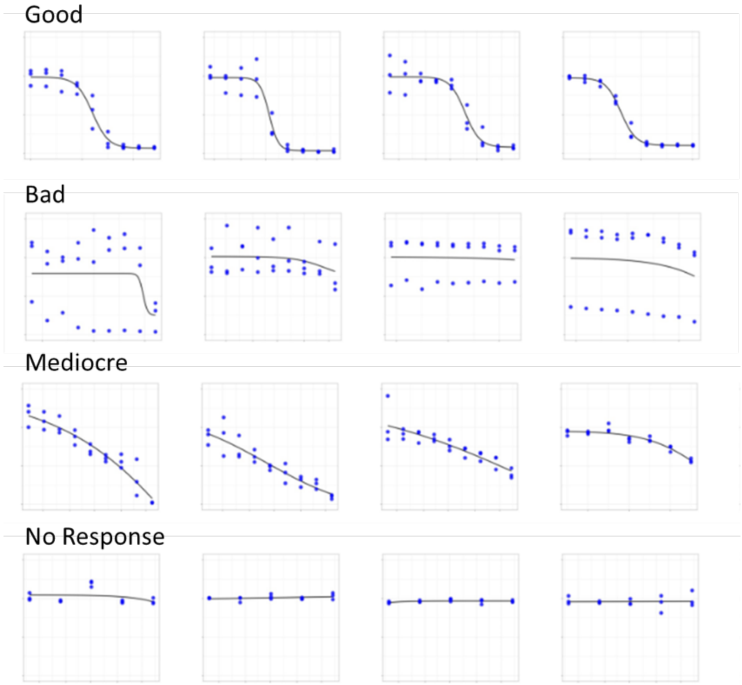
Examples of dose response curves shown in **Figure 5**, with a simplydrug-generated curve on the left and a dr4pl-generated curve on the right. While there are aesthetic differences between the curves, they relay much of the same basic information including the shape of the curve, where the datapoints lie, and the standard deviation of replicates (as shown by either the error bars or the spacing between the replicates). The simplydrug image also includes the placement of the IC50. In the end, we chose to move forward with the dr4pl curves in R because the code ran much faster and we were able to generate a final image dataset of 62,159 curves.

Unsupervised learning clustering

We next used unsupervised methods to cluster the images based on visual similarity as demonstrated in (8). The tensorflow and keras python libraries were used to process each image to extract features. We explored both the VGG16 and VGG19 convolutional neural network models. Following feature extraction of each image, using modules from the scikit-learn library, we applied a PCA to reduce the number of features and then performed K-means clustering to group the images. A range of cluster numbers were investigated and silhouette analysis was used to determine the optimal number of clusters. Matplotlib was used to visualise the silhouette analysis and the clusters via the first two PCA components.

Following a quick proof of concept exploration of unsupervised image clustering on an initial subset of 240 dose curve images, we undertook to perform the clustering on a larger dataset of approximately 11,000 images to investigate if clustering based on curve quality could be achieved and explore roughly how many clusters is required to achieve good separation. We achieved acceptable clusters separating the curves by their quality. It was determined that 40 clusters were required to allow for the variation in the curves and produce the desired clusters. Examples of the clusters achieved can be seen in **Figure 6** below. Upon reviewing the data, we decided to expand our labeling beyond the planned binary “Good” and “Bad”. We added the additional labels of “Mediocre” for curves which weren’t optimum but still have merit for further exploration, and “No Response” for curves which were straight lines signifying that no dose of drug compound impacted cell viability. Both the VGG16 and VGG19 models produced similar results.

Figure 6



Following this exploration of a subset of the data, we next proceeded to run our unsupervised clustering models on the full set of 62,159 curve images. Again, we investigated both VGG16 and VGG19 models. While the results achieved were similar for each model, manual review of the images in each cluster suggested the VGG19 model produced more uniform clusters. We found that 20 components explained the majority of variance in the PCA and that was used going forward (**Appendix Figure 1**). To determine the optimal number of clusters for the K-means clustering, we first look at a k-means inertia plot, also known as an elbow plot (**Appendix Figure 2**). There was no clear delineation indicating what the optimal number of clusters was so we then turned to a silhouette analysis. This is a way of measuring the separation distances between clusters by calculating how close each point is to neighboring clusters (silhouette coefficient). An ideal silhouette plot will show approximately equal thicknesses of the silhouettes, all values above 0, and a peak above the average silhouette score for all points. We determined the optimal number of clusters to be 40 as shown in the left panels of **Figure 7** and **Appendix Figure 3**. In the right panels, the first and second PCA components are plotted and colored by each cluster. Once the clusters had been obtained, the quality labels were assigned on a cluster-level for each image.

Figure 7

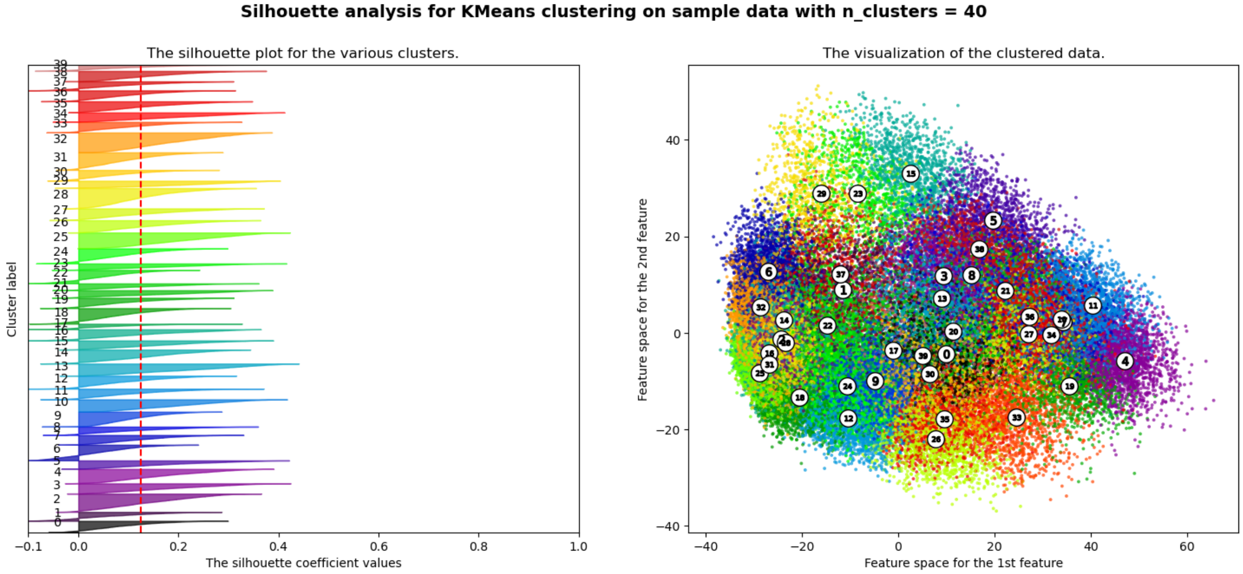


Image labeling

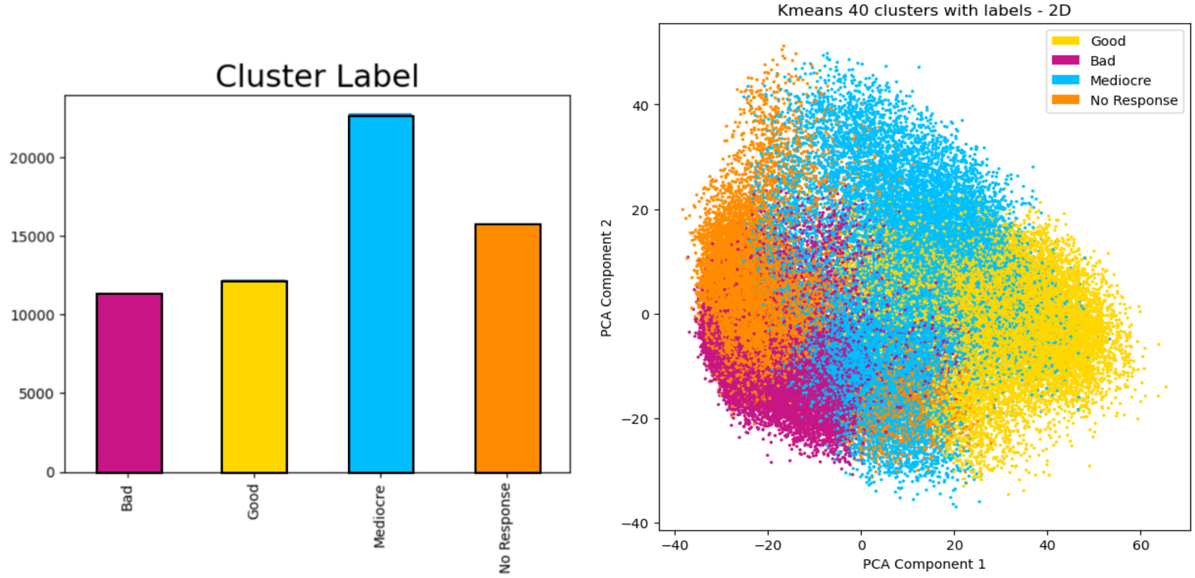
Once the clusters were defined, we randomly selected 100 images from each cluster to ensure uniformity of images in said cluster. We then labeled the cluster based on the quality of the curves using the labels “Good”, “Bad”, “Mediocre”, and “No Response”. The labels were then mapped back to the individual curves using the metrics table that was generated alongside the images. These metrics could then be used as features for supervised learning with the labels as output. As shown in the left panel of **Figure 8**, the most number of images were labeled as “Mediocre” and there was roughly the same amount of “Good” and “Bad” labels. Those labels were used to plot the first and second components of the PCA in the right panel of **Figure 8**. There are clear clusters present, with the “Good” and “Bad” clusters separated from each other, the “Mediocre” cluster in the middle of the plot, and the “No Response” cluster overlapped with the “Bad” cluster.

Supervised learning

We next moved on to generate a supervised learning model using the labels of each image derived from the cluster. One of the reasons that we wanted to use a supervised model using metrics from the raw data and curves is to potentially give researchers a simple, easily understood model to classify curves. Although our unsupervised clustering did well at grouping like images, it is not able to be used to describe the characteristics of the data that led to that grouping. However, a supervised model using only the metrics generated and not the actual images could highlight which features (metrics) are the most important for good quality curves.

The data from image-generated metrics produced thirteen features, which included cell line ID, compound ID, IC50, and mean standard deviation of cell viability. We normalized the IC50, which contained several extremely high and low values, to have a limit on the minimum and maximum of that data as these skewed values would have impacted the model's performance. Another feature, convergence, had Boolean values that we converted to numerical values. The final data for models contained 62159 rows. We generated training and test sets from the dataset with a 70/30 split and stratified by our label to ensure the training and test sets mirror the label distribution found in the full dataset. For select models with the best initial performance, we performed model tuning and preliminary evaluation through a grid search with 5-fold cross validation over a set of key hyperparameters.

Figure 8



We evaluated several supervised machine learning models. These models include decision tree (DT), random forest (RF), gradient boosting (XGB), linear support vector machine (LSVM), and k-nearest neighbors (KNN) algorithms. A DT consists of a series of nodes, which includes a root node, internal nodes and leaf nodes. The root node and internal nodes each contain conditional statements regarding the features used for classification. RF is a classifier which builds upon decision trees to enhance classification performance. SVM creates a decision boundary which maximizes the minimum distance from the boundary point to the nearest example of each class. XGB is a method of transforming weak learners into strong learners. KNN is a non-parametric classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. In addition, we also utilized ensemble methods. The ensemble methods are machine learning techniques that use several base models to produce one optimal predictive model and the ability to avoid overfitting. The ensemble methods included Extra Tree, AdaBoost, Voting, and Stacking classifiers.

Additionally, we applied feature selection methods to provide most significant features from our dataset and enhance the performance of the Logistic Regression (LR) classifier. LR uses a logistic sigmoid hypothesis function in combination with a cost function and optimization technique to output probability values which can be mapped to two or more discrete classes. We also implemented the Recursive Feature Elimination (RFE) method, which is a type of wrapper feature selection method. It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute.

Table 1 summarizes the performance of the supervised models we compared. Balanced accuracy is an accuracy score that takes into account label imbalance. The balanced accuracy scores ranged from 0.86

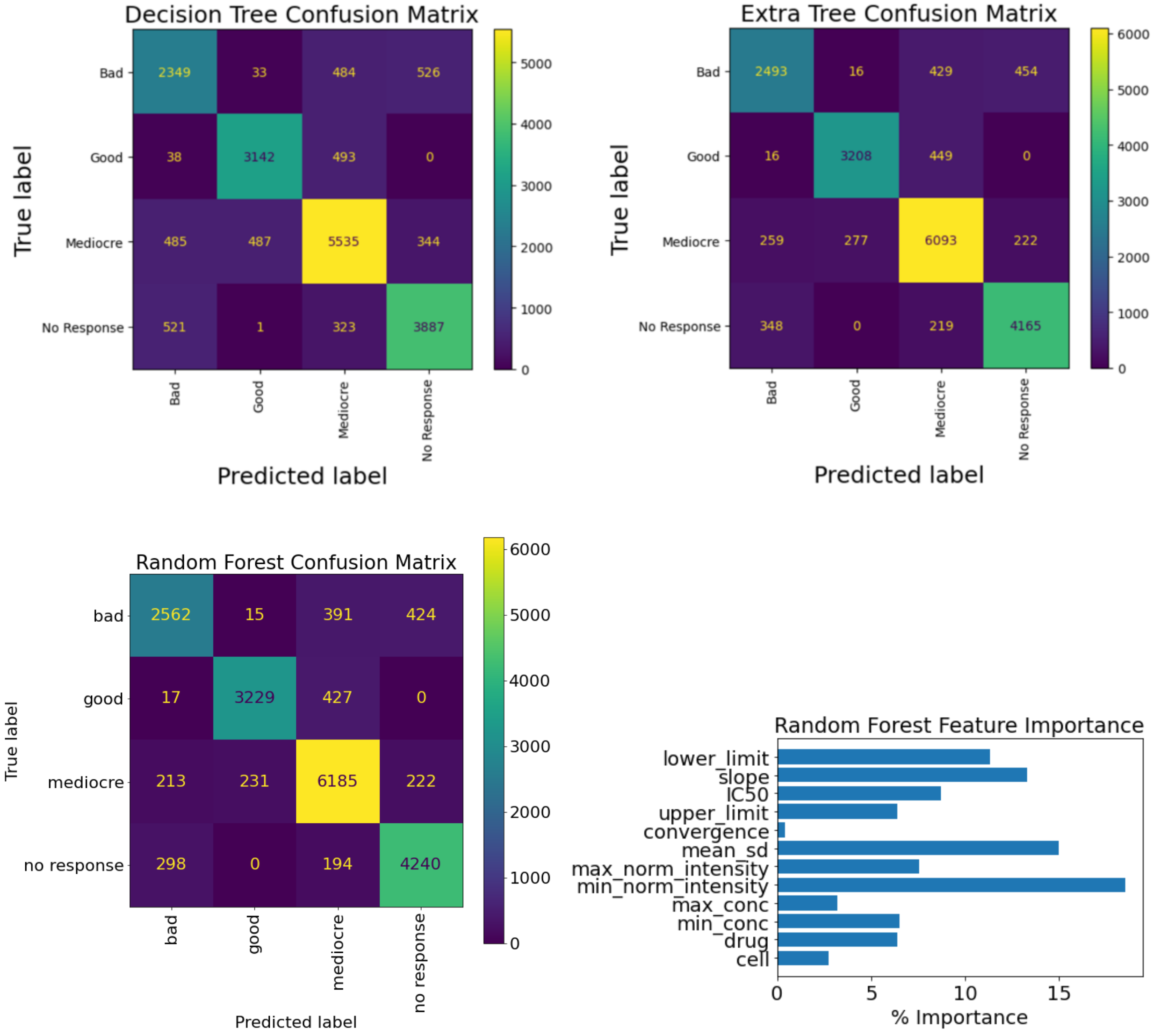
(RF) to 0.79 (AdaBoost). F1 score is another way of measuring a model's performance and is a balance of precision and recall. The F1 scores ranged from 0.87 (RF) to 0.80 (DT).

Table 1

Model	Parameters	Accuracy	Balanced Accuracy	Weighted Precision	Weighted Recall	Weighted F1 Score
Random Forest	n_estimators=100, max_depth=50, min_samples_leaf=1, min_samples_split=5	0.8693	0.8579	0.8696	0.8693	0.8687
Decision Tree	default	0.8005	0.7901	0.8081	8005	0.8008
Voting	estimators=dict_items([('dt0', DecisionTreeClassifier(max_depth=3, random_state=1000)), ('dt1', DecisionTreeClassifier(min_samples_leaf=7, random_state=1000)), ('dt2', DecisionTreeClassifier(class_weight='balanced', random_state=1000))], n_jobs=-1)	0.8258	0.8146	0.8272	0.8258	0.8259
Stacking	estimators=clfs.items(), final_estimator=LogisticRegression(max_iter=1000, random_state=1000), cv=5, n_jobs=-1	0.8379	0.8259	0.8377	0.8379	0.8371
Extra Tree	Bootstrap = 'False', Criterion = 'gini', max_depth = 'None', max_features = 10, min_samples_leaf = 1, min_samples_split = 3, n_estimators = 200	0.8551	0.8433	0.8551	0.8551	0.8544
AdaBoost	base_estimator__criterion = 'gini', base_estimator__splitter = 'best', algorithm = 'SAMME', n_estimators = 250, learning_rate = 0.5	0.8045	0.7890	0.8049	0.8045	0.8030

Confusion matrices produced by the DT, ET, and RF models are shown in **Figure 9**. This is a clear way of visualizing their performances by highlighting the similarities and differences between the actual and predicted labels. The results showed these classifiers were performing well. Of particular importance, very few "good" curves were misclassified as "bad" or "no response", meaning researchers using this model to focus on only "good" curves could have confidence in the results. By examining the confusion matrix along with the evaluation metrics in Table 1, we concluded that the RF is the best performing model. To further our understanding of the RF model, we performed feature extraction to see which features were the most important (**Figure 9**, bottom left). The top three features were minimum normalized intensity, mean standard deviation of the normalized intensity, and the slope of the curve.

Figure 9



Discussion

In this project we have successfully created a machine learning model capable of classifying dose response curves by their quality. This achievement will be useful to researchers who need to review a high volume of such curves following a large-scale assay of potential drug candidates. Through grading the curves by quality, manual review of each individual curve by researchers is no longer required, freeing up their valuable time and allowing efforts to be focused on the more promising chemical compounds.

While this project focused solely on dose response curves in *in vitro* cancer cells, the concepts and processes used could be expanded to numerous additional scenarios. For example, the dose response of bacterial or viral cells to potential anti-bacterial/anti-viral compounds could be investigated in a similar manner. The results of a large-scale screen could be made more manageable by implementing a model capable of classifying the dose response curves and excluding those where no response was observed or the data is of poor quality. Beyond dose response curves, if a researcher was interested in exploring factors to optimize growth of cells in culture, they could run a range of experiments and plot cell growth over time. While this is the inverse of a dose response curve where the aim is to reduce cell viability, the same process of labelling clusters of cell growth images for their quality and training a model to classify curves can be implemented, thus eliminating time-consuming manual review. To note, while this current project utilised data from established cell lines, should the processes herein be adapted to explore data from primary patient samples, ethical considerations for informed consent of the patient and ensuring the patient population used is unbiased and representative of the true population would need to be addressed.

In conclusion, we have achieved our goal of developing a dose response curve classifier and have produced a model which also has the potential to be adapted to broader settings to reduce burdensome manual review of curve images.

Statement of Work

Contributor	Areas of Contribution
Sinéad	Exploration and cleaning of GDSC datasets, VGG model set-up, PCA, K-means clustering, visualisations, report preparation and presentation preparation and recording
Alyson	Exploration and cleaning of GDSC datasets, image generation in R, silhouette analysis, Random Forest, GitHub set-up, report and presentation preparation
Rohit	Exploration and cleaning of GDSC datasets, image generation in Python, Decision Tree, Voting, Stacking, Extra Tree, AdaBoost, report and presentation preparation

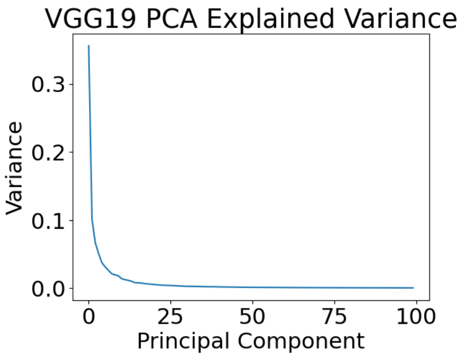
Bibliography

1. *Genomics of Drug Sensitivity in Cancer.* Wellcome Sanger Institute and Massachusetts General Hospital. https://www.cancerrxgene.org/
2. *Genomics of Drug Sensitivity in Cancer, bulk data download.* Wellcome Sanger Institute and Massachusetts General Hospital. https://www.cancerrxgene.org/downloads/bulk_download
3. Lorio F, Knijnenburg TA, Vis DJ, Bignell GR, et al. A Landscape of Pharmacogenomic Interactions in Cancer. Cell. 2016; 166(3):740-754. Published 2016 Jul 7. doi: 10.1016/j.cell.2016.06.017
4. Garnett MJ, Edelman EJ, Heidorn SJ, et al. Systematic identification of genomic markers of drug sensitivity in cancer cells. Nature. 2012;483(7391):570-575. Published 2012 Mar 28. doi:10.1038/nature11005
5. Lubbock AL, Harris LA Harris, Quaranta V, et al. Thunor: visualization and analysis of high-throughput dose–response datasets. Nucleic Acids Research. 2021;49(W1): W633-W640. Published 2021 Jul 2. doi.org/10.1093/nar/gkab424

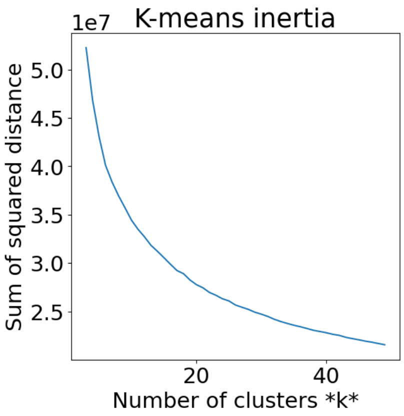
6. *simplydrug GitHub Repository*. Blavatnik Center for Drug Discovery. https://github.com/disc04/simplydrug. Accessed 10 Feb 2023.
7. *dr4pl Bitbucket Repository.* Dittmer Lab. https://bitbucket.org/dittmerlab/dr4pl/src/master/ Accessed March 2023.
8. Using Keras' Pre-trained Models for Feature Extraction in Image Clustering. franky. *Medium*. https://franky07724-57962.medium.com/using-keras-pre-trained-models-for-feature-extraction-in-image-clustering-a142c6cdf5b1

Appendix

Appendix Figure 1

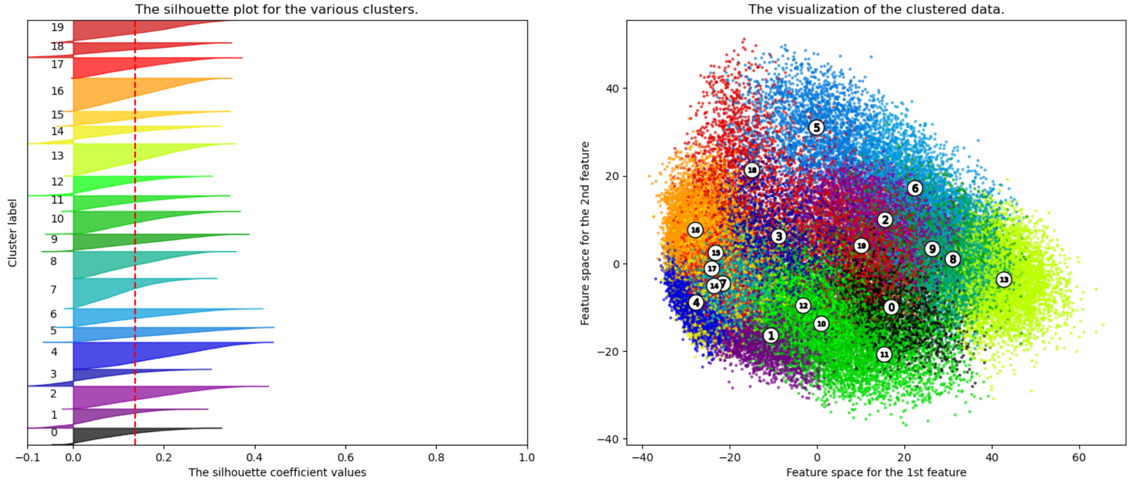


Appendix Figure 2

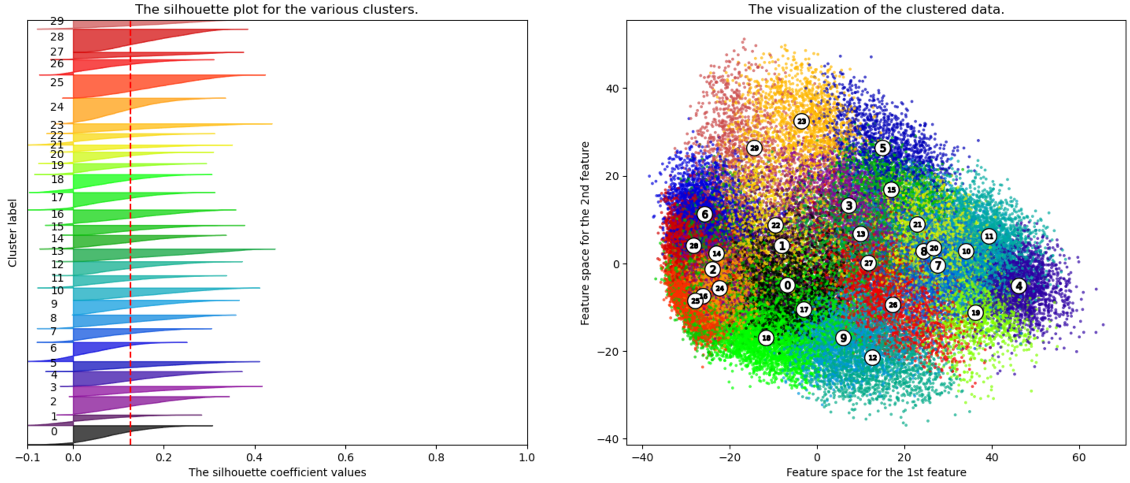


Appendix Figure 3

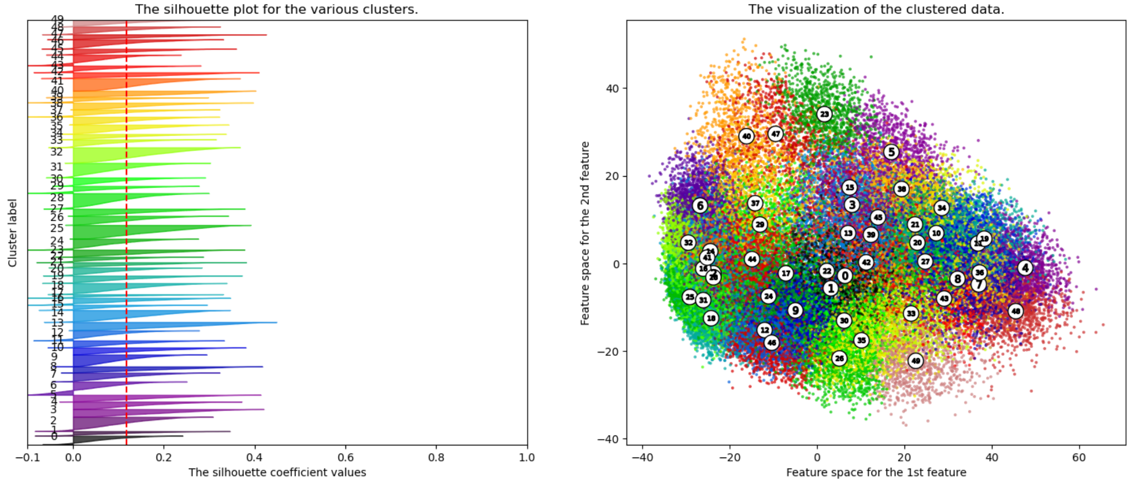
Silhouette analysis for KMeans clustering on sample data with n_clusters = 20



Silhouette analysis for KMeans clustering on sample data with n_clusters = 30



Silhouette analysis for KMeans clustering on sample data with n_clusters = 50


The silhouette plot for the various clusters.
Cluster label
The silhouette coefficient values
The visualization of the clustered data.
Feature space for the 2nd feature
Feature space for the 1st feature