Predicting mechanisms of action for new drugs

Fabio A Oliveira

11/26/2020

Contents

Introduction

This report describes the construction of a statistical model that predicts the mechanism of action of a drug based on numeric cellular data. The problem has been proposed by the Laboratory of Innovation Science at Harvard (LISH) and is currently the subject of an ongoing machine learning competition hosted at ***kaggle.com***.

Traditionally, drugs were developed and put into use before the actual biological mechanisms through which they acted were completely understood. Today, technology has made it possible to identify proteins associated with a particular disease and to develop drugs that modulate those proteins. A ***mechanism of action*** *(MoA)* is a label attributed to an agent to describe its biological activity. By being able to properly identify a molecule's *MoA*, it can subsequently be used in a targeted manner to obtain a desired cell response.

This project is part of an effort in which cellular data acquired from samples treated with drugs with known *MoAs* is used to construct models that will be able to identify the *MoAs* for new drugs.

Dataset

The full dataset has been divided by the hosts of the competition into a `train` and a `test` sets. Predictors are provided for both sets, but the actual outcomes are only provided for samples in the `train` set.

A number of files have been made available and are used in this project. They are briefly described below:

- `train_features.csv` and `test_features.csv`: features for the `train` and `test` sets. Each observation corresponds to complete cellular data from an unique sample and is identified with an id in variable `sig_id`. Variables `cp_type`, `cp_time` and `cp_dose` indicate whether the sample has been treated with an actual compound or a control perturbation, the time of exposure to the compound and the dosage, respectivelly. Additionally, there are 772 variables indicating gene expression levels (`g-0` through `g-771`) and 100 variables with cell viability data (variables `c-0` through `c-99`);

- `train_targets_scored.csv`: outcomes for the `train` set. Each observation contains a `sig_id` matching that in the `train_features.csv` file, along with 206 binary variables, each corresponding to a particular *MoA*. These binary outcomes indicate wheter each mechanism is in play for the particular sample. The mechanisms are not mutually exclusive, so many samples have multiple mechanisms with a target of 1;

- `train_targets_nonscored.csv`: nonscored outcomes for the `train` set. These have the exact same structure as the scored targets, but with a different set of mechanisms that will not be used for calculating the final competition scores;

- `sample_submission.csv`: a sample file demonstrating the expected format for the submission. Submitted predictions are expected to contain observations for each of the samples in the `test` set. There must be a column for each of the scored mechanisms, containing the predicted probability that the mechanism is in effect in that sample.

Additionally, a `train_drug.csv` file has been made available after the start of the competition, containing an anonymous `drug_id` for each sample in the `train` set. Since this file has only recently been release, when this report was nearly finished, it has not been used in this project.

Overview of the project

The goal of this project is to construct a model that will predict the probability of each scored *MoA* for each observation in the `test` set. In order to do so, the following steps are performed:

1. The data in the `train` set is split into the `trainSingle`, `trainEnsemble` and `validate` partitions;

2. A selection of machine learning models is trained on the `trainSingle` set and used to make predictions for the entire dataset;

3. Using the predictions from the previous models, an ensemble model is trained on the `trainEnsemble` set and used to make predictions for the entire dataset;

4. The `validate` partition is used to report performance metrics for all models;

5. Predictions on the `test` set are saved to a submission file, but no performance metric is reported since there is no public file with the expected outcomes for the entries in this partition.

Each of these steps is explained in more details in the corresponding section of this report.

The metric set by the hosts of the competition is the ***log loss***, which is given by the following expression:

$$logloss = -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{N}\sum_{i=1}^{N}[y_{i,m}log(\hat{y}_{i,m}) + (1 - y_{i,m})log(1 - \hat{y}_{i,m})]$$

where:
N is the number of `sig_id` observations in the test data ($i = 1, ..., N$)
M is the number of scored *MoA* targets ($m = 1, ..., M$)
$y_{i,m}$ is the predicted probability of a positive *MoA* response for a `sig_id`
$y_{i,m}$ is the ground truth, 1 for a positive response, 0 otherwise

The nature of the proposed problem imposes some limitations on the choice of machine learning models that can be applied to the task. Because the chosen metric evaluates the predicted probabilities rather than the predicted outcomes, the machine learning models applied here need to be selected among those capable of calculating class probabilities. Additionally, given that the targets are not mutually exclusive, this problem is in fact categorized as a ***multi-label classification task***. Most classification algorithms are constructed so that class probabilities sum to 1, so a series of adaptations are necessary to use them in this scenario. Finally, since this is the capstone project for an online course, techniques that where presented during the course (or that are closely related to those) where preferred.

The *R* code for this project is divided into two different files. All of the data preparation and the modeling are done in the `HX9_CYO_Main.R` script. This script prepares the data and does all of the calculations necessary for the development of the models, calculates all the predictions, and outputs the submission file. It then saves the resulting objects into files that are loaded by the `HX9_CYO_Report.RMD` script (the *R Markdown* script that generates this report) to create all the tables and graphics in this report.

Relevant links

The coding competition that inspired this project is hosted at https://www.kaggle.com/c/lish-MoA.

The GitHub page for this project is https://github.com/fabio-a-oliveira/mechanisms-of-action.

This analysis is part of the capstone project for the Data Science Professional Certificate offered by HarvardX and hosted on edX. More information can be found at https://www.edx.org/professional-certificate/harvardx-data-science.

Methods & analyses

Before attempting to construct models to predict the mechanisms in action for each sample in the dataset, steps were taken to load the data, organize it into *R* objects appropriate for analysis and perform a thorough exploratory data analysys to gain valuable insights.

Then, a two-step approach was taken to build the statistical model:

1. A series of models were constructed using the `trainSingle` partition. During this step, the provided features were used to make predictions for the probability of a positive outcome for each of the *MoAs*;
2. A ensemble model was constructed using the models developed during the previous step. At this stage, the predictions made by each model on the `trainEnsemble` set were used as features for the construction of the ensemble model.

Data Preparation

This section addresses the steps taken in preparing the data for analysis and modeling.

Load data

The full set of files used for this project is provided by *kaggle* as a *.zip* file, containing all the data in *.csv* format. Unfortunately, *kaggle* does not provide a way to download datasets programmatically using *R* (there is an API for *Python* though). In order to meet the capstone project requirement that all the data must either be automatically downloaded from the *R* script or provided on a GitHub repository, the files were unzipped and uploaded to GitHub.

However, GitHub imposes a 10MB file size limit on their free accounts, so the *.csv* files where split into smaller ones, each containing a chunk of the original data. The resulting predictions made by each model are also loaded to GitHub in the same manner.

Upon navigating the repository online or cloning it to a machine, one will be able to identify all the files in the */files* folder. These are loaded by the *HX9_CYO_Main.R* script and used to perform all the analyses.

Partition data

As discussed previously, the full set of observations had already been split by the sponsors of the competition into a `train` and a `test` sets. Outcomes were only provided for the `train` set, so further splits where necessary in order to allow for two stages of modeling and for performance measurements.

For the purposes of this project, the original `train` set has been split into three partitions:

1. The `trainSingle` partition is used for the first stage of modeling and contains 80% of the samples in the `train` set;

2. The `trainEnsemble` partition is used for the second stage of modeling (an ensemble of the models in the first stage) and contains 10% of the samples in the `train` set;
3. The `validate` partition is only used for measuring the final performance of the full model and contains 10% of the samples in the `train` set.

Additionally, some code is included int the *HX9_CYO_Main.R* script to ensure that the `trainSingle` partition contains at least one positive sample for each of the *MoAs* and that the `trainEnsemble` contains at least one positive sample for the *MoAs* present in the `validate` set.

Auxiliary variables

Before proceding with the analysis, the data has been organized into a set of *R* variables so as to facilitate manipulation.

- The `train` object was created as a list of objects containing the full set of features, scored and non-scored targets on the `train` set. It also contained a PCA object with results from a principal component analysis on the features in the `trainSingle` partition, as well as a data frame containing coordinates of the centroids of each of the mechanisms in the feature space;

- The `test` object was created as a list with the data frame of features in the `test` partition;

- The `partitions` object was created as a list of character vectors, with each list containing the `sig_id` unique identifier for the samples in the appropriate partition. Apart from the `trainSingle`, `trainEnsemble`, `validate` and `test` partitions, character vectors were also included with the control samples (which have no mechanism by definition) and with the train samples without any mechanisms;

- The `predictions` object was created as a data frame in tidy format, with one observation for each combination of samples and mechanisms. During the modeling phase, predictions made by each of the different models were added as new variables to this data frame.

- The `mechanisms` object was created as a data frame with relevant information pertaining to each of the individual mechanisms. For each mechanism, a corresponding entry with its name, whether it would be scored or not, and its prevalence (proportion of positive outcomes) in the `trainSingle` partition.

Principal components analysis

Due to the large number of predictors (875 in total), a principal component analysis *(PCA)* was performed as a dimensionality reduction technique.

Normally, the `prcomp()` function in *R* would be applied to the full set of predictors, generating a rotation to a coordinate system with uncorrelated predictors each accounting for a decreasing proportion of the total variability. However, this technique has a limitation in the fact that it is *unsupervised*: the resulting predictors are not dependent on the actual outcome the models will attempt to predict. This means that, even though predictors are ordered according to their total variance, they are not necessarily relevant in distinguishing between different classes.

Thus, an adaptation to the principal component analysis technique was applied. First, the samples in the `trainSingle` partition were grouped according to the *MoAs* for which they had a positive outcome. This information was then used to calculate the centroids for each of the mechanisms. Finally, the matrix containing the coordinates (in the feature space) for each of the centroids went through a *PCA* with the `prcomp()` function.

This amounts to a rotation to a coordinate system corresponding to the hyperplane connecting each class' centroid. In the case of uncorrelated predictors, this rotated coordinate system is optimal for a classification task.

The following figure illustrates this approach for a 2-class scenario:

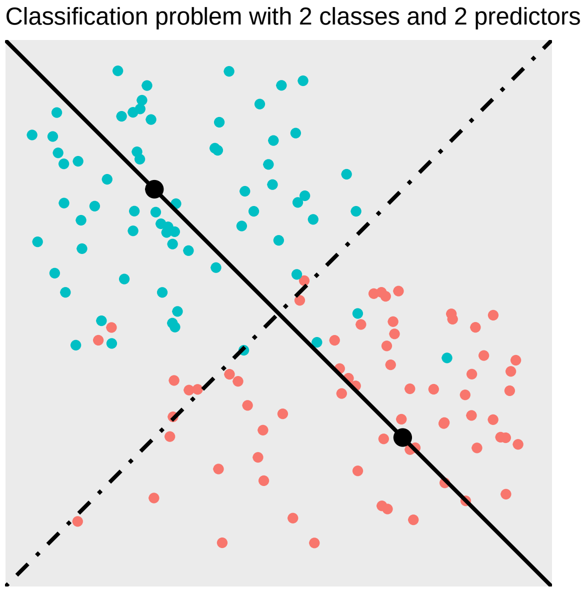

FIGURE: *Observations for two classes plotted as points with different colors. Black points show their respective centroids. The solid line connects the centroids and defines the 1-dimension hyperplane along which the separation is optimal (for uncorrelated predictors). It corresponds to the direction of the first principal component. The distance between each observation and the solid line is irrelevant for classification and corresponds to the second principal component. The dashed line shows the optimal separating line.*

Exploratory Data Analysis

In this section, we describe the dataset which is the subject of the project. We first glimpse at `train` and `test` sets, in order to understand their relative sizes, amount of samples and variables. Then, we look more closely to the outcomes which we attempt at predicting, and finally at the predictors available for the task.

Overview of the dataset

We begin by looking at the structure of the first 10 columns of the scored outcomes object:

```
## Rows: 23,814
## Columns: 10
## $ sig_id                           <chr> "id_000644bb2", "id_000779bfc", "...
## $ `5-alpha_reductase_inhibitor`     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ `11-beta-hsd1_inhibitor`          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ acat_inhibitor                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ acetylcholine_receptor_agonist   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ acetylcholine_receptor_antagonist <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ acetylcholinesterase_inhibitor   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ adenosine_receptor_agonist       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ adenosine_receptor_antagonist    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ adenylyl_cyclase_activator       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

We see that this object is comprised of the `sig_id` column, representing an unique identifier for each of the 23,814 samples in the `train` set. Additional columns are present for each of the scored mechanisms. Active mechanisms for a given sample are encoded with a "1", while others are encoded with "0".

The non-scored mechanisms are arranged similarly, with the `sig_id` column followed by binary variables indicating the active *MoA*.

```
## Rows: 23,814
## Columns: 10
## $ sig_id                             <chr> "id_000644bb2", "id_000779bfc",...
## $ abc_transporter_expression_enhancer <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ abl_inhibitor                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ ace_inhibitor                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ acetylcholine_release_enhancer     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ adenosine_deaminase_inhibitor      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ adenosine_kinase_inhibitor         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ adenylyl_cyclase_inhibitor         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ age_inhibitor                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ alcohol_dehydrogenase_inhibitor    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
```

Then, we look at features available for prediction. There are several hundred columns with the different predictors, so we start by looking at the names of the first few:

```
##  [1] "sig_id"  "cp_type" "cp_time" "cp_dose" "g-0"     "g-1"     "g-2"
##  [8] "g-3"     "g-4"     "g-5"     "g-6"     "g-7"     "g-8"     "g-9"
## [15] "g-10"    "g-11"    "g-12"    "g-13"    "g-14"    "g-15"    "g-16"
## [22] "g-17"    "g-18"    "g-19"    "g-20"    "g-21"    "g-22"    "g-23"
##  [ reached getOption("max.print") -- omitted 1055 entries ]
```

Similarly to the outcomes, there is a `sig_id` column with an unique identifier for each sample, followed by the remainder of the predictors. Below we see a glimpse at the first four columns:

```
## Rows: 23,814
## Columns: 4
## $ sig_id  <chr> "id_000644bb2", "id_000779bfc", "id_000a6266a", "id_0015fd3...
## $ cp_type <chr> "trt_cp", "trt_cp", "trt_cp", "trt_cp", "trt_cp", "trt_cp",...
## $ cp_time <dbl> 24, 72, 48, 48, 72, 24, 24, 48, 48, 48, 72, 48, 48, 48, 72,...
## $ cp_dose <chr> "D1", "D1", "D1", "D1", "D2", "D1", "D2", "D1", "D1", "D2",...
```

We can see that the `train` set contains 23,814 observations, each characterized by an unique identifier in column `sig_id`.

Below we see a summary of the `cp_type`, `cp_time`, and `cp_dose` variables:

```
##          cp_type      cp_time   cp_dose
##  ctl_vehicle: 1866   24:7772   D1:12147
##  trt_cp     :21948   48:8250   D2:11667
##                      72:7792
```

We notice that `cp_type` has two levels: `ctl_vehicle`, which corresponds to samples where no agent has been applied and that by definition have no active *MoA*; and `trt_cp`, corresponding to samples with an agent that may have an active *MoA*. Additionally, `cp_time` and `cp_dose` indicate the time of exposure (24, 48 or 72 hours) and the dosage (D1 or D2).

The remaining 872 predictors account for the majority of the features and are comprised of expression levels measured for a selection of 772 genes (variables `g-0` through `g-771`) and 100 variables related to cell viability (variables `c-0` through `c-99`).

On the `test` set, we see the same variables for in a total of 3982 samples. The structure of the first four columns is shown below:

```
## Rows: 3,982
## Columns: 4
## $ sig_id  <chr> "id_0004d9e33", "id_001897cda", "id_002429b5b", "id_00276f2...
## $ cp_type <chr> "trt_cp", "trt_cp", "ctl_vehicle", "trt_cp", "trt_cp", "ctl...
## $ cp_time <dbl> 24, 72, 24, 24, 48, 24, 48, 72, 48, 24, 24, 48, 24, 72, 24,...
## $ cp_dose <chr> "D1", "D1", "D1", "D2", "D1", "D1", "D2", "D2", "D2", "D2",...
```

We proceed to explore the composition of the `train` and `test` sets and the relations between the different variables in more detail.

Outcomes

As described above, the dataset contains the ground truth for the `train` set, with a total of 608 different mechanisms of action. Out of this total, 206 were selected by the hosts of the competition as scored mechanisms, for which the performance metrics will be calculated. The remaining 402 mechanisms may be used in the models, but will not count for performance calculation and should not be part of the submission.

The following table contains totals for the scored and nonscored mechanisms, as seen in the `train` set. We see that the scored mechanisms correspond to roughly 1/3 of the total (206 from a total of 408), but they are more frequent on average.

Table 1: distribution of mechanisms in scored and nonscored sets

	source	number of mechanisms	mean prevalence
1	scored	206	0.003434
2	nonscored	402	0.000523

The following table shows their distribution with some additional details.

Table 2: basic statistics on outcomes

	metric	value
1	mean positive mechanisms	0.918
2	mean scored mechanisms	0.707
3	mean nonscored mechanisms	0.210
4	most mechanisms in single sample	11.000
5	most scored mechanisms in single sample	7.000
6	proportion of samples with at least one positive mechanism	0.768
7	proportion of samples with no mechanisms	0.232

Wee see that many of the available samples have several positive outcomes for different mechanisms, with one sample in particular showing as many as 11 positive targets. Meanwhile, there is a significant proportion of samples without any active mechanisms. We can see the distribution of the number of different mechanisms per sample in the tables below, when considering all the mechanisms and only the scored mechanisms:

Table 3: number of positive targets per sample

	number of mechanisms	scored & nonscored	scored only
1	0	5530	9367
2	1	15636	12532
3	2	2057	1538
4	3	450	303
5	4	74	55
6	5	18	13
7	6	19	0
8	7	18	6
9	10	6	0
10	11	6	0

Many samples do not have a known active mechanism of action, and the majority have one. However, a significant portion of the samples have multiple active mechanisms, which indicates that the task of predicting the *MoA* is not a simple multi-class classification task, but a *multi-label classification task* instead. To illustrate the difference, we look at the set of all outcomes present in the `train` set (considering only the scored mechanisms for simplicity).

Table 4: combinations of MoAs found in train set with 4 or more single mechanisms

	outcome	instances
1	apoptosis_stimulant | bcl_inhibitor | ikk_inhibitor | nfkb_inhibitor | nitric_oxide_production_inhibitor | nrf2_activator | ppar_receptor_agonist	6
2	bcr-abl_inhibitor | kit_inhibitor | pdgfr_inhibitor | src_inhibitor | tyrosine_kinase_inhibitor	6
3	fgfr_inhibitor | kit_inhibitor | pdgfr_inhibitor | raf_inhibitor | vegfr_inhibitor	1
4	flt3_inhibitor | kit_inhibitor | pdgfr_inhibitor | raf_inhibitor | vegfr_inhibitor	6
5	11-beta-hsd1_inhibitor | acetylcholinesterase_inhibitor | atpase_inhibitor | nfkb_inhibitor	6
6	adrenergic_receptor_antagonist | norepinephrine_reuptake_inhibitor | serotonin_receptor_antagonist | serotonin_reuptake_inhibitor	6
7	akt_inhibitor | fgfr_inhibitor | mtor_inhibitor | vegfr_inhibitor	6
8	apoptosis_stimulant | caspase_activator | hiv_inhibitor | topoisomerase_inhibitor	12
9	aurora_kinase_inhibitor | bcr-abl_inhibitor | flt3_inhibitor | jak_inhibitor	7
10	dna_alkylating_agent | dna_inhibitor | estrogen_receptor_agonist | tubulin_inhibitor	6
11	fgfr_inhibitor | kit_inhibitor | pdgfr_inhibitor | vegfr_inhibitor	6
12	flt3_inhibitor | kit_inhibitor | pdgfr_inhibitor | vegfr_inhibitor	6

We now turn our attention to the individual mechanisms. The table below shows the top 10 most frequent mechanisms, with their number of positive outcomes and the prevalence (proportion of positive outcomes over total samples). We notice that all of these most frequent *MoAs* are part of the scored set, for which performance will be evaluated.

Table 5: mechanisms arranged by prevalence on train set

	mechanism	source	positives	prevalence
1	nfkb_inhibitor	scored	832	0.035
2	proteasome_inhibitor	scored	726	0.030
3	cyclooxygenase_inhibitor	scored	435	0.018

	mechanism	source	positives	prevalence
4	dopamine_receptor_antagonist	scored	424	0.018
5	serotonin_receptor_antagonist	scored	404	0.017
6	dna_inhibitor	scored	402	0.017
7	glutamate_receptor_antagonist	scored	367	0.015
8	adrenergic_receptor_antagonist	scored	360	0.015
9	cdk_inhibitor	scored	340	0.014
10	egfr_inhibitor	scored	336	0.014

The figure below shows the prevalence in more details, with the scored and nonscored mechanisms in separate plots.

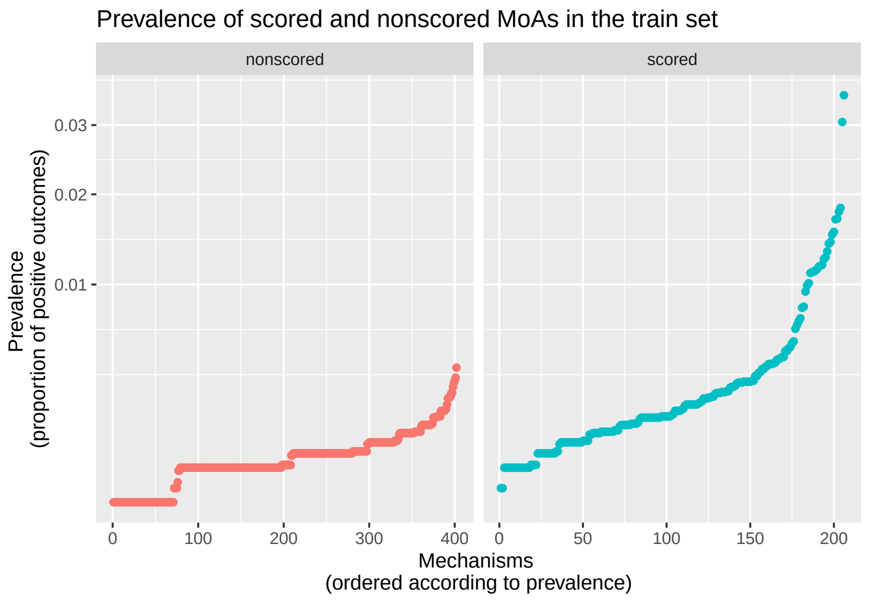

FIGURE: *Proportion of positive outcomes for each mechanism of action, measured in the train set. Prevalence is scaled with the sqrt() function. Nonscored mechanisms amount to around 2/3 of the total, but the most frequent ones are in the scored set. Only the two of the most common mechanisms are present in over 3% of the samples.*

The figure makes it evident that the prevalence varies widely. Mechanism `nfkb_inhibitor` is the most common, with 832 positive outcomes in the `train` set, followed by `proteasome_inhibitor` with 726. In contrast, some mechanisms only appear a few times. It is relevant then to investigate the concentration of positive outcomes in the most common mechanisms.

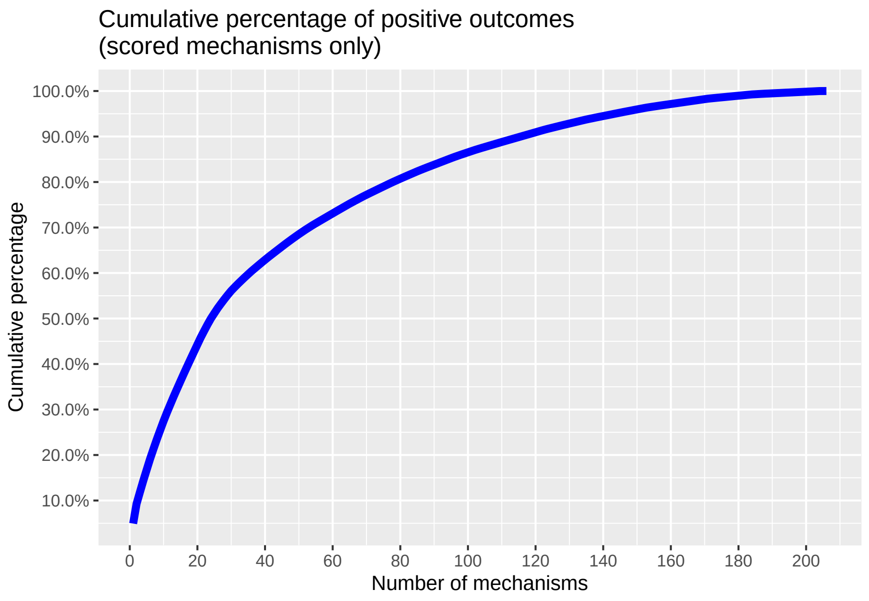

FIGURE: *Cumulative proportion of positive outcomes in the scored mechanisms set. The 24 most common MoAs account for 50% of the positive outcomes, while the 116 most common account for 90%.*

With the vast amount of mechanisms whose outcomes we attempt at predicting, and the fact that they are not mutually exclusive, it is relevant to investigate whether some mechanisms present a high correlation. To see that, we apply the `cor()` function to the matrix containing the binary outcomes for each mechanism and each sample, and arrange the results into a data frame with correlations for each pair of mechanisms:

Table 6: pairs of mechanisms with correlation larger than .25

	pair of mechanisms	correlation
1	nfkb_inhibitor & proteasome_inhibitor	0.921
2	kit_inhibitor & pdgfr_inhibitor	0.916
3	flt3_inhibitor & kit_inhibitor	0.758
4	flt3_inhibitor & pdgfr_inhibitor	0.705
5	aldehyde_dehydrogenase_inhibitor & trpv_agonist	0.529
6	nitric_oxide_production_inhibitor & nrf2_activator	0.408
7	apoptosis_stimulant & caspase_activator	0.403
8	insulin_sensitizer & ppar_receptor_agonist	0.403
9	norepinephrine_reuptake_inhibitor & serotonin_reuptake_inhibitor	0.342
10	fgfr_inhibitor & vegfr_inhibitor	0.334
11	caspase_activator & hiv_inhibitor	0.332
12	ikk_inhibitor & nitric_oxide_production_inhibitor	0.316
13	bcl_inhibitor & nitric_oxide_production_inhibitor	0.311
14	mtor_inhibitor & pi3k_inhibitor	0.296
15	pdgfr_inhibitor & vegfr_inhibitor	0.269
16	ikk_inhibitor & nrf2_activator	0.257
17	chloride_channel_blocker & glutamate_inhibitor	0.256
18	bcl_inhibitor & nrf2_activator	0.253

We see from the table that only 18 pairs of mechanisms have a correlation above .25 (out of a total of $\binom{206}{2} = 21115$ possible pairs). This indicates that the task of predicting the probability of a positive outcome can be treated as an independent prediction task for each of the mechanisms.

Predictors

We now move to the analysis of the predictors contained in the dataset and of how they relate to the known outcomes in the `train` set.

The following three tables show the effects of variables `cp_type`, `cp_time`, and `cp_dose` in the proportion of samples with positive outcomes for the scored mechanisms.

Table 7: effect of cp_type variable on amount of outcomes

	cp_type	samples with any MoA	samples with no MoA	average positive outcomes
1	ctl_vehicle	0.000	1.000	0.000
2	trt_cp	0.658	0.342	0.767

Table 8: effect of cp_time variable on amount of outcomes

	cp_time	samples with any MoA	samples with no MoA	average positive outcomes
1	24	0.607	0.393	0.707
2	48	0.606	0.394	0.707
3	72	0.607	0.393	0.708

Table 9: effect of cp_dose variable on amount of outcomes

	cp_dose	samples with any MoA	samples with no MoA	average positive outcomes
1	D1	0.607	0.393	0.708
2	D2	0.606	0.394	0.707

We see that, as expected, samples with `ctl_vehicle` on the `cp_type` variable have no *MoA*. As for the other two variables shown above, there is virtually no effect on the proportion of samples with positive outcomes.

We now turn our attention to the gene expression and cell viability features, which account for the vast majority of the available predictors.

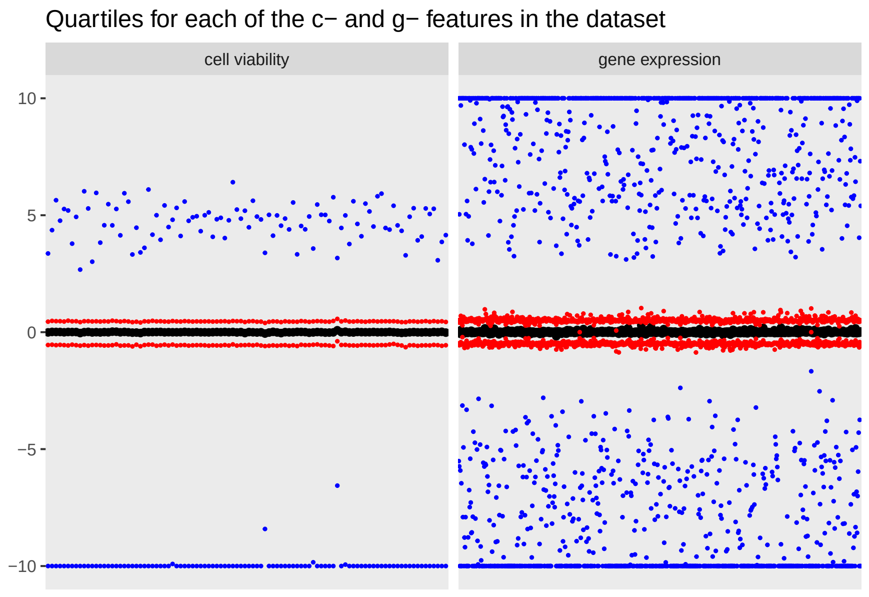

FIGURE: *Quartiles for each of the cell viability and gene expression features in the dataset. The black points in the center are the median for each feature. The two lines of red points below and above the center mark the 1st and 3rd quartiles, and the blue points at the bottom and top are the minimum and maximum.*

From the figure above, we see that the cell viability and gene expression features are already nearly centered, and that the interquartile range (IQR) is relatively narrow in comparison with the minimum and maximum values. We also notice that both sets of features are limited to the -10:10 range, which may indicate that they have already been pre-processed according to standard practices in this subject matter.

We proceed with an evaluation of the results of a principal components analysis on the `c-` and `g-` features. As discussed in the Data Preparation section above, we choose to apply this dimensionality reduction technique to the set of class centroids, with the goal of finding a coordinate system better suited for the classification task. We also limit the scope of the PCA to the samples in the `trainSingle` partition, so as to avoid data leakage between the partitions leading to overconfident performance estimates.

The figure below shows the cumulative proportion of variance explained by the principal components. We see that there is a significant concentration of the total variance in the first few coordinates, which is an important indication that the prediction problem can be well represented in a reduced feature space.

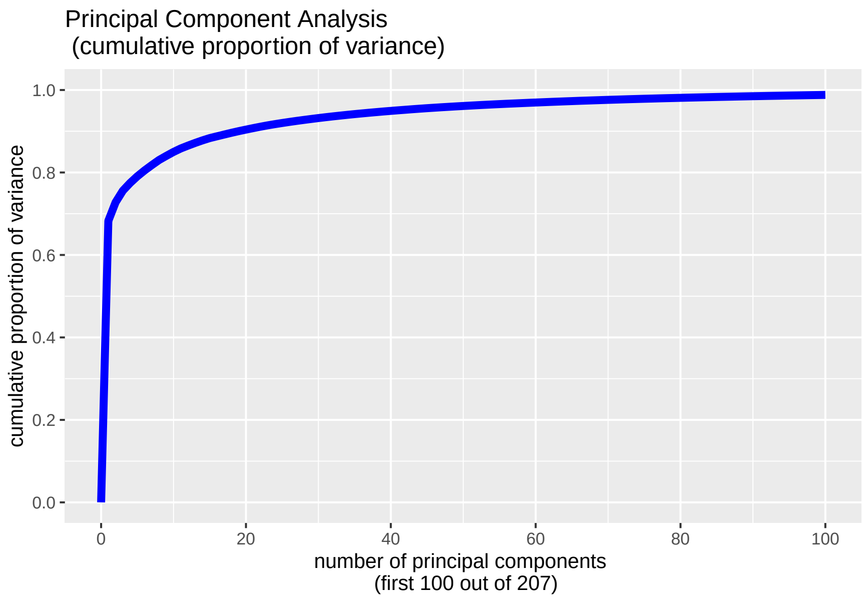

FIGURE: *cumulative proportion of total variance after application of PCA to the class centroids. Only 3 PCs are necessary to account for 75% of the total variance, while 41 PCs account for 95%.*

It is worth noting that `prcomp()` returned 207 principal components (one more than the 206 scored mechanisms). This is due to the inclusion of an additional centroid in the analysis, corresponding to the samples with no positive mechanisms.

Finally, we look at how effective the principal components seem to be in separating the positive from the negative targets. The figure below shows a scatter plot with a sample containing equal amounts of positive and negative outcomes for a selection of the most frequent *MoAs*, situated according to the first and second principal components.

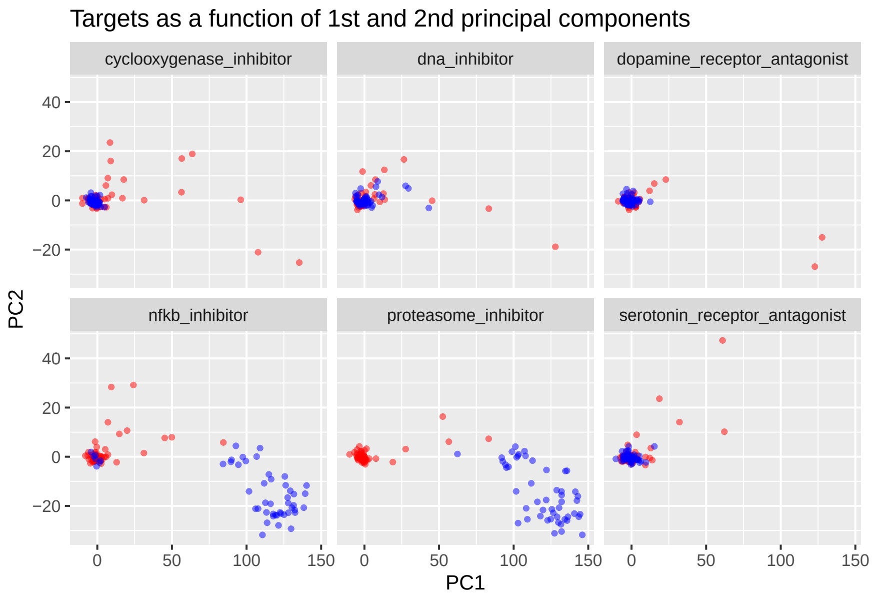

FIGURE: *scatter plots of outcomes as a function of first and second principal components for a selection of mechanisms. Red dots represent negative targets, while blue dots represent positive ones. The selected mechanisms are the ones with higher prevalence. Each plot contains 50 positive and 50 negative outcomes.*

These scatter plots provide an optimistic outlook for the classification problem. Mechanisms `nfkb_inhibitor` and `proteasome_inhibitor` (which happen to be the two most frequent) are very neatly clustered on the right of the PC1/PC2 plane. One might expect that the other mechanisms would be as clearly separated when looking at a different combination of principal components. However, that does not happen to be the case. A scatter plot matrix for the `cyclooxygenase_inhibitor` mechanism (the third most common) and other pairs of PCs illustrate the situation.

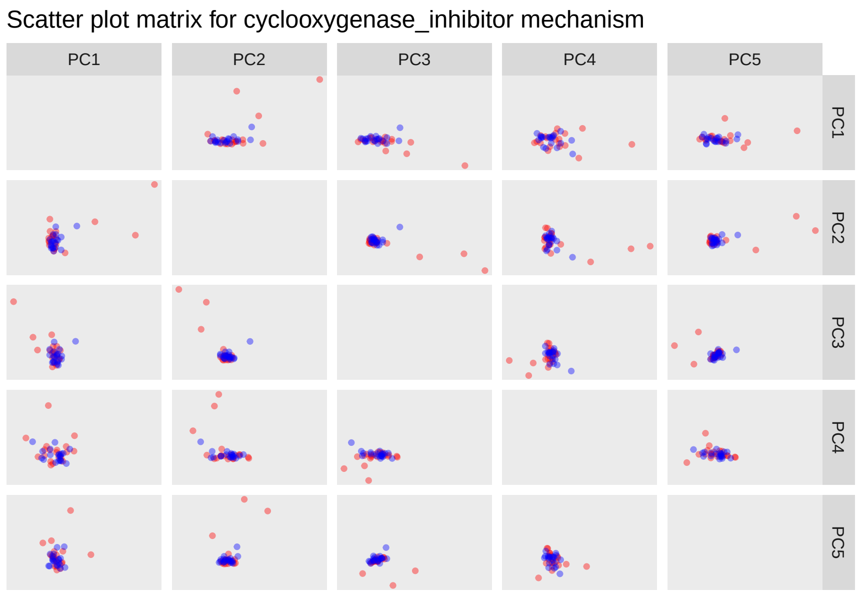

FIGURE: *scatter plot matrix for the cyclooxygenase_inhibitor mechanism and the five first principal components. Red dots indicate a negative outcome and blue dots indicate a positive one. A random sample of 20 positive and 20 negative outcomes was selected. Although some negative samples are spread out away from the origin at each of the plots, the majority of samples are concentrated in a single cluster, regardless of the outcome. There is no clear separating line in any of the plots.*

The setup we see for this particular mechanism is typical of the entire selection. Similar scatter plot matrices for other *MoAs* show a very similar distribution of positive and negative outcomes. This indicates that, for the majority of the mechanisms, the classification algorithms will have difficulty in identifying positive outcomes.

The table below demonstrates this point for the entire set of mechanisms and principal components. Out of a total of $206 \times 207 = 42,642$ pairs of mechanisms and principal components, only 16 have the absolute value of the correlation larger than 15%.

Table 10: mechanisms with at least one PC having abs(cor) > 15%

	mechanism	PC	correlation
1	proteasome_inhibitor	PC1	0.7878580
2	nfkb_inhibitor	PC1	0.7344576
3	hdac_inhibitor	PC5	0.3579651
4	glucocorticoid_receptor_agonist	PC13	-0.3559390
5	cdk_inhibitor	PC11	0.3275150
6	raf_inhibitor	PC2	0.3018807
7	tubulin_inhibitor	PC3	-0.2796824
8	hsp_inhibitor	PC2	0.2763136
9	mtor_inhibitor	PC8	0.2717162
10	hmgcr_inhibitor	PC46	-0.1919193

	mechanism	PC	correlation
11	atpase_inhibitor	PC34	0.1908298
12	topoisomerase_inhibitor	PC11	0.1844219
13	mek_inhibitor	PC2	0.1765932
14	protein_synthesis_inhibitor	PC39	0.1687375
15	pi3k_inhibitor	PC8	0.1605075
16	egfr_inhibitor	PC29	-0.1594351

To make sure that this limitation is not a result of a poor choice of pre-processing technique, we look at a similar table addressing the correlations between the outcomes and the original predictors:

Table 11: mechanisms with at least one predictor having abs(cor) > 15%

	mechanism	predictor	correlation
1	proteasome_inhibitor	c-78	-0.8155053
2	nfkb_inhibitor	c-78	-0.7598257
3	raf_inhibitor	g-202	-0.4008768
4	cdk_inhibitor	g-432	-0.3237779
5	glucocorticoid_receptor_agonist	g-100	0.3205263
6	hsp_inhibitor	g-117	0.2819912
7	egfr_inhibitor	g-235	0.2530632
8	tubulin_inhibitor	g-392	0.2356839
9	hdac_inhibitor	g-476	-0.2280774
10	hmgcr_inhibitor	g-157	0.2265903
11	mek_inhibitor	g-769	-0.2170929
12	topoisomerase_inhibitor	g-90	0.2132032
13	mtor_inhibitor	g-599	0.1710197

The table shows that the poor correlation between the outcomes and the predictors is a feature of the original dataset.

Modeling

After having performed an exploratory data analysis of the dataset in the previous section, we have sufficient information to decide on the appropriate modeling approaches.

The first relevant insight is that the problem is categorized as a *multi-label classification task*. The different possible outcomes are not mutually exclusive, which means that the task cannot be formulated as a "choose the most likely among the different possibilities" problem. In fact, we have seen that the different mechanisms are barely correlated at all.

There are a few different approaches to this case. One of them is to translate the problem into a traditional *multi-class classification task*, where the classes are in fact mutually exclusive. In order to do so, new classes are created, one corresponding to each of the combinations of outcomes found in the `train` set. After that, any model capable of handling classification with more than 2 classes can be applied to make predictions. Finally, the predictions are translated to the original set of outcomes by adding up, for each mechanism, the predictions for all the combinations in which it appears. This is the approach applied in the multi-class penalized mixture discriminant analysis (PMDA) model described below.

A different approach is to treat the problem as a set of 206 independent classification tasks (one for each mechanism), where for each of them we attempt at distinguishing between the positive outcomes (the mechanism is in action) and negative outcomes (the mechanism is not in action). In either case, additional mechanisms may or may not also have positive outcomes. This is the One-vs-Rest (OvR) approach, and is the one taken for all models in this report except for the multi-class PMDA.

When using the former approach, a possible adaptation is to implement a *classifier chain*, where each classifier uses the predictions made by the other ones as features. For example, predictions made for the `nfkb_inhibitor` could be used as features in the `proteasome_inhibitor` classifier. However, this approach does not lend itself well for the case at hand, where the vast majority of mechanisms are uncorrelated, and therefore offer very little "predictive power" to classifying the others.

Another important insight gained through the exploratory data analysis is the fact that the dataset is highly imbalanced. Even the most frequently found mechanisms are only positive in around 3% of the samples. In a scenario like this, a classification algorithm would only indicate a positive outcome in cases with a very clear separation between positive and negative samples.

While some techniques are available to deal with imbalanced datasets, like upsampling (taking bootstrap samples of the underrepresented class until proportions match) and SMOTE (downsampling the overrepresented class and creating synthetic samples for the underrepresented class until proportions match), they are not appropriate for the task at hand. Because the performance metric chosen for the competition is *log loss*, a misrepresentation of the prior probabilities for each class significantly hurts the performance. If the performance metric were not dependent on predicted class probabilities (as is the case with the *F-measure*) or if there were different penalties for misclassifying positive and negative outcomes (as is the case with the results from a medical exam), these techniques would be appropriate.

An additional consequence of the choice of *log loss* as a performance metric is that there is a severe penalty to overconfident incorrect predictions. In this setting, it is advantageous to impose a cap on the level of certainty with which the different models predict the outcome. Upon some experimentation, a lower limit of 10% of the mechanisms' prevalence and an upper limit of 99% certainty of a positive outcome seemed optimal.

Another relevant point is the fact that most mechanisms do not show a clear separation between positive and negative outcomes, both in the original feature space and in the PCA-transformed space. This characteristic, combined with the high class imbalance discussed above, means that for most mechanisms the models will not be able to clearly identify positive outcomes. Rather than that, predictions will fluctuate around very low probabilities of a positive outcome.

In this regard, the prediction task may seem a lot more like a regression on the conditional probability of a positive outcome than a proper binary classification.

With all these insights in mind, a selection of different prediction models were trained to predict the probability of a positive outcome at each sample for each of the mechanisms. Then, an ensemble model was created using the predictions from the previous models as features to make a final prediction.

The following sections describe each of those models. Results are then discussed in details in the Results section.

A note on optimizing performance through the choice of tuning parameters: even though some of these models have tuning parameters that can be finely tuned for optimal performance, the fact that we each of them is actually being used to fit 206 separate models makes is very time-consuming to do a grid search for optimal values. Thus, the parameters used here were chosen via a combination of experimentation and good judgment.

Benchmark

We begin the modeling phase by constructing the benchmark model, against which the subsequent ones will be compared. For this model, we predict the probability of a positive outcome for each samples/mechanism combination as the prevalence for that mechanism, as measured in the `trainSingle` partition without the control samples. For the control vehicle samples (`cp_type == ctl_vehicle`), we set the prediction to zero.

Upon measuring the performance of this prediction method on the `validate` partition, we get a *log loss* of 0.0202 and a *root mean squared error* (RMSE) of 0.0576.

Logistic Regression

We then perform a logistic regression for each of the scored mechanisms, using the `train()` function from the `caret` package and the method `glm` with parameter `family = "binomial"`. Predictions for control samples are explicitly set to zero and the model is trained on the remaining samples in the `trainSingle` partition. Some experimentation shows that a combination of the `cp_time`, `cp_dose`, and the first 20 principal components provide the best results.

Predictions are then made for the entire dataset and a lower and upper limit on predictions is imposed, as described above.

The logistic regression model arrives at a *log loss* of 0.0168 and *RMSE* of 0.0530 in the `validate` partition.

K Nearest Neighbors

A K nearest neighbors model is also fit to the data, using the `train()` function from the `caret` package and the `knn` method. Variables `cp_time`, `cp_dose` and the first 10 principal components are used as predictors. Predictions for control samples were explicitly set to zero, and a cap was imposed on the minimum and maximum levels of certainty.

The number of neighbors is a tuning parameter that needs to be set in order to optimize performance. Because we are trying to predict a continuous value (probability of positive outcome) but the outcomes are actually binary, K needs to be large enough so as to avoid predictions that are too “digitized”. For example, a value of $K = 20$ would only permit predictions that are integer multiples of 5%. Since the mechanisms have prior probabilities in the range of 0-3%, this value of K does not provide enough granularity to properly predict the probabilities.

On the other hand, large values of K make it so that each prediction averages the outcomes of samples that are not necessarily close to the one we are trying to predict. Additionally, predictions with a value of K that is too big have an annoying tendency of crashing R in the process.

An appropriate compromise was found at a value of $K = 200$. Measurements in the `validate` partition show a *log loss* of 0.0179 and *RMSE* of 0.0527.

Naïve Bayes with loess smoothing

A Naïve Bayes model was fit to the data, using a selection of principal components as predictors. With this approach, each predictor is assumed to be independent and has the effect on the outcome calculated separately.

For each mechanism, each principal component was cut into numerous bins and the conditional probability of a positive outcome was calculated on each bin. Then, a *loess* smoothing was applied to the conditional probability curve for each of the PCs with the `loess()` function. The predicted probability of a positive outcome was then calculated as the average of the predictions obtained separately from each of the principal components. Control samples had the predictions set to zero and were not used in tuning.

Parameters were chosen globally and applied to every mechanism/PC combination. A selection of the first 3 principal components was used, each with its range divided into 15000 bins and smoothed with a 0.3 span.

With these parameters, a *log loss* of 0.0182 and *RMSE* of 0.0557 was obtained in the `validate` partition.

Support Vector Machine

A support vector machine (SVM) model was fit to the data using the `train()` function with method `svmLinear2`. This method permits tuning the model using the `cost` parameter. After fitting a model to each mechanism, the `predict()` function was used with parameter `type = "prob"` to get the predicted probability of a positive outcome.

The SVM model makes predictions by finding the optimal separating hyperplane between the classes. It has a significant advantage for addressing classification problems in the fact that it is able to find separating hyperplanes that are oblique (as opposed to models like classification trees that only create new branches by splitting one variable at a time). This model can also make continuous estimates of the class probabilities according to each point's distance to the separating hyperplane.

For each mechanism, an SVM model was created using the `cp_time` and `cp_dose` predictors and the first 50 principal components. The cost parameter was chosen for each mechanism through *leave-group-out cross-validation*, with 90% of the samples used for training.

With these settings, a *log loss* of 0.0161 and *RMSE* of 0.0515 were obtained.

Multi-class Penalized Mixture Discriminant Analysis

With some manipulation on the outcome variables, the *multi-label classification task* was formulated as a *multi-class classification task* and a penalized mixture discriminant analysis (PMDA) was fit to the data.

First, the combinations of outcomes observed in the `trainSingle` partition were identified and given an unique id. As seen in the exploratory data analysis, these combinations can be made up of up to 11 different mechanisms. A total of 695 different combinations of scored and nonscored mechanisms has been identified, a sample of which is shown below:

Table 12: sample with different combinations of outcomes found in the trainSingle partition

	outcome	appearances	id
1	flt3_inhibitor | jak_inhibitor | growth_factor_receptor_inhibitor	1	id_693
2	gat_inhibitor	6	id_397
3	ampk_inhibitor	5	id_458
4	serine_threonine_kinase_inhibitor	34	id_094
5	androgen_receptor_antagonist | cytochrome_p450_inhibitor	5	id_460
6	mucus_protecting_agent	22	id_153
7	neurotensin_receptor_agonist	10	id_292
8	platelet_aggregation_inhibitor | structural_glycoprotein_antagonist	5	id_543
9	fabi_inhibitor	10	id_288
10	lxr_agonist | abc_transporter_expression_enhancer | ror_inverse_agonist	4	id_623
11	cc_chemokine_receptor_agonist	3	id_654
12	contraceptive_agent	10	id_283

In this formulation, the outcomes are in fact mutually exclusive, which means that any classification algorithm capable of dealing with more than 2 classes can be applied. Variations of the linear discriminant analysis (LDA) are particularly adequate for this scenario, as the feature distribution for each of the classes is calculated separately and the class probabilities are then calculated via Bayes' formula.

In a mixture discriminant analysis model, the feature distribution for each class is calculated as a combination of distinct gaussian functions. The number of such functions can be chosen as a tuning parameter, and its centroids are determined during the fit process via a clustering algorithm. This adaptation is advantageous when the data is not normal, as is the case here.

The "penalized" term indicates that a shrinkage procedure is automatically applied to the class centroids, which minimizes overfit and allows for more robust predictions.

The PMDA model is fit using the `mda()` function from the `mda` package. The parameter `method = gen.ridge` tells the function to automatically apply the regularization. The number of subclasses is chosen as 2 after some experimentation, and the predictors are chosen as the first 10 principal components. After fitting the model, predictions are made for the entire dataset and multiplied by a conversion matrix that translates the predictions from the space of combinations of outcomes to the space of individual mechanisms.

Upon measuring the performance of this model in the `validate` partition, a *log loss* of 0.0175 and a *RMSE* of 0.0543 are found.

Ensemble with weighted average

As a final model for the probability of a positive outcome, we construct an ensemble of the previous models. A simplistic approach would be to predict the average between the other predictors, giving equal weight to all the models. However, we have seen that the *log loss* performance of the different models are significantly different, so an ideal ensemble would give an increased weight to the best performing models.

A linear regression model could be applied to the predictions, using the predicted probabilities from the previous models as features to a regression. However, the nature of the problem poses limitations to this approach. Because the features are highly correlated (evidently, since they are all attemps at predicting the same thing), a simple linear regression would lead to coefficients that are either very large positive or negative numbers. In this setup, there is a strong tendency to overfit the data, and to construct a model that is not robust.

An alternative is to apply a linear regression with some sort of shrinkage of each model's coefficient towards unity, meaning that the fit would penalize a combination that strays too much from a simple average.

Since penalized models normally provide shrinkage towards zero (instead of unity), we begin by calculating the simple average of the predictions from the previous models and then performing a linear regression on the residuals between the simple average and the ground truth. A model constructed like this does result in shrinkage towards the simple average, while allowing for adjustments to the weights of each of the predictors.

To do so, we calculate the residuals and then use the `train()` function from the `caret` package with the `penalized` method, which permits tuning from a selection of L1 and L2 penalty parameters. We use the `trainEnsemble` partition to fit one model for each of the scored mechanisms, with 10-fold cross-validation to evaluate a grid of penalty parameters.

After fitting the model and calculating predictions on the `validate` partition, we get a *log loss* of 0.0158 and a *RMSE* of 0.0513.

Results

In this section, we discuss the results obtained after applying the models constructed above to make probability predictions on the `validate` partition.

First, we look at a selection of performance metrics for each of the individual models:

Table 13: model performance on validate partition

	method	logLoss	RMSE	accuracy	balancedAccuracy
1	benchmark	0.020162	0.057550	0.996654	0.500000

	method	logLoss	RMSE	accuracy	balancedAccuracy
2	logit	0.016777	0.052989	0.997070	0.543395
3	knn	0.017861	0.054280	0.996926	0.527204
4	naiveBayes	0.018237	0.055684	0.996771	0.512528
5	svmLinear	0.016104	0.051457	0.997273	0.551100
6	pmda	0.017529	0.054350	0.996924	0.543018
7	ensembleMean	0.016420	0.052801	0.997066	0.532140
8	ensembleWeighted	0.015845	0.051326	0.997279	0.550799

As expected, the ensemble model with weighted means performs better than any of the other individual models both in terms of *log loss* and *RMSE*. In fact, it outperforms the benchmark by around 21%.

Both accuracy and balanced accuracy were calculated by rounding the probability of a positive outcome to the nearest integer. This amounts to assigning a positive outcome to a particular mechanism on a particular sample when the predicted probability of positive is greater than 50%.

Since the dataset is so highly imbalanced (there are many more negative outcomes than positive ones), the accuracy metric does not provide much valuable information. On the other hand, balanced accuracy (the average between the accuracy obtained in the negative cases and the positive cases) seems to be more informative.

We note that the balanced accuracy metric is not much better than the benchmark for any of the models. At look at the confusion matrix and related metrics for the final model helps elucidate.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction       0       1
##          0 4888583   13150
##          1     257    3694
##
##                Accuracy : 0.9973
##                  95% CI : (0.9972, 0.9973)
##     No Information Rate : 0.9966
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3544
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.2193066
##             Specificity : 0.9999474
##          Pos Pred Value : 0.9349532
##          Neg Pred Value : 0.9973173
##              Prevalence : 0.0034336
##  [ reached getOption("max.print") -- omitted 6 rows ]
```

The confusion matrix reveals a very low sensitivity, indicating that the model has a severe difficulty in identifying positive outcomes. In fact, there are many more cases of false negatives than there are true positives.

Since the majority of the single models that made up the final ensemble model were themselves ensembles of models fit individually for each of the different *MoAs*, it is interesting to evaluate the performance separately for each of the mechanisms.

The table below shows information on the mechanisms for which a sensitivity above 50% was reached. We see that it was only the case for 14 out of the 206 scored mechanisms. Not surprisingly, this list favors the ones with a significant correlation with at least of the predictors. One mechanism in particular - `proteasome_inhibitor` - has a sensitivity of 1, indicating that every single positive outcome has been correctly identified.

Table 14: mechanisms with over 50% sensitivity measured in the validate partition

	mechanism	sensitivity	meanLogLoss	predictor	correlation
1	proteasome_inhibitor	1.000	0.005	c-78	-0.816
2	glucocorticoid_receptor_agonist	0.870	0.013	g-100	0.321
3	nfkb_inhibitor	0.868	0.030	c-78	-0.760
4	raf_inhibitor	0.850	0.011	g-202	-0.401
5	hsp_inhibitor	0.833	0.004	g-117	0.282
6	chk_inhibitor	0.800	0.002	g-148	0.074
7	flt3_inhibitor	0.682	0.019	g-157	0.100
8	cdk_inhibitor	0.657	0.016	g-432	-0.324
9	tubulin_inhibitor	0.629	0.027	g-392	0.236
10	hmgcr_inhibitor	0.621	0.023	g-157	0.227
11	kit_inhibitor	0.600	0.022	c-65	-0.103
12	mek_inhibitor	0.600	0.004	g-769	-0.217
13	pdgfr_inhibitor	0.517	0.030	c-65	-0.117
14	egfr_inhibitor	0.500	0.035	g-235	0.253

We proceed to evaluate a plot of the *log loss* performance obtained by each of the methods and for each of the mechanisms. In the figure below, individual mechanisms are in the horizontal axis, while the performance for each of the models is represented by points of different colors for each of the models.

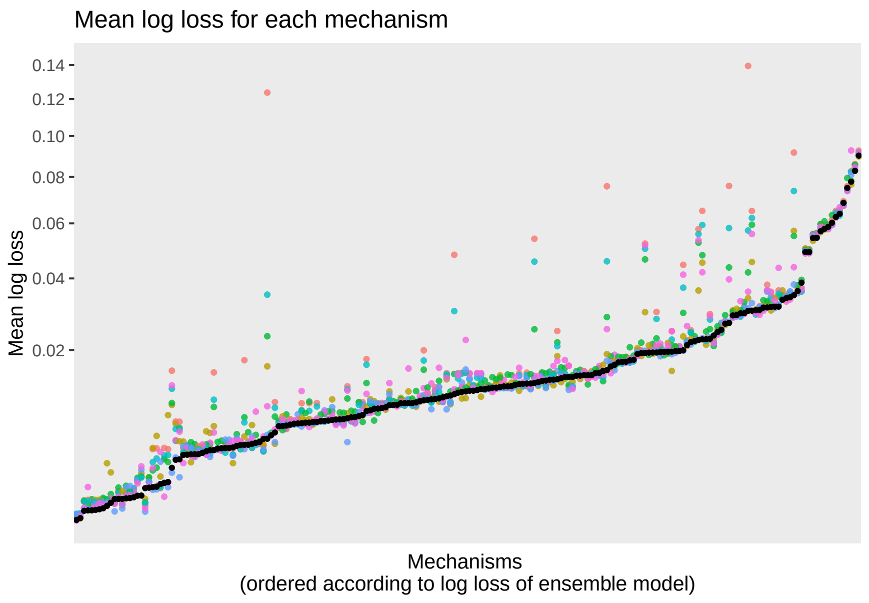

FIGURE: *mean log loss measured separately for each mechanism. The colored dots represent each of the individual models, while the black dots represent the final weighted ensemble. The vertical axis is scaled with the square root.*

Because the mechanisms are ordered according to the performance obtained by the final model, there is a distinguishable line formed by the black dots corresponding to this model. Even though the ensemble model with the weighted averages does perform better than any of the single models, when looking at each mechanism individually the final model most times is not the one with the lowest score.

We also note that the final model does not seem to be negatively affected by the outliers - cases where a particular models performs a lot worse than the others for a particular mechanism. This shows that the technique of applying a linear regression model to the residuals after averaging out the single models was effective in reducing the weight of these under-performing models.

Also, these results do seem to provide a visual indication that there is a lower limit to the prediction capability of any model.

We proceed to look at the *log loss* obtained by the final model for each of the mechanisms, as a function of the prevalence of the mechanisms on the `trainSingle` partition:

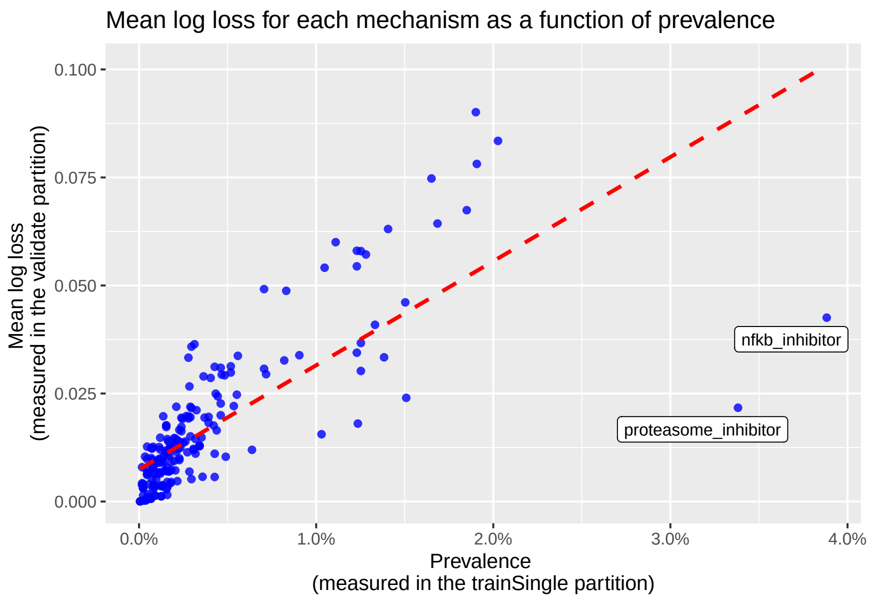

FIGURE: *mean log loss for each mechanism measured in the validate partition. There is a clear positive correlation between the log loss and each mechanism's prevalence in the trainSingle partition, as illustrated by the red dashed linear regression line. Two clear outliers are identified, corresponding to the nfkb_inhibitor and proteasome_inhibitor mechanisms, both of which had a high correlation with at least one of the predictors and thus had particularly low losses.*

This scatter plot shows a positive correlation between the loss metric and each mechanism's prevalence. This can be explained by the fact that, with a higher prevalence, predicted probabilities of a positive outcome are typically not as low, so the cases with a negative outcome suffer a larger performance penalty.

Another relevant indication of the performance of a probability prediction model is whether the model is balanced: that is, whether a prediction with a certain confidence does prove to be correct with a frequency matching that confidence.

To evaluate this feature, we begin by dividing the predictions into bins and evaluating the amount of predictions that fell into each bin. Because there is a tendency for concentration into predictions that are either close to 0% or 100%, we create a larger number of bins around these points. The table below shows the resulting quantities and mean *log loss* for each bin:

Table 15: binned number of predictions in validate partition

	range of predictions	quantity	mean log loss
1	[0,0.01]	458050	0.0107946
2	(0.01,0.02]	21842	0.0725392
3	(0.02,0.03]	6116	0.1106241
4	(0.03,0.04]	662	0.1665553
5	(0.04,0.05]	180	0.2684639
6	(0.05,0.06]	120	0.1988925
7	(0.06,0.07]	96	0.3160992
8	(0.07,0.08]	73	0.3174481
9	(0.08,0.09]	54	0.3510149
10	(0.09,0.1]	58	0.4073738
11	(0.1,0.2]	232	0.3454566
12	(0.2,0.3]	96	0.4794203
13	(0.3,0.4]	51	0.6894959
14	(0.4,0.5]	25	0.7264276
15	(0.5,0.6]	25	0.6598812
16	(0.6,0.7]	14	0.6462603
17	(0.7,0.8]	26	0.6482975
18	(0.8,0.9]	32	0.2745220
19	(0.9,1]	262	0.0679060

The table confirms the intuition that most of the predicted probabilities are in the low numbers. In fact, the vast majority of them are under 1% chance of positive outcome. Because of the significant class imbalance, we see that there are a lot less positive predictions (probabilities over 50%), but these do have a tendency to concentrate on the 90-100% range.

We also note that the mean *log loss* is severely penalized outside the 0-1% prediction range. This is also a direct consequence of the class imbalance, confirming that under these conditions a low probability prediction is a "safer bet".

To evaluate whether the prediction model is balanced, we calculate the rate of actual positive outcomes inside each of the bins and plot them against the average prediction inside each bin. The figure below depicts this information:

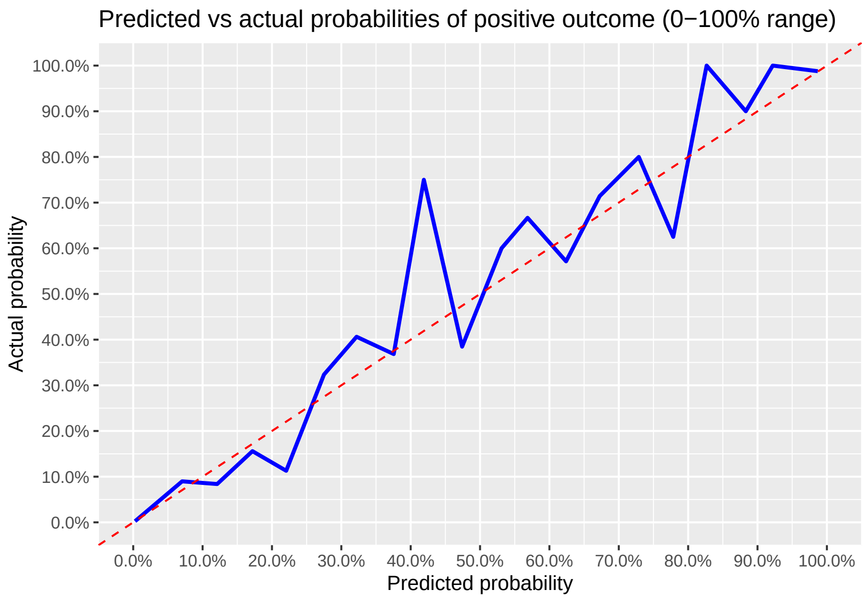

FIGURE: *predicted probability vs actual probability of a positive outcome, measured in the validate partition. The blue line indicates the relationship between the two probabilities. The red dashed line is the identity line and serves as a reference of the expectation of a balanced prediction.*

The figure shows that the predicted probability do somewhat follow the identity line, albeit with some oscillation. Notably, there is a significant difference close to the line of 40% predicted probability.

However, since the majority of predictions are concentrated in the lower range, the 0-5% range of predictions has a lot more weight on the final performance metric. Therefore, having a balanced prediction model in this range is much more important to the final results.

In order to evaluate this aspect in more details, we look at a plot with the same information, but focussed on the 0-5% range, where the prior for all mechanisms lie:

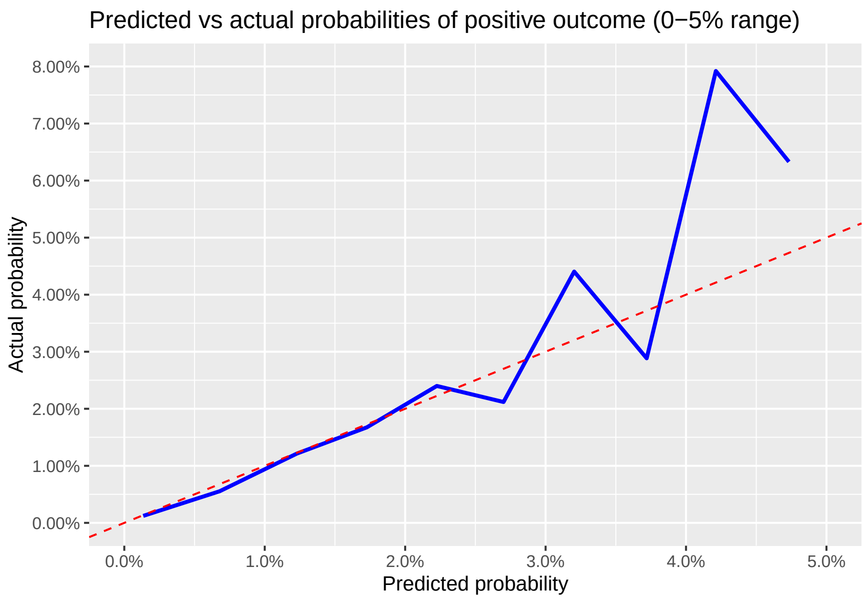

FIGURE: *predicted probability vs actual probability of a positive outcome, measured in the validate partition, zoomed to interval from 0 to 5%.*

We see that, up to around 3%, the predictions tend to be reasonably balanced. Since the plot was created only with samples from the `validate` partition, this phenomenom is not due to overfitting.

Finally, we look at a table with the results for the mechanisms where the best performance has been achieved with the final model:

Table 16: 15 mechanisms with lowest log loss from final model (only mechanisms with more than 10 positives)

	mechanism	logLoss	sensitivity	numPositives
1	proteasome_inhibitor	0.0048	1.0000	65
2	raf_inhibitor	0.0111	0.8500	20
3	glucocorticoid_receptor_agonist	0.0131	0.8696	23
4	cdk_inhibitor	0.0156	0.6571	35
5	flt3_inhibitor	0.0194	0.6818	22
6	bromodomain_inhibitor	0.0195	0.1818	11
7	topoisomerase_inhibitor	0.0200	0.3889	18
8	kit_inhibitor	0.0224	0.6000	25
9	hmgcr_inhibitor	0.0226	0.6207	29
10	protein_synthesis_inhibitor	0.0227	0.0909	11
11	tubulin_inhibitor	0.0268	0.6286	35
12	androgen_receptor_antagonist	0.0288	0.0000	11
13	prostanoid_receptor_antagonist	0.0289	0.0000	11
14	gaba_receptor_agonist	0.0294	0.0000	11
15	cytochrome_p450_inhibitor	0.0294	0.0000	12

We see that, for some mechanisms, the final model obtained a reasonably high sensitivity and low *log loss*. However, some of them have a low loss metric only at the cost of a null sensitivity and low number of positive occurrences in the `validate` partition.

Conclusion

In this project, we constructed a model that uses cell viability and gene expression data to estimate the probability that a particular mechanism of action is active when a drug is applied in a certain dosage and exposure time to a cell sample. The model was constructed as an ensemble of several individual models, calculated as a weighted average with different weights for each mechanism.

These individual models were fit using the `trainSingle` partition of the training set, with the ensemble tuned using the `trainEnsemble` partition. Results were evaluated in the `validate` partition and additional predictions were made in the `test` partition, for which the correct results were not provided.

This problem has been posed as machine learning competition at the *kaggle.com* website.

Results showed a significant improvement over the benchmark model, with the *log loss* performance metric being reduced from 0.0202 to 0.0158, a 21% reduction.

Among the single models, 5 of them treated the problem as 206 individual prediction *multi-label classification tasks* (one for each mechanism, in an One-vs-Rest approach), while 1 treated it as a *multi-class classification task*, with tasks made up of combinations of individual mechanisms. The final model also addressed the task individually for each mechanism.

Results fluctuated a lot among different mechanisms, especially when looking at the sensitivity metric (ratio of correct classifications on actual positive outcomes). For many of the mechanisms, the models were grossly unable to identify positive outcomes, reaching very low (sometimes even null) sensitivity levels.

These results seem to indicate that the selection of predictors have very low predictive power over many of the mechanisms, holding very little information about whether or not these mechanisms are in effect. The application of the same methodology to a different set of genes might arrive at much better results.

However, for some other mechanisms, very high sensitivity levels and low *log loss* were achieved, meaning that the model does have the potential to be used in practice.

Because this project is being presented as part of the capstone project for the Data Science Professional Certificate program, the models were selected in an attempt to use techniques that were either addressed in the courses or are somewhat related to these techniques. In an extension to this work, other methods could be experimented with, such as boosting and neural networks.

Nevertheless, it is worth noting that a choice of more sophisticated models does not necessarily ensure better performance. In the selection presented in this report, the simple logistic regression presented a lower loss than KNN, naïve Bayes and PMDA.

The choice of the cross-validation setup has been influenced by the fact that some sort of performance reporting is necessary for the project. The decision to set aside the `validation` partition of the training data permits calculating the performance on a set of samples with known outcomes, but effectively reduces the size of the training set and degrades the performance measured in the `test` set. However, this choice does allow for the evaluation of results in a setting with no risk of overfitting.

Ideally, after deciding on a modeling technique using this cross-validation approach, a final model would be fit to the entire `train` set and used to make preditions in the `test` set.

It is also worth noting that the term "prevalence" has been used in a loose sense during this project. It refers to the frequency with which each mechanism had a positive outcome, as measured in the `trainSingle` partition. However, there is no direct connection to a mechanism's proportion in a population, as the term suggests. In fact, it does not even make much sense to think of a population in this context, since the observations corresponded to samples (in the biological, not statistical sense) taken from a larger collection. The use of the "prevalence" term during modeling, be it explicitly or in the underlying assumption each of the models count on, assumes that the `train` set is statistically representative of the `test` set, which may not be the case.

As a final comment on the project, the challenges involved in this prediction task turned out to be very different to what one might have expected. Surprisingly, the reduction of the loss metric did not depend heavily on discerning whether a particular mechanism should be classified as a positive or negative for a particular sample. It did depend much more heavily on accurately identifying the very low probability of a positive outcome, which typically lied in the 0-5% range. As such, the problem bears a much closer resemblance to a *regression* on the level of certainty with which it can be stated that the outcome is negative than to a *classification* where positive and negative outcomes are separated from one another.