Cancer Drugs Approvals Prediction with Machine Learning

Alassane Mamadou Watt : alassane.watt@student.ecp.fr,
Mohamed Amine Kessa : mohamed-amine.kessa@student.ecp.fr
CentraleSupelec

2018

Abstract

In oncology, developing a drug is not free from risks. It often requires huge investment and time and at the end, the drug might not fit the standards to be sold in the market. We propose a machine learning algorithm that predicts drugs approvals using pharmacological, biochemical and phase I clinical trials results data. For that purpose, we trained 3 models (Random Forest Classifier, Logistic Regression and Support Vector Machine) in different configurations and evaluated them with several metrics. We obtained very good results, up to 93.9% for ROC AUC metric on the testing set.

1 Introduction

Drug development in oncology is complex, time-consuming, uncertain and very expensive. It can take between 10 and 15 years to develop a drug, at a cost of about US$ 800 millions. After its development, the Food and Drug Administration (FDA) ensures that the drug is safe and especially that it is not making any false claims before it can be sold. Many of the potential new medicines fail during development, but some may become important new treatments of tomorrow. For every 5000 compounds tested, it is estimated that only one will be approved for use in patients. There are multiple steps to the development process. Those steps are drug discovery, preclinical tests and clinical trials. There are four clinical trial phases. Each one is designed to answer specific questions about a drug's safety and effectiveness. Predicting approvals of drugs will substantially save money and time on drugs that won't be approved by the FDA. That is why we propose a machine learning based method to make that achievement. Actually, the FDA doesn't reject the drug for good. Instead it tells drug developer to continue working on the project until it yields better results. Then the drug will be submitted again for approval by the FDA. So that way the drug will ultimately be approved by the FDA. Thus, our goal is to predict whether a given drug will be approved by the FDA after a certain period thanks to data we have collected from its discovery to the end of the phase 1 of the clinical trials. More precisely, the input to our models is pharmacological, biochemical and phase I results data and the output is FDA-approval after a certain period from its submission. Note that this is a supervised machine learning classification problem. The models we worked with are Random Forest Classifier, Support Vector Machine (with linear, radial basis function and polynomial kernels) and Penalized Logistic Regression. We tuned the hyperparameters of our models using the Grid Search algorithm. Finally, we assessed our models with Accuracy, Precision, Recall, ROC AUC and Specificity metrics.

2 Related Work

To the best of our knowledge, some works on drug approval prediction already exist but most of them are concentrated on specific therapeutic areas, and involve only one or a small number of predictive factors: Malik et al. [6] examined the trial objective responses of 88 anticancer agents in phase 1; Goffin et al. [3] studied the tumor response rates of 58 cytotoxic agents in 100 phase 1 trials and 46 agents in 499 phase 2 trials; El-Maraghi and Eisenhauer [2] looked at the objective responses of 19 phase 2 anticancer drugs in 89 single agent trials; Jardim et al. [4] examined the response rates of 80 phase 3 oncology drugs to identify factors associated with failures; and DiMasi et al. [1] analyzed 62 cancer drugs

and proposed an approved new drug index (ANDI) algorithm with four factors to predict approval for lead indications in oncology after phase 2 testing; Lo et al. [5] apply machine-learning techniques to predict drug approvals and phase transitions using drug-development and clinical-trial data from 2003 to 2015 involving several thousand drug-indication pairs with over 140 features across 15 disease groups

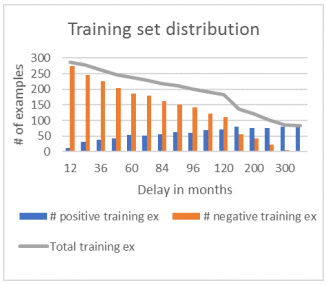
3 Data Preprocessing

3.1 Data cleaning

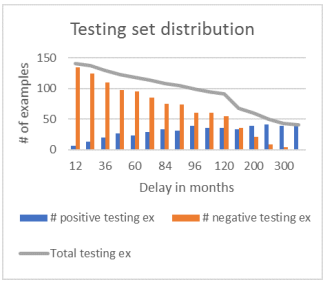
As we are doing this project for a clinic our data was collected from them. There are three types of data: pharmacological, biochemical and phase 1 results data. Initially we had multiple files of data. The format of the files was either txt or csv. We turn the txt files into csv for the sake of consistency and because it is easier to work with them in excel. Most of the features we had was categorical. Actually, there was some work already done in those files to turn most of the features categorical before we received the data. First of all, we explored the data thanks to the powerful tools provided by excel. All files didn't have the same number of drugs listed. The maximum number of drugs we found was 554. But all drugs were not labeled. So, we had to drop the unlabeled ones since we're facing a supervised machine learning problem. The number of labeled drugs was 486. We began our feature extraction manually in excel. There was obviously irrelevant data that was not going to impact the prediction of the labels. So, we dropped all the features we deemed irrelevant to our prediction. Afterward we loaded the data files in python using the Pandas library and merged them into a single data frame. The number of feature was then 4359 which is huge given the number of training data we had. As we expected it, there was some missing data that needed to be handled. We could delete the drug or the features with missing data or impute the missing data. Imputing data may lead to some errors and as we didn't have much data we avoided that practice. As a first approach, we chose to delete drugs with missing data as we couldn't assess the importance of those features with missing data. After deletion there was 431 labeled drugs without missing values.

3.2 Data engineering

As stated above our goal is to predict whether a given drug will be approved by the FDA after a certain period. In our work the period in months belonged to the following set $\{12, 24, 36, 48, 60, 72, 84, 90, 96, 108, 120, 200, 240, 360\}$. In the dataset, the delay between the first submission of the drug and the FDA-approval is known. We defined a function that modifies the labels of the drugs or delete them. If a drug is approved before the actual period then its label is 1. If the drug is not approved before that period its label is 0. And ultimately, if the delay between the submission of the drug and the censorship date is less than the actual period, the drug is removed since we can't say if it will be approved after that period. Still in the function we split the dataset into training and testing set with a ratio of 0.33 for the testing set. This is what our python function does. Thus, our function takes as input the full dataset and the period and outputs the training set, the testing set and their labels. Here below is the distribution of our dataset over the periods.



(a) Training set distribution.



(b) Testing set distribution.

Figure 1: Dataset distribution

We notice in these graphs that the size of the dataset decreases as time goes. That is in perfect sync with our expectations since the delay between the submission of a drug to the censorship remains fixed while the actual period increases, which means more and more drugs fall in the category of drugs whose delay are less than the actual period and therefore get dropped since we can't say whether they will be approved after that period. Besides, we notice a change in the relative importance of the labels. In the extreme sides of our period set, the classes are a heck of a lot imbalanced. In the early months, there are much more 0-labeled than 1-labeled examples, while in the late months, that's

the other way around (see figure 1). The converse occurs after around period = 180 months where we have quite balanced classes both in the training set and the testing set. However, the downside of that period is that there isn't that much data since the size of the data decreases over the months. In the sequel, since it's better to train a model with balanced classes, whenever we want to give specific examples, we will summon the results associated to that period.

3.3 Feature Selection

As we had over 4359 features, we felt the need to reduce the dimensionality of our data as they would probably be noise, irrelevant data and redundancy. For that we used a powerful tool provided by the Random Forest Classifier (RFC). RFC measures the importance of each feature giving it a score. This is how we proceeded to reduce the dimensionality of our data to around a few hundreds of features depending on the actual period. To understand what we were working on we didn't use another method of feature selection like PCA or LDA as it would mix the features and we wouldn't make sense out of the features fed to our models.

4 Models

We trained three models in different configurations. The models are Random Forest Classifier, Support Vector Machine (with linear, radial basis function and polynomial kernels) and Penalized Logistic Regression. The configurations we adopted are the following:

- Default parameters of the model and all the features of the dataset
- Best parameters of the model and the most important features of the dataset
- Best parameters of the model and the most important features scaled
- Best parameters of the model and all the features of the dataset
- Best parameters of the model all the features of the dataset scaled

Scaling some data means to transform it to so it has zero-mean and unit-variance.

4.1 Random Forest Classifier

To explain what Random Forest is, we need to first introduce the notion of a decision tree. A decision tree is a Machine Learning algorithm capable of fitting complex datasets and performing both classification and regression tasks. The idea behind a tree is to search for a pair of variable-value within the training set and split it in such a way that will generate the "best" two child subsets. The goal is to create branches and leafs based on some optimal splitting criteria, a process called tree growing. Specifically, at every branch or node, a conditional statement classifies the data point based on a fixed threshold in a specific variable, therefore splitting the data. To make predictions, every new instance starts in the root node (top of the tree) and moves along the branches until it reaches a leaf node where no further branching is possible. For a better understanding, see the figure 2 below:

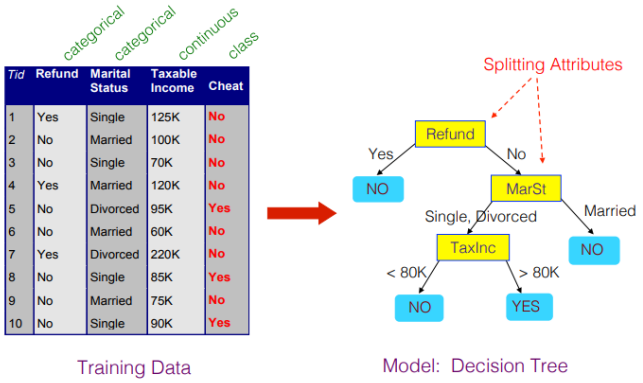


Figure 2: Decision tree training.

Now to predict a class, the new data is matched against the tree as the figure below shows:

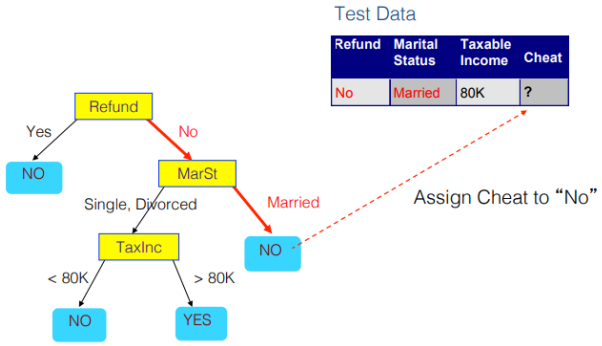


Figure 3: Decision tree class prediction.

Random Forest is built as an ensemble of Decision Trees to perform different tasks such as regression and classification. An ensemble method or ensemble learning algorithm consists of aggregating multiple outputs made by a diverse set of predictors to obtain better results. Formally, based on a set of "weak" learners we are trying to use a "strong" learner for our model. Therefore, the purpose of using ensemble methods is: to average out the outcome of individual predictions by diversifying the set of predictors, thus lowering the variance, to arrive at a powerful prediction model that reduces overfitting our training set. A good illustration of what random forest does is right after [8]:

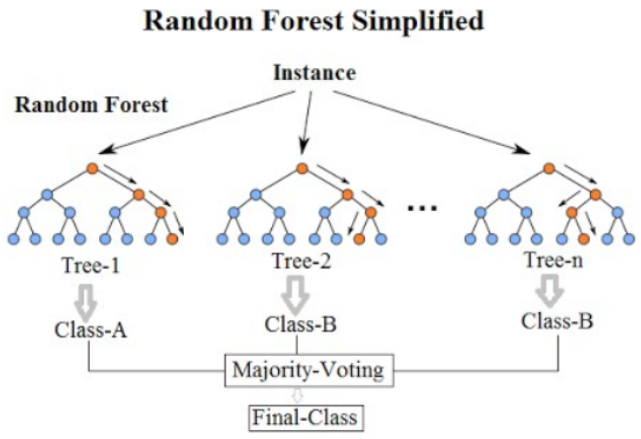


Figure 4: Random forest class prediction.

The parameters we tuned are "n_estimators", "max_features", "max_depth", "min_samples_split", "min_samples_ leaf" and "bootstrap".

4.2 Logistic Regression

Logistic Regression fits a linear separator in our feature space by maximizing the likelihood of observing the training set data. It aims to model the probability of a label given a piece of data, p(label|data), using the sigmoid or logistic function:
$y(x) = \sigma\left(\theta^T x\right) = \frac{1}{1+e^{-\theta^T x}}$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
The Probabilistic aspect is interpreted by the Bernoulli distribution:
$p(y=1|x;\theta) = y(x)$, which is the probability of success (i.e., y=1),
$p(y=0)|x;\theta) = 1 - y(x)$, which is the probability of failure (i.e., y=0).
Combined together we have more generally:
$p(y|x;\theta) = y(x)^y (1 - y(x))^{1-y}$
The label of the data is predicted to be 1 if $y(x) > 0,5$.
Now to train the model, assuming that the m training examples were generated independently, we can write down the likelihood of the parameters:
$L(\theta) = p(y|X;\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$
where X is the training set matrix. The model is trained by maximizing the log likelihood of the data,
$\mathcal{L}(\theta) = \log(L(\theta)) = \sum_{i=1}^{m} y^{(i)} log\left(y\left(x^{(i)}\right)\right)$
$+\left(1-y^{(i)}\right) log\left(1-y\left(x^{(i)}\right)\right)$
using the Gradient Descent algorithm on the negative loglikelihood function:
$\theta \leftarrow \theta - \lambda \nabla_\theta (-\mathcal{L}(\theta))$
We chose this model because of its simplicity and also because it outputs probabilities very naturally. Our logistic regression takes as parameters a regularization type (i.e. L1 or L2) and a regularization strength that we tuned thanks to Grid Search.

4.3 Support Vector Machine

The second model we chose was Support Vector Machine the reason being that SVMs can learn non-linear separators in our feature space via the use of kernels. The idea of SVM is simple: the algorithm creates a line or a hyperplane which separates the data into classes.Given a training set $S = \left\{\left(x^{(i)}, y^{(i)}\right); I = 1, \ldots, m\right\}$, it seems that a natural desideratum is to try to find a decision boundary that maximizes the margin γ, since this would reflect a very confident set of predictions on the training set and a good "fit" to the training data. Specifically, this will result in a classifier that separates the positive and the negative training examples with a "gap". If $y^{(i)} = 1$, then for the margin to be large (i.e., for our prediction to be confident and correct), we need $w^T x^{(i)} + b$ to be a large positive number. Conversely, if $y^{(i)} = -1$, then for the margin to be large, we need $w^T x^{(i)} + b$ to be a large negative number. Moreover, if $y^{(i)}\left(w^T x^{(i)} + b\right) > 0$, then our prediction on this example is correct. Figure 5, from [7], illustrates well our wish.

The optimization problem comes down to:

$$\min_{w,b} \frac{1}{2}||w||^2$$
$$\text{s.t} \quad y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, i = 1, ..., m$$

Above, the data is assumed to be linearly separable. However, all data is not linearly separable. That's when the kernel trick comes into play. The

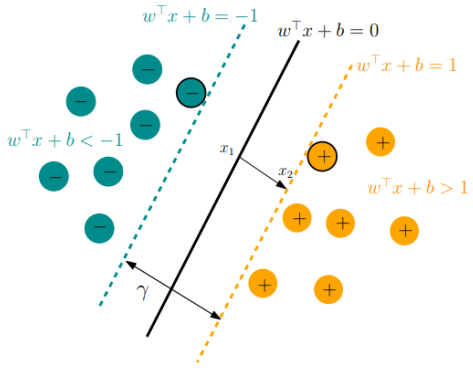


Figure 5: Support vector classifier.

dimensionality of the data is increased so it can be linearly separable. Specifically, we applied the Linear, Radial Basis Function and the Polynomial kernels for comparison. The hyperparameters we tuned are the regularization strength, the degree (specific to the polynomial kernel) and gamma. One drawback is that SVMs take substantially longer to train.

5 Results and discussion

We trained our models in multiple configurations. For each model we aimed at several goals differing by the period by which we want to know whether a drug is going to be approved. As metrics, we evaluated the following ones:

Accuracy (A): Accuracy gives a measure of the correctness of our predictions over all the dataset.

$$A = \frac{tp+tn}{tp+tn+fp+fn}$$

where t, n, p and f mean respectively true, negative, positive and false.

Precision (P): Precision is the ratio of the truly positive labels predicted by the models over all the predicted positive labels.

$$P = \frac{tp}{tp+fp}$$

Recall (R): Recall is the ratio of the truly positive labels predicted by the model over the real number of positive labels in the dataset.

$$R = \frac{tp}{tp+fn}$$

ROC AUC (RA): ROC AUC plots TPR against FPR. It measures the overall performance of a model.

$$TPR = \frac{tp}{tp+fn}, FPR = \frac{fp}{fp+tn}$$

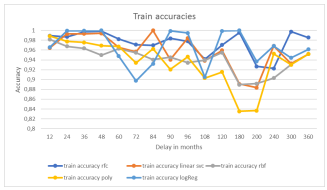
Specificity (S): Specificity is the ratio between the correctly predicted negative labels and the real number of negative labels in the dataset.

$$S = \frac{tn}{tn+fp} = 1 - FPR$$

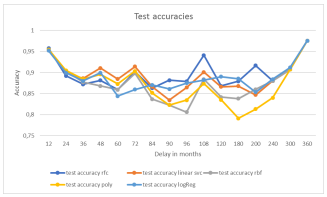
Overall, as we see in the figures below, we have very good results. However, some results are biased since the classes are imbalanced for some periods. In fact, in the extreme sides of the periods, we observe either the best results or the worst ones. It is due to the fact that there are few examples of one of the classes at those places. Therefore, either the models correctly classify those examples and then achieve a good classification, or they wrongly classify them and then they perform poorly. As those few examples on either of the classes are not representative, we cannot really trust those performances. For a better understanding, let's give an example. Let's say we have 100 drugs and only 5 of them are really approved by the FDA. Let's also say our model is very bad and predicts every case as not approved. In doing so, it has classified those 95 non-approved drugs correctly and 5 truly approved drugs wrongly as non-approved. Now even though the model is terrible at predicting the approval, the accuracy of such a bad model is 95%. We definitely want to avoid these kinds of situations. To deal with this issue, we will principally focus on the periods around 180 months where the classes are pretty well balanced. We must however recall that there is one main drawback when proceeding that way; the size of our dataset is cut to around 204 examples which is not that much. In the sequel, for each metric we first display the performances of our models averaged over all the configurations listed above for the testing set as well as the training set. Then we present the performances of the different models for the period of 180 months on the testing set only this time.

Accuracy

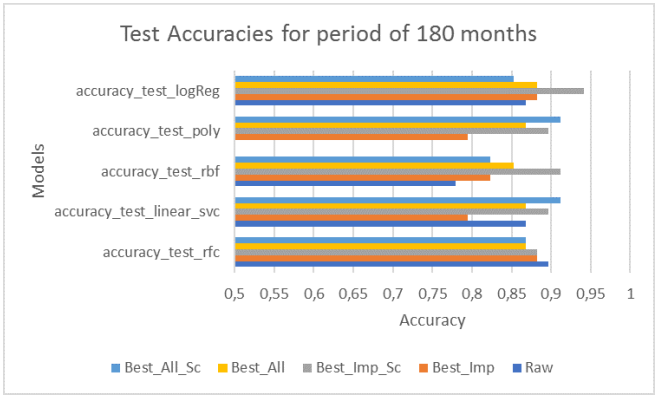
Logistic regression and random forest are the most on top on average (among all configurations). But logistic regression has a higher accuracy of 94.1%



(a) Average training set accuracy over the period set



(b) Average testing set accuracy over the period set

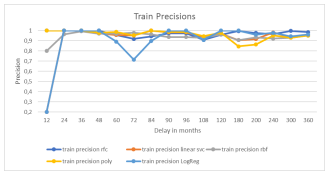


(c) Testing set accuracy at 180 months

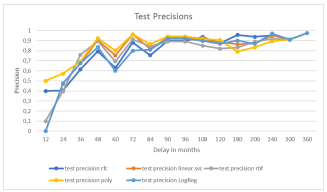
Figure 6: Accuracy results.

when the parameters are tuned and the features scaled.

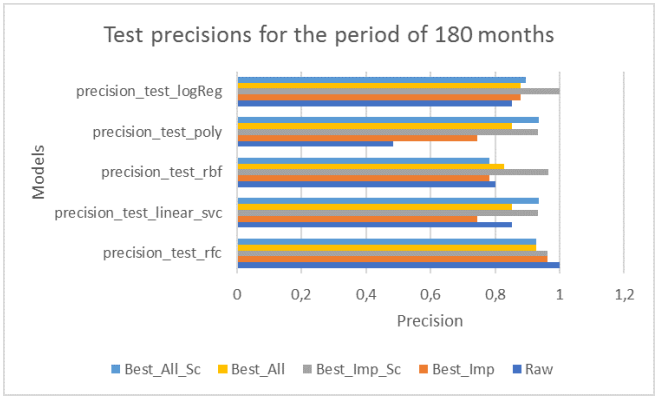
Precision



(a) Average training set precision over the period set



(b) Average testing set precision over the period set

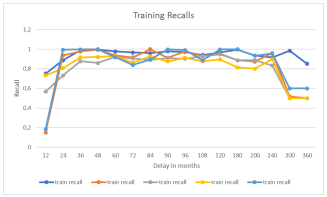


(c) Testing set precision at 180 months

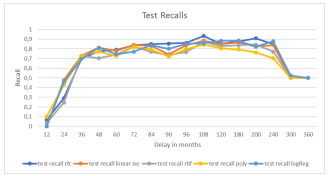
Figure 7: Precision results.

Here, random forest outperforms the other models on average with a precision of 95.7%. At 180 months, it has a precision of 100% on the testing set with the raw model, which is at the same time great and surprising. Note that logistic regression has also a precision of 100% when tuned and with scaled data.

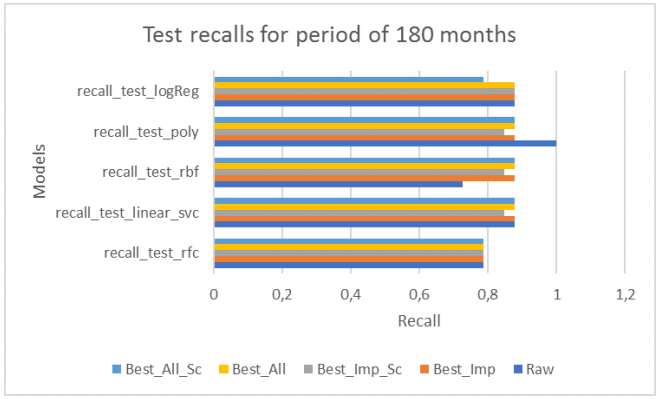
Recall



(a) Average training set recall over the period set



(b) Average testing set recall over the period set



(c) Testing set recall at 180 months

Figure 8: Recall results.

As previously, random forest and logistic regression have the highest recalls on average. However, at 180 months, random forest performs poorly compared to the other models. We note a recall of 100% for raw support vector classifier with polynomial kernel that performs even better than the when it is tuned.

ROC AUC

When it comes to ROC AUC metric, logistic regression and random forest are most often above the other models around 180 months. For ROC AUC metric, logistic regression gives a value of 88,5% when we take the average of all the configurations, while random forest gives a value of 87.7%. At 180 months, logistic regression outperforms the other models with a ROC AUC of 93.9% when it is tuned and the data is scaled.

Specificity

Once again, random forest outperforms the other models in term of specificity with an average over

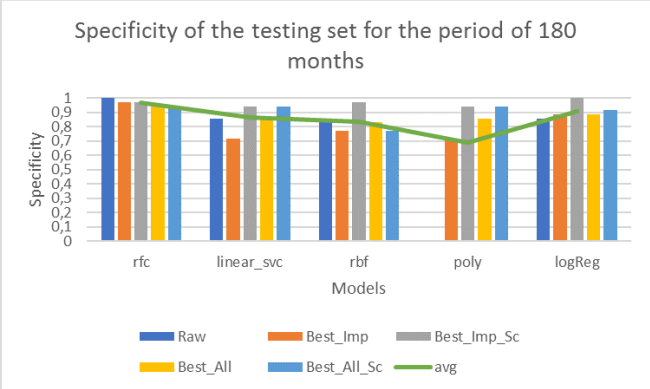
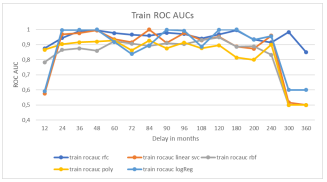
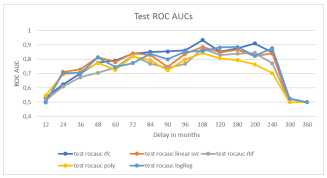


Figure 9: Testing set specificity at 180 months.

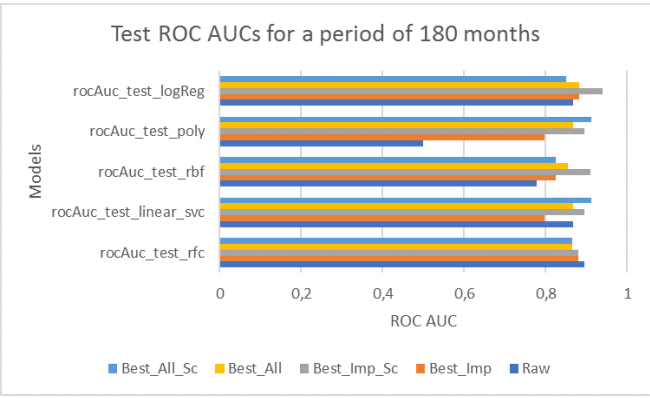
all the configurations of 96.6%. Without tuning it achieves a score of 100% which is once again a little bit unexpected. But when random forest is tuned, it yields a specificity up to 97.1% which is rather spectacular.



(a) Average training set roc auc over the period set



(b) Average testing set roc auc over the period set



(c) Testing set roc auc at 180 months

Figure 10: ROC AUC results.

6 Conclusion and future work

Obviously, the training performances are better than the testing ones. But the difference is quite decent most of the times. This means that we managed well to avoid overfitting. This is due to the hyperparameter tuning. This is confirmed by the ROC AUC measures of our testing set which gives an overview of the global performance of a model. Besides, the plots show that after tuning, the models perform globally better. We think that the most important metrics in our case are ROC AUC and specificity. ROC AUC because it is a good measure of the overall performance of a model and specificity because one of the goal of predicting whether a drug will be approved is to save money and time for the industrials. So, we want to avoid false positive predictions. In other words, we want, partly, a model that maximizes the specificity of the data. If we had more time, we could work further on this project. We would try to impute some missing data thanks to K-Nearest Neighbors algorithm. We would try to collect more balanced data so we can have more robust measures. We would also consider the cox model to make more complex predictions than what we have made so far.

7 Contribution

There were two of us, Alassane Mamadou Watt and Mohamed Amine Kessa, who did this project, but we did have oversight from Loic Verlingue and Guillaume Beinse from the Institut Gustave Roussy in Paris (France). Their contribution was essentially that of supervising our work and giving us some suggestions.

References

[1] J. A. DiMasi, J. C. Hermann, K. Twyman, R. K. Kondru, S. Stergiopoulos, K. A. Getz, and W. Rackoff. A tool for predicting regulatory approval after phase ii testing of new oncology compounds. *Clinical Pharmacology & Therapeutics*, 98(5):506–513, 2015.

[2] R. H. El-Maraghi and E. A. Eisenhauer. eview of phase ii trial designs used in studies of molecular targeted agents: outcomes and predictors of success in phase iii. *Journal of Clinical Oncology*, 26(8):1346–1354, 2008.

[3] J. Goffin, S. Baral, D. Tu, D. Nomikos, and L. Seymour. Objective responses in patients with malignant melanoma or renal cell cancer in early clinical studies do not predict regulatory

approval. *Clinical Cancer Research*, 11(6):5928–5934, 2005.

[4] D. L. Jardim, E. S. Groves, P. P. Breitfeld, and R. Kurzrock. Factors associated with failure of oncology drugs in late-stage clinical development: A systematic review. *Cancer Treatment Reviews*, 52:12–21, 2017.

[5] A. W. Lo, K. W. Siah, and C. H. Wong. Machine-learning models for predicting drug approvals and clinical-phase transitions. Online, July 2017. https://canvas.harvard.edu/courses/37109/files/5464426/.

[6] L. Malik, A. Mejia, H. Parsons, B. Ehler, D. Mahalingam, A. Brenner, J. Sarantopoulos, and S. Weitman. Predicting success in regulatory approval from phase i results. *Cancer Chemotherapy And Pharmacology*, 74(5):1099–1103, 2014.

[7] F. Malliaros. Ma2823: Introduction to machine learning, 2018. https://piazza.com/centralesupelec/fall2018/ma2823/resources.

[8] I. Reinstein. Random forests(r), explained. Online, October 2017. https://www.kdnuggets.com/2017/10/random-forests-explained.html.