A Machine Learning Driven Approach to Predicting the Effects of Drugs on Protein Expression

Kedar Chintalapati

Author email: kedarchin@gmail.com

Abstract

With growing computational abilities, machine learning is gaining more applicability in drug discovery, but the entire process can take over 12 years and cost ~2.6 billion dollars, rendering it intensive in both time and physical resources. One of the main steps in non-clinical development is hit identification and discovery, where drug molecules with desired efficacies are identified. *In silico* modeling of changes in protein expression resulting from drugs could greatly speed up the hit identification process, as changes in protein expression determine the outcomes and thus the effectiveness of the drugs. In this study, a framework has been developed and validated to predict the impacts of drug molecules on certain proteins' expressions through a machine learning approach. Using of three distinct molecular featurization techniques – molecular fingerprints and numerical properties extracted using two different libraries – several machine learning models were trained to predict the impacts of drugs on the protein expression of Caspase-3. The best result in this study was from a convolutional neural network (CNN) trained on molecular fingerprints data, with some other algorithms and featurization-methods also getting similar results; the CNN model achieved an 86.36% accuracy on balanced (oversampled minority class), SMOTE-augmented test data, and a 77.50% accuracy on the original imbalanced data, showing that effective generalizations must have been made between drugs and protein expression. This framework can be applied to the early stages of drug discovery to develop more machine learning models on more proteins and use them to speed up and cheapen the process through their abilities to generalize on the relations between drug molecules and changes in protein expression.

Introduction

Drug discovery/Hit identification

Finding new drugs is a complex problem with a low success rate [1], as highlighted in a drug discovery pipeline in Figure 1. The entire process can take an estimated 12 years and cost ~2.6 billion dollars [2], with only 250 compounds out of 5,000 to 10,000 drug candidates making it to preclinical testing, and only 5 compounds making it to the clinical phase [3]. Among the drug discovery process is hit identification and discovery, which involves finding and validating molecules that result in desired therapeutic effects [4]. Existing approaches are mainly based on intuition and expert driven identification of drug candidates, which does not scale efficiently when

screening 10,000+ candidates. Moreover, this step is limited by availability or synthesis of potential candidates, and time and cost it takes to access and validate them in the lab. If a certain proposed drug candidate (i.e., through computational tools) is not commercially available, then it is often discarded unless the proposed tools or algorithms achieve high predictive accuracy in the success of the drug candidate.

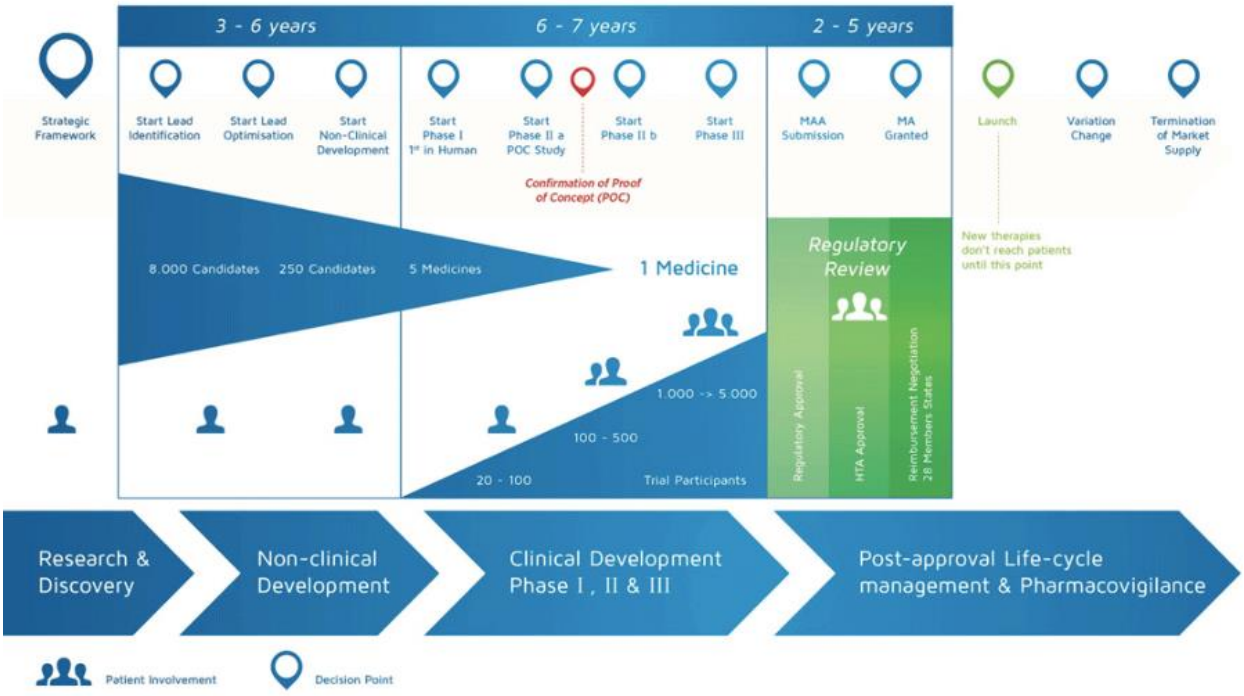

Figure 1. Representation of a drug discovery pipeline and the success rate. The image was adopted from ref [3].

In order to improve the efficiency of the hit discovery step, certain methods have been utilized in academic research and in industry. For instance, Lipinski's Rule-of-Five serves as a rule-of-thumb in preparing the initial list of drug candidates to be screened. Lipinski's Rule-of-Five states that an ideal drug candidate would have less than 5 H-bond donors, 10 H-bond acceptors, a molecular weight of less than 500, and a calculated LogP less than 5 [5]**.** However, for more data-driven approaches, screening simply based on five conditions may not be efficient. Moreover, for a smarter design of drugs using ML algorithms, drug candidates (i.e., chemical structures or IDs) need to be expressed in a machine-readable format in the form of chemical descriptors (e.g., molecular weight, boiling point, solubility). However, it is often challenging to identify what molecular descriptors are the most relevant in predicting drug activities.

In addition to chemical properties of drug candidates, there are only 324 [6] targets for all the FDA approved drugs, highlighting the challenge of (1) identifying drug targets and (2) predicting ligand (i.e., drug candidate) and target interactions such as binding energy (i.e., docking score). Moreover,

a drug candidate could interact with dozens of proteins, posing a great challenge in designing a selective drug candidate. As a result of these complexities, the majority of drug candidates fail, and it is difficult to entirely track and quantify their interactions with target proteins. One of many variables in tracking the influences of drug molecules in the central dogma is protein expression. Predicting impacts of a drug candidate on protein expression is an important challenge in identifying hit molecules because they influence the therapeutic effects of such proteins, and thus a drug should achieve desired protein expression (i.e., increasing protein quantity or, in some cases, inhibiting proteins to deactivate the protein activity) to achieve desired therapeutic effects.

Protein Expression

In this project, we studied the impact of drug candidates on Caspase-3 (CASP3) expression, which is an important protein involved in cell apoptosis [7], making it highly relevant to tumor-development, which involves lack of cell death. As a result, higher levels of activated CASP3 correlate to increased rates of recurrences and deaths in cancer patients [8]. Furthermore, cleaved Caspase 3 expression could predict prostate cancer biochemical progression [9].

ML algorithms

Machine learning models can primarily be divided into two categories – supervised and unsupervised ML. In supervised machine learning, algorithms learn from a labeled data where algorithms are trained on an input set (x_1, x_2, …, xN) to map it to the outputs (i.e., target values). This means that they are essentially trained to make generalizations on the trained dataset such that they can learn a computational method to map such inputs to outputs, even with samples outside of the dataset they were trained on (i.e., extrapolation). To do so, supervised ML algorithms must specifically be trained on datasets that contain both features – input data – and labels – output data. Some commonly used supervised ML algorithms include linear regression, logistic regression (LR), support vector machines (SVMs), decision trees (DTs), and neural networks (NNs). Supervised models learn through repeatedly making predictions on the samples of the dataset and being gradually tweaked to minimize the errors in their predictions. In unsupervised machine learning, unlike with supervised learning, there are not labels, rather, there are only data with features. In various ways such as clustering, unsupervised algorithms learn to find patterns in the data rather than mapping inputs to outputs. There is no supervision to correct these models. Some commonly used unsupervised learning algorithms include K-means, anomaly detection, and principal component analysis (PCA).

Methods and Materials

Data Processing
Data Collection
The Chemical-gene Interactions dataset used in this project was collected from the Comparative Toxicogenomics Database (CTD) [10]. The dataset contains the following eleven columns: ChemicalName, ChemicalID, (MeSH identifier) CasRN (CAS Registry Number, if available), GeneSymbol, GeneID (NCBI Gene identifier), GeneForms, Organism (scientific name), OrganismID (NCBI Taxonomy identifier), Interaction, InteractionActions, and PubMedIDs.

Data with the CASP3 protein was extracted for modeling given its significance in research [7, 8, 9] as elaborated in depth in the Introduction section, and due to its best availability of data points after cleaning (i.e., ~6,000 of initial points, and ~400 after removing the duplicates). The CAS Registry Number of the drug molecule, the protein source (e.g., Homo Sapiens), and the target (i.e., influence on the protein expression) were selected in the final dataset.

Data Cleaning
Cleaning of the dataset involved removal of rows with empty values for any of the features, and removal of duplicates. Since there were more variables in the original dataset than just the drugs, there were significant numbers of duplicates in the extracted data. Furthermore, there were samples of duplicates with the same features but different labels as a result. This issue was approached through a voting system, where the label – which was either a decrease or an increase in protein expression – that had the highest number of instances among the same features, would be selected. For example, if the same drug candidate was reported six times in the dataset, with four instances being labeled with increased expression and two with decreased expression, then the label would be interpreted as increased expression.

Featurization
Three types of chemical features were generated for the drugs to convert the Drug IDs into SMILES representation and then into molecular descriptors. These three types of features were molecular fingerprints, all the numerical properties of the molecules available from the RDKit library [11], and all the numerical properties available from the PubChemPy library [12]. The reader can refer to the cited references for the full lists of calculated features.

The molecular fingerprints (Morgan fingerprints with radius=2, which is roughly equal to ECFP4) were matrices, with the dimensions 1x2048, within which each value was either 1 or 0 depending on the presence of certain substructures. The dimensions for the RDKit- and PubChemPy-extracted descriptors were 1x43 and 1x33, respectively. Along with this, except for when training the Convolutional Neural Network, all the extracted feature types included an extra column for the protein's original organism. The organisms were all numerically encoded. The labels – which were

under the “InteractionActions” column, were labeled either 0 or 1, based on whether protein expression decreased or increased, respectively.

Feature engineering

The original dataset was imbalanced in favor of increased expression. Specifically, after cleaning, ~87% of the samples had the label “1,” while ~13% had the label “0.” This means that a hypothetical naïve model that simply always predicts “1” would result in 87% accuracy on the data. Given supervised machine learning algorithms learn data by minimizing the error on the predicted values versus actual values, an imbalanced dataset would make it very likely that ML models simply learn to always predict one output – in this case, “1” – like a naïve model, rather than actually learning a pattern to predict the protein expression. This problem was addressed using the SMOTE algorithm to oversample the minority class in the form of generating synthetic data points to be added to the minority class to balance the dataset. The process of implementing SMOTE involved splitting the data into train and test sets with 10% being the test data, except for certain algorithms where 20 or 30% test data size was used. Then, SMOTE was individually applied to each of the train and test sets by using the “Imbalanced-Learn” (imblearn) library [13]. This allowed for the trained models to be tested both on the original data and also on the augmented data with SMOTE.

Algorithm choice

Since the project objective was to predict the impact of drug molecules on protein expression, supervised machine learning algorithms were used. The “inputs” are the drug molecules with associated features of choice, and the “outputs” being the protein expression, and thus an algorithm would need to be trained to map the influence of drug molecules to protein expression. In order to develop an efficient framework, several different ML algorithms were trained and optimized, as opposed to using a single ML algorithm. This is important given different algorithms have different learning efficiency and different scalability over various dataset sizes, distributions, and structures. Therefore, the algorithms used in this project were a Convolutional Neural Network (CNN), and multiple vanilla Artificial Neural Networks (ANN), XGBoost models (XGB), Random Forests (RF), Support Vector Machines with various kernels (SVM), and Logistic Regression models (LR).

Results and Discussion

The results on the original and SMOTE-engineered data are provided in Table 1 and Table 2, comparing three different featurization techniques and six different ML algorithms.

Out of all the models, based on evaluations of performance on the original, unaltered test data, the Convolutional Neural Network, Vanilla Neural Network, XGBoost, and Random Forest models achieved the best performances in terms of prediction accuracy, with SVM and Logistic Regression models also achieving notable performances when trained on molecular fingerprints data.

Table 1. Model Performances on original unaltered data. The values represent cross-validated model performances using Molecular Fingerprints, RDKit descriptors, and PubChemPy descriptors, respectively (e.g., 99.28/99.82/96.34) for drug molecule featurization. SVM (polynomial) degree of 11/200/200.

Model	Train Accuracy [%]	Test Accuracy [%]
CNN	98.03	77.50
ANN	99.44	75
XGB	82.82 / 95.57 / 95.25	65 / 79.75 / 83.52
RF	99.71 / 100 / 100	75 / 70 / 84.78
SVM (Linear)	96.62	70
SVM (RBF)	83.10 / 87.32 / 80.78	72.5 / 82.5 / 67.39
SVM (Polynomial)	55.21 / 87.04 / 40.625	35 / 82.5 / 50
LR	91.55 / 58.59 / 65.23	72.5 / 50 / 50

SVM models with RBF and Polynomial kernels both achieved 82.5% accuracies on the test data when RDKit descriptors were used. However, these can likely be disregarded because these models resulted in 50% accuracies on the SMOTE-engineered data. The SMOTE-engineered data was perfectly balanced, meaning these models could have always predicted one output. The 82.5% accuracies would have been the result of 82.5% of the unaltered test data being labeled as the same one output that these SVM models always predicted. This suggests that these models did not learn the underlying patterns in the data, but rather simply performed similar to hypothetical naïve models.

The 84.78% accuracy on the unaltered data using the Random Forest algorithm can also likely be disregarded for similar reasons to the seemingly high-performing SVMs; that is, the model resulted in 100% accuracies on both the SMOTE-engineered and unaltered train datasets, but it achieved a performance of 63.75% accuracy on the SMOTE-engineered test data. This means that the model overfitted – it only fit on the train data by mapping all the inputs to the outputs, but it did not make real, applicable generalizations about the data, resulting in the significantly lower accuracy on the SMOTE-engineered test data. If the model does not generalize even on the synthetic data, then the model's results on the unaltered data could not be reliably used for real-life predictions**.**

Table 2. Model Performances on SMOTE-engineered Data. The values represent cross-validated model performances using Molecular Fingerprints, RDKit descriptors, and PubChemPy descriptors, respectively (e.g., 99.28/99.82/96.34) for drug molecule featurization. SVM (polynomial) degree of 11/200/200.

Model	Train Accuracy [%]	Test Accuracy [%]
CNN	97.66	86.36
ANN	99.28 / 99.82 / 96.34	84.85 / 85.51
XGB	90.13 / 97.45 / 97.16	78.79 / 72.49 / 81.88
RF	99.84 / 100 / 100	84.84 / 60.61 / 63.75
SVM (Linear)	98.06	81.82
SVM (RBF)	89.00 / 51.78 / 60.65	81.82 / 50 / 41.25
SVM (Polynomial)	74.27 / 50 / 45.14	60.61 / 50 / 60
LR	95.15 / 63.11 / 71.06	83.33 / 56.06 / 28.75

Especially notable is the models' performances on the (partly) synthetic data. Unlike with the original data, since the synthetic data included oversampling of the minority class, the class balance was equal – 50% of the data were labeled each of the two classes. This meant that the benchmark accuracy would be only 50%, rather than 82.5%, given a naïve model that always predicted one of the classes would only achieve 50% accuracy.

Given the equally balanced dataset, many of the algorithms achieved higher than 80% prediction accuracy on the SMOTE-engineered test data, with the Convolutional Neural Network that was

trained on the Molecular Fingerprints data performing the best with an 86.36% accuracy. The 86.36% accuracy is especially promising because not only was the model not exposed to the data, but also that data was balanced. This performance shows that the models clearly identified some underlying patterns in the data that included synthetic minority-class samples. SMOTE is used in real-life applications as an algorithm that allows for the creation of synthetic data based on existing samples, meaning that the generalizations the models learned to make on the augmented data can likely be applicable and relevant to real data. This also suggests that there likely are real patterns in drug molecules that determine their impacts on protein expression.

Overall, the results had varied accuracies between 75% and 83.52% on the unaltered test data. Based on two key reasons – performance on unaltered test data, as explained above, and performance on SMOTE-engineered synthetic data – it can be determined that the algorithms implemented in this project have found significant results with real applications. For the unaltered test data, although 75% to 83.52% accuracies are technically lower than or similar to the 82.5% baseline accuracy that a hypothetical naïve model could achieve, the results are still significant given the models clearly learned to make generalizations that worked well on the synthetic data. This means that the models did not perform like naïve models, and thus their accuracies on the unaltered test data were authentic; this makes it evident that the models do have a potential for effectiveness in the real-world applications.

Model Comparison

Overall, the Neural Network, XGBoost, and Random Forest models performed significantly better than the SVM and Logistic Regression models, although SVM and LR models had some notable performance on Molecular Fingerprints data. This can likely be explained by the complexities of the models. Neural Networks were the most complex models evaluated, as the input variables undergo multiple nonlinear transformations, with the many neurons across multiple layers allowing for the deep learning algorithm to have the highest capability to find the most complex patterns in the data. Both Random Forest and XGBoost are tree-based algorithms, but, in summary, XGBoost is a gradient boosting algorithm, while Random Forest is a bagging algorithm [14, 15]**.** They both involve ensembles of decision trees, which allows them to also have higher capabilities of fitting complex data. Support Vector Machines and Logistic Regression – although they can be powerful – are simpler and thus they are more likely to underperform, resulting in the algorithms struggling to find patterns on complex molecular problems.

Featurization Comparison

Overall, the models performed the best on the Molecular Fingerprints data. For Molecular Fingerprints, the average model's performance was 80.30% on SMOTE-engineered test data and 67.81% on the unaltered test data. The average model's performances on the SMOTE-engineered data and the unaltered test data, respectively, were 62.45% and 72.95% for the RDKit-extracted-numerical-features, and 55.13% and 67.14% for the PubChemPy-extracted numerical-features.

The averages on the unaltered data for RDKit and PubChemPy are inflated by the models that achieved ~82.5% accuracies by simply almost always predicting the same output.

The models performed the best on Molecular Fingerprints data, probably due to representing structural molecular information through the fragments, thus including vital data about the drugs' substructure-compositions. This was not true for the numerical data extracted using RDKit and PubChemPy. However, given they only included certain properties of the molecules, such as LogP, these featurization techniques do not give as much information about the molecular structure. The models likely did better on the data extracted using RDKit because it provided more numerical properties and thus more information than PubChemPy – 43 for RDKit and 33 for PubChemPy.

Conclusion

Overall, through evaluating several different ML models and using three different feature extraction methods, this project demonstrated the possibility of predicting drug effectiveness on protein expressions using machine learning. Specifically, it was found that ML algorithms, especially deep learning algorithms, have a real and promising applicative potential in making such predictions. Precisely, the best performance was achieved using a Convolutional Neural Network with an 86.36% accuracy on predicting the test data, when partly synthetic data was used for a balanced data generation using SMOTE. Having a larger and balanced dataset allows for potential use of the developed framework in this project in real-life applications. This can be potentially replicated for different proteins where large experimental datasets are also available. All three methods of featurization techniques were considerably high-level, meaning both numerical properties and molecular fingerprints may not provide all relevant descriptors that play a role in determining protein expression. Along with working on more balanced data, this project could be enhanced by using quantum mechanical methods instead to get more descriptive features of the molecules. One such example would be with using the computationally demanding Density Functional Theory (DFT), which is used for electron density calculations, and can have various applications in drug modeling.

By discovering that drugs' impacts on protein expressions are predictable, this project offers a framework for potentially expanding on *in silico* drug discovery and thus helping to reduce the need of physical experimentation, which can often be far more cost-, resource-, and time-intensive. Overall, achieving this task computationally is significantly faster and cheaper than attempting it through physical experimentation methods only, and this approach offers a complementary tool to narrow down the numbers of experiments needed.

References

1. Altevogt, B. M., Davis, M., Pankevich, D. E., & Norris, S. M. P. (Eds.). (2014). *Improving and accelerating therapeutic development for nervous system disorders: workshop summary*. National Academies Press.
2. Chan, H. S., Shan, H., Dahoun, T., Vogel, H., & Yuan, S. (2019). Advancing drug discovery via artificial intelligence. *Trends in pharmacological sciences*, *40*(8), 592-604.
3. *Making a Medicine. step 1: Pre-discovery*. EUPATI Toolbox. (2021, April 16). Retrieved August 24, 2022, from https://toolbox.eupati.eu/resources/making-a-medicine-step-1-pre-discovery/.
4. Hughes, J. P., Rees, S., Kalindjian, S. B., & Philpott, K. L. (2011). Principles of early drug discovery. *British journal of pharmacology*, *162*(6), 1239-1249.
5. Benet, Leslie Z., Chelsea M. Hosey, Oleg Ursu, and Tudor I. Oprea. "BDDCS, the Rule of 5 and drugability." *Advanced drug delivery reviews* 101 (2016): 89-98.
6. Overington, J. P., Al-Lazikani, B., & Hopkins, A. L. (2006). How many drug targets are there?. *Nature reviews Drug discovery*, *5*(12), 993-996.
7. Porter, A. G., & Jänicke, R. U. (1999). Emerging roles of caspase-3 in apoptosis. *Cell death & differentiation*, *6*(2), 99-104.
8. Sharma, A., Boise, L. H., & Shanmugam, M. (2019). Cancer metabolism and the evasion of apoptotic cell death. *Cancers*, *11*(8), 1144.
9. Huang, Q., Li, F., Liu, X., Li, W., Shi, W., Liu, F. F., ... & Li, C. Y. (2011). Caspase 3–mediated stimulation of tumor cell repopulation during cancer radiotherapy. *Nature medicine*, *17*(7), 860-866.
10. *Illuminating How Chemicals Affect Human Health. Comparative Toxicogenomics Database*. CTD. (n.d.). Retrieved July 28, 2022, from http://ctdbase.org/downloads/.
11. *Getting Started with the RDKit in Python*. Getting Started with the RDKit in Python - The RDKit 2022.03.1 Documentation. (n.d.). Retrieved August 15, 2022, from https://www.rdkit.org/docs/GettingStartedInPython.html#list-of-available-descriptors.
12. *Properties*. PubChemPy 1.0.4 documentation. (n.d.). Retrieved August 15, 2022, from https://pubchempy.readthedocs.io/en/latest/guide/properties.html.
13. *Imbalanced-Learn Documentation*. Imbalanced-Learn. (n.d.). Retrieved August 15, 2022, from https://imbalanced-learn.org/stable/.
14. Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).
15. Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

Appendix

More Information

Code used for this study, hyperparameters of models, other metrics of models (e.g., precision and recall), and other information are all available upon request. Please contact the author for any questions or requests.