Ensemble Machine Learning Approach for Classifying hERG Channel Blockers

Hanson N. Hoang[a,*]
[a]Department of Bioengineering and Therapeutic Sciences, University of California San Francisco, 550 16th St, USA
*Correspondence: Hanson Hoang (hanson.hoang@ucsf.edu)

hERG channel inhibition is a critical concern in drug safety, as it can lead to severe cardiac complications. Traditionally, in vitro assays have been used to identify potential hERG blockers, but these methods are both time-consuming and expensive. To address these limitations, in silico models offer a promising alternative, leveraging computational techniques to predict hERG blocking activity based on molecular structures. Machine learning approaches were applied to predict hERG channel blocking activity using the hERG_Karim dataset from the Toxicology Data Challenge (TDC), which contains SMILES strings representing molecular structures. These strings were converted into molecular fingerprints and used as input features for our models. Comparison between two modeling approaches, a Random Forest model and an ensemble model consisting of Gradient Boosting, Random Forest, Logistic Regression, Extra Trees, and K-Nearest Neighbors, was conducted. The Random Forest model achieved an accuracy of 84.83% and an AUC-ROC of 0.9207 on the test set, while the ensemble model reached an accuracy of 84.19% and an AUC-ROC of 0.9186. Despite slight differences in performance, both models demonstrated strong predictive capabilities. The Random Forest model showed higher precision for non-blockers and recall for blockers, while the ensemble model exhibited more balanced precision and recall across both classes. Despite showcasing more balanced percision/recall, by being having less false positives and negatives, the Random Forest model outperforms the ensemble model in identifying potential hERG blockers.

Introduction

In the field of drug development, one of the critical factors in determining the safety and viability of a new pharmaceutical compound is its potential to interact with key biological targets. In this paper, the target I'm focusing on is the hERG (human Ether-à-go-go-Related Gene) channel, a vital potassium ion channel responsible for regulating the electrical activity of the heart. Disruptions in the functioning of the hERG channel, specifically through the inhibition or blockage of its activity, can lead to severe cardiac complications, including long QT syndrome, arrhythmias, and in extreme cases, sudden cardiac death. As a result, accurately predicting whether a drug is a hERG channel blocker has become a crucial step in the early stages of drug development. This prediction is essential not only for identifying potentially harmful drugs but also for avoiding costly late-stage clinical trial failures. Therefore, early identification of drug candidates that might act as hERG blockers is a high priority in pharmaceutical development, as it helps to avoid adverse drug reactions that could compromise patient safety.

Traditionally, hERG channel activity has been studied through in vitro assays, which are time-consuming, expensive, and sometimes unreliable. To address these challenges, in silico models have emerged as a promising alternative. These models use computational approaches to predict whether a compound will block the hERG channel based on its molecular structure. One common method for generating predictive models involves the use of machine learning algorithms, which can analyze large datasets of molecular features and learn to classify compounds as hERG blockers or non-blockers.

The hERG_Karim dataset, used in this project, is part of the Toxicology Data Challenge (TDC), a public initiative aimed at developing predictive models for drug safety. This dataset contains molecular data represented as SMILES (Simplified Molecular Input Line Entry System) strings, which encode the 2D structure of molecules. These SMILES strings are then used to generate molecular fingerprints, which are numerical representations of the molecular structure that serve as inputs to the machine learning models. Each compound in the dataset is labeled as either a hERG channel blocker or a non-blocker, essentially creating a binary classification target.

The ability to predict hERG blocking activity in silico is advantageous in that it allows for faster screening of drug candidates, saving both time and resources. By identifying potential hERG blockers early in the drug development process, pharmaceutical companies can eliminate harmful compounds before they reach clinical trials, thereby reducing the risk of adverse events and improving the overall safety profile of new drugs.

In this paper, I aim to develop and evaluate a machine learning model that classifies drug compounds as hERG blockers or non-blockers using the hERG_Karim dataset. My approach involves using an ensemble learning method that combines multiple classifiers, including Gradient Boosting, Random Forest, Logistic Regression, Extra Trees, and K Neighbors. The ensemble approach leverages the strengths of different models to improve overall performance and robustness. In addition to using these classifiers, I also perform hyperparameter tuning to optimize the ensemble model's performance using RandomizedSearchCV in order to find the best parameters for each classifier.

The rationale for choosing an ensemble method is inspired by the original paper the dataset is from by Karim (https://doi.org/10.1186/s13321-021-00541-z), which utilizes deep learning models within its ensemble model to create a robust predictor for the hERG channel. The only two machine learning models similar to the paper I will be using are Random Forest and Gradient Boosting. Inspiration aside, Random Forest model is known for its robustness and ability to handle overfitting, so it may not always capture the complex relationships between features as effectively as a Gradient Boosting model, which is adept at optimizing decision boundaries. On the other hand, Logistic Regression, though a simpler model, provides insights into the relationship between molecular features and the target class, making it a useful comparison, albeit not as complex as the other two mentioned. Extra Trees and K Neighbors were chosen out of pure interest in how Extra Trees is similar to Random Forest and K Neighbors was touched upon in class. By combining the predictions of

multiple classifiers, I expect to achieve a more accurate and reliable prediction than using any single model alone.

By focusing on these algorithms and using ensemble methods, this project aims to develop a model that can predict whether a drug is a hERG channel blocker with high accuracy.

Methods

GitHub Repo

https://github.com/h5hoang/CDD-203-Final-Project

Dataset Description

The dataset used for this project is the hERG_Karim dataset, which is part of the Toxicology Data Challenge (TDC). The dataset consists of SMILES (Simplified Molecular Input Line Entry System) representations of molecules, a widely adopted format for encoding the 2D structure of chemical compounds into text strings. These SMILES strings are transformed into molecular fingerprints, numerical vectors representing the chemical features of each compound. The dataset contains labels indicating whether each compound is a hERG channel blocker (class 1) or a non-blocker (class 0).

The hERG_Karim dataset includes over 13,445 samples, each consisting of a SMILES string and its corresponding class label. The dataset is divided into three sets: a training set, validation set, and test set. The training set is used to train the machine learning model, the validation is used to evaluate and improve a machine learning model during training, and finally the test set is used to evaluate the performance of the model on unseen data or in this case, the test set.

Before training the model, the dataset for both my group's model and my individual model was preprocessed to ensure the features were suitable for machine learning algorithms. This involved the following steps:

1. **Feature Extraction**: The SMILES strings were processed to generate molecular fingerprints, which were then used as the input features for the model. These fingerprints are high-dimensional binary vectors that represent the presence or absence of specific substructures within the molecule.
2. **Train-Validation-Test Split**: The dataset was split into a training set (70% of the data), a validation set (10% of the data), and a test set (20% of the data).

Group Model: Random Forest Classifier

For my group's model, a **Random Forest Classifier** was used. The rationale behind our model was given the dataset, we wanted to find the most optimal model that required the least amount of finetuning. With no hyperparameters tweaked, Random Forest outperformed Logistic Regression, SVM, and Gradient boost, which made us pick Random Forest over the other 3. In addition, we also looked at runtime too, SVM had comparable performance scores compared to Random Forest, but took over 30 minutes to run in comparison to Random Forest's 1–2-minute runtime. Essentially, our model selection boiled down to inputting random ML models we found interesting, seeing which ones performed the best and fastest, and choosing said model, which was Random Forest, to move forward in with our hypertuning phase.

The random forest classifier was initialized with the following parameters:

- random_state=42: This ensures that the results can be replicated each time the model is run.
- n_estimators=500: This sets the number of decision trees in the forest to 500, which was chosen based on empirical testing to provide good results.

Individual Model: Ensemble Classifier

I utilized an ensemble approach to model the classification task, combining five individual models: Gradient Boosting, Random Forest, Logistic Regression, Extra Trees, and K-Nearest Neighbors. As mentioned in the introduction, this ensemble approach was selected to capitalize on the diverse strengths of these models while taking inspiration in Karim's paper in developing CardioTox using an ensemble model, allowing for improved accuracy and generalization compared to using a single model. The ensemble was implemented using a VotingClassifier, which combines predictions from multiple models and outputs the class label based on the majority, which is the soft voting approach, which considers the predicted probabilities.

Each individual model in the ensemble was selected based on its unique strengths and characteristics, where when combined, would ideally result in a more comprehensitve and accurate overall model. Here's a breakdown of why I chose each model and what it contributes to the ensemble:

Gradient Boosting Classifier:
Like Random forest, it's an ensemble-based algorithm that builds trees sequentially, with each new tree correcting errors made by previous trees with a learning_rate. This iterative approach helps it capture complex patterns in the data, especially in situations where interactions between features are non-linear.

Random Forest Classifier:
Random Forest was selected in our group portion and outperformed a our selected few models, which is why I wanted to keep Random Forest in the ensemble model. It differs from Gradient Boosting in that it builds multiple decision trees independently and aggregates their outputs. This "bagging" approach helps to reduce variance and improve model stability. In addition, Random Forest is known for its robustness and ability to handle high-dimensional datasets with minimal tuning.

Logistic Regression
Logistic Regression is a simple and interpretable linear model that performs well when the relationship between the features and the target is linear or nearly linear. Despite being less powerful in capturing non-linear relationships, it serves as a good baseline for classification problems. Additionally, since it models probabilities directly, it enhances the ensemble's soft voting mechanism by adding a model that is quick to compute and provides reliable probability estimates.

Extra Trees Classifier
Extra Trees are similar to Random Forest but with more randomness introduced at each split during tree construction. This randomness can make Extra Trees faster to train and can help reduce overfitting, while adding diversity to the ensemble as it makes splits on random subsets of features and data points, which introduces additional randomness. My though process is that this diversity can improve the overall generalization capability by reducing bias and variance, helping the ensemble perform better on unseen data.

K-Nearest Neighbors
KNN is simple to implement and does not make strong assumptions about the data distribution as it captures local patterns in the data, making it particularly effective for problems where class boundaries

are not well-defined globally but can be identified locally. But that aside, this model was added into the mix out of interest.

The decision to combine these five models into an ensemble stems from the fact that each one brings a distinct strength to the table. Each model captures a different aspect of the data, with Gradient Boosting and Random Forest focusing on global patterns and Extra Trees and KNN focusing more on local patterns. Logistic Regression provides a contrast with its linear assumptions. And combining models typically leads to improved generalization compared to individual models. With such a complex dataset with SMILES strings and Morgan Fingerprints, where no single model can capture all the relationships in the data, the ensemble approach, with its "voting" mechanism, helps smooth out any weaknesses of individual models, leading to improved performance overall.

Individual Model: Hyperparameter Tuning

To find the optimal hyperparameters for each individual model, I used RandomizedSearchCV with a search over a range of parameter values. This technique was preferred over GridSearchCV because of the lack of computational resources I have on my system along with the prioritization of runtime. I selected 50 iterations (n_iter=50) to balance computation time (around 1 hour each run) with the quality of the results. The scoring metric used during the search was ROC-AUC, as it is particularly useful for binary classification tasks.

The best parameters obtained from RandomizedSearchCV for each model were:

- **Gradient Boosting**: n_estimators=619, learning_rate=0.1194, max_depth=8
- **Random Forest**: n_estimators=675, max_depth=28
- **Logistic Regression**: C=57.0071
- **Extra Trees**: n_estimators=842, max_depth=22
- **KNN**: n_neighbors=5

Once the individual models were fine-tuned, I combined them into a single ensemble model using VotingClassifier with soft voting. Soft voting aggregates the predicted probabilities of each model, taking a weighted average to make the final prediction. This method was chosen to leverage the strengths of each classifier's probabilistic output, aiming for a better generalization and reduced overfitting compared to hard voting.

The ensemble model was trained on the training data (X_train, y_train), and I used the fit() method to optimize the model's parameters. Once trained, the model was evaluated on the validation set with the new paramaters and finaly the test set to assess its performance.

Model Comparison

My group's Random Forest model is a homogeneous approach, using a single type of classifier with decision trees. This makes it simpler and efficient for many tasks but limits its ability to adapt to diverse data characteristics. In contrast, the ensemble model leverages heterogeneous classifiers (Gradient Boosting, Random Forest, Logistic Regression, Extra Trees, and K-Nearest Neighbors), which collectively address different patterns in the data. For instance, Logistic Regression provides linear separability, while Random Forest and Extra Trees focus on capturing non-linear relationships.

My group's Random Forest model used fixed parameters 500 trees with no addition hyperparameter tuning with GridSearchCV or RandomizedSearchCV. 500 trees was an arbitrary number we chose that performed the best out of the ones we randomly tried, whereas my ensemble model benefited from using **RandomizedSearchCV**. This process ensured that each parameter for each model was optimized for performance before combining them, contributing to its higher effectiveness.

Results

Summary

In this section, I'll discuss the performance of the my individual ensemble model and my group's Random Forest model.

Performance of Group's Model

The random forest classifier was initialized with the following parameters:

- random_state=42: This ensures that the results can be replicated each time the model is run.
- n_estimators=500: This sets the number of decision trees in the forest to 500, which was chosen based on empirical testing to provide good results.

Parameters for selected based on trial and error by manually inputting values in until we believed the model had good performance. The model was trained using the training set and then evaluated on the test set, where predictions were made using the predict() function. The performance was assessed using several classification metrics, including precision, recall, F1-score, accuracy, and AUC-ROC, which are later discussed in the Results section.

With 500 decision trees and an accuracy of 84.83%, the results were as follows:

- **Performance on Test Split:**
  - **Accuracy:** 84.83%
  - **Precision (Class 0):** 0.85
  - **Recall (Class 0):** 0.84
  - **Precision (Class 1):** 0.84
  - **Recall (Class 1):** 0.86
  - **F1- Score (Macro Avg)**: 0.85
  - **AUC-ROC:** 0.9207

The Random Forest model performed well, with an accuracy of 84.83%. However, the AUC-ROC score and some of the precision/recall values were marginally lower compared to the ensemble model, which may be due to the lack of additional diversity in my group's model compared to the ensemble approach.

Performance of the Ensemble Model

The ensemble model combines the following classifiers: Gradient Boosting, Random Forest, Logistic Regression, Extra Trees, and K-Nearest Neighbors, was evaluated using accuracy, classification report metrics (precision, recall, F1-score), and AUC-ROC. The parameters were finetuned with RandomizedSearchCV from the Methods section where the best parameters for each individual model were:

- **Gradient Boosting**: n_estimators=619, learning_rate=0.1194, max_depth=8
- **Random Forest**: n_estimators=675, max_depth=28
- **Logistic Regression**: C=57.0071
- **Extra Trees**: n_estimators=842, max_depth=22
- **KNN**: n_neighbors=5

Now I will output the performance metrics for the ensemble model.

- **Performance on Validation Set**:
  - **Accuracy**: 84.52%

- **Precision (Class 0)**: 0.85
- **Recall (Class 0)**: 0.84
- **Precision (Class 1)**: 0.84
- **Recall (Class 1)**: 0.85
- **F1-Score (Macro Avg)**: 0.85
- **AUC-ROC**: 0.9193

The model achieved high performance on the validation set, indicating that it is well tuned and generalizes well across different data splits. The AUC-ROC score further emphasizes its effectiveness at distinguishing between the two classes.

- **Performance on Test Split**:
  - **Accuracy**: 84.19%
  - **Precision (Class 0)**: 0.85
  - **Recall (Class 0)**: 0.84
  - **Precision (Class 1)**: 0.84
  - **Recall (Class 1)**: 0.85
  - **F1-Score**: 0.84
  - **AUC-ROC**: 0.9186

Model Performance Comparison

The ensemble model and the Random Forest model both performed well, with accuracies exceeding 84% and high AUC-ROC scores, and just the overall classification report suggests that the performance between the two are very similar looking at Figure 1.

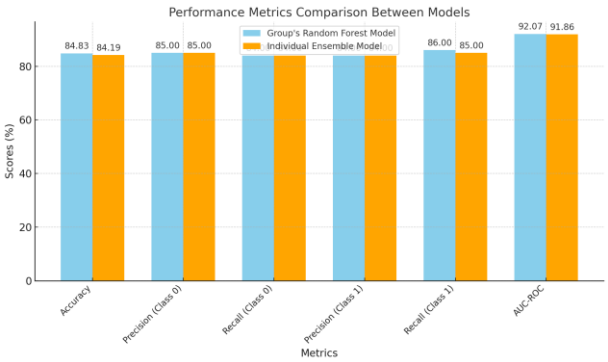

Figure 1: Performance Metrics between Models. The performance metrics of the two models, highlighting the minor differences across accuracy, precision, recall, and AUC-ROC

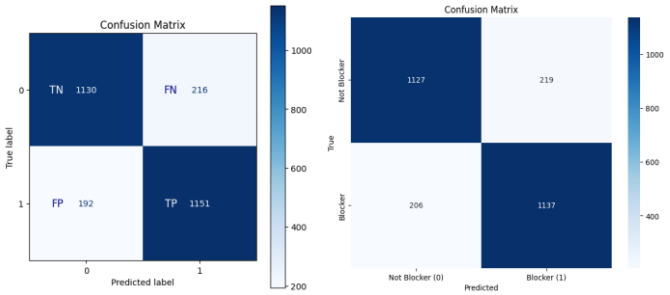

Figure 2: Confusion Matrices. The left confusion matrix represents the group's random forest model, and the right confusion matrix represents the individual ensemble model. Both matrices showcase the number of True Positives, True Negatives, False Positives, and False Negatives for each model respectively.

But looking at the specifics, I can see that the ensemble model exhibited better balance in precision and recall across both classes as the range of the values are 0.84-0.85 whereas for the group's random forest, it was 0.84-0.86. For the confusion matrices (Figure 2) for both models, the random forest model outperformed the ensemble model in fewer false positives (192 vs. 206 for the ensemble model) and also fewer false negatives (216 vs. 219). This indicates a higher precision for Class 0, as fewer non-blockers were incorrectly classified as blockers and slightly better recall for Class 1, meaning more actual blockers were correctly identified. Both models achieved strong AUC-ROC scores (>0.91), indicating that both are able to distinguish between the two classes well. The difference of 0.9207 for random forest vs. 0.9186 for ensemble model is most likely negligible.

Hyperparameter Tuning Attempts

I performed RandomizedSearchCV to finetune the hyperparameters for each classifier in the ensemble. Key parameters that worked well include:

- **Gradient Boosting**: n_estimators=619, learning_rate=0.1194, max_depth=8
- **Random Forest**: n_estimators=675, max_depth=28
- **Logistic Regression**: C=57.0071
- **Extra Trees**: n_estimators=842, max_depth=22
- **KNN**: n_neighbors=5

This fine-tuning significantly improved the ensemble model's performance, particularly for Gradient Boosting and Random Forest components.

What didn't work for finetuning the ensemble model was utilizing the GridSearchCV as mentioned in the Methods section because of how computationally intensive it was. The initial run of it ran for ~3 hours before it killed my system, which is why I utilized RandomizedSearchCV instead with 50 iterations. Even with 50 iterations, it was still computationally intensive with each run requiring ~1 hour to run, which is why tweaking the parameter ranges for RandomizedSearchCV would've been time intensive, so I set the ranges be in what I would expect the numbers to be so that within the 50 iterations, it randomly lands on a number for the parameter that would result in a good performance.

For the group's Random Forest model used 500 estimators with a random state of 42. This parameter setup worked well and provided high accuracy. But the 500 trees was chosen arbitrarily and so further hyperparameter tuning could've been done. This is not a case for what hypertuning attempts didn't work, but rather what hypertuning attempts should've been conducted as RandomizedSearchCV or GridSearchCV could've been done to adjust the number of trees, maximum depth, or minimum samples split so that the group's model could have a higher performance.

Model Attempts

SVM was one of the models I wanted to include in tthe ensemble as it had comparable performance results to the random forest model back in the group model work. But it was very computationally intensive and time intensive as well with it taking ~30+ minutes each run for training and prediction, and with the hypertuning I've done for the ensemble model, the RandomizedSearchCV would've taken hours or even days to run as with just 50 iterations and 5 models, it ran for ~1+ hours. If each SVM model took ~30 minutes to run with 50 iterations, that would result in 25 hours of finetuning for the model with the parameters I wanted. And I've already dialed down the number of iterations for the RandomizedSearchCV already to account for the time complexity, so SVM wasn't feasible time wise to to include in the ensemble model.

Discussion

Current Limitations

Despite the promising performance of the ensemble model, there are several limitations worth addressing. The ensemble model is computationally expensive due to the inclusion of multiple

classifiers. Training these each model and optimizing their hyperparameters using RandomizedSearchCV required significant time and computational resources. This makes the approach less feasible for larger datasets or in resource-constrained environments. With RandomizedSearchCV only having 50 iterations, the performance could definitely improve with more iterations given more compute power.

Although the ensemble model performed slightly better in terms of maintaining balance between precision and recall, Random Forest's was pretty comparable in that space as well. The ensemble model's AUC-ROC score was marginally lower than the group's Random Forest model and the confusion matrices show that the model has more false positives and false negatives compared to just the Random Forest. This suggests potential overfitting or difficulty in generalizing to unseen data.

In addition, the SMILES-based molecular fingerprints used in the models might not fully capture the complex biochemical properties of the molecules. This limitation could lead to suboptimal feature representation, constraining the model's ability to differentiate between hERG blockers and non-blockers effectively. I tried going into the RDKit library and extracting specific features of the molecule like molecular weight and pLogP values and combine those features with the whole morgan fingerprint, but it just overloaded the dataset and made it incredibly intensive to run on, so I settled for engineering the dataset in the same manner as the group. Again, if compute power was there, more specific feature extraction could've been done in combination with the methods of engineering the data my group worked on.

Finally, there is an imbalanced contribution of the selected models within the ensemble model. The ensemble model equally weighted all classifiers in the VotingClassifier, which may not be optimal. Some classifiers may contribute more valuable insights than others, such as Random Forest over Logistic Regression, but this was not accounted for in my current implementation.

Future Plans

The first key area that can improve the ensemble model's performance is to come back to feature engineering where I can try to reattempt extracting the specific features and utilizing them in tandem with the Morgan Fingerprints. I was concerned for runtime and compute complexity, but there are other techniques like dimensionality reduction techniques, such as Principal Component Analysis (PCA), that could also be explored to optimize feature space and help reduce the load on my system.

Weighted voting in the ensemble model could help prioritize the contributions of more effective classifiers, like Gradient Boosting and Random Forest, while reducing the influence of weaker models like Logistic Regression. Additionally, implementing a model selection pipeline that dynamically evaluates each model's contribution during training could allow underperforming models to be removed from the ensemble, which is something I did not account for in the workflow process.

The hyperparameter search space in RandomizedSearchCV could be expanded with more iterations, or even other techniques like the Bayesian Optimization and Stratified cross-validation could be utilized.

Finally, coming back to SVM's is a possibility I want to explore too since it had comparable performance results to Random Forest, which the ensemble model could definitely benefit from.

Conclusion

The ensemble model's complexity requires more computational resources and time for training, particularly during hyperparameter tuning. The Random Forest model, on the other hand, is computationally efficient and straightforward to implement. Its performance indicates that a simpler model can still achieve strong results on this dataset, particularly when computational efficiency and interpretability are priorities.

Overall, the ensemble model had a greater diverse architecture and extensive optimization, but the simplicity and efficiency of the Random Forest model and its performance, even if by a smaller margin, is greater than that of the ensemble model. By having less false positives and false negatives, the group's Random Forest model will have less potential candidate drugs scrapped from the start of the by being falsely labeled as a blocker while also preventing adverse outcomes later in the clinical trial stages where less blockers are falsely being labeled as non-blockers.