Examiner's report

Machine Learning in Drug Discovery and Design

Ibraheem Ajibola Ganiyu

12844772

BSc. (Hons) Software Engineering.

School of Computing, Engineering and Mathematics
University of Brighton

Contents

1 Project Evaluation

The title of my dissertation was *Machine Learning in Drug Discovery and Design* with the main focus being on predicting the penetration of drugs into the blood brain barrier.

It is a research-based project that is inspired by my placement spent at AstraZeneca, a biopharmaceutical company, where I gained practical experience in software development for clinical trials and research- based tasks. The project draws on the knowledge I have gained from my programming modules in the second year and also my final year module *CI346 Programming Languages and Client-Server Computing* and also the research skills I have gained both at the university and at AstraZeneca.

A significant amount of the time spent on the project was used to learn the necessary techniques needed to perform machine learning on large datasets and also on the required chemoinformatics knowledge required to successfully complete the project.

2 Background Research

I conducted research into Artificial Intelligence especially Machine Learning before embarking on the project. With little real world knowledge on how to perform machine learning, I turned to machine learning texts such as *Machine Learning by Tom Mitchell* and *Artificial Intelligence: A Modern approach by Peter Norvig.*

For the implementation of the machine learning algorithms, I also researched and experimented with numerous machine learning libraries but eventually settled on the Scikit-Learn and RDKit software libraries, due to their popularity and ease of use for data science. I also read *Chemoinformatics: Concepts, Methods and Tools for Drug Discovery* to get the necessary chemistry knowledge I needed to successfully work with the dataset.

3 Methodology and Planning

As the project is a research-based one, the core requirements or experiments were established before the project life cycle began. The core requirement of the project was to build a machine learning classifier that could be trained with our dataset on how to predict the probabilities of drugs passing through the blood brain barrier. As stated in the interim report, the main stages of the implementation phase with deliverables of the projects are

- Data Cleaning

- Classifier Training
- Ensemble Classifier training
- Deployment
- Testing and Quality Assurance

3.1 Detailed discussion the project stages

3.1.1 Data Cleaning

As is the norm for most data science tasks, the dataset needed to be parsed, unnecessary data removed and the required portions transformed into another form for use.

3.1.2 Classifier and Ensemble Classifier Training

Arguably one of the most important sections, the respective machine learning classifiers chosen, needed to be trained with the processed dataset. Here, four machine learning classifiers (K-Nearest Neighbours, Support Vector Machines, Random Forests and Neural Networks) were trained. In the original project plan, deep neural networks were scheduled to be trained with the dataset but this couldn't be done due to time constraints.
All the trained classifiers were then combined successfully into an Ensemble classifier with an average accuracy of $87\%(+/-7\%)$.

3.1.3 Deployment

In the interim report, it was stated that the project would yield 3 deliverables: a web client, a web server and a prediction API (engine). The ensemble classifier was successfully wrapped in a web server and a REST API was exposed to receive and return prediction results. A web client, however, was not built; This was noted down as an extension that would be done given enough time as it would provide an intuitive and easy accessible UI to make the product easier to use.

3.1.4 Testing and Quality Assurance

All the classifiers were tested using the standard cross-validation technique for evaluating machine learning classifiers and a ROC (Receiver Operating Characteristic) curve was drawn to compare the ensemble classifier against a dummy classifier that is making random guesses.

4 Research Evaluation

The core of the research was to build a classifier to predict the probability of a potential drug candidate passing through the blood brain barrier. This was successfully achieved with an Ensemble classifier that performs at best with an accuracy of 94% and at worst with an accuracy of 80%.

4.1 Problems encountered

One of the common problems encountered throughout the training of the classifiers was achieving accuracy scores lower than 50% on the classifiers. A classifier achieving lower than 50% on prediction results would achieve poorer results than a dummy classifier making random guesses.

Also, due to the high dimensional nature of the dataset, it took longer to train some classifiers, especially the support vector machine classifier as based on read literature we know it would perform poorly on a high dimensional dataset which it did.

Another problem encountered at the beginning during the preprocessing of the dataset was the inability of RDKit to parse some SMILE formats. This is most likely due to the SMILE formats being malformed and probably due to a bug in the RDKit library. However, only a small number of molecule SMILEs (13) , failed to be parsed. This was safely ignored as we still had 2040 molecule SMILEs left that successfully got parsed.

The last problem encountered in the project was combining all the individually trained classifiers into an Ensemble classifier. An option explored was to create a custom class in python where it receives as input the feature vectors and then passes this vector to the individual classifiers and collects their results, however, this resulted in scope creep as doing such as task turned out to be much more difficult than expected.

4.1.1 Solutions put forward

A solution implemented to prevent the classifiers from achieving poor results was to perform feature engineering on the input dataset. An example was the scaling of values in the dataset to have a maximum of 1 and a minimum of -1 using a *MinMaxScaler*. This helped normalize the dataset as some classifiers that utilise distance metrics would return poor results due to the large variance in the dataset. Also, the neural network classifier would converge much faster during gradient descent with a normalised data due to the normalisation.

To help reduce the training time of the support vector machine (SVM) classifier. The Principal Component Analysis (PCA) algorithm was used on the dataset before feeding it to the SVM classifier. Different values were experimented with to the determine the best number of components to use in the SVM classifiers that would achieve the maximum result whilst still reducing the training time of the algorithm.

The solution to the incompatible ensemble classifier problem was to utilise the *Voting Classifier* class in Scikit-Learn. This had the advantage of being an Ensemble classifier and also had the options of specifying the voting mechanism to use. The voting mechanism here refers to the metric to use when deciding the final prediction result of the Ensemble from the results of the individual classifiers.

4.2 Assessment on the success of the project

Based on the prediction accuracy of the ensemble classifier, the project could be said to be successful. In a paper by Ana Teixeira et al, *A Bayesian Approach to in Silico Blood-Brain Barrier Penetration Modeling*, they listed all the accuracy of the classifiers numerous other researchers have applied to the BBB problem. The lowest result was an accuracy of 74.8% in 2000 using the partial least-squares method whilst the highest was an accuracy of 97.2% in 2007 using a recursive partitioning model and partial least squares method.

This project achieves an average prediction accuracy of 87% (+/- 7%) which lies in between the results mentioned earlier.

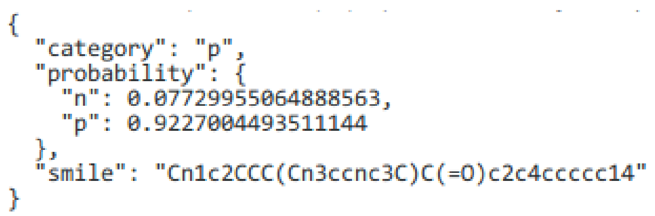
5 Areas of Interest

5.1 Prediction API

The prediction API is one of the areas of the project that can provide real world value. The decision to use a REST API was to enable the integration of the Ensemble classifier into any other program or software that the scientist uses in the form of micro-services. A sample use case would be web client that draws a list of molecules from a virtual library micro-service, the user of the client then performs some filtering on the list and sends the remaining molecules to our prediction micro-service, which then returns a list of molecules with their respective probabilities of passing through the blood-brain barrier returned. An example taken from the project is shown below, where a sample curl call is made.

```
$ curl -H "Content-Type: application/json" -X POST -d
'{"smile":"Cn1c2CCC(Cn3ccnc3C)C(=O)c2c4ccccc14"}'
http://localhost:5000/api/prediction
```

And a prediction result is returned as shown in figure 1



```
{
  "category": "p",
  "probability": {
    "n": 0.07729955064888563,
    "p": 0.9227004493511144
  },
  "smile": "Cn1c2CCC(Cn3ccnc3C)C(=O)c2c4ccccc14"
}
```

Figure 1: Sample prediction result for the BBB Rest API

5.2 Learning curve of Ensemble classifier

The learning curve of the ensemble classifier is also another area of interest as shown in figure 2.

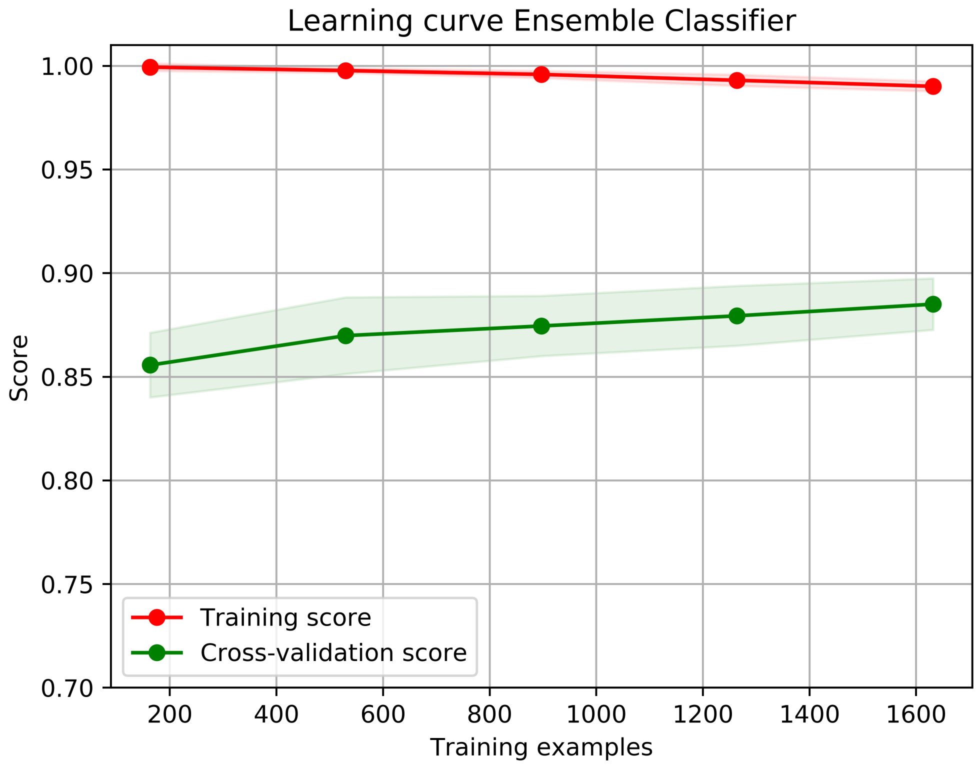


Figure 2: Learning Curve for the Ensemble classifier

The reduction in training score shows that the classifier over fits less with more data which is ideal as we want it to perform well on new datasets. The cross-validation score also increases with more data, the total number of samples in the dataset was 2048; Which is a small amount of data compared to large body of chemical datasets available. The assumption here re-enforced by figure 2 is that with more data, we can achieve a higher accuracy score on the ensemble classifier

6 Further Areas for improvement

Given more time, other machine learning classifiers could have been explored, especially deep learning techniques. Based on read literature, they have found a profound and effective use in virtual screening.

Also, the total training time of the classifiers takes roughly about 30 minutes. One of the proposed solutions at the beginning of the project was to implement "Map Reduce" using Apache Spark to train all the respective classifiers on different clusters and combine the result. This would greatly reduce the training time, especially if dealing with a larger dataset.