Predicting Molecular Blood-Brain Barrier Penetration Using Machine Learning

Caleb Kwon

AOS C111 - Introduction To Machine Learning For Physical Sciences

Dr. Alexander Lozinski

12/05/2025

1 Introduction

The blood-brain barrier (BBB) is a semi-permeable membrane that selectively filters molecules to enter the region containing the central nervous system (CNS). Blood-barrier penetration is difficult to accurately predict, due to its complex structure and dependence on heterogeneous molecular properties. Factors like molecular size, hydrophobicity, polarity, and interaction within the thousands of other molecules present in the human body can affect a molecule's ability to penetrate the blood brain barrier (Nau et al. 2010). While rules such as the Lipinski Rule of 5 (Lipinski et al. 2012) provide general guidelines for pharmacokinetics, these are only heuristic tools, with often many exceptions.

In this project, I aim to train supervised machine learning models to predict whether or not molecules can cross the blood brain barrier, given their molecular structures via 1-dimensional Simplified Molecular Input Line Entry System (SMILES) strings (Weininger 1988).

2 Data

The dataset used in this report is the Blood-Brain Barrier Penetration (BBBP) dataset, created by Martins et al. (2012) and accessed via MoleculeNet. The data consists of over 2,000 compounds, and contains the molecule name, its SMILES string and a binary blood brain barrier penetration label (1 if penetrating, 0 if not).

	num	name	p_np	smiles
0	1	Propanolol	1	[Cl].CC(C)NCC(O)COc1cccc2ccccc12
1	2	Terbutylchlorambucil	1	C(=O)(OC(C)(C)C)CCCc1ccc(cc1)N(CCCl)CCCl
2	3	40730	1	c12c3c(N4CCN(C)CC4)c(F)cc1c(c(C(O)=O)cn2C(C)CO...
3	4	24	1	C1CCN(CC1)Cc1cccc(c1)OCCCNC(=O)C
4	5	cloxacillin	1	Cc1onc(c2ccccc2Cl)c1C(=O)N[C@H]3[C@H]4SC(C)(C)...

Figure 1 ***First five rows of the raw BBBP dataset.***

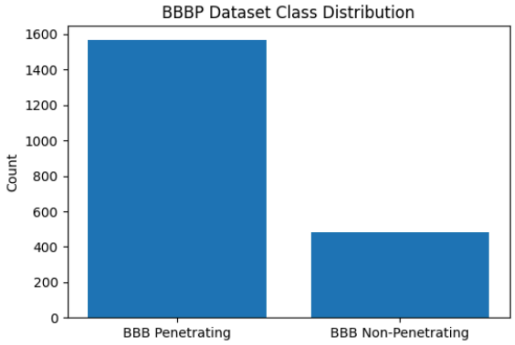

Figure 2 ***Class distribution of the 2039 molecules in the BBBP dataset.*** The dataset is characterized by moderate class imbalance, with over 75% of molecules in the penetrating (=1) class.

Feature engineering:
To extract meaningful molecular properties from the data above, I used RDKit – an open-source cheminformatics Python library – to obtain molecular descriptors from the SMILES strings in the dataset. I selected nine molecular properties:

- **Molecular Weight** (MolWt): Mass of the molecule. Smaller, less massive molecules are expected to more easily diffuse through the BBB.
- **# Hydrogen-Bond Donors** (NumHDonors): Number of atoms capable of acting as hydrogen-bond donors. A higher value increases the polarity of the molecule, decreasing its ability to pass through the non-polar BBB.
- **# Hydrogen-Bond Acceptors** (NumHAcceptors): Number of atoms capable of acting as hydrogen-bond acceptors. Like hydrogen-bond donors, this serves as a proxy for molecule polarity.
- **Octanol-Water Partition Coefficient** (MolLogP): Tendency of a molecule to dissolve in octanol versus water; used as a proxy for lipophilicity. The BBB is lipid-based, so I expected higher LogP values to correspond to higher likelihoods of BBB penetration.
- **Topological Polar Surface Area** (TPSA): Measures the surface area contributed by polar atoms. Provides a scalar value reflecting the molecule's 3-dimensional polarity profile.
- **Rotatable Bonds** (NumRotatableBonds): Number of non-ring, single bonds. Higher values correspond to higher flexibility, which is typically *not* favorable for BBB penetration.
- **Heavy Atom Count** (HeavyAtomCount): Number of non-hydrogen atoms. Acts as a proxy for molecular size.
- **Ring Count** (RingCount): Number of rings in the molecular structure. Rings contribute to molecular rigidity, and also create more defined 3D structure in the molecule, which can affect its membrane permeability.

- **Aromatic Rings** (NumAromaticRings): Number of aromatic rings. Aromatic rings are known to increase lipophilicity, due to the hydrophobicity of the delocalized π electron systems characteristic of aromatic rings.

I appended these features to my dataset, and conducted a train test split using scikit-learn train_test_split to partition 70% of the data to training, and 30% of the data to testing. The split was conducted with stratification of the target label, blood brain barrier penetration, to balance class proportions in each split of the data.

	MolWt	NumHDonors	NumHAcceptors	MolLogP	TPSA	NumRotatableBonds	HeavyAtomCount	RingCount	NumAromaticRings
min	28.05	0.00	0.00	-11.74	0.00	0.0	2.00	0.00	0.00
max	1879.68	24.00	33.00	10.81	662.41	35.0	132.00	16.00	7.00
mean	344.54	1.55	4.58	2.32	70.73	4.2	24.05	2.99	1.41
median	324.41	1.00	4.00	2.47	55.12	4.0	23.00	3.00	1.00

Figure 3 ***Summary statistics for the selected molecular descriptors.*** The data includes molecules with a diverse range of properties. Incongruent scaling between features necessitates scaling to zero mean, unit variance for scale-variant models explored in this project such as logistic regression, multilayer perceptrons, and principle component analysis.

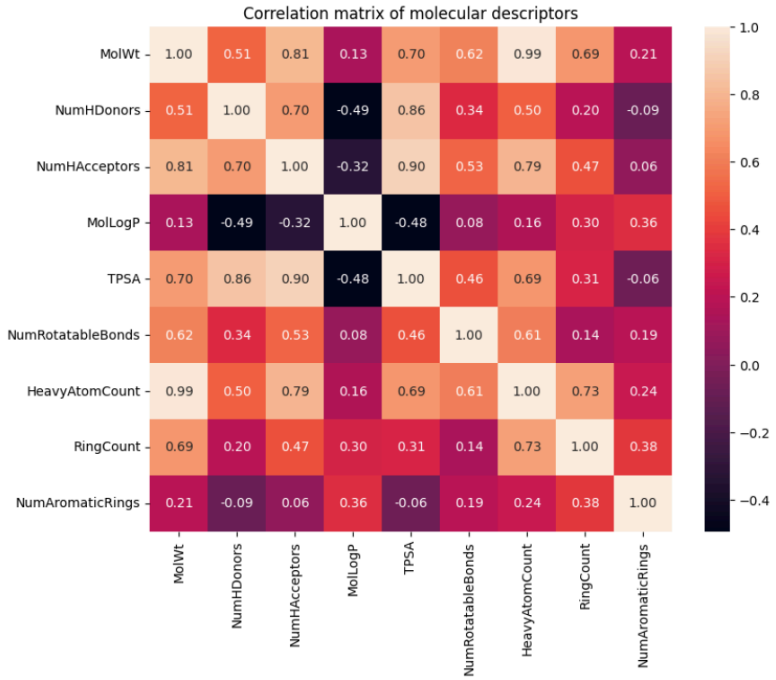

Figure 4 ***Correlation matrix between molecular descriptors.*** Moderate correlations are observed between several of the descriptors, with particularly high correlation (0.99) between HeavyAtomCount and MolWt. Collinearity could hinder model interpretability and affect model performance, potentially alleviated by PCA.

For models trained on scale data (logistic regression, MLP, PCA), data was standardized and scaled to zero mean and unit variance using scikit-learn StandardScaler().

3 Modelling

For this classification task, I used several different machine learning models with fixed random seeds. I used ROC-AUC as my primary metric for evaluating model performance, due to its ability describe model performance across a linear sweep of decision threshold values. This is useful for a feature like blood-brain barrier penetration, where researchers may prioritize blood-brain barrier penetration/non-penetration differently. For example, researchers optimizing a neurotoxic drug may want a lower decision threshold to prioritize avoiding misclassifying neurotoxic drugs as non-neurotoxic. Researchers optimizing an Alzheimer's therapeutic that acts in the brain may want a higher threshold to explore more options for potentially viable molecules.

Below is the function I used for the majority of model deployments. For a provided model, it trains and applies the model, and produces accuracy, F1 score, ROC-AUC, and confusion matrices to quantify model performance.

```
def perform_classification(model, X_tr, X_te, y_tr, y_te):

    # Fit and predict
    model.fit(X_tr, y_tr)

    y_pred = model.predict(X_te)
    y_score = model.predict_proba(X_te)[:, 1]   # y_score is the predicted probability that the label
is 1

    # Accuracy
    accuracy = accuracy_score(y_te, y_pred)
    print(f"Accuracy: {accuracy:.3f}")
    balanced_acc = balanced_accuracy_score(y_te, y_pred)
    print(f"Balanced Accuracy: {balanced_acc:.3f}")

    f1 = f1_score(y_te, y_pred)
    print(f"F1 score:, {f1:.3f}")

    # ROC-AUC
    fpr, tpr, thresholds = roc_curve(y_te, y_score)
    roc_auc = auc(fpr, tpr)
    print(f"ROC-AUC: {roc_auc:.3f}")

    plt.figure(figsize=(6, 5))
    plt.plot(fpr, tpr)
    plt.plot([0, 1], [0, 1], linestyle='--', label='Random Guess')
    plt.xlim(0, 1); plt.ylim(0, 1)
    plt.xlabel('False Positive Rate (FPR)'); plt.ylabel('True Positive Rate (TPR)')
```

```
    plt.legend([f'ROC-AUC = {roc_auc:.3f}'], loc='lower right')
    plt.show()

    # Confusion Matrix
    cm = confusion_matrix(y_te, y_pred)

    disp = ConfusionMatrixDisplay(cm)
    disp.plot()
    plt.show()

    return accuracy, balanced_acc, f1, roc_auc
```

3.1 Random Forest

The first model I implemented was the random forest model, and specifically the sklearn RandomForestClassifier. Random forests are powerful classifiers that can capture non-linear data, avoid overfitting through bootstrap aggregation of numerous base-estimators, and can provide an idea of which features are most important to BBB penetration. I used class_weight='balanced' to account for class imbalance. For this method, I also implemented k-fold cross validation. 5-fold cross validation offers more stable evaluations of model performance by partitioning the data into five folds, and training the model five times on all but one fold reserved for testing, and assigning a new testing fold for each training iteration.

3.2 Logistic Regression

Following the implementation of my random forest model, I compared it to a sklearn logistic regression baseline. Logistic regression is a linear classifier, and I expected it to struggle due to complex relationships that govern BBB penetration.

3.3 Multilayer Perceptron

To address the non-linearity of the dataset, I also implemented a sklearn multilayer perceptron (MLP) classifier. Multilayer perceptrons use activation functions to learn highly nonlinear patterns with the data, and their complexity can easily be finetuned by adjusting hidden layer counts, neuron counts, and regularization strength. I tested five separate MLP models of varying hidden layer and neuron counts.

3.4 PCA + Logistic Regression

To address the high dimensionality and potential redundancy between the selected features, I additionally tried applying principal component analysis (PCA) to the data before training the logistic regression classifier, to see if it would significantly affect the classification performance. I first plotted the cumulative explained variance ratio plot to determine how many principle components to keep. Because I noticed a sharp “elbow” after three principal components, I reduced the dimensionality of the dataset to three PCs, which still explains over 90% of the variance in the data.

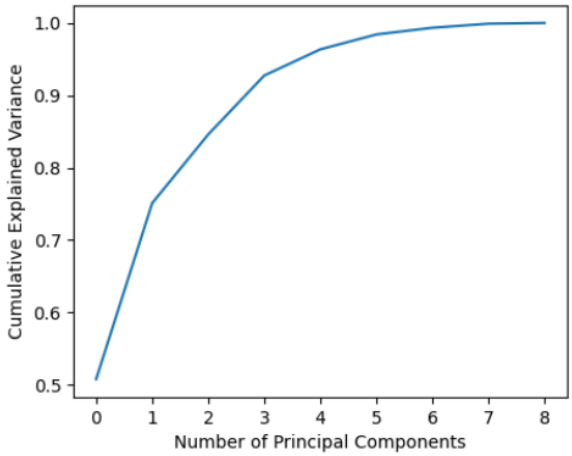

Figure 5 ***Cumulative Explained Variance plot of PCA.*** Following two PCs, 80% of the variance is explained. Following three PCs, 90% of the variance is explained. Following four PCs, 95% of the variance is explained.

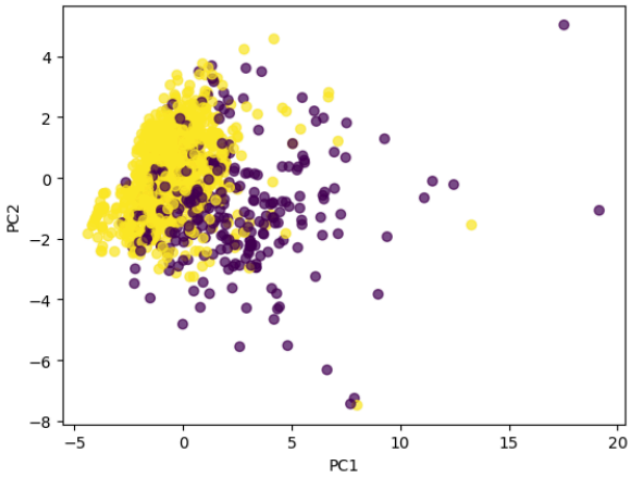

Figure 6 ***First two (of three) principal components of BBBP PCA.*** The third PC is omitted from this graph for the sake of 2D visualization. The data is nonlinear, with high densities of both penetrating and non-penetrating molecules near the origin.

With the same hyperparameters as in 3.2, I applied the logistic regression model.

3.5 PCA + Support Vector Machine

Building on the PCA applied in 3.4, I also trained a support vector machine (SVM) classifier using sklearn SVC. SVMs, despite drawing decision boundaries using linear hyperplanes, were selected because of their ability to separate non-linearly related classes using the "kernel trick," which maps data into a higher-dimensional feature space. To this end, I used the radial basis

function kernel. I hoped that this would learn non-linear data much like a random forest or MLP model would.

4 Results

4.1 Random Forest

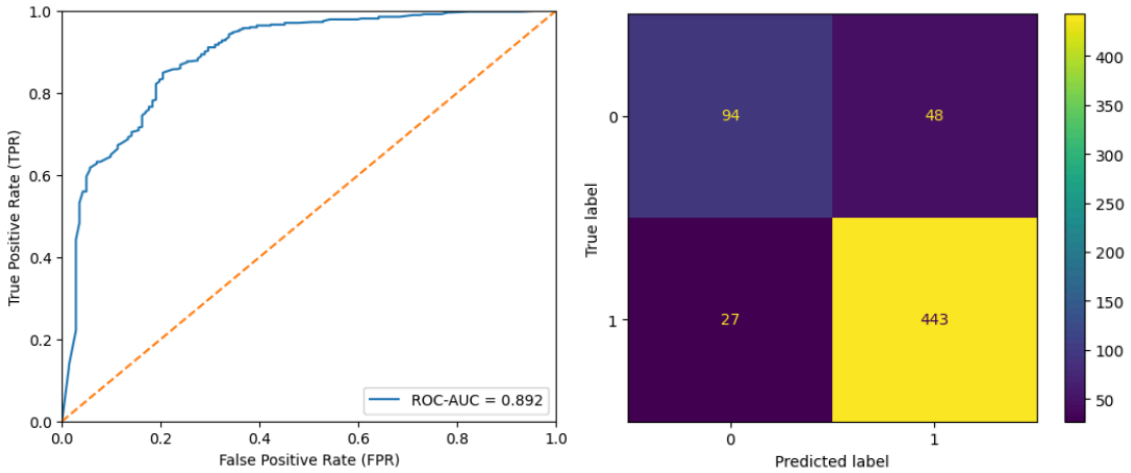

Figure 7 ***Random forest classification: ROC curve (left) and confusion matrix (right).*** (1=BBB penetrating, 0=BBB non-penetrating).

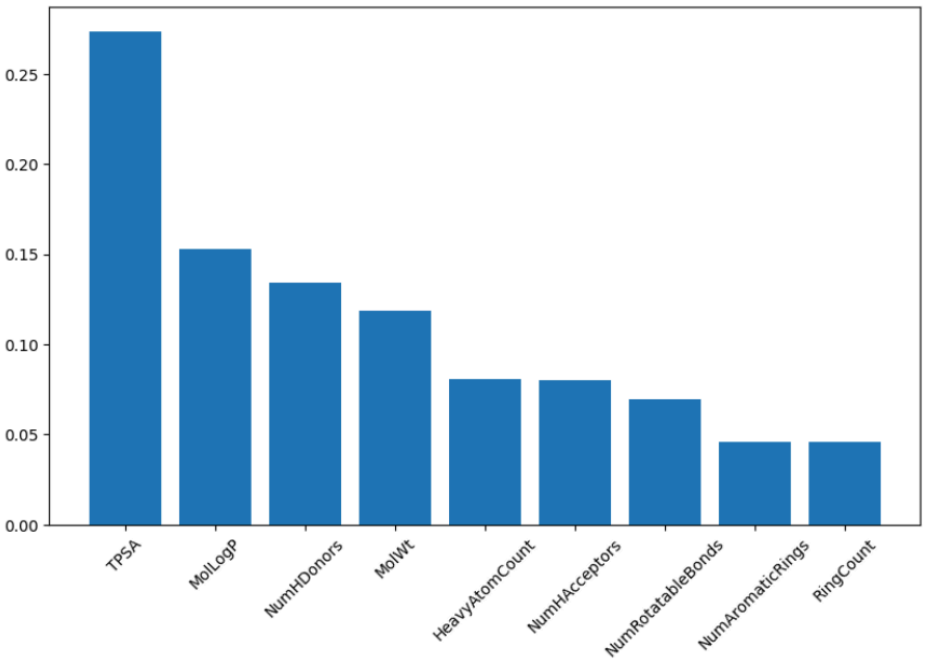

Figure 8 ***Feature importances of random forest classification.*** TPSA is the feature that holds the most predictive power.

4.2 Logistic Regression

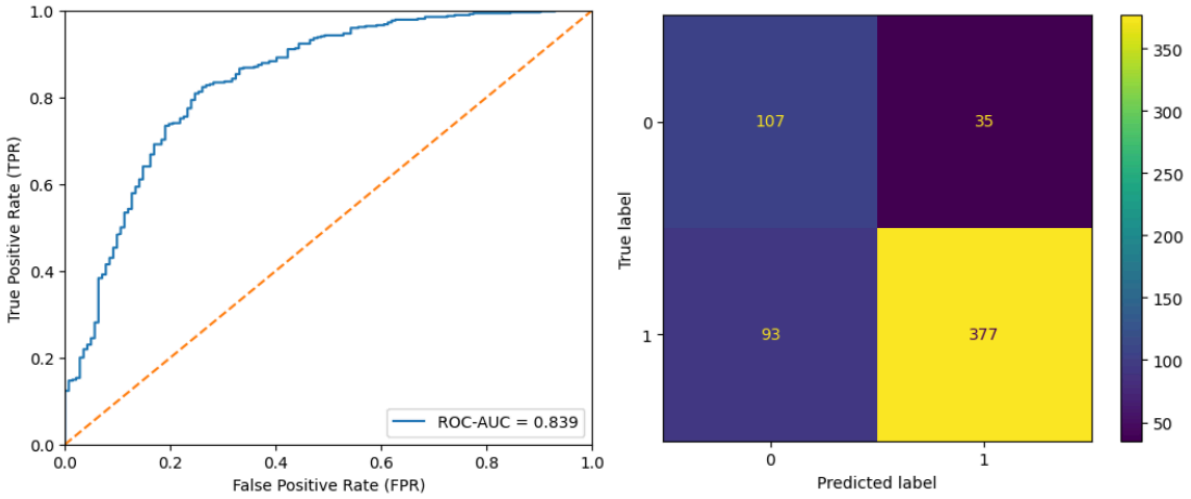

Figure 9 ***Logistic regression: ROC curve (left) and confusion matrix (right).*** The ROC curve is noticeably smaller, and despite an improvement in true negative predictions, we see a strong decrease in true positives.

4.3 Multilayer Perceptrons

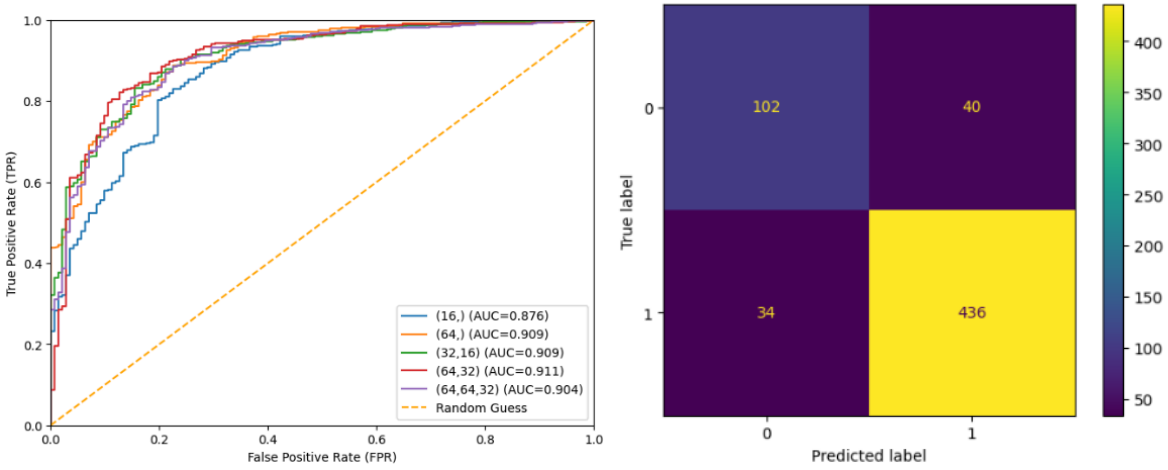

Figure 10 ***Multilayer perceptrons: ROC curves for each of five tested MLP layer sizes (left) and confusion matrix for the highest performing (64,32) MLP (right).*** ROC-AUC on the left was computed on the testing set. Due to the absence of a validation set, the model performance was averaged between the five variations tested, yielding a ROC-AUC of 0.902.

4.4 PCA + Logistic Regression

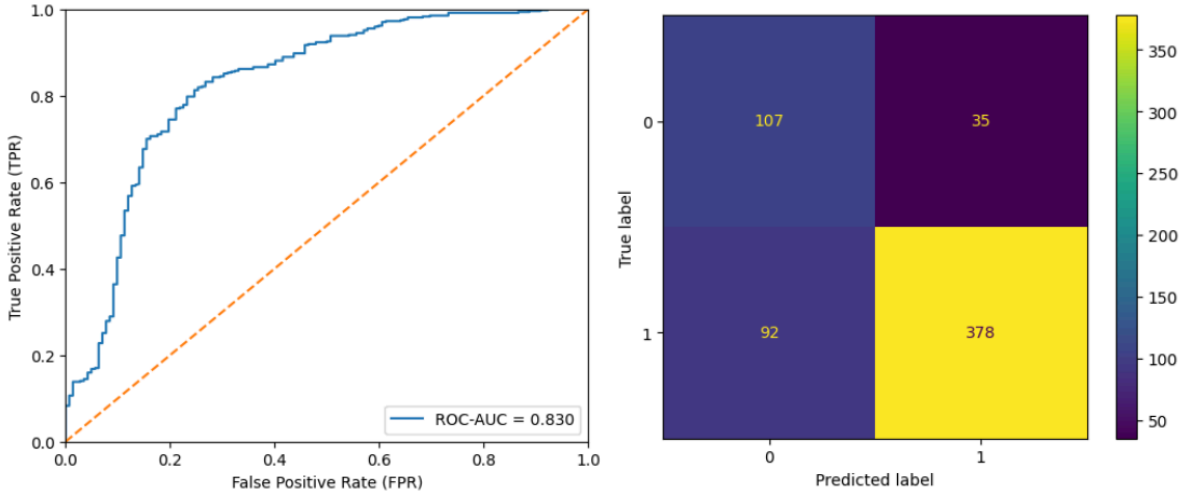

Figure 11 ***PCA and logistic regression: ROC curve (left) and confusion matrix (right).*** There is a small decrease in model performance compared to logistic regression alone.

4.5 PCA + SVM

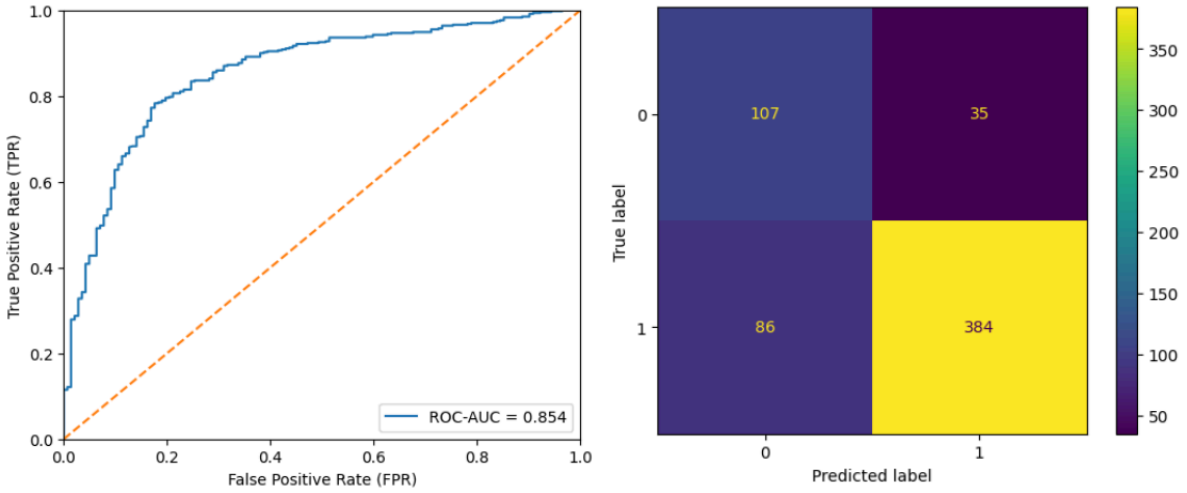

Figure 12 ***PCA and SVM: ROC curve (left) and confusion matrix (right).*** The model slightly outperforms the logistic regressions, but underperforms compared to the random forest and MLP.

Table 1 ***ROC-AUC scores by machine learning method used.*** The method that yielded the highest ROC-AUC was the multilayer-perceptron, with a ROC-AUC of 0.906. F1 score and class-balanced accuracy are also reported. ROC-AUC for the MLP is an average of five tested variations of hidden layer size. F1 score and balanced accuracy for MLP not obtained.

Model	**ROC-AUC**	F1 score	Balanced Accuracy
Random Forest	0.892	0.922	0.802
Logistic Regression	0.839	0.855	0.778
MLP	**0.902**	-	-
PCA + Logistic Regression	0.830	0.856	0.779
PCA + SVM	0.878	0.874	0.805

5 Discussion

The objective of this work was to see to what extent simple machine learning models would be capable of learning and predicting blood-barrier penetration of therapeutic molecules based on their molecular descriptors. I found that the model that yielded the highest performance was the multilayer perceptron. Between the five tested hidden size variations, four of them achieved a ROC-AUC above 0.900, with the (64,32) variation reaching 0.911. However, because the performance of the different variations of the MLP model were evaluated on the testing set, the testing set no longer represents an unbiased measure of final model performance. Without a separate validation set, I chose to use the average of the five variations, 0.902, as my final metric for the model's performance, electing not to "select" a hyperparameter.

The random forest also performed admirably, and it predicted more true positives than any other model. However, it underperformed the multilayer perceptron due to its higher false positive rate. As expected, the random forest and MLP models performed well due to their ability to learn nonlinear relationships in data. The logistic regression classified more true negatives than both the random forest and the MLP models, but at the cost of a large number of false negatives. Adjusting the decision threshold could potentially improve future deployments of this model.

Because nine molecular descriptors were selected as input features for the models, I also performed PCA to see the effects of dimensionality reduction on this dataset. The 2D PCA plot demonstrated that following dimensionality reduction via PCA, the data was still highly nonlinear. Because of the loss of information facilitated by PCA, I expected a decrease in

performance for the logistic regression model. In fact, logistic regression on the PCA-reduced data scored 0.009 ROC-AUC lower than the logistic regression on the original data. The support vector machine with the rbf kernel was an improvement over the PCA, but still scored lower than the random forest and MLP models. In essence, with the rbf kernel applied, the SVM model served as middle ground with regards to model linearity. While random forests and MLP models represent highly nonlinear learning methods, while PCA (an affine transformation) and logistic regression represent linear methods, SVM acts both linearly and nonlinearly by drawing a linear decision boundary after a nonlinear transformation of data into its feature space. Accordingly, the MLP and random forest models did best, followed by SVM, followed by logistic regression. The evidence supports that the BBBP dataset benefits from nonlinear learning methods.

Furthermore, in Figure 10, where I explored the effect of hidden layer size on model performance, I observed that the worst performing model was the one hidden layer, 16 neuron model. This indicates that our dataset requires a reasonable amount of complexity in order to learn the data in greater detail. Conversely, a three layer, 64, 64, 32 neuron MLP had the second worst performance. This suggests that the MLP models can begin overfitting when equipped with an excessive amount of parameters, likely due to the relatively small size of the database (~2000 molecules).

The relative feature importance values in Figure 8 also provides insight into the mechanism by which molecules are selectively filtered by the BBB. TPSA (Topological Polar Surface Area) was deemed by the random forest to have the greatest predictive value among the nine molecular descriptors. While this indicates a strong association between TPSA and BBB penetration, we can *not* make any assumption of any causal relationships from this data, due to the multicollinearity of the descriptors as seen in Figure 4 – for instance, TPSA is highly correlated with molecular weight and the number of H-bond donors/acceptors. Rather, TPSA demonstrates its function as a useful heuristic for BBB penetration, as it synthesizes information regarding molecular size and polarity. However, rigorous predictions of BBB penetration should certainly regard other descriptors as well.

6 Conclusion

In this project, I tested various machine learning models in an applied problem of determining blood-brain barrier penetration from nine physicochemical descriptors. The results demonstrated that machine learning models can achieve reasonable performance for this purpose, with the multilayer-perceptron achieving a ROC-AUC of 0.902. This reflects the ability of MLP models to learn nonlinear data, in cases where linear models like logistic regression underperform. MLP models of two hidden layers appear to have the greatest success with this dataset, while overly simple (16,) or complex (64,64,32) models may underfit or overfit the data. With the random forest model, I also determined that Topological Polar Surface Area is the most predictive feature among the nine tested for determining blood-brain barrier penetration, aligning with my pre-existing chemical insights.

References

Lipinski, C. A., Lombardo, F., Dominy, B. W., & Feeney, P. J. (2012). Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. Advanced drug delivery reviews, 64, 4-17.

Martins, I. F., Teixeira, A. L., Pinheiro, L., & Falcao, A. O. (2012). A Bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*, 52(6), 1686-1697.

Nau, R., Sörgel F., & Eiffert, H. (2010). Penetration of Drugs through the Blood-Cerebrospinal Fluid/Blood-Brain Barrier for Treatment of Central Nervous System Infections. *Clin Microbiol* Rev 23:.https://doi.org/10.1128/cmr.00007-10

Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1), 31-36.