Machine learning prediction of patients with schizophrenia from anatomical brain imaging

1 Introduction

There has been considerable interest in the ability of machine learning techniques to assist in the diagnosis of schizophrenia from brain anatomy, including from structural MRI scans.[11] Schizophrenia is a critical and high-risk inherited mental disorder. Patients experience debilitating symptoms that may present over a long clinical period; it takes an average of ten years from the onset of the first symptoms to definitive diagnosis of the disease.[11]

The application of machine learning in this domain involves various choices as to the selection of machine learning architectures, and the design of the respective data pre-processing and cross-validation ("CV") pipelines. This study aimed to investigate the impact of such choices on the performance of three types of machine learning model in predicting clinical status, i.e. patient with schizophrenia versus healthy.[2] The selected model architectures included logistic regression with elastic net regularization ("logistic regression"), light gradient boosting machine ("LightGBM"), and a support vector machine with a radial basis function ("RBF") kernel ("SVM"). Each model was applied to a low or high-dimensional version of pre-processed structural MRI data. The study also considered the impact of a common CV strategy versus group stratified CV by sex. The best performing model was uploaded to the RAMP challenge environment.[2]

2 Materials and methods

2.1 Imaging data

Voxel-based morphometry data from structural MRIs was obtained from the RAMP platform.[2] This included the data for 277 patients with schizophrenia, and 236 control patients, ranging from 14-65 years of age (see Figure 1. There were 322 and 191 patients of each sex, encoded in binary format.

2.2 Models used for comparison

The objective for each model is to train a classifier that can make a binary decision about whether a patient has schizophrenia or not. Binary classification models learn a function to predict class labels, given a set of N observations $\mathbf{x}_i \in \mathbb{R}$, with associated target variables y_i. The models and CV pipelines were from sci-kit learn version 1.4.1.post1 unless otherwise noted. Notation is defined upon its first occurrence and is used consistently throughout this document.

Logistic regression Logistic regression is a linear discriminative classifier which learns a vector of weights $\mathbf{w}$ and a bias term b, and uses this to predict the distance to the boundary $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$.[12] For each sample $\mathbf{x}_i$ the log-odds of the positive class $y_i \in \{0, 1\}$ can be represented as a linear function $f(\mathbf{x}_i) = \log \frac{p_i}{1-p_i}$, where $p_i = \Pr(y_i = 1)$ is the estimated probability of the positive class. This log-odds function $f(\mathbf{x}_i)$ is then transformed into a probability estimate via the logistic sigmoid function: $p_i = \sigma(f(\mathbf{x}_i)) = \frac{1}{1+e^{-f(\mathbf{x}_i)}}$.[6]

The weights $\mathbf{w}$ and bias term b can be found by minimizing the loss function on the training data. The generic form of a regularized loss function for logistic regression is $\mathbf{w}^*, b^* = \underset{\mathbf{w},b}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, p_i) + \lambda J(\mathbf{w})$.

The loss function $L(y_i, p_i)$ is the binary cross-entropy loss, which represents the negative log-likelihood of a model that returns the predicted probabilities for the training data, $L_{\log}(y_i, p_i) = -(y_i \log(p_i) + (1-y_i) \log(1-p_i))$.[6]

The regularization term $J(\mathbf{w})$ serves to control the complexity of the model, and prevent overfitting. The λ term balances the amount of regularization with the closeness of the model fit to the training data.[7] In this case, elastic net regularization was used, which uses both the l_1 and l_2 norms of the weights: $J(\mathbf{w}) = \alpha||\mathbf{w}||_1 + (1-\alpha)||\mathbf{w}||_2^2$. The trade off between the two is controlled by the α hyperparameter, providing flexibility when tuning the model to suit either dataset. l_1 regularization encourages sparse solutions, essentially performing feature selection. l_2 regularization encourages smoothness and low but not necessarily 0-valued, weights.

Accordingly, logistic regression is a flexible and relatively simple model, able to handle both datasets. The elastic net regularization in combination with the model's linear decision boundary makes it relatively robust to noise, which can prevent overfitting.

LightGBM LightGBM is a gradient boosting decision tree algorithm ("GBDT"), from the lightgbm library version 4.3.0. GBDT models sequentially apply M weak decision tree classifiers $f_m(\mathbf{x})$ to a repeatedly modified version of the data, then combine the sequence of weak classifiers to create a strong learner.

At each iteration, the negative gradient of the loss function $-\frac{\partial L(y_i, f_m(\mathbf{x}_i))}{\partial f_m(\mathbf{x}_i)}$ is calculated for each instance in the dataset, with the target variable $y_i \in \{0, 1\}$. The loss

function utilized for binary classification in LightGBM is the binary cross-entropy loss as defined in 2.2 above.[4] The average of these gradients is used to select the classifier that minimizes the loss function in that iteration.

The predictions from the selected weak classifiers are combined through a weighted majority vote to produce the final prediction, $f(\mathbf{x}) = \sum_{m=1}^{M} w_m f_m(\mathbf{x})$. w_m is the weight for the m-th weak learner, derived by minimizing the loss when $f_m(x)$ is added to the current ensemble, $w_m^* = \arg\min_\alpha \sum_{i=1}^{N} L(y_i, f_{m-1}(\mathbf{x}_i) + \alpha f_m(\mathbf{x}_i))$.[9]

The iterative process associated with GBDT models is associated with reduced bias, and the combination of many weak learners serves to reduce variance. While this behaviour is useful for both datasets, LightGBM has particular features that are tailored for efficiency and scalability in high-dimensional feature contexts. LightGBM uses a leaf-wise split allows the model to grow the tree in areas of the feature space where the current ensemble of trees is performing poorly, regardless of its depth in the tree. Please see [10] for the approach to leaf splitting. This approach can contribute towards overfitting if not used in conjunction with regularization within the individual trees, which was addressed through the hyperparameter tuning described in 2.3. [5]

SVM with RBF Kernel When applied to a binary classification problem, SVM finds a function of the data that maximizes the margin of a separating hyperplane in a high-dimensional feature space between the data points associated with each class. Mathematically y_i is typically considered in this context as $y_i \in \{-1, 1\}$.[14] In this case, $\mathbf{w}$ refers to the separating hyperplane in some dot product space $\mathcal{H}$, $\langle \mathbf{w}, \mathbf{x}_i \rangle + b = 0$.[15]

The 'soft margin' SVM optimization problem aims to find the hyperplane that maximizes the margin while also allowing some misclassifications.[12] This is achieved by introducing a regularization hyperparameter C that controls the trade-off between minimizing the misclassifications and maximizing the margin, in the objective function $\mathbf{w}^*, b^* = \arg\min_{w,b} -C \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + J(\mathbf{w})$.

The degree of misclassification of each data point is measured by the hinge loss function, defined as $L(y_i, f(\mathbf{x}_i)) = \max(0, 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b))$. This function is zero for correctly classified points that are outside the margin and linearly increasing for points inside the margin or misclassified points. The most common approach to regularization entails minimizing the l_2 norm of $\mathbf{w}$.[8], so $J(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$.

This optimization problem may be considered in the dual form, $\min_\alpha \frac{1}{2}\alpha^T Q \alpha - e^T \alpha$, where $0 \leq \alpha_i \leq C, \quad \forall i y^T \alpha = 0$. Please see [15] for a detailed derivation. In this form, α is a vector of the Lagrange multipliers, Q is a matrix where $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, and e is a vector of ones. The kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ can vary and extend the SVM to find non-linear boundaries, where the optimal margin hyperplane is computed in a feature space non-linearly related to the input data space.[15] The kernel function selected was the RBF kernel, which refers to $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$ where $\gamma > 0$. The value of γ determines the influence of the distance of the data points, with a small γ will giving a wide influence range to each data point and vice versa.

Once the separating hyperplane is learned from the training data, it corresponds to a decision function that can be used for classifying a test example based on the sign of $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x}_i \rangle + b$.[14,1] The SVM with the RBF kernel should then be able to find complex decision boundaries regardless of the original dimensionality of the data.

2.3 Pipelines

The low-dimensional version of the imaging data included Regions of Interest of Grey Matter ("GM") scaled for Total Intracranial Volume ("TIV"), with 284 features. The high-dimensional version of the dataset included GM 3D maps of voxels in the Montreal Neurological Institute ("MNI") space, masked to provide a flattened representation of 331,695 voxels for each patient. Given that logistic regression and SVM models are sensitive to the scale of input features, both versions of the datasets were standardized by removing the mean and scaling to unit variance. The same scaling method was applied to both datasets to ensure it did not obscure comparability of the study targets.

Two CV pipelines were explored, to ensure that the hyperparameters used to optimize the model's performance were not overly dependent on the particular data splits. Both pipelines were run on the training data as split by RAMP.[2] The first pipeline ("common CV") utilized five-fold CV, whereby the dataset was split into five equal folds. The model was then trained five times, each time using four of the folds for training and the remaining fold for validation. The performance of the model on the validation set was averaged over the five folds. While all comparison metrics from 2.4 were tracked, performance was optimized for the ROC AUC.

The second pipeline, ("group CV") used group stratified two-fold CV, grouped by sex. Each group appears exactly once in the test set across all folds, and the folds preserve the percentage of samples for healthy/patients with schizophrenia to the extent possible.

RandomSearchCV was incorporated in each of the CV strategies, with the hyperparameter ranges in Tables 3, 4, and 5. RandomSearchCV randomly selects a subset of the hyperparameter space and evaluates the model performance for each selection. In this study, this is evaluated for 10 hyperparameter combinations for each model, with each combination applied in each of the CV folds. While less exhaustive than GridSearchCV, RandomSearchCV is often able to find comparable outcomes in an efficient manner.[3] Notably, tuning of GBDT methods may be more effective if done in an iterative

manner.[13] However this was not done due to computational and time constraints.

The best performing models selected (and trained within) each pipeline were applied to the test set presplit by RAMP.[2] The best performing model on the test set was uploaded to the RAMP challenge website, namely the logistic regression model applied to the high-dimensional dataset using the common CV strategy.

2.4 Comparison metrics

Classifier performance was assessed using the area under the Receiver Operating Characteristics curve ("ROC AUC"). This metric considers both the true positive rate ("TPR"), also known as sensitivity, and the false positive rate ("FPR"), also known as specificity, for different threshold values: $\text{TPR} = \frac{\text{TP}}{\text{TP+FN}}$, and $\text{FPR} = \frac{\text{FP}}{\text{FP+TN}}$. The ROC AUC gives a single scalar value to summarize the trade-off between sensitivity and specificity. An ROC AUC of 1.0 means the classifier has perfect sensitivity and specificity, while an ROC AUC of 0.5 indicates that the classifier is no better than random guessing.

Given the dataset was relatively balanced, each classifier's accuracy was also considered, calculated as the number of correct predictions divided by the total number of predictions, $\frac{\text{TP+TN}}{\text{TP+FP+TN+FN}}$.

The training time was tracked as a proxy for computational intensiveness, given it was the most computationally demanding aspect of running the models in the study.

3 Results

The hyperparameters associated with the best performing models from each pipeline can be seen in Table 6. The CV performance of each model is shown in Table 1, displaying the mean and standard deviation of the ROC AUC, accuracy, and training time of each model across validations sets in all folds.

Table 2 reports the ROC AUC and accuracy of the best performing models from CV, when predicting on the test set. It also includes the mean and standard deviation of the training time of the best performing models during CV.

4 Discussion

4.1 CV performance

Note that "performance" is used to refer to ROC AUC and accuracy, and train time is addressed separately.

High-dimensional dataset: Logistic regression and LightGBM performed in almost the same manner when trained using the common CV strategy. Logistic regression had a slightly higher ROC AUC and slightly lower accuracy than LightGBM, and both had similar degrees of variation. When trained using the group CV strategy, LightGBM's performance dropped and was much lower than logistic regression, with more variation. Regardless of CV strategy, the SVM did not perform well on the high-dimensional dataset. However, across both CV strategies the SVM was significantly faster, and logistic regression was by far the most computationally intensive model.

Low-dimensional dataset: When using the common CV strategy, the LightGBM model performed best on both metrics. The SVM achived a better ROC AUC than logistic regression, but worse accuracy. When the group CV strategy was utilized, logistic regression performed best on both metrics, followed by a better ROC AUC by the SVM, but better accuracy from LightGBM. The accuracy of the SVM using the group CV strategy was quite low compared to the other models. The LightGBM model was associated with higher standard deviations regardless of strategy. The training times were broadly similar across all models, albeit slightly higher for LightGBM in both CV strategies.

These trends suggest that LightGBM has strong sensitivity to the partitioning of data, regardless of dataset dimensionality. It may be overfitting to sex-specific features. While the SVM was extremely efficient in handling both datasets, despite use of the RBF kernel the model struggled to form an accurate decision boundary on the high-dimensional data. This could be due to the complexity required, overfitting, or the sparsity of data in the high-dimensional space. Overall, logistic regression appears to be associated with the most stable performance across datasets and CV strategies.

4.2 Test performance

The performance of the best estimators from the CV pipelines exhibited slightly different trends when applied to the test set. Given the relative stability in performance exhibited by logistic regression during CV, it is not too surprising that this model achieved the best, or equivalent, performance across all data dimensionalities and CV strategies. Overall, the different CV strategies made little impact in the high-dimensional context, whereas the group CV strategy was associated with better performance in the low-dimensional context.

High-dimensional dataset: Regardless of CV strategy, logistic regression achieved the best performance, followed by LightGBM. The SVM did not do well on this dataset, although it was the most computationally efficient.

Low-dimensional dataset: The models achieved almost identical performance when utilizing the common CV strategy. When using the group CV strategy, logistic regression achieved the best performance, followed by

Table 1. Mean and standard deviation CV results

Model	Data dimension	CV strategy	ROC AUC (%)	Accuracy (%)	Train time (s)
Logistic regression			80.9 ± 0.7	71.8 ± 1.4	961.0 ± 56.2
LightGBM	High	Common	80.6 ± 0.8	72.2 ± 1.0	483.9 ± 71.3
SVM			54.3 ± 2.0	54.9 ± 0.1	28.1 ± 2.4
Logistic regression			78.7 ± 0.3	67.1 ± 0.7	1202.3 ± 218.9
LightGBM	High	Group	68.9 ± 4.1	58.8 ± 2.1	133.1 ± 39.9
SVM			56.7 ± 3.3	41.7 ± 2.4	10.9 ± 0.6
Logistic regression			76.6 ± 1.2	69.5 ± 1.8	1.6 ± 0.1
LightGBM	Low	Common	80.0 ± 2.4	71.7 ± 2.2	2.4 ± 0.1
SVM			77.6 ± 2.4	66.7 ± 1.0	1.1 ± 0.1
Logistic regression			78.3 ± 1.5	70.8 ± 1.3	1.4 ± 0.1
LightGBM	Low	Group	72.5 ± 5.3	63.9 ± 3.9	1.9 ± 0.1
SVM			76.4 ± 2.2	54.4 ± 2.0	1.1 ± 0.0

Table 2. Best performing estimator test results, and train time of best estimators from CV

Model	Data dimension	CV strategy	ROC AUC (%)	Accuracy (%)	Train time (s)
Logistic regression			76.7	76.7	621.0 ± 2.6
LightGBM	High	Common	74.8	74.4	456.7 ± 112.1
SVM			52.4	50.4	29.1 ± 0.2
Logistic regression			76.7	76.7	1577.0 ± 494.0
LightGBM	High	Group	73.8	73.6	261.2 ± 8.0
SVM			52.4	50.4	11.7 ± 4.9
Logistic regression			72.8	72.4	1.2 ± 0.1
LightGBM	Low	Common	72.8	72.2	4.6 ± 0.1
SVM			72.8	72.2	1.1 ± 0.0
Logistic regression			74.8	74.1	1.2 ± 0.0
LightGBM	Low	Group	73.8	73.5	2.0 ± 0.2
SVM			71.8	71.1	1.2 ± 0.0

LightGBM. Logistic regression and the SVM had similar computational requirements on this dataset across all CV strategies, followed by LightGBM.

4.3 Applied context

Consideration should be given to what levels of performance are "good enough" to be deployed in a clinical context. In cases where computational constraints apply, the results indicate that any of the three models could be utilized on the low-dimensional dataset with fairly comparable performance. In order to use the model associated with the *best* performance, using the high-dimensional dataset, it would be useful to obtain more robust results through assessing the models on several test sets e.g. using a nested CV strategy. While there is no conclusive trend as to the impact of the different CV strategies, the differences suggest that there are characteristics of schizophrenia that differ by patient sex. This should be considered if predictive models were to be trained on an imbalanced (by sex) dataset in the future.

5 Conclusion

In this work we investigated the performance of different feature sets, models and CV strategies on the classification of patients with schizophrenia. We found that a relatively simple linear model, logistic regression, performed at least as well if not better than the other models on the test set in all scenarios. While it performed best on the high-dimensional dataset, it was also associated with higher computational intensity. This presents an interesting trade off between performance and computational intensity. There were mixed trends related to the different CV strategies, which had little impact in the high-dimensional context but improved performance in the low-dimensional scenarios. Overall, the slightly different trends during CV as opposed to the when models were applied to the test set, and relatively similar performance of the three models on the low-dimensional training set, suggest that further testing could be explored for more robust model assessment.

6 Appendix: Tables and figures

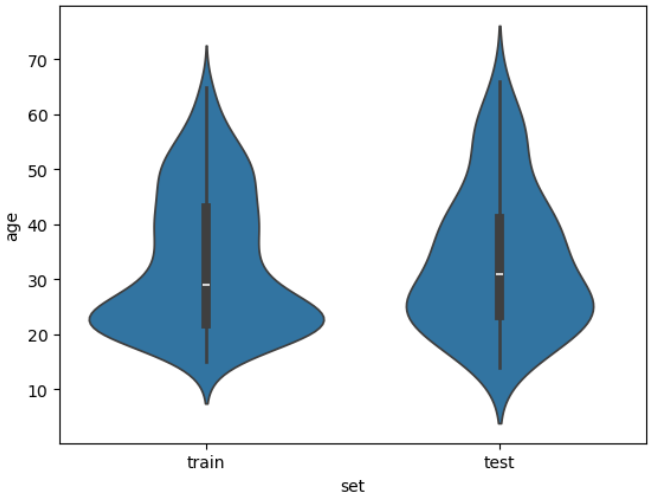

Fig. 1. Distribution of age in train and test set

Table 3. Hyperparameters tuned for logistic regression

Hyperparameter	**Description**	**Range/Values**
C	Inverse of regularization strength	0.01 to 100 (log scale, 5 values*)
fit_intercept	Whether to calculate the intercept	True, False
l1_ratio	The Elastic-Net mixing parameter	0 to 0.1 (5 values*)
max_iter	Maximum number of iterations for the solvers to converge	1000, 5000, 10000

Table 4. Hyperparameters tuned for LightGBM

Hyperparameter	**Description**	**Range/Values**
num_leaves	Number of leaves in the tree	15 to 33 (step 5)
max_depth	Maximum tree depth	5 to 11 (step 2)
learning_rate	Learning rate	0.01 to 1 (log scale, 3 values*)
n_estimators	Number of boosted trees to fit	200 to 1100 (step 200)
min_data_in_leaf	Minimum number of data in one leaf	20 to 100 (step 20)
max_bin	Maximum number of bins for feature values	200 to 255 (step 10)
feature_fraction	Fraction of features to be used at each split	0.6 to 1.0 (5 values*)
bagging_fraction	Fraction of data to be used for bagging	0.7 to 1.0 (4 values*)
bagging_freq	Frequency for bagging	10 to 50 (step 20)
force_col_wise	Whether to force column-wise calculation	True, False

Table 5. Hyperparameters tuned for SVM

Hyperparameter	Description	Range/Values
C	Penalty parameter C of the error term ($\frac{1}{\lambda}$)	0.1 to 100 (log scale, 4 values*)
gamma	Kernel coefficient for 'rbf'	0.00001 to 0.1 (log scale, 5 values*)

*Values are evenly spaced

Table 6. Best hyperparameters selected during CV

Model	Data dimension	CV	Best Hyperparameters
Logistic regression	High	Common	'max_iter': 1000, 'l1_ratio': 0.1, 'fit_intercept': True, 'C': 0.01
	High	Group	'max_iter': 10000, 'l1_ratio': 0.07500000000000001, 'fit_intercept': False, 'C': 0.01
	Low	Common	'max_iter': 10000, 'l1_ratio': 0.025, 'fit_intercept': False, 'C': 0.01
	Low	Group	'max_iter': 5000, 'l1_ratio': 0.0, 'fit_intercept': True, 'C': 0.01
LightGBM	High	Common	'num_leaves': 30, 'n_estimators': 200, 'min_data_in_leaf': 40, 'max_depth': 7, 'max_bin': 230, 'learning_rate': 0.1, 'force_col_wise': True, 'feature_fraction': 0.8, 'bagging_freq': 50, 'bagging_fraction': 1.0
	High	Group	'num_leaves': 25, 'n_estimators': 1000, 'min_data_in_leaf': 40, 'max_depth': 7, 'max_bin': 210, 'learning_rate': 0.1, 'force_col_wise': False, 'feature_fraction': 0.8, 'bagging_freq': 30, 'bagging_fraction': 1.0
	Low	Common	'num_leaves': 15, 'n_estimators': 1000, 'min_data_in_leaf': 20, 'max_depth': 7, 'max_bin': 210, 'learning_rate': 0.1, 'force_col_wise': True, 'feature_fraction': 0.9, 'bagging_freq': 10, 'bagging_fraction': 1.0
	Low	Group	'num_leaves': 15, 'n_estimators': 200, 'min_data_in_leaf': 40, 'max_depth': 9, 'max_bin': 210, 'learning_rate': 0.1, 'force_col_wise': True, 'feature_fraction': 1.0, 'bagging_freq': 50, 'bagging_fraction': 0.9
SVM	High	Common	'gamma': 1e-05, 'C': 10.0
	High	Group	'gamma': 1e-05, 'C': 100.0
	Low	Common	'gamma': 0.0001, 'C': 10.0
	Low	Group	'gamma': 1e-05, 'C': 100.0

References

1. 1.4. Support Vector Machines — scikit-learn 1.4.1 documentation, https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation
2. Brain Anatomy Schizophrenia UCL 2024 Description - RAMP, https://ramp.studio/events/brain_anatomy_schizophrenia_UCL_2024
3. Comparing randomized search and grid search for hyperparameter estimation — scikit-learn 1.4.1 documentation, https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html#
4. Features — LightGBM 4.0.0 documentation, https://lightgbm.readthedocs.io/en/stable/Features.html
5. Parameters Tuning — LightGBM 4.3.0.99 documentation, https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html#deal-with-over-fitting
6. sklearn.metrics.log_loss — scikit-learn 1.4.1 documentation, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html
7. Altmann, A., Mourao-Miranda, J.: Evidence for bias of genetic ancestry in resting state functional MRI. Proceedings - International Symposium on Biomedical Imaging **2019-April**, 275–279 (4 2019). https://doi.org/10.1109/ISBI.2019.8759284
8. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press (3 2000). https://doi.org/10.1017/CBO9780511801389, https://www-cambridge-org.libproxy.ucl.ac.uk/core/books/an-introduction-to-support-vector-machines-and-other-kernelbased-learning-methods/A6A6F4084056A4B23F88648DDBFDD6FC
9. Hastie, T., Tibshirani, R., Friedman, J.: Elements of Statistical Learning: Data Mining, Inference, and Prediction (2009)
10. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: LightGBM: A Highly Efficient Gradient Boosting Decision Tree https://github.com/Microsoft/LightGBM.
11. Montazeri, M., Montazeri, M., Bahaadinbeigy, K., Montazeri, M., Afraz, A.: Application of machine learning methods in predicting schizophrenia and bipolar disorders: A systematic review. Health Science Reports **6**(1) (1 2023). https://doi.org/10.1002/HSR2.962, /pmc/articles/PMC9795991//pmc/articles/PMC9795991/?report=abstracthttps://www.ncbi.nlm.nih.gov/pmc/articles/PMC9795991/
12. Mourao-Miranda, J.: Lecture Week 3: Linear Models, Regularisation and Kernel Methods. In: COMP0189 Applied Artificial Intelligence (2024)
13. Mourao-Miranda, J.: Lecture Week 5: Trees and Ensemble Models. In: COMP0189 Applied Artificial Intelligence (2024)
14. Mourão-Miranda, J., Friston, K.J., Brammer, M.: Dynamic discrimination analysis: A spatial–temporal SVM. NeuroImage **36**(1), 88–99 (5 2007). https://doi.org/10.1016/J.NEUROIMAGE.2007.02.020
15. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. The MIT Press (2018). https://doi.org/10.7551/MITPRESS/4175.001.0001, https://direct-mit-edu.libproxy.ucl.ac.uk/books/book/1821/Learning-with-KernelsSupport-Vector-Machines