1. INTRODUCTION

The human body has close to ~400 odorant receptors. These receptors are activated by an infinite number of ligands. The wet lab methods to determine the possible activating ligands is an expensive process. Machine Learning is known to decrease the search space for such problems, by suggesting the set of possible activating ligands.

In this project, we aim to predict activating ligands for a given receptor using sophisticated Machine Learning techniques. We have also developed a python library to streamline this whole process for the future. The developed library aims to reduce a significant amount of development time.

2. Related Work

Caroline Burshid [3] in his work has laid down the workflow that can be followed to predict novel ligands for an Olfactory receptor. In his work, he has predicted ligands for two human olfactory receptors and one mouse receptor using Machine Learning. Jabeen et al. [2] in their work have used a similar Machine Learning approach to predict novel ligands for OR1G1.

3. Data Set

"Machine Learning is, after all, Data-Driven AI and Your Model will only be as good or bad as the data you have"

Even the most advanced Machine Learning techniques cannot work in the absence of data. The data set was collected from the scientific literature by surveying more than a thousand research papers. The major bottleneck in making predictive models was the scarcity of data in this domain. Nevertheless, we were able to gather sufficient data for four Olfactory Receptors to proceed with the machine learning workflow. The collected dataset for each receptor has the following columns: Odorant/Ligand, SMILE and Activation Status. The Activation Status column has 1 for Activating Ligand and 0 for Non-Activating Ligand.

An open-source software 'Padel' was used to generate chemical descriptors using Smiles.

SMILES stands for Simplified molecular-input line-entry system. It is a specification in the form of a line notation for describing the structure of chemical species using short ASCII string. Padel can generate ~1875 1-D and 2-D descriptors for a given SMILE string. **Table 1** shows the record of data collected.

Receptor Name	**Number of Positive Ligands**	**Number of Non-Activating Ligands**
OR1E1	40	133
OR2J2	18	81
OR1A1	76	268
OR2W1	75	218

Table 1

4. Methodology

The following steps were taken to perform the classification task.

Step 1: Collect data from the literature available: Collect activating and non-activating ligands for receptors.

Step 2: Calculate chemical descriptors for the ligands using their SMILES. There are a number of popular open-source software like Padel and Mordred. Alva Desc has also been used previously [3].

Step 3: Split the data set into train and test set (hold outset). In this case, we have used a 70-30 split i.e. 70% of the data was used for training and 30% for testing. The train set was used to perform 5 fold cross-validation and tune the model parameters. The model that gave the best performance (F1-score) on the holdout set was chosen to make predictions on the unknown test set.

Step 4: A test set of unknown ligands that can possibly be Activating Ligands is prepared using the HMDB database. Owing to the limitation of Machine Learning Algorithms of identifying similar patterns [3] only those metabolites from the HMDB database that were above the threshold of 0.80 Tanimoto similarity score when compared with the Activating ligands, were selected to be a part of the final test set.

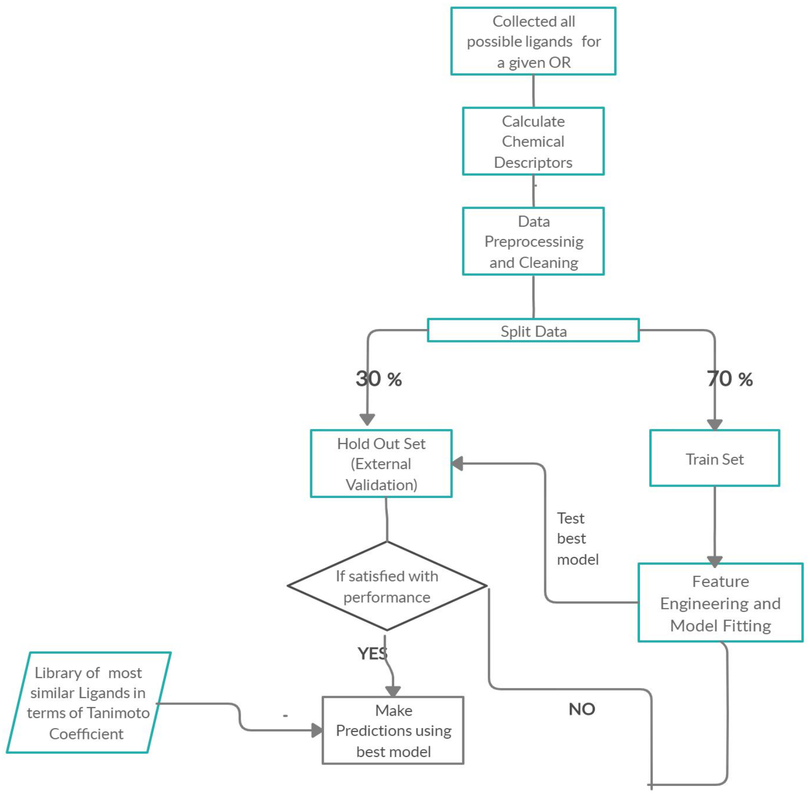

Collected all possible ligands for a given OR
Calculate Chemical Descriptors
Data Preprocessinig and Cleaning
Split Data
30 %
70 %
Hold Out Set (External Validation)
Train Set
Test best model
If satisfied with performance
Feature Engineering and Model Fitting
YES
NO
Library of most similar Ligands in terms of Tanimoto Coefficient
Make Predictions using best model

Figure 1

4.1 Data Preprocessing

Data Preprocessing is a key step in any Machine Learning Workflow. The collected dataset often has missing values, redundant features and features in different ranges. Using the dataset directly without performing any preprocessing will lead to useless results. Hence, it is important to perform data preprocessing. The following steps were taken to preprocess the data set:

- **Handling Missing Values**: Certain chemical descriptors might have missing values for certain Odorant/Ligand. Those features for which more than 25% of the Ligands did not carry any value were discarded and others were imputed using Mean Value Imputation.
- **Normalizing Data**: Normalizing data brings values in each column of the dataset to a common scale. We have used min-max normalization in this case.

  Min-Max normalization transforms **X to Y** :

  $$Y = (X-min)/(max-min)$$

- **Variance Filter:** Certain Features that have no or very little variance do not provide any information to perform classification tasks. Hence, it is important to remove such features for the sake of computational complexity. All those features which have zeros variance are hence, removed.
- **Correlation Filter:** Features having a high correlation with some other feature(s) do not provide any new information and are redundant. Correlation filter keeps one of the many correlated features.

4.2 Feature Selection and Extraction

We have used **Principal Component Analysis for feature extraction and Boruta for feature selection**. Boruta enjoys supremacy over traditional feature selection algorithms in the case when the data set has a large number of features.

Boruta: Boruta works in the following steps[6]:

- ***Creating Shadow Features***: It adds randomness to data by creating shuffled copies of all features. These features are also called shadow features.
- ***Training a Random Forest Classifier:*** A random forest classifier is then trained

on this extended data set and applies a feature importance measure to evaluate the performance of each feature.

- ***Steps taken at each Iteration:*** At every iteration, it checks whether a real feature has more significance than best of its shadow features. In the process, it keeps on removing features that are not important.
- ***Stopping stage:*** The algorithm stops when either all features are rejected or accepted or a subset of those features has been selected.

PCA: Principal Component Analysis [7] is a statistical method for feature extraction. It calculates the projection of the original data onto equal or lower dimensions. Following are the steps taken to perform PCA:

- **Calculating Mean**: Mean is calculated for each dimension and is subtracted from each value. This will ensure that the data set has mean zero.
- **Calculating the Covariance Matrix**
- **Eigen Value and Eigen Vectors are calculated for the Covariance Matrix**
- **Eigen Vectors are ordered according to their Eigen Values:** The Eigen Vectors having higher Eigen Value hold more importance than the lower ones.
- **Finally, the old dataset is projected onto the chosen Eigen Vectors.**

4.3 Handling Data Imbalance

Imbalance Classes is a common problem in Machine Learning tasks. Imbalance classes is a problem because most of machine learning algorithms work best when all classes are somewhat equally represented in the datasets. The following sampling-based techniques are popularly used to counter this problem.

- **Ignoring the problem**
- **Undersampling**: In undersampling based approach, the data points belonging to the majority class are undersampled (discarded) randomly. This technique is easy to use and is computationally less expensive. The **drawback** of this technique is that we may end up losing valuable information and for a case like ours where we have very limited data, it does not make sense to discard any of it.
- **Oversampling**: In oversampling based approaches, the data points belonging to the majority class are randomly oversampled (repeated). This method is prone to

overfitting.

- **SMOTE**: Synthetic Minority Oversampling Technique synthesises samples from the minority classes to create new samples. It is a widely used technique known for its simplicity. We have used SMOTE to handle data imbalance[1].

4.4 Machine Learning Models

We tried using **Gaussian Naive Bayes (GNB), Logistic Regression (LR), Support Vector Machine(SVM), Random Forest(RF), Multi-Layer Perceptron (MLP**) along with an Ensemble-based technique known as **Voting** to make precise predictions. We have used Grid SearchCV in python to tune the parameters of these models and select the best estimator. Overall, it was found that on hold out test set for most receptors Random Forest and Support Vector Machine were found to give better results as compared to other models on the basis of F1-score. The python package developed provides an implementation of all of the above models by calling just a single function.

4.5 Evaluation

In order to evaluate the performance of any Classification based model, there are a number of evaluation metrics.

- **Confusion Matrix:** Although Confusion Matrix is not a metric, it provides valuable information. It a tabular visualization of the predicted values of the model vs the ground truth values.
- **Accuracy Score:** Accuracy Score is a simple metric.
  It is given by:

  Accuracy= (TP+TN)/(TP+TN+FP+FN)

  TP: True Positive

  TN: True Negative

  FP: False Positive

  FN: False Negative

- **Precision Score:** It measures the ability of the classifier to not label as positive what is negative. It is given by:

  Precision = TP/(TP+FP)

- **Recall:** It measures the ability of the classifier to retrieve relevant results. It is given by:

  $$Recall = TP/(TP+FN)$$

- **F1-score:** Precision is a measure of exactness of a quantity while recall is a measure of completeness. Depending upon the use case, the more important metric between these two can be chosen. It is intuitive to think about a metric that combines both of them. F1-score does exactly that. It takes the geometric mean of Precision and Recall. It is given by the formula:

  $$\text{F1-score} = (2 \times \text{Precision} \times \text{Recall})/(\text{Precision}+\text{Recall})$$

  We have used F1-score as the primary metric to choose the best model.

- **ROC Curve:** The receiver operating characteristic curve is plot which shows the performance of a binary classifier as a function of its cut-off threshold. It essentially shows the true positive rate (TPR) against the false positive rate (FPR) for various threshold values [4].
- **AUC:** AUC specifies the area under the ROC curve. On a fundamental level the higher the value of AUC, the better is the model. But, there can be a use case where we do not care about a threshold-based metric to evaluate the performance of a model.
- **MCC:** Mathews Correlation Coefficient takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. It is given by the formula:

  $$\text{MCC} = (\text{TP} \times \text{TN})\text{-}(\text{FP} \times \text{FN})/\sqrt{(TP+TN)(TP+FN)(TN+FP)(TN+FN)}$$

4.6 Comparison of Various Similarity Measures

Bajusz et al.[8] in their study have shown that Tanimoto, Dice coefficient, Cosine and Soergel similarity measures are at par with each other. They have also shown experimentally that Distance-based metrics like Manhattan and Euclidean distance are far from optimal. In order to calculate similarity, we have used the Tanimoto coefficient similarity measure. The table given below lists the other similarity measures.

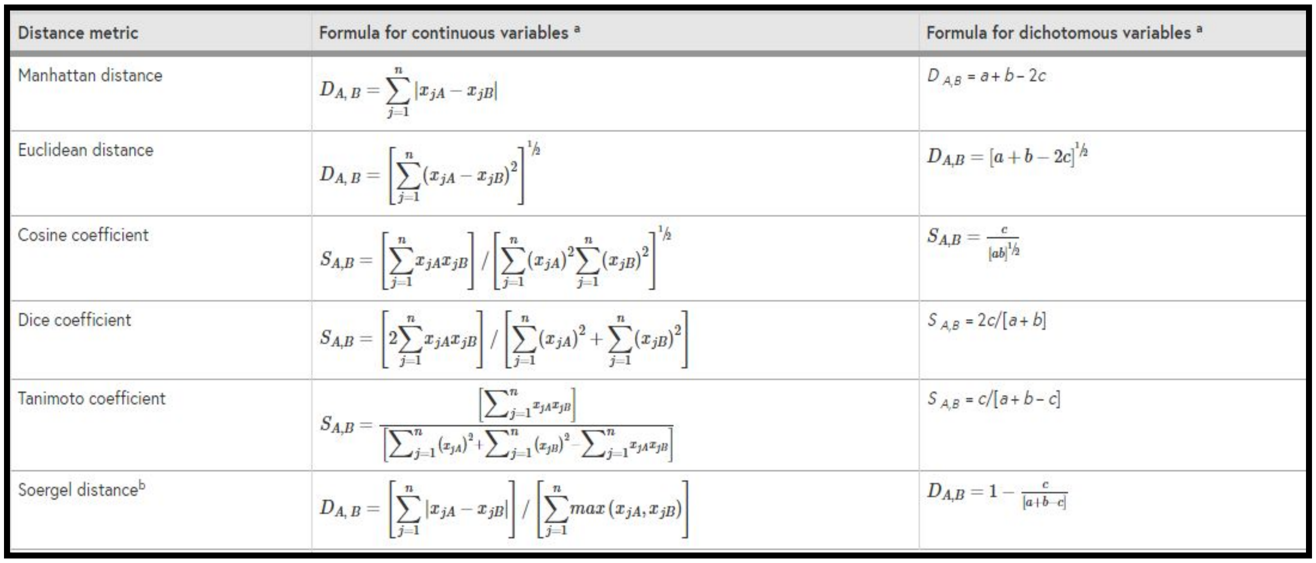
- Tanimoto Coefficient: SMILES are converted to binary encoding. Each bit records the presence or absence of a fragment in the molecule.

Tanimoto coefficient for two molecules A and B is given by:

$T(a,b) = N_c/(N_a+N_b+N_c)$

N_c is the number of common bits set in both A and B

N_a+N_b is the number of bits set in A and B respectively



Distance metric	Formula for continuous variables [a]	Formula for dichotomous variables [a]
Manhattan distance	$D_{A,B} = \sum_{j=1}^{n} \lvert x_{jA} - x_{jB} \rvert$	$D_{A,B} = a + b - 2c$
Euclidean distance	$D_{A,B} = \left[\sum_{j=1}^{n} (x_{jA} - x_{jB})^2\right]^{1/2}$	$D_{A,B} = [a + b - 2c]^{1/2}$
Cosine coefficient	$S_{A,B} = \left[\sum_{j=1}^{n} x_{jA}x_{jB}\right] / \left[\sum_{j=1}^{n} (x_{jA})^2 \sum_{j=1}^{n} (x_{jB})^2\right]^{1/2}$	$S_{A,B} = \frac{c}{[ab]^{1/2}}$
Dice coefficient	$S_{A,B} = \left[2\sum_{j=1}^{n} x_{jA}x_{jB}\right] / \left[\sum_{j=1}^{n} (x_{jA})^2 + \sum_{j=1}^{n} (x_{jB})^2\right]$	$S_{A,B} = 2c/[a + b]$
Tanimoto coefficient	$S_{A,B} = \frac{\left[\sum_{j=1}^{n} x_{jA}x_{jB}\right]}{\left[\sum_{j=1}^{n} (x_{jA})^2 + \sum_{j=1}^{n} (x_{jB})^2 - \sum_{j=1}^{n} x_{jA}x_{jB}\right]}$	$S_{A,B} = c/[a + b - c]$
Soergel distance[b]	$D_{A,B} = \left[\sum_{j=1}^{n} \lvert x_{jA} - x_{jB} \rvert\right] / \left[\sum_{j=1}^{n} max(x_{jA}, x_{jB})\right]$	$D_{A,B} = 1 - \frac{c}{[a + b - c]}$

Source:https://jcheminf.biomedcentral.com/articles/10.1186/s13321-015-0069-3#Sec12

Results

After performing Feature selection using variance and correlation filters and using Boruta, out of the total 1875 features 75-85 features were selected, the performance of Machine Learning Algorithms after fine-tuning were evaluated on these features. Some of these features are : 'RDF20m', ‘ALogP', 'ATSC2m', 'nBondsS2','SpMax1_Bhv'. There is visible variation in features selected across different Receptors. However, it was observed that the performance of all models was better when on features selected by Boruta, PCA was applied conserving 98% of Eigen energy. All of this was done after having synthesised the minority class using SMOTE for training and also once without SMOTE to analyze the difference in results.

Classifier	F1-score	MCC	Precision	Recall	Accuracy	AUC
SVM	0.69	0.39	0.73	0.67	0.80	0.71
GNB	0.59	0.18	0.59	0.60	0.69	0.72
LR	0.52	0.255	0.89	0.54	0.788	0.66
MLP	0.67	0.352	0.70	0.66	0.78	0.80
RF	0.67	0.352	0.70	0.66	0.78	0.71
Voting (RF+MLP+RF)	0.65	0.31	0.67	0.65	0.77	0.77

Table 2: Results on Hold Out set for OR51E1 with Boruta (No SMOTE)

Classifier	F1-score	MCC	Precision	Recall	Accuracy	AUC
SVM	0.67	0.35	0.70	0.66	0.79	0.64
GNB	0.59	0.18	0.59	0.60	0.69	0.72
LR	0.52	0.25	0.89	0.54	0.79	0.66
MLP	0.67	0.35	0.70	0.66	0.79	0.76
RF	0.64	0.27	0.64	0.63	0.75	0.56
Voting (RF+MLP+RF)	0.67	0.35	0.70	0.66	0.79	0.62

Table 3: Results on Hold Out set for OR51E1 with Boruta+PCA (NO SMOTE)

Classifier	F1-score	MCC	Precision	Recall	Accuracy	AUC
SVM	0.67	0.35	0.70	0.66	0.78	0.76
GNB	0.65	0.36	0.65	0.71	0.69	0.72
LR	0.66	0.31	0.65	0.66	0.76	0.79
MLP	0.65	0.311	0.67	0.65	0.76	0.80
RF	0.65	0.31	0.67	0.65	0.76	0.76
Voting (RF+MLP+ RF)	0.65	0.311	0.67	0.65	0.77	0.77

Table 4: Results on Hold Out set for OR51E1 with SMOTE+Boruta

Classifier	F1-score	MCC	Precision	Recall	Accuracy	AUC
SVM	0.67	0.35	0.70	0.66	0.78	0.75
GNB	0.65	0.36	0.65	0.71	0.69	0.72
LR	0.66	0.31	0.65	0.66	0.75	0.79
MLP	0.65	0.311	0.67	0.65	0.76	0.76
RF	**0.73**	**0.47**	**0.76**	**0.71**	**0.82**	**0.71**
Voting (RF+MLP+ RF)	0.69	0.39	0.73	0.67	0.81	0.75

Table 5: Results on Hold Out set for OR51E1 with SMOTE+Boruta+PCA

Classifier	**F1-score**	**MCC**	**Precision**	**Recall**	**Accuracy**	**AUC**
SVM	0.47	0.176	0.87	0.52	0.74	0.562
GNB	0.61	0.22	0.62	0.61	0.71	0.73
LR	0.54	0.23	0.75	0.56	0.76	0.73
MLP	0.57	0.21	0.66	0.57	0.75	0.73
RF	0.59	0.20	0.62	0.59	0.73	0.70
Voting (RF+MLP+ RF)	0.68	0.36	0.69	0.68	0.76	0.78

Table 6: Results on Hold Out set for OR51E1 with Boruta

Classifier	**F1-score**	**MCC**	**Precision**	**Recall**	**Accuracy**	**AUC**
SVM	0.47	0.17	0.87	0.52	0.74	0.56
GNB	0.61	0.22	0.62	0.61	0.736	0.71
LR	0.54	0.236	0.75	0.56	0.76	0.734
MLP	0.57	0.219	0.66	0.57	0.74	0.736
RF	0.49	-0.01	0.49	0.49	0.60	0.55
Voting (RF+MLP+ RF)	0.49	0.074	0.57	0.52	0.73	0.68

Table 7: Results on Hold Out set for OR51E1 with Boruta+PCA

Classifier	**F1-score**	**MCC**	**Precision**	**Recall**	**Accuracy**	**AUC**
SVM	0.71	0.41	0.72	0.70	0.78	0.73
GNB	0.61	0.30	0.63	0.67	0.63	0.78
LR	0.60	0.22	0.60	0.63	0.65	0.73
MLP	0.67	0.33	0.66	0.68	0.73	0.79

RF	0.65	0.30	0.66	0.65	0.73	0.76
Voting (RF+MLP+ RF)	0.72	0.44	0.73	0.71	0.79	0.75

Table 7 : Results on Hold Out set for OR2W1 with SMOTE+Boruta

Classifier	**F1-score**	**MCC**	**Precision**	**Recall**	**Accuracy**	**AUC**
SVM	0.70	0.39	0.70	0.70	0.77	0.73
GNB	0.61	0.30	0.63	0.67	0.64	0.78
LR	0.60	0.22	0.60	0.63	0.65	0.73
MLP	0.70	0.41	0.70	0.72	0.76	0.77
RF	**0.73**	**0.48**	**0.77**	**0.71**	**0.81**	**0.75**
Voting (RF+MLP+ RF)	0.70	0.39	0.70	0.70	0.77	0.73

Table 8: Results on Hold Out set for OR2W1 with Smote+Boruta+PCA

Classifier	**F1-score**	**MCC**	**Precision**	**Recall**	**Accuracy**	**AUC**
SVM	0.67	0.34	0.67	0.67	0.8	0.77
GNB	0.73	0.48	0.71	0.79	0.80	0.864
LR	0.59	0.25	0.72	0.57	0.82	0.86
MLP	0.68	0.369	0.69	0.68	0.81	0.86
RF	0.73	0.47	0.78	0.71	0.85	0.85
Voting (RF+MLP+ RF)	0.73	0.469	0.75	0.72	0.84	0.889

Table 9: Results on Hold Out set for OR1A1 with Boruta

Classifier	F1-score	MCC	Precision	Recall	Accuracy	AUC
SVM	0.69	0.37	0.68	0.70	0.80	0.79
GNB	0.73	0.48	0.71	0.79	0.80	0.86
LR	0.59	0.25	0.72	0.57	0.82	0.86
MLP	0.75	0.52	0.73	0.80	0.86	0.82
RF	0.66	0.38	0.78	0.63	0.84	0.77
Voting (RF+MLP+ RF)	0.73	0.478	0.78	0.71	0.86	0.86

Table 10: Results on Hold Out set for OR1A1 with Boruta+PCA

Classifier	F1-score	MCC	Precision	Recall	Accuracy	AUC
SVM	0.42	-0.133	0.39	0.46	0.733	0.64
GNB	0.71	0.48	0.70	0.79	0.77	0.90
LR	0.68	0.44	0.68	0.77	0.73	0.7847
MLP	0.58	0.16	0.58	0.58	0.73	0.73
RF	0.58	0.16	0.58	0.58	0.733	0.8
Voting (RF+MLP+ RF)	0.40	-0.19	0.38	0.42	0.67	0.76

Table 11: Results on Hold Out set for OR1A1 with Smote+Boruta

Classifier	F1-score	MCC	Precision	Recall	Accuracy	AUC
SVM	0.78	0.59	0.89	0.73	0.88	0.87
GNB	0.71	0.45	0.69	0.77	0.77	0.81
LR	0.71	0.54	0.69	0.77	0.78	0.867

MLP	**0.80**	**0.59**	**0.80**	**0.79**	**0.88**	**0.845**
RF	0.77	0.539	0.76	078	0.855	0.91
Voting (RF+MLP+ RF)	0.76	0.54	0.79	0.74	0.83	0.86

Table 12: Results on Hold Out set for OR1A1 with SMOTE+Boruta+PCA

Classifier	**F1-score**	**MCC**	**Precision**	**Recall**	**Accuracy**	**AUC**
SVM	0.42	-0.133	0.39	0.46	0.73	0.64
GNB	0.71	0.48	0.70	0.79	0.77	0.90
LR	0.68	0.44	0.68	0.77	0.73	0.78
MLP	0.58	0.16	0.58	0.58	0.73	0.73
RF	0.58	0.16	0.58	0.58	0.73	0.80
Voting (RF+MLP+RF)	0.40	-0.19	0.38	0.42	0.67	0.76

Table 13: Results on Hold Out Set for OR2J2 with SMOTE+Boruta

Classifier	**F1-score**	**MCC**	**Precision**	**Recall**	**Accuracy**	**AUC**
SVM	0.42	-0.133	0.39	0.46	0.733	0.687
GNB	0.71	0.48	0.70	0.79	0.77	0.90
LR	0.68	0.44	0.68	0.77	0.73	0.78
MLP	0.61	0.22	0.62	0.60	0.77	0.71
RF	0.79	0.583	0.79	0.79	0.87	0.84
Voting (RF+MLP+RF)	0.58	0.16	0.58	0.58	0.73	0.77

Table 14: Results on Hold Out Set for OR2J2 with SMOTE+Boruta+PCA

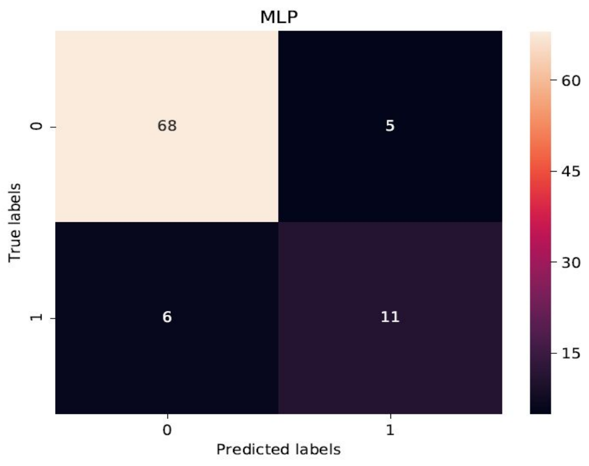


Figure 3: Confusion Matrix for OR1A1 using SMOTE+Boruta+PCA

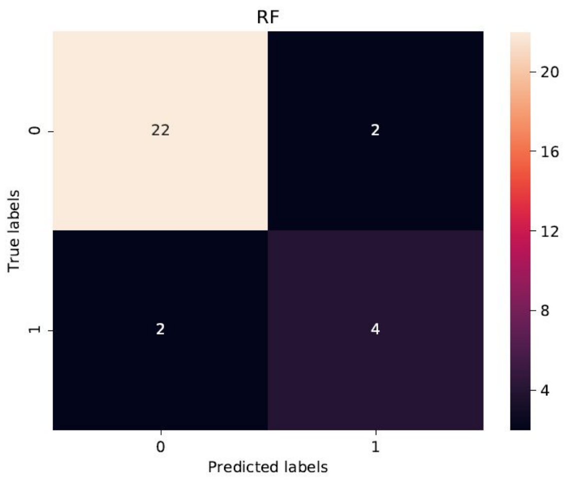


Figure 4: Confusion Matrix for OR2J2 using SMOTE+Boruto+PCA

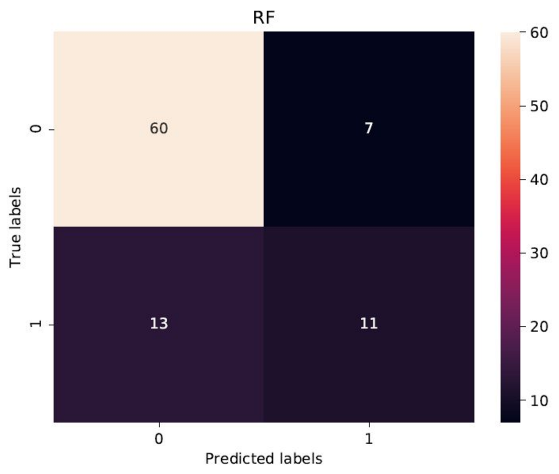


Figure 5: Confusion Matrix for OR2W1 using SMOTE+Boruta+PCA

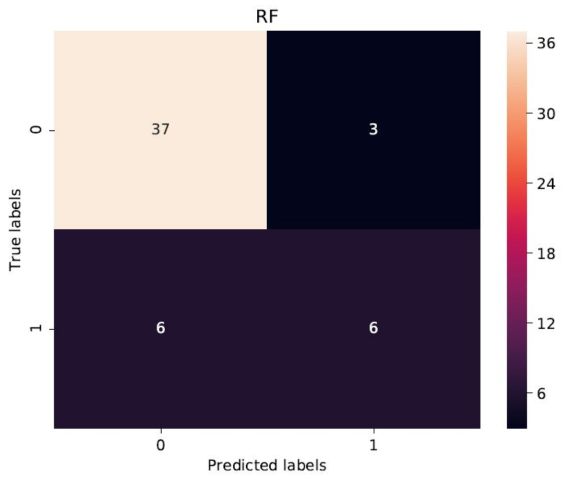


Figure 6: Confusion Matrix for OR51E1 using SMOTE+Borura+PCA

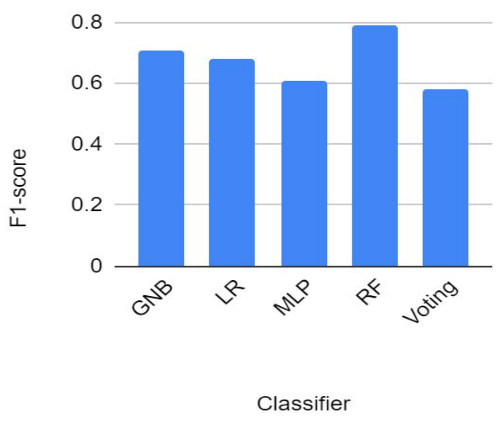


Figure 7: Bar Graph for OR2J2

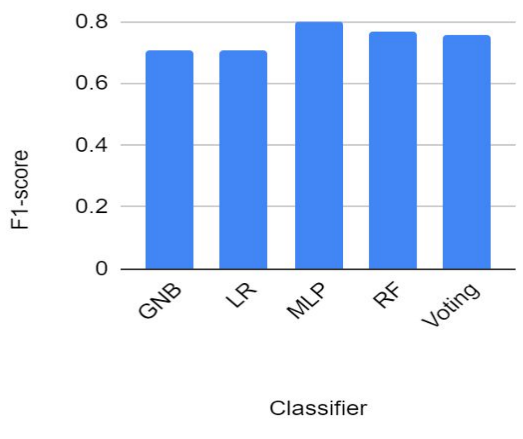


Figure 8: Bar Graph for OR1A1

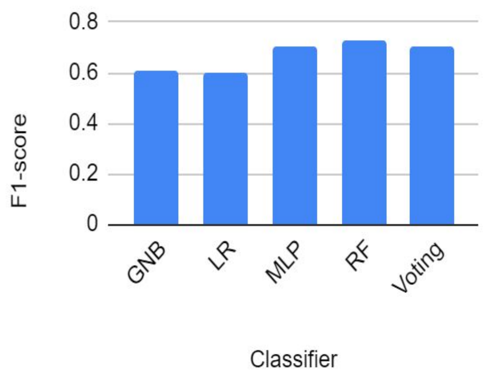


Figure 9: Bar Graph for OR2W1

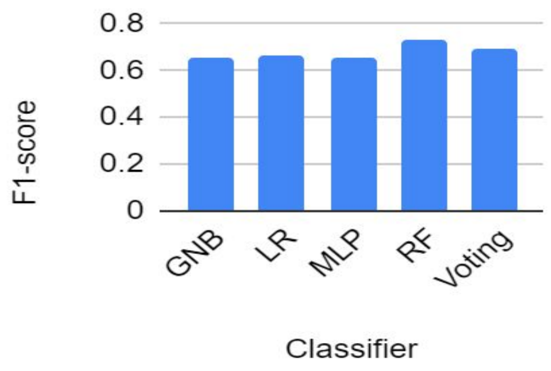


Figure 10: Bar Graph for OR51E1

CONCLUSION

It can be observed from Table 5, 8, 12 and 14 that SMOTE followed by Boruta and PCA gives best results. In the case of OR1A1 Multi-Layer Perceptron with hidden_layer_sizes=(5, 5, 5), activation = ‘relu’, learning rate=0.01 and momentum=0.9 gave best performance (Table 12). For OR51E1 Random Forest with criterion='gini', max_depth=100, max_features='sqrt',in_samples_leaf=2, min_samples_split=2 and -n_estimators=100 gave best performance. OR2W1 gave best results using Random Forest using criterion='gini', max_depth=100, max_features='sqrt', min_samples_leaf=1 and min_samples_split=2. The best results with OR2J2 was achieved using Random Forest with criterion='gini', max_depth=100, max_features='sqrt', min_samples_leaf=1, min_samples_split=2 and n_estimators=78.

REFERENCES

1. *Blagus, R., Lusa, L. SMOTE for high-dimensional class-imbalanced data. BMC Bioinformatics 14, 106 (2013) doi:10.1186/1471-2105-14-106*

2. *Jabeen A., Ranganathan S. Applications of machine learning in GPCR bioactive ligand discovery. Curr. Opin. Struct. Biol. 2019;55:66–76. doi: 10.1016/j.sbi.2019.03.022*

3. *Caroline Bushdid. Numerical modeling of olfaction. Other. Université Côte d'Azur, 2018. English. ffNNT : 2018AZUR4091ff. Fftel-02010618f*

4. *https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce*

5. *https://en.wikipedia.org/wiki/Matthews_correlation_coefficient*

6. *https://www.analyticsvidhya.com/blog/2016/03/select-important-variables-boruta-package/*

7. *http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf*

8. Bajusz, D., Rácz, A. & Héberger, K. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations?. J Cheminform 7, 20 (2015) doi:10.1186/s13321-015-0069-3

9. Yap CW (2011) PaDEL-descriptor: an open-source software to calculate molecular descriptors and fingerprints. J Comput Chem 32: 1466–1474