Extension of Transformational Machine Learning to Classification Problems

Oghenejokpeme I. Orhobor[1], Mahmud M. Adnan[2]

[1] National Institute of Agricultural Botany, 93 Lawrence Weaver Road, Cambridge, CB3 0LE

[2] Clare College, Trinity Ln, Cambridge CB2 1TL

Abstract

In supervised ML (Machine Learning), the ML system learns a model that can predict the labels of unseen samples by generalising from labelled examples. For example, in drug design, ML employs features that represent molecule structure, molecular affinity, and so on. When there are numerous linked ML problems, a new type of feature can be used: predictions made about the instances by ML models learnt on other problems. This method is defined as TML (Transformational Machine Learning) [1].

The key question TML solves is that: *Given that domains share common attributes, how to exploit such similarities to build a composite that performs better than traditional base models?*

It has been proven that TML yields better predictions and improved understanding when applied to the regression analysis [1] of scientific problems. This paper aims to examine the performance of Transformational Machine Learning in classification analysis in the field of drug design.

TML has repeatedly underperformed the fundamental ML algorithm when tested on 100 QSAR datasets. Regardless of the kind of method, as the number of datasets increases, all evaluation metrics demonstrate improvement. Moreover, when datasets are balanced, the performance of all matrices improves.

In addition, it is theorized that the rate of learning is not proportional to the amount of QSAR datasets or that TML does not perform linearly.

[1] Email: orhobor@niab.com
[2] Email: mam255@cam.ac.uk

[1.0] Introduction

[1.1] What is TML?

Given that, TML has a significant amount of resemblance with ensemble machine learning algorithms[3] and in specific to *stacking* and *blending*; it would be ideal to understand TML via building up on the understanding of *stacking* and *blending*.

First of all, training a learning algorithm to integrate the predictions of many different learning algorithms is stacking. Initially, all of the other algorithms are trained using the available data, and then a combiner algorithm is taught to create a final prediction utilising all of the other algorithms' predictions as extra inputs. *Stacking* allows one to use the strength of each individual estimator by using their output as input of a final estimator. The primary distinction between TML and stacking is that, in stacking, several baseline models are typically trained on the same task, but TML learns across a wide number of tasks, each of which may possibly trained with different models.

Second of all, blending aims to create models that have high accuracy without losing interpretability [2]. It is difficult to define (mathematically) interpretability. However, Miller *et al.* [3] provide a (non-mathematical) definition of interpretability: "interpretability is the degree to which a person can grasp the cause of a choice". Another definition is that interpretability is the degree to which a model's outcome can be reliably anticipated [4].

Let there be, n number of datasets (DS) and DS #T (dataset number T) is the target dataset ($T \in n$), i.e., the aim of the TML model is to maximise the performance metrics in this dataset. DS #1 to DS #n (except DS#T) will be trained on their respective datasets by their respective algorithms. The predictions from these algorithms are used as the input variables for target dataset's algorithm. For other datasets similar iterations should be performed, i.e., for n number of dataset, in the first step (n-1) algorithms are trained.

[3] A general meta-approach to machine learning that seeks better predictive performance by combining the predictions from multiple models.

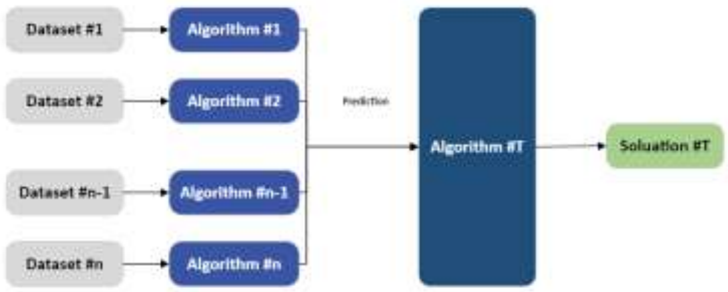


Figure 1: Schematic representation of TML.

[1.2] Why use TML?

One of the key problems with QSAR[4] models is that they are difficult to understand chemically. The machine's incapacity to communicate its ideas and behaviours to researchers restricts QSAR models efficacy. Combining the predictions of a variety of standard interpretable models is one potential technique to improve accuracy without sacrificing interpretability.

However, various ensemble learning models (bagging, boosting, or stacking) may result in different prediction outcomes and feature selections for interpretation. As a result, employing feature importance supplied by a single DT-based ensemble learning model may restrict the generalisation of predictability and interpretability. Given that TML is situated in the eutectic point of stacking, bagging, and blending, it is relatively better suited to mitigate the generalisation problem.

[1.3] Limitations of TML

A limitation of the TML is that it treats all models the same, meaning all models contribute equally to the prediction. This is a problem if some models are good in some situations and poor in others.

[2.0] Imbalance in Classification

[2.1] Overview

Classification predictive modelling is the process of predicting a class label for a given observation. An imbalanced classification issue is a classification problem in which the distribution of instances across the known classes is unequal. The distribution can range from a little bias to a severe imbalance, with one instance in the minority class for hundreds, thousands, or millions in the majority class or classes.

[4] Quantitative Structure Activity Relationship: mathematical models that can be used to predict the physicochemical, biological and environmental fate properties of compounds from the knowledge of their chemical structure.

In supervised machine learning, it is crucial to train an estimator with balanced data so that the model is equally knowledgeable about all classes. Due to the imbalanced distribution of classes, the majority of machine learning algorithms will perform poorly and require adjustment to prevent always forecasting the majority class. Additionally, measures like as classification accuracy lose significance, necessitating other approaches for assessing predictions on imbalanced samples. Otherwise, an imbalanced dataset will lead to models with low prediction accuracy, particularly for minority classes. This is a problem because, in general, the minority class is more significant than the majority class, and the problem is thus more susceptible to classification mistakes for the minority class than for the majority class.

The problem domain from which the examples were derived adds an extra layer of complexity [5]. Typically, the majority class represents a regular instance in the domain, whereas the minority class indicates an aberrant situation, such as a fault, fraud, outlier, anomaly, illness condition, etc. Consequently, misclassification mistakes may be interpreted differently across classes. For instance, misclassifying a member of the majority class as a member of the minority class, known as a false positive, is frequently undesirable, but less critical than misclassifying a member of the minority class as a member of the majority class, known as a false negative. This is known as the cost sensitivity of misclassification mistakes, and it is the second fundamental problem of imbalanced classification. The scenarios investigated in this report will not have this added level of complexity.

[2.2] Model Evaluation Metrics

A classifier is only as effective as the evaluation metric used to assess it. Choosing a suitable measure is tough in applied machine learning in general but is especially problematic for imbalanced classification situations. First, since the majority of frequently used standard metrics assume a balanced class distribution, and second, because not all classes, and hence not all prediction errors, are generally equal for imbalanced classification. Although the datasets used in this report might be exception (as they are naturally equally distributed) and it is safer to choose evaluation models that are prepared for the worst cases.

According to Ferri et al [6], evaluation metrics can be divided into three useful groups:

(1) *Threshold Metrics*: designed to summarize the fraction, ratio, or rate of when a predicted class does not match the expected class in a holdout dataset. One of the most widely used threshold metric is classification accuracy (3.1).

Precision summarizes the fraction of examples assigned the positive class that belong to the positive class (3.2).

Recall summarizes how well the positive class was predicted and is the same calculation as sensitivity (3.3).

Table 1: Definition of the performance matrices used in the classification analysis

$\mathrm{Accuracy} = \frac{\mathrm{Correct\ Predictions}}{\mathrm{Total\ Predictions}}$	(3.1)
$\mathrm{Precision} = \frac{\mathrm{TruePositive}}{\mathrm{TruePositive} + \mathrm{FalsePositive}}$	(3.2)
$\mathrm{Recall} = \frac{\mathrm{TruePositive}}{\mathrm{TruePositive} + \mathrm{FalseNegative}}$	(3.3)

(2) *Ranking Metrics*: concerned with evaluating classifiers based on how effective they are at. ROC Curve or ROC Analysis is the most often employed ranking metric. Receiver Operating Characteristic is an abbreviation that describes a topic of research for examining binary classifiers based on their ability to discern classes.

(3) *Probability Metrics*: built particularly to assess the prediction uncertainty of a classifier. These are beneficial for scenarios in which one is less concerned with erroneous vs accurate class predictions and more concerned with the uncertainty the model has in its predictions, punishing incorrect but highly confident forecasts. Maximum likelihood estimation and Brier Score are examples of such types of evaluation matrix.

[2.3] Dummy Classifier

Dummy Classifier is a classification model that provides predictions without attempting to detect data patterns. The default model produces predictions based on the most prevalent label in the training data.

Dummy classification is used as control classification algorithm.

[2.4] Other Types of Classifiers

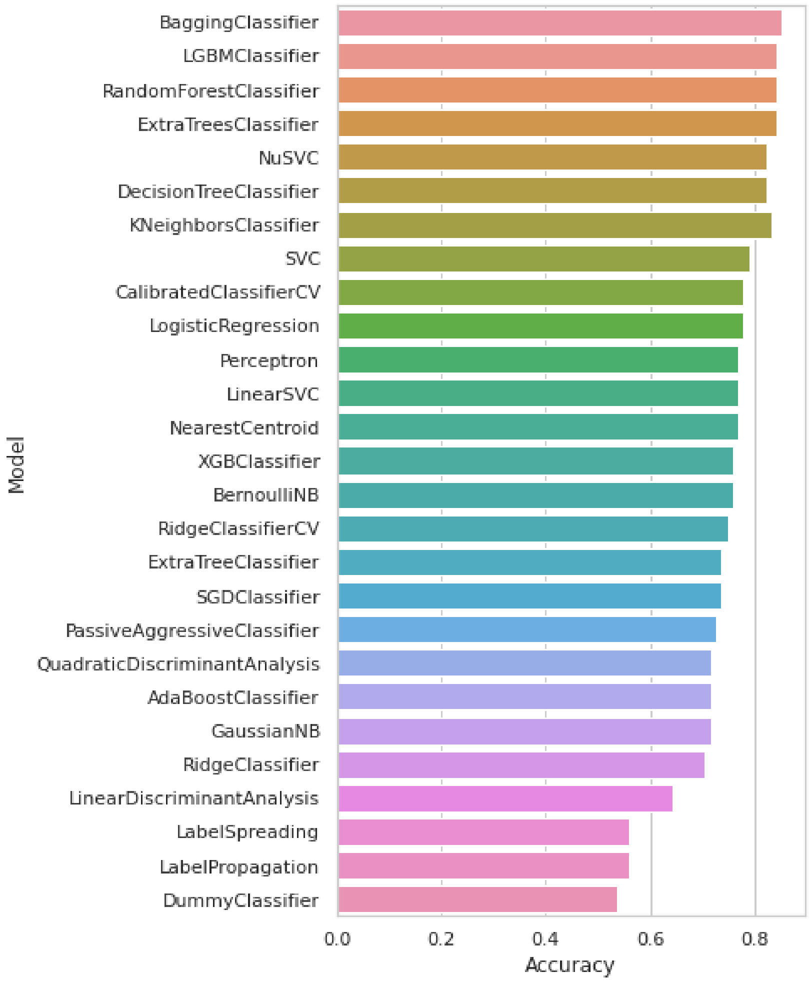
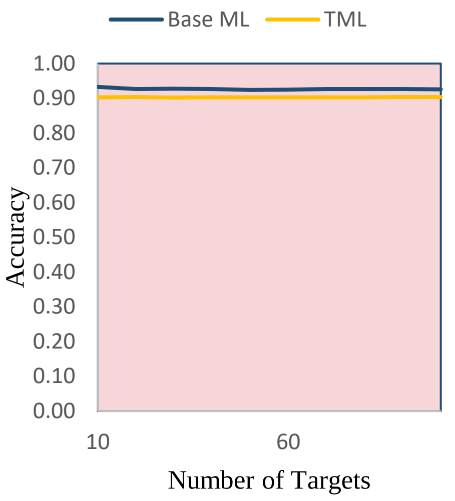


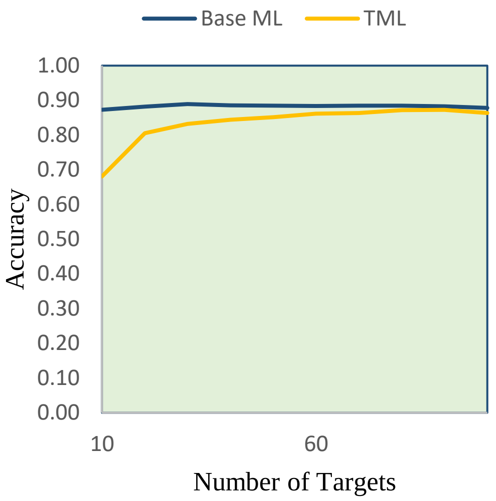
Figure 2: Comparing 27 classification algorithm to under ***Accuracy*** *metric for* ***a*** *QSAR dataset.*

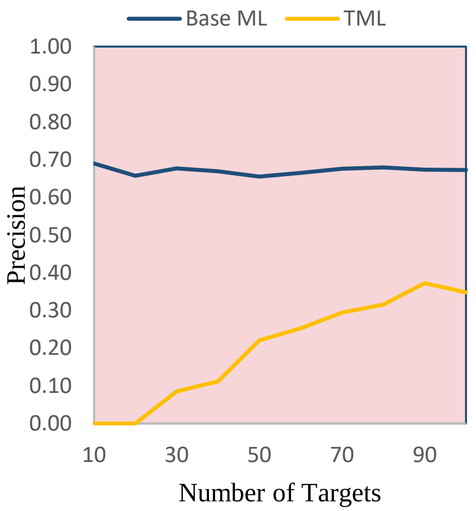
From the array of 27 classification algorithm: Random Forest, XGB Boosts, Bagging Classifier and KNN is selected to further investigate. The classifiers are selected so that they are distributed across the entire performance metric's range. It should be noted that, only accuracy metric is used to make the decision. Ideally, computational time, F1 scores, recall and precision metrics should be taken into consideration.

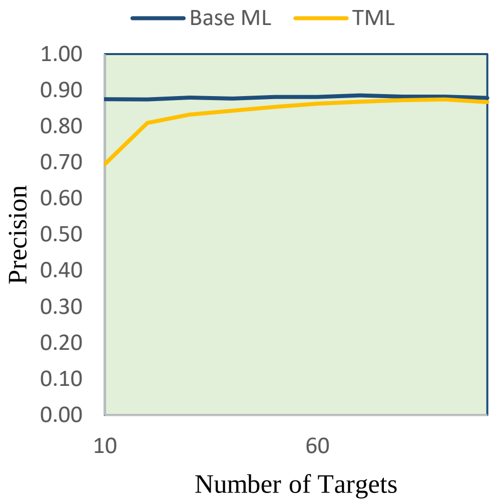
[3.0] Results

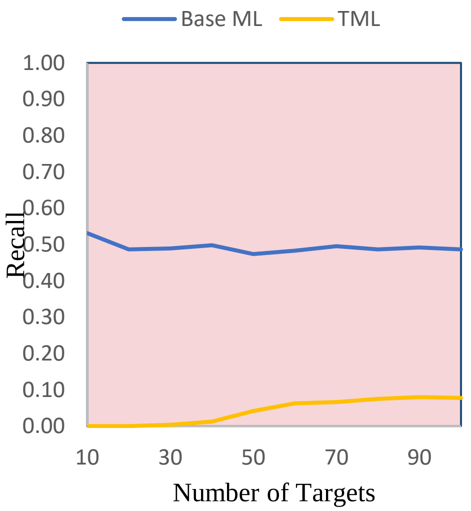
*Table 2: Comparison of balanced and imbalanced dataset under the evaluation metrics of **Accuracy**, **Precision** and **Recall**. The model is trained under **Random Forest** algorithm. Both **Base ML** method and the **TML** method is depicted in the table.*

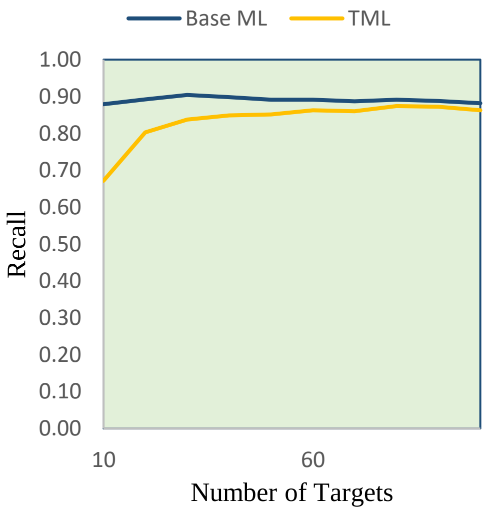
<table>
<tr><th></th><th>Imbalanced</th><th>Balanced (Under Sampled)</th></tr>
<tr><td><b>Accuracy</b></td><td>


</td><td>


</td></tr>
<tr><td><b>Precision</b></td><td>


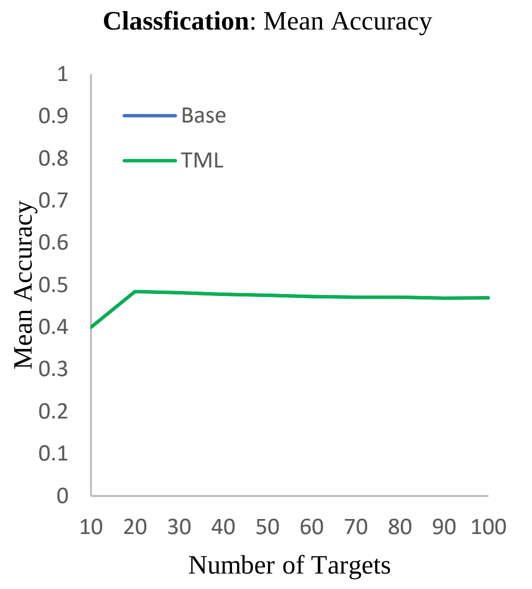
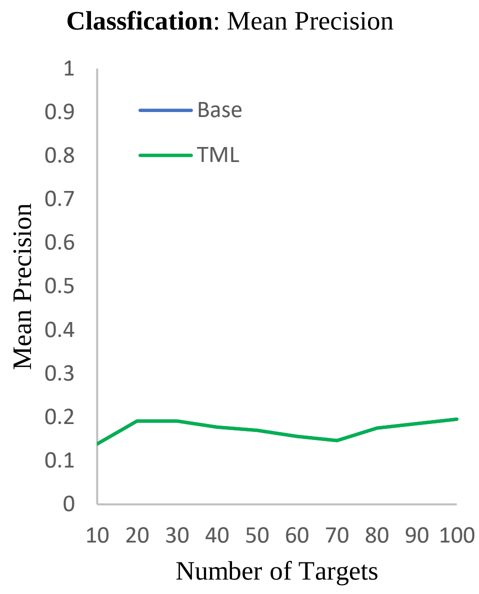
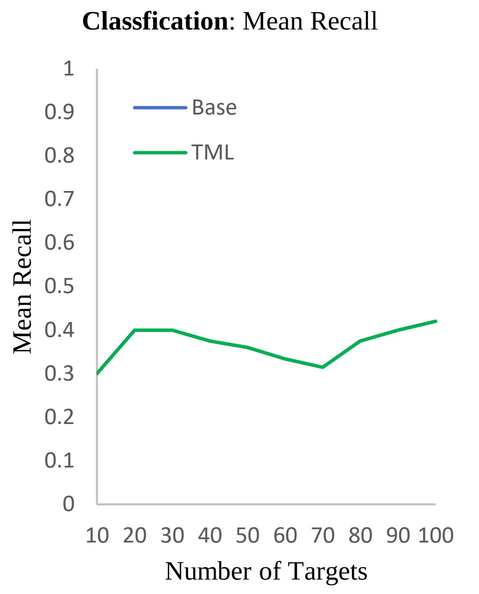
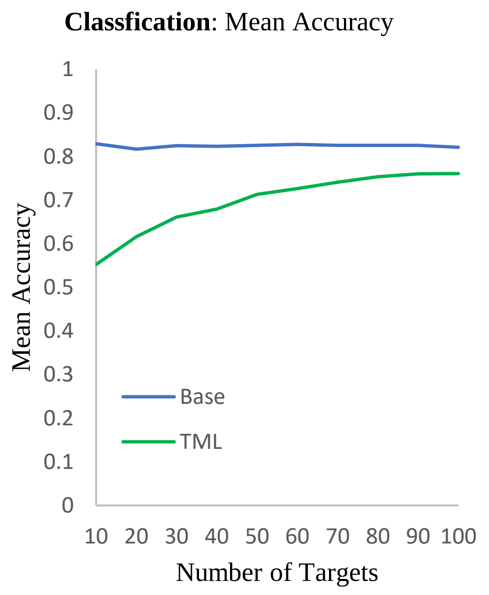
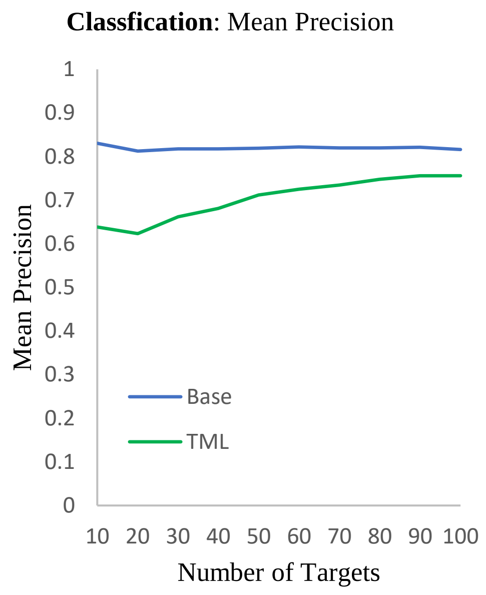
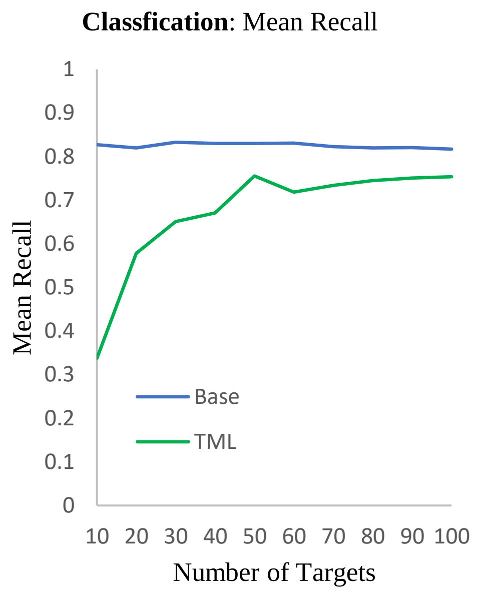
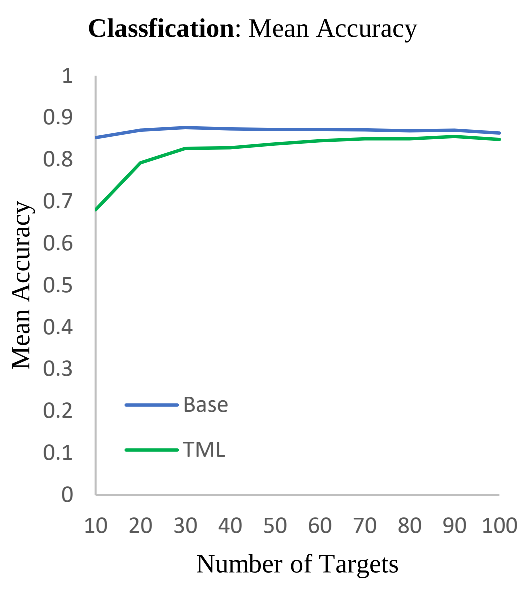
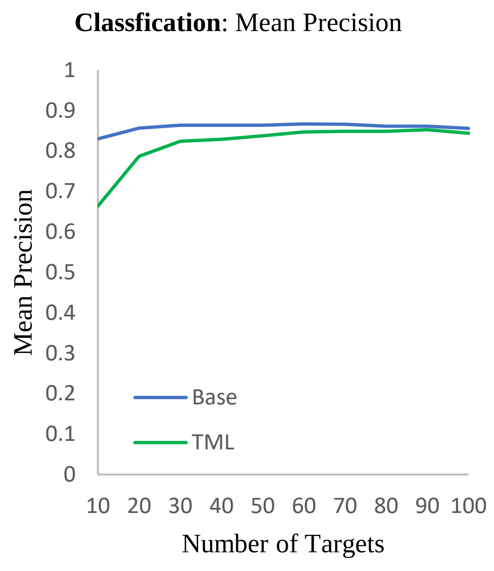
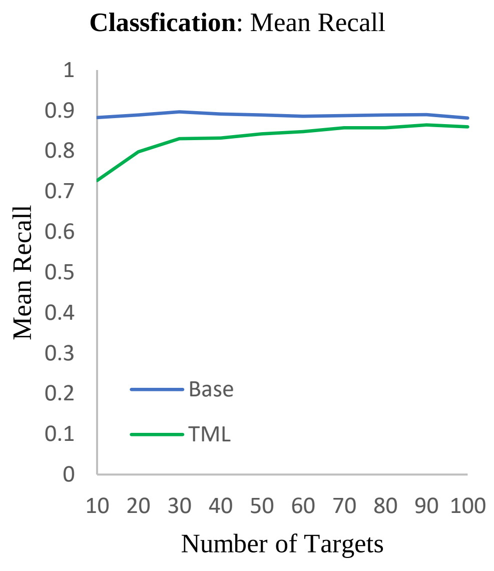
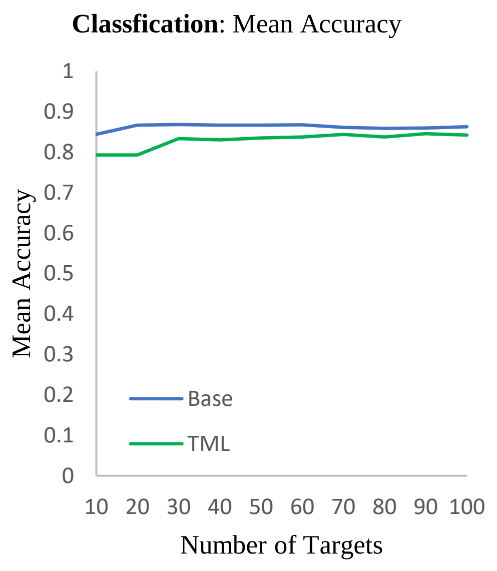
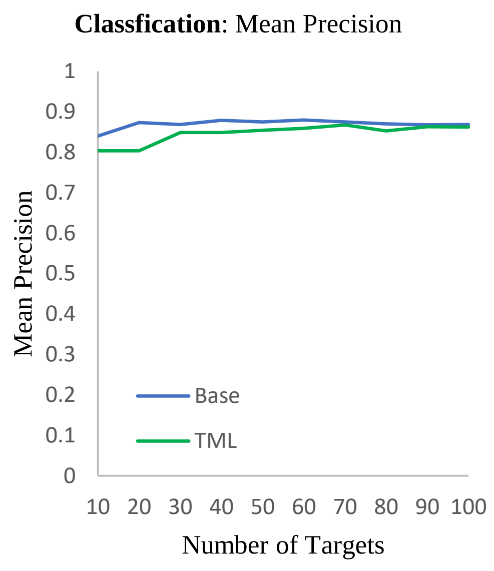
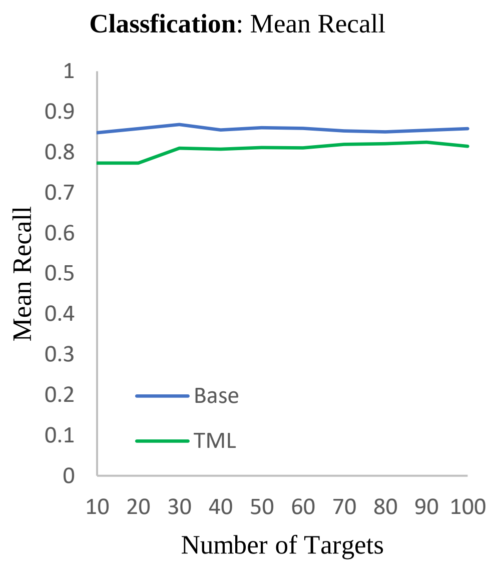
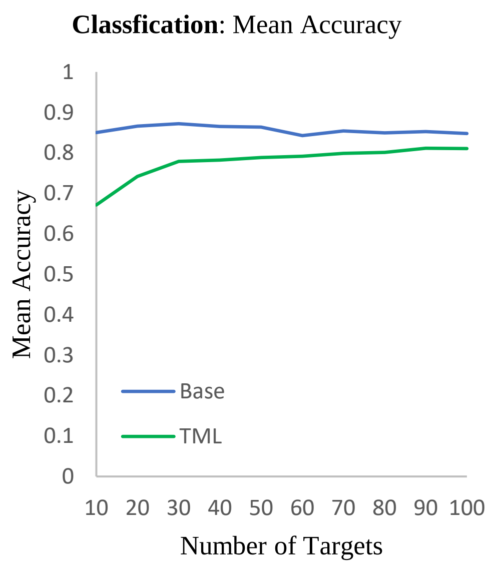
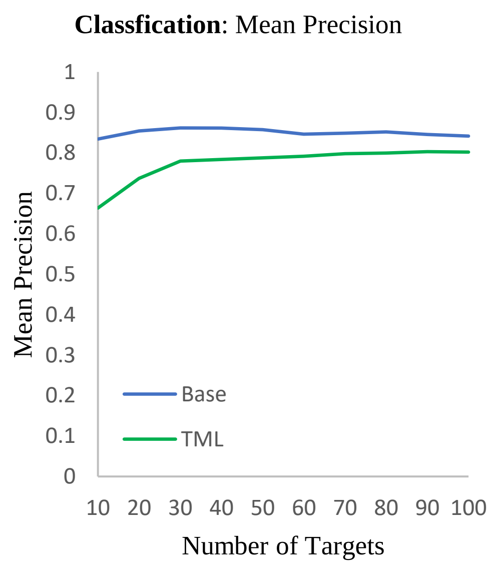
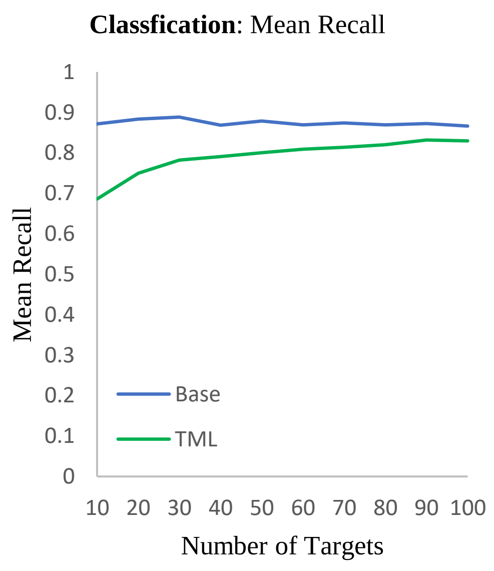
</td><td>


</td></tr>
<tr><td><b>Recall</b></td><td>


</td><td>


</td></tr>
</table>

*Table 3: Comparison of five distinguished classifying algorithms (****Dummy Classifier****,* ***Random Forest****,* ***XGB Boosts****,* ***Bagging Classifier****,* ***KNN Classifier****) under the evaluation metrics of* ***Accuracy****,* ***Precision*** *and* ***Recall****. Both Base ML method and the TML method is depicted in the table.*

Models	**Accuracy**	**Precision**	**Recall**
Dummy Classifier			
Random Forest Classifier			
XGB Boosts			
Bagging Classifier			
KNN Classifier			

[4.0] Analysis

[4.1] Balanced vs Imbalanced Datasets

In the analysis Table 2, the ***Random Undersampling*** method was used to balance the dataset. Random Undersampling entails identifying examples at random from the majority class and removing them from the training dataset.

As expected, when the datasets are balanced all matrices perform better.

It can be seen that there is a kink at the target 90. The experiment was repeated several times and every time a similar behaviour is observed. Further work needs to be done in order to theorise the observation.

For the base model, when undersampled, i.e., the size of individual datasets is reduced, the performance matrices of Precision and Recall increase. This appears to be counterintuitive. Further work needs to be done in order to theorise the observation.

For the TML, the rate at which performance reaches an apparent asymptote is interesting. It takes around 20 to 80 targets (subject to the algorithm used) to reach the asymptotic values for all three performance matrices. Further work needs to be done in order to theorise the observation.

[4.2] Number of QSAR Dataset

Regardless of the type of algorithms, as the number of datasets is increased, improvement throughout evaluation metrics is observed. Typically, more data is preferable since it gives more domain coverage, albeit there may be a point of diminishing returns. As can be observed in *Table 3* where after 80 QSAR datasets the values across all performance matrix starts to improve rather gently. It should be noted for all the experiments done (with a total dataset of 100) in this report, TML has consistently underperformed relative to the base ML method. However, in general, more data improves the representation of combinations and variances of features in the feature space, as well as their mapping to class labels. This allows a model to better learn and generalise a class boundary in order to differentiate subsequent cases.

If the ratio of cases in the majority class to those in the minority class is relatively constant, we would anticipate the minority class to increase as the dataset size increases.

[4.2.1] Theoretical QSAR Dataset Requirement

*Table 4: The tabulated data on mean the percentage difference between the base ML and TML cases in accuracy, precision and recall for classification analysis. Only **Random Forest** algorithm was used for this analysis.*

Number of Target	Δ Mean Accuracy (%)	Δ Mean Precision (%)	Δ Mean Recall (%)
09	31.65006	26.684838	42.421091
19	23.158207	23.06114	24.14433
29	18.518955	18.25101	19.646108
39	16.414037	16.001568	18.46151
49	13.477179	12.773872	16.0159
59	11.099604	10.315111	13.647717
69	10.182506	9.618477	13.022596
79	9.770596	8.949876	13.094932
89	9.09144	8.009204	12.790212
99	9.285202	8.057161	13.351512

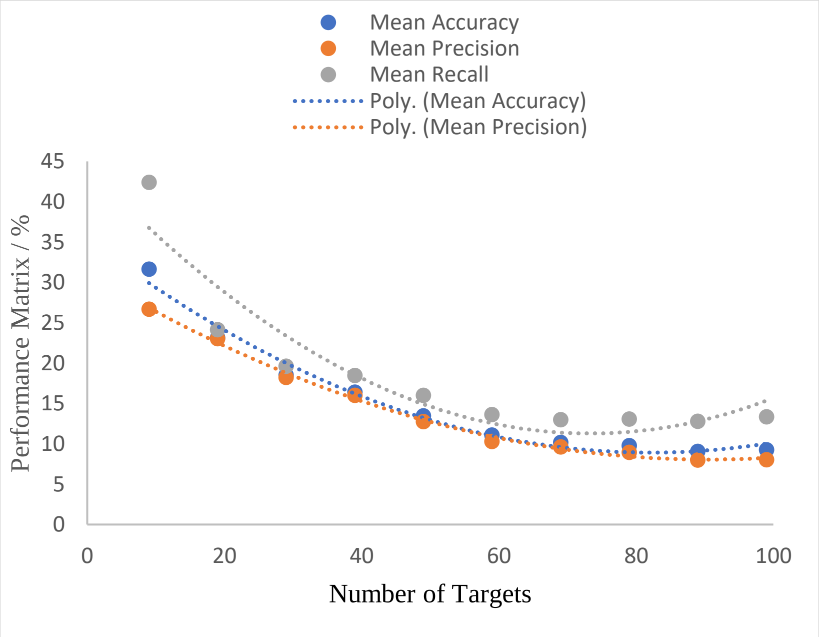


*Figure 3: Graphical representation of the mean delta performance metrics against number of targets for classification case. Only **Random Forest** algorithm was used for this analysis.*

*Table 5: Modelling equation, confidence of modelling (R2 value) and the breakeven target value between base ML and TML case for classification case. Only **Random Forest** algorithm was used for this analysis.*

Types of Performance	Modelling Equation	R2	Breakeven target number
Δ Mean Accuracy (%)	$y = 0.0039x^2 - 0.6453x + 35.411$	0.98	x is not a member of ℝ
Δ Mean Precision (%)	$y = 0.0028x^2 - 0.5102x + 31.211$	1.00	x is not a member of ℝ

Δ Mean Recall (%)	$y = 0.0062x^2 - 0.9033x + 44.39$	0.88	x is not a member of $\mathbb{R}$

In order to investigate the number of targets required to match the performance of TML and base ML, i.e. the breakeven target value, the difference of mean performance matrices were modelled using polynomial equation with a power of 2. The modelling were quite accurate as the R2 values were above 0.90. It was found that for no real numbers of target TML can outperform base ML method. This observation is in clear contradiction against the core paper on this topic by Olier et al. [1]. Olier et al. observed that, for ~2000 QSAR dataset TML outperforms the base ML method for regression analysis. It should be noted that *Theoretical QSAR Dataset Requirement* analysis was parallelly conducted for *regression* cases and the observations were similar to the *classification* cases.

From this observation, it can be theorised that, the rate of learning (by the *Algorithm #T*) is not linearly proportional to the number of QSAR datasets or TML performs. Further work needs to be done to justify the permanent underperformance of TML against base ML observed in this report and quantitatively analyse the relation between rate of learning and number of QSAR datasets.

It should be further noted that, only *Random Forest* algorithm was used for the *Theoretical QSAR Dataset Requirement* analysis. Ideally, other algorithms should have been tested laterally. However, from the observations in *Table 3* it is unlikely that other algorithms would yield any different observations.

[4.3] Different Types of Classifiers

Since the Dummy Classifier, does not aim to learn the data pattern. it is expected that the Base ML and TML would perform exactly the same. This observation validates the decision of using a Dummy Classifier as a control algorithm.

Although, the macro-level behaviours are similar across the algorithms (i.e., as datasets increase performance gets better), however, some algorithms perform fundamentally better than others. This claim is supported by two main observations:

1. The apparent asymptote of the Bagging Classier is higher than Random Forest Classifier.

1.1 Further research needs to be done to understand the theoretical reason behind it

1.2 More experiments are needed to answer the question: is there a fundamental gap between the performances of algorithms or is it a matter of the number of QSAR datasets?

2. The rate of learning is quite higher in some algorithms. For example, it takes the KNN classifier around 20 QSAR datasets to reach the apparent asymptote but Random Forest around 80 QSAR datasets.

It should be remarked that using only 100 datasets, under the XGB Boosts algorithm TML method's precision metric performed nearly as well as the Base ML method.

[5.0] Source Codes

https://github.com/Adnan1729

[6.0] Conclusion

Experimenting with 100 QSAR datasets, TML has consistently underperformed relative to the base ML method. Regardless of the type of algorithms, as the number of datasets is increased, improvement throughout evaluation metrics is observed. Moreover, when the datasets are balanced all matrices perform better.

It is also theorised that the rate of learning (by the Algorithm #T) is not linearly proportional to the number of QSAR datasets or that TML performs

Further research needs to be done in the following areas:

1. Why is there a repeated kink when the number of the QSAR dataset is 90?

2. Why do the performance matrices of Precision and Recall increase when undersampled, i.e., the size of individual datasets is reduced?

3. What is the relationship between the learning rate and the number of QSAR datasets?

4. Why are some algorithms fundamentally better than others?

Reference:

1. Olier, I., Orhobor, O., Dash, T., Davis, A., Soldatova, L., Vanschoren, J. and King, R., 2021. Transformational machine learning: Learning how to learn from many related scientific problems. Proceedings of the National Academy of Sciences, 118(49).

2. Chen, CH., Tanaka, K., Kotera, M. et al. Comparison and improvement of the predictability and interpretability with ensemble learning models in QSPR applications. J Cheminform 12, 19 (2020). https://doi.org/10.1186/s13321-020-0417-9

3. Miller, Tim. "Explanation in artificial intelligence: Insights from the social sciences." arXiv Preprint arXiv:1706.07269. (2017).

4. Kim, Been, Rajiv Khanna, and Oluwasanmi O. Koyejo. "Examples are not enough, learn to criticize! Criticism for interpretability." Advances in Neural Information Processing Systems (2016).

5. Fernández, A., García, S., Galar, M., Prati, R., Krawczyk, B. and Herrera, F., n.d. Learning from Imbalanced Data Sets.

6. Ferri, C., Hernández-Orallo, J. and Modroiu, R., 2009. An experimental comparison of performance measures for classification. Pattern Recognition Letters, 30(1), pp.27-38.

Appendix:

Table 6: Performance of evaluation metrics before balancing, trained under ***Random Forest*** *algorithm.*

	Balanced = *False*					
	Base			**TML**		
Target	**Acc**	**Pres**	**Rc**	**Acc**	**Pres**	**Rc**
10	0.932816	0.689920	0.531237	0.902489	0.000000	0.000000
20	0.926195	0.657723	0.486850	0.903399	0.000000	0.000000
30	0.927339	0.677046	0.489472	0.902047	0.085185	0.003260
40	0.926541	0.669258	0.497577	0.902427	0.111306	0.012788
50	0.924062	0.655141	0.473626	0.902484	0.220425	0.041340
60	0.924871	0.664623	0.483272	0.902388	0.252871	0.062541
70	0.926646	0.675678	0.495197	0.902643	0.294504	0.065859
80	0.926284	0.679735	0.486079	0.902733	0.315297	0.075180
90	0.926550	0.673744	0.491481	0.903325	0.372387	0.079798
100	0.925963	0.672928	0.486171	0.903532	0.347720	0.077680

Table 7: Performance of evaluation metrics after balancing, trained under ***Random Forest*** *algorithm.*

	Balanced = *Ture*					
	Base			**TML**		
Target	**Acc**	**Pres**	**Rc**	**Acc**	**Pres**	**Rc**
10	0.872831	0.874947	0.879123	0.680809	0.694976	0.671663
20	0.881682	0.874327	0.891925	0.805730	0.809430	0.802794
30	0.889280	0.879813	0.904167	0.832329	0.831958	0.837411
40	0.885268	0.876891	0.897995	0.844241	0.843118	0.848596
50	0.884224	0.880931	0.891345	0.851018	0.853346	0.851070
60	0.883673	0.880954	0.891705	0.861169	0.862878	0.863041
70	0.884194	0.885574	0.887375	0.863045	0.867570	0.860030
80	0.884339	0.882269	0.891005	0.871363	0.872256	0.874096

90	0.882937	0.881785	0.887799	0.872557	0.874302	0.872580
100	0.878436	0.878423	0.882148	0.863012	0.867260	0.862280

Table 8: Performance of evaluation metrics for Dummy Classifier algorithm.

Number of Targets	Accuracy		Precision		Recall	
	Base	**TML**	**Base**	**TML**	**Base**	**TML**
10	0.400000	0.400000	0.138601	0.138601	0.300000	0.300000
20	0.484573	0.484573	0.191293	0.191293	0.400000	0.400000
30	0.481686	0.481686	0.190958	0.190958	0.400000	0.400000
40	0.477762	0.477762	0.177381	0.177381	0.375000	0.375000
50	0.475545	0.475545	0.170123	0.170123	0.360000	0.360000
60	0.473077	0.473077	0.155597	0.155597	0.333333	0.333333
70	0.470949	0.470949	0.146558	0.146558	0.314286	0.314286
80	0.470971	0.470971	0.174997	0.174997	0.375000	0.375000
90	0.469063	0.469063	0.185526	0.185526	0.400000	0.400000
100	0.469620	0.469620	0.195394	0.195394	0.420000	0.420000

Table 9: Performance of evaluation metrics for Random Forest Classifier algorithm.

Number of Targets	Accuracy		Precision		Recall	
	Base	**TML**	**Base**	**TML**	**Base**	**TML**
10	0.829343	0.552454	0.830337	0.638715	0.827300	0.337475
20	0.816929	0.616248	0.812507	0.623313	0.819955	0.578456
30	0.824854	0.660916	0.817619	0.661793	0.832801	0.651225
40	0.823306	0.679316	0.817601	0.681289	0.830115	0.670520
50	0.825660	0.713334	0.819407	0.711706	0.830276	0.755496
60	0.827868	0.726854	0.822176	0.725097	0.831124	0.718308
70	0.825522	0.741269	0.819664	0.734764	0.822942	0.733499
80	0.825355	0.753348	0.820088	0.747576	0.819641	0.744949
90	0.825757	0.760038	0.821411	0.755611	0.820241	0.750748
100	0.821000	0.761000	0.816000	0.756000	0.817000	0.754000

Table 10: Performance of evaluation metrics for XGB Boost Classifier algorithm.

Number of Targets	Accuracy		Precision		Recall	
	Base	**TML**	**Base**	**TML**	**Base**	**TML**
10	0.852188	0.679958	0.829824	0.663732	0.882592	0.726888
20	0.869692	0.791823	0.856321	0.786615	0.889004	0.797717
30	0.876132	0.826772	0.863602	0.823759	0.896571	0.830539
40	0.872899	0.828313	0.863707	0.828673	0.89093	0.831768
50	0.871146	0.837074	0.863217	0.837291	0.888583	0.842141

60	0.871673	0.844906	0.866727	0.847086	0.886008	0.847498
70	0.870696	0.849439	0.86556	0.848739	0.88758	0.857213
80	0.868501	0.849439	0.861145	0.848739	0.888505	0.857213
90	0.869921	0.854907	0.861506	0.852458	0.889685	0.864227
100	0.863402	0.848066	0.855824	0.843665	0.881349	0.859435

Table 11: Performance of evaluation metrics for Bagging Classifier algorithm.

Number of Targets	Accuracy		Precision		Recall	
	Base	**TML**	**Base**	**TML**	**Base**	**TML**
10	0.844103	0.793067	0.839791	0.803331	0.848050	0.773012
20	0.866803	0.793067	0.872785	0.803331	0.857988	0.773012
30	0.868233	0.833196	0.868557	0.848331	0.868276	0.810004
40	0.866777	0.830490	0.878316	0.848719	0.854621	0.807112
50	0.867013	0.835341	0.874444	0.854208	0.860689	0.811557
60	0.867900	0.837417	0.879518	0.858922	0.858763	0.810535
70	0.861591	0.844057	0.874365	0.867256	0.852897	0.819522
80	0.859201	0.837409	0.869734	0.852207	0.849744	0.820941
90	0.859934	0.845579	0.867792	0.862944	0.854323	0.824564
100	0.862688	0.841988	0.868339	0.861833	0.857729	0.814760

Table 12: Performance of evaluation metrics for KNN Classifier algorithm.

Number of Targets	Accuracy		Precision		Recall	
	Base	**TML**	**Base**	**TML**	**Base**	**TML**
10	0.849941	0.671232	0.834142	0.663588	0.871580	0.685928
20	0.866132	0.741690	0.854322	0.737305	0.883603	0.750020
30	0.872012	0.779043	0.861601	0.779369	0.888415	0.781767
40	0.865501	0.781767	0.860946	0.783960	0.868447	0.790710
50	0.863501	0.788767	0.856946	0.787960	0.878447	0.800710
60	0.842676	0.791860	0.846575	0.791868	0.869036	0.808728
70	0.854279	0.798860	0.848589	0.797868	0.873709	0.813728
80	0.849279	0.801417	0.851890	0.799868	0.869371	0.819728
90	0.852676	0.811417	0.845575	0.802971	0.872036	0.831765
100	0.847638	0.810364	0.841217	0.801994	0.866242	0.829673