A Report On

Machine Learning Based Classification of Non-functionalized MXenes into Metals and Non-metals

By

MURARKA UTSAV ANIL
2017B5A70854G

For completion of
DESIGN PROJECT (PHY F266)
IN THE SECOND SEMESTER 2019-2020

Under Supervision of
Dr. Swastibrata Bhattacharyya
Dr. Kinjal Banerjee

AT THE DEPARTMENT OF PHYSICS, BITS PILANI K.K. BIRLA GOA CAMPUS

ACKNOWLEDGEMENTS

The project has provided me a great opportunity to explore the field of machine learning by applying it to a real-world problem. I am highly obliged towards my mentors Dr. Swastibrata Bhattacharyya and Dr. Kinjal Banerjee for providing me this project and helping me in every part of this project with resources, suggestions and knowledgeable insight in the topic. I would like to extend my sincere gratitude to them for the constant support and supervision.

TABLE OF CONTENTS

ABSTRACT

MXenes are a class of materials whose physical properties are studies extensively in material science. One of the important physical properties of MXenes is its metallic or non-metallic nature. Some MXenes have a band gap of 0 eV, which means they are metals and some have a positive band gap which means they are non-metals. Using machine learning (more specifically, supervised learning) it is possible to train models on some features of MXene molecules which can classify new molecules into metals and non-metals. The choice of these features plays a crucial role in determining how accurately the model gets trained. This project explores various features that we can use to classify MXenes accurately and various machine learning algorithms that will train the classification models on existing data. In the end, the performance of all algorithms is compared by using suitable evaluation metrics.

Link to codebase: https://github.com/UtsavMurarka/MXene-machine-learning

CLASSIFICATION PROBLEMS: A MACHINE LEARNING APPROACH

Machine Learning can broadly solve two types of problems, regression and classification. This project deals with an instance of a classification problem.

In a classification problem, we have many data points described by a finite set of features, and each data point belongs to a group called “a class”. There can be two or more classes in a classification problem. A problem where the points can be in one of 2 classes is called a “binary classification problem” and where the points can be in more than 2 classes is called a “multi-class classification problem”. This project deals with a binary classification problem.

To solve a binary classification problem, we use supervised learning algorithms which are trained on labelled data and can be used to predict classes of unlabeled/new data. The major algorithms used for classification problems are:

1. Naïve Bayes Classifier
2. Logistic Regression
3. K-Nearest Neighbors
4. Decision Trees
5. Random Forests
6. Support Vector Machines
7. Neural Networks

1. Naïve Bayes Classifier

Naïve Bayes Classifier employs Bayes theorem to calculate the probability of a data point being in a particular class. The class that has the highest probability for a particular data point is assigned to that data point. The probability distribution of individual features is modelled by the training data provided to the model. The probability is modelled as a Gaussian distribution.

2. **Logistic Regression**
   Logistic Regression works by fitting a hyperplane (this “hyperplane” is a line for 2D data, a plane for 3D data, and a hyperplane for higher dimensional data) through the data. This hyperplane acts as a separating boundary between the classes. The parameters that determine the orientation and position of this hyperplane are computed by minimizing a cost function based on the training data.

3. **K-Nearest Neighbors**
   K-Nearest Neighbors is a very simple but effective algorithm for classification problems. It works by calculating the distance of every new point with all the training data points, and determining in which classes do its “nearest neighbors” belong. The class to which maximum out of its K nearest points belong is assigned to the new data point, where K is a tunable hyperparameter.

4. **Decision Trees**
   Decision Trees work by dividing the feature space into smaller regions based on some threshold values of features computed by minimizing the information entropy of the training data. Each of the smaller region associated with a class. For a new data point, we calculate that in which of the smaller regions does it lie and then assign it to the corresponding class.

5. **Random Forest**
   Random Forests are a collection of decision trees, they are used to average out the errors made by individual decision tree.

6. **Support Vector Machines**
   Support Vector Machines are very popular for creating non-linear separation boundaries between classes. SVM uses the kernel trick to map the feature space to a non-linear space and then create a separation boundary. The most popular kernel (which has also been used in this project) is the radial basis function (rbf) kernel.

7. **Neural Networks**
   Neural Networks are a group of many layers made up of several neurons. Neurons in adjacent layers are connected to each other. Every neuron is associated with a mathematical function (called activation function) whose output determines the value of that neuron, and it takes as input the values of all the neurons it is connected to in the previous layer. The final (output) layer consists of neurons associated with the classes, and the i^{th} neurons value tells us the probability of that particular input being in the i^{th} class.
   The values of every neuron in the neural network is determined by minimizing a cost function on the training data.

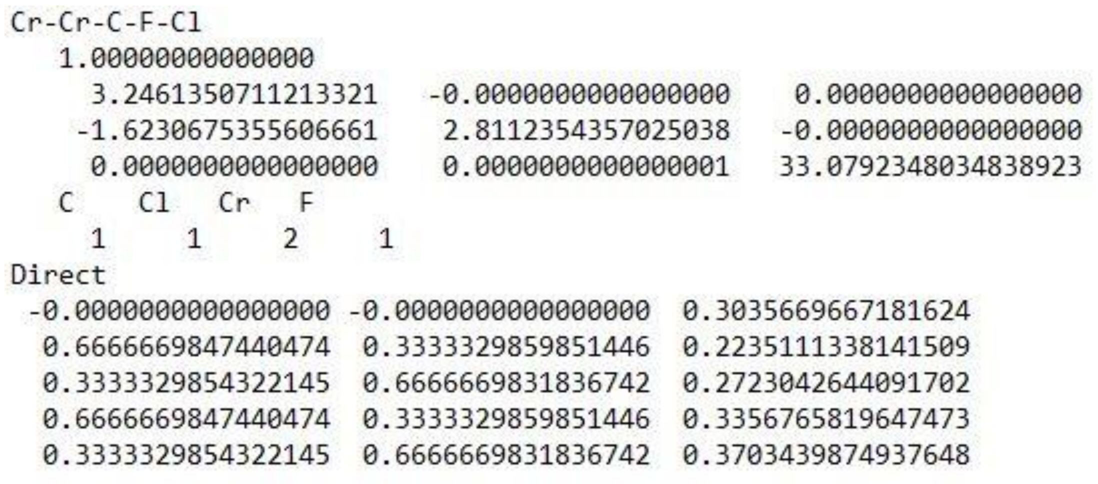
DATASET

1. **Data Acquisition**
   The data required for this project was acquired from the aNANt database of IISc, Bangalore. To acquire that data from the website, a web scraper was created. The code for this webscraper can be found here: (https://github.com/UtsavMurarka/MXene-machine-learning/blob/master/anant_data_miner/mine.py).

2. **Data Cleansing**
   The data on the aNANt website is available in a downloadable POSCAR file. However, for this project, only the numerical values were required which were extracted by parsing the HTML of the aNANt webpage.
   The cleansing also included removal of functionalized MXenes because this project focusses only on non-functionalized MXenes.

UNDERSTANDING THE DATA

The first step to solve any problem using machine learning is to understand the data we have. To understand the data that we collected from aNANt database, it is important to understand the contents of a general POSCAR file. Below is a snapshot of a POSCAR file of Cr-Cr-C-F-Cl.



```
Cr-Cr-C-F-Cl
   1.00000000000000
     3.2461350711213321   -0.0000000000000000    0.0000000000000000
    -1.6230675355606661    2.8112354357025038   -0.0000000000000000
     0.0000000000000000    0.0000000000000001   33.0792348034838923
   C    Cl   Cr   F
     1     1     2     1
Direct
 -0.0000000000000000 -0.0000000000000000  0.3035669667181624
  0.6666669847440474  0.3333329859851446  0.2235111338141509
  0.3333329854322145  0.6666669831836742  0.2723042644091702
  0.6666669847440474  0.3333329859851446  0.3356765819647473
  0.3333329854322145  0.6666669831836742  0.3703439874937648
```

The first line is a comment line.
The second line has the scaling parameter.
The next 3 lines give us the lattice vectors in cartesian coordinates.
The 5 lines after “Direct” give us the coordinates of each atom in the lattice.

Therefore, by having these 24 numbers, we can construct the lattice of that particular MXene.

The next step is to map all this information to features that will be ultimately used for training ML models.

FEATURE ENGINEERING

(data in csv format can be found here(https://github.com/UtsavMurarka/MXene-machine-learning/tree/master/anant_data_miner/post_midsem_work))

We map all the 24 numbers discussed above to 24 features (F1 to F24). This has given us complete information about the structure of the molecule.
However, these 24 numbers do not contain the information about which atoms constitute the molecule. So, to include that information, 5 additional features were constructed (F25 to F29). These are the atomic numbers of the 5 molecules in that atom. Atomic number was chosen since it uniquely identifies an atom.

List of Features and their meaning:

F1	x-component of first lattice vector
F2	y-component of first lattice vector
F3	z-component of first lattice vector
F4	x-component of second lattice vector
F5	y-component of second lattice vector
F6	z-component of second lattice vector
F7	x-component of third lattice vector
F8	y-component of third lattice vector
F9	z-component of third lattice vector
F10	x-coordinate of atom 1
F11	y-coordinate of atom 1
F12	z-coordinate of atom 1
F13	x-coordinate of atom 2
F14	y-coordinate of atom 2
F15	z-coordinate of atom 2
F16	x-coordinate of atom 3
F17	y-coordinate of atom 3
F18	z-coordinate of atom 3
F19	x-coordinate of atom 4
F20	y-coordinate of atom 4
F21	z-coordinate of atom 4
F22	x-coordinate of atom 5
F23	y-coordinate of atom 5
F24	z-coordinate of atom 5
F25	Atomic number of atom 1
F26	Atomic number of atom 2
F27	Atomic number of atom 3
F28	Atomic number of atom 4
F29	Atomic number of atom 5

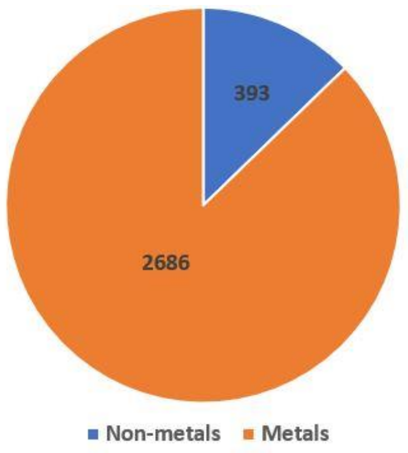
Dimensionality Reduction

While working with high dimensional data, it is a common occurrence that the data points become sparse in the feature space which leads to improper training of statistical and distance-based models. Therefore, methods like Principal Component Analysis and Neighborhood Component Analysis are used to decrease the dimensionality of the data.

Feature Scaling

If we are working with data where every feature is of a different order of magnitude, it becomes difficult for models to run optimizations like gradient descent due to the highly skewed feature space. Therefore, it is a good practice to scale all the features either between -1 to 1 (Standard Scaler) or 0 to 1 (MinMaxScaler).
Since, our features are mostly of the order 1 to 10, we don't need to do feature scaling now, however, afterward we will add additional features which won't be of the same order, there we will do feature scaling.

PROBLEM OF CLASS IMBALANCE



The data that we have, consists of a total of 3079 instances of MXenes, and 2686 of them are Metals and only remaining 393 are metals. Due to this imbalance, training a classification model becomes difficult because if the number of training points in a particular class are dominant, the model will generally favor that class while making predictions. So, to eliminate this problem, there are many things that can be done at the level of data preprocessing and also at the level of model training. At the level of data preprocessing, we can handle class imbalance by oversampling and undersampling, and at the level of model training we can eliminate it by using cost-sensitive training which uses a skewed cost function to counter the effect of skewed class sizes in training data.

Oversampling:
In oversampling, we choose all the data from our majority class (metals), and oversample the data of minority class (non-metals). This oversampling can be done in multiple ways, the two methods used in this project are random oversampling and synthetic minority oversampling technique (SMOTE).
Undersampling:
In undersampling, we choose all the data from our minority class, and undersample the data of majority class. This undersampling is done by randomly choosing a small number of data points from majority class.
Cost-Sensitive Training:
In Cost sensitive training, a skewed cost-function is used, which penalized an incorrect classification of a minority sample more than a majority sample. This makes the training "fair" and counteracts the class imbalance.

MODEL EVALUATION METRICS

After training a model, it is used to make prediction on a test set, and compare the output of the model to actual answers to see how well the model performs. However, it is important that we quantify the performance of the model so that we can make a comparison between various models. Therefore, we use different evaluation metrics. Every evaluation metric has its own importance, and based on the kind of problem we are dealing with, we must choose the metric most suitable for our use case.

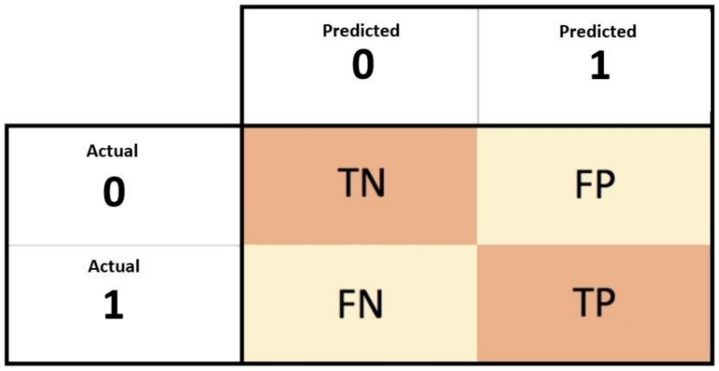
1. **Accuracy**
   It is the simplest and the most intuitive evaluation metric for a classification problem.

$$accuracy = \frac{No.\ of\ points\ correctly\ classified}{Total\ no.\ of\ points\ in\ test\ set}$$

However, accuracy is not a very good metric when it comes to problems with a class imbalance. For instance, in our problem, 2686 out of 3079 data points belong to the class “metals”, so, even if a model classifies every test point into metals, it will still have an accuracy close to 80%, which in no way reflects the performance of the model, because such a model would be of no practical use.

2. **Confusion Matrix**

Confusion matrix is the most commonly used evaluation metric (a set of evaluation metrics, to be precise) used for classification problems. A confusion matrix gives a clear picture of the number of false negatives, false positives, true negatives and true positives.



	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

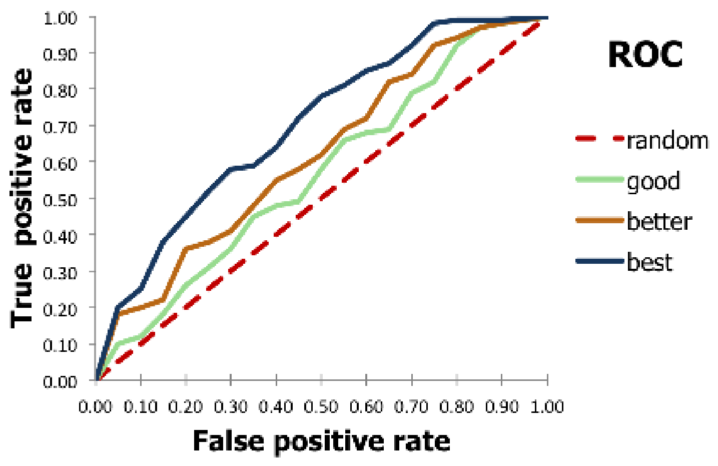
It is also possible to derive a few more metrics from the confusion matrix. They are the following

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

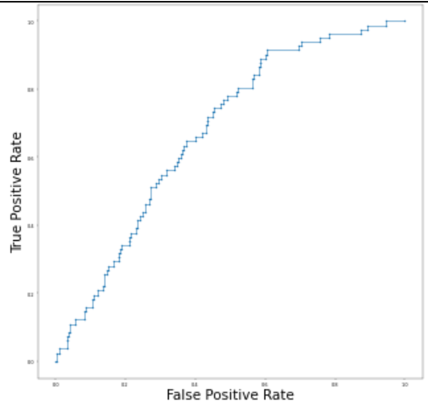
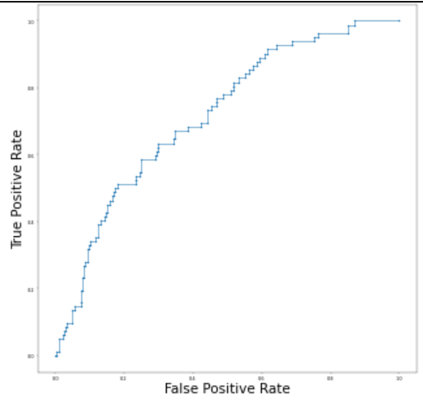
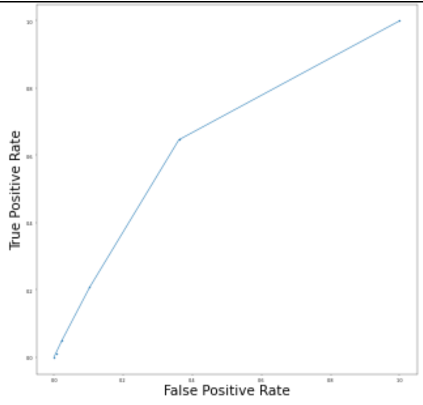
3. **Receiver Operating Characteristic – Area Under Curve (ROC-AUC)**
   This metric is very popular when it comes to imbalanced classification. The ROC curve is a curve plotted on the axes true positive rate (TPR) and false positive rate (FPR). However, it must not be interpreted as TPR being a function of FPR.
   Every machine learning model outputs a probability of a point being in a particular class, and then the class for which this probability is more than 50% is assigned to that particular point. This "50%" is called as the threshold. However, one may choose a different threshold for assigning the class to the point. The ROC curve is plotted by varying this threshold.
   This threshold is varied from 0% to 100%, and for every threshold we compute the TPR and FPR, and then mark a point (TPR,FPR) on the graph. We mark several such points each corresponding to a different threshold and then join then. The curve thus obtained is the ROC curve. The area under that curve is the ROC-AUC value.
   This area can be between 0 and 1. An AUC of 0.5 means that the classifier is predicting random outputs. Ideally, the AUC should be 1. But practically, it should be as close to 1 as possible.

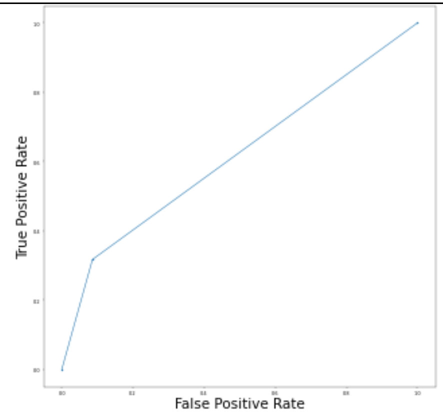
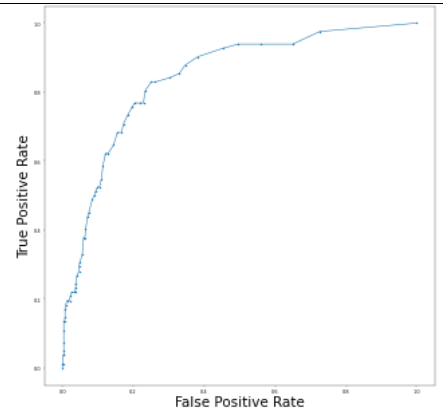
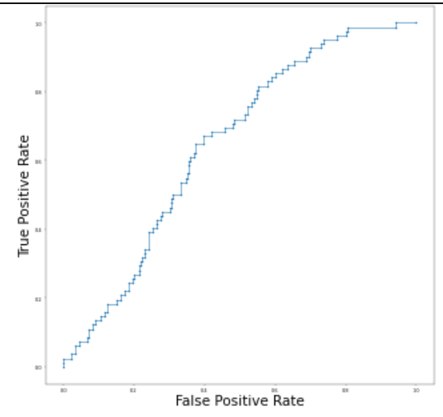


COMPARING MODELS

(code: https://github.com/UtsavMurarka/MXene-machine-learning/blob/master/MXene_ML.ipynb)

In this section, we will compare the performance of various machine learning models based on accuracy, confusion matrix and ROC-AUC.

Model	Accuracy	Confusion Matrix	ROC-AUC
Naïve Bayes	75.9%	$\begin{array}{\|c\|c\|}\hline 445 & 89 \\ \hline 59 & 23 \\ \hline\end{array}$	 AUC: 0.674
Logistic Regression	86.2%	$\begin{array}{\|c\|c\|}\hline 530 & 4 \\ \hline 81 & 1 \\ \hline\end{array}$	 AUC: 0.719
K-Nearest Neighbors	85.3%	$\begin{array}{\|c\|c\|}\hline 522 & 12 \\ \hline 78 & 4 \\ \hline\end{array}$	 AUC: 0.647

Decision Tree	83.9%	$\begin{array}{\|c\|c\|}\hline 487 & 47 \\ \hline 52 & 30 \\ \hline\end{array}$	 AUC: 0.639
Random Forest	87.9%	$\begin{array}{\|c\|c\|}\hline 524 & 10 \\ \hline 64 & 18 \\ \hline\end{array}$	 AUC: 0.822
SVM	86.6%	$\begin{array}{\|c\|c\|}\hline 534 & 0 \\ \hline 82 & 0 \\ \hline\end{array}$	 AUC: 0.649

Analysis: Most of the models performed poorly (ROC-AUC < 0.8). Random Forest's performance was best according to accuracy as well as ROC-AUC.

One common feature in the confusion matrix of all models is the higher number of False negatives.

NEED FOR MORE INFORMATION

Since most models are performing poorly on the test data, it is safe to assume that the features we have for each data point are not sufficient to accurately determine the class of the MXene. From some knowledge of the physics of the problem, we can argue that the physical properties of the molecule are not well represented in the data, because the data contains majority information of only the structure of the molecule. Hence, we should also include more features about the physical properties of the individual atoms that make up the molecule.

REDEFINING FEATURES

For each of the 5 atoms in the molecule we add the following properties to the molecule's data:

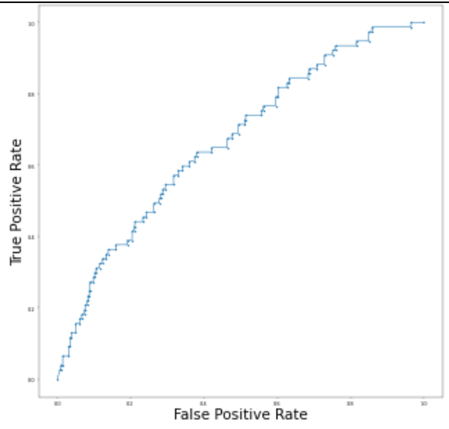
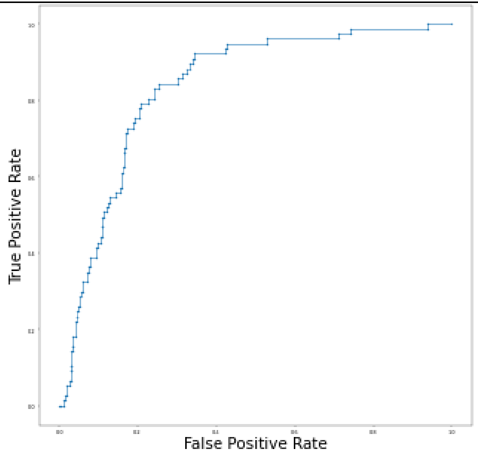
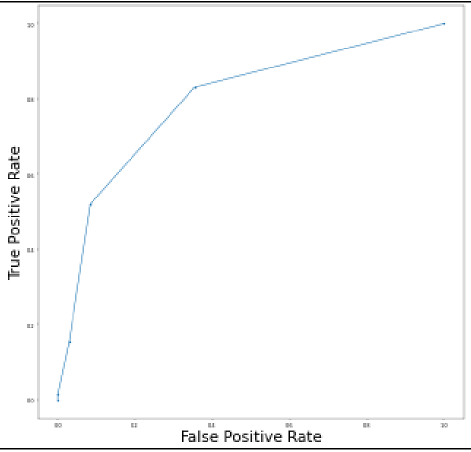
1. Neutron number
2. Proton number
3. Number of electrons
4. Period
5. Group number
6. Atomic radius
7. Electronegativity
8. First Ionization Energy
9. Density
10. Melting point
11. Boiling point
12. Number of isotopes
13. Number of shells
14. Specific heat
15. Mass Magnetic Susceptibility
16. Molar Magnetic Susceptibility
17. Thermal Conductivity

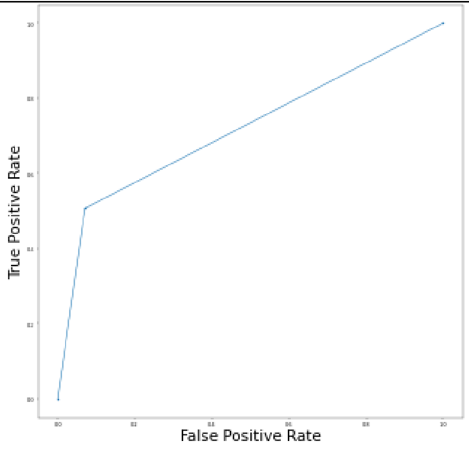
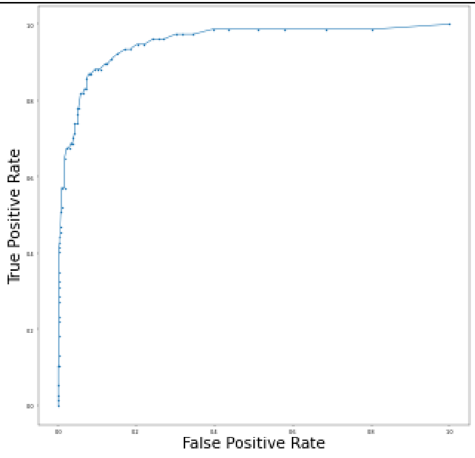
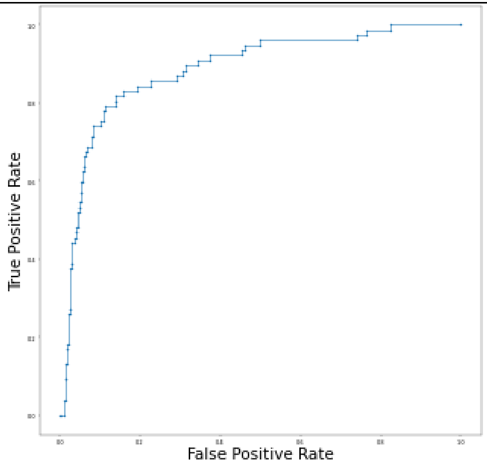
So, now, in addition to the 29 features we had earlier, we now have 85 new features (17 properties x 5 atoms per molecule). We also add their variances within the molecule as a separate feature. Therefore, a total of 131 features. With the addition of these features, feature scaling also becomes important, because every feature will be of a different order of magnitude.

IMPROVEMENT IN RESULTS

(code: https://github.com/UtsavMurarka/MXene-machine-learning/blob/master/MXene_physicalProperties%20(1).ipynb)

As expected, the results improve after considering additional features.

Model	Accuracy	Confusion Matrix	ROC-AUC
Naïve Bayes	66.5%	366 173 33 44	 AUC: 0.667
Logistic Regression	86.7%	517 22 60 17	 AUC: 0.837
K-Nearest Neighbors	86.8%	523 16 65 12	

			AUC: 0.795
Decision Tree	87.7%	501 | 38 38 | 39	 AUC: 0.718
Random Forest	93.2%	530 | 9 33 | 44	 AUC: 0.952
SVM	86.8%	533 | 6 75 | 2	 AUC: 0.887

Analysis: The results of all models (except Naïve Bayes) have improved. However, the best model is still random forest. Its accuracy has improved to 93.3% and ROC-AUC value has improved to 0.931.

Observation: Even after doing dimensionality reduction on the data using PCA and NCA, the results were not affected much.

HANDLING CLASS IMBALANCE

(code for oversampling & undersampling with random forest: https://github.com/UtsavMurarka/MXene-machine-learning/blob/master/Oversampling_undersmpling.ipynb)

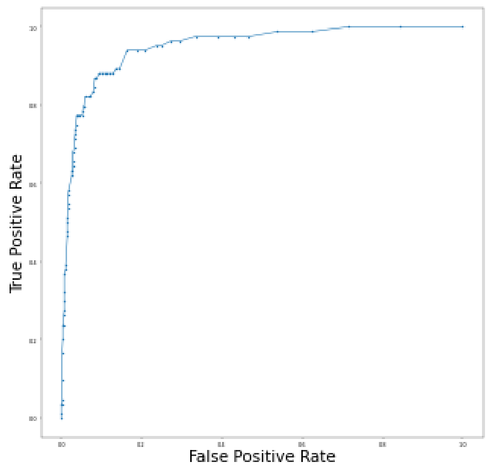
(code for cost sensitive neural networks: https://github.com/UtsavMurarka/MXene-machine-learning/blob/master/cost_sensitive_NN.ipynb)

As described in "Problem of Class Imbalance" section, 3 methods have been used to tackle the problem. Oversampling, undersampling and cost-sensitive training.

Oversampling with Random Forest:

Accuracy: 93.2%

ROC Curve:



ROC-AUC: 0.949

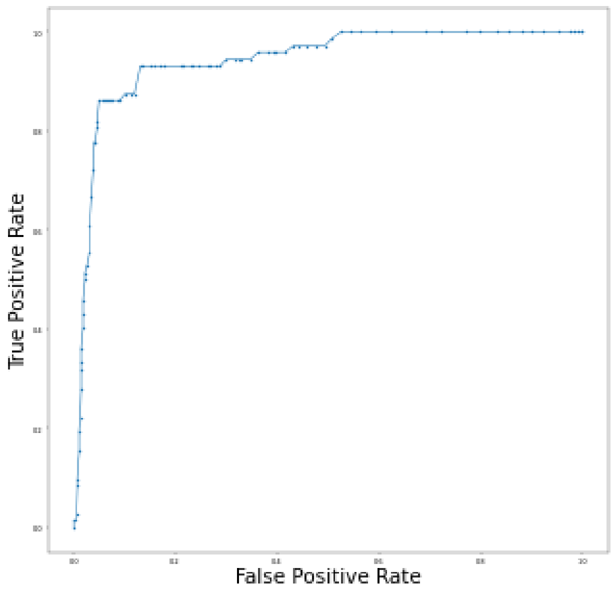
Confusion Matrix:

517	15
57	27

Undersampling with Random Forest:

Accuracy: 85.06%

ROC Curve:



ROC-AUC: 0.943

Confusion Matrix:

458	87
5	67

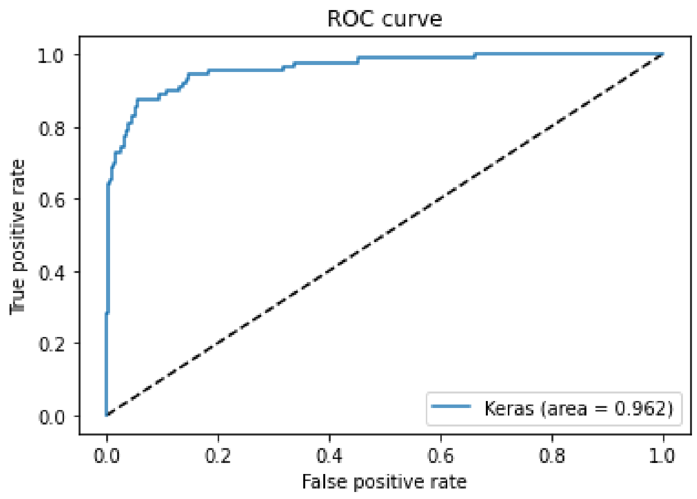
Cost-Sensitive Neural Network:

Neural Network model summary:

```
Layer (type)                 Output Shape              Param #
=================================================================
dropout_24 (Dropout)          (None, 130)               0
_________________________________________________________________
dense_38 (Dense)              (None, 130)               17030
_________________________________________________________________
dropout_25 (Dropout)          (None, 130)               0
_________________________________________________________________
dense_39 (Dense)              (None, 250)               32750
_________________________________________________________________
dense_40 (Dense)              (None, 1)                 251
=================================================================
Total params: 50,031
Trainable params: 50,031
Non-trainable params: 0
_________________________________________________________________
```

Accuracy: 93.67%

ROC Curve:



ROC-AUC: 0.962

Confusion Matrix:

526	1
38	51

CONCLUSIONS

The best performance in terms of ROC-AUC value was obtained by Cost-Sensitive Neural Network. It gave a ROC-AUC of 0.962 and an Accuracy of 93.67%. The results of Random Forest were also comparable, it gave a ROC-AUC of 0.952 and an Accuracy of 93.2%. It was also successfully demonstrated that including more information about the physical properties of the atoms constituting the MXene molecule improves the results. An attempt was also made to train models by removing the lattice features and using only the atom properties as features, but it did not yield any improvement in results.

However, by including more features and using more complex and advanced machine learning techniques, it might be possible to train models which classify MXenes even more accurately. Machine learning in physics is relatively new, and it still has a long way to go!

All the codes written for doing various machine learning, data analysis and acquisition tasks are available on GitHub (https://github.com/UtsavMurarka/MXene-machine-learning).

REFERENCES

1. A. Burkov. The Hundred-Page Machine Learning Book. Andriy Burkov,2019.

2. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel,M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Pas-sos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research,12:2825–2830, 2011.

3. Machine-Learning-Assisted Accurate Band Gap Predictions of Functionalized MXene. Arunkumar Chitteth Rajan, Avanish Mishra, Swanti Satsangi, Rishabh Vaish, Hiroshi Mizuseki, Kwang-Ryeol Lee, and Abhishek K. Singh. Chemistry of Materials 2018 30 (12), 4031-4038 DOI: 10.1021/acs.chemmater.8b00686

4. Gabriel R Schleder et al 2019 J. Phys. Mater. 2 032001

5. aNANt: A functional materials database (http://anant.mrc.iisc.ac.in)