Explainable Machine Learning Algorithms For Predicting Glass Transition Temperatures

SOUSTAB HALDAR 21074029
RITURAJ BARAI 21074024
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSITITUTE OF TECHNOLOGY (BHU) VARANASI

Overview

- Introduction
- Problem Statement
- Dataset Descrption
- Work done in the paper

(i) Different Featurization methods used in the paper

(ii) Machine Learning and deep learning algorithms used

(iii) Other things that you would like to mention

- Your progress so far

(i) Different Featurization methods built by you

(ii) Machine Learning and deep learning algorithms used by you

(iii) Other things that you would like to mention

- Conclusion

Introduction

- Machine Learning (ML) algorithms can be used to predict the glass transition temperature (Tg) of glasses based on their chemical composition.
- A dataset of 43,240 oxide glass compositions and found that the Random Forest algorithm provided the best predictive performance for Tg.
- The article also highlights the challenge of predicting extreme Tg values and suggests that the k-Nearest Neighbours algorithm performs well in these cases. The authors propose a new visual approach to explain what the RF model learned about the importance of each chemical element in obtaining glasses with extreme Tg.
- The study suggests ML algorithms can replace empirical approaches in developing novel glasses with relevant properties and applications.
- Future Work done by Meta Models can further improve the work.

Dataset

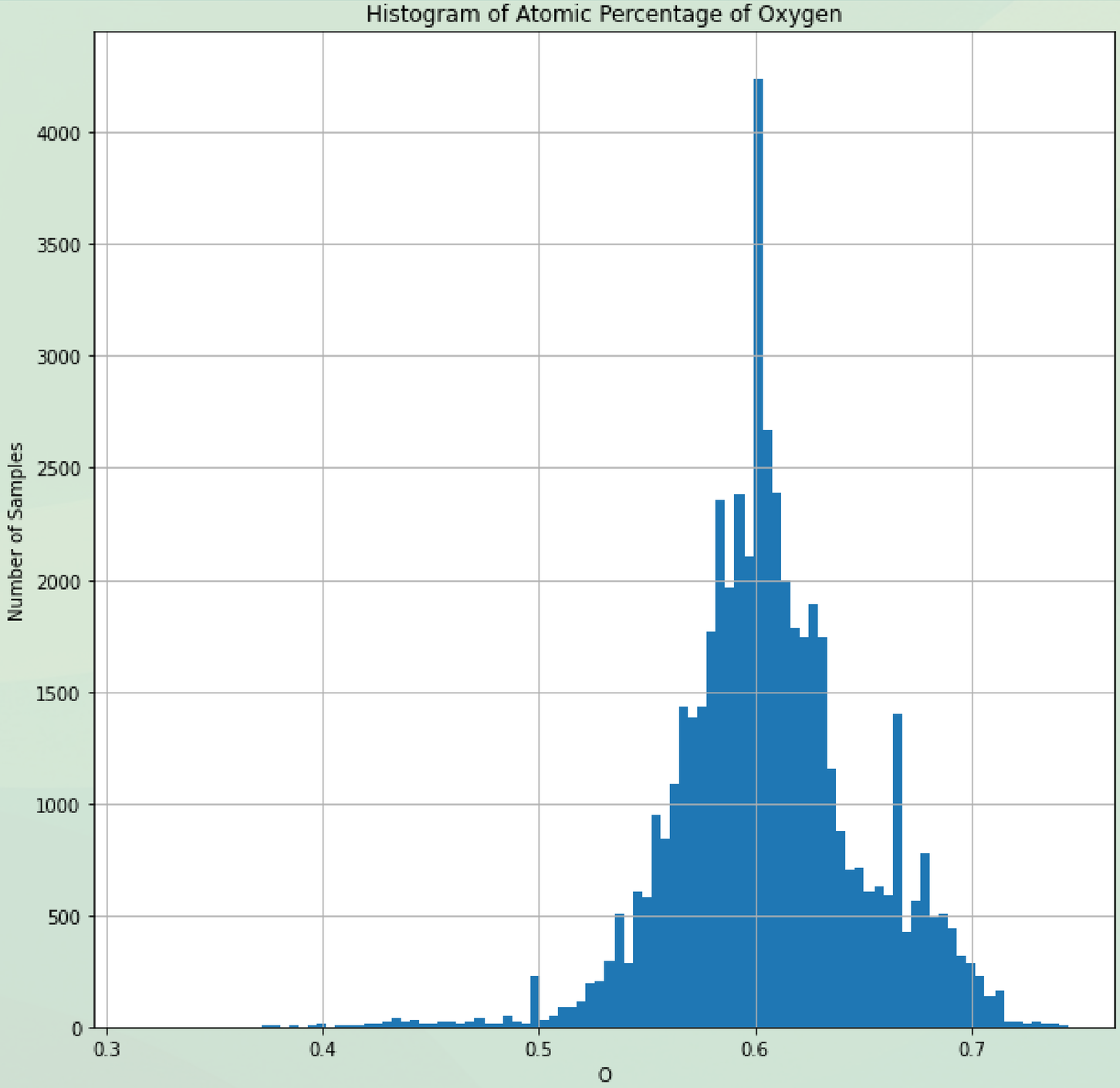
- The Tg dataset used in this work was collected from scientific journals, books, and patents obtained from the SciGlass database V12.0 (downloaded from the dataset included with the paper)
- Oxide glasses were considered i.e glasses with over 30% of the atomic fraction of oxygen.
- The collected raw dataset has approximately 51000 glass compositions, each composition with 2 to 32 different chemical elements from a total of 65 chemical elements.
- The final dataset has Tg values in range from 342K to 1495K and they follow a normal distribution with mean approx. 776K.

ID	Li	Be	B	O	Na	Mg	Al	Si	P	K	Ca
478474_40951	0	0	0.099699	0.628564	0	0.008412	0.052889	0.180302	0	0	0.026445
308661_26527	0	0	0	0.578947	0	0	0	0	0	0	0
448275_37908	0	0	0	0.52381	0	0	0	0	0	0	0
371146_28035	0	0	0	0.655172	0.103448	0	0	0	0.241379	0	0
354054_26065	0	0	0	0.6	0	0	0	0	0	0	0
448273_37908	0	0	0	0.52381	0	0	0	0	0	0	0
354055_26065	0	0	0	0.6	0	0	0	0	0	0	0
448276_37908	0	0	0	0.538462	0	0	0	0	0	0	0
448272_37908	0	0	0	0.540231	0	0	0	0	0	0	0
308660_26527	0	0	0	0.585366	0	0	0	0	0	0	0
308662_26527	0	0	0	0.571429	0	0	0	0	0	0	0
448270_37908	0	0	0	0.54023	0	0	0	0	0	0	0
449027_38029	0	0	0	0.6	0	0	0	0	0.2	0	0
450192_28531	0	0	0	0.640687	0	0	0.007926	0.017173	0.161162	0	0

Data Analysis

- The dataset comprises of glass compositions each having their individual unique ID followed by 65 elements and their respective atomic compositions. The transition temperatures (Tg) follow each of the individual glass compositions.

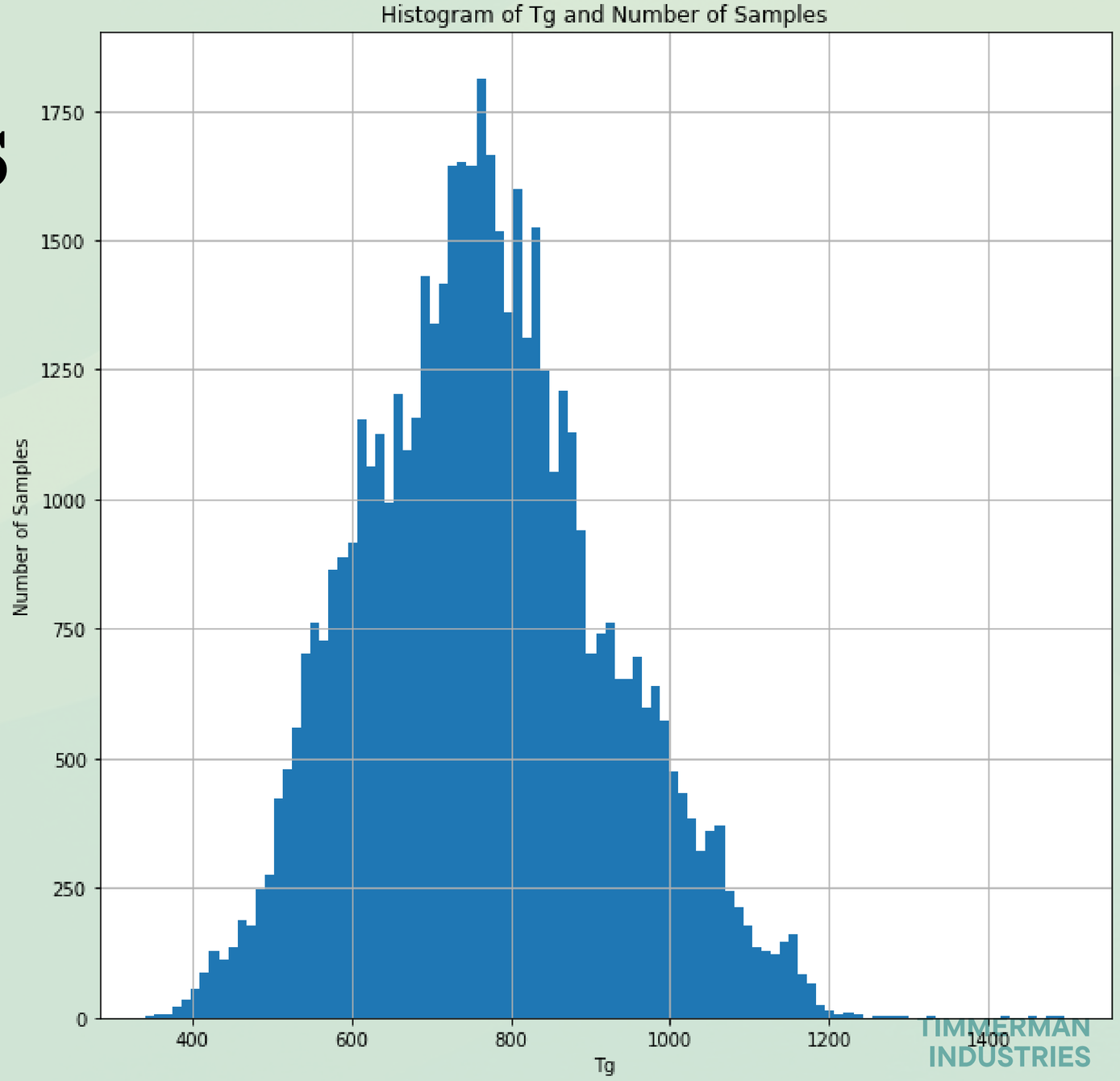
ID	Li	Be	B	O	Na	Mg	Al	Si	P	K	Ca
478474_40951	0	0	0.099699	0.628564	0	0.008412	0.052889	0.180302	0	0	0.026445
308661_26527	0	0	0	0.578947	0	0	0	0	0	0	0
448275_37908	0	0	0	0.52381	0	0	0	0	0	0	0
371146_28035	0	0	0	0.655172	0.103448	0	0	0	0.241379	0	0
354054_26065	0	0	0	0.6	0	0	0	0	0	0	0
448273_37908	0	0	0	0.52381	0	0	0	0	0	0	0
354055_26065	0	0	0	0.6	0	0	0	0	0	0	0
448276_37908	0	0	0	0.538462	0	0	0	0	0	0	0
448272_37908	0	0	0	0.540231	0	0	0	0	0	0	0
308660_26527	0	0	0	0.585366	0	0	0	0	0	0	0
308662_26527	0	0	0	0.571429	0	0	0	0	0	0	0
448270_37908	0	0	0	0.54023	0	0	0	0	0	0	0
449027_38029	0	0	0	0.6	0	0	0	0	0.2	0	0
450192_28531	0	0	0	0.640687	0	0	0.007926	0.017173	0.161162	0	0


Histogram of Atomic Percentage of Oxygen
Number of Samples
0
500
1000
1500
2000
2500
3000
3500
4000
0.3
0.4
0.5
0.6
0.7
O

Dataset Analysis

Analyzing the given dataset gives the following parameters

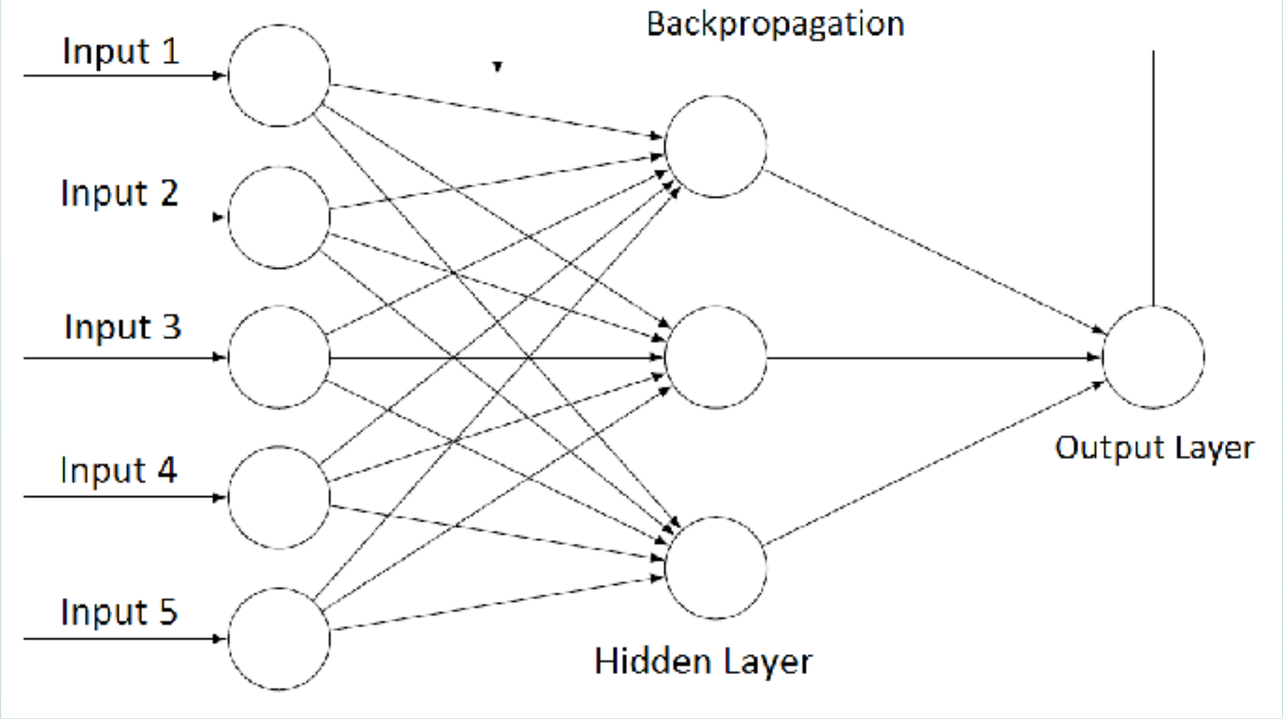
- Mean 767.6484784313724
- Median 762.15



Possible algorithms to predict Tg

Multi-layer Perception

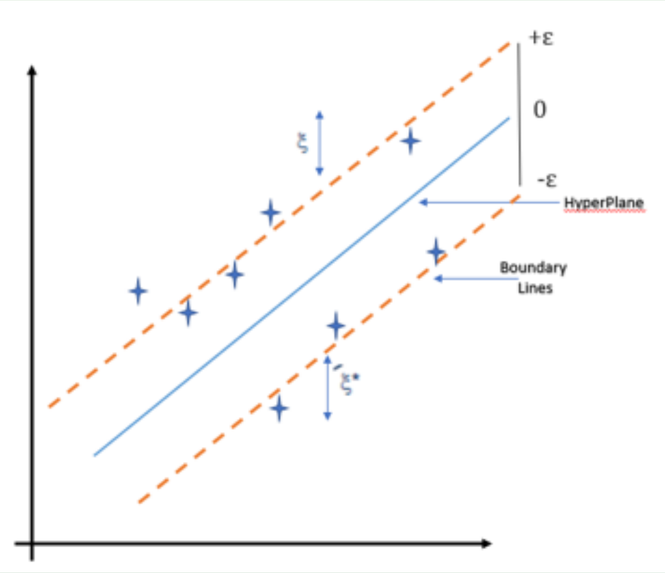
Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.

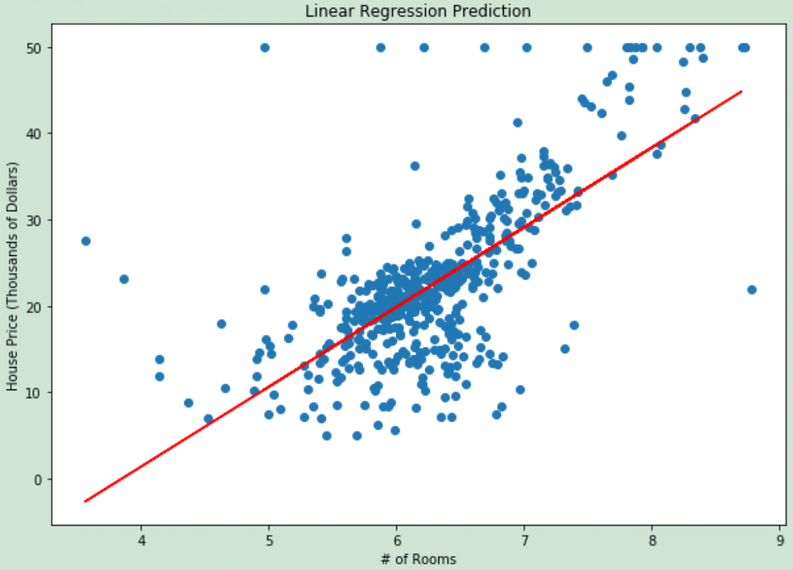


Support Vector Regression

Support Vector Regression as the name suggests is a regression algorithm that supports both linear and non-linear regressions. This method works on the principle of the Support Vector Machine. SVR differs from SVM in the way that SVM is a classifier that is used for predicting discrete categorical labels while SVR is a regressor that is used for predicting continuous ordered variables.

SVR works on the principle of SVM with few minor differences. Given data points, it tries to find the curve. But since it is a regression algorithm instead of using the curve as a decision boundary it uses the curve to find the match between the vector and position of the curve. Support Vectors helps in determining the closest match between the data points and the function which is used to represent them.

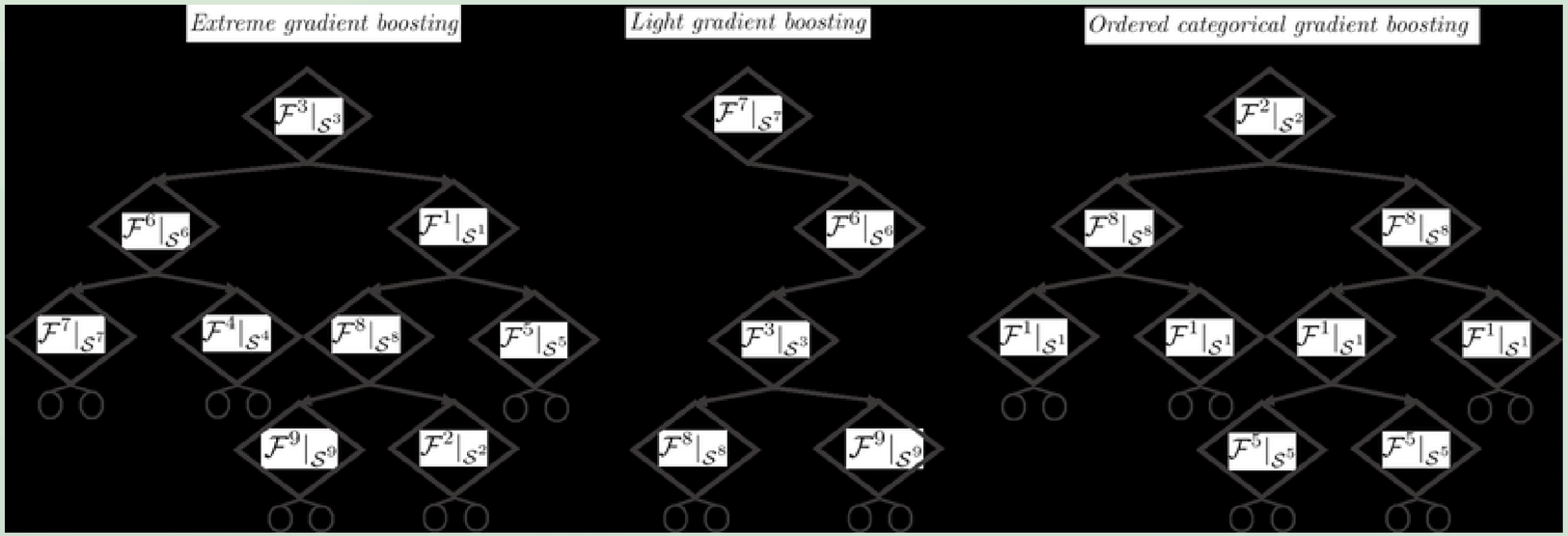




Categorical boosting or CatBoost

Gradient Boosting is an ensemble machine learning algorithm and typically used for solving classification and regression problems. It is easy to use and works well with heterogeneous data and even relatively small data. It essentially creates a strong learner from an ensemble of many weak learners.

CatBoost or Categorical Boosting is an open-source boosting library developed by Yandex. In addition to regression and classification, CatBoost can be used in ranking, recommendation systems, forecasting and even personal assistants.

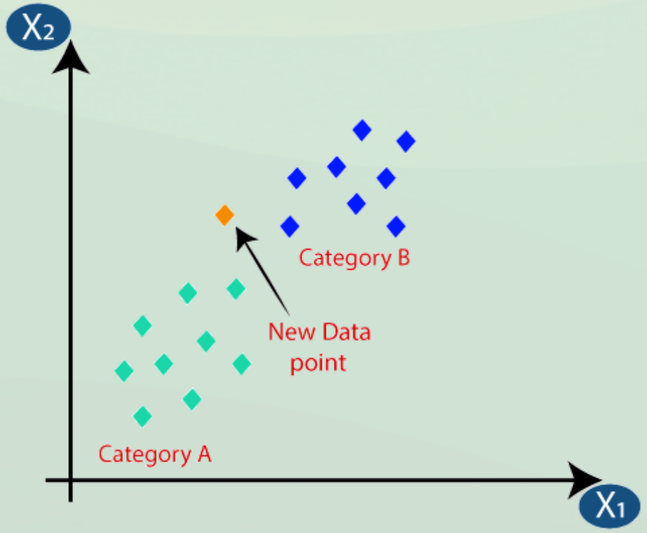


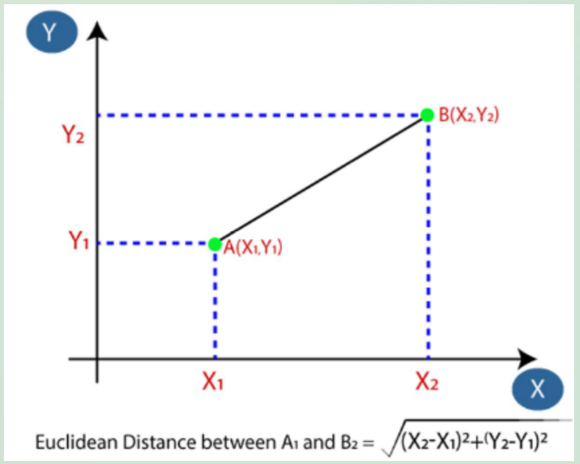
K-Nearest Neighbor(KNN) Algorithm

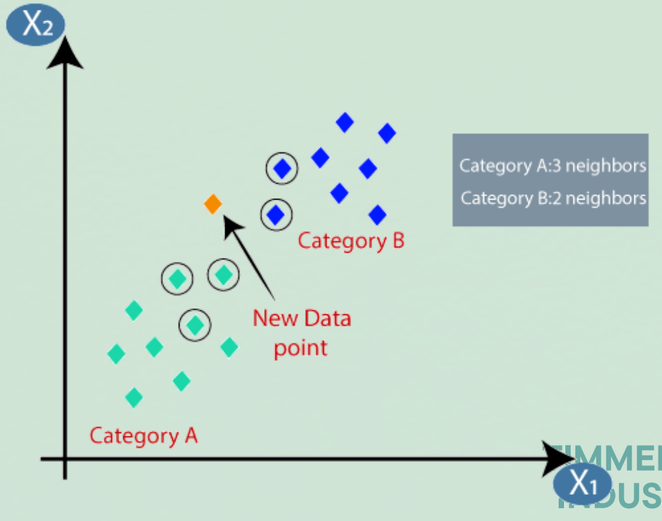
K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.







Random Forest Regression

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

We need to approach the Random Forest regression technique like any other machine learning technique

- Design a specific question or data and get the source to determine the required data.
- Make sure the data is in an accessible format else convert it to the required format.
- Specify all noticeable anomalies and missing data points that may be required to achieve the required data.
- Create a machine learning model
- Set the baseline model
- Train the data machine learning model.
- Provide an insight into the model with test data
- Now compare the performance metrics of both the test data and the predicted data from the model.
- If it doesn’t satisfy your expectations, you can try improving your model accordingly or dating your data, or using another data modeling technique.
- At this stage, you interpret the data you have gained and report accordingly.

Our Progress So Far

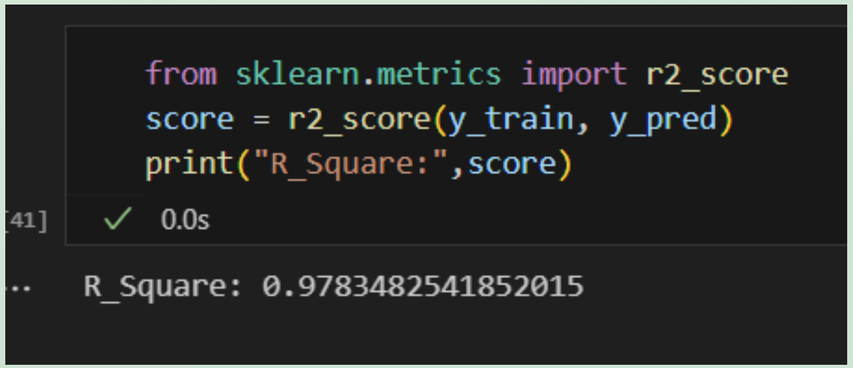
- Implementation of the entire code along with Algorithm Optimization
- Use of META MODELS to get better R^2 values (better than proposed by paper)
- Relative Importance of Parameters with data analysis on partitioned dataset.

LINK:

https://colab.research.google.com/drive/1thMZSYOQMpUA6OqliFYXuL-rnN2nlJeE?usp=sharing

META Model I

- Using a META Model to train the dataset on level_0 with Decision Tree Regressor and on level_1 using the estimators of level_0 by K-Neighbors Regression.
- This model improves the Variance (R^2) to **97.83%** which is significantly better than 94.75% (using KNN) and 95.37% (using Random Forest Regression)



META Model II

- Using a META Model to train the dataset on level_0 with Random Forest Regressor and on level_1 using the estimators of level_0 by K-Neighbors Regression.
- Parameters used for training: ***n_estimators = 933, max_features = "sqrt"***
- This model improves the Variance (R^2) to **98.45%** which is significantly better than 94.75% (using KNN) and 95.37% (using Random Forest Regression)

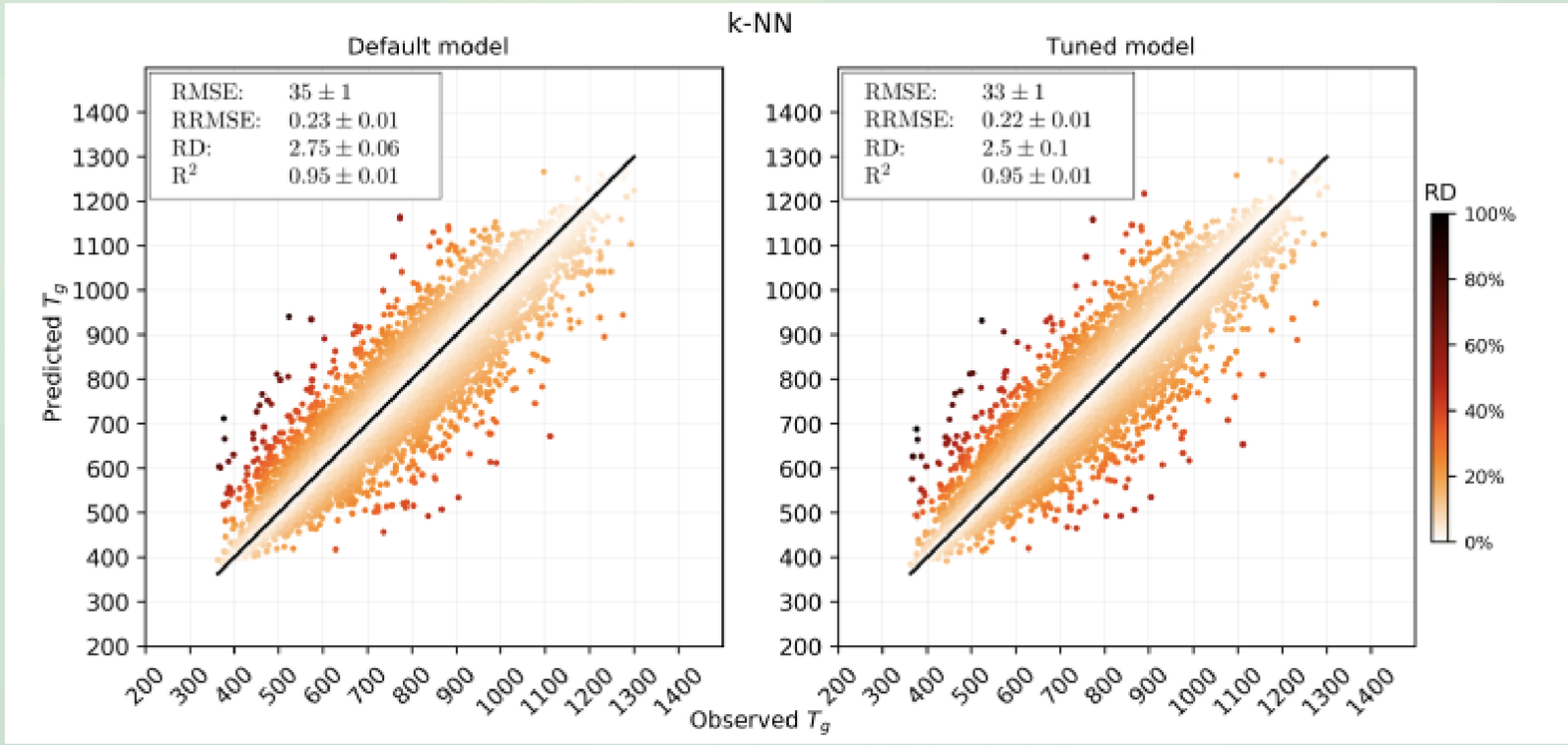
```
from sklearn.metrics import r2_score
score = r2_score(y_train, y_pred)
print("R_Square:",score)
```

✓ 0.0s

```
R_Square: 0.9845241876702696
```

Algorithm Parameter Optimization:

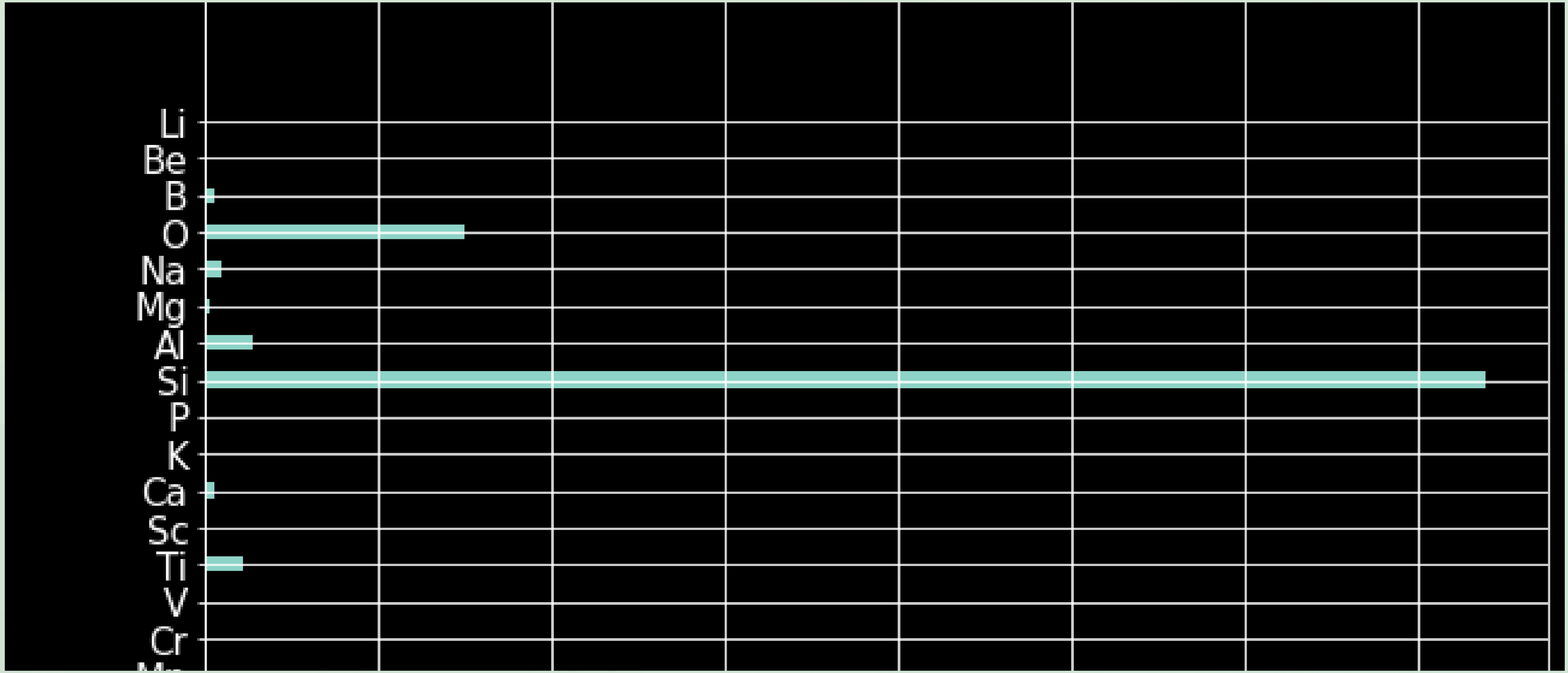
- The algorithms demonstrated (K-NN and Random Forest Regression) have parameters which can be optimized for best results of the dataset.



Relative Importance of Parameters

- The Relative variable importance graph plots the predictors in order of their effect on model improvement when splits are made on a predictor over the entire forest. The variable with the highest improvement score is set as the most important variable, and the other variables follow in order of importance.
- Relative variable importance standardizes the importance values for ease of interpretation. Relative importance is defined as the percent improvement with respect to the most important predictor, which has an importance of 100%.
- An element with high relative importance is used more often in the trees of the Random Forest and is thereby an important parameter.

Relative Importance of Parameters



*Full chart plotted on Colab file

Analysis on Partitioned Dataset

- As mentioned in the paper, the dataset is divided into two parts: high_extreme_values and low_extreme_values based on Tg (>= 1150K) and Tg(<= 450K) respectively.
- The regression coefficient is found out for the the partitioned datasets and the results are shown in the IPYNB file.

Thank you!