Report submission for CLD881

Dual Degree Project Entitled

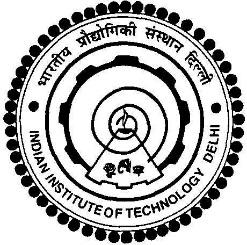
Surface Binding Energy Prediction using

Machine Learning Models

Submitted by

Shivanshu Verma

(2015CH70186)



Department of Chemical Engineering

Indian Institute of Technology Delhi

New Delhi - 110016

Nov 2019

|  |  |
| --- | --- |
| Supervisor’s signature | Date |
|  |  |

# Table of Contents

[Abstract 5](#_Toc24979728)

[Introduction 6](#_Toc24979729)

[Data Extraction 7](#_Toc24979730)

[Feature Engineering 8](#_Toc24979731)

[Supervised Models 9](#_Toc24979732)

[Results and Discussion 11](#_Toc24979733)

[References 16](#_Toc24979734)

[Appendix 17](#_Toc24979735)

# List of Figures

[Figure 1: Machine Learning broad classification 6](#_Toc24978751)

[Figure 2 : Feature engineering for different bimetallic catalysts 8](#_Toc24978752)

[Figure 3 : Linear Regression over 2D input 9](file:////Users/shivanshu/Downloads/MTP%20CLD881%20Report.docx#_Toc24978753)

[Figure 4 : Support Vectors over 2D input 9](file:////Users/shivanshu/Downloads/MTP%20CLD881%20Report.docx#_Toc24978754)

[Figure 5 : Gradient Boosting Ensemble 10](file:////Users/shivanshu/Downloads/MTP%20CLD881%20Report.docx#_Toc24978755)

[Figure 6 : K nearest neighbours 10](file:////Users/shivanshu/Downloads/MTP%20CLD881%20Report.docx#_Toc24978756)

[Figure 7: Percentage error comparison for Carbon on 111 11](#_Toc24978757)

[Figure 8 : Percentage error comparison for O on 111 12](#_Toc24978758)

[Figure 9 : R square score comparison for C on 111 12](#_Toc24978759)

[Figure 10 : R square comparison for O on 111 13](#_Toc24978760)

[Figure 11 : RMSE comparison for C on 111 in eV 13](#_Toc24978761)

[Figure 12 : RMSE comparison for O on 111 (eV) 14](#_Toc24978762)

# 

# List of Tables

[Table 1 : Extracted dataset of 88, 587 reactions 7](#_Toc24946003)

[Table 2 : Filtered dataset for Carbon reactions 7](#_Toc24946004)

[Table 3 : Filtered dataset for oxygen reactions 7](#_Toc24946005)

# Abstract

Binding energy of an adsorbate is the energy required to bind or leave the surface by an atom. The surface may have many sites as terrace, kink and step sites and the binding energy for all of them would be different. To understand the phenomenon of catalysis, binding energy plays an important role. According to Sabatier Principle, binding energy is a good descriptor for catalytic activity. Using DFT calculations to find adsorbate binding energies on the surface takes a lot of time and computation even on HPC. Machine learning can predict the results based on previous results training in no time. More the data, the more accurate results.

In this project, different machine learning models were applied to predict the binding energy of oxygen and carbon atoms on different catalytic surfaces. The features used for determining binding energy are no complex but the periodic properties of elements such as electronegativity, period, group, ionization energy, radius, atomic number, atomic mass, etc. Different machine learning models were compared, and root mean square error is found for every model applied. The results were found to be useful and there is full potential to improve.

# 

# Introduction

Machine learning is the capability of computers to learn based on the previous experiences. Now a days, it has huge applications in prediction, classification and clustering the data to extract similar data clusters from a set a data. Due to its ability to learn and predict data, we can use this technology in any field which has predictable data such as medical, cancer detection, image processing, automated games, etc. Broadly it is classified into two types based on the learning i.e. Supervised and unsupervised. Further in supervised, we can classify that into classification and regression. Figure 1 shows the broad classification based on learning techniques.

Machine Learning

Unsupervised

Supervised

Classification

Regression

Clustering

Figure 1: Machine Learning broad classification

Learning is supervised if the input values of data and output values of data both are given. For e.g. prediction of rain based on humidity, pressure, temperature of the day. In this dataset, values of all the variables is given for a year and whether there is rain or not is also given. Further into this, we can predict true or false, whether rain will fall or not, this is called classification problem, if we had to predict how much rain is going to be there, then it would be regression problem. So how can we use this technique to predict binding energy? Here, based on the different properties of catalyst atoms and surfaces, we will predict what should be the binding energy of an atom on that. This prediction will fall into the category of supervised regression problem.

# Data Extraction

All the data for prediction of binding energies is taken from *catalysis-hub.org* from the publication "High-throughput Calculations of Catalytic Properties of Bimetallic Alloy Surfaces". The data is extracted using python script in JSON format. Finally, it is converted into CSV format using another python script. Table 1 represent the three different rows of overall data extracted.

|  |  |  |  |
| --- | --- | --- | --- |
| Equation | Chemical Composition | Facet | Reaction Energy |
| 0.5N2(g) + \* -> N\* | Sc3V9 | 111 | -2.810391181 |
| 0.5H2(g) + \* -> H\* | Pt9Ti3 | 111 | -0.292809865 |
| CH4(g) - 2.0H2(g) + \* -> C\* | Ag6Pd6 | 211 | 4.541566089 |

Table : Extracted dataset of 88, 587 reactions

This data is then filtered according to the reactions and surfaces. Table 2 and table 3 shows filtered data for oxygen and carbon reactions and 111 surface to determine the binding energy of oxygen and carbon on different chemical compositions of bimetallic catalysts.

|  |  |  |  |
| --- | --- | --- | --- |
| Equation | Chemical Composition | Facet | Reaction Energy |
| CH4(g) + 5\* -> C\* + 4H\* | Ag3Au9 | 111 | 5.951248021 |
| CH4(g) + 5\* -> C\* + 4H\* | Ag3Co9 | 111 | 0.47687923 |
| CH4(g) + 5\* -> C\* + 4H\* | Ag3Mo9 | 111 | -2.630615364 |

Table : Filtered dataset for Carbon reactions

|  |  |  |  |
| --- | --- | --- | --- |
| Equation | Chemical Composition | Facet | Reaction Energy |
| H2O(g) - H2(g) + \* -> O\* | Cr9Sn3 | 111 | -0.1206 |
| H2O(g) - H2(g) + \* -> O\* | W9Zr3 | 111 | -1.4925 |
| H2O(g) - H2(g) + \* -> O\* | Cd9Pd3 | 111 | 2.9345 |

Table : Filtered dataset for oxygen reactions

This filtered data is then featured using different periodic properties of the elements shown in “Chemical Composition” column. A feature vector of 24 properties is used as the input value to the machine learning model and predict “Reaction Energy” column as the binding energy of the atom.

# Feature Engineering

Feature engineering is the process of converting input values to the features which a machine can understand and differentiate different outputs on the basis of input features. Feature engineering plays an important role in how accurate the model is and how precisely it can differentiate between two different values. In this project for calculating binding energies of an oxygen and carbon atom on different catalytic surfaces and for 111 site only, we have taken periodic properties of the metals in bimetallic catalyst. Figure 2 shows how the features are selected.

Input Catalyst

(AgAu3)

Ag

(12 properties)

Au

(12 properties)

Vector with 24 values

Figure : Feature engineering for different bimetallic catalysts

**Feature Vector**

**[** atomic no**,** atomic mass**,** group**,** period**,** radius**,** electronegativity**,** melting point**,** boiling point**,** fusion enthalpy**,** density**,** ionization energy**,** surface energy**]**

Feature vector shown above shows the feature or properties of one atom of bimetallic catalyst. As shown in figure 2, combining both of them results in final input vector. As the ratio of two atoms is 3 for all the bimetallic atoms in the data, ratio is not used as feature. d band center, pauling, work function and electron affinity were also added in case of carbon binding energy prediction in the feature vector to improve results.

To form the feature vectors, a python script was written which firstly splits the catalyst into two metals and find their ratio. After that, script given element wise features as input add the features of both the elements and concatenate them with corresponding binding energy.

# Supervised Models

1. Linear Regression

This model assumes a linear relationship between input features and output value. The following equation parameters are determined in the fitting process:

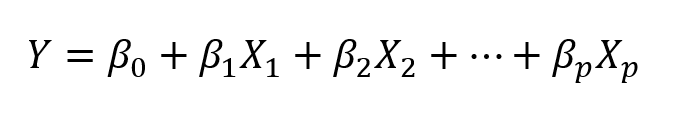


Figure 3 shows the example of linear regression on 2-dimensional input.

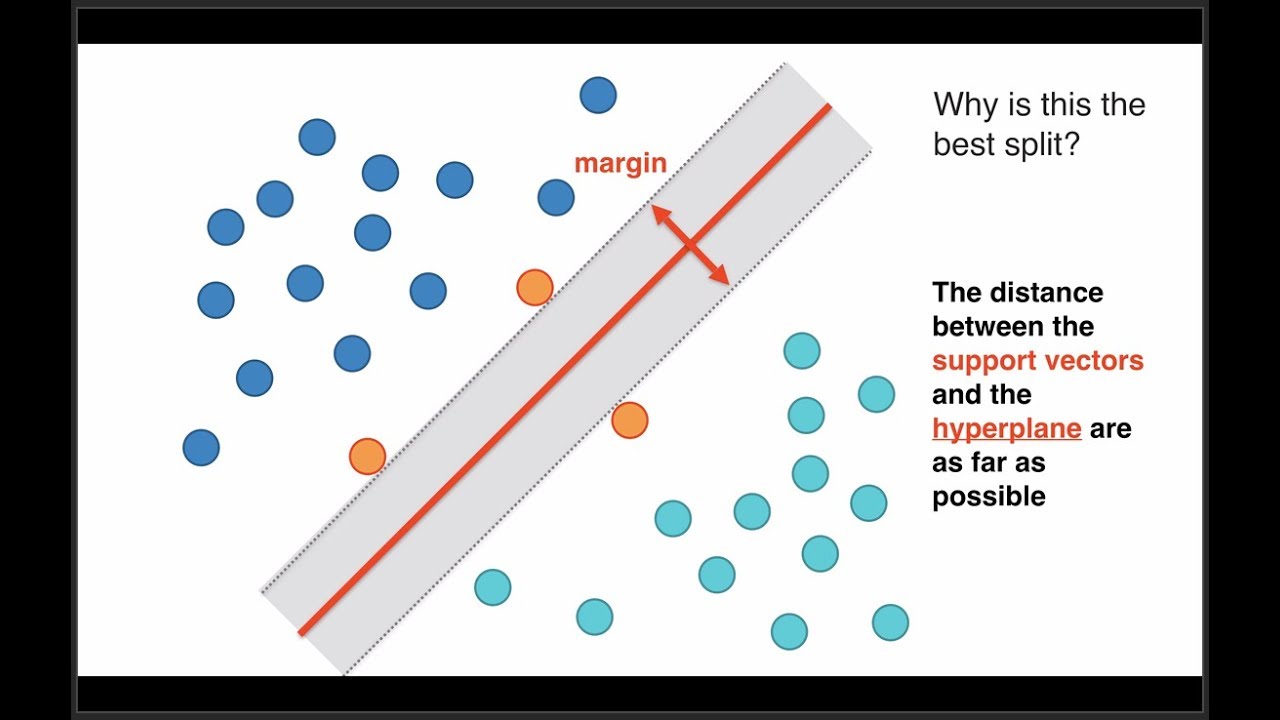
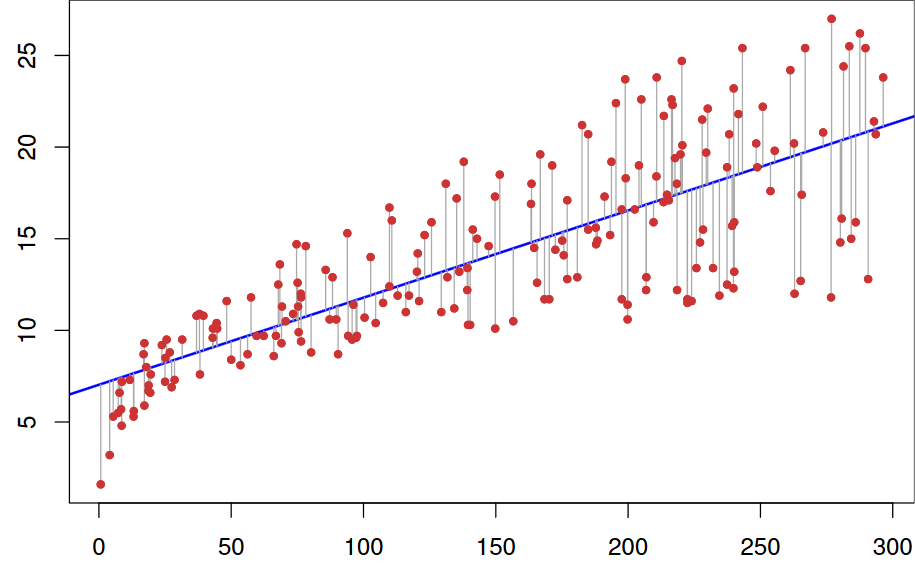


Figure : Linear Regression over 2D input

Figure : Support Vectors over 2D input

1. Support Vector Regression

This method marginalizes the points on the basis of support vectors. The algorithm minimizes the maximum distance of points from the decision boundary. Following equation parameters are determined in the fitting process:

Minimize:

Figure 4 shows the margins and example of 2D input classification using Support vector machines.

1. Gradient Boosting Regression

This model is the ensemble of weak prediction model decision trees. Following equation is optimized in the fitting process:

Figure 5 shows how gradient boosting works.

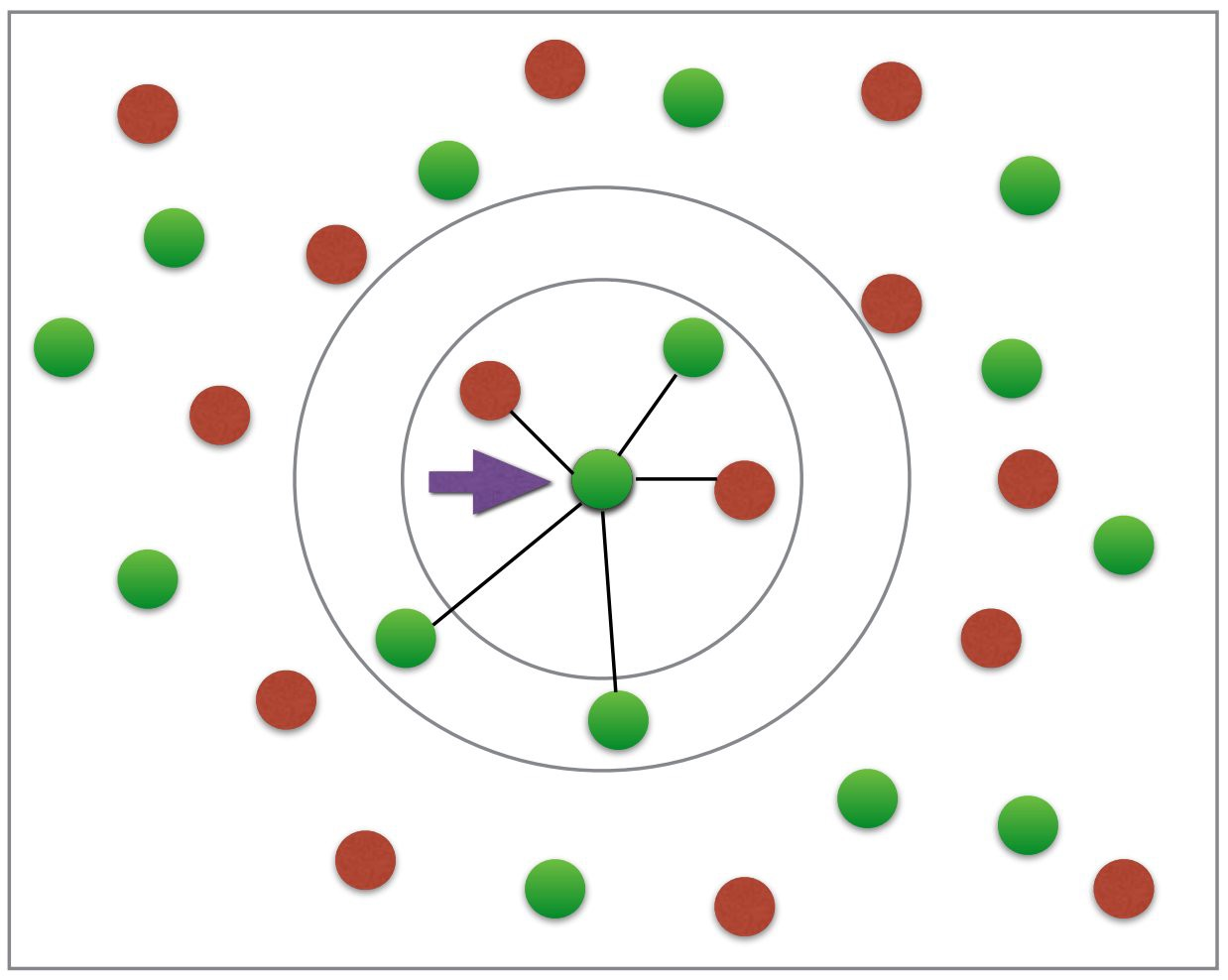
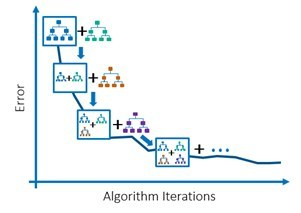


Figure : Gradient Boosting Ensemble

Figure : K nearest neighbours

1. K Nearest Neighbors Regression

This algorithm is based on unsupervised k means classification in which total k clusters are made based on the distances between different points and points lying within them are defined in the respective clusters.

In k nearest neighbours, the same algorithm is used for supervised classification but her we define k nearest neighbours of a point based on the values of that point and other points. Figure 6 shows the example of k nearest neighbours with two classes.

# Results and Discussion

A total of four different well-known machine learning models were applied on carbon and oxygen dataset to predict their binding energy using their periodic properties as input features. For optimization, different models were optimized using different parameters in GridSearchCV. For GBR, a combination of 3\*3\*3\*5\*4 parameters were used to find the best suiting optimal method with 5-fold verification leading to 900 combinations of parameters. All the models were optimized the same way with different number of combinations for each.

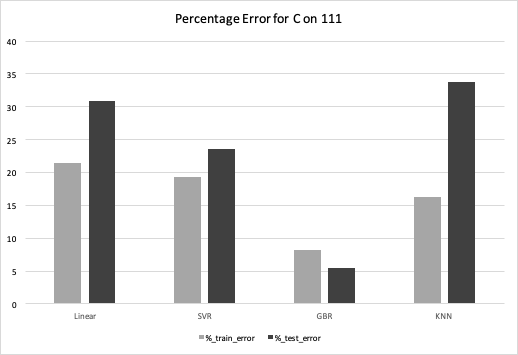


Figure : Percentage error comparison for Carbon on 111

To test the correctness of model, a test dataset has been splitted from the original dataset and has never been trained in anyway and was completely unknown to the system. Three different measures were used to check the precision and accuracy of the model. As shown in figure 7 and figure 8, the graph compares different models on the basis of percent accuracy observed on carbon and oxygen dataset respectively.

Figure : Percentage error comparison for O on 111

Another measure taken to analyze the goodness of model was R square score. More the r square score close to 1, better the model. Figure 9 and figure 10 shows the r square score of different models for carbon and oxygen dataset respectively.

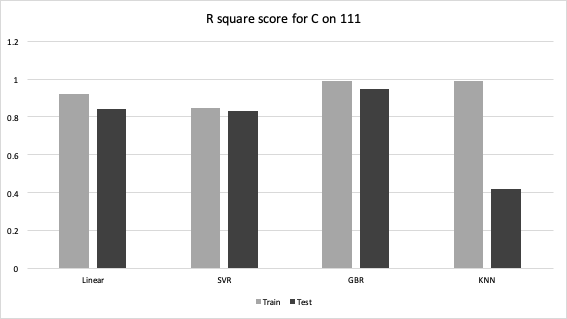


Figure : R square score comparison for C on 111

Figure : R square comparison for O on 111

Third measure was taken as root mean square error value which sums up the magnitude of error in all the values and is in unit eV for the current dataset. Figure 11 and figure 12 shows the comparison of all the models on the basis of RMSE for carbon and oxygen respectively. This value must be as low as possible and a minimum of 0.3 eV is the threshold as this means approx. 30 KJ/mol which is the expected error by DFT calculations also. Aim is to get better and better and reduce the error as much as we can.

A screenshot of a cell phone

Description automatically generated

Figure : RMSE comparison for C on 111 in eV

Figure : RMSE comparison for O on 111 (eV)

Till now in all the models applied so far, GBR gave the best results in terms of RMSE and percentage error. As we can see, in other models than GBR, training error is very low whereas GBR has comparable training and test errors for oxygen dataset (figure 12). In spite of this, GBR shows best results among four of them. The reason for that is overfitting of data. Other models overfits the training data and shows better results for training subset while worst when the data is new i.e. test dataset. But in case of GBR, we can see that model is fitting better and giving similar results for both training and test dataset.

As the best optimization has been done till now, the scope of reducing error further with the same dataset and features is difficult. To reduce the error up to the threshold of 1-3 eV, we need to understand the data more deeply and change the feature vectors. A good feature engineering can result in very good results and in the future work, I would like to apply neural networks and work upon features to improve results.

# References

1. Shivam Saxena, Tuhin S. Khan, Fatima Jalid, Manoj C. Ramteke, M. Ali Haider, In Silico High Throughput Screening of Bimetallic and Single Atom Alloys Using Machine Learning and Ab Initio Microkinetic Modelling,

2. https://www.simplilearn.com/regression-machine-learning-tutorial

3. http://www.statsoft.com/textbook/support-vector-machines

4. http://www.cse.chalmers.se/~richajo/dit866/lectures/l8/gb\_explainer.pdf

# Appendix

### Sample code for Gradient Boosting on carbon 111

import os, sys

import numpy as np

import pandas as pd

from math import sqrt

from sklearn.svm import SVR

from joblib import load, dump

from sklearn import model\_selection

from sklearn.decomposition import PCA

from sklearn.metrics import mean\_squared\_error

from sklearn.ensemble import ExtraTreesRegressor

from sklearn.model\_selection import GridSearchCV

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.linear\_model import LinearRegression

from sklearn.neighbors import KNeighborsRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_predict

from sklearn.ensemble import GradientBoostingRegressor

dirname = os.path.dirname(\_\_file\_\_)

sys.path.insert(0, dirname)

filename = os.path.join(dirname, 'featured\_data\_111\_C.csv')

GBR\_model\_name = os.path.join(dirname, 'GBR\_C\_111.pickle')

data = pd.read\_csv(filename)

data = data.drop(['Unnamed: 0'],axis=1)

Y = data.loc[:,'reactionEnergy'].values

X = data.loc[:, 'AN' : 'dBandCenter'].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.20, shuffle=True)

def Best\_estimator\_grid(X, Y, clf, params, cv):

""" Returns the best estimator from a dictionary of parameters"""

print("Within Best\_estimator")

clf\_grid = GridSearchCV(clf, params, cv=cv, n\_jobs=2, verbose=10)

clf\_grid.fit(X, Y)

return clf\_grid

clf = GradientBoostingRegressor()

params = {'n\_estimators': [ 10, 100, 200, 500, 1000],

'learning\_rate':[.1, 1, .01],

'loss':['ls', 'lad', 'huber'], 'max\_depth' : [2, 3, 4],

'max\_features' : [10, 15, 20, 28] }

print("Finding optimal GBR model ............")

optimal\_model = Best\_estimator\_grid(X\_train, y\_train, clf, params, 5)

best\_model = optimal\_model.best\_estimator\_

print("Optimal Model of GBR is" , best\_model)

dump(best\_model, os.path.join(dirname, GBR\_model\_name))

print("Model has been dumped Successfully !!!")

print("Finding root mean square error ............")

errors=[]

errors\_train=[]

feature\_importances\_array=np.zeros(28)

error\_list=[]

error\_all\_i=np.zeros(28)

for i in range(0,100):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, shuffle=True)

best\_model.fit(X\_train, y\_train)

y\_train\_predicted=best\_model.predict(X\_train)

y\_test\_predicted=best\_model.predict(X\_test)

# Later find which values create most errors

errors.append(sqrt(mean\_squared\_error(y\_test, y\_test\_predicted)))

errors\_train.append(sqrt(mean\_squared\_error(y\_train, y\_train\_predicted)))

b=best\_model.feature\_importances\_

feature\_importances\_array=np.add(feature\_importances\_array,b)

i=i+1

print(i)

print ("Train Error" + str(sum(errors\_train)/float(len(errors\_train))))

print ("Test Error" + str(sum(errors)/float(len(errors))))

final\_importance=np.divide(feature\_importances\_array,100.0)

with open('GBR\_C\_111\_results.txt', 'w') as f:

print('Test\_error:', str(sum(errors)/float(len(errors))), file=f) # Python 3.x

print('Train\_error:', str(sum(errors\_train)/float(len(errors\_train))), file=f) # Python 3.x