Empirical Comparison of Machine Learning Models for Band Gap Prediction

A thesis submitted in partial fulfilment of the requirements for the award of the degree of

B.Tech.

in

Metallurgical and Materials Engineering

by

Anantha Natarajan S (Roll No. 112112008)

Ezhilvel ME (Roll No. 112112019)

Varadhan R (Roll No. 112112042)

METALLURGICAL AND MATERIALS ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPPALLI - 620015

MAY 2016

BONAFIDE CERTIFICATE

This is to certify that the project titled **Empirical Comparison of Machine Learning Models for Band Gap Prediction** the work done by

Anantha Natarajan S (Roll No. 112112008)

Ezhilvel ME (Roll No. 112112019)

Varadhan R (Roll No. 112112042)

in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Engineering** at the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI,** during the years 2012-2016.

Project Guide

Head of the Department

Project viva-voce held on _____

Internal Examiner

External Examiner

ABSTRACT

Exploring novel machine learning approaches to predict band gaps of binary compounds using their known physical features. The study also empirically compares various models developed, discussing their respective performances, advantages, shortcomings and flaws.

Data is mined from open sources enabling us to programmatically access several thousand training examples whose properties were calculated properly and consistently by a single research group. Using various data representation methods and after careful analysis, important features contributing to the desired property are identified. Different models are built using regression and its variations. We also analyze different combination of features and their impact on the prediction. The problem is also converted into a classification problem and classifiers like Support Vector Machines and Random Forest classifiers are used to estimate the band gap.

Finally the performance of each model is compared and their respective accuracies are discussed. For the sake of computational simplicity we propose models that work with binary compounds, but these can easily be extended to work on more complex compounds and alloys too.

ACKNOWLEDGEMENTS

The contribution of many different people, in their different ways, has made this possible. I would like to extend my appreciation to the following:

Dr. S. Kumaran, Associate Professor in the Department of Metallurgical and Materials Engineering, our project guide, for making this project possible. His enthusiasm, support, guidance, advice throughout the project, as well as his painstaking effort in proofreading the drafts, is greatly appreciated.

Dr. S.P. Kumaresh Babu, Head of the Department, Metallurgical and Materials Engineering for the valuable insights he has shared from time to time and steadfast encouragement to complete this project. He has been our inspiration as we hurdled through obstacles in the successful completion of our project.

Dr. Srinivasan Sundarrajan, Director, National Institute of Technology, Tiruchirappalli, for providing all facilities to carry out this project.

Finally, we are extremely thankful to all other faculty members, as well as our friends and family for their help and encouragement during the execution of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Since the late 20th century the exponential growth of the electronics and communication sector has led to numerous innovations in the semiconductor industry. Everyday newer and more efficient semiconductors are being discovered. The main property the scientists are targeting is the conductivity of the material this is directly related to the band gap of it.

The concept of band gap pertains to solid state physics: it generally refers to the energy difference in electron volts between the top of the valence band and the bottom of the conduction band in insulators and semiconductors. It helps us understanding characteristics of conductors, semiconductors and insulators. The band gap is the minimum amount of energy required for an electron to break free of its bound state. When the band gap energy is met, the electron is excited into a free state, and can therefore participate in conduction.

1.1 COMPLEXITIES IN BAND GAP PREDICTION

The band gap being an intrinsic property, is calculated using experiments like UV spectroscopy or differential and cyclic voltammetry. These experiments require large, expensive equipments and are tedious. Some materials are difficult to handle and require special environments to perform these experiments. Calculating band gaps accurately is still one of the unsolved problems in solid state physics. There is no equation or direct solution to calculate band gaps accurately. To help overcome these difficulties we propose and compare machine learning models to recognize meaningful patterns in band gap values across thousands of compounds and their chemical properties. Predicting the band gap, or more specifically predicting the range in which the band gap of a new material would lie between would give researchers and students a good idea of what to expect in the experiments before going for more elaborate ways of band gap calculation.

1.2 OBJECTIVES

The objective is to build a reliable (near accurate) model to determine materials properties using machine learning algorithms and analyze the results of different models to support materials research facilitated by

1. Building machine learning models using different (naive and physical) features and models to improve the accuracy
2. Comparing the models against given material properties using different methods and varying combinations of properties
3. Developing a web application to encapsulate the complete tool

CHAPTER 2

LITERATURE SURVEY

Accelerating materials property predictions using machine learning

This paper, in the present contribution, shows that efficient and accurate prediction of a diverse set of properties of material systems is possible by employing machine (or statistical) learning methods trained on quantum mechanical computations in combination with the notions of chemical similarity. According to them harnessing such learning paradigms extends recent efforts to systematically explore and mine vast chemical spaces, and can significantly accelerate the discovery of new application-specific materials. The materials discovery process can be significantly expedited and simplified if we can learn effectively from available knowledge and data.

Owing to the staggering compositional and configurational degrees of freedom possible in materials, it is fair to assume that the chemical space of even a restricted subclass of materials (say, involving just two elements) is far from being exhausted, and an enormous number of new materials with useful properties are yet to be discovered. Given this formidable chemical landscape, a fundamental bottleneck to an efficient materials discovery process is the lack of suitable methods to rapidly *and* accurately predict the properties of a vast array (within a subclass) of new yet-to-be-synthesized materials.

Prediction model of band-gap for AX binary compounds by combination of density functional theory calculations and machine learning techniques

This paper shows how machine learning techniques are applied to make prediction models of the G0W0 band-gaps for 156 AX binary compounds using Kohn-Sham band-gaps and other fundamental information of constituent elements and crystal structure as predictors. Ordinary least square regression (OLSR), least absolute shrinkage and selection operator (LASSO) and non-linear support vector regression (SVR) methods are applied with several levels of predictor sets. When the Kohn-Sham band-gap by GGA (PBE) or modified Becke-Johnson (mBJ) is used as a single predictor, OLSR model predicts the G0W0 band-gap of a randomly selected test data with the root mean square error (RMSE) of 0.54 eV. When Kohn-Sham band gap by PBE and mBJ methods are used together with a set of various forms of predictors representing constituent elements and crystal structures, RMSE decreases significantly. Their best model by SVR yields the RMSE of 0.18 eV. According to them a large set of band-gaps estimated in this way should be useful as predictors for materials exploration.

CHAPTER 3

DATA CLEANING AND VISUALIZATION

The objective is to structure the data to facilitate the data analysis which is set out to perform. It is only after data is tidy that is useful for data analysis. Tidy data makes it easy to perform the tasks of data analysis with tools that are designed for tidy data:

1. Manipulation: Variable manipulation such as aggregation, filtering, reordering, transforming and sorting.
2. Visualization: Summarizing data using graphs and charts for exploration and exposition.
3. Modelling: This is the driving inspiration for tidy data, modelling is what assigned out to do.

3.1 MAJOR STEPS IN DATA PREPROCESSING:

Data cleaning: Identify and fill in missing values, detect and remove noisy data and outliers.

Data transformation: Normalize data to reduce dimensions and noise.

Data reduction: Sample data records or attributes for easier data handling.

Data discretization: Convert continuous attributes to categorical attributes for ease of use with certain machine learning methods.

Text cleaning: remove embedded characters which may cause data misalignment, for e.g., embedded tabs in a tab-separated data file, embedded new lines which may break records, etc.

3.2 DATA PREPARATION PROCESS:

The process for getting data ready for a machine learning algorithm can be summarized in three steps:

Step 1: Select Data

Step 2: Preprocess Data

Step 3: Visualize Data

3.2.1 SELECT DATA:

This step is concerned with selecting the subset of all available data that will be considered working with. Assumptions are made about the data requirement and are recorded to test them later if needed. The data regarding the binary system and its bandage energy is obtained from an REST API of https://materialsproject.org[1]. The obtained subset data has 4096 rows of different binary systems. The data of binary compound systems and its band gap energy is collected in a csv format for further processing and usage. The generated csv file contains 4096 binary compound systems and its band gap energy.

1	LiH,2.981
2	BeH2,5.326
3	B9H11,2.9118
4	B2H5,6.3448
5	BH3,5.3234
6	B5H7,3.5551
7	H34C19,5.4526
8	H3N,4.3287
9	H2O,5.5175
10	HF,6.7187
11	NaH,3.7382
12	MgH2,3.7097
13	AlH3,2.1979
14	SiH4,6.5621
15	H2S,4.3246
16	HCl,5.8162
17	KH,3.4331
18	CaH2,3.028
19	ScH2,0.0136
20	TiH2,0.0161
21	VH2,0
22	V2H,0
23	CrH,0
24	NiH,0
25	HBr,4.681
26	RbH,2.9163
27	SrH2,3.1544
28	YH3,0.0405
29	YH2,0
30	ZrH2,0.0181
31	NbH2,0
32	HRh,0
33	HPd,0.216
34	HI,4.0739
35	CsH,2.5693
36	BaH2,2.8699
37	LaH2,0
38	CeH2,0.0015
39	PrH2,0.0583
40	NdH2,0.0445
41	PmH3,0
42	PmH,0

Fig 3.1 Band gap CSV

3.2.2 PREPROCESS DATA:

After selection of data, the data needs to be tuned for building machine learning models. The preprocessing steps are about getting the selected data into a form that is used to emulate machine learning models.

Three common data preprocessing steps are formatting, cleaning and data transformation:

Formatting: Formatting represents the way the data is stored either in a database or a local readable repository/file. The data is stored in a convenient way for easier I/O processing. Python's numpy [11] is used to read the csv data file for analysis and processing.

Cleaning: Cleaning data is the removal of noisy data or fixing missing samples/datasets. Incomplete datasets and outliers are addressed before it is used to build models. There may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

Data transformation: The binary compound system is transformed into various features that depicts the model. The following are the features that are considered:

- Electronegativity difference of elements in binary compound
- Atomic fractions of elements in binary compound
- Position of elements of binary compounds in periodic table
- Atomic number of elements in binary compound

To derive these features from a single binary compound, Pymatgen [7] is used. Pymatgen (Python Materials Genomics) is a robust, open-source Python library for materials analysis.
The following are some of the main features of Pymatgen:

1. Highly flexible classes for the representation of Element, Site, Molecule, Structure objects.
2. Powerful analysis tools, including generation of phase diagrams, Pourbaix diagrams, diffusion analyses, reactions, etc.
3. Electronic structure analyses, such as density of states and band structure.
4. Integration with the Materials Project REST API [6].

It is (fairly) robust. Pymatgen is used by thousands of researchers, and is the analysis code powering the Materials Project. The analysis it produces survives rigorous scrutiny every single day. Bugs tend to be found and corrected quickly.
It is well documented. A fairly comprehensive documentation has been written to help users get to grips with it quickly.
It is fast. Many of the core numerical methods in pymatgen have been optimized by vectorizing in numpy. This means that coordinate manipulations are extremely fast and are in fact comparable to codes written in other languages. Pymatgen also comes with a complete system for handling periodic boundary conditions.

NumPy is used for data exploration and manipulation. NumPy is the fundamental package needed for scientific computing with Python. This package contains:

1) A powerful N-dimensional array object.
2) Sophisticated (broadcasting) functions.
3) Tools for integrating C/C++ and Fortran code.
4) Useful linear algebra, Fourier transform, and random number capabilities.

3.2.3 DATA VISUALIZATION:

Data visualization is a general term that describes any effort to understand the significance of data by placing it in a visual context. Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization software. A primary goal of data visualization is to communicate information clearly and efficiently via statistical graphics, plots and information graphics. Numerical data may be encoded using dots, lines, or bars, to visually communicate a quantitative message. Effective visualization helps users analyze and reason about data and evidence. It makes complex data more accessible, understandable and usable.

Matplotlib[4] is a python 2D plotting and data visualization library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

The following are some of the main features of Matplotlib:

1. Very fast when processing large datasets.
2. Easier to manipulate plot details.
3. Standalone program: no external dependencies.
4. Deep integration with Python.

Matplotlib is used to draw relational plots between the bandgap energy and the features. The plot drawn is used to visualize the meaningful patterns hidden within the data set.

Plot between bandgap and Electronegativity difference:

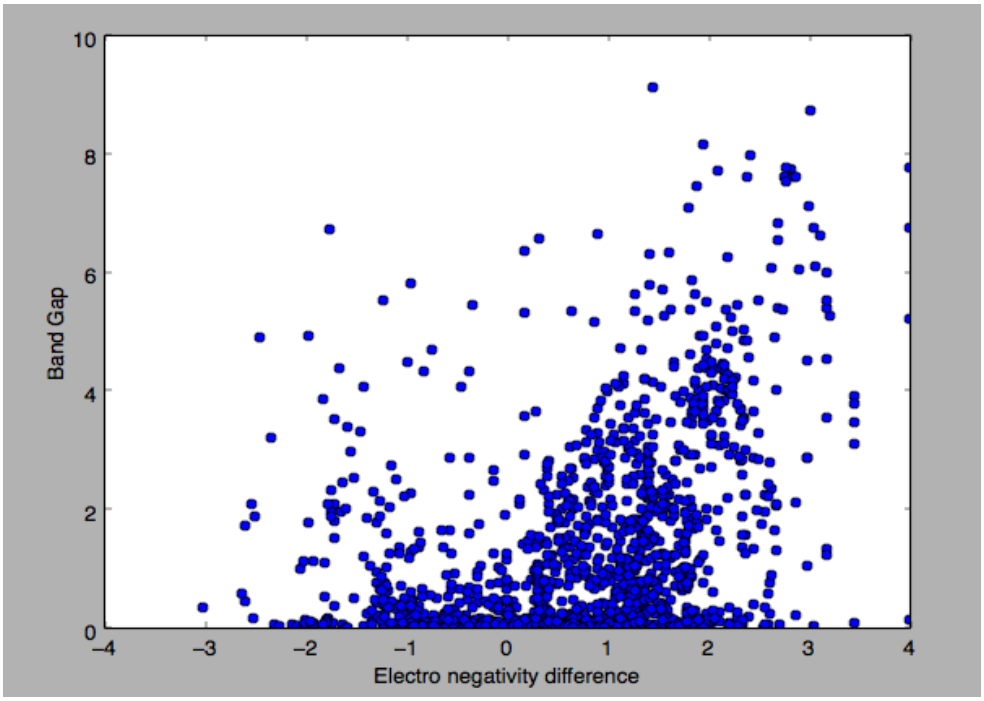


Fig 3.2 Plot Bandgap vs Electronegativity difference

Plot between bandgap and atomic weight of the binary compound:

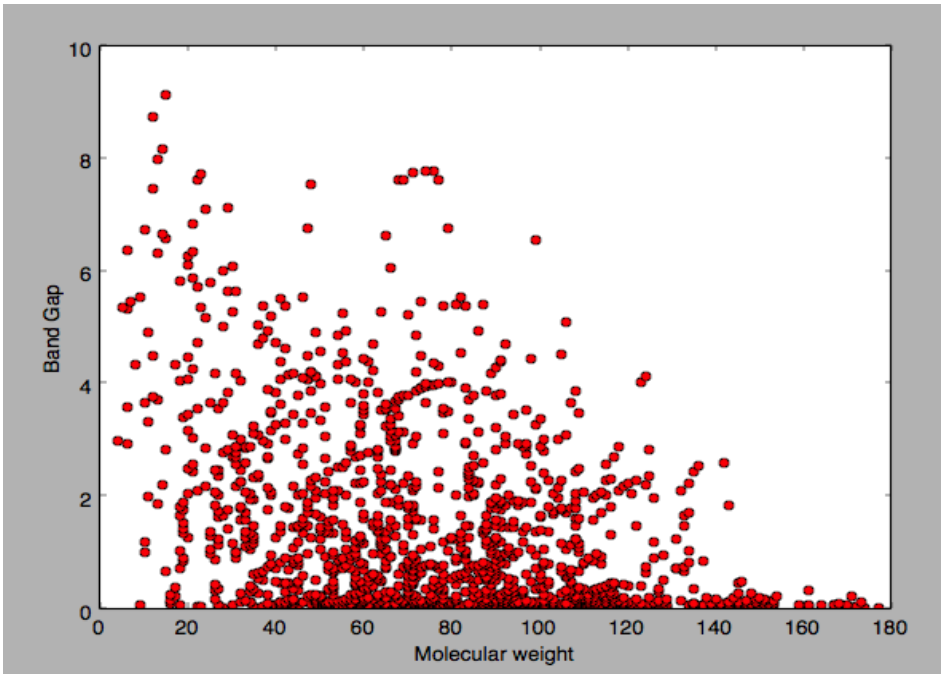


Fig 3.3 Band gap vs Atomic Weight

A comparison of 3 properties:

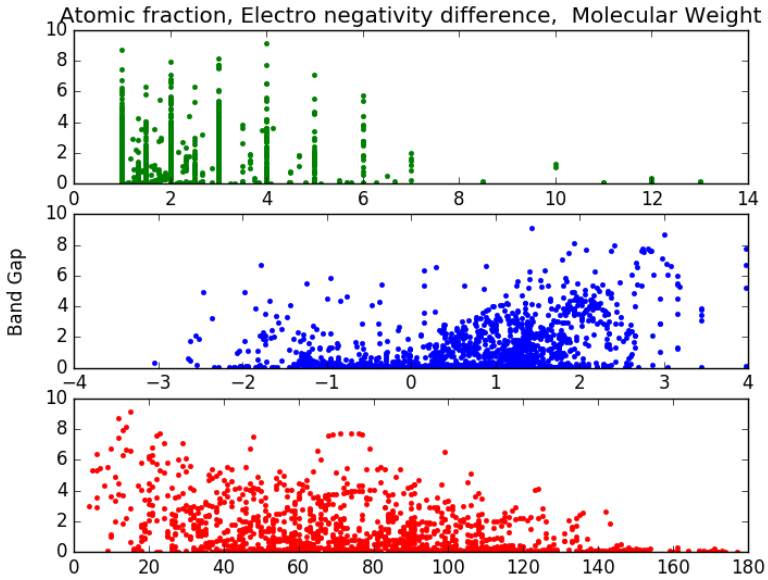


Fig 3.4 Subplot of all the three features

CHAPTER 4

PROPOSED MODELS AND ANALYSIS

4.1 SCHEMATIC BLOCK DIAGRAM

The above block diagram explains the flow of the project. After data collection and cleaning, preprocessing and feature extraction, machine learning algorithms are used to predict the band gap.

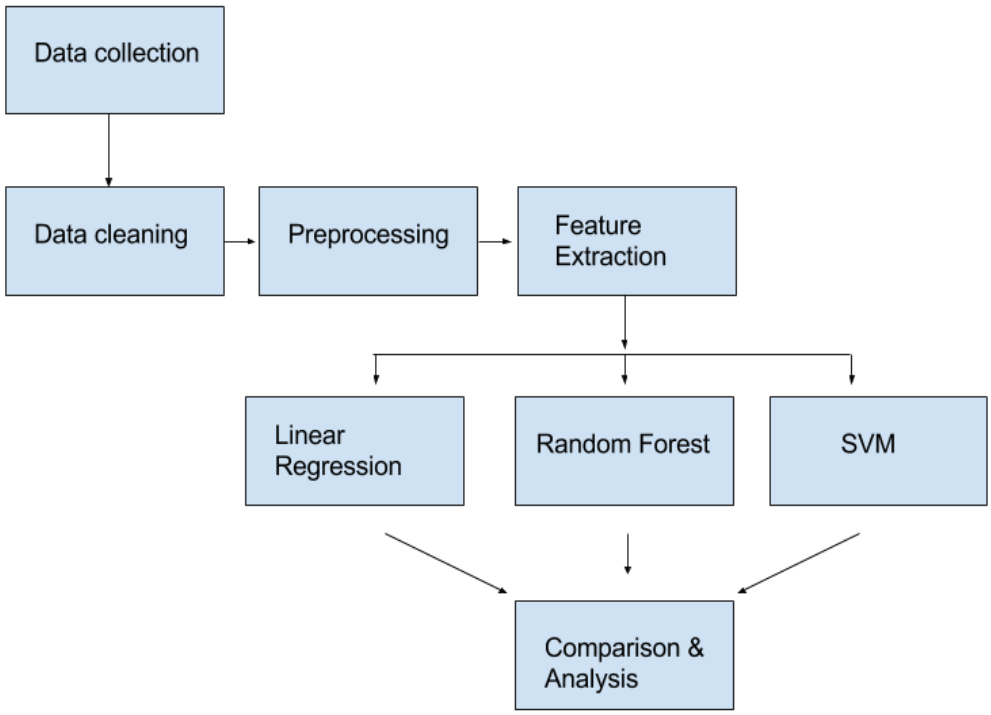


Fig 4.1 Block diagram

4.2 FEATURES

The main features contributing to the model were selected after performing Principal Component Analysis (PCA) on the available input features. PCA can be thought of as fitting an n-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipse is small, then the variance along that axis is also small, and by omitting that axis and its corresponding

principal component from our representation of the dataset, we lose only a commensurately small amount of information.

To find the axes of the ellipse, we must first subtract the mean of each variable from the dataset to center the data around the origin. Then, we compute the covariance matrix of the data, and calculate the eigenvalues and corresponding eigenvectors of this covariance matrix. Then, we must orthogonalize the set of eigenvectors, and normalize each to become unit vectors. Once this is done, each of the mutually orthogonal, unit eigenvectors can be interpreted as an axis of the ellipsoid fitted to the data. The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.

The first loading vector $\mathbf{w}_{(1)}$ thus has to satisfy:

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \sum_i (t_1)^2_{(i)} \right\} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \sum_i \left(\mathbf{x}_{(i)} \cdot \mathbf{w} \right)^2 \right\}$$

Equivalently, writing this in matrix form gives

$$\mathbf{w}_{(1)} = \arg\max_{\|\mathbf{w}\|=1} \{ \|\mathbf{X}\mathbf{w}\|^2 \} = \arg\max_{\|\mathbf{w}\|=1} \left\{ \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \right\}$$

Since $\mathbf{w}_{(1)}$ has been defined to be a unit vector, it equivalently also satisfies

$$\mathbf{w}_{(1)} = \arg\max \left\{ \frac{\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \right\}$$

Based on PCA and the availability of the feature data sets to train on, we selected the following properties to train the model:

- Atomic fraction
- Electronegativity
- Molecular weight
- Group of the constituent elements

4.3 VECTORIZATION

Vectorization of a matrix is a linear transformation which converts the matrix into a column vector. Specifically, the vectorization of an m×n matrix A, denoted by vec(A), is the mn × 1 column vector obtained by stacking the columns of the matrix A on top of one another:

$$\operatorname{vec}(A) = [a_{1,1}, \ldots, a_{m,1}, a_{1,2}, \ldots, a_{m,2}, \ldots, a_{1,n}, \ldots, a_{m,n}]^T$$

Here $a_{i,j}$ represents the (i,j)-th element of matrix A and the superscript T denotes the transpose.

Vectorization expresses the isomorphism $\mathbf{R}^{m\times n} := \mathbf{R}^m \otimes \mathbf{R}^n \cong \mathbf{R}^{mn}$ between these vector spaces (of matrices and vectors) in coordinates.

For example, for the 2×2 matrix $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, the vectorization is $\operatorname{vec}(A) = \begin{bmatrix} a \\ c \\ b \\ d \end{bmatrix}$.

We converted the features into a single column vector, or an array in programming terms

4.4 REGRESSION

Linear regression is an approach for modeling the relationship between a scalar dependent variable and one or more explanatory variables (or independent variables). Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the "lack of fit" in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares loss function as in ridge regression (L2-norm penalty) and lasso (L1-norm penalty). Conversely, the least squares approach can be used to fit

models that are not linear models. Thus, although the terms "least squares" and "linear model" are closely linked, they are not synonymous.

Given a data set $\{y_i,\, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of n statistical units, a linear regression model assumes that the relationship between the dependent variable y_i and the p-vector of regressors x_i is linear. This relationship is modeled through a disturbance term or error variable ε_i — an unobserved random variable that adds noise to the linear relationship between the dependent variable and regressors. Thus the model takes the form:

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where $^{\mathrm{T}}$ denotes the transpose, so that $\boldsymbol{x_i}^{\mathrm{T}}\boldsymbol{\beta}$ is the inner product between vectors $\boldsymbol{x_i}$ and $\boldsymbol{\beta}$.

Often these *n* equations are stacked together and written in vector form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_n^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

The mean absolute scaled error (MASE) is a measure of the accuracy of forecasts . It was proposed in 2005 by statistician Rob J. Hyndman and Professor of Decision

Sciences Anne B. Koehler, who described it as a "generally applicable measurement of forecast accuracy without the problems seen in the other measurements."

$$\mathrm{MASE} = \frac{1}{n}\sum_{t=1}^{n}\left(\frac{|e_t|}{\frac{1}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}\right) = \frac{\sum_{t=1}^{n}|e_t|}{\frac{n}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}$$

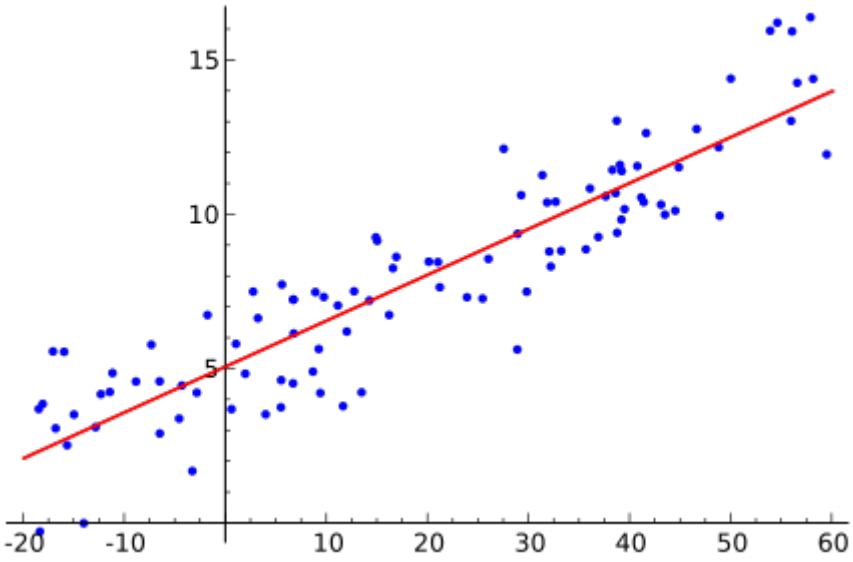


Fig 4.2 Linear regression

We use python as the language of choice due to the multitude of excellent machine learning and statistical libraries are available.

The MAE of always guessing the average band gap is: **1.077 eV**

So any model developed must at least perform better than the always guessing model.

4.4.1 LINEAR RIDGE REGRESSION

Linear Ridge Regression is regression with multiple independent discoveries, it is also variously known as the Tikhonov–Miller method, the Phillips–Twomey method, the constrained linear inversion method, and the method of linear regularization.

Suppose that for a known matrix A and vector b, we wish to find a vector $\mathbf{x}$ such that

$$A\mathbf{x} = \mathbf{b}.$$

The standard approach is ordinary least squares linear regression. However if no x satisfies the equation or more than one x does -- that is the solution is not unique -- the problem is said not to be well posed. In such cases, ordinary least squares estimation leads to an over determined (over-fitted), or more often an underdetermined system of equations. Most real-world phenomena have the effect of low-pass filters in the forward direction where A maps $\mathbf{x}$ to $\mathbf{b}$. Therefore in solving the inverse-problem, the inverse mapping operates as a high-pass filter that has the undesirable tendency of amplifying noise (eigenvalues / singular values are largest in the reverse mapping where they were smallest in the forward mapping). In addition, ordinary least squares implicitly nullifies every element of the reconstructed version of $\mathbf{x}$ that is in the null-space of A, rather than allowing for a model to be used as a prior for $\mathbf{x}$.

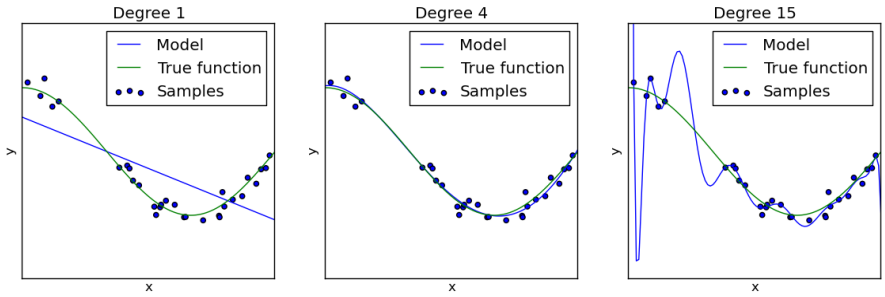
The MAE of the linear ridge using the naive features: **0.583 eV**

This is a huge improvement over the worst case model, but the features used were really naive: the constituent elements and their atomic fractions alone. The next step is to use meaningful physical features to perform regression.

The physical features used are:

- Electronegativity difference of elements in binary compound
- Atomic fractions of elements in binary compound
- Position of elements of binary compounds in periodic table
- Atomic number of elements in binary compound

You will see from the code that we are using an approach called tenfold cross-validation to determine the quality of our model. One of the most fundamental tenets of statistics and machine learning is that one cannot evaluate the quality of a model by examining the error it achieves on the data to which it was fitted. A simple way to understand this concept is to consider the fact that a polynomial regression with enough terms can perfectly fit any training data. However, the question then would be, what happens if we introduce new data and would like such an over fitted model to generalize? A polynomial regression with too many terms will almost certainly fail such a test. In this respect, machine learning models are no different.



Cross-validation is a rigorous approach to avoiding over fitting. In n-fold cross validation (in this case, n=10), we partition our training data into n equally-sized chunks, fit our model to the data in n-1 of those chunks, and use the resulting model to predict the held-out nth chunk. We perform this procedure n times and average the resulting error on the held-out data, reporting a chosen error statistic such as mean squared error or mean absolute error (our choice here). An over fitted model will perform poorly in cross-validation, as it will tend to give very large errors on the held-out data.

When we perform this cross-validation procedure using our linear ridge regression model, the linear model manages to outperform our guess-the-mean approach from above:

The MAE of the linear ridge using the physical features: **0.8 eV**

The MAE of using physical features is actually greater than using naive features. This can be attributed to the poor performance of regression models when the number of

features increased from 1 to 6. A detailed analysis of this result is discussed in the next chapter.

4.4.2 RANDOM FOREST REGRESSION

Random forests is a notion of the general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set.

The algorithm for inducing Breiman's random forest was developed by Leo Breiman[2] and Adele Cutler, and "Random Forests" is their trademark. The method combines Breiman's "bagging" idea and the random selection of features, introduced independently by Ho and Amit and Geman in order to construct a collection of decision trees with controlled variance.

The selection of a random subset of features is an example of the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.
A relationship between random forests and the k-nearest neighbor algorithm (k-NN) was pointed out by Lin and Jeon in 2002. It turns out that both can be viewed as so-called weighted neighborhoods schemes. These are models built from a training set $\{(x_i, y_i)\}_{i=1}^{n}$ that make predictions for new points x' by looking at the "neighborhood" of the point, formalized by a weight function W:

$$\hat{y} = \sum_{i=1}^{n} W(x_i, x')\, y_i.$$

Here, $W(x_i, x')$ is the non-negative weight of the *i*'th training point relative to the new point *x′*. For any particular *x′*, the weights must sum to one. Weight functions are given as follows:

- In *k*-NN, the weights are $W(x_i, x') = \frac{1}{k}$ if x_i is one of the k points closest to x', and zero otherwise.
- In a tree, $W(x_i, x') = \frac{1}{k'}$ if x_i is one of the k' points in the same leaf as x', and zero otherwise.

Since a forest averages the predictions of a set of m trees with individual weight functions W_j, its predictions are

$$\hat{y} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} W_j(x_i, x')\right) y_i.$$

This shows that the whole forest is again a weighted neighborhood scheme, with weights that average those of the individual trees. The neighbors of x' in this interpretation are the points x_i which fall in the same leaf as x' in at least one tree of the forest. In this way, the neighborhood of x' depends in a complex way on the structure of the trees, and thus on the structure of the training set. Lin and Jeon show that the shape of the neighborhood used by a random forest adapts to the local importance of each feature.

The MAE of the nonlinear random forest band gap model using the naive feature set is: **0.366 eV**

The MAE of the nonlinear random forest band gap model using the physical feature set is: **0.265 eV**

4.5 CLASSIFICATION

Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition.

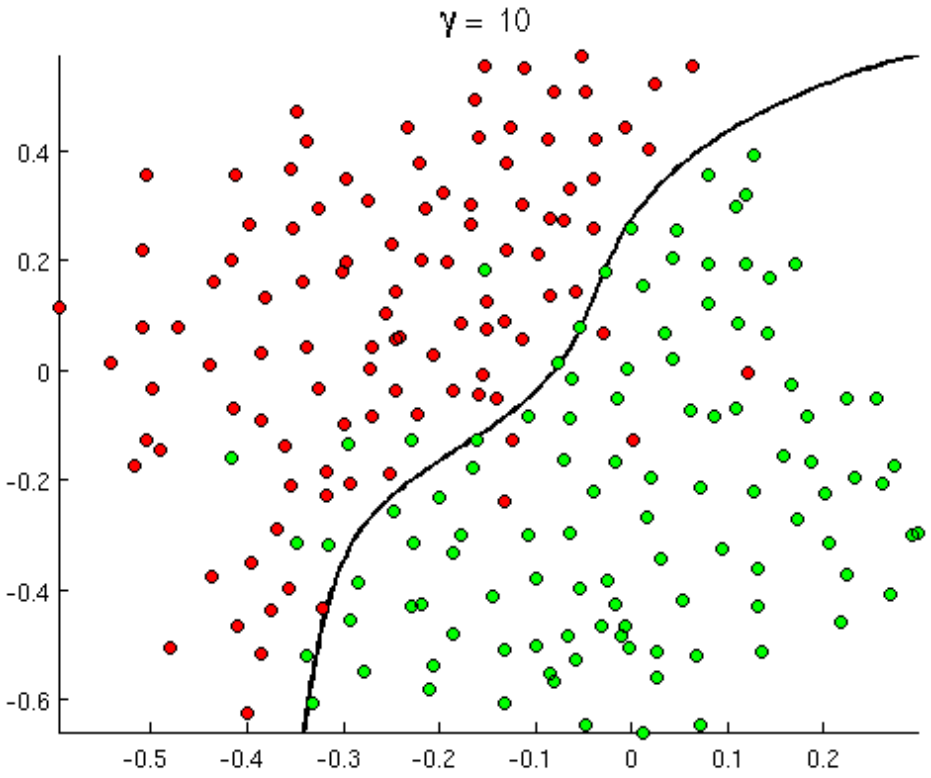


Fig 4.3 Classification in a 2d plane

4.5.1 BUCKETIZATION

Bucketization is the process of defining the several records grouping based on their sensitive values. The apparent sensitive values of the attributes are identified and sorted based on the frequencies in ascending order. After sorting is done, the contiguous sensitive values are grouped into the similar bucket. Only the buckets

contain at least ℓ distinct sensitive values which are kept after bucketing process completion.

Fig 4.4 Bucketized representation

The use of bucketized attributes is to establish the least cost splitting up of a multidimensional data set into B set of buckets (where the value of B is smaller than the number of data points in the dataset). The value of a bucket is calculated by the cost function. Naturally, the cost measure detains the compression of a bucket. A formation of bucketization is measured based on minimizing the average of total number of buckets of a bucket's weight (number of data points) and its perimeter (or sum of its extents along each dimension).

The purpose of integrating bucketization into the machine learning models is to convert the regression into a classification problem. The attribute bucketized is the band gap energy of binary compound system. The apparent sensitive value of each bucket it set to 0.25 eV. 40 buckets, each size of 0.25eV is generated across ranges of band gap energy. This process would consequently reduce the Mean Absolute error (MAE) produced by regression models. This way of classification with bucketization will increase the accuracy of the built model.

The following is the flow classification algorithm for bucketization of data:

Step 1: Extract the dataset from the I/O file.

Step 2: Divide the set of records present in the dataset.

Step 3: Identify the sensitive attributes.

Step 4: Sort the remaining set of records based on the Occurrence of the sensitive attributes.

Step 5: Group the similar set of records with set of buckets.

Step 6: Analyze the set of records in each bucket.

Step 7: Compute the lookup table.

Step 8: Diversity maintains the set of buckets with matching attributes.

Step 9: Combine the set of correlated attributes.

Step 10: Display the set of secured data.

4.5.2 SUPPORT VECTOR MACHINES

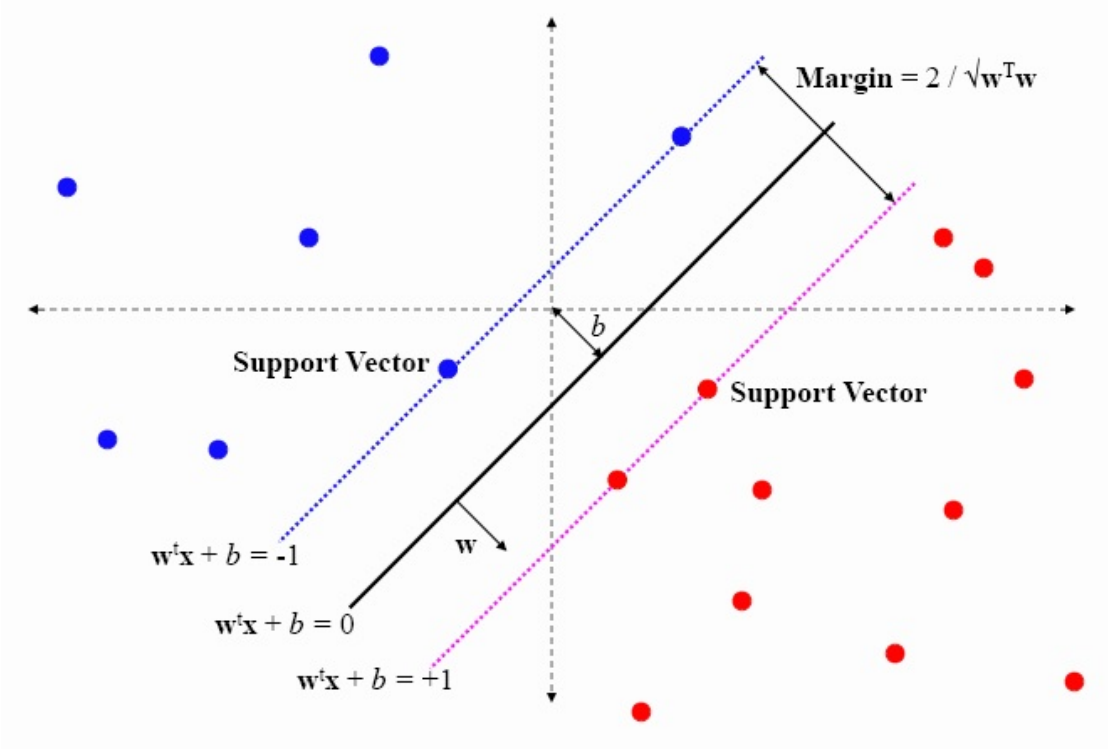


Fig 4.5 Support Vector Machine classification

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a

model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In addition to performing linear classification, SVMs can efficiently perform a nonlinear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a (p-1)-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum margin classifier; or equivalently, the perceptron of optimal stability.

Computing the (soft-margin) SVM classifier amounts to minimizing an expression of the form:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0,1-y_i(w\cdot x_i+b)\right)\right]+\lambda\|w\|^2. \qquad (2)$$

$$\begin{cases} w^T\phi(x_i)+b\geq 1, & \text{if} \quad y_i=+1, \\ w^T\phi(x_i)+b\leq -1, & \text{if} \quad y_i=-1. \end{cases}$$

The Kernel Trick can be applied to map features into higher dimensions. Suppose now that we would like to learn a nonlinear classification rule which corresponds to a linear classification rule for the transformed data points $\varphi(\vec{x}_i)$. Moreover, we are given a kernel function k which satisfies:

$$k(\vec{x}_i, \vec{x}_j) = \varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)$$

where the c_i are obtained by solving the optimization problem:

$$\begin{aligned} \text{maximize } f(c_1 \dots c_n) &= \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i (\varphi(\vec{x}_i) \cdot \varphi(\vec{x}_j)) y_j c_j \\ &= \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i k(\vec{x}_i, \vec{x}_j) y_j c_j \end{aligned}$$

Accuracy using svm.SVC() is : **71.1673699015%**

4.5.3 DECISION TREE CLASSIFIER

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a finite set of values are called classification trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Trees used for regression and trees used for classification have some similarities - but also some differences, such as the procedure used to determine where to split.

Some techniques, often called ensemble methods, construct more than one decision tree:

Bagging decision trees, an early ensemble method, builds multiple decision trees by repeatedly resampling training data with replacement, and voting the trees for a consensus prediction.

A Random Forest classifier uses a number of decision trees, in order to improve the classification rate.

Boosted Trees can be used for regression-type and classification-type problems.

Rotation forest - in which every decision tree is trained by first applying principal component analysis (PCA) on a random subset of the input features.

A special case of a decision tree is a Decision list, which is a one-sided decision tree, so that every internal node has exactly 1 leaf node and exactly 1 internal node as a child (except for the bottommost node, whose only child is a single leaf node). While less expressive, decision lists are arguably easier to understand than general decision trees due to their added sparsity, permit non-greedy learning methods and monotonic constraints to be imposed.

4.5.4 RANDOM FOREST CLASSIFIER

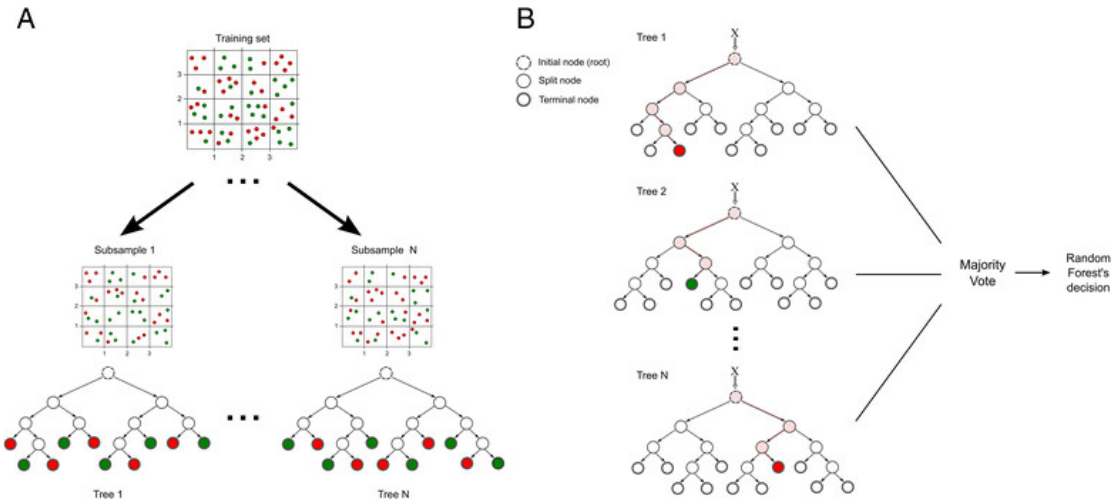


Fig 4.6 Random Forest Classification tree

Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.

Each tree is grown to the largest extent possible. There is no pruning.

The forest error rate depends on two things:

- The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.

- The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Reducing m reduces both the correlation and the strength. Increasing it increases both. Somewhere in between is an "optimal" range of m - usually quite wide. Using the oob error rate (see below) a value of m in the range can quickly be found. This is the only adjustable parameter to which random forests is somewhat sensitive.

Accuracy using Random Forest is: **91.4233576642%**

CHAPTER 5

RESULTS AND DISCUSSION

The results obtained from each model is very interesting and we can learn a lot about the features and their variance from the error rates obtained.

The following table lists the predicted band gap of 10 binary compounds of varying band gap ranges. Though the classifier actually predicts the range within which the band gaps fall into, we have used Sigmoidal and Gaussian functions to estimate a decimal value inside that range.

Binary Compound	linear ridge using naive feature (eV)	linear ridge using physical feature (eV)	random forest regression using physical features(eV)	SVM using physical features (eV)	Random forest classifier using physical features (eV)	Actual Band Gap (eV)
LiH	2.39	2.18	2.62	2.8	2.92	2.981
HF	6.135	5.91	6.36	6.75	6.75	6.718
H3N	3.73	3.52	3.96	4.4	4.45	4.32
Li2Te	1.9	1.73	2.24	2.32	2.4	2.498
BeS	2.62	2.34	2.85	3.42	3.25	3.143
CO2	6.15	5.92	6.38	6.5	6.72	6.633
SiC	1.56	1.34	1.85	2.12	2.2	2.023
AlN	3.58	3.42	3.83	3.96	4.12	4.056
SiO2	5.2	4.96	5.42	5.68	5.72	5.703
Mg3N2	1.34	1.28	1.53	1.6	1.74	1.68

TABLE 4.1 Table comparing the predictions

Linear Ridge regression gives surprisingly good results, given that the features used to rather too naive. This can be attributed to over fitting. The dataset contains a mere 4000 entries of binary compounds, and linear regression without any kernel would cause heavy over fitting of the weights. It would be interesting to see how it would perform when a new binary compound is fed in, containing a element that is not present in the data set.

Linear Ridge regression using physical features does not fare as well the naive features model for many reasons. It appears that five degrees of freedom (four coefficients plus a constant) are too few to reasonably model band gaps with a linear model. The naive model is also prone to over fitting and hence the error is not really representative of the actual performance of the model.

The Random Forest being an ensemble method performs better than the regression models. We observe that a random forest-based approach outperforms linear ridge regression with both feature sets, and the physically-motivated four-feature vector outperforms the naive 100-component composition vector. The random forest trained on the physical features is actually decent at estimating band gaps of materials. It turns out that, while a single decision tree is often a poor classifier, a collection of many decision trees trained on different subsets of data can be very powerful for modeling data. Random forests have a number of tuning parameters (as with most machine learning algorithms, unfortunately), but here we highlight only the number of trees in the forest; we will chose 50 for computational expediency. Our code will thus constructed 50 independent decision trees and average the band gap predictions from each of them.
However, predicting the band gap to 2 decimal points accuracy is a complicated task. Multiple researches [4] suggest and point out that band gaps when calculated at different temperatures, pressures, methods differ by up to +- 1eV. This let us to convert the regression model into a classification model, and predicting band gaps in ranges of 0.25 eV.

SVM classifier using the same physical features presented us an accuracy of 72%. This is equivalent to guessing a 0.25 window range of the correct band gap 3 out of 4 times. SVM though one of the most reliable classifiers, does not perform as well as expected due to poor choice of kernel functions. Why kernels, as opposed to feature vectors? One big reason is that in many cases, computing the kernel is easy, but computing the feature vector corresponding to the kernel is really really hard. The feature vector for even simple kernels can blow up in size, and for kernels like the RBF kernel the corresponding feature vector is infinite dimensional.

Finally we use a random forest classifier, and obtain an accuracy of about 92%. Random forest being an ensemble method, performs really well for classification problems and in this case as there are only about 40 buckets of 0.25 eV each, we get surprisingly accurate estimates.

CHAPTER 6

CONCLUSIONS

Ever since the advent of data science and efficient machine learning tools, diverse fields have taken advantage and have been using them to solve domain specific problems. There are a lot of unsolved problems in material science and this provides us an opportunity to experiment with unique ways of approaching the problems. This is one such attempt. This project does not target replacing experimentation to calculate band gap with a simple mathematical model, instead explores the possibility of the use of machine learning to assist and validate experimentation.

From here, the two main levers we have to further enhance our model are: adding more training data and further feature engineering. Probably our simple four-feature approach is not yet optimized. Again, this is where our background as a materials scientist is essential.

A 92% accuracy of 0.25 eV ranges is a huge advancement compared to ongoing research in this field [5] which present a 0.5 eV error with multiple assumptions. We should also put our results here in context. Machine learning is indeed a potent tool for analyzing materials data; when used properly, it can produce powerful, original insights. But it is just that: another tool in the materials scientist's toolbox. Machine learning (or any other computational technique) is not a substitute for scientific judgment or common sense.

However, with a combination of robust training data and insightful features, we can build very fast and very effective models of materials behavior, without a supercomputer and without a long and expensive scientific procedures.

REFERENCES

[1] http://research.ijcaonline.org/volume81/number12/pxc3892245.pdf

[2] Kleinberg, Eugene (2000). "On the Algorithmic Implementation of Stochastic Discrimination". IEEE Transactions on PAMI 22 (5)

[3] Using Random Forest to Learn Imbalanced Data, Chao Chen, Andy Liaw & Leo Breiman, July 2004

[4] "Prediction model of band-gap for AX binary compounds by combination of density functional theory calculations and machine learning techniques"
Joohwi Lee1, Atsuto Seko, Kazuki Shitara, Isao Tanaka

[5] "Accelerating materials property predictions using machine learning"
Ghanshyam Pilania, Chenchen Wang, Xun Jiang, Sanguthevar Rajasekaran & Ramamurthy Ramprasad

[6] https://materialsproject.org/

[7] http://pymatgen.org/

[8] Theodoridis, Sergios; and Koutroumbas, Konstantinos; "Pattern Recognition", 4th Edition, Academic Press, 2009, ISBN 978-1-59749-272-0

[9] Hsu, Chih-Wei; Chang, Chih-Chung; and Lin, Chih-Jen (2003). A Practical Guide to Support Vector Classification (PDF) (Technical report). Department of Computer Science and Information Engineering, National Taiwan University.

[10] Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning 20

[11] www.numpy.org/

APPENDIX: SOURCE CODE

```
from pymatgen import *
from numpy import zeros, mean
from sklearn import *
import matplotlib.pyplot as plt
import bucket
from sklearn.metrics import accuracy_score

trainFile = open("bandgapDFT.csv", "r").readlines()



def naiveVectorize(composition):
    vector = zeros((MAX_Z))
    for element in composition:
        fraction =
composition.get_atomic_fraction(element)
        vector[element.Z - 1] = fraction
    return(vector)

materials = []
bandgaps = []
naiveFeatures = []

MAX_Z = 100

for line in trainFile:
```

```
    split = str.split(line, ',')
    if(float(split[1]) == 0):
        x = 1
        continue
    material = Composition(split[0])
    materials.append(material)
    naiveFeatures.append(naiveVectorize(material))
    bandgaps.append(float(split[1]))

baselineError = mean(abs(mean(bandgaps) - bandgaps))
print("The MAE of always guessing the average band gap
is: " +
      str(round(baselineError, 3)) + " eV")


linear = linear_model.Ridge(alpha=0.5)

cv = cross_validation.ShuffleSplit(len(bandgaps),
     n_iter=10, test_size=0.1, random_state=0)

scores = cross_validation.cross_val_score(
    linear,
    naiveFeatures,
    bandgaps,
    cv=cv,
    scoring='mean_absolute_error')
```

```
print("The MAE of the linear ridge using the naive
features: " +
      str(round(abs(mean(scores)), 3)) + " eV")


physicalFeatures = []


atmno = []
plotter = {}
plotter2 = {}
plotter3 = {}
it = 0


for material in materials:
    theseFeatures = []
    fraction = []
    atomicNo = []
    eneg = []
    group = []


    for element in material:


fraction.append(material.get_atomic_fraction(element))
        atomicNo.append(float(element.Z))
        eneg.append(element.X)
        group.append(float(element.group))
```

```
    mustReverse = False

    if fraction[1] > fraction[0]:
        mustReverse = True

    for features in [fraction, atomicNo, eneg, group]:
        if mustReverse:
            features.reverse()
    theseFeatures.append(fraction[0] / fraction[1])
    theseFeatures.append(eneg[0] - eneg[1])
    theseFeatures.append(group[0])
    theseFeatures.append(group[1])
    theseFeatures.append(atomicNo[0] + atomicNo[1])
    physicalFeatures.append(theseFeatures)
    ZZ = 0
    for z in atomicNo:
        ZZ += z
    atmno.append(ZZ)
    plotter[bandgaps[it]] = ZZ
    plotter2[bandgaps[it]] = eneg[0] - eneg[1]
    plotter3[bandgaps[it]] = fraction[0] / fraction[1]
    it += 1

linear = linear_model.Ridge(alpha=0.5)

f, (ax1, ax2, ax3) = plt.subplots(3)
```

```
c = sorted(plotter3.iteritems(), key=lambda (x, y):
float(x))

key1 = []

val1 = []

for j in c:

    key1.append(j[0])

    val1.append(j[1])

ax1.plot(val1, key1, 'g.')

ax1.set_title('Atomic fraction, Electro negativity
difference, ' +

              ' Molecular Weight')


# ax1.xlabel('Atomic Fraction')

# ax1.ylabel('Band Gap')



d = sorted(plotter2.iteritems(), key=lambda (x, y):
float(x))

key2 = []

val2 = []

for k in d:

    key2.append(k[0])

    val2.append(k[1])

ax2.plot(val2, key2, 'b.')
```

```
# ax2.xlabel('Electro negativity difference')
# ax2.ylabel('Band Gap')


b = sorted(plotter.iteritems(), key=lambda (x, y):
float(x))

key = []
val = []
for i in b:
    key.append(i[0])
    val.append(i[1])

# ax3.xlabel('Molecular weight')
# ax3.ylabel('Band Gap')
ax3.plot(val, key, "r.")

f.text(0.06, 0.5, 'Band Gap', ha='center',
       va='center', rotation='vertical')

plt.show()

cv = cross_validation.ShuffleSplit(len(bandgaps),
       n_iter=10, test_size=0.1, random_state=0)

scores = cross_validation.cross_val_score(
```

```
    linear,
    physicalFeatures,
    bandgaps,
    cv=cv,
    scoring='mean_absolute_error')

print("The MAE of the linear ridge using the
physicalFeatures: " +
      str(round(abs(mean(scores)), 3)) + " eV")

rfr = ensemble.RandomForestRegressor(n_estimators=10)
scores = cross_validation.cross_val_score(
    rfr,
    physicalFeatures,
    bandgaps,
    cv=cv,
    scoring='mean_absolute_error')

print("The MAE of random forest using physicalFeatures
feature set is: " +
      str(round(abs(mean(scores)), 3)) + " eV")


# Using SVM to classify
# training set size:3000 test set size:1096
```

```
from sklearn.ensemble import RandomForestClassifier
import sys

# print(sys.argv[1])
physicalFeatures1 = []
binary_compound = Composition(str(sys.argv[1]))
print(binary_compound)
binary_compounds = []
binary_compounds.append(binary_compound)
for i in binary_compounds:
    theseFeatures1 = []
    fraction1 = []
    atomicNo1 = []
    eneg1 = []
    group1 = []

    for j in i:
        fraction1.append(i.get_atomic_fraction(j))
        atomicNo1.append(float(j.Z))
        eneg1.append(j.X)
        group1.append(float(j.group))
    mustReverse = False

    if fraction1[1] > fraction1[0]:
        mustReverse = True
```

```
    for features1 in [fraction1, atomicNo1, eneg1,
group1]:
        if mustReverse:
            features1.reverse()
    theseFeatures1.append(fraction1[0] / fraction1[1])
    theseFeatures1.append(eneg1[0] - eneg1[1])
    theseFeatures1.append(group1[0])
    theseFeatures1.append(group1[1])
    theseFeatures1.append(atomicNo1[0] + atomicNo1[1])
    physicalFeatures1.append(theseFeatures1)



train_X = physicalFeatures[0:3000]
convertedBandgap = bucket.create_bucket(bandgaps)
train_Y = convertedBandgap[0:3000]
test_X = physicalFeatures1
clf = RandomForestClassifier(n_estimators=50)
clf.fit(train_X, train_Y)
predict = clf.predict(test_X)

#print predict
#print type(predict)

predicted_bandgap=bucket.bucket_to_bandgap_conversion(pre
dict[0])
```

```
print("The Predicted Band Gap for the given binary
compound is "+str(predicted_bandgap))
```

bucket.py

```
bucket_list = {0.25: 1, 0.5: 2, 0.75: 3, 1: 4, 1.25: 5,
1.5: 6, 1.75: 7,
               2: 8, 2.25: 9, 2.5: 10, 2.75: 11, 3: 12,
3.25: 13, 3.5: 14,
               3.75: 15, 4: 16, 4.25: 17, 4.5: 18, 4.75:
19, 5: 20, 5.25: 21, 5.5: 22, 5.75: 23, 6: 24, 6.25: 25,
6.5: 26, 6.75: 27, 7: 28, 7.25: 29, 7.5: 30, 7.75: 31, 8:
32, 8.25: 33, 8.5: 34, 8.75: 35, 9: 36, 9.25: 37, 9.5:
38, 9.75: 39, 10: 40}

bucket_list_new = sorted(bucket_list.iteritems(),
key=lambda (x, y): float(x))
new_bucket = []


def create_bucket(a):
    new_bucket = []
    for i in a:
        for j in bucket_list_new:
            if(j[0] >= i):
                new_bucket.append(j[1])
                break
    return new_bucket

def bucket_to_bandgap_conversion(b):

#    print(b)
    for key,value in bucket_list.iteritems():
        if(value==b):
             return key
```