Computational Materials Science with Machine Learning

Abstract

In materials and process design, knowledge of the physical properties of materials as a function of temperature, composition, applied external stresses, etc. is a crucial factor. Such properties may be unknown and difficult to measure or estimate for new systems using numerical simulations such as molecular dynamics. Engineers rely on machine learning to anticipate the properties of new systems using existing data. Currently, numerous techniques are employed for such purposes. In report, I compared different machine learning algorithm for prediction of two different physical properties- electric conductivity and molar volume of a mixture of ten different metal oxides with the data extracted from literature.

1 Introduction

Various machine learning techniques have been employed over the years to apply sets of known data associated with certain properties to sets of unknown data in order to predict these properties. There exists a problem for which a particular method functions well and another problem for which the same method fails miserably. This paper seeks to compare various machine learning techniques for predicting the properties of various data types. Literature-gathered material science data on molten oxides systems is our primary emphasis. Material science engineers must understand the physical properties of such systems in order to design new materials and enhance existing processes.In this work, I perform a comparative study of the predicting power of four of the most popular and emerging machine learning techniques in prediction of large scale data set. The different techniques are tested on data set of electrical conductivity and molar volume containing non-smooth and smooth data respectively. We consider data on molar volume to be smooth because the theory tells us that it should vary almost linearly and also because the experimental datasets are in good agreement at equal composition and temperature. For the electrical conductivity, the data is very scattered and that is why we consider it to be non-smooth.

2 About material dataset

I have access to two databases of experimental points culled from the literature for this project. The database utilised for predicting molar volume has 2700 data points (n = 2700) with various compositions in mole percent on 10 dimensions (D = 10), temperature in Kelvin, and a corresponding molar volume value in cubic centimetres per mole. The electrical conductivity database contains approximately 9300 data points with compositions in mole percent across 10 dimensions, temperature (T) in Kelvin (D = 10), and an electrical conductivity (EC) value in Siemens per metre. Measuring the molar volume of liquid oxides at elevated temperatures can result in a significant amount of uncertainty and discordance between existing data sources. In spite of this, we consider the molar volume to be smooth because the majority of the data set has little variance. Electrical conductivity is also measured at elevated temperatures, resulting in a diminished degree of certainty. We consider EC to be non-smooth due to this, its intricate dependence on compositions, and its adherence to the Arrhenius laws.

3 Machine Learning Algorithm used

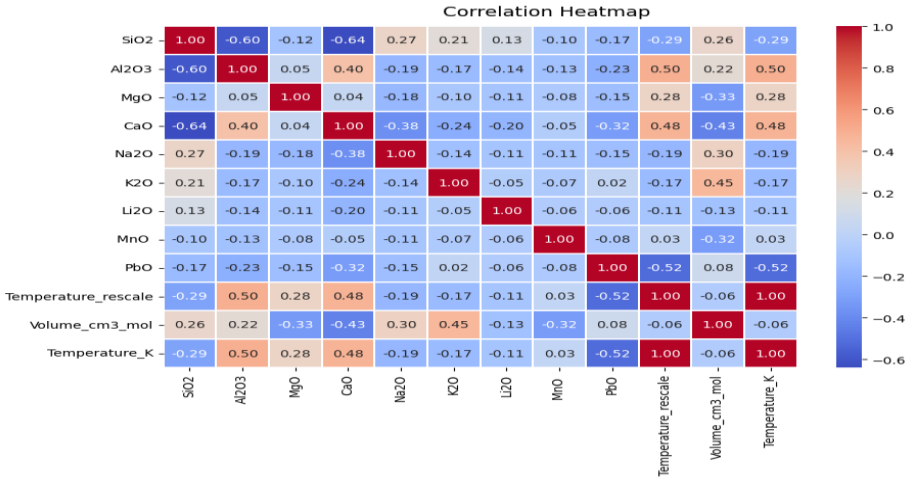
Prediction of properties is done in this paper based on five most popular machine learning algortthms used. Before diving deep into algorithms, let us understand basics of few glamorous machine learning algorithms used in computational materials science to know their feasibility working with different models-

- **Linear Regression**- a relationship is established between independent and dependent variables by fitting them to a line.

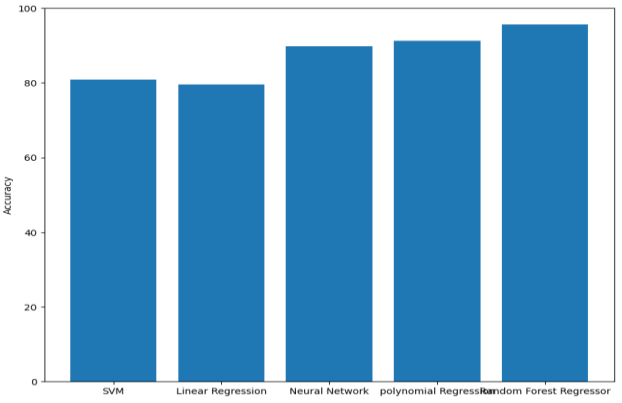
- **Logistic Regression-** to estimate discrete values (usually binary values like 0/1) from a set of independent variables
- **Decision Tree-** This algorithm divides the population into two or more homogeneous sets based on the most significant attributes/ independent variables.
- **SVM**- classification algorithm in which you plot raw data as points in an n-dimensional space. The value of each feature is then tied to a particular coordinate, making it easy to classify the data.
- **Naïve Bayes Algorithm-** classify the features considering their properties independent to each other and calculate the probability of occurrence of particular event
- **Random Forest-** Take a collection of decision tree into consideration and classify the new object based on attributes of votes of every individual tree is classified
- **Neural Network**- A neural network is made up of a series of layers, each containing a number of interconnected nodes (or "neurons").
- **Polynomial Regression**- instead of fitting a linear line to the data, as in linear regression, we fit a polynomial curve to the data. Polynomial regression is used when relationship between dependent and independent variable in non-linear.

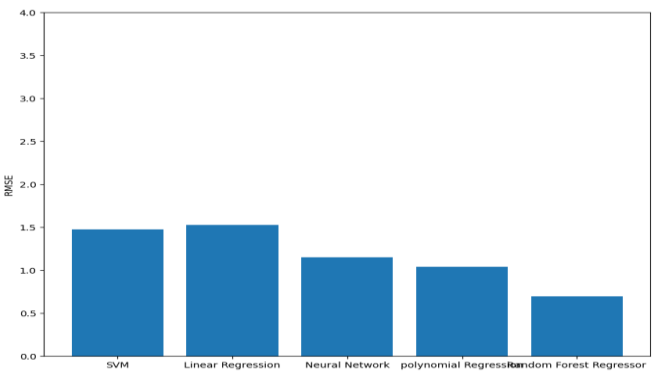
4 Working with molar volume database

Figure below states that there do not exist direct relationship between any independent variable with the dependent variable i.e the molar volume. So while working with model, it is necessary that each oxide percentage is taken care of.



In total five different techniques were used in prediction of molar volume dataset. The techniques used were Linear regression, Polynomial Regression, Support-vector Regressor, Neural network and random forest regressor. All five techniques worked well with the dataset with accuracy at least 70% with each and rmse values less than 2. The below figure indicates rmse and accuracy% with each of the model.





4a Applying Linear Regression

Technique of linear regression on multiple variable first comes to mind where there is not direct relation between independent and dependent variable is observed. To check that which distribution of training and test data split gives best accuracy, I applied regression on every 10% increase split it is found that 50% of given data when used for training the model, it gives best accuracy.

Maximum accuracy at **50% : 50%** split gives us clear indication that the target values are random because with lower percentage of dataset at training the model, we get highest accuracy. Usually, best accuracy achieves at around 70:30 split as most of target values gets trained and with part left in test, prediction is better to make. But here, we need more of testing data, indicating that splitting is not leading to accurate results. But still, on observation we see that the accuracy is not must changing with change in percentage splits in train and test. This implies that the data points are all in near values and no such redenceny is occurring throughout dataset.

Observe that maximum rmse is with 10% training data set. This rmse value is at testing data. So higher the testing data, Lower rmse we get initially.This implies that model is **underfitted** because it tend to prefer more data in test set. But still error is not exceeding 1.625%. This means LR making good prediction at any train-test split ratio.

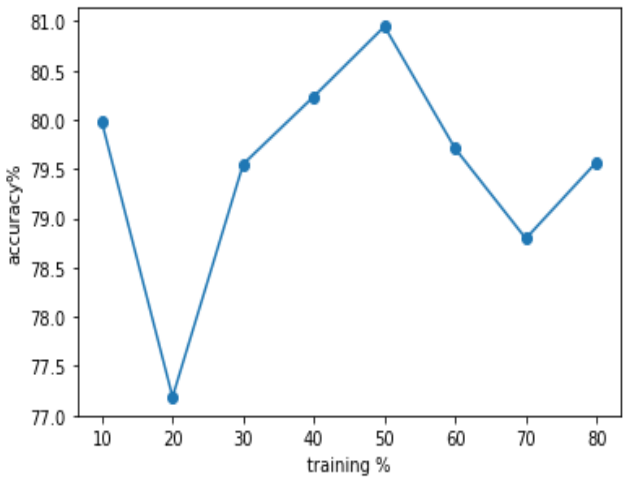


Figure 3 Accuracy% at different %of training set

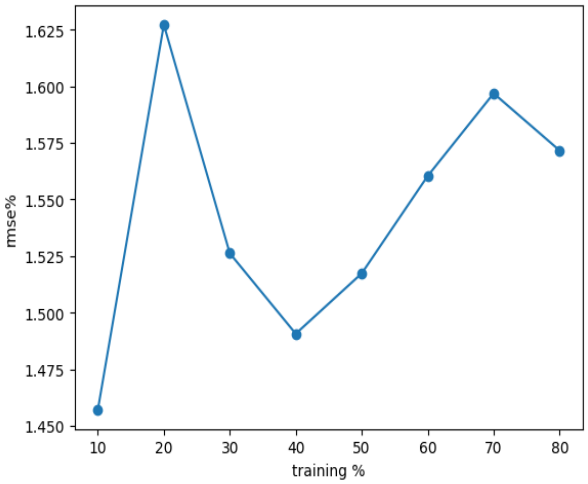


Figure 4: rmse at different %training set for linear regression

Below is the code snippet of loop used for accuracy calculation at different training%

```
#This code snippet contains analysis of accuracy achieved using linear regression with different percentage of training-test split
t=[]
for i in range (1,9):
 x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=(i*10)/100,random_state=0)
 reg=LinearRegression()
 reg.fit(x_train,y_train)
 y_pred=reg.predict(x_test)
 acc=r2_score(y_test,y_pred)*100
 t.append(acc)
t
plt.scatter(range(10,90,10),t)
plt.plot(range(10,90,10),t)
plt.xlabel("training %")
plt.ylabel("accuracy%")
plt.show()
```

From the above algorithm of linear regression used, it is clear that model given is smooth as with the most basic regression technique ie, Linear regression , we are getting approximately 80% accuracy. The major concern faced

here is that with optimum train:test split, we are getting lower accuracy% as compared with different percentage splits.

4b Applying Support Vector Regressor

Support vector machine (svm) is one the known algorithm used for both classification as well as regression problems. Concept which svm uses is that it creates a hyperplane or decision boundary which segregates points in n-dimensions And the new variable depending upon its match can easily be classified to a particular dimension. Form of svm used while regressor is called support vector regressor.

Again, while working with svr, we plotted accuracy achieved with every 10% increase in traing percentage of dataset. Accuracy is reaching to maximum strength of ~81%. As 60:40 or 70:30 split is giving maximum accuracy, it is evident that svr is splitting data in better way as without underfitting or overfitting dataset, we are getting good accuracy. So support vector regressor is working perfectly with model, except for the only issue that accuracy of prediction of molar volume is not very good if not bad.

Comparing with LR- Svr is getting almost equal maximum accuracy to linear regression but with more percentage of data trained. This is because **svm handles outliers** in better way than LR and there are too many outliers with given data set.

Root Mean square error with SVR- Looking at accuracy plot, it can be estimated that least rmse would be at 60:40 or 70:30 Split. By visualizing plot of training% vs rmse, we find that lowest rmse is at 60:40, which is 1.46 ie is a very low value. Another evident to prove that svm is performing better than LR because for a good model to fit, rmse should decrease till train-test ratio is 60:40 or 70:30 which is evident here. For a decent model, plot of rmse with training% should appear a horizontal mirror image of accuracy Vs training set plot, which is almost achieved here.

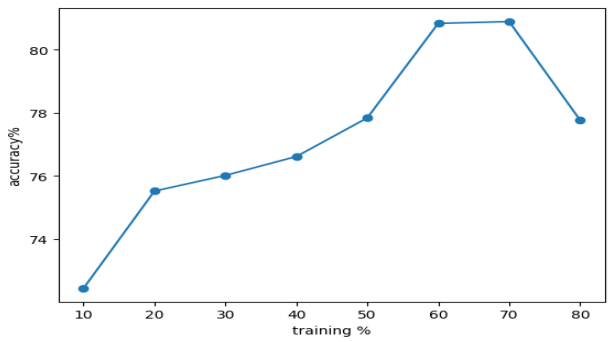


Figure 4 accuracy% vs training dataset% plot

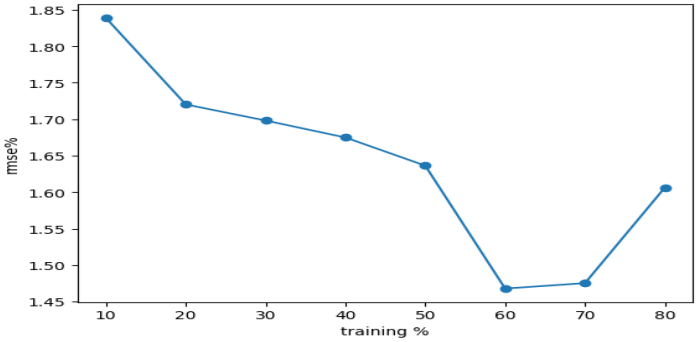


Figure 5: RMSEvs training % of dataset plot for SVR algo

Below provides the code snippets for support vector regressor algorithm which I applied for accuracy calculation at different training% of dataset

```
#rmse values at different train-test split percentage
t=[]
for i in range (1,9):
 x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=(i*10)/100,random_state=0)
 reg=SVR(kernel='rbf')
 reg.fit(x_train,y_train)
 y_pred=reg.predict(x_test)
 rmse=np.sqrt(mean_squared_error(y_test,y_pred))
 t.append(rmse)
plt.scatter(range(10,90,10),t)
plt.plot(range(10,90,10),t)
plt.xlabel("training %")
plt.ylabel("rmse%")
plt.show()
```

4c Neural Network

A neural network is made up of a series of layers, each containing a number of interconnected nodes (or "neurons"). The first layer takes in input data, such as an image or a text sequence, and subsequent layers use these inputs to make predictions or classify the data. During training, the network adjusts the weights and biases of each neuron to minimize the error between its predictions and the actual output. This process is repeated over multiple epochs until the network's accuracy on a validation set reaches an acceptable level**.**

As the dataset seem to be too complex, best analysis can be carried out using neural network with more than 1 hidden layer. In order to prevent overfitting, let's carry stimulation with only 2 hidden layers.

There few important terms to be kept in knowledge while working with neural networks-

- **Epochs**- defines how many times training data is passed through the neural network. Generally complexed dataset should work with higher epochs so that ambiguity could be handled but too high epochs leads to overfitted data and also code execution time increases with increase in epochs. Here, epochs used is 50
- **Optimizer-** optimizers consist of algorithm used to update weights and biases so as to reduce loss in test data.Here we are using **Adam** optimizer which updates weights considering both first order grad descent and second order gradient descent together.

(i)Why use Neural Network-

- **Non-linearity**-From results achieved by correlation heatmap it is confirmed that dataset does not follow any trends towards independent data set
- **Number of variables-**There are 9 independent variable and there no such variable which can be dropped in our analysis, so to carry prediction with this large independent columns neural network suits best.
- **Data availability-**Neural network works best on large dataset and the given dataset contains 2700 data points which suites well for neural network.

(ii)When is neural network disadvantageous

Neural network proves to be time consuming algorithm as there are several iteration to the training dataset because the results are backpropogated from hidden layers for more learning before moving to output layer. Also number of epochs increases time complexity**.**

(iii)Using neural network on molar volume dataset

As the dataset was appearing more complex compared to others, it is necessary to have more than 1 hidden Layer in our network but having many hidden layers can lead to overfitting of dataset.

Plot of accuracy % vs training dataset% shows increase in accuracy% continuously till optimum split ratio of 70:30 is achieved. This is because It is evident that with increase in training size, network is having more data points as an example to learn from and overfitting also reduce with we gives required amount of epochs.

Which is best training set- According to theory, point at which slope of train vs accuracy curve tends to decrease, the best training percentage is that point because it means that the maximum rate of accuracy increase is achieved and rest of data set can be used as test. So best split is 70:30.

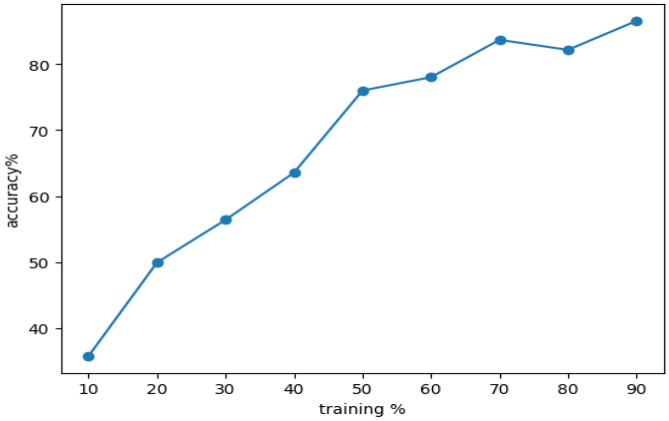


Figure 6 accuracy% vs training% dataset

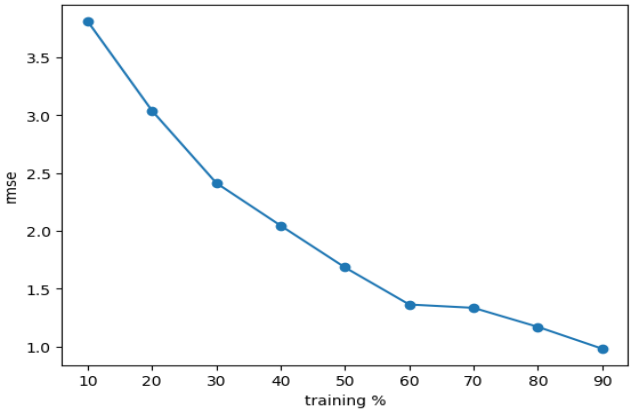


Figure7 rmse vs training dataset% for neural network

The below is the accuracy estimating code using neural networks

```
for i in range (1,10):
 x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=i/10,random_state=0)
 sc = StandardScaler()
 x_train = sc.fit_transform(x_train)
 x_test = sc.transform(x_test)
 model = keras.Sequential([
    keras.layers.Dense(32, activation='relu', input_shape=(x_train.shape[1],)),
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dense(1)])
 model.compile(loss='mean_squared_error', optimizer='adam')
 model.fit(x_train, y_train, epochs=50,batch_size=32)
 predictions = model.predict(x_test)
 acc=r2_score(predictions,y_test)
 acc
 t.append(acc*100)
plt.scatter(range(10,100,10),t)
plt.plot(range(10,100,10),t)
plt.xlabel("training %")
plt.ylabel("accuracy%")
plt.show()
```

4d Polynomial Regression

Where to use polynomial Regression-Polynomial regression is often used when the relationship between the independent and dependent variables is non-linear and cannot be captured by a simple linear model. It can be used in a wide range of applications, such as predicting sales based on advertising expenses, estimating the relationship between temperature and energy consumption, or modeling the relationship between vehicle speed and fuel consumption.

From the heatmap, it can be visualized that there is existing no linear relationship between independent variables and also neural network is failing to give 90%+ accuracy with optimum train test split.

Here we increase dimensions of input variable to n degrees where general equation for poly regression is such that- $y = b_0 + b_1x + b_2x^2 + ... + bn*x^n$

Using with Polynomial Regression on this model

Using degree=3 for our problem, polynomial regression table increases number of feature distinction from 9 to 285.This provides better classification of dataset and hence would increase accuracy. With degree=3 we get accuracy as high as 91% while with degree=2, accuracy achieved is 85.8% !! As time execution of

dataset produced by 3 degree polynomial is not too high, prediction with polynomial regression with degree=3 can be made, thus giving a decent accuracy of 91%.

Below is the code snippet of polynomial regression on molar volume dataset

```
# Fit a linear regression model to the transformed data
model = LinearRegression()
model.fit(X_poly, y)

# Predict the response for the input data
y_pred = model.predict(X_poly)
print(y_pred.shape,X_poly.shape,y.shape)
# Calculate the R-squared score
r2 = r2_score(y, y_pred)
# Print the coefficients and the R-squared score
print('Coefficients: ', model.coef_)
print('R-squared score: ', r2)
```

4e Random Forest Regressor

Random Forest Regressor is a type of ensemble learning algorithm used for regression tasks. It is an extension of the decision tree algorithm, where a forest of decision trees is built and the final output is the average of the outputs of individual trees.

(i)How random forest works-

In a Random Forest Regressor, each decision tree is built using a random subset of the training data and a random subset of features. This introduces randomness and reduces overfitting, resulting in better generalization performance. During the prediction phase, each tree in the forest makes a prediction, and the final output is the average of all predictions made by the trees. Random Forest Regressor is a robust algorithm that can handle large datasets with high-dimensional features, handle missing values and maintain good performance even in noisy data.

The prediction accuracy with random forest depends on the number of trees we choose, which further decides the splitting of features into different variables. Trees should be selected so that the time limit of the program and accuracy both should be optimized. A higher number of trees can improve accuracy, but code execution time would be significant. In our prediction code, we used number of trees estimation=50

(ii)Results with Random Forest Regressor

At last, best results obtained with this ML algorithm

Maximum accuracy obtained with RFR is as high as 96%.It is to be noted that rate of increase in accuracy is maximum as train:test ratio increased to 60:40. This implies that model is getting Optimum dataset in train section and accuracy keeps on increasing as.we move to 70:30 split.

(iii)Why Random Forest Regressor worked best

We saw while analyzing our dataset that random forest gave best results in term of accuracy of prediction.

The reasons can be as such-

- **Nonlinear relationships:** Random forest regressors are good at capturing nonlinear relationships between the input variables and the target variable. We obtained during visualization that there is no such linear relationship occurring between dependent and independent variables.
- **Ensemble learning:** Random forest regressors are an ensemble of decision trees, which means they combine the predictions of multiple decision trees to make a final prediction.
- **Robustness to outliers**: Random forest regressors are less sensitive to outliers compared to other regression models.This is because the predictions are based on the average of many decision trees, which reduces the impact of individual outliers. There were large number of outliers as presented in graph above due to which neural network also failed to make correct predictions. Random forest proves itself useful here as it help get rid of outliers.
- **Continual learning**: Random forest can be trained incrementally over time, allowing them to adapt to new data and changes in the underlying relationships between the input variables and the target variable.

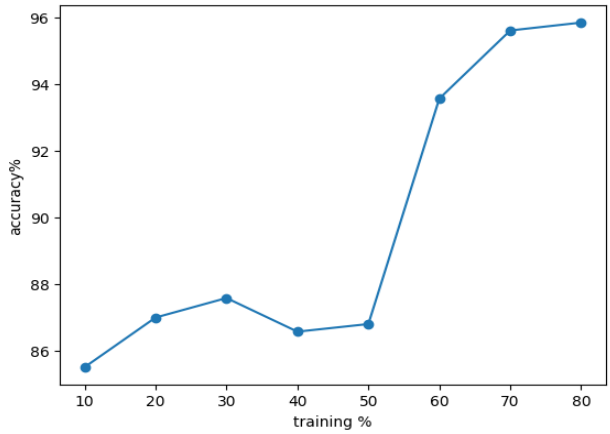


Figure 8 acuracy% vs training dataset% for RFR

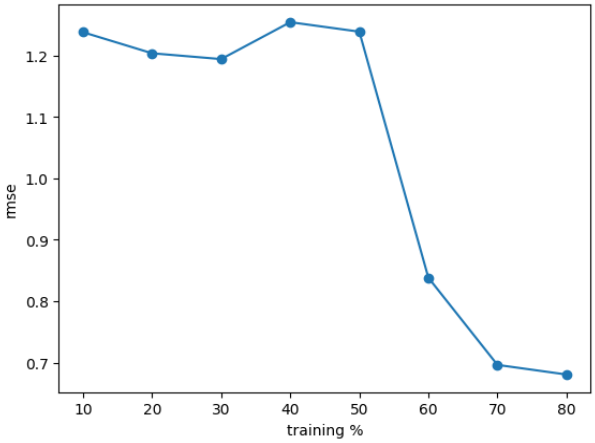


Fig9: rmse vs different training% dataset

Below is the code snippets used for calculating accuracy by using random forest regressor-

```
#code using random forest regrresor
t=[]
for i in range (1,9):
 x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=(i*10)/100,random_state=0)
 reg= RandomForestRegressor(n_estimators=50, random_state=0)
 reg.fit(x_train,y_train)
 y_pred=reg.predict(x_test)
 acc=r2_score(y_pred,y_test)*100
 t.append(acc)
plt.scatter(range(10,90,10),t)
plt.plot(range(10,90,10),t)
plt.xlabel("training %")
plt.ylabel("accuracy%")
plt.show()
```

5 Working with electric conductivity dataset

We test the linear, polynomial regression, neural network, support vector regressor and random forest regressor techniques with actual electrical conductivity values conductivity as well as ln(cond) and ln(T . cond). As mentioned earlier, electrical conductivity here refers to the ionic conductivity. Table 4 shows that using log(T* cond) gives the best predictions, therefore we compare the RMSE making predictions on this value. This can be explained because in general, the electrical conductivity (j) temperature dependence obeys Arrhenius laws, that is: $\kappa=\alpha+\beta/T$ where b is the activation energy and a is a value of electrical conductivity at a reference temperature. However, for silicate systems, there is a deviation from this law. Consequently, we decided to test all three cases mentioned above to evaluate how the prior knowledge of the problem influences predictions quality.

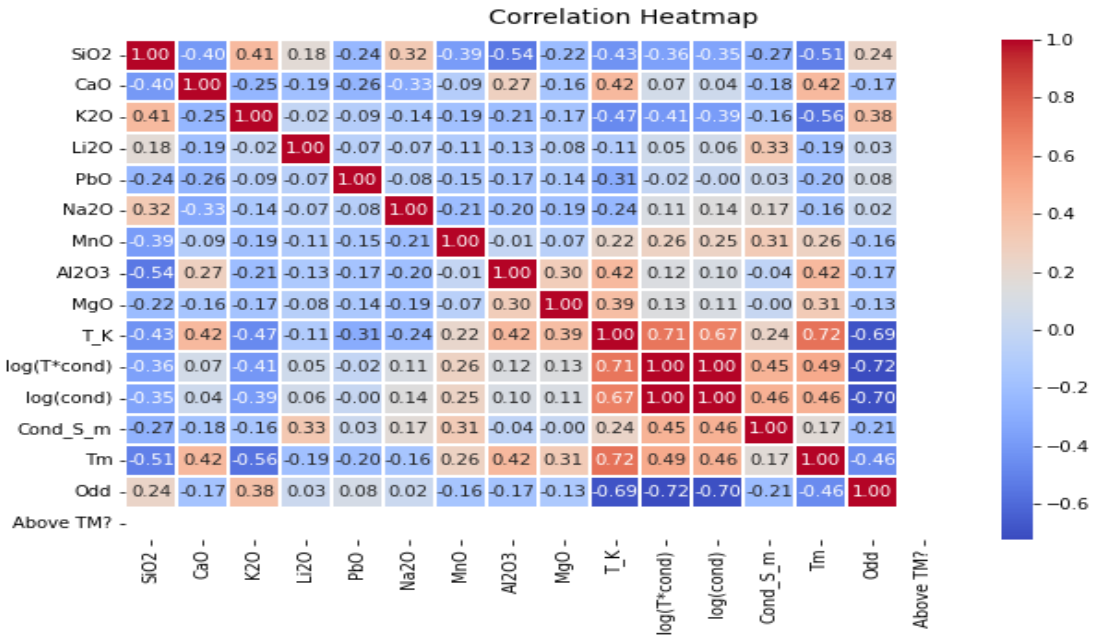
Table 1 Showing accuracy with predicting respective algorithm on different independent variables

Algorithm	Log(T*cond)	Log(cond)	conductivity
Linear Regression	78.58	76.4	58.07
Polynomial Regression	97.67	97.30	86.77
Neural network	97.72	97.7	82.55
Support vector Regressor	84.29	82.01	-2
Random Forest Regressor	98.06	97.78	94.85

From table, it is clear that while predicting conductivity, we need to take temperature of sample as input variable because of direct relationship of conductivity and temperature. Analysing table we see that accuracy is not much differing between log(T*cond) and log(cond). In fact, with polynomial regressor we are having more accuracy while predicting log(cond). This is because the dataset contains SiO2 as a component in mixture. Silicate system is deviated form Arhenius law and so presence of temperature is not affecting much. If we analyse dataset without SiO2, we find that accuracy in predicting log(cond) goes **down drastically.**

Analysing dataset

The below heatmap indicates the relationship between each variables present in dataset.



See that except SiO2, log(T*cond) is having a positive relation with each oxides. But still with each oxide there is no direct relationship with the result.

The below regression plot has predicted value on y axis and actual value on x axis of log(T*cond) with scatter points. The line is the regression line of model. Except for few outliers, plot revolve around regression line. This shows that linear regression is working well and for dealing with outliers other algorithms can be used.

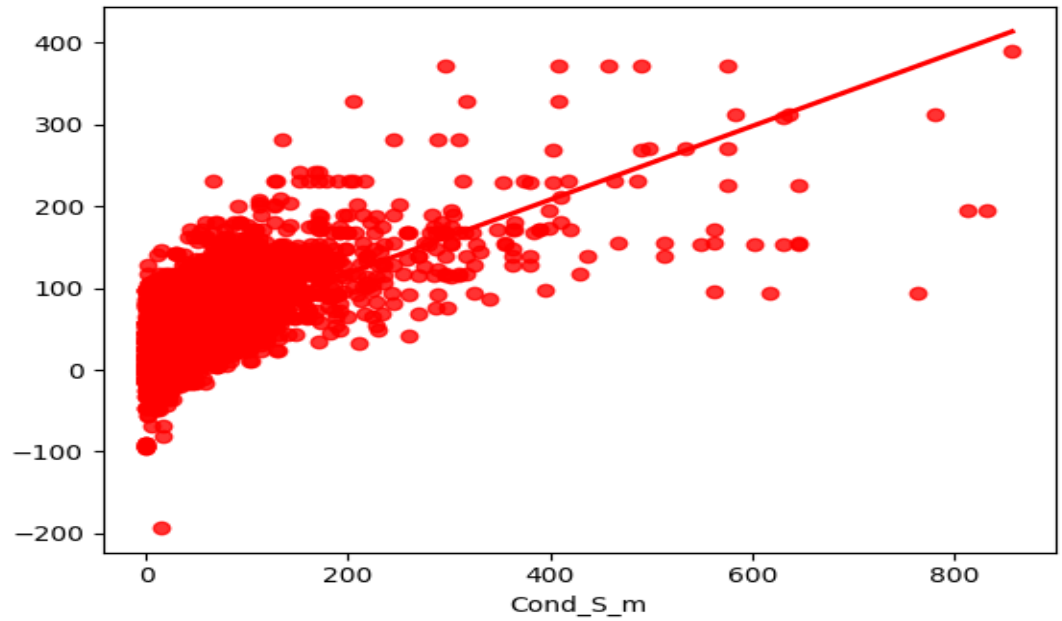


Figure 10 Regression plot with predicted and actualy test value scatter plot around it

Let us go deep with each algorithm predicting Log(T*cond) like we did with molar volume prediction.

5a Using Linear Regression

Graph of accuracy at different test:train split indicate maximum accuracy at 40:60 train test ratio but still accuracy value is not changing much. Accuracy is all between 78% aand 79% with each ratio of train:test.

Reason for constant accuracy- Reason behind it may be the small variance in dataset as the dataet is quite large ie of 9300 datapoints.So few estimations can be made by either reducing dataset or removing few independent varaibles.

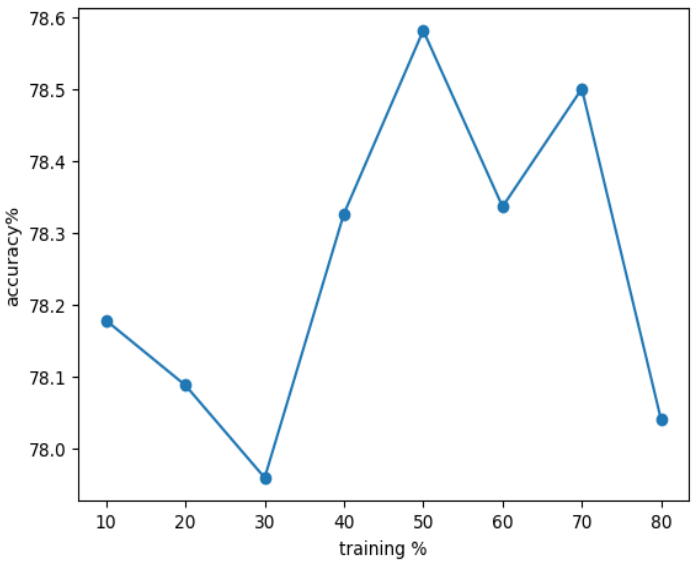


Figure 21 Accuracy vs training% for lin regression

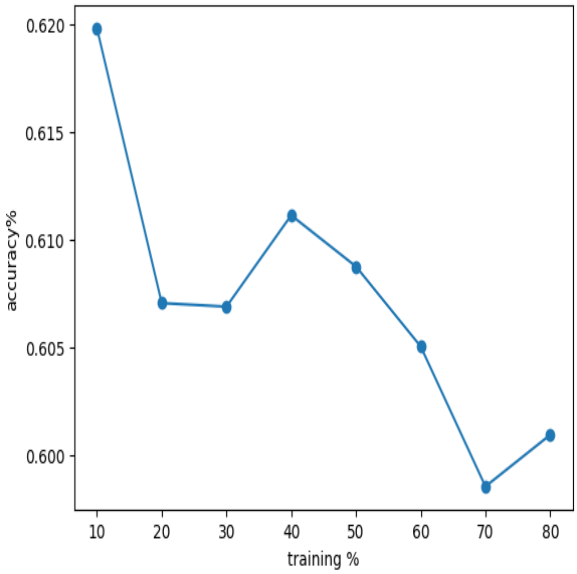


Figure12: rmse vs training%

Data Altering

It must be noted in the dataset has a lot of combination of independent variables where concentration of particular oxides. So in order to see its affects let us drop few variables and then may predction. Below figure shows accuracy change when droping CaO concentration as it has maximum number of concentration of 0 concentration. We observe that accuracy is not changing much.This is because there is almost equal distribution of terms with conc of particular oxides being 0. Though CaO has maximum number of 0 concentration, but the count is just few more than 0s observed in concentration of other oxides.

Another feature which we could realise is by reducing size of dataset, as dataset is quite large. Analyses in figure 13 represents plot of accuracy with only initial 3000 data points. Observed feature is that accuracy variance is increased along with increase in maximum accuracy value. This indicates that original dataset is quite large due to which variance was getting squeeze, resulting in constant accuracy earlier. These kind of analyses helps us observe dataset and properties quite relevantly

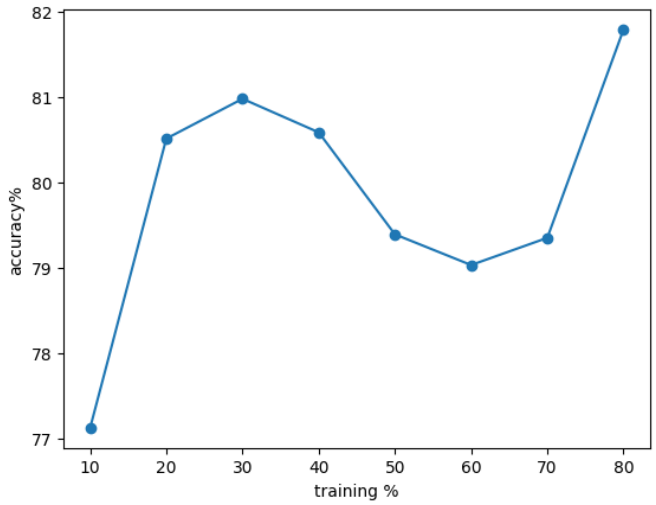


Figure 13 accuracy with only 3000 data points

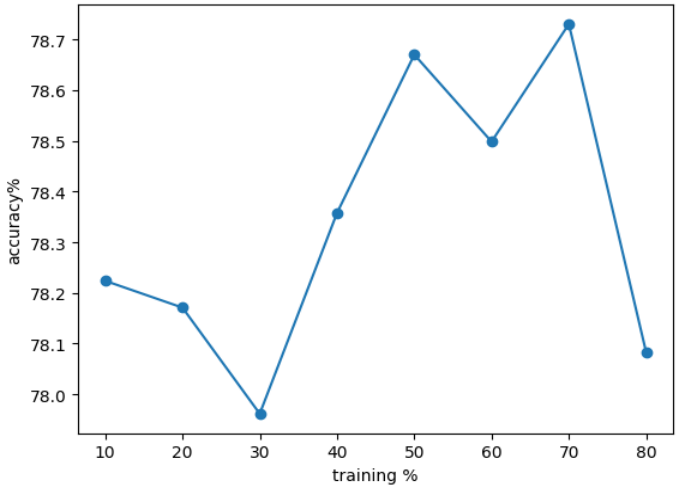


Figure 14: Accuracy while removing CaO as independent var

With rest of the algorithms, analyses will be with whole dataset used in predicting Log(T*cond)

5b Support Vector Regressor

With plot of accuracy vs training dataset %, it I observed that accuracy of model is continuously increasing with almost same rate throughout the model as we increase training dataset points. This indicates overfitting of model with increasing dataset but still accuracy of model at optimum split is better.

It is useful to observe that what we get while predicting conductivity(without considering temperature) as indicated in initial table is that accuracy is negative. This implies svm totally fails in measuring just conductivity as prediction score is below predicting mean as a result. This is another reason for us using log(T*cond) in prediction using every algorithms.

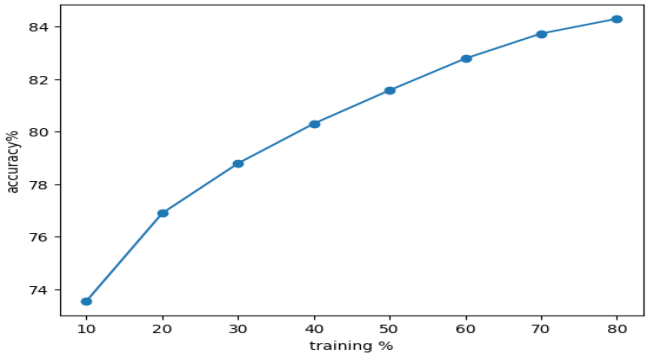


Figure 15: Accuracy% vs train split with svr

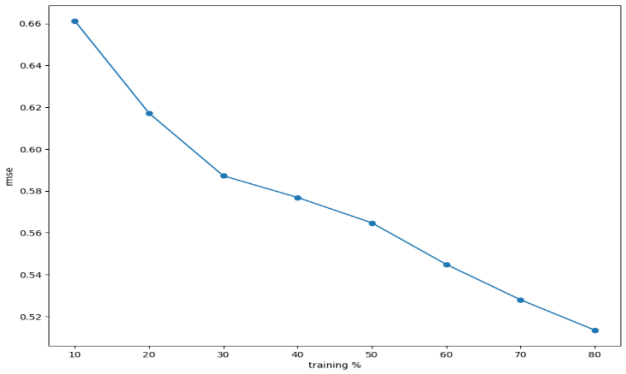


Figure 16: rmse at different training % of dataset

5c Polynomial Regression

Polynomial regression also worked well with electric conductivity dataset. This can be estimated as linear regression gave almost constant accuracy. Only problem was a relatively lower accuracy value which is sorted by expanding the input variable to n dimensions. Here input variable are expanded to 3 degree.

The accuracy achieved by polynomial regression is as high as 97.67%.

Below is the code snippet of applying poly regression on conductivity dataset.

```
#working with polnomial regressor
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size=0.3, random_state=0)
poly = PolynomialFeatures(degree=3, include_bias=False)
X_poly = poly.fit_transform(x)

# Fit a linear regression model to the transformed data
model = LinearRegression()
model.fit(X_poly, y)

# Predict the response for the input data
y_pred = model.predict(X_poly)
print(y_pred.shape,X_poly.shape,y.shape)
# Calculate the R-squared score
r2 = r2_score(y, y_pred)
# Print the coefficients and the R-squared score
print('Coefficients: ', model.coef_)
print('R-squared score: ', r2)
```

```
  1.34299398e+01 -3.04641256e+00  3.27059992e+00 -3.07296344e+00
```

5d Neural Network

As observed by heatmap, dataset is quite rambled up and also non-smooth, so it is strongly adviced to see such model behavoir under neural network so that number of visits to dataset may be high and with backpropogation better estimation is sure to achieved.

As expected, high accuracy% is achieved by using neural network algorithm. Highest accuracy achieved is more than 98%. Relatively less epochs=20 is used due to which execution time of code is reduced to one minute. It is to be noted that accuracy of model is increasing continuously with **higher slope and reaches its maximum value at training dataset=70%, which is considered optimum to avoid any overfitting of model.** The accuracy vs training% plot is almost similar is we take randomly different different subset, indicating the performance of model to best level.

```
# using neural network
t=[]
for i in range (1,10):
 x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=i/10,random_state=0)
 sc = StandardScaler()
 x_train = sc.fit_transform(x_train)
 x_test = sc.transform(x_test)
 model = keras.Sequential([
    keras.layers.Dense(32, activation='relu', input_shape=(x_train.shape[1],)),
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dense(1)])
 model.compile(loss='mean_squared_error', optimizer='adam')
 model.fit(x_train, y_train, epochs=20,batch_size=32)
 predictions = model.predict(x_test)
 rmse=np.sqrt(mean_squared_error(predictions,y_test))
 rmse
 t.append(rmse)
plt.scatter(range(10,100,10),t)
plt.plot(range(10,100,10),t)
plt.xlabel("training %")
plt.ylabel("rmse")
plt.show()
```

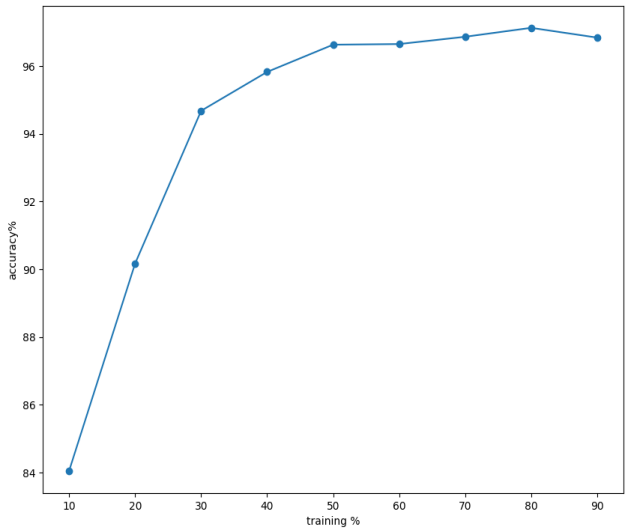


Figure 17 Accuracy% vs training dataset plot for neural network

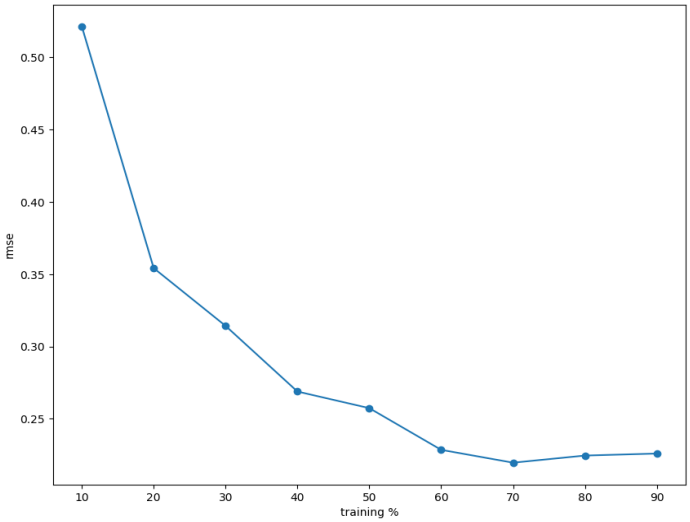


Figure 18: rmse vs train% plot

5e Using Random Forest Regressor

Random Forest regressor again gives one of the best result for prediction. Reason is similar that dataset is bifurcated to different combination of decision trees. Thus different types of features get trained and cumulative result is obtained. It must be noted that unlike molar volume dataset, here we took number of estimators=25 instead of 50 because already we are getting very good accuracy of predictions due to larger amount of dataset. The maximum accuracy achieved is 97.98% abd that too on 70:30 train test split, indicating the perfect working of model

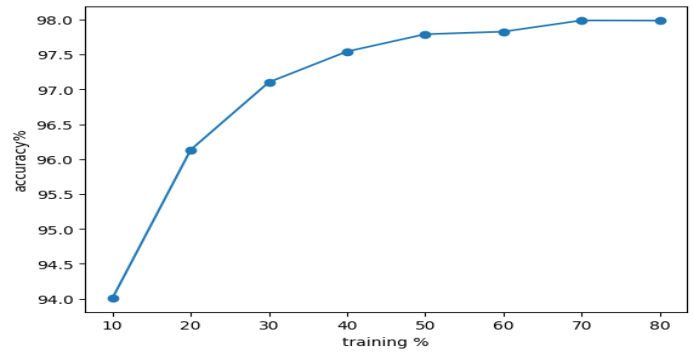


Figure 18 Accuracy vs training set plot

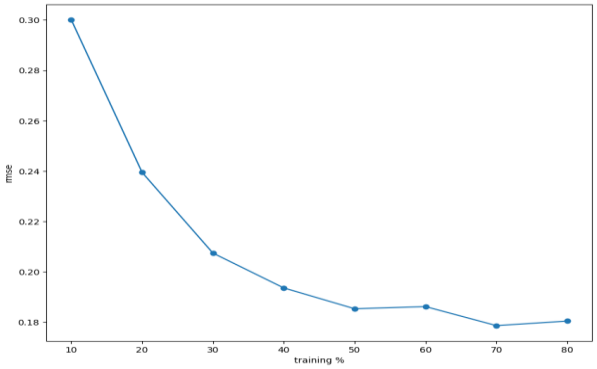


Figure 19: rmse vs traning% plot for random forest regressor

It is worth admiring that accuracy is constantly increasing with increase in training data points till 70% of dataset is trained. This tells us that there is no such case of either overfitting or underfiting is obtained.

5d Summing up the Results

So we analyzed five different prediction techniques and got better results. With the bar graph below, we know that polynomial regression, neural network, and random forest regressor all give almost the same high accuracy of ~98%. So while deciding on the best model, we need to consider a few other considerations.

Time of execution is significantly less in random forest regressor, giving almost high accuracy for all % of the training set. But this may be due to the independent variable's similar values at many data points. While with a neural network, the time of execution is 1 minute, but data is scanned several times, because of which confidence should be high. Also, we get almost the same accuracy curve with any subset of the dataset taken for the experiment.

So any neural network and random forest regression may work, but a neural network is better suited if any addition is given to data points

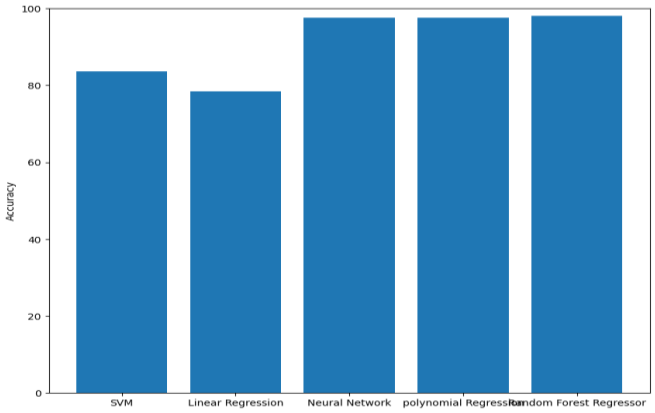


Figure 20 Accuracy of different models at 70:30 split

6 Conclusion

So in this work, primary analysis depends on the theory about the need for computer-assisted knowledge for materials science. With all the different combinations of elements occurring, it is almost impossible to experiment with every sort of estimation. Instead, the study of DFT and, more modernly, machine learning algorithms are used for predictions of different physical property values without going through the tedious process of experiments. This machine learning analysis also tests DFT values, which are predicted with other techniques. This study demonstrated that knowing the behavior of electrical conductivity data led to more accurate results using a logarithmic value. Overall, within the tested techniques, the neural network gave the best prediction accuracy but was by far the slowest technique. If we have as big a dataset as this one, one can use a random forest regressor which almost gives similar accuracy. Another important result obtained that accuracy of all models falls drastically if we directly predict log(cond) instead of log(t*cond) by skipping the values of SiO2 in mixture. This further demonstrates that all oxides, except SiO2 follows Arhenius equation. Moreover, with molar volume dataset we get highest prediction accuracy of 96% with optimum train test split.

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=(7*10)/100,random_state=0)
reg= RandomForestRegressor(n_estimators=50, random_state=0)
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
acc=r2_score(y_pred,y_test)
oxide=["SiO2" ,"Al2O3", "MgO" ,"CaO","Na2O","K2O","Li2O"  ,"MnO", "PbO"]
new_input = np.random.rand(1, 10)
pred_vol=reg.predict(new_input)
print("Predicted volume is",pred_vol[0])
```

Above code is a snippet for predicting molar volume with any input random dataset using random forest. Same can be done for electric conductivity dataset.