Final Report: Kaggle Soil Property Prediction Challenge

Firstname Lastname

1 Project Goal

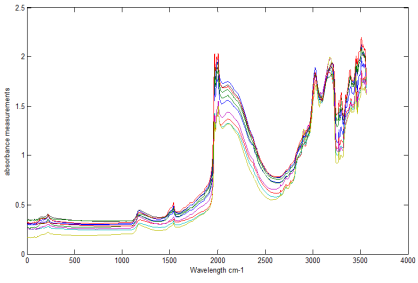
Low cost and rapid analysis of soil samples using infrared spectroscopy provide new opportunities for predicting soil functional properties. These soil functional properties are directly related to a soil's capacity to support essential ecosystem services such as primary productivity, nutrient and water retention, and resistance to soil erosion. If successful, it will provide digital mapping of soil properties in data sparse regions such as Africa and will improve mangament of agricultural and natural resources.

1.1 Problem Statment

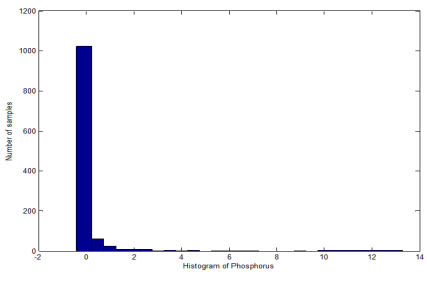
To predict physical and chemical properties of soil using spectral measurements. Regression models for high dimensional data needs to be developed in order to predict five output variables namely Ca, P, pH, SOC and Sand content.

2 Data Description and Preprocessing

Training dataset contains 1,158 soil training samples and 789 testing samples. Each sample is represented by 3,564 features of mid-infrared absorbance measurements. Figure 1(a) shows the infrared spectrum of some soil samples. This figure also suggests that features with more standard devaition might be more important than others. Figure 2(a) depicts the feature similarity matrix and reveals high correlation among them. Also, Figure 2(b) shows feature-output correlation and points out that CO2 band are irrelevant features as they have very low correlation with output. Finally, outliers are removed by inspecting the histogram of output variables such as Phosphorus shown in Figure 1(b). Here samples with $P > 2$ are removed. For SVM and principal component analysis data is normalized by setting mean as zero and standard deviation equal to one.

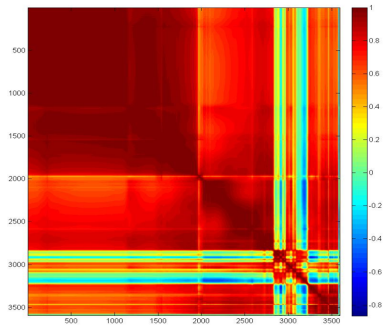


(a) Infrared Spectrum of some soil samples

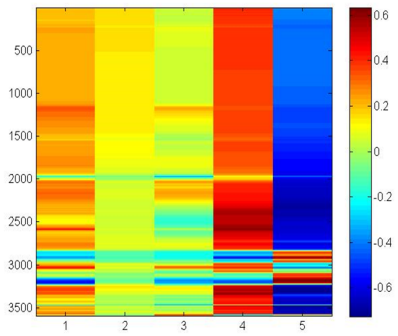


(b) Histogram of Phosphorus

Figure 1



(a) Feature similarity matrix

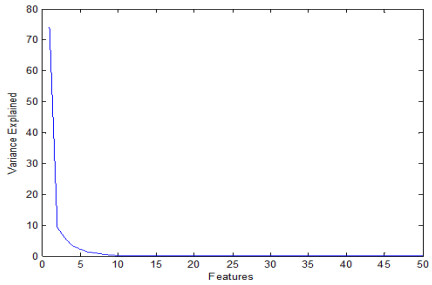


(b) Feature-Output Correlation matrix

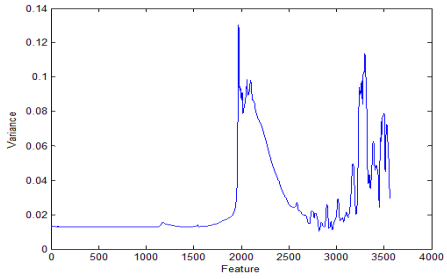
Figure 2

3 Feature Selection

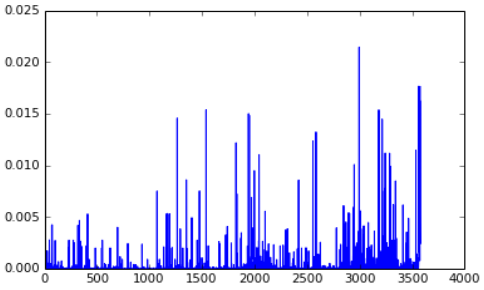
To reduce the high dimension of data, feature selection is performed based on principal component analysis, standard deviation of original features and selection via random forest. Principal components are chosen based on Figure 3(a) as its shows that the first 12 components are enough to explain the data. Figure 3(b) shows the standard deviation of original features and suggests that features with deviation greater than 0.002 might more appropriate to select. And with random forest as shown in Figure 3(c), features with variable importance greater than 0.005 are chosen. Please refer to appendix B for detail about selecting features via random forest.



(a) PCA dimension reduction.



(b) Feature ranking via standard deviation of features.



(c) Feature ranking via Random Forest (shown for Ca).

Figure 3: Feature Selection

4 Modeling Methods

As output variables are not correlated, independent models are created for each output separately. Different SVM models are created based on different feature selection process. Also, for benchmark purpose random forest is used to compare against SVM models.

4.1 SVM Modeling

Due to high dimension, nautral choice of kernel is linear function but rbf is also conisdered. SVM parameter C is calculated according to $Eq.7.40$. While ϵ is calculated by first estimating noise in data by putting $\epsilon = 0$ and then obaining value via $Eq.7.41$. In second approach C is kept high and ϵ is calculated via 10-fold cross validation. Both these approches were used to find out the optimal paramters for each SVM model created.

Following combinations of feature selection and SVM models are considered:

- Simple SVM: No feature selection was performed and regularization paramter C is kept high and epsilon is calculated via 10-fold cross validation.
- PCA+SVM: First 12 principal components are chosen and SVM optimal parameters are chosen according to both approaches. Analytic approach performs well in this case.
- SD features+SVM: Features are ranked according stand deviation and are selected if threshold is greater than 0.02. SVM optimal parameters are chosen according to both approaches. Resampling approach performs well in this case.
- RF features+SVM: Features are ranked according to random forest and are selected if threshold is greater than 0.005. Here also analytic approach performs well.
- Subset of Spectral features+SVM: Due to high correlation of features, every other feature is selected to reduce the dimensions to half. This surprising did well when trained with SVM as expalined below.
- Ensemble of SVMS: Original features are divided into five subsets containing features with indexes represented by 5k, 5k+1, 5k+2, 5k+3, 5k+4. Optimal weights of subsets models are chosen via 10-fold cross validation. This worked because of two reasons. First features are highly correlated, so we are not loosing important information and able to reduce the dimension to one fifth. Second creating ensemble of such SVMs in such a way make sure that all features are used for modeling without loosing any important features in feature selection process. Please refer to appendix A for finding optimal weights of combined models.

4.2 Benchmark: Random Forest

For random forest number of trees are chosen according to out of bag error. As shown in Figure 4 this optimal number is around 60. While for regression depth of decision tree is set to 2 and size of samples features are taken according to theoretical recommended values i.e. square root of original number of features.

5 Experiment and Results

5.1 Software Package

Data preprocessing and visualization is done using matlab. While training and testing is performed using open source machine learning library of Python Scikit-Learn package.

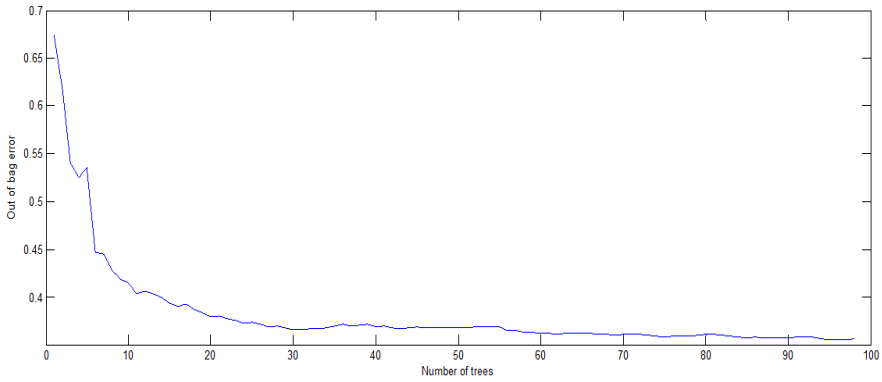


Figure 4: Out of bag error vs number of trees.

5.2 Evaluaion Metric

Each model is evaluated based on root mean squared error calcuated for each output variable. To calculate overall performance of a model, mean of rmse errors of each output variables is taken. This is called as mean columnwise root mean squared error (MCRMSE) $= \frac{1}{5}\sum_{j=1}^{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{ij}-\hat{y}_{ij})^2}$.

5.3 Results and Discussion

Training error is shown in Figure 5 for each output variable and different models. Figure 6 represents the testing error based on MCRMSE metric calcuated for all models which are discussed above.

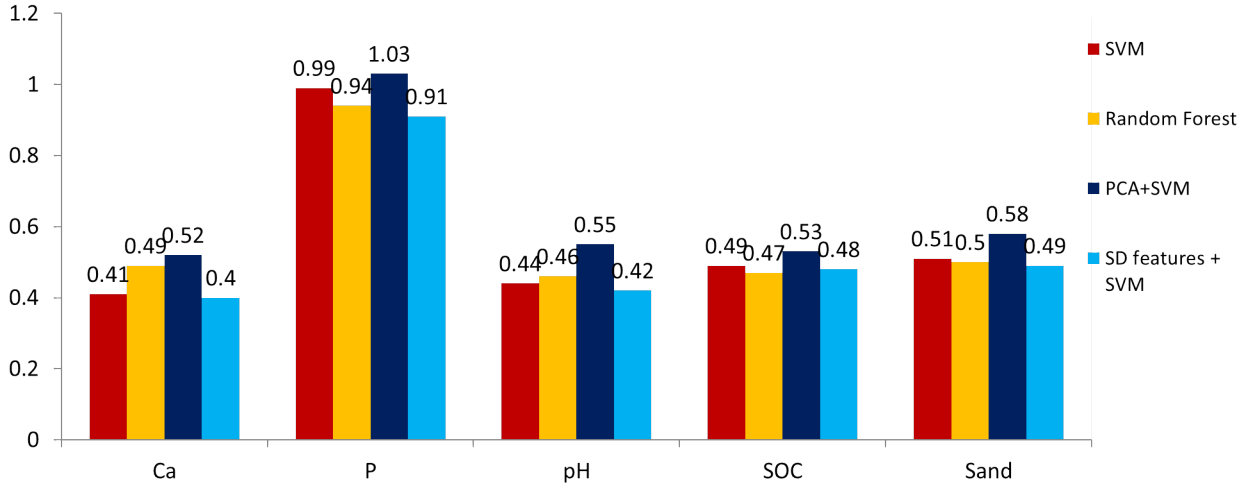


Figure 5: Training Error of different models.

Observing the training error, it becomes clear that linear dimension reduction via PCA is not useful and suggests that data is more complicated in lower dimensions. Phosphorus training error for all models are higher than other variables indicating presence of noise in its measurement. So, methods robust against noise such as ensembling might improve the prediction for this output variable. Also, SVM and random forest performs equally well in training error. But SVM beats random forest in testing error proving its generalization capabilties in higher dimensions. Features

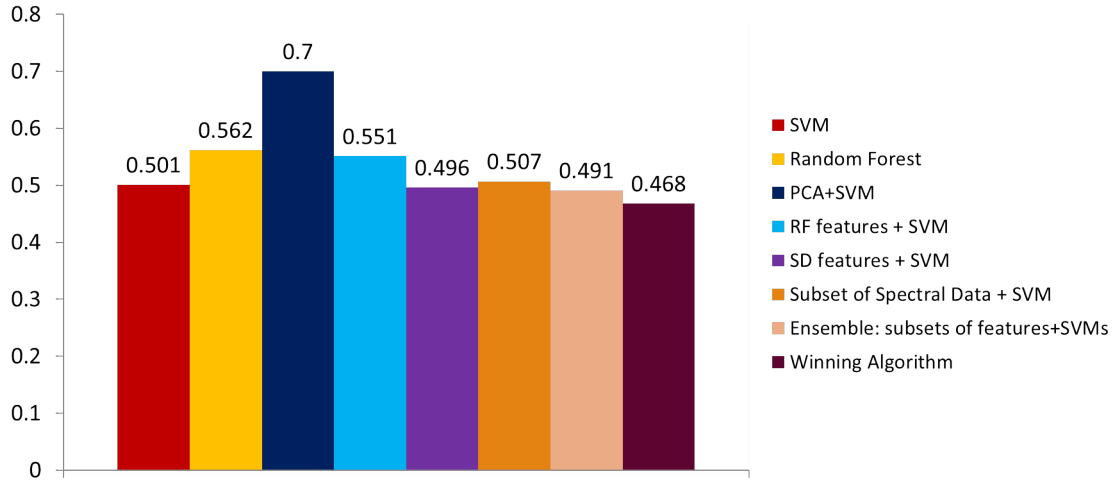


Figure 6: Testing Error of different models.

selection based on standard deviation helped in improving the SVM testing error by 0.004. Finally, ensemble of SVMs proved to be the best model in testing error and gained lead of 0.01 over simple SVM. Above results seem to be good enough for predicting Ca, pH, SOC and sand contents as their rmse errors are low and there might be some room for improving phosphorus output further.

6 Conclusion and Future work

Results shows that SVM generalises well in higher dimensions. Also, there are some disparities between training error and testing error suggesting overfitting in modeling. Ensembling of models helped in reducing that disparity, but its useful only if computational cost is less. As for future work, new features will be considered based on the domain knowledge of spectral methods such as first derivative and second derivate of spectra.

7 Appendix A: Finding optimal weights of combined models

I am presenting two approaches for finding optimal weights of models in ensembling process.

Greedy Algorithm: Weights are adjusted in greedy fashion using 10 K-fold cross validation error. I have formulated a very simple algorithm shown below in Algorithm 1. This algorithm and its parameters are based on my knowledge and experience with the problem given in hand. Initially, I choose all the models weights equal and then find the optimal weights in greedy fashion. Basically, pick i^{th} model and adjust i.e increase and decrease its weight by 0.1 and check cross validation error for any improvement. If there is an improvement of atleast 0.01, keep adjusting it

or else move to the next model weight.

Find optimal weights:
Initialize equal prior weights for each model;
Pick first model with $i = 1$;
while $i < m$ **do**
  Adjust weight of i^{th} model using $w_i' = w_i + 0.1$ and $w_i'' = w_i - 0.1$;
  Keep the weights for other models with indices $i = i + 1, i + 2, ...m$ same and calculate their value using the fact that sum of all weights must be equal to one;
  Compute RMSE error using 10 K-fold cross validation for both new w_i;
  Pick the weight with lowest value RMSE error;
  if $\triangle RMSE > 0.01$ **then**
    Continue adjusting the weight of the same model;
    Follow the same above procedure;
  else
    Adjust the weight of next model $i = i + 1$;
  end
end

Algorithm 1: Optimal weights using greedy algorithm and cross validation error:

Table 1 shows the optimal weights for five SVM models considered for "Ca". In iteration 1 each model has same weight. In iteration 2 weight w_1 is increased and decreased by 0.1. Then rmse error for both different weights are calculated and we find that increased weight rmse error is lower than the decreased weight. So we chose this with rmse error =0.468. But since its rmse error is greater than the original rmse error=0.401, so there is no improvment. Hence we move to the next weight w_2. Turns out, our initial weights were the best and make sense because our inputs are almost same for all five SVM models.

Iterations	**1**	**2**	**3**	**4**	**5**	**6**
w_1	0.2	0.3	0.2	0.2	0.2	0.2
w_2	0.2	0.175	0.3	0.2	0.2	0.2
w_3	0.2	0.175	0.167	0.3	0.2	0.2
w_4	0.2	0.175	0.167	0.15	0.3	0.2
w_5	0.2	0.175	0.167	0.15	0.1	0.2
RMSE@10 K-fold	0.401	0.468	0.477	0.481	0.445	0.401

Table 1: Optimal weights for five SVM models

Convex Optimization method: This method, I realized later and presenting just for sharing purpose or can be considered as future work. We want to combine our models output such that their rmse error with respect to true output is minimum over all samples. Thus, our optimization problem is of form shown below where $\widetilde{Y}$ represents prediction matrix and y is the true output column for training data. Each column of $\widetilde{Y}$ represents a single model predicted values for a output variable. So if we have five SVM models (say for "Ca") then $\widetilde{Y}$ matrix dimension is $n \times 5$, where n equals to number of samples and each column represents predicted values of "Ca" by different models. And w will be column vector of size equals to number of models.

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & ||\widetilde{Y}w - y||^2 \\ \text{subject to} \quad & \sum_{i=1}^{m} w_i = 1, \quad w_i \geq 0,\ i = 1, \ldots, m. \end{aligned}$$

A simple convex optimization problem is formulated for finding optimal weights and can be solved using CVX optimization package for matlab.

8 Appendix B: Feature Ranking via Random forest

Practically, one can perform feature selection via random forest as follows:

1. Train random forest for a output variable say "Ca". Select its optimal parameters based on oob error and recommended values as discussed in section 4.2.
2. Plot feature importance calculated internally while training the above model.
3. Inspect the plot and choose the appropriate threshold.

Computing feature importance: A fast method is to add up the gini importance (i.e. higher descrease in gini for a feature means its more important) of each feature over all trees in the forest and sort them for ranking. For another method based on permutation test, I would refer to authors explanation[2].

9 References

1. Kaggle African Soil property competition: `http://www.kaggle.com/c/afsis-soil-properties/`
2. Random forest by Leo Breiman: `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm`
3. Learning from Data: Concepts, Theory and Methods, by V.Cherkassky and F.Mulier, Second edition.
4. Textbook Predictive Learning, by V.Cherkassky, 2013.