Predicting rainfall using ensemble of learners

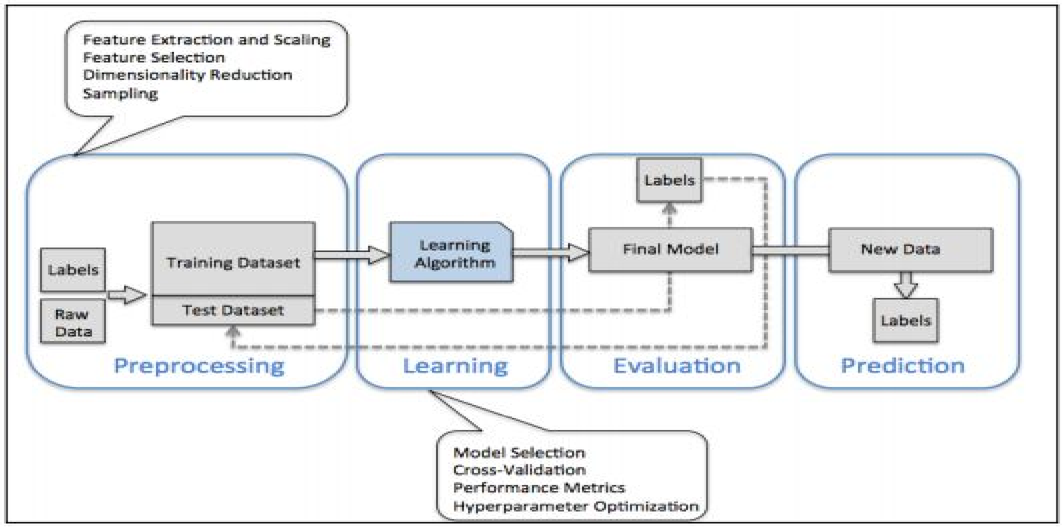
Barada Prasanna Acharya, Jithendra Gandikota, Vinay Bagade
{vbagade, acharyab, jgandiko}@uci.edu

Abstract

The goal behind ensemble methods is to combine different classifiers into a meta-classifier that has a better generalization performance than each individual classifier alone. The ensemble decreases the variance in the models caused because of overfitting. Neural networks are notorious for overfitting if there are a large number of parameters. Various techniques like Dropout and early training are used to remedy this. In this project, we have used a Mixture of Experts (Voted Classifier) of a neural network and a large number of random forests (with decent classification accuracy) to decrease the variance of the neural network. The result is a weighted ensemble which gave us a top 5 ranking on the kaggle competition with an area under roc of 0.66806.

1. Introduction

When it comes to a predictive analysis just selecting a classifier and varying its parameter to get maximum accuracy is only half the job. There is a workflow involved. The diagram below shows a typical workflow diagram for using machine learning in predictive modeling.



For this project we have followed the following workflow:

Preprocessing Step – getting data into shape

1. Scaling the data
2. Removal of outliers

3. Selecting meaningful features
    - Sparse solutions using L1 regularization
    - Feature importance with Random Forests

Data visualization - Understanding the data

1. Exploratory data analysis (EDA)
2. Transforming the data into lower dimension for plotting
    - PCA
    - Kth highest entropy features of random forests
2. Plotting the data in the lower dimension
3. Subplots of features

Modelling the data- selecting a classifier and understanding its limitations

1. Mixture of experts
2. Neural Networks
3. random forests
4. Optimizing the parameters (Grid Search)
5. Improve on the current classifier

In this project preprocessing and removal of outliers was done by Vinay .Data visualization step was done by Jithendra and Barada , the random forest part is done by Barada and the neural network part is done by Vinay and Jithendra. The final ensemble collaboration part is done by Vinay.

2. Data Preprocessing – getting data into shape

Scaling the data

Raw data rarely comes in the form and shape that is necessary for the optimal performance of an algorithm. Many machines learning algorithms (especially neural networks) require that the selected features are on the same scale for optimal performance, which is often achieved by transforming the features in the range [0, 1] or a standard normal distribution with zero mean and unit variance.

Removal of Outliers

Removal of outliers is an important preprocessing task. The presence of outliers interferes with good prediction and affects accuracy. In our project, we used an approach to clustering: **Density-based Spatial Clustering of Applications with Noise (DBSCAN)** .

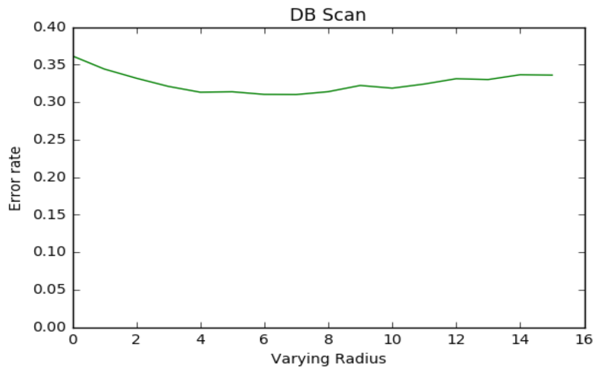
In DBSCAN, a tag is assigned to each data point using the following criteria:

• A point is considered as **core point** if at least a specified number (MinPts) of neighboring points fall within the specified radius r

• A **border point** is a point that has fewer neighbors than MinPts within r, but lies within the r radius of a core point.

• All other points that are neither core nor border points are considered as **noise points**.

In our project, we picked the optimal value r which improved the accuracy(reduced the error the most) the most after removal of noise points. The plot is shown below.



Selecting meaningful features

1) **Sparse solutions with L1 regularization**

   In, L1 regularization yields sparse feature vectors; most feature weights will be zero. Sparsity can be useful in practice if we have a high-dimensional dataset with many features that are irrelevant. In the project, we tried to get the most meaningful data by training on a linear classifier with L1 regularization. Then we observed the weights of the classifier. The features corresponding to non zero entries were taken as the most important features. The weight matrix for the dataset on the classifier is shown below.

   [0.280, 0.000, 0.000, -0.0282, 0.000, 0.230, 0.710, 0.000, 0.000, 0.130, 0.000, 1.236, 1.246, 0.000]

2) **Accesing Feature Importance from random forests**

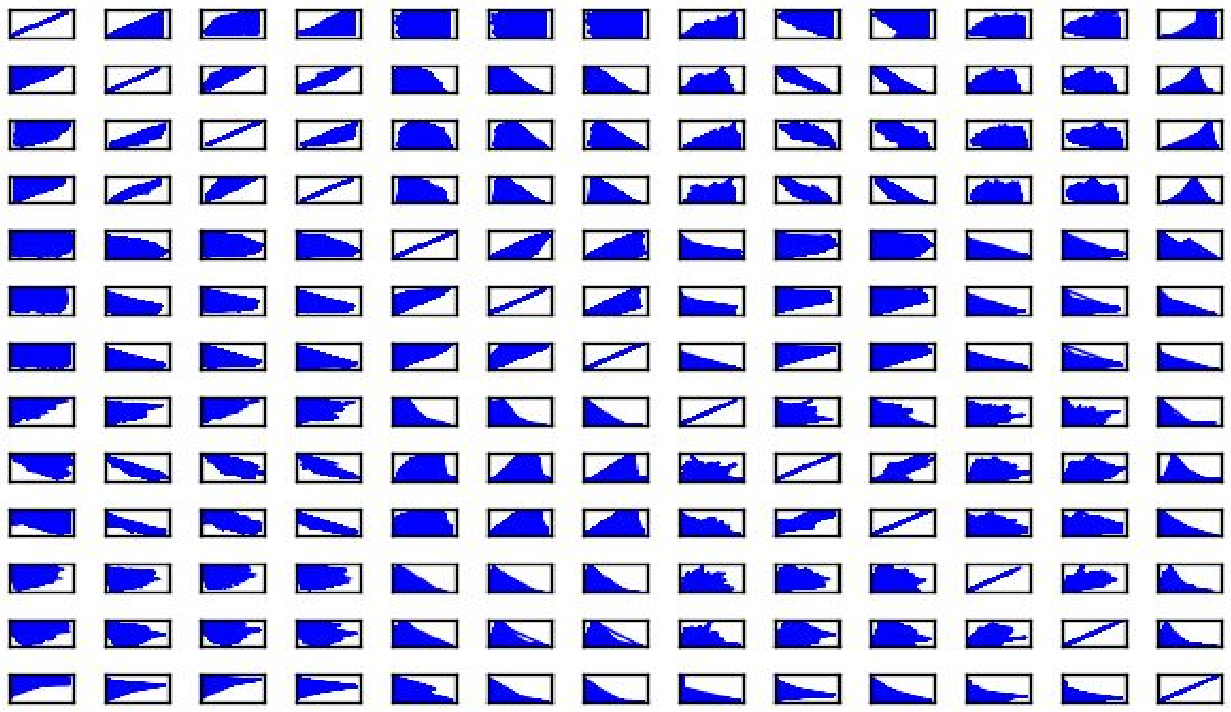
   Using Random Forest we computed the average impurity decrease of all the trees to measure the importance of features. Scikit learn has an implementation of the same (**RandomForestClassifier**). We observed that the importance of features matched the inference from the L1 regularization method. The feature importance computed from **Forest.feature_importances_** of **RandomForestClassifier** are shown below.

[0.182508 ,0.058574 ,0.050954 , 0.131983 , 0.006564 , 0.78249, 0.060717 ,0.032039 0.25385, 0.022369, 0.22070 , 0.22070, 0.005070]

3. Data Visualization- Understanding the data

Exploratory data Analysis (EDA)

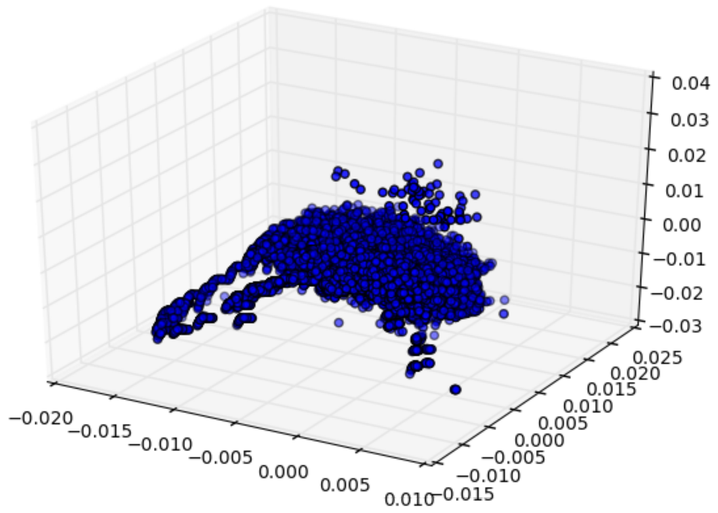
Before applying any classifier we should first get a "feel" of data. What we mean by that is we will create a scatterplot matrix that allows us to visualize the pair-wise correlations between the different features in this dataset in one place. This is shown below as a 13X 13 matrix



Using this scatterplot matrix, we can now quickly eyeball how the data is distributed and whether it contains outliers.

Principal Component Analysis

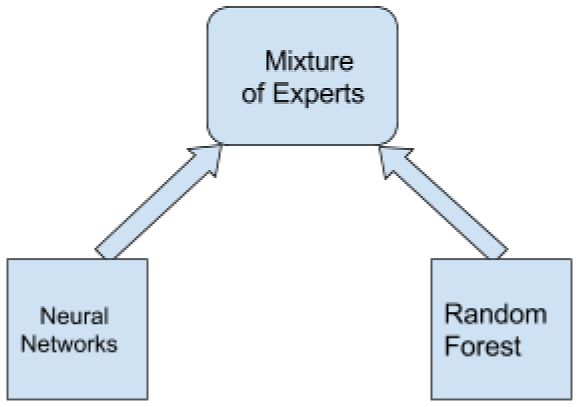
Each data point in our data is 14-dimensional vector. So, it becomes hard to visualize the data. What we came up with was getting the three most principal components using singular value decomposition(**SVD**) and taking three eigenvectors corresponding to three eigenvalues with the highest magnitude and then plotting the data on these. So we can look at the data in a 3-D space, feel the clusters, outliers and select the classifiers accordingly. Shown below is the plot of data on its three most principal components.



4. Modelling the data- Selection of the classifier

Mixture of Experts

A mixture of experts is a neural network based architecture which when trained gives a weight value to each classifier in our ensemble. It is sort of a soft variant of max voting. In our Project, we have used an ensemble of a Neural network and a bunch of random forests to decrease the variance of the neural network.



Why mixture of experts?

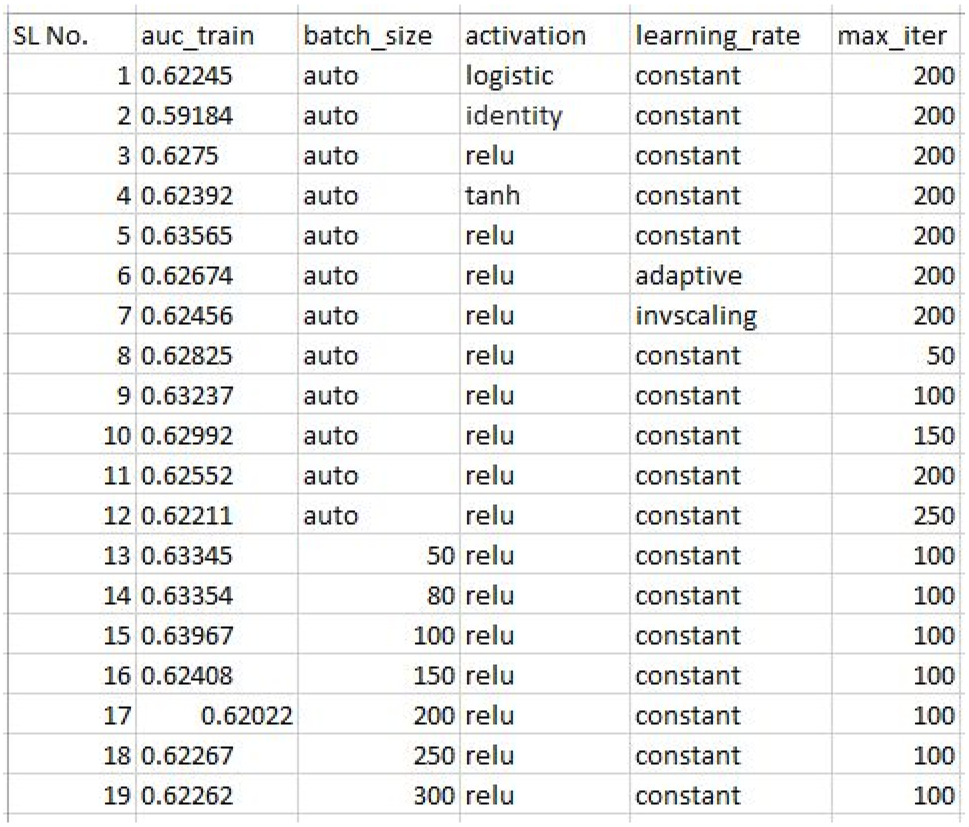
As it can be seen from the visual plot of the Principal Components, that the data is clustered around two major clusters and the intuition behind using a weighted voting classifier is that a single classifier in ensemble will try to fit the data in a cluster almost perfectly (overtraining it) ignoring data from other clusters. Thus by applying a weighted voting we would decrease the overall variance and shape the final classifier to perform better on all the clusters.

Neural Networks

We build the neural network using sklearn Multilayer Perceptron class and trained it on data from the preprocessing step. We took a 70-30 split of the data and performed a grid search for hyperparameter tuning. Following are the results from the dataset. From the below table we are selecting parameter values for neural network learner by making comparison between auc of test data.

Batch_size = 100 , activation = relu, learning_rate = constant, max_iter =100

*relu = rectified linear unit function



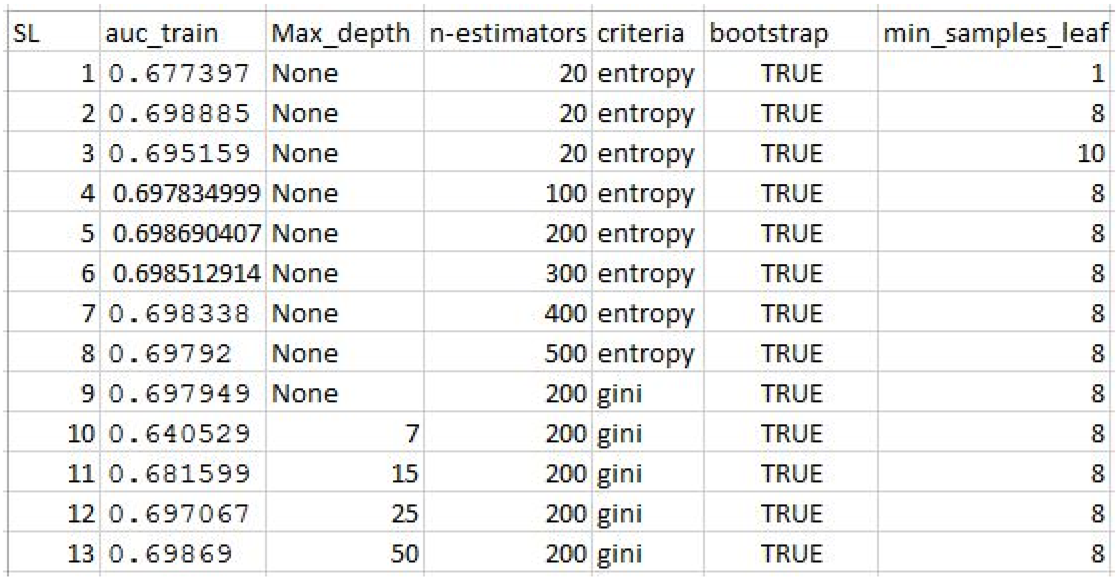
SL No.	auc_train	batch_size	activation	learning_rate	max_iter
1	0.62245	auto	logistic	constant	200
2	0.59184	auto	identity	constant	200
3	0.6275	auto	relu	constant	200
4	0.62392	auto	tanh	constant	200
5	0.63565	auto	relu	constant	200
6	0.62674	auto	relu	adaptive	200
7	0.62456	auto	relu	invscaling	200
8	0.62825	auto	relu	constant	50
9	0.63237	auto	relu	constant	100
10	0.62992	auto	relu	constant	150
11	0.62552	auto	relu	constant	200
12	0.62211	auto	relu	constant	250
13	0.63345	50	relu	constant	100
14	0.63354	80	relu	constant	100
15	0.63967	100	relu	constant	100
16	0.62408	150	relu	constant	100
17	0.62022	200	relu	constant	100
18	0.62267	250	relu	constant	100
19	0.62262	300	relu	constant	100

Random Forest

To reduce the variance of the neural network an ensemble of random forests is used. These trees participate in the decision making in the committee of classifiers. Shown below is the error rate on the forest of random forests by varying the number of trees. Here we perform a grid search on the various parameters of the Random Forest. From the below table we are selecting parameter values for random-forest learner by making comparison between auc of test data.

.

Max_depth = 50, n-estimators = 200, criteria = entropy, bootstrap = true, min_sample_leaf =8



SL	auc_train	Max_depth	n-estimators	criteria	bootstrap	min_samples_leaf
1	0.677397	None	20	entropy	TRUE	1
2	0.698885	None	20	entropy	TRUE	8
3	0.695159	None	20	entropy	TRUE	10
4	0.697834999	None	100	entropy	TRUE	8
5	0.698690407	None	200	entropy	TRUE	8
6	0.698512914	None	300	entropy	TRUE	8
7	0.698338	None	400	entropy	TRUE	8
8	0.69792	None	500	entropy	TRUE	8
9	0.697949	None	200	gini	TRUE	8
10	0.640529	7	200	gini	TRUE	8
11	0.681599	15	200	gini	TRUE	8
12	0.697067	25	200	gini	TRUE	8
13	0.69869	50	200	gini	TRUE	8

Ensemble of learners

Finally we combine both these methods a do a final parameter tuning to predict error rate.We use weighted average probabilities (soft voting) for the final prediction.

5. Conclusion

In this project we have implemented the complete machine learning pipeline. We have scaled the data by normalizing it , cleaned the data by removing outliers using **DBSCAN.** Further**,** we have extracted meaningful features from the data using L1 regularization and Important features from Random Forests. We have then visualized the data by inspecting clusters and suggested a classifier accordingly. Finally, we have performed a grid search for hyperparameter tuning and reported the result of area under curve of **0.66806 on Kaggle** which gave us a **top 5 ranking** on the same.