Solar power systems project

Abstract

The analysis focuses on the use of different supervised classification models on the DeepSolar dataset which contains various features corresponding to different areas and the target variable is a binary variable which indicates the number of solar power systems in the area. Initially, the data is pre-processed, and only certain important features are selected for the analysis using the method of correlation and variable importance function of random forest. Different models are fitted on the data separately to assess the performance on the train and test data. Various methods are used to indicate the performance of the models such as ROC curve, contingency tables and accuracies. Later, resampling method is used to evaluate the models on the validation data and thereby predict the target variable for the test data using the best model chosen from the validation data for each iteration. Finally, using the summary statistics of the models chosen for each iteration, the best model for the data set is found as the one selected for most number of iterations.

Introduction

The dataset for the analysis is a subset of the DeepSolar database where each row is a “tile” which is an area corresponding to a detected solar power system constituted by a set of solar panels in the area. A set of other features like social, economic, environmental, geographical, and meteorological aspects of the tile (area) in which the system has been detected are also recorded corresponding to each tile.

The dataset is a 20763 x 81 dataset with 80 related features and one binary target variable solar_system_count. It indicates the coverage of solar power systems in a given tile and takes value “low” if the tile has a low number of solar power systems (less than or equal to 10), while it takes value “high” if the tile has a large number of solar power systems (more than 10).

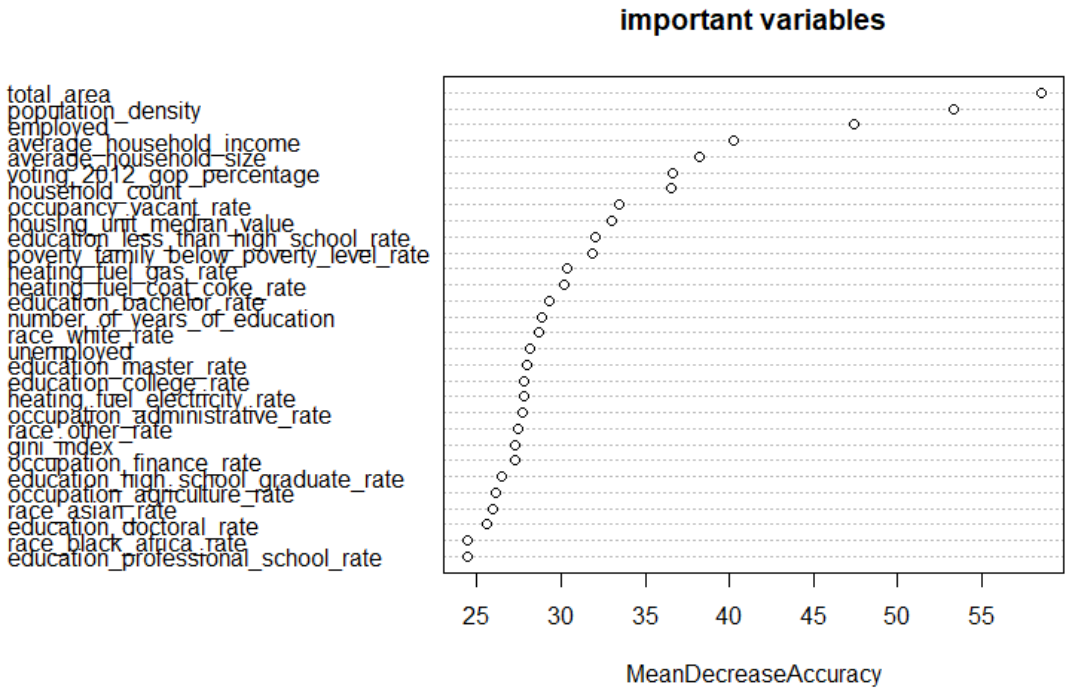
The main aim of the analysis is to predict the target variable i.e., given a particular tile and the social, economic , geographical and other features corresponding to the tile, we need to predict whether the tile has low or high number of solar power systems (i.e., whether less than or greater than 10). Hence this is a classification problem under supervised learning methods using various classification models so that the model that classifies the data best is chosen.

Methods

Data pre-processing

The data set for the analysis has dimension 20763 x 81 which is large. Hence in order to reduce the size of the data, we need to remove certain variables which are insignificant in the analysis from the data. In order to do that we can remove the variables which are highly correlated with each other, i.e., highly dependent on each other.

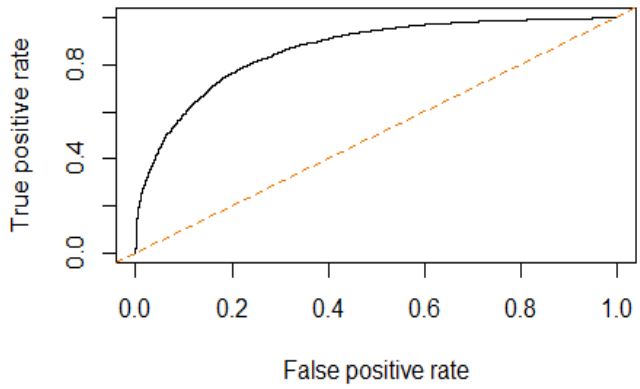
Examining the variables of the data indicates that there are categorical variables in the data which prevents us from finding the correlation of the whole variables. Hence, we first discover the important variables using the varImpplot function from randomForest and we can infer from the output that no categorical variables are included in that i.e., categorical variables are not among the important variables. Thus, we delete the categorical variables from the original dataset and then apply the varImpplot function in order to find out the important variables from these set of uncorrelated variables.



Next, we need to split the dataset into train and test data in order to run few classification models. We first run various model on the training data separately to examine their accuracy and then run all of them simultaneously on random samples drawn 50 times and asses their performance.

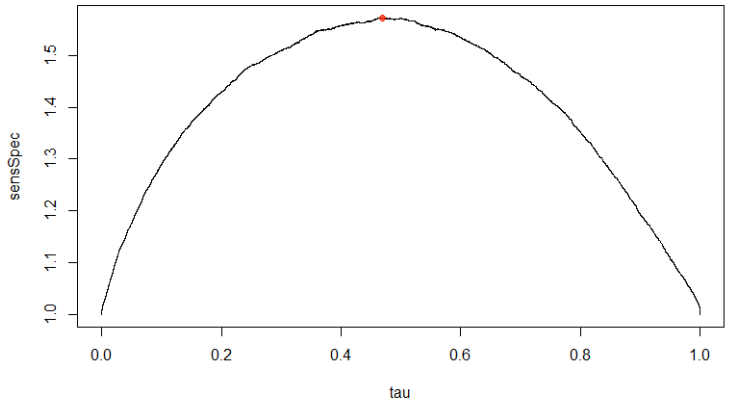
Logistic Regression

The logistic regression model is fitted on the training data using the function gml(). A threshold value tau is taken as 0.5 and the target variable values are predicted. Here value 1 is taken for “high” and 2 for “low” categories. The actual values of “solar_system_count” is then compared with the predicted value and a ROC curve is plotted and its area under curve which indicates the accuracy of the model. The AUC vale for the model’s prediction on the train data is approximately 0.87.



	Predicted class	
	high	**low**
high	1154	4323
low	3822	1069

Although the dataset is not highly unbalanced, we can compute the optimal tau for which the observations are classified by balancing the sensitivity and specificity. The optimal tau is the maximum value of tau which satisfies the condition and can be seen at the red dot in the graph. Here the optimal value of tau is 0.469 which classifies the data in best possible way.



Random forest

The random forest model is fitted to the training data using the function randomForest() from the package randomForest. The model is used to predict values for both train data and test data and the

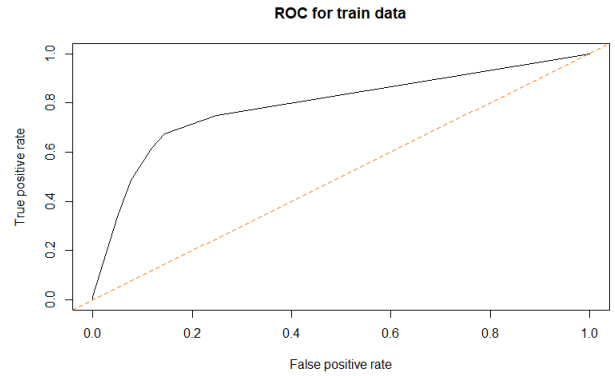
accuracies are compared. The accuracy for training data is 100% and for test data is approximately 86% . The training data accuracy will always be higher than the test data accuracy since the model is fitted on the training data.

	Predicted class on test data	
	high	**low**
high	4785	664
low	773	4146

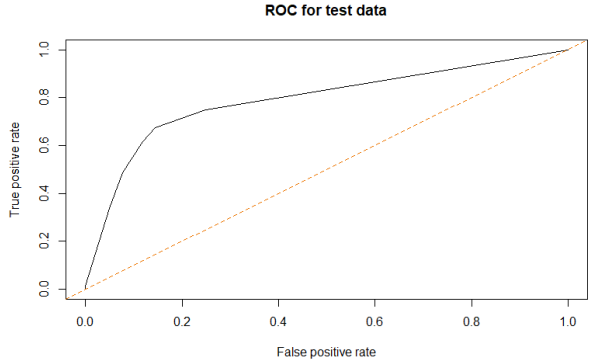
	Predicted class on train data	
	high	**low**
high	5451	0
low	0	4917

Classification tree

Classification tree is implemented on the data using the function rpart() from the package rpart. The model is fitted to the training data. The function predict is then used to perform predictions and also the class probabilities according to the fitted tree is calculated (indicated with the variable phat). A cross tab between the predicted classes and the actual target variable is computed. Similar to logistic regression, ROC curve can be plotted for classification tree as well and the area under the curve is computed which turns out to be 0.76 in this case. The predictions are again computed on the test data and similarly the ROC curve and the area under the curve is computed. The AUC value for test data is 0.49.



	Predicted class on train data	
	high	**low**
high	4685	1521
low	766	3396



	Predicted class on test data	
	high	**low**
high	3307	2965
low	2144	1952

Support Vector Machines

Support vector machines is another model used for binary classification and is implemented using the function kvsm(). The model is fitted on the training data and the performance is measured both on the training data and test data for comparison similar to the random forest model. Here the training accuracy is 87% and the test accuracy is 85%.

	Predicted class on train data	
	high	**low**
high	4747	704
low	564	4353

	Predicted class on test data	
	high	**low**
high	4703	746
low	746	4173

Bagging

Bagging is a classification technique which uses bootstrapping to increase the stability of a classification method. It is implemented using the function bagging() from the package “adabag”. The model is fitted on the training data and the prediction is done on both training and testing data using the function predict(). The accuracy of the training data is a bit higher than the testing data by few decimal points. The training accuracy is around 79% and test accuracy is around 78%.

	Predicted class on train data	
	high	**low**
high	4536	915
low	1246	3617

	Predicted class on test data	
	high	**low**
high	4531	918
low	1299	3620

Classifier selection

We use resampling method to evaluate the performance of the different classifiers on the test and the validation data. The models are run 50 times and each time the train, test and validation data samples are drawn. The training data consist of the 50% of the data and the rest 50% is divided equally between the validation and the test data. A data frame “out” is declared in order to store the validation accuracies, the best model and the test accuracy of the best model for each of the iteration.

For each iteration, after the samples are drawn, the various models are fitted on the training data using different functions. Next, all the models are used to predict the solar_system_count using the validation data and the corresponding accuracies are stored in the data frame “out”. The validation data serves the purpose of comparison of various models and the model with highest accuracy is

chosen as the best model. After the validation accuracies of the models are computed, their accuracies are compared using “switch()” and the model with the highest validation accuracy is used to predict on the test data. The test data is used to evaluate the generalization error of the chosen classifier. The chosen best model name and the test accuracy of the model is stored in the “out”.

After the completion of all the iterations, we compute a table of the 5th column of “out” which contains the best model chosen for each of the iteration and the test error of this best model in the last column. Finally, a boxplot with the best models from the resampling method is plotted with their test accuracies.

Results and discussion

The analysis consisted of running various classification models on the training data and evaluating the performance on the test data separately, and then evaluating the performance of the models with validation data several times with resampling method.

When the models are run separately on the train and test datasets and their performance is evaluated, we can see that the train accuracy is always higher than the test accuracy. This is because the model is fitted on the training dataset and hence the data is not new for the model when it predicts. However, when is the prediction is made on the test data which is new for the model, the model will not perform as good as the training accuracy.

Next, the performances of the different models are evaluated on the validation data sets in 50 iterations and the model with the highest validation accuracy is used to predict on the test data. The output from the iterations is a data frame “out” which contains the validation accuracies of all the models in each iteration, the best model of the iteration and the test accuracy of that best model. The first 30 values of the data frame “out” look like this.

```
> out
   val_logistic val_randomforest val_classification_tree   val_svm val_bagging          best test accuracy
1     0.2119985        0.8643904               0.7696759 0.8532022   0.7853009 random_forest     0.8613040
2     0.2116127        0.8626543               0.7775849 0.8614969   0.7789352 random_forest     0.8576389
3     0.2181713        0.8516590               0.7669753 0.8464506   0.7814429 random_forest     0.8688272
4     0.2125772        0.8670910               0.7847222 0.8632330   0.7903164 random_forest     0.8589892
5     0.2108410        0.8626543               0.7731481 0.8587963   0.7800926 random_forest     0.8584105
6     0.2189429        0.8605324               0.7638889 0.8526235   0.7820216 random_forest     0.8640046
7     0.2112269        0.8570602               0.7754630 0.8499228   0.7785494 random_forest     0.8587963
8     0.2146991        0.8699846               0.7633102 0.8576389   0.8016975 random_forest     0.8670910
9     0.2062114        0.8584105               0.7534722 0.8584105   0.7766204 random_forest     0.8547454
10    0.2119985        0.8640046               0.7816358 0.8591821   0.7883873 random_forest     0.8645833
11    0.2139275        0.8595679               0.7741127 0.8557099   0.7814429 random_forest     0.8667052
12    0.2210648        0.8601466               0.7758488 0.8485725   0.7872299 random_forest     0.8657407
13    0.2050540        0.8659336               0.7798997 0.8605324   0.8020833 random_forest     0.8559028
14    0.2229938        0.8555170               0.7673611 0.8508873   0.7781636 random_forest     0.8647762
15    0.2206790        0.8551312               0.7714120 0.8514660   0.7731481 random_forest     0.8593750
16    0.2137346        0.8578318               0.7795139 0.8520448   0.7941744 random_forest     0.8532022
17    0.2148920        0.8611111               0.7791281 0.8597608   0.7849151 random_forest     0.8624614
18    0.2170139        0.8599537               0.7611883 0.8539738   0.7729552 random_forest     0.8568673
19    0.2121914        0.8572531               0.7723765 0.8520448   0.7883873 random_forest     0.8609182
20    0.2224151        0.8539738               0.7673611 0.8499228   0.7820216 random_forest     0.8599537
21    0.2143133        0.8665123               0.7777778 0.8599537   0.7883873 random_forest     0.8545525
22    0.2199074        0.8505015               0.7658179 0.8535880   0.7702546           svm     0.8587963
23    0.2172068        0.8564815               0.7793210 0.8545525   0.7883873 random_forest     0.8572531
24    0.2127701        0.8559028               0.7625386 0.8528164   0.7860725 random_forest     0.8624614
25    0.2025463        0.8667052               0.7689043 0.8532022   0.7839506 random_forest     0.8643904
26    0.2137346        0.8640046               0.7613812 0.8514660   0.7926312 random_forest     0.8514660
27    0.2031250        0.8622685               0.7777778 0.8547454   0.7947531 random_forest     0.8647762
28    0.2100694        0.8638117               0.7746914 0.8599537   0.7868441 random_forest     0.8634259
29    0.2166281        0.8628472               0.7667824 0.8634259   0.7968750           svm     0.8580247
30    0.2116127        0.8572531               0.7714120 0.8510802   0.7798997 random_forest     0.8586034
```

The table of the best model chosen for each iteration is also computed. This says that out of 50 iterations, 47 times random forest was chosen as the best model on the validation data and SVM as chosen 3 times.

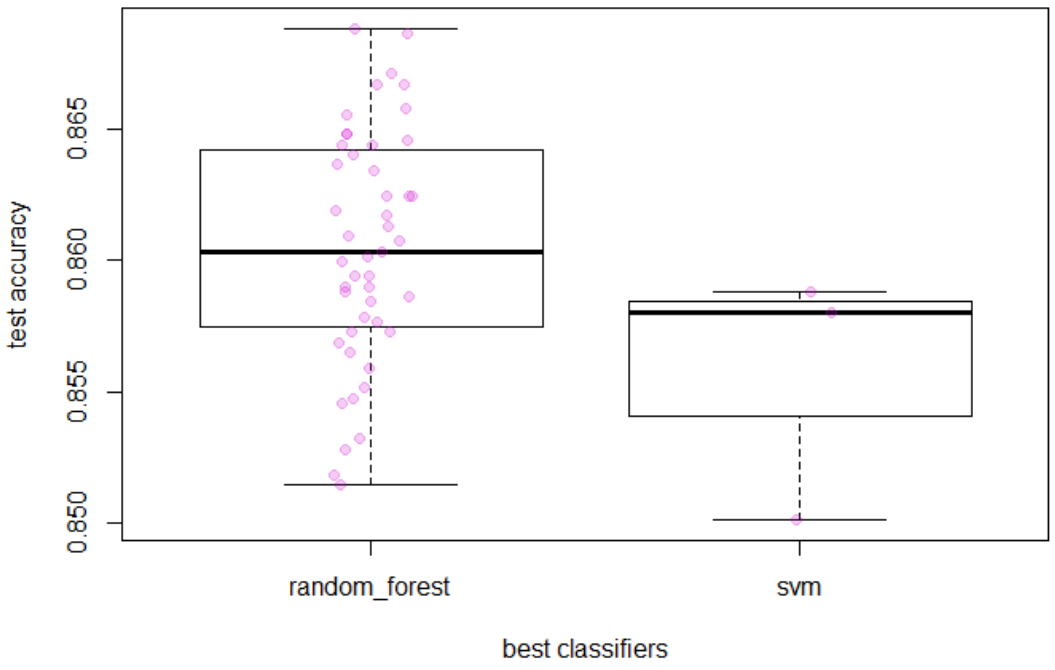
random_forest	SVM
47	3

The summary statistics of the test accuracies of both the models, random forest and SVM is given below.

```
$random_forest
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8515  0.8574  0.8603  0.8605  0.8642  0.8688

$svm
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8501  0.8541  0.8580  0.8556  0.8584  0.8588
```

The boxplot of the best classifiers and their test accuracies is also plotted.



According to statistics and the box plot, the test accuracy of random forest model lies between 85.1% and 86.8% and that of SVM lies between 85% and 85.8%. The average test accuracy of random forest is 86.05% slightly higher than that of SVM which is 85.56%.

Conclusion

The following inferences were drawn from the analysis :

- When the models were separately used to predict values in train and test data, the train accuracy was always higher test accuracy. In some cases ,there were also signs of overfitting.
- When samples were drawn repeatedly and models were fitted in the validation data, the highest accuracy was for random forest in almost all the iterations and it was used to predict the test data.
- Hence random forest is chosen as the best model to classify the target variable “solar_system_count” on the basis of social, economic and other features as “low” or “high” indicating the number of solar power systems in the corresponding tile, with average test accuracy of 85.9%.