Assignment 4

Data Science In Industry

Megha Swarini Sinha

71189549

**Comparison of MLR, sk-learn and Caret Packages:**

*"Scikit-Learn is known for its easily understandable API and for Python users and MLR became an alternative to the popular Caret package with more a large suite of algorithms available and an easy way of tuning hyperparameters. These two packages are somewhat in competition due to the debate where many people involved in analytics turn to Python for machine learning and R for statistical analysis.*

*One of the reasons for a preference to Python could be because that current R packages for machine learning are provided via other packages that contain the algorithm. The packages are called through MLR but still requires extra installation. Even external feature selection libraries are needed and they will have other external dependencies that need to be satisfied as well." ([8](https://blog.exxactcorp.com/scikitlearn-vs-mlr-for-machine-learning/))*

**Scikit- Learn:**

Easily understandable API for tuning hyper parameters.

This package is coded in Python which is faster in computation and can be used to carry out a wide range of operations. This is mostly used in application development and is easy to implement machine learning algorithms.

Mainly it is famous because of its BSD license which is permissive free software license and convenient for industry to use.

**3**Many people chose R for statistical learning and python for Machine Learning. This might be due to the fact that R uses machine learning algorithms which is provided via other packages. Thus, each time we use a function for algorithm in R, we need to install the package. Whereas, scikit learn is dubbed as a unified API which does not require external libraries or packages.

However, this package does not work well with the neural network.

**MLR:**

1Machine Learning in R package provides a generic, object- oriented, and extensible framework for classification, regression, survival analysis and clustering for the R language. It provides a unified interface to more than 160 basic learners and includes meta-algorithms and model selection techniques to improve and extend the functionality of basic learners with, e.g., hyperparameter tuning, feature selection, and ensemble construction. Parallel high-performance computing is natively supported. The package targets practitioners who want to quickly apply machine learning algorithms, as well as researchers who want to implement, benchmark, and compare their new methods in a structured environment.

This is an easy way of tuning hyper parameters which has large range of algorithms.

One of the disadvantages of mlr is, it uses machine learning algorithms via other packages which needs to be installed and if any dependencies are present that as well needs to be installed.

**Caret:**

The caret package (short for \_C\_lassification \_A\_nd \_RE\_gression \_T\_raining) is a set of functions that attempt to streamline the process for creating predictive models.

Caret is more popular than mlr and released earlier compared to mlr. Caret is downloaded a greater number of times compared to mlr.

Caret takes longer time to install compared to the mlr package.5

Caret and mlr both have online tutorials for the beginners however, documentation for mlr is better than caret package. It however has great advantage of covering Applied Predictive Modelling Book, though we have not much documentation in caret it covers good books in its documentation.

Both mlr and caret has the wide range of pre-processing step, however advantage of mlr is, it can also make use of caret methods via wrapper.

Caret only seems to have possibilities for Classification, Regression and Cost-sensitive whereas, mlr has more possibilities to create different tasks such as Classification, Regression, Survival, Clustering, Multilabel, Cost-sensitive, Imbalanced data, Functional data, Spatial data.

Both mlr and caret has the option of tuning the model. For caret, the tuning is done internally and automatically. This can be changed by the user to grid search or random search.

Caret has the possibility to use out-of-bag methods that are available for methods that use bagging like e.g. random forest. This is not available for mlr, only the out-of-bag predictions of a normally trained random forest model can be extracted here.

In general R performance with big data is better as it can perform better results when performing large computations. R offers one-line-of-code solution to manipulating the threshold to account for class imbalances. Python does not have a built-in function for this and is up to the user to programmatically manipulate the threshold by defining their custom scripts/functions. R does not provide standardized interface for any algorithm thus for any experiments which are not so important, we need to write lengthy codes. R is good in visualization and graphics whereas python is more understandable and better is speed comparatively.6

**Comparison of the three models based on different parameters:**

Splitting Dataset: There is an inbuild function in this framework for partitioning of the dataset whereas mlr does not have in built function however caret has createDataPartition function to split data and mlr does not have inbuilt function for the same and needs to rely on R sample function for this.

Resampling Method:

In all the packages we have range of functions to take care of the class imbalance problem.

Pre-Processing:

In scikit learn we have a function to convert our data with polynomial features whereas we do not have the same in case of mlr and caret packages.

Choosing Algorithm: Scikit Learn has more convenience while choosing the algorithm as we just need to put the model name we want to work with.

Example: for Logistic regression – function is LogisticRegression()

If we are training the model using mlr we need to call function makeLearner() and pass our model name as an argument. This function is called to make initialization.

The other function which is called as makeClassifTask() is used for classification problem. Thus, for a beginner to know these functions, becomes a bit problematic.

Example: For random forest - makeLearner('classif.rpart')

For classification using random forest - makeClassifTask(data=, target=) This needs two variables training data and the name of the target variable.

In caret it is more convenient to put the algorithms we want to use as we just need to change the method parameter which is passed in train function. It is more convenient to use different models with caret.

Prediction Output: Scikit learn gives us an array of predicted result whereas mlr gives us dataframe of the predicted result. This is not available in caret or mlr. However, in mlr we can convert our predictions into dataframe. Using as.data.frame() function.

Model Evaluation in scikit learn framework is not as informative as R. It returns the matrix alone with no labels. User needs to find out which column and row represents what. The method classification\_report() gives us information of precision, recall and F1 score which is important metrics for imbalanced problems. The precision, recall and F1 score of scikit learn are weighted which is quite confusing for beginners.

Model Evaluation Metric are more informative for mlr compared to scikit learn as confusion matrix has labels in the output. It gives more detailed values such as False positive rate and false negative rate, kappa and all the values which helps in easy calculation of f1 score, precision and recall.

The model evaluation metric for Caret is also quite informative as mlr package with labels in the output for classification confusion matrix. This also gives us information on f1 score, precision and recall values like scikit learn.

Fine Tuning classifier:

There is no function for fine tuning the classifier in sckit learn framework whereas mlr has this advantage. For scikit learn we have to use paramGrid to set the Grid and use it through cross validation for tuning the parameters. Whereas both caret and mlr has parameter for tuning the model. In caret we can set the tune length and if we want our own tuning values, we can use tuneGrid parameter in train function. In mlr we have function tuneParams and parameter par.set for setting our grid for the hyperparameter tuning in addition to its makeTuneControlRandom() function which does a random search for finding best hyperparameters.

Parallelization:

In the caret package only works internally when applying a tuning strategy like grid search by using the [doParallel](http://topepo.github.io/caret/parallel-processing.html) package which is based on [snow](https://cran.r-project.org/web/packages/snow/index.html). This is not the case with mlr. In mlr we can achieve parallelization at different levels such as while resampling, selecting features, while tuning and also when tuning the ensembles. The parallelization in mlr is done via parallelMap which works well with all major parallelization backends such as parallel, snow and batch tools. In mlr, we have flexibility of choosing what we want to parallelize- for example if we do not want to parallelize benchmark () function but want only the resample method to be parallelized, we can do so by passing the parameter mlr.resample to the parallelStart() function.

In scikit learn we have some builtin single machine multi-core parallelism support using joblib for things like hyperparameters grid search and cross validation. Parallelization during ensembing can be achieved using n\_jobs function in scikit learn.

Visualization:

Visualization of the results and getting information about the models is easy in both the caret and mlr packages. For both the packages we have ROC analysis and calibration curves. For mlr there are additionally partial dependence plots, learning curves, cut-off curves and residual plotting, while caret provides method specific variable importance and lift curves. ([5](http://philipppro.github.io/2018-11-9-mlr_vs_caret/))

Predictions:

Just like training the model predictions can be done with only one line of code in all the three packages mlr, scikit learn and caret.

Changing classification threshold:

Changing classification threshold is very easy in mlr in comparison to scikit learn.

In general,

Large computations are done better in Python compared to mlr or caret. Whereas, R is better in statistical modelling.

Mlr is easy to understand and easy to use for beginners in comparison to python or caret.

References:

1. <https://mlr.mlr-org.com/>
2. <http://jmlr.org/papers/v17/15-066.html>
3. <https://www.kdnuggets.com/2019/09/scikit-learn-mlr-machine-learning.html>
4. <https://datascience-enthusiast.com/R/ML_python_R_part1.html>
5. <http://philipppro.github.io/2018-11-9-mlr_vs_caret/>
6. <https://www.codementor.io/sayantinideb/r-vs-python-best-programming-language-for-data-science-and-analysis-te05xgx98>
7. <http://matthewalanham.com/Students/2018_PURC_caretvsscikit.pdf>
8. <https://blog.exxactcorp.com/scikitlearn-vs-mlr-for-machine-learning/>

**Question 2:**

Assessment Of the report:

1. Sampling of the data has been done for any random 10 working days for each month, they can bring more consistency by doing the sampling on the same day for all the branches. -> **Medium**
2. Response variable and number of predictors are not mentioned clearly in the report. It should be mentioned somewhere in first paragraph clearly as it leaves the reader to calculate the number of predictors based on the variables mentioned. Could have indicated calculation showing (number of observation)/ (number of predictors). This will tell us whether modelling is possible on the given dataset. The response variable used in models is Net\_Productivity which has been calculated, but it’s not mentioned in the report, if this is the response variable. By looking at the formula of the model, we can understand that Net\_Productivity is the response variable. -> **High**
3. It is mentioned anything we achieve must be improvement- Any model we create must be compared with the null model. If a model is not able to perform better than the null model then we cannot call it is as improvement – **Medium**
4. Could have checked and reported if there are any unusual patterns in the unique identifiers. -> **Low**
5. All the strings are changed into factor according to the report, but the three unique identifiers might be of character, strings or text format but should not be considered as factor as it has too many levels. Just converting Strings to factors is not a right approach, need to check the number of levels for the factors. -> **High**
6. Outliers need to be checked for all numeric values. Here only Non-sales column has been checked for outliers. Outliers were present in non-sales value but it does not mention the interquartile range which was kept for the outliers and should have checked by increasing the range a bit. There should have been discussion about the reason for novelties and if these can occur in future. The statistical argument for removing outliers (based on IQR = 1.5) is not strong even if the assumption about Gaussian Distributions proves to be correct – not checked. Could have checked YeoJohnson transformations and check IQR multipliers first. -> **High**
7. The argument for removing outliers assumes that outliers will not be present in future unseen data – There is no evidence for that. - > **High**
8. Outliers (if removed) should be done automatically so it is repeatable. Report did not mention whether the outliers were removed automatically or not. -> **High**
9. If the outlier removal is the strategy chosen in this condition, then there should be two models. One with outliers and one without outliers, which will give more clear idea whether these outliers are affecting our model. This has not been written in the report. -> **High**
10. The report has not mentioned if these outliers can occur in future. If it can occur in future then, we should use all the data and try robust method. If not then we can omit the outliers and make two models- one with outlier and one without. -> **High**
11. No investigation into observations with excessive missing values -> **High**
12. No investigation into variables with excessive missing values -> **High**
13. Imputation was done on the whole dataset before test train split, this might lead to data leakage **-> High**
14. As we cannot assume that the unseen dataset will not have missing values, it should be discussed in the report how they are taking care of it. Based on the mean/ mode of the current dataset? Could make use of imputation method such as bagimpute or knnimpute to check the performance of the model. -> **Medium**
15. The imputation was undertaken without considering whether the data is MCAR/MAR. Should also discuss if data will be missed in future. -> **Medium**
16. The report does not mention why mean/mode imputation is done for this dataset. If the whole row or column is missing in the dataset then imputation might not help and these rows or column might have to be dropped off. As there are huge number of observations and if omitting the missing values give better model there should be two models, one with omitting and one without. If omitting the rows, should take care of future missing values by imputation. -> **High**
17. Need to plot a correlation plot, in order to see if certain columns are highly correlated and should be written in report. Highly correlated values should be removed depending upon the model. Not dropping the correlated variables will not improve the model instead can cause unnecessary computation cost. -> **Medium**
18. There is no discussion about the homogeneity of the dataset. Need to check if there is pattern to the data in the supplied order. Report should have mentioned if there are few variables which lies significantly away from rest of the datapoints which can be viewed by changing all the variables into 2 dimension and plotting it. -> **Low**
19. There is no discussion of the summary statistics of the dataset. Mean and median for numeric variables and mode for categorical variables gives us overview of the dataset. -> **Medium**
20. There are only four levels of hair colour mentioned in the dataset, but it might happen that there is additional level for hair colour thus it might be useful to coalesce levels of factor predictors with low frequency into a new level called “other”. -> **Medium**
21. Manual method has been applied for feature engineering, can also try using filters or discuss on why this step was chosen for feature engineering. -> **Low**
22. In the feature engineering section, it has been mentioned that sales value and non-sales value are used for derivation of Net\_productivity, but in feature selection these values are being removed. For interpretability if combination of sales value and non-sales value is important, we should not throw away sales value and non-sales values even if they might not be important. -> **Medium**
23. Imputation has been done before outlier detection. If the outliers are being removed in this case, we are introducing missing values. Thus, we should deal with the missing values after outliers are removed again which has not been mentioned in the report. -> **High**
24. Doing imputation before train/test split leaks data between the two. -> **Medium**
25. For Feature selection the report does not tell anything about what method was used for doing so. Explanation is required for feature selection. Maybe it is manual process, in that case needs to be automated. However random forest and penalised GLM does not need feature selection as this is done implicitly in these methods. -> **Medium**
26. A train/test split that is supervised (i.e. stratified) will be better than a random one as we are taking samples from different countries. -> **High**
27. Repeats mentioned for the cross validation for each of the model is different. KNN has 3 cross validation folds whereas penalized GLM has 5 cross validation and random forest has 10 cross validation folds for finding the best parameters. Should mention why these parameters are different -> **Low**
28. No discussion on why KNN, penalised GLM and Random forest has been chosen as candidate model. -> **High**
29. No discussion on Hyper parameter tuning. Could have shown the best hyper parameter for relevant models. -> **Medium**
30. Training the model and getting lowest RMSE on train data does not prove that the best model has been found. It is necessary to test our model in unseen data. -> **High**
31. Cannot deploy the model without testing if the model is fit for purpose. -> **High**
32. Monitoring of the model is required, which is not mentioned in the report. We cannot expect the model to be the best until we monitor the model and see if it is actually producing some valid results. -> **Medium**
33. The report in short has not managed to consider the model life cycle and it has been assumed that the model produced is the best model which will never fail and is perfect for prediction which is not correct to assume. -> **High**
34. Standardization is done before we are doing test train split, which is not recommendable. -> **Medium**

**Ethical Issues:**

1. It has not been mentioned anywhere in the report if the employees are aware that their sensitive information is being used. -> **High**
2. Hair colour/ employee religion are one of the variables which is an ethical concern as if we are using these variables, we are predicting based on religion which is not ethical. -> **High**