**Lazy Panda project,**

**integrated functions for data preprocessing**

**in Machine Learning with R**

***Introduction:***

As a data scientist living in a fast paced technological industry, you are required to manipulate things really fast, otherwise, you will be left behind. We cannot deny the fact that nobody want to be a sloth at work, especially in tech field, where we ought to quickly write a bunch of complex syntax along with a million of commands, functions, algorithms, ect. This is the most time consuming and headache task for any data scientist. With those challenges in mind, a “lazy” code writer like me, came up with an ideal solution for other “lazy” data scientist, called Lazy Panda project.

So, what will Lazy Panda project can assist you in data preprocessing? At this moment, Lazy Panda project can help you in cleansing up your dataset, visualizing data in general, transformating data, preparing data for modeling, setting up machine learning models, and evaluating results upon request. Common algorithms such as Regression, Naïve Bayes, Tree based, SVM and Artificial Neural Network are used in packages to perform supervised learnings either in classification or numerical prediction. Basically, your task is now curtailed to merely input raw dataset together with some initial conditions. The entire part of heavy work will be processed by functions.

Exciting? alright, so let take a look at functions in details

**1. Input (things you have to manually modify) :**

* Target variable location:

You will need to locate which column in your train and test data contains Target variable. After doing this, everything has been set up and ready for processing.

* Input dataset

All you need to do is to prepare your raw dataset. So, in R script, you may want to change the name of input data with a .csv extension. If your data only have one .csv file, you have to specify the split.need variable to TRUE with a desired ratio. Thus, the project can help you split data into train and test data for modeling. Otherwise, you can directly load train and test data with corresponding .csv files if you have already those two files in hand. Dataset should be placed in the same directory for successful import

* Initial conditions

We have initial conditions here just in case that your data contains ordinal features. If that will be the case then you just simply set the ord.var to TRUE, specify which column(s) your train and test data contain ordinal features, and list down all those ordinal variables in ascending order.

* Target Variable class define:

Finally, you need to define suitable training type for Target variable. Logical variable classify will be TRUE if Target variable is suitable for classification (categorical data). Otherwise, classify will be FALSE and thus Target variable will be used for numerical prediction (numerical data)

**2. Visualization function:**

This function will return:

* Data summary

Summary training and testing data using summary() in R, summary result will be returned after running visualization function

* Histogram of target variable (if target variable is numerical data)

Function will return the histogram of numeric target variable

* Visualization and number of any missing value in training and testing data

Using packages reshape2, dplyr, and ggplot2 to visualize missing value in training and testing data. The plot will map missing value in both dataset. The function will also return the count of missing value in each column of data.

* Correlation matrix of numeric variables

Build up a correlation matrix among numeric variables in training dataset

* Scatter plot to visualize the relationship between features

**3. Cleanup function:**

For cleanup function, it will use sub functions to imputate missing values (if dataset has any missing value), arrange column data, normalize data, and encode dataset

* missing values imputation (miss.imputate() function):

First, this function will detect any missing value in training and testing data. If any missing value has been detected, it will convert that missing value into NA.

Then, the function will imputate all missing values by using missForest() with missForest package, and assign dataset into a new data.frame variable

* data rearrangement (rearrange() function):

The purpose of this function is to rearrange data into a fixed format for data modeling. Particularly, it take input training and testing data as parameters, reallocate columns based on their attribute class. The function is going to change the name of Target variable column into “Target” for convenience.

* data normalization (normf() function):

After arranging datasets, normf() function will normalize data using min-max method. Algorithms are going to used normalized dataset include: Naïve Bayes, SVM, and ANN

* data encoding (feature.encode() function):

Algorithms use encoded data include: Tree based, SVM and ANN. The feature.encode() function will consider encoding data under 2 circumstances:

* If dataset contains any ordinal features, arbitrary encode will be performed. Basically, the function will continue looping into each ordinal column and alternate one by one its values with corresponding representative numbers which you defined earlier in the input step. Otherwise, the function will ignore ordinal encoding and move to one hot encoding stage.
* In one hot encoding, all categorical features will be encoded using onehot() of onehot package. The function assign new datasets into new variables and return them into list.

After all, the function will return 4 lists which corresponding to 4 types of algorithm training: Regression; Tree based; Naïve Bayes; SVM and ANN

**4. Regression algorithm (regress() function):**

The regress() function will take train data and test data as input parameter and consider modeling input data under 3 various cases:

* If input training data has non-numeric Target Variable with 2 levels, the regress() function will build up a Logistic Regression Model for classification. It will use glm() for training dataset, consider significant independent variables (alpha = 0.005) base on anova result and perform regression again for training dataset with selected significant predictors only. The final result will be returned as model training accuracy.
* If input training data has non-numeric Target Variable with more than 2 levels, the regress() function will build up a Multinomial Regression Model for classification. The final result will be returned as model training accuracy.
* If input training data has numeric Target Variable, the regress() function will regress training data with Linear Regression. The final result will be returned as the correlation between predicted value from model and actual value from testing data

**5. Tree based algorithm (dtree() function):**

The dtree() function will training the data base on class of Target Variable:

* If Target Variable is suitable for classification, Decision Tree is going to be used to classify training data. C5.0() function from C50 package will be applied to create classifier. The function then calculates and returns the accuracy of model.
* If Target Variable is suitable for numerical prediction, CART regression tree will be applied. The training data will be trained using rpart(), the function then calculate the correlation between predicted and actual values. Final result will include correlation and Mean Absolute Error (MAE).

**6. Naïve Bayes algorithm (nb() function):**

The nb() function will classify training dataset using naiveBayes() function from e1071 package. The function then compute and return the accuracy of modeling

**7. Support Vector Machine SVM algorithm (suvema() function):**

Similar to other algorithms, suvema() function will train the input data based on class of Target Variable:

* If the class of Target Variable is factor (suitable for classification), svm() function from e1071 package will be applied to create classifier. Model accuracy will then be calculated and returned as final result
* If the class of Target Variable is suitable for numerical prediction, tune.svm() will be applied. The function will train data under 4 different kernals: Linear, Polynomial, Radial, and Sigmoid. For each kernal, best performance will be recorded and MSE (Mean Squared Error) will be computed. The function then compare amongs 4 kernals’ results to select the best kernal with the smallest MSE. Finally, it will return the best kernal with its correlation.

**8. Artificial Neural Network ANN (ann() function):**

The ann() function will take train and test dataset as input parameter. It will consider training Target Variable based on Target’s class:

* If Target Variable is suitable for classification, h2o package will be used. The function will initiate 8-node h2o server on local machine. Next, it will automatically split train data for validation with ratio of 2:8. h2o.deeplearning() will train dataset with “Tanh” activation function and 2 hidden layers for 200 nodes in each layer. The function then predict the model with h2o.predict() function. Finally, the function compute and return the accuracy of the model.
* If Target Variable is suitable for numerical prediction, neuralnet() function from neuralnet package will be used. The function will train data under 2 different activation functions: Sigmoid or Logistic activation function and Tanh or Hyperbolic Tangent activation function. For each activation function selected, best performance will be recorded and MSE (Mean Squared Error) will be computed. The function then compare between 2 activation functions’ results to find the best performance with the smallest MSE. Finally, it will return the best activation function with its correlation.

**Appendix:**

All packages need to be install and load with library() before using project:

***Visualization:*** reshape2; dplyr; ggplot2; psych

***Missing Values Imputation:*** missForest

***Feature Encode:*** onehot

***Regression:*** nnet

***Tree based:*** gmodels; C50; rpart

***Naïve Bayes:*** gmodels; e1071

***Support Vector Machine (SVM):*** gmodels; e1071

***Artificial Neural Network (ANN):*** gmodels; h2o; neuralnet