The package: MegaLearn(dataset,response_variable)

The package is capable of running a classification as well as a regression problem.

In order to run the package, you need to provide two arguments as input.

1. Dataset
2. Response_Variable

Caution: You need to make sure that all the data types are in the right format including the response variable. All the categorical variables should already be tagged as factor variables before you run the package.

You need to install the following packages installed in order to run the package

install.packages("caret")
install.packages ("glmnet")
install.packages ("dummies")
install.packages ("devtools")
install.packages ("MASS")
install.packages ("gbm")
install.packages ("e1071")
install.packages ("rpart")
install.packages("kernlab")
install.packages("miscTools")

The package doesn't do any specific missing value imputation. It simply omits all the missing values. In order to build the package, I divided a generic problem statement into two types. Either it can be classification problem or a regression problem.

The package starts with 4 functions.

1. Evaluate – This function takes confusion matrix as a input and calculates accuracy, precision, recall and f1_measure. I am however only reporting the accuracy metric.
2. Acc – This function returns the accuracy for a given model input and predictor and dependent label. It uses the evaluate function mentioned above to return accuracy. Using this function, I can return accuracy for training, testing, or in the context of cross validation.
3. R2 – This function returns the r-square coefficient for a given model input and x and y label.
4. MSE - This function returns the Mean Square Error for a given model input and x and y label.

- **Data Preparation**

1. I separate the categorical variables and the numeric variables. The categorical variables are then dummy coded and eventually binded with the numerical variables.
2. The next step involves data partition into training and testing set. I have considered 20% as the benchmark for testing dataset and 80% for the training data.

- **Modeling Part:**

1. Based on the type of response variable, the function either enters a regression framework or a classification framework. I have used some different algoirthms based on the problem type.

2. In the case of regression, I use the following algorithms to develop my model

    a. Linear Regression
    b. Random Forest
    c. Regression Tree
    d. Ridge Regression
    e. Lasso Regression
    f. Neural Net
    g. PCA_Regression
    h. Weighted_Linear_Regression

3. Across all the regression based methods presented above, I am presenting the following metrics

    a. Training R-square
    b. Training Mean Square Error
    c. Testing R-square
    d. Testing Mean Square Error
    e. Cross Validation R-square
    f. Cross Validation Mean Square Error

4. In the case of classification, I am running the following classification algorithms

    a. GLMNET
    b. Random Forest
    c. Support Vector Machine
    d. Naïve Bayes
    e. CART
    f. Neural Net

g. Penalized SVM
h. Adaboost
i. LDA-PCA
j. Weighted_Glmnet
k. Weighted_SVM

5. Across all the classification based methods presented above, I am presenting the following metrics

   a. Training Accuracy
   b. Testing Accuracy
   c. Cross Validation Accuracy

In order to test the model, I used two datasets. One of the datasets is Loan.csv. This dataset has a binary dependent variable. So, this was a classification problem.

However, before running it, I converted the categorical variables into factor.

The following variables were converted from numeric to factor, since they represented categorical variables.

- CAR - Owns a car 1:no 2:yes
- RACE - Chinese/Non Chinese
- GENDER - M/F
- OFFICER - One of four loan officers working for the bank
- RESPONSE – The dependent variable

- Remove ID variable

Preprocessing steps to follow to make structure of data consistent with function.

```
>mydata <- read.csv("loan.csv", header=TRUE, sep=",")
>mydata <- mydata[,-1]
>mydata$CAR <- as.factor(mydata$CAR)
>mydata$RACE <- as.factor(mydata$RACE)
>mydata$GENDER <- as.factor(mydata$GENDER)
>mydata$OFFICER <- as.factor(mydata$OFFICER)
>mydata$RESPONSE <- as.factor(mydata$RESPONSE)
>str(mydata)
```

Following are the results for running the package on the loan.csv dataset. The command given is **megalearn(mydata, "RESPONSE")**

	GLMNET	Random Forest	SVM	Neural_Net	Penalized_SVM	Naive_Bayes
Training_Accuracy	0.7252033	0.9658537	0.7512195	0.7203252	0.7951220	0.7398374
Test_Accuracy	0.7320261	0.7516340	0.7647059	0.7058824	0.7189542	0.6993464
Cross_Validated_Accuracy	0.6666667	0.7304662	0.7061801	0.7296629	0.7238903	0.7238903
	CART	Adaboost	PCA_LDA	Weighted_GLMnet	Weighted_SVM	
Training_Accuracy	0.7382114	0.8710938	0.7330729	0.7268293	1.0000000	
Test_Accuracy	0.7124183	0.6718750	0.7450980	0.7189542	0.6470588	
Cross_Validated_Accuracy	0.7056911	0.7239583	0.7278646	0.6470588	0.7080504	

- REGRESSION PROBLEM EXAMPLE

To test the package for predicting a continuous variable, I used the 'mtcars' dataset. In this dataset, I again had to convert two variables into factors.

-vs
-am

Following are the results for the mtcars dataset.

	Linear_Regression	Random Forest	Regression_Tree	Ridge	Lasso	Neural_Net	PCA_Regression	Weighted_Linear_Regression
Training_R-Square	0.8800395	0.95793685	0.7819170	0.8705825	0.8060768	0.92347472	0.8439636	0.8800395
Training_MSE	0.1165525	0.04086819	0.2118875	0.1257409	0.1884141	0.07435129	0.1516036	4.2336663
Testing_R_Square	0.6514858	0.63906430	-0.1764695	0.6737828	0.7660447	0.57450929	0.6510723	0.6514858
Testing_MSE	0.3171966	0.32850183	1.0707514	0.2969032	0.2129319	0.38725590	0.3175729	11.5218807
Cross_Validated_MSE	0.4209811	0.18661942	0.4697970	0.2761291	0.5869406	0.26061611	0.1508864	13.0766874
Cross_Validated_R_Square	0.8684781	0.91262171	0.7864579	0.8565203	0.8675925	0.87448376	0.9293804	0.8214358

THE STEPS FOLLOWED TO DEVELOP THE FUNCTION:

1. Install all packages

2. Define functions – evaluate, acc, r2, mse

3. Prepare Data

   a. Make response variable the last column
   b. Standardize data (mydata)
   c. Separate categorical and continuous predictors
   d. Dummy code categorical variables
   e. Bind the dummy coded categorical, numeric and response variables into a binded dataset.
   f. Create Training index for data split – train and test dataset.
   g. Create train_x, train_y, x_test, y_test. These matrices contains the categorical variables as factors
   h. Create binded_x, binded_y, binded_x_test, binded_y_test. These matrices contains the categorical variables as dummy variables. So this has additional dimensionality.

4. Now to the modeling part. Check if it's a regression problem or a classification problem

5. For a regression problem

   a. Create a matrix table that will store all the performance results
   b. For each alogorithm, fit the model, calculate training and testing R-square and MSE.
   c. For each algorithm, also calculate the k-fold cross validation R-square and MSE.
   d. The PCA is applied to the binded dataset that contains categorical variables dummy coded. For PCA, I use the number of principal components that explains 80% of the data variance.

6. For the weighted_method in case of regression, I use the following steps

a. I use the non-standardized copy of the mydata to separate it into numeric and categorical predictors.
b. I use the Anova test(aov) for calculating the p-values between categorical predictors and continuous dependent variable.
c. I use the Correlation test(cor.test) for calculating the p-values between continuous predictors and continuous dependent variable.
d. Calculate weights using q-value. I use the following formula to calculate weights vector separately for categorical and continuous predictors.

log(pvalue_vector*length(pvalue_vector)/rank(pvalue_vector))

e. Create a weighted bind for the modeling purpose. I run linear regression on the weighted dataset to calculate the above mentioned performance metrics.

7. For a classification problem

e. Create a matrix table that will store all the performance results
f. For each algorithm, fit the model, calculate training and testing accuracy.
g. For each algorithm, also calculate the k-fold cross validation accuracy.

8. For the classification problem, I calculated the p-values for the weighted method using following tests.

a. Fisher test for categorical predictors vs categorical response_variable
b. Anova test for continuous predictors vs categorical response_variable.
c. Then test the glmnet and SVM model on the weighted dataset. I keep the standardize option = FALSE for them so that the weights do not lose their relevance.

The package will finally return a classification performance matrix or regression performance matrix.