Introduction to Data Science (BGU course)
Assignment 3

Instructions

Submission is in pairs. Please submit a PDF file, and use R Markdown.

Answer on 4 of the following questions.

Answer the questions using brief explanations and code, and clear figures. If your code is not straightforward, add minimal annotations using comments in the code itself.

Do not hesitate to search for answers online. People asked similar questions all the time.

Before any operation that includes a random component (e.g. generating random numbers, random sampling, etc.) use `set.seed(1)`.

Note: the grade is based also on the elegance of the code. Try to write clear, concise and short code.

Question 1

download the "train.csv" file from kaggle's house price Competition and read about the variables in the Data Description section. Answer the following questions:

a. Create new dataset named `d_clean` which contain only columns with non-NA values from "train.csv".
b. load the `caret` package. use the `nearZeroVar` function (check `?nearZeroVar`) to identify features with very low variability. These variables are usually not informative and many time it is recommended to remove them in prediction problems. Remove these columns from `d_clean`.
c. Split the dataset randomly into 2 sets: the training set, named `d_train` (70% of `d_clean`) and your validation set, named `d_validation` (the remaining).
d. Provide a short EDA of the d_train data: Present 3 interesting/unexpected relations.
e. Train a linear model (use `lm()`) on `d_train` to predict the property's sale price in dollars (`SalePrice`). You can chose any combination of features that you want. Include interactions and discretized continuous variables as well.
f. What is the root mean squared error (RMSE) on your training set?
g. Use your model to predict `SalePrice` on the validation set. What is the RMSE on your validation set? Explain why they differ.
h. Now use step-wise regression using `step()` and repeat sections c and d. Explain the differences in your results. (use the argument `trace=0` in `step` call). Why step-wise regression will not necessarily return an optimal model? Why it might be problematic in big data?

Question 2

In this question we will train an SVM model for classification.

a. Use `d_clean` from question 1. Replace `SalePrice` variable by a factor that gets "high" when `SalePrice` is higher than the average `SalePrice` in `d_clean`, and "low" otherwise. (make sure it is of class factor). This time split it randomly into training set (60%), validation set (20%) and test set (20%), name these: `d_trn`, `d_val`, `d_tst`.
b. Fit an SVM model with linear kernel (`kernel='linear'`) on the training set predicting `SalePrice` (which is a binary factor now) using any features you want. Don't specify the cost hyperparameter. Use your model to predict `SalePrice` on the test set. present a confusion matrix, accuracy, recall and precision.

c. You will now use the training and the validation sets to tune an SVM model by choosing the cost hyperparameter with the best performance:
    1. create a vector of possible costs to chose among them i.e. `seq(0.01,20,len = 30)`
    2. use `for` loop: for each cost fit an SVM model by specifying `cost =` in the `svm()` call.
    3. for each cost compute the training error (average correct classification rate) and the validation error.
    4. Plot both the training and validation errors aginst the cost values. Describe this plot.
    5. Choose the cost that resulted in the best performance on the validation set. Mark it on the plot.

d. Use your model and predict the test set. Reporte your performance: present the confusion matrix, accuracy, recall and precision. Explain the differences from section b

Question 3

In this question we will fit a Random-forest algorithm from the the `randomForest` package, and practice the procedure of cross validation. We use the `diamonds` dataset and consider the prediction of the `price`. (take a subset from `diamonds` to facilitate computation efort; 5000 random samples is enough) Note that some features are in text format, and we need to encode them to numerical.

1. Split your dataset into 10 equal-size folds.
2. Estimate the performance of a randomForest algorithm using Cross validation:
    a. Use `for` loop, in each round:
        1. Determine the train and the validation sets. Choose any features combination that you want, and use default `randomForest` parameters.
        2. Train the model on the training set
        3. Validate the model on the validation set, use RMSE measure.
    b. aggregate the results
3. Repeat section 2, this time change any default `randomForest` parameters you want. In particular, change `mtry` and `ntree`. Try to improve your predictions (the cross-validated performance) from section 2. Compare algorithms' performances.
4. Explain the `mtry` and `ntree` hyperparameter. Why use them? How do you suggest to choose them?

5. Another cross validation estimation procedure is "Leave-One-out CV". This is a K-fold cross validation where K = number of samples. Give one advantage and one disadvantage of the Leave-One-out CV.

Question 4

1. Use the `iris` dataset
    a. Split `iris` randomly to 70% training set and 30% test set. Train a decision tree on the training set to predict `Species`. Use `rpart` package.
    b. Use `rpart.plot` package to plot your model. Explain this plot precisely.
    c. Predict with your model the class of the test set data, present your performance (confusion matrix and accuracy).
    d. In which prediction tasks you would consider to fit a decision tree? In which tasks not?

2. The Gradient Boosting is a family of some very powerful machine learning models. Roughly speaking, these models are an ensemble of weak prediction models, typically decision trees. It is recommended to watch this youtube video on Gradient Boosting. The **XGBoost** provides a gradient boosting framework for R and is considered as one of the most successful prediction models. Use the training set from section 1 to train an xgboost model that classify wheter a flower is setosa or not (binary classification). Present the performance on the test. Explain your steps.
3. Bonus: tune your xgboost model's hyperparameters with packages such as `caret`, or `mlr`.

link: https://www.youtube.com/watch?v=wPqtzj5VZus

Question 5

Glmnet is a package that fits a generalized linear model with complexity penalty (regularization). It fits linear, logistic and multinomial, poisson, and Cox regression models. `glmnet` solves the following problem:

$$\arg\min_{\beta_0,\beta} \sum_{i=1}^{N} w_i L(y_i, \beta_0 + \beta^T x_i) + \lambda\big[(1-\alpha)||\beta||_2^2 + \alpha||\beta||_1\big]$$

1. Explain the following components in the above formula:

- L
- λ
- $||\beta||_2^2$
- $||\beta||_1$
- α

Run:

```
library(ggplot2)
library(data.table)
library(magrittr)
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-18
```

```
d <- diamonds[1:10000,]
X_trn <- model.matrix(price~.-1, data=d[1:6000,]) %>% scale()
X_tst <- model.matrix(price~.-1, data=d[6001:10000,]) %>% scale()
y_trn <- scale(d$price[1:6000])
y_tst <- scale(d$price[6001:10000])
```

2. Train a glmnet model on `X_trn` and `y_trn` to predict `y_tst` with `X_tst` using:

a. Ridge penalty (named `glmn_ridge`);
b. Lasso penalty (named `glmn_lasso`). Set all other parameters to their default.

3. Explain the following difference between the ridge and lasso models' coefficients:

```
cbind(coef(glmn_ridge, s = 0.01), coef(glmn_lasso, s = 0.01))
```

```
## 25 x 2 sparse Matrix of class "dgCMatrix"
##                         1             1
## (Intercept)   3.716759e-15  2.244527e-15
## carat         2.019635e-01  .
## cutFair      -5.756890e-02 -3.221156e-02
## cutGood      -6.925643e-03  .
## cutVery Good  4.102803e-03  .
## cutPremium   -1.958177e-02  .
## cutIdeal      4.615485e-02  2.735089e-02
## color.L      -1.831673e-01 -1.783942e-01
## color.Q      -2.077644e-02 -1.028403e-02
## color.C       2.934195e-03  .
## color^4       1.296455e-02  4.582382e-03
## color^5      -6.206214e-03 -4.411516e-05
## color^6       1.009079e-02  9.405453e-04
```

```
## clarity.L      2.894125e-01  3.024729e-01
## clarity.Q     -9.136226e-02 -7.046140e-02
## clarity.C      4.552440e-02  4.326517e-02
## clarity^4     -5.957098e-02 -5.550650e-02
## clarity^5      1.829414e-02  1.195471e-02
## clarity^6     -7.071043e-03  .
## clarity^7      4.490655e-03  .
## depth          3.556697e-02  8.260729e-02
## table          1.224596e-02  .
## x              3.037485e-01  4.167267e-02
## y              3.517225e-01  9.452276e-01
## z              1.913878e-01  6.500350e-02
```

3. The function glmnet returns a sequence of models for the users to choose from. How do you suggest to choose which to use?
4. Run a lasso model, and choose the best fit with cross validation. use `cv.glmnet()` function. call the model `cvfit`. Run `plot(cvfit)` and explain this plot. What are the vertical lines in this plot.
5. When you predict with glmnet model, you need to specify the value of λ (add `s=` argument in predict). Predict `X_tst` with λ that has minimum cross-validated MSE. Report your performance.
6. Run a model with many more interactions variables and use `cv.glmnet`. What is the MSE on the test?
7. Use `glmnet` to train a model with `X_trn` and `y_trn_binar=y_trn>0`, to predict `y_tst_binar=y_tst>0`. Use `lambda=0.1`. Provide confusion matrix. what is your accuracy?
8. Use `glmnet` to train a model with input `d[1:6000,]`, to predict the **color** `d$color[6001:10000]` with only the input `d[6001:10000,-3]` (in which `color` is removed). use `lambda=0.01`, `alpha=0.5`. Provide a confusion matrix. What colors can be more easily classified with your model? Which pairs of colors do you tend to confuse?