**Name:** ______________________ **ID:** ______________________

This lab is a quick and short introduction on using R for

- logistic
- KNN
- K-means
- Hierarchical
- Association analysis

Instructions:

- Answer questions on this sheet when a box is provided.
- If R code is asked, it shall be written in a single script file.
- Name the script file `lab2_[your_id].R`. For example, `lab2_5123700044.R`.
- Separate your code into sections according to `task_id` and `part_id`. For example,

```
# Task 1 part (a) ------------------------------------------------------------
tmp.df = data.frame( x = rnorm(10), y = rbinom(10, size = 1, prob = 0.3))

# Task 1 part (b) ------------------------------------------------------------
stem(tmp.df$x)

# Task 1 part (c) ------------------------------------------------------------
my.func = function(x){t.test(x[[1]]~x[[2]])}

my.func(tmp.df)

.
.
.

# Task 2 part (a) ------------------------------------------------------------
cut(tmp.df$x, breaks = seq(-3, 3, length.out = 5))

rm(list = ls())
```

**Task 1** (7 points)
This task will use `Smarket` data, which is part of of `ISLR` library. This dataset consists of percentage returns for the S&P 500 stock index over 1,250 days, from the beginning of 2001 until the end of 2005. For each date, it recorded the percentage returns for each of the five previous trading days, `Lag1` through `Lag5`. It also recorded `Volume` (the number of shares traded on the previous day, in billions), `Today` (the percentage return on the date in question) and `Direction` (whether the market was `Up` or `Down` on this date).

```
> # install.packages("ISLR")
> library(ISLR)
> str(Smarket)
```

```
'data.frame':   1250 obs. of  9 variables:
 $ Year     : num  2001 2001 2001 2001 2001 ...
 $ Lag1     : num  0.381 0.959 1.032 -0.623 0.614 ...
 $ Lag2     : num  -0.192 0.381 0.959 1.032 -0.623 ...
 $ Lag3     : num  -2.624 -0.192 0.381 0.959 1.032 ...
 $ Lag4     : num  -1.055 -2.624 -0.192 0.381 0.959 ...
 $ Lag5     : num  5.01 -1.055 -2.624 -0.192 0.381 ...
 $ Volume   : num  1.19 1.3 1.41 1.28 1.21 ...
 $ Today    : num  0.959 1.032 -0.623 0.614 0.213 ...
 $ Direction: Factor w/ 2 levels "Down","Up": 2 2 1 2 2 2 1 2 2 2 ...
```

(a) (1 point) Run and study both of the followings. Notice what the error message is for.

```
> cor(Smarket)
> cor(Smarket[, -9])
```

Why is it not meaningful to compute the Pearson correlation coefficient of a factor and numeric variables despite of having been assigned an integer value for each level.

The following is often useful to study correlation, especially when comes to a large number of predictors/features. Having a colour map can be useful to visualise the relationships.

```
> # install.packages("corrplot")
> library("corrplot")
> corrplot(cor(Smarket[, -9]), method = "ellipse", type = "upper", diag = FALSE)
```

We can fit a logistic regression model in order to predict `Direction` using `Lag1` through `Lag5` and `Volume`. The `glm()` function fits generalized linear models, a class of models that includes logistic regression. The syntax of the `glm()` function is similar to that of `lm()`, except that we must pass in the argument `family=binomial` in order to tell **R** to run a logistic regression rather than some other type of generalized linear model.

(b) (1 point) Run and study the followings.

```
> s.GLM = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
+               data = Smarket, family = binomial)
> summary(s.GLM)
```

Comment on the relationship between `Lag1` and `Direction`, and the statistical significance of this relationship.

The `predict()` function can be used to predict the probability that the market will go up, given values of the predictors. The `type="response"` option tells **R** to output probabilities of the form $\Pr(Y = 1 \mid X)$, as opposed to other forms such as the logit.

```
> pred.df = data.frame(Year = 2001, Lag1 = 0.5, Lag2 = 0.5, Lag3 = 0.5,
+                      Lag4 = 0.5, Lag5 = 0.5, Volume = 1, Today = 0.6)
> predict(s.GLM, pred.df, type = "response")
```

```
        1
0.4917836
```

If no data set is supplied to the `predict()` function, then the probabilities are computed for the training data that was used to fit the logistic regression model.

```
> head({probs.training = predict(s.GLM, type = "response")})
```

```
        1         2         3         4         5         6
0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565
```

(c) (2 points) Suppose we base the classification on whether the predicted probability of a market increase is greater than or less than 0.5. Write a few R statements to find the proportion of correctly classified observations for this training set, and state it below.

If you take the number you find in part (c) away from 100, you will have the training error rate. It is usually overly optimistic, it tends to smaller than the test error rate. So to judge the quality of a classifier, we need to divide the data set into training and test sets.

```
> training = (Smarket$Year < 2005)
> Smarket.test = Smarket[!training, ]
> dim(Smarket); dim(Smarket.test)
```

```
[1] 1250    9
[1] 252   9
```

(d) (2 points) Run the same logistic regression model but on the training set using the `subset` argument in `glm()`. Find and report the test error rate. Comment on this rate.

KNN can be performed easily using the `knn()` function, which is part of the `class` library. The function requires four inputs:

- A matrix containing the predictors associated with the training data, `train.X` below.
- A matrix containing the predictors associated with the data for which we wish to make predictions, `test.X` below.
- A vector containing the class for the training observations, `train.Direction` below.
- A value of $K$, the number of nearest neighbours to be used by the classier.

```
> library(class)
> train.X = cbind(Smarket$Lag1, Smarket$Lag2)[training,]
> test.X = cbind(Smarket$Lag1, Smarket$Lag2)[!training,]
> train.Direction = Smarket$Direction[training]
>
> set.seed(1)
> knn.pred = knn(train = train.X, test = test.X, cl = train.Direction, k = 3)
> table(knn.pred, Smarket$Direction[!training])
```

```
knn.pred Down Up
    Down   48 55
    Up     63 86
```

(e) (1 point) Find the test error rate in this case.

**Task 2** (10 points)

This task uses the `Weekly` dataset, which is part of the `ISLR` package. This data is similar in nature to the `Smarket` data, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

(a) (2 points) Produce some numerical and graphical summaries of the `Weekly` data. Do there appear to be any patterns?

(b) (2 points) Use the full data set to perform a logistic regression with `Direction` as the response and the five lag variables plus `Volume` as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

(c) (2 points) Now fit the logistic regression model using a training data period from 1990 to 2008, with `Lag2` as the only predictor. Compute the overall fraction of correct predictions for the test data period, that is, the data from 2009 to 2010.

(d) (2 points) Repeat (c) using KNN with $K = 1$.

(e) (2 points) Experiment both logistics regression and KNN with different predictors, include possible transformations and interactions. Note that you should also experiment with values for $K$ in the KNN classifier.

**Task 3** (13 points)

This task will use a simple simulated data to study the function `kmeans()` and `hclust()`, which performs $K$-means and hierarchical clustering in **R**, respectively.

```
> set.seed(2)
> x = matrix(rnorm(50*2), ncol = 2)
> x[1:25, 1] = x[1:25, 1] + 3
> x[1:25, 2] = x[1:25, 2] - 4
```

(a) (1 point) Given we know how the data is simulated, suggest a way to assign those observations into 2 clusters.

(b) (1 point) Use `kmeans()` with `centers = 2` and `nstart = 20` on this simulated data.

(c) (1 point) Plot the result of (b) using two colours according to its cluster assignment.

(d) (1 point) Repeat (b) with `centers = 3`, notice the difference between (b).

(e) (2 points) Use `hclust()` and `dist()` to perform hierarchical clustering on this simulated data with Euclidean distance as the dissimilarity measure and complete linkage.

(f) (1 point) Use `plot` on the object created by `hclust()` to produce a dendrogram.

(g) (2 points) Study `cutree()`, and illustrate its usage on the result you have for (e).

(h) (2 points) Study `scale()`, and repeat (e) but to the scaled features/variables.

Correlation-based distance can be computed using the `as.dist()` function, which converts an arbitrary square symmetric matrix into a form that the `hclust()` function recognises as a distance matrix.

```
> dd = as.dist(1-cor(t(x)))
```

(i) (2 points) Run the following.

```
> corrplot(cor(t(x)), method = "ellipse", type = "upper", diag = FALSE)
```

What do you notice? What does it mean in terms of using correlation-based distance?

**Task 4** (10 points)
This task will use the dataset `data(Groceries)` and the function `apriori()`, which is an implementation of the apriori algorithm provided by `library(arules)`. In addition, we will also use `library(arulesViz)` to visualise the rules mined by `apriori()`.

(a) (1 point) Explore the data using the following

```
> library(arules); library(arulesViz); data(Groceries)
> itemFrequencyPlot(Groceries,topN=20,type="absolute")
```

(b) (2 points) Use `apriori()` to mine rules with $s_{min} = 0.001$ and $c_{min} = 0.8$.

(c) (1 point) Use `inspect()` to show the top 5 rules, and `summary()` to produce a summary of the rules generated in (b).

(d) (1 point) Use `sort()` with the option `by="confidence"` to the rules generated in (b).

(e) (4 points) Use the option `appearance` in `apriori()` to answer the following to questions.

(i.) What are customers likely to buy before buying whole milk

(ii.) What are customers likely to buy if they purchase whole milk?

(f) (1 point) Use the following to produce a visualisation of rules generated in (e) (i.).

```
> plot(rules, method = "graph", interactive = TRUE, shading = NA)
```