Lab 2: Basic Classification Algorithms

First Name: _____ **Last Name**: _____ **NetID**: _____

This lab is due March 10th (Friday) in the homework box at 2nd floor of Rhodes Hall. You need to submit your code, summary of the models you fit as well as your answers to the problems.

In this lab we will learn the basics of logistic regression and k-nearest neighbor algorithm, and learn how to measure the accuracy of the models we build.

Part (a): Logistic regression

Go to Blackboard and download the `Titanic` data set, make sure to download both the data set and the metadata file.

The Titanic data set contains information(age, gender, passenger class etc.) about 1309 passengers that travelled on the ship Titanic and whether or not they survived. We will fit a logistic regression model to predict what sorts of passengers were more likely to survive.

We will divide the Titanic data set into 2 parts: the training data set and the test data set. We will fit the model and learn the parameters with the training set, and then test the accuracy of our model using the test set by comparing the predicted outcomes of the data in the test set using our model and their true outcomes. The training and test sets must be created from the original data without bias. In this lab, we'll randomly sample 2/3 of the original data to be the training data and let the rest be the test data.

Load the titanic data set into R. There are 11 variables in the data set except the outcome variable. Note that 'Passenger ID', 'Name', 'Ticket' and 'Cabin' are probably not useful in predicting survival chance so we can remove them. Also note the 'Cabin' column is very sparse (has lots of missing data). Check if there is still any other missing data in the data set. You'll find the 'Age' column also has many rows where the data is missing. What can we do for these rows?

After figuring out how to deal with missing data, we need to check if all categorical variables have been identified by R and coerce them if not. Finally, we can create the training and test sets. Note that you need to store the outcome value of the test set separately and remove the outcome column from the test set.

Now you can fit the logistic regression model to the training data. Report the summary of your model.

Once you have fitted the model, use your model to predict the outcomes of the test data.

The logistic regression model we built will output the probability of surviving of each passenger in the test data set. To get a binary prediction of whether a passenger survives or not. We need to pick a threshold. For now we simply predict a passenger would survive if the output survival probability is greater than or equal to 0.5. How accurate is your model? What are some measures of accuracy that you can think of?

Now that you know how to measure accuracy, can you find a very simple classifier that would give you a decent accuracy (˜65% or more) by just looking at the data?

As you can see, the data is biased. As the data gets more biased our formula for accuracy becomes less indicative of how good our model is. A way to solve this is to use the following formula:

$$accuracy = \frac{1}{2} \times \frac{t_p}{(t_p + f_n)} + \frac{1}{2} \times \frac{t_n}{(t_n + f_p)}$$

Using this formula for accuracy, calculate the accuracy of our logistic regression model on the test dataset.

Identify which predictors are most influential to the final prediction.

Part (b): K Nearest Neighbors

We will continue analyzing the `Titanic` data set, but with a different classification method. In particular, We will learn how to do KNN in R, and compare the performance of KNN with the logistic regression approach used in part (a).

Do the same procedures to the Titanic data set as we did in part(a), i.e.

1. Delete unuseful columns 'Passenger ID', 'Name', 'Ticket' and 'Cabin'.
2. Deal with missing data by removing all associate rows. (Alternatively, you may replace missing values with mean/median).
3. Check for factor variables and convert them into dummy variables.

Note that we did not need the third step in part(a). Neither KNN nor linear/logistic regression deals with factor variables directly. The `lm()` and `glm()` functions in R automatically convert factors to dummies, but the built-in function `knn()` for KNN does not.

Can you think of the reason why KNN cannot deal with factor variables directly?

To convert factor variables into dummy variables, we use the function `model.matrix()`.

```
> titanic$Sex <- model.matrix( ~ Sex - 1, data=titanic )
> titanic$Embarked <- model.matrix( ~ Embarked - 1, data=titanic )
```

Why don't we just use the `as.numeric()` function to do the conversion?

We will divide the Titanic data set into training and test sets, again by randomly sampling 2/3 of the data to be the training set and using the rest as the test set. We will set a random seed so the the sampling is reproducible.

```
> set.seed(1)
> train_ind <- sample(1:nrow(titanic), 2/3*nrow(titanic))
```

We will now perform KNN using the `knn()` function, which is part of the `class` library. This function works rather differently from the other model fitting functions that we have encountered thus far. Rather than a two-step procedure in which we first fit the model and then we use the model to make predictions, the `knn()` function computes the predictions directly in a single command. Running `?knn` to display the help page, we see that the function requires four inputs:

1. A matrix containing the predictors associated with the training data, labeled **titanic.train** below.
2. A matrix containing the predictors associated with the data for which we wish to make predictions, labeled **titanic.test** below.
3. A vector containing the outcomes (class labels) for the training observations, labeled **train.survive** below.
4. A value for **k**, the number of nearest neighbors to be used by the classifier.

Before applying the `knn()` function, we normalize each predictors to have unit variance. Here we use function `scale()` to make each variables to have 0 mean and variance 1.

```
> titanic_normal <- scale(titanic[,!names(titanic) %in% 'Survived'])
> titanic.train <- titanic_normal[train_ind, ]
> titanic.test <- titanic_normal[-train_ind, ]
```

Why do we need this step?

We next create two variables that store the outcome values of the training/test sets.

```
train.survive=titanic$Survived[train_ind]
test.survive=titanic$Survived[-train_ind]
```

We can now apply the `knn()` function to predict the survival of the passengers. Note that in KNN, if several observations are tied as nearest neighbors, R will randomly break the tie.

```
> knn.pred = knn(titanic.train,titanic.test, train.survive, k=1)
> table(knn.pred,test.survive)
> mean(knn.pred==test.survive)
```

Here we choose $K = 1$, which gives an error rate of about 23% (according to my experiment).

We explore the performance of KNN with other values of K. Run KNN for $K = 1, 3, 5, 10, 20, 50$. (You may use a for loop for convenience). Report the corresponding error rates, and find the value of K that gives the lowest error rate.

Finally, compare the prediction performance of KNN (using your optimal K) with the logistic regression model from part(a). Which method gives a better accuracy rate in this case?

Take Home Questions

(1) Recall in Lab 1 you tried to get the units of the ω coefficients of the linear regression model. What are the units of the coefficients of the logistic regression model.
(Hint: Recall the logistic regression model assumes:

$$P(y = 1|\mathbf{x}, \beta) = \frac{1}{1 + \exp(-\beta^T \mathbf{x})}$$

Try to find an expression $f(p)$ of $p = P(y = 1|\mathbf{x}, \beta)$ which satisfies $f(p) = \beta^T \mathbf{x}$. We call $f(p)$ "logit" or "log-odds". Recall the interpretation of β coefficients in the linear regression model is the change of outcome variable when a predictor changes by 1 unit. Interpret β coefficients of the logistic regression model in a similar way.)

For every one year of age, a passenger's logit of surviving on Titanic will decrease (or increase) by a factor of what?

(2) Go to Blackboard and download the Cancer data set: `pros.xls` and `pros_meta.txt`. We want to build a logistic regression model to predict whether the tumor has penetrated the prostatic capsule.

Load the data into R. Note the 'VOL' column has lots of missing data and you need to remove these instances first. We will fit a logistic regression model with the data and test the accuracy of our model.
Divide the data set into training and test data. Fit a logistic regression model with the training data. Report the summary of your model.

(3) Now let's check the accuracy of your model in Q2. Remember the logistic regression model outputs a probability of penetration p for each patient. We could pick a threshold p_0 so that we predict the outcome to be 1 if $p \geq p_0$. Hence different p_0 gives different classifiers.

Define $FPR = \frac{fp}{fp+tn}$ to be the false positive rate: the proportion of negative instances that are falsely predicted to be positive instances.

And define $TPR = \frac{tp}{tp+fn}$ to be the true positive rate: the proportion of positive instances that are also predicted correctly. Each p_0 gives a new pair of (FPR, TPR).

Pick a list of p_0 and plot the corresponding (FPR, TPR) pairs on a graph. We obtain the FPR vs. TPR plot (i.e., the ROC curve) for our logistic regression model as shown in Figure 1. (If you generate this plot yourself, it will look slightly different due to the randomness in splitting in to training and test datasets. But the general trend of the curve will be the same.)

Next, assume we have a balanced data set, i.e. with 50% positive instances and 50% negative ones. What would the FPR vs. TPR plot look like for the following two classifiers:

(a) You were guessing the prediction randomly.
(b) You were predicting only one of the classes for everything.

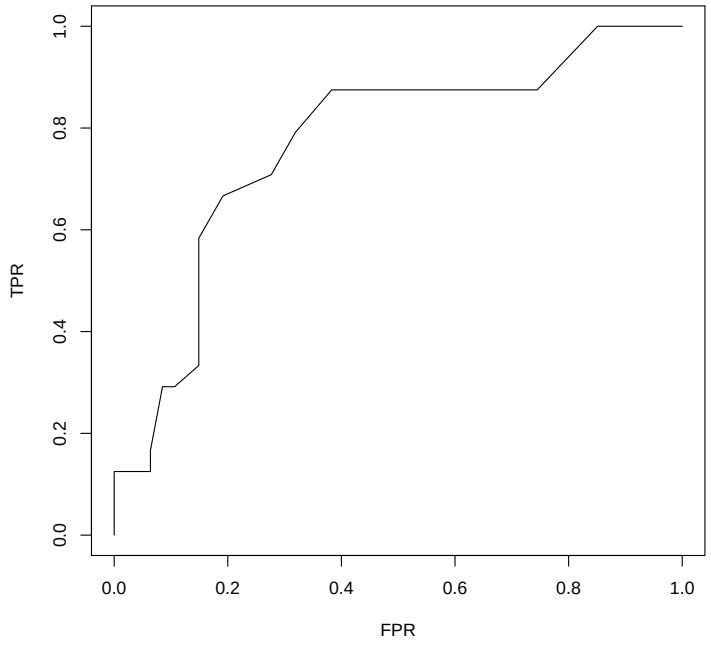

Figure 1: ROC curve: False Positive Rate vs. True Positive Rate

What would the plots in (a) (b) look like if we have an unbalanced data set, say p% positive class?

(4) In predicting cancer diagnosis, we would like to be more conservative and prefer to error on the side of a false diagnosis rather than missing cancer. What would you do in this case when making predictions according to the probabilities output by your logistic regression model?

Bonus Take Home Question

Compare the performance of KNN with and without normalization of the data set.

Create a plot of the value of K vs. the associating error rate, where $K \in [1, 20]$, with normalization. Next, make the same plot but without normalizing the data set. Compare the two plots. (You may attach additional page for printing out the 2 plots.)

What can you say about the accuracy rates with and without normalization?