Connections between Perceptron and Logistic Regression

Jason Marcell Setiadi

March 1st, 2022

Contents

1 Introduction and Background

Binary classification is one of the simplest classification problems we often encounter in real life. There are many statistical techniques that are built to tackle this problem. One of the classical methods that is specifically built for this type of problem is logistic regression. In this research, I am interested in studying the connections/relationships between logistic regression and single-layer neural network also known as perceptron (Durrett (2021)), which is a modern technique. Perceptron is a more versatile model than logistic regression because it can be used for diverse types of problem (classification, regression, etc). I would like to highlight the similarities and differences between the two, and compare their prediction performance on a real-world dataset.

2 Methods and Materials

2.1 Dataset Information

The Adult dataset (Dua and Graff (2017)) is from the Census bureau and the task is to predict whether a given adult makes more or less than $50,000 a year based on attributes such as education, hours of work per week, etc. This dataset contains 48842 observations along with 14 predictor variables, 6 of which are quantitative and 8 are qualitative. The response is a binary variable indicating low (<=50K) vs high (>50K) income. More details about the variables in the dataset are presented in Table (1).

Variable Name	Variable Description	Variable Type
age	Age	Numerical
workclass	Work industry	Categorical (8 Levels)
fnlwgt	Weight in population	Numerical
education	Education level	Categorical (16 Levels)
educational.num	Total years of education	Numerical
marital.status	Marital status	Categorical (7 Levels)
occupation	Occupation	Categorical (14 Levels)
relationship	Relationship	Categorical (6 Levels)
race	Race	Categorical (5 Levels)
gender	Gender	Categorical (2 Levels)
capital.gain	Income gain from sources outside salary	Numerical
capital.loss	Income loss from sources outside salary	Numerical
hours.per.week	Number of working hours per week	Numerical
native.country	Native country	Categorical (41 Levels)
income	Salary income	Categorical (2 Levels)

There are 3620 rows with missing values which accounts for 7.4% of the total observations in the dataset. Since we have a good amount of observations in the dataset, we decide to get rid of these observations with missing values. We end up with a total of 45222 observations. Furthermore, we can perform several variable transformation by getting rid of redundant/useless columns and reducing the number of levels in categorical variables by combining them, adapted from (Zhu (2016)). We created a new variable called "us.citizen" which determines whether the individual is a US Citizen or not based on native.country variable. We then remove "native.country", "education" since it is redundant with educational.num variable, and "relationship" since it can be accounted by "gender" and "marital.status" variables. We also remove "capital.gain" and "capital.loss" variables since most observations have 0 entries in both columns, as well as "fnlwgt" for simplicity of analysis. In addition, we transform the "workclass" variable to just 4 levels, "occupation" variable to just 5 levels, and "marital.status" variable to just 3 levels. Details about our variables after the transformations are shown in Table (2).

Variable Name	Variable Description	Variable Type
age	Age	Numerical
workclass	Work industry	Categorical (4 Levels)
educational.num	Total years of education	Numerical
marital.status	Marital status	Categorical (3 Levels)
occupation	Occupation	Categorical (5 Levels)
race	Race	Categorical (5 Levels)
gender	Gender	Categorical (2 Levels)
hours.per.week	Number of working hours per week	Numerical
income	Salary income	Categorical (2 Levels)
us.citizen	Is a US citizen or not	Categorical (2 Levels)

We now perform exploratory data analysis to obtain more information about the distribution of our data. We observe in Figure (1) that there is class imbalance in our response variable which contains 37155 low income and 11687 high income observations. Therefore, we need to address this issue by evaluating our models with different metrics such as precision and f1-scores which can account for the skewness (Tan, Steinbach, and Kumar (2006)).

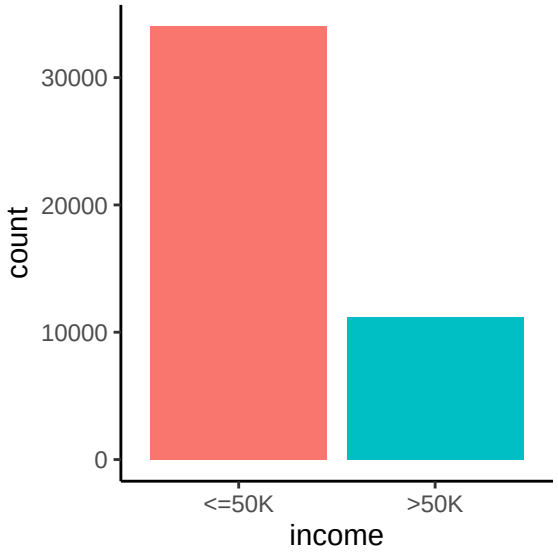

Figure 1: Income Class Distribution

2.2 Logistic Regression

Based on the given binary classification task, it's natural for us to apply logistic regression since it is made specifically for such tasks. It gives us a probabilistic output $p(X)$ which ranges from 0 to 1 for a given class and $1 - p(X)$ for the other class, rather than linear regression which gives a continuous output from $-\infty$ to ∞. The way this model gives such output is through the logistic function given in Equation (1) (Gareth et al. (2013)). Here, $X_1, ..., X_p$ are our predictor variables, $\beta_1, ..., \beta_p$ are their respective coefficients, and β_0 is the intercept.

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}} \tag{1}$$

In order to make interpretations of the model parameters, we might be interested in Equation (2) (Gareth et al. (2013)) through manipulating Equation (1).

$$log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{2}$$

This way, we can interpret β_i where $i \in (1, p)$ as the change in the log odds of a given class, associated with a one-unit increase in X_i. Another interpretation we can make is that it multiplies the odds of a given class by e^{β_i}. Furthermore, we generally are not interested in β_0 and so no interpretation is needed. The structure of logistic regression is given by the following figure (Figure (2)).

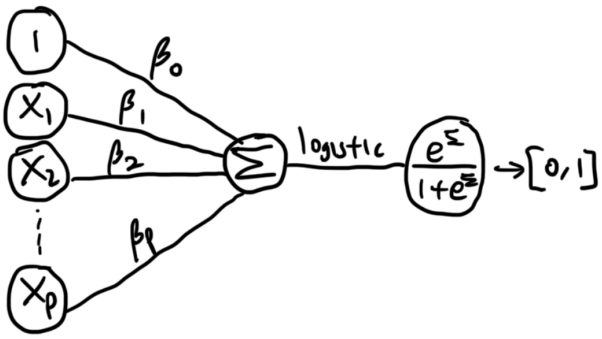

Figure 2: Structure of Logistic Regression

In general, the parameters $\beta_0,\ldots,\beta_p$ are unknown and we need to estimate them using our training data. Rather than using the least squares approach as in linear regression, we will use the maximum likelihood approach for logistic regression as it has better statistical properties for non-linear models (Gareth et al. (2013)). It chooses estimates $\hat{\beta}_0,\ldots,\hat{\beta}_p$ such that the predicted probability is as close to the observed class (0 or 1) by maximizing the likelihood function (Equation (3)); hence the name. Furthermore, the likelihood function is actually the product of the probability mass function of Bernoulli distribution.

$$\begin{aligned} l(\beta_0,...,\beta_p) &= \prod_{i=1}^{n} p(x_i)^{y_i}(1-p(x_i))^{1-y_i} \\ &= \sum_{i=1}^{n} y_i * log(p(x_i)) + (1-y_i) * log(1-p(x_i)) \end{aligned} \tag{3}$$

Maximizing the product of probabilities is equivalent to maximizing the sum of log probabilities (Durrett (2021)), which is given in Equation (3), commonly known as log loss. Moreover, minimizing this log loss is essentially the same as minimizing the cross-entropy loss (Equation (4)), which is a more general form of the loss. Cross-entropy is basically the negation of the dot product of true labels (y_i) and the log of the softmax values ($f(x_i)$). Thus we can use the estimates that minimize this loss to predict the unseen data.

$$\min_{\beta_0,\beta_1,\ldots,\beta_p} \left\{ \sum_{i=1}^{n} -y_i * log(p(x_i)) - (1-y_i) * log(1-p(x_i)) \right\} = \min_{\beta_0,\beta_1,\ldots,\beta_p} \left\{ -\sum_{i=1}^{n} y_i * log(f(x_i)) \right\} \tag{4}$$

Before doing prediction, logistic regression has several assumptions (Gupta (2020)) that we need to check to ensure reliable results. First is independence of observations, usually obtained through random sampling. Secondly, we have the linearity assumption which requires each quantitative predictor to have a linear relationship with the logit of the response. Next is absence of multicollinearity, meaning no predictor variables are highly correlated with each other. This is evaluated through calculating variance inflation factors (VIF) values for each predictor. Finally, we have to check for any influential values by looking at the Cook's distance of each observation. We determine an observation as influential if its Cook's distance is above the threshold of 4/number of observations. Once these assumptions are met, we can proceed with our prediction task.

2.3 Perceptron

Besides logistic regression, we are also interested in the perceptron algorithm and how it connects with logistic regression. Perceptron is a single-layer neural network that can also be used for binary classification

tasks. It also incorporates coefficients for each predictor variable known as weights. However, instead of one probabilistic output, the perceptron has two probabilistic outputs to account for the two classes in a binary classification problem. It can be easily extended to multi-class classification problems by using k outputs given k classes. Below is the diagram of the perceptron's structure (Figure (3)).

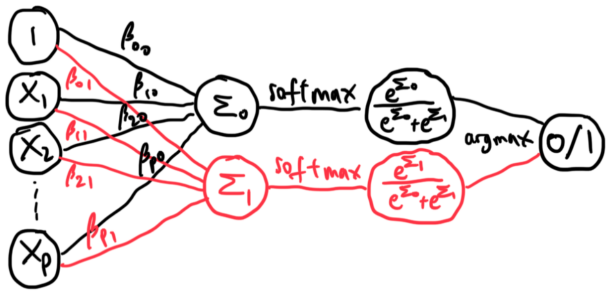

Figure 3: Structure of Perceptron

Here, the inputs consist of the intercept and our predictor variables $X_1, ..., X_p$. However, since there are two outputs given the two classes, the number of weights/parameters are double the input size $(\beta_{00}, ..., \beta_{p0}, \beta_{01}, ..., \beta_{p1})$. Then, the net input function sums up the product of weights and inputs that provides us with two values for each class (Equation (5)).

$$\Sigma = \begin{cases} \beta_{00} + \beta_{10}X_1 + ... + \beta_{p0}X_p & \text{for class 0} \\ \beta_{01} + \beta_{11}X_1 + ... + \beta_{p1}X_p & \text{for class 1} \end{cases} \tag{5}$$

Now we apply the softmax function (Equation (6)) to obtain two probabilistic values for each of the classes. The key property of softmax is that the probability outputs will sum up to 1. This lets the perceptron easily classify the observation since one class will certainly have higher probability than the other.

$$softmax(\Sigma_i) = \frac{e^{\Sigma_i}}{\sum_{i=0}^{1} e^{\Sigma_i}} \tag{6}$$

Finally, the algorithm classifies the observation as class 0 or 1 according to which has a higher softmax probability value. Although perceptron is pretty similar to logistic regression, it is difficult to provide meaningful interpretations for the perceptron's predictions or derive the perceptron as a maximum likelihood estimation algorithm according to Gareth et al. (2013). On the other hand, unlike logistic regression, it does not have any model assumptions we need to agree upon before doing prediction. Another key feature of perceptron (neural networks in general) is that the data needs to be numerically-valued, so we have to convert factors (qualitative variables) to dummy variables and standardize the quantitative variables so each column has mean zero and variance one (Gareth et al. (2013)).

In order to fit the perceptron for model training, we will use cross-entropy (Equation (4)) as our model's loss function, which is explained in Section (2.2). The goal is to minimize the loss function in order to obtain the model parameters. However, as our model gets more complex, it is sometimes not so easy to minimize the loss, which brings us to the stochastic gradient descent (SGD) method. According to Gareth et al. (2013), this method is the state of the art for neural networks since it enables us to reach a good local minimum of complex loss functions in a more efficient way. The way this method works is to randomly select a subset of the training data and use it to perform gradient descent, which means to keep updating the parameters (θ) at every iteration m that reduce the gradient of the loss function $(L(\theta))$ until it reaches a minimum (Equation (7)). How fast it descends is controlled by the learning rate parameter (ρ) which we can specify during the model training process (Gareth et al. (2013)).

$$\theta^{m+1} = \theta^m - \rho * \nabla L(\theta^m) \tag{7}$$

The use of SGD adds some additional parameters to our model fitting process such as epochs and batch size. Epochs is the number of training iterations while batch size is the number of observations used for gradient computation at each step of SGD (Gareth et al. (2013)). After the model fitting process then we can do our prediction task.

2.4 Analysis Procedure

Since the goal of our analysis is to see how well logistic regression and the perceptron performs on our dataset, we will first split the data into training and test sets so we only train the models on the training set and let the models predict the unseen observations in the test set. We will randomly select two-thirds of our data as training and the remaining as test set. Furthermore, we will use 9 predictor variables as mention in Section (2.1). We are only interested in the main effects of these predictors so we don't need to incorporate interactions in our model fitting process.

After fitting the models to the training set, we need to check the logistic regression model assumptions. Once they are met, we can proceed with performing predictions. We use a 0.5 probabilistic cutoff to determine our predicted classes for logistic regression. We then evaluate the prediction performance of both techniques using several metrics (Tan, Steinbach, and Kumar (2006)) such as accuracy, precision, recall/TPR, FPR, F1-Score, and TPR/FPR ratio (Equation (8)). These metrics can be calculated from the confusion matrix of our predictions (Table (3)). One key metric we want to focus on is the F1-score due to the nature of class imbalance in our dataset since it can account for the skewness of the class distribution (Tan, Steinbach, and Kumar (2006)).

Table 3: Confusion Matrix

	Negative Predicted Class	Positive Predicted Class
Negative Actual Class	True Negative (TN)	False Positive (FP)
Positive Actual Class	False Negative (FN)	True Positive (TP)

$$\begin{aligned} accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ precision(p) &= \frac{TP}{TP + FP} \\ recall(r)/TPR &= \frac{TP}{TP + FN} \\ FPR &= \frac{FP}{FP + TN} \\ F1 - Score &= \frac{2rp}{r + p} = \frac{2TP}{2TP + FP + FN} \end{aligned} \tag{8}$$

3 Results

3.1 Logistic Regression

We now present our analysis results on the adult dataset. We end up with 30148 observations for training and 15074 for testing. After doing the train-test-split and fitting the logistic regression model as mentioned

in Section (2.4), we have to check the model assumptions and make sure everything is met before evaluating on the test set. First, we assume data is independent due to random sampling since it is collected from a census survey.

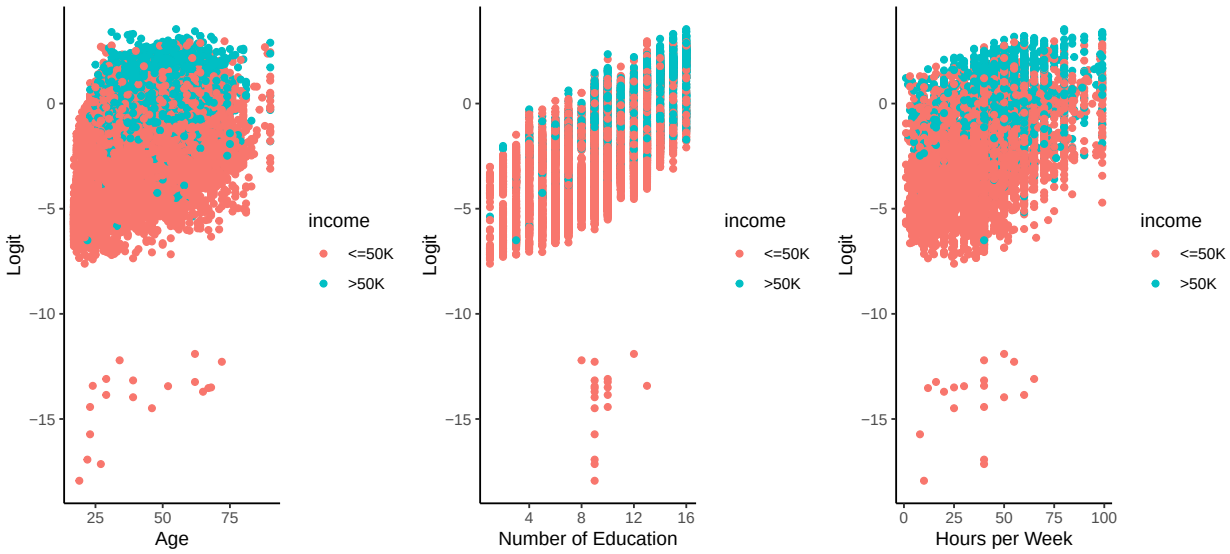

Figure 4: Linearity Assumption Plot

Second, based on Figure (4), we can see quite a linear relationship between the quantitative predictors and the logit of response, although we can also see some separation in each plot due to the different income classes.

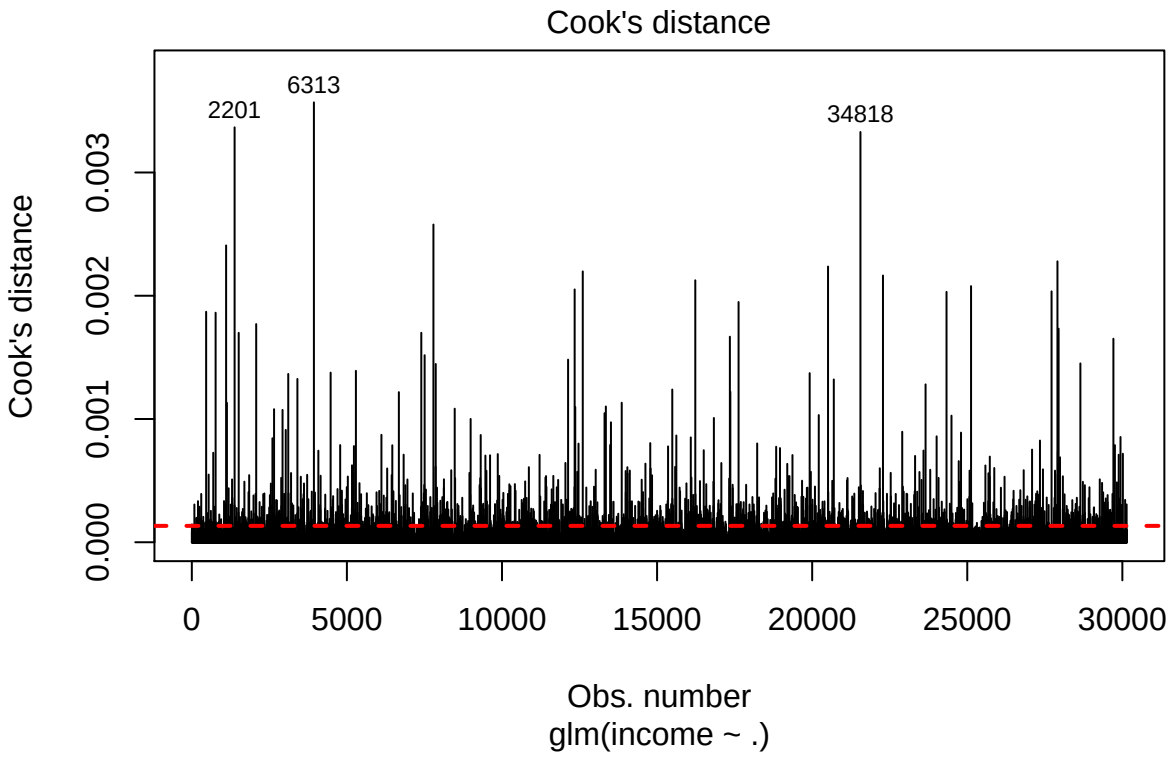

Figure 5: Influential Values Plot

Next, we check for any influential values. We use $\frac{4}{30148}$ as our Cook's distance cutoff (red line in Figure (5)) to determine influential values. Surprisingly, there are approximately 1217 influential observations which is quite a lot. Therefore, we tried to remove these values and see how it affects our performance. Since it doesn't change our results, we stick to using the full model.

Table 4: Predictor VIF Values

	VIF
age	1.082
workclass	1.030
educational.num	1.200
marital.status	1.089
occupation	1.060
race	1.035
gender	1.154
hours.per.week	1.040
us.citizen	1.131

Lastly, we check whether there is multicollinearity in the data by computing the variance inflation factors (VIF) values. We can see from Table (4) that all the VIF values are within reasonable range and thus we can say that there is no collinearity among the predictors. Let's now observe the model summary.

Table 5: Logistic Regression Model Summary

	Estimate	Pr(>|z|)
age	0.030	0.000
workclassSelf-Employed	-0.298	0.000
educational.num	0.325	0.000
marital.statusSeparated	-2.068	0.000
marital.statusSingle	-2.539	0.000
raceAsian-Pac-Islander	0.442	0.050
raceWhite	0.475	0.016
genderMale	0.362	0.000
hours.per.week	0.030	0.000
us.citizenYes	0.324	0.000

We can see that out of the 18 predictor variables in our logistic regression model, 11 of them have p-value less than 0.05. However, out of the 11, 8 are highly significant which are age, self-employed work class (yes/no), number of education years, single and separated marital status (yes/no), male gender (yes/no), working hours per week, and us citizen (yes/no), given the p-values are very close to 0. Let's try to interpret some of these important predictors based on their model coefficients.

To conclude, single or separated individuals have lower probability of having high income. Specifically, being single decrease the log odds of income by 4.792 units while being separated decrease the log odds by 3.339. On the contrary, US citizens, male individuals, and individuals with more education years have higher probability of having high income. Being a US citizen increase the log odds of income by 0.842, being a male increase the log odds by 0.425, while one year increase in number of education increase the log odds by 0.474.

We also tried performing forward and backward selection to see whether a simpler model is chosen. Turns out, both methods chose the full model.

3.2 Perceptron

Table 6: Perceptron Input Variables

Input Variables/Nodes
(Intercept)
age
workclassGovernment
workclassOther
workclassPrivate
workclassSelf-Employed
educational.num
marital.statusSeparated
marital.statusSingle
occupationBlue-Collar
occupationProfessional
occupationService
occupationWhite-Collar
raceAsian-Pac-Islander
raceBlack
raceOther
raceWhite

Input Variables/Nodes
genderMale
hours.per.week
us.citizenYes

```
## Model: "sequential"
## _________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =========================================================================
##  dense (Dense)                      (None, 2)                       40
##
## =========================================================================
## Total params: 40
## Trainable params: 40
## Non-trainable params: 0
## _________________________________________________________________________
```

Now, we want to fit our perceptron model on the training set. We follow the procedure that has been described in Section (2.4). Based on the model summary, we can see that there are 20 input nodes, 40 parameters, and 2 output nodes in our perceptron model, as described in the Section (2.3).

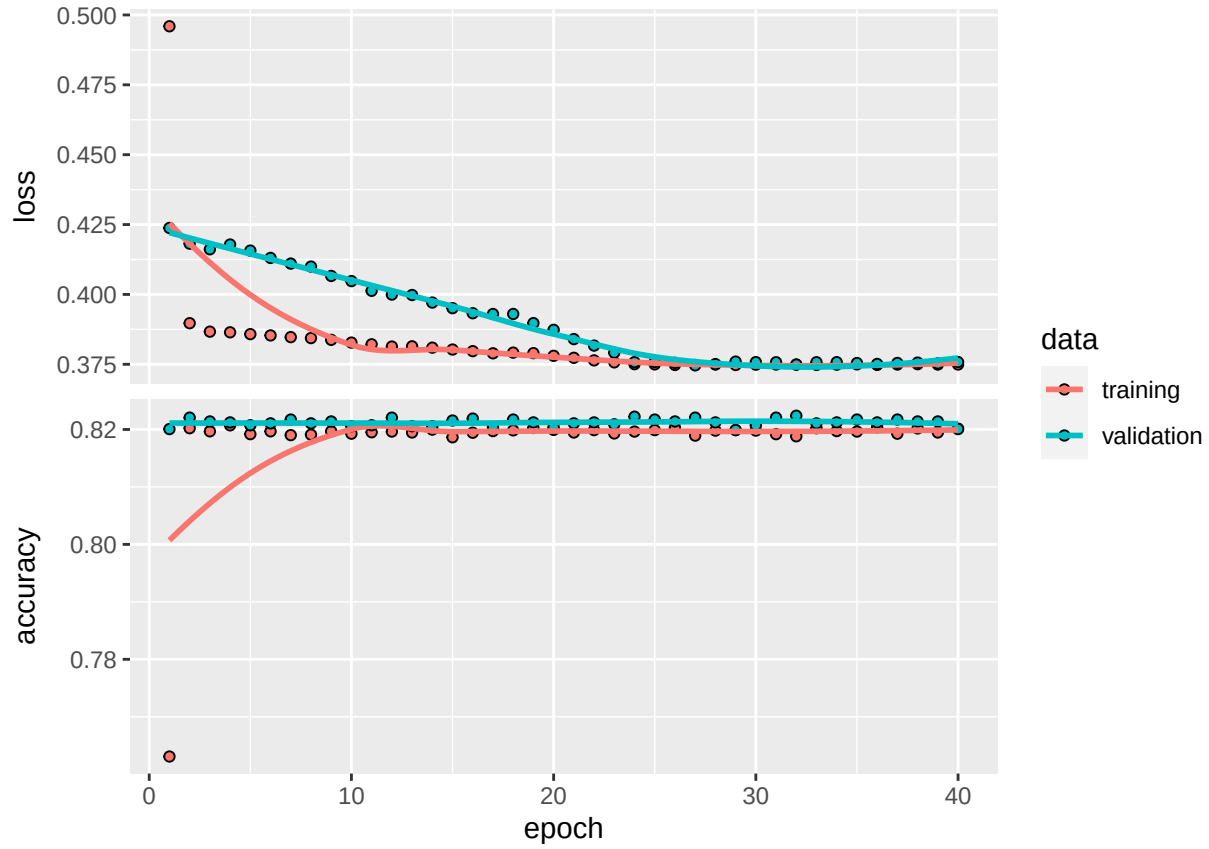

Figure 6: Loss and Accuracy Plots for Training and Validation

For our model training, we perform 40 epochs (training iterations) along with batch size of 25 observations for the SGD process. By default, the learning rate parameter is set to 0.001 ((n.d.)). We can see from Figure (6) that there is a drastic decrease in training loss as we perform more iterations of perceptron training, which is what we desired. We don't see much decrease in training loss after 40 iterations so there is no need to perform more. However, the accuracy doesn't seem to change that much as number of training iterations (epoch) increases.

Table 7: Logistic Regression vs Perceptron Prediction Performance

	Accuracy	Precision	Recall/TPR	FPR	F1-Score	TPR/FPR
Logistic Regression	0.820	0.675	0.530	0.084	0.594	6.312
Perceptron	0.819	0.669	0.533	0.087	0.593	6.144

After training both models on the training set, we fit them to our test set to obtain our predictions. We compare the results between logistic regression and perceptron prediction performance in Table (5). We can see that logistic regression is the superior model for this dataset based on all the metrics.

4 Discussion and Summary

We have explained the main connections between logistic regression and perceptron algorithms both theoretically and applied using the adult dataset. Some key differences are as the following. Number of parameters in perceptron are twice as much as in logistic regression, assuming the same number of predictor/input variables. Furthermore, there is a slight difference in activation function being used, sigmoid/logistic in logistic regression while perceptron uses softmax function. In the case of binary classification, logistic regression has one continuous probabilistic output while perceptron has one discrete class output. Moreover, logistic regression has several model assumptions need to be met while perceptron does not. However, logistic regression is more interpretable than the perceptron.

Based on our results, we can see that modern techniques doesn't always perform better than classical techniques. Besides, we can see that they have very similar prediction performance. This might be caused by the minimization of the same loss function (cross-entropy) by both techniques. Nevertheless, each have their own strengths and weaknesses in different real-world scenarios. Further research can be done by looking at relationships with SVM since it also have some connections with perceptron and logistic regression or we could also fit a more complicated neural network with hidden layers and see how the results differ from the perceptron.

5 References

Dua, Dheeru, and Casey Graff. 2017. "UCI Machine Learning Repository." University of California, Irvine, School of Information; Computer Sciences. http://archive.ics.uci.edu/ml.

Durrett, Greg. 2021. "Connections Between Perceptron and Logistic Regression (and Svm)." https://www.cs.utexas.edu/~gdurrett/courses/sp2021/perc-lr-connections.pdf.

Gareth, James, Witten Daniela, Hastie Trevor, and Tibshirani Robert. 2013. *An Introduction to Statistical Learning: With Applications in R*. Spinger.

Gupta, Deepak. 2020. "Logit Reression Assumptions." https://rpubs.com/guptadeepak/logit-assumptions.

Tan, Pang-Ning, Michael Steinbach, and Vipin Kumar. 2006. "Data Mining Introduction." *People's Posts and Telecommunications Publishing House, Beijing.*

Zhu, Haojun. 2016. "Predicting Earning Potential Using the Adult Dataset." https://rstudio-pubs-static.s3.amazonaws.com/235617_51e06fa6c43b47d1b6daca2523b2f9e4.html.

n.d. https://keras.rstudio.com/reference/optimizer_rmsprop.html.

6 Appendix

```
library(tensorflow)
library(keras)
library(car)
library(leaps)
library(ggplot2)
library(dplyr)
library(ggpubr)
set.seed(4893)

# Load the data set
data = read.csv('adult.csv', stringsAsFactors = T)
data = droplevels(data, exclude = "?")

# dataset information
var_names = colnames(data)
var_meaning = c("Age", "Work industry", "Weight in population", "Education level", "Total years of educa
var_type = c("Numerical", "Categorical (8 Levels)", "Numerical", "Categorical (16 Levels)",
             "Numerical", "Categorical (7 Levels)", "Categorical (14 Levels)",
             "Categorical (6 Levels)", "Categorical (5 Levels)", "Categorical (2 Levels)",
             "Numerical", "Numerical", "Numerical", "Categorical (41 Levels)",
             "Categorical (2 Levels)")
# knitr::kable(cbind(var_names,var_meaning,var_type), col.names = c("Variable Name", "Variable Descript
summary(data)
```

```
##       age                    workclass         fnlwgt
##  Min.   :17.00   Private         :33906   Min.   :  12285
##  1st Qu.:28.00   Self-emp-not-inc: 3862   1st Qu.: 117550
##  Median :37.00   Local-gov       : 3136   Median : 178144
##  Mean   :38.64   State-gov       : 1981   Mean   : 189664
##  3rd Qu.:48.00   Self-emp-inc    : 1695   3rd Qu.: 237642
##  Max.   :90.00   (Other)         : 1463   Max.   :1490400
##                  NA's            : 2799
##         education     educational.num                marital.status
##  HS-grad     :15784   Min.   : 1.00   Divorced              : 6633
##  Some-college:10878   1st Qu.: 9.00   Married-AF-spouse     :   37
##  Bachelors   : 8025   Median :10.00   Married-civ-spouse    :22379
##  Masters     : 2657   Mean   :10.08   Married-spouse-absent:  628
##  Assoc-voc   : 2061   3rd Qu.:12.00   Never-married         :16117
##  11th        : 1812   Max.   :16.00   Separated             : 1530
##  (Other)     : 7625                   Widowed               : 1518
##            occupation            relationship                    race
##  Prof-specialty : 6172   Husband       :19716   Amer-Indian-Eskimo:  470
##  Craft-repair   : 6112   Not-in-family :12583   Asian-Pac-Islander: 1519
##  Exec-managerial: 6086   Other-relative: 1506   Black             : 4685
```

```
##  Adm-clerical   : 5611   Own-child     : 7581   Other             :  406
##  Sales          : 5504   Unmarried     : 5125   White             :41762
##  (Other)        :16548   Wife          : 2331
##  NA's           : 2809
##     gender       capital.gain    capital.loss    hours.per.week
##  Female:16192   Min.   :    0   Min.   :   0.0   Min.   : 1.00
##  Male  :32650   1st Qu.:    0   1st Qu.:   0.0   1st Qu.:40.00
##                 Median :    0   Median :   0.0   Median :40.00
##                 Mean   : 1079   Mean   :  87.5   Mean   :40.42
##                 3rd Qu.:    0   3rd Qu.:   0.0   3rd Qu.:45.00
##                 Max.   :99999   Max.   :4356.0   Max.   :99.00
##
##        native.country    income
##  United-States:43832   <=50K:37155
##  Mexico        :  951   >50K :11687
##  Philippines   :  295
##  Germany       :  206
##  Puerto-Rico   :  184
##  (Other)       : 2517
##  NA's          :  857
```

```
# check for missing values
apply(is.na(data),2,sum) #by column
```

```
##             age       workclass          fnlwgt       education educational.num
##               0            2799               0               0               0
##  marital.status      occupation    relationship            race          gender
##               0            2809               0               0               0
##    capital.gain    capital.loss  hours.per.week  native.country          income
##               0               0               0             857               0
```

```
sum(apply(data,1,anyNA)) #by row
```

```
## [1] 3620
```

```
sum(apply(data,1,anyNA))*100/nrow(data) #percentage of missing values
```

```
## [1] 7.411654
```

```
# remove rows with missing values
data = na.omit(data)

# transform native country to us citizen column
data$us.citizen = factor(ifelse(data$native.country == "United-States", "Yes", "No"))
data$native.country = NULL

# remove education column since it is redundant with educational.num column
data$education = NULL

# remove relationship column since it can be assessed from gender and marital.status
data$relationship = NULL
```

```r
# remove capital.gain and capital.loss because of 0 entries as well as fnlgwt
data$capital.gain = NULL
data$capital.loss = NULL
data$fnlwgt = NULL

# transform workclass column
data$workclass <- gsub('Federal-gov', 'Government', data$workclass)
data$workclass <- gsub('Local-gov', 'Government', data$workclass)
data$workclass <- gsub('State-gov', 'Government', data$workclass)
data$workclass <- gsub('Self-emp-inc', 'Self-Employed', data$workclass)
data$workclass <- gsub('Self-emp-not-inc', 'Self-Employed', data$workclass)
data$workclass <- gsub('Never-worked', 'Other', data$workclass)
data$workclass <- gsub('Without-pay', 'Other', data$workclass)
data$workclass <- as.factor(data$workclass)

#transform occupation column
data$occupation <- gsub('Adm-clerical', 'White-Collar', data$occupation)
data$occupation <- gsub('Craft-repair', 'Blue-Collar', data$occupation)
data$occupation <- gsub('Exec-managerial', 'White-Collar', data$occupation)
data$occupation <- gsub('Farming-fishing', 'Blue-Collar', data$occupation)
data$occupation <- gsub('Handlers-cleaners', 'Blue-Collar', data$occupation)
data$occupation <- gsub('Machine-op-inspct', 'Blue-Collar', data$occupation)
data$occupation <- gsub('Other-service', 'Service', data$occupation)
data$occupation <- gsub('Priv-house-serv', 'Service', data$occupation)
data$occupation <- gsub('Prof-specialty', 'Professional', data$occupation)
data$occupation <- gsub('Protective-serv', 'Service', data$occupation)
data$occupation <- gsub('Sales', 'White-Collar', data$occupation)
data$occupation <- gsub('Tech-support', 'Service', data$occupation)
data$occupation <- gsub('Transport-moving', 'Blue-Collar', data$occupation)
data$occupation <- as.factor(data$occupation)

# transform marital_status column
data$marital.status <- gsub('Divorced', 'Separated', data$marital.status)
data$marital.status <- gsub('Widowed', 'Separated', data$marital.status)
data$marital.status <- gsub('Married-AF-spouse', 'Married', data$marital.status)
data$marital.status <- gsub('Married-civ-spouse', 'Married', data$marital.status)
data$marital.status <- gsub('Married-spouse-absent', 'Married', data$marital.status)
data$marital.status <- gsub('Never-married', 'Single', data$marital.status)
data$marital.status <- as.factor(data$marital.status)

# new variable description
var_names = colnames(data)
var_meaning = c("Age", "Work industry", "Total years of education", "Marital status", "Occupation", "Ra
var_type = c("Numerical", "Categorical (4 Levels)", "Numerical", "Categorical (3 Levels)",
             "Categorical (5 Levels)", "Categorical (5 Levels)", "Categorical (2 Levels)",
             "Numerical", "Categorical (2 Levels)","Categorical (2 Levels)")
# knitr::kable(cbind(var_names,var_meaning,var_type), col.names = c("Variable Name", "Variable Descript
summary(data)
```

```
##       age                workclass     educational.num   marital.status
##  Min.   :17.00   Government   : 6452   Min.   : 1.00   Married  :21639
##  1st Qu.:28.00   Other        :   21   1st Qu.: 9.00   Separated: 8985
##  Median :37.00   Private      :33307   Median :10.00   Single   :14598
```

```
##  Mean   :38.55   Self-Employed: 5442   Mean   :10.12
##  3rd Qu.:47.00                         3rd Qu.:13.00
##  Max.   :90.00                         Max.   :16.00
##         occupation                    race          gender      hours.per.week
##  Armed-Forces:   14   Amer-Indian-Eskimo:  435   Female:14695   Min.   : 1.00
##  Blue-Collar :14832   Asian-Pac-Islander: 1303   Male  :30527   1st Qu.:40.00
##  Professional: 6008   Black             : 4228                  Median :40.00
##  Service     : 7436   Other             :  353                  Mean   :40.94
##  White-Collar:16932   White             :38903                  3rd Qu.:45.00
##                                                                 Max.   :99.00
##    income      us.citizen
##  <=50K:34014   No : 3930
##  >50K :11208   Yes:41292
##
##
##
##
```

```
# class distribution
class_dist = count(data, income, sort = TRUE)
ggplot(class_dist)+theme_classic()+theme(legend.position="none")+
  geom_col(aes(x=income, y=n, fill=income))+labs(y="count")
```

```
cm = matrix(c("True Negative (TN)","False Negative (FN)","False Positive (FP)","True Positive (TP)"),nr
rownames(cm) = c("Negative Actual Class","Positive Actual Class")
# knitr::kable(cm,col.names = c("Negative Predicted Class", "Positive Predicted Class"),
#              caption = "Confusion Matrix")

# split data into train and test data
n = nrow(data)
ntest = trunc(n / 3)
testid = sample(1:n, ntest)
train_data = data[-testid,]

# change the response to 1 for high income and 0 for low income
y = ifelse(data$income == ">50K", 1, 0)

# fit model on training set
glm.fits = glm(income ~ ., family = "binomial", data = train_data)
glm.logit = predict(glm.fits, train_data)

# we assume data is random and independent
# linearity assumption
base = ggplot(train_data)+theme_classic()
age = base+geom_point(aes(y=glm.logit,x=age,color=income))+labs(x="Age", y="Logit")
education = base+geom_point(aes(y=glm.logit,x=educational.num,color=income))+labs(x="Number of Educatio
hours.per.week = base+geom_point(aes(y=glm.logit,x=hours.per.week,color=income))+labs(x="Hours per Week
ggarrange(age,education,hours.per.week,nrow=1,ncol=3)
```

```
# influential values
plot(glm.fits, which=4); abline(h = 4/nrow(train_data), lty = 2, lwd=2, col = "red")
```

```r
cooks.dist = sort(cooks.distance(glm.fits),decreasing=TRUE)
influential.pts = as.numeric(names(cooks.dist)[cooks.dist > (4/nrow(train_data))])

# fit model without influential values in the training set
# subset = !(as.numeric(rownames(train_data)) %in% influential.pts)
# glm.fits = glm(income ~ subset+., family = "binomial", data = train_data)
# summary(glm.fits)
# glm.fits = glm(income ~ ., family = "binomial", data = train_data[subset,])

# multicollinearity
vif(glm.fits)
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## age             1.171796  1        1.082495
## workclass       1.192078  3        1.029716
## educational.num 1.440590  1        1.200246
## marital.status  1.403905  2        1.088515
## occupation      1.596469  4        1.060218
## race            1.319542  4        1.035268
## gender          1.330830  1        1.153616
## hours.per.week  1.080962  1        1.039693
## us.citizen      1.279704  1        1.131240
```

```r
# knitr::kable(round(vif(glm.fits)[,3],3), col.names = "VIF", caption="Predictor VIF Values")

# model summary
(glm.summary = summary(glm.fits))
```

```
##
## Call:
## glm(formula = income ~ ., family = "binomial", data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4554  -0.5939  -0.2626  -0.0487   3.6062
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -18.215260 202.413905  -0.090   0.9283
## age                        0.030190   0.001509  20.006  < 2e-16 ***
## workclassOther           -12.705868 126.381087  -0.101   0.9199
## workclassPrivate           0.022268   0.047986   0.464   0.6426
## workclassSelf-Employed    -0.298238   0.062185  -4.796 1.62e-06 ***
## educational.num            0.325198   0.008839  36.792  < 2e-16 ***
## marital.statusSeparated   -2.067781   0.055542 -37.229  < 2e-16 ***
## marital.statusSingle      -2.538589   0.057549 -44.112  < 2e-16 ***
## occupationBlue-Collar     10.557953 202.413767   0.052   0.9584
## occupationProfessional    11.281082 202.413770   0.056   0.9556
## occupationService         10.705235 202.413769   0.053   0.9578
## occupationWhite-Collar    11.272792 202.413767   0.056   0.9556
## raceAsian-Pac-Islander     0.441763   0.225047   1.963   0.0496 *
## raceBlack                  0.257631   0.208008   1.239   0.2155
```

```
## raceOther                 0.045536   0.318107   0.143   0.8862
## raceWhite                 0.475000   0.197915   2.400   0.0164 *
## genderMale                0.362103   0.047082   7.691 1.46e-14 ***
## hours.per.week            0.030453   0.001523  19.997  < 2e-16 ***
## us.citizenYes             0.323588   0.074090   4.368 1.26e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 33770  on 30147  degrees of freedom
## Residual deviance: 22502  on 30129  degrees of freedom
## AIC: 22540
##
## Number of Fisher Scoring iterations: 12
```

```
significant = glm.summary$coefficients[,4]<0.05
# knitr::kable(round(glm.summary$coefficients[significant,c(1,4)],3),
#              caption = "Logistic Regression Model Summary")

# predict on test set
glm.probs = predict(glm.fits, data[testid,], type="response")
glm.pred = ifelse(glm.probs > 0.5, 1, 0)

# calculate using different metrics
accuracy = mean(glm.pred == y[testid])
conf.mat = table(y[testid],glm.pred)
tn = conf.mat[1,1]
tp = conf.mat[2,2]
fp = conf.mat[1,2]
fn = conf.mat[2,1]
precision = tp/(tp+fp)
recall = tp/(tp+fn)
f1 = 2*precision*recall/(precision+recall)
fpr = fp/(fp+tn)
tpr.fpr = recall/fpr
log.reg.metrics = round(c(accuracy,precision,recall,fpr,f1,tpr.fpr),3)

null.m = glm(income~1, data=train_data, family = "binomial")
step(glm.fits, trace = F, scope = list(lower=formula(null.m), upper=formula(glm.fits)),
     direction = 'backward')
```

```
##
## Call:  glm(formula = income ~ age + workclass + educational.num + marital.status +
##     occupation + race + gender + hours.per.week + us.citizen,
##     family = "binomial", data = train_data)
##
## Coefficients:
##              (Intercept)                       age           workclassOther
##                -18.21526                   0.03019                -12.70587
##         workclassPrivate   workclassSelf-Employed          educational.num
##                  0.02227                  -0.29824                   0.32520
## marital.statusSeparated     marital.statusSingle    occupationBlue-Collar
##                 -2.06778                  -2.53859                  10.55795
```

```
##  occupationProfessional        occupationService   occupationWhite-Collar
##                11.28108                10.70523                 11.27279
##  raceAsian-Pac-Islander               raceBlack               raceOther
##                 0.44176                 0.25763                  0.04554
##               raceWhite              genderMale          hours.per.week
##                 0.47500                 0.36210                  0.03045
##          us.citizenYes
##                 0.32359
##
## Degrees of Freedom: 30147 Total (i.e. Null);  30129 Residual
## Null Deviance:       33770
## Residual Deviance: 22500     AIC: 22540
```

```
step(null.m, trace = F, scope = list(lower=formula(null.m), upper=formula(glm.fits)),
     direction = 'forward')
```

```
##
## Call:  glm(formula = income ~ marital.status + educational.num + hours.per.week +
##     age + occupation + gender + workclass + us.citizen + race,
##     family = "binomial", data = train_data)
##
## Coefficients:
##             (Intercept)  marital.statusSeparated     marital.statusSingle
##               -18.21526                -2.06778                 -2.53859
##         educational.num          hours.per.week                      age
##                 0.32520                 0.03045                  0.03019
##   occupationBlue-Collar   occupationProfessional        occupationService
##                10.55795                11.28108                 10.70523
##  occupationWhite-Collar              genderMale          workclassOther
##                11.27279                 0.36210                -12.70587
##        workclassPrivate   workclassSelf-Employed           us.citizenYes
##                 0.02227                -0.29824                  0.32359
##  raceAsian-Pac-Islander               raceBlack               raceOther
##                 0.44176                 0.25763                  0.04554
##               raceWhite
##                 0.47500
##
## Degrees of Freedom: 30147 Total (i.e. Null);  30129 Residual
## Null Deviance:       33770
## Residual Deviance: 22500     AIC: 22540
```

```
# scale the predictors and perform one-hot encoding for categorical variables
x = scale(model.matrix(income ~ .-1, data = data))
x = t(na.omit(t(x)))
y.mat = to_categorical(y, 2)

# create model
modnn = keras_model_sequential()

# add layers and compile the model
modnn %>%
  layer_dense(units = 2, activation = "softmax", input_shape = ncol(x)) %>%
  compile(
```

```
    loss = "categorical_crossentropy",
    optimizer = optimizer_rmsprop(),
    metrics = c("accuracy")
  )

# observe model structure
var = c("(Intercept)",colnames(x))
# knitr::kable(var, col.names = "Input Variables/Nodes", caption = "Perceptron Input Variables")
summary(modnn)
```

```
## Model: "sequential_1"
## _________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## =================================================================================
##  dense_1 (Dense)                    (None, 2)                       40
##
## =================================================================================
## Total params: 40
## Trainable params: 40
## Non-trainable params: 0
## _________________________________________________________________________________
```

```
# train the model, iterating on the data in batches of 25 samples
# history = modnn %>%
#   fit(x[-testid,], y.mat[-testid,], epochs = 40, batch_size = 25, validation_split=0.2)
# plot(history)

# evaluate on test set
# score = modnn %>% evaluate(x[testid,], y.mat[testid,], batch_size=25)
y.pred = modnn %>% predict(x[testid,]) %>% k_argmax()

# calculate using different metrics
accuracy = mean((as.numeric(y.pred)) == (y[testid]))
conf.mat = table(y[testid],as.numeric(y.pred))
tn = conf.mat[1,1]
tp = conf.mat[2,2]
fp = conf.mat[1,2]
fn = conf.mat[2,1]
precision = tp/(tp+fp)
recall = tp/(tp+fn)
f1 = 2*precision*recall/(precision+recall)
fpr = fp/(fp+tn)
tpr.fpr = recall/fpr
perc.metrics = round(c(accuracy,precision,recall,fpr,f1,tpr.fpr),3)
(metrics = rbind(log.reg.metrics,perc.metrics))
```

```
##                 [,1]  [,2] [,3]  [,4]  [,5]  [,6]
## log.reg.metrics 0.82 0.675 0.53 0.084 0.594 6.312
## perc.metrics    0.48 0.214 0.41 0.497 0.281 0.826
```

```
rownames(metrics) = c("Logistic Regression","Perceptron")
# knitr::kable(metrics, col.names = c("Accuracy","Precision","Recall/TPR","FPR","F1-Score","TPR/FPR"),
#              caption = "Logistic Regression vs Perceptron Prediction Performance")
```