**Data Analytics (CMP330)**

# **Practical 6 – Linear and Logistic Regression**

Review the lecture slides on Linear and Logistic Regression to familiarise yourself with the algorithms. You should also try out the exercises in the **Week 6 Exercises** slides.

Remember to setup your working directory to your folder where script and data are saved e.g. P:\CMP330\week6

**Part A – Linear regression**

**1. Dataset**

For this analysis, we will use a simulated dataset containing medical expenses for patients in the United States. These data were created for this book using demographic statistics from the U.S. Census Bureau, and thus approximately reflect real-world conditions. **The csv file (insurance.csv) can be found under this week (Week 6) material on Blackboard**.

The insurance.csv file includes 1,338 examples of beneficiaries currently enrolled in the insurance plan, with features indicating characteristics of the patient as well as the total medical expenses charged to the plan for the calendar year. The features are:

* age: This is an integer indicating the age of the primary beneficiary (excluding those above 64 years, since they are generally covered by the government).
* sex: This is the policy holder's gender, either male or female.
* bmi: This is the body mass index (BMI), which provides a sense of how over or under-weight a person is relative to their height. BMI is equal to weight (in kilograms) divided by height (in meters) squared. An ideal BMI is within the range of 18.5 to 24.9.
* children: This is an integer indicating the number of children / dependents covered by the insurance plan.
* smoker: This is yes or no depending on whether the insured person regularly smokes tobacco.
* region: This is the beneficiary's place of residence in the U.S., divided into four geographic regions: northeast, southeast, southwest, or northwest.

**2. Exploring and preparing the data**

As we have done before, we will use the read.csv() function to load the data for analysis. We can safely use stringsAsFactors = TRUE because it is appropriate to convert the three nominal variables to factors:

**insurance <- read.csv("insurance.csv", stringsAsFactors = TRUE)**

The str() function confirms that the data are formatted as we had expected:

**str(insurance)**

Since the dependent variable is charges, let's take a look to see how it is distributed:

**summary(insurance$charges)**

Because the mean value is greater than the median, this implies that the distribution of insurance charges is right-skewed. We can confirm this visually using a histogram:

**hist(insurance$charges)**

The large majority of individuals in our data have yearly medical expenses between zero and $15,000, although the tail of the distribution extends far beyond this.

Another problem at hand is that regression models require that every feature is numeric, yet we have three factor type variables in our data frame. We will see how R's linear regression function treats our variables shortly. From the summary() output, we know that region has four levels, but we need to take a closer look to see how they are distributed.

**table(insurance$region)**

From the output, we see that the data have been divided nearly evenly among four geographic regions.

**2. Exploring relationships among features – the correlation matrix**

Before fitting a regression model to data, it can be useful to determine how the independent variables are related to the dependent variable and each other. A correlation matrix provides a quick overview of these relationships. Given a set of variables, it provides a correlation for each pairwise relationship. To create a correlation matrix for the four numeric variables in the insurance data frame, use the cor() command:

**cor(insurance[c("age", "bmi", "children", "charges")])**

At the intersection of each row and column pair, the correlation is listed for the variables indicated by that row and column. None of the correlations in the matrix are considered strong, but there are some notable associations. For instance, age and bmi appear to have a moderate correlation, meaning that as age increases, so does bmi. There is also a moderate correlation between age and charges, bmi and charges, and children and charges.

**3. Visualizing relationships among features – the scatterplot matrix**

It can also be helpful to visualize the relationships among features, perhaps by using a scatterplot. We can use R's graphical capabilities to create a scatterplot matrix for the four numeric features: age, bmi, children, and charges. Here, we'll limit the insurance data frame to the four numeric variables of interest:

**pairs(insurance[c("age", "bmi", "children", "charges")])**

If we add more information to the plot, it can be even more useful. An enhanced scatterplot matrix can be created with the **pairs.panels()** function in the psych package. If you do not have this package installed, type **install.packages("psych")** to install it on your system then load it using the **library(psych)** command. Then, we can create a scatterplot matrix as we had done previously:

**install.packages("psych")**

**library(psych)**

**pairs.panels(insurance[c("age", "bmi", "children", "charges")])**

Above the diagonal, the scatterplots have been replaced with a correlation matrix. On the diagonal, a histogram depicting the distribution of values for each feature is shown. Finally, the scatterplots below the diagonal are now presented with additional visual information. The oval-shaped object on each scatterplot is a correlation ellipse. It provides a visualization of how strongly correlated the variables are. The dot at the centre of the ellipse indicates the point of the mean value for the x axis variable and y axis variable. The correlation between the two variables is indicated by the shape of the ellipse; the more it is stretched, the stronger the correlation. An almost perfectly round oval, as with bmi and children, indicates a very weak correlation (in this case 0.01).

**4. Training a model on the data**

To fit a linear regression model to data with R, the **lm()** function can be used. This is included in the stats package, which should be included and loaded by default with your R installation. The following command fits a linear regression model called **ins\_model**, which relates the six independent variables to the total medical charges. The R formula syntax uses the tilde character ~ to describe the model; the dependent variable charges goes to the left of the tilde while the independent variables go to the right, separated by the + sign. There is no need to specify the regression model's intercept term, as it is assumed by default:

**ins\_model <- lm(charges ~ age + children + bmi + sex + smoker + region, data = insurance)**

After building the model, simply type the name of the model object to see the estimated beta coefficients:

**ins\_model**

Because the **“.”** character can be used to specify all features (excluding those already specified in the formula), the following command is equivalent to the preceding command:

**ins\_model\_1 <- lm(charges ~ . , data = insurance)**

After building the model, simply type the name of the model object to see the estimated beta coefficients, you should see the same outputs as the outputs from **ins\_model**:

**ins\_model\_1**

The estimated beta coefficients indicate the increase in charges for an increase of one in each of the features when the other features are held constant. For instance, for each year that age increases, we would expect $256.90 higher medical expenses on average, assuming everything else is equal. Dummy coding allows a nominal feature to be treated as numeric by creating a binary variable for each category of the feature, which is set to 1 if the observation falls into that category or 0 otherwise. For instance, the sex variable has two categories, male and female. This will be split into two binary values, which R names sexmale and sexfemale. For observations where sex = male, then sexmale = 1 and sexfemale = 0; if sex = female, then sexmale = 0 and sexfemale = 1. The same coding applies to variables with three or more categories. The four-category feature region can be split into four variables: regionnorthwest, regionsoutheast, regionsouthwest, and regionnortheast.

When adding a dummy-coded variable to a regression model, one category is always left out to serve as the reference category. The estimates are then interpreted relative to the reference. In our model, R automatically held out the sexfemale, smokerno, and regionnortheast variables, making female non-smokers in the northeast region the reference group. Thus, males have $131.30 less medical costs each year relative to females and smokers cost an average of $23,848.50 more than non-smokers. Additionally, the coefficient for each of the other three regions in the model is negative, which implies that the northeast region tends to have the highest average medical expenses.

**5. Evaluating model performance**

The parameter estimates we obtained by typing **ins\_model** tell us about how the independent variables are related to the dependent variable, but they tell us nothing about how well the model fits our data. To evaluate the model performance, we can use the **summary()** command on the stored model:

**summary(ins\_model)**

1. The Residuals section provides summary statistics for the errors in our predictions, some of which are apparently quite substantial. Since a residual is equal to the true value minus the predicted value, 50 percent of errors fall within the 1Q and 3Q values (the first and third quartiles), so the majority of predictions were between $2,850 over the true value and $1,400 under the true value.
2. The stars (for example, \*\*\*) indicate the predictive power of each feature in the model. A common practice is to use a significance level of 0.05 to denote a statistically significant variable. If the model had few features that were statistically significant, it may be cause for concern, since it would indicate that our features are not very predictive of the outcome. Here, our model has several significant variables, and they seem to be related to the outcome in logical ways.
3. The Multiple R-squared value (also called the coefficient of determination) provides a measure of how well our model as a whole explains the values of the dependent variable. It is similar to the correlation coefficient in that the closer the value is to 1.0, the better the model perfectly explains the data. Since the R-squared value is 0.7494, we know that nearly 75 percent of the variation in the dependent variable is explained by our model.

**Part B – Logistic regression**

**1. Convert continuous variable to categorical variable**

We will use the same data set as in Part A. For logistic regression, we will need to convert the dependent variable from numerical variable to categorical variable. Here we use **cut()** function:

**insurance$charges\_new <- cut(insurance$charges, breaks = quantile(insurance$charges, probs = seq(0,1,1/4)), labels = c('Low','Medium','Medium-high','High'), include.lowest = TRUE)**

‘include.lowest’ is set to TRUE to avoid NAs.

**str(insurance)**

New dependent variable – charges\_new is created using the Quantiles of the continuous variable charges and the new dependent variable is a factor which contains four levels - 'Low','Medium','Medium-high','High'.

**summary(insurance$charges\_new)**

The frequencies of each value of the new dependent variable are almost the same, and there is no NAs.

**2. Creating training and test datasets**

The library ‘caTools’ will be used in creating training and test datasets. The **sample.split ()** function will be used to perform the training, test dataset split. A seed value is set to allow the replication of the randomization process.

**install.packages("caTools")**

**library(caTools)**

**set.seed(1)**

**split = sample.split(insurance $charges\_new, SplitRatio = 0.7)**

**insurance\_train = subset(insurance, split == TRUE)**

**insurance\_test = subset(insurance, split == FALSE)**

To check the size of the training and test dataset:

**str(insurance\_train)**

**str(insurance\_test)**

As expected, the resulting **insurance\_train** object is a vector of 936 values (70% of the total 1338 values) and the **insurance\_test** object is a vector of 402 values (30% of the total 1338 values)

To check the distribution of the training and test dataset:

**summary(insurance\_train$charges\_new)**

**summary(insurance\_test$charges\_new)**

Both the training and test datasets had similar distributions of **charges\_new**, so we can now build our logistic regression model.

As we have 4 classes in our dependent variable, we need to use Multinomial Logistic Regression. It is a classification method that generalizes logistic regression to multiclass problems, i.e. with more than two possible discrete outcomes.

**3. Training a model on the training data**

There are many R packages to fit this model but here we are going to use the **nnet** package. We will only select three numerical variables to train the model.

**install.packages("nnet")**

**library(nnet)**

**multi\_lg <- multinom(charges\_new ~ age + sex + bmi, data = insurance\_train)**

The summary can be found using:

**summary(multi\_lg)**

**4. Evaluating model performance**

To apply the Multinomial Logistic Regression model to the test dataset, we use the **predict()** function as below:

**predicted <-predict(object=multi\_lg, newdata=insurance\_test, type="class")**

This creates a vector of predicted class values, which we can compare to the actual class values using **confusionMatrix()** function from **caret** package. Install this package using the command:

**install.packages("caret")**

To see the confusion matrix result, we use the command:

**library(caret)**

**confusionMatrix(predicted, insurance\_test$charges\_new)**