Problem Set 1

Shuai Yuan

January 17, 2020

Question 1

In terms of definition, supervised learning is simply a process of learning algorithm from the training dataset, while unsupervised learning is modeling the underlying or hidden structure or distribution in the data in order to learn more about the data.

As for supervised machine learning, it normally has the predictor feature measurement X associated with Y. The relationship between X and Y can be expressed in a function of X.

And our target is to find a proper function based on observed pairs of X and Y. These paired X and Y are also known as the training datasets. In other words, we are trying to find an optimal model to fit the training data, or the paired X and Y, in a function of X.

In addition, these training data are usually well "labeled" during the generating process. We usually know exactly how the datasets are generated, which means some data is already tagged with the correct answer. Therefore, in this way, supervised learning is a simpler method in terms of computational complexity. Meanwhile, if the data pairs are labelled properly, the accuracy of supervised machine learning can be well guaranteed.

And we train the machine with the proper training datasets, we can now estimate the function. We can easily predict the outcome Y with other datasets that are different from the training datasets. Being able to predict the outcome accurately is our goal to conduct supervised machine learning.

Typical examples of supervised machine learning include classification, regression, linear regression and support vector machine. It is often used for export systems in image recognition, speech recognition, forecasting, financial analysis and training neural networks and decision trees, etc.

As for unsupervised machine learning, it does not have a Y associated with the predictor feature measurement X. To be more specific, there is no Y in the unsupervised machine learning world. As a result, of course we can't even find the relationship between X and Y.

And basically there is no target for unsupervised machine learning. We don't even know if all the Xs can form a pattern or not before the machine itself identifies one.

In addition, there is no such thing as training data for unsupervised machine learning. Usually the algorithms are used against data to be tested directly. And we generally have no idea of how these datasets are generated. In other words, these datasets are not labelled during the collection process. Therefore, in this way, unsupervised learning is usually a much more complicated method in terms of computational complexity. Meanwhile, since all the data pairs are not labelled, the accuracy and trustworthiness of supervised machine learning can't be ensured.

Given all the unknown situations, the goal of approaching these datasets is to examine, explore and figure a pattern for these Xs underlying the dataset. To be more specific, it is a data structure exploration process.

Typical examples of unsupervised machine learning generally include clustering, association, and K-means. It is often used to pre-process the data, during exploratory analysis or to pre-train supervised learning algorithms.

Question 2

2a. The population regression model is: $mpg_i = \beta_0 + \beta_1 * cyl_i + \epsilon_i$.

This model is estimating the impact of the number of cylinders on the number of miles per gallon can sustain.

The output in this model is miles per gallon (mpg), and two parameter values are β_0 = 37.88, β_1 = -2.88.

As for the interpretation of β_0 = 37.88, it simply means when there is no cylinder for a car, the number of miles one gallon can sustain for the car is 37.88, which is of course not practical.

As for the interpretation of β_1 = -2.88 , it means that when the number of cylinders increases by 1 unit, the miles one gallon can sustain decreases by 2.88 units, and the result is statistically significant.

```r
model_2a <- lm(mpg ~ cyl, data = mtcars)
tidy(model_2a)%>%kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	37.88458	2.0738436	18.267808	0
cyl	-2.87579	0.3224089	-8.919699	0

2b. The population regression function is:

$mpg_i = \beta_0 + \beta_1 * cyl_i + \epsilon_i$, where β_0 = 37.88, β_1 = 2.88 based on our estimation above.

2c. By adding weight to the specification, the regression model now changes to:

$mpg_i = \beta_0 + \beta_1 * cyl_i + \beta_2 * wt_i + \epsilon_i$

Comparing (a) and (c), the intercepts, β_0, are basically the same, but the intercept in c is slightly higher. As for the β_0 = 39.68 in (c), it simply means when there is no cylinder and no weight for a car, the number of miles one gallon can sustain for the car is 39.68, which is not practical.

After adding weight (wt) to the regression model, the coefficient size of β_1 decreases by nearly 1.4 from -2.88 to -1.51, but still negative and statistically significant. The β_1 = -1.51 in (c) means that holding other variables constant, when the number of cylinders increases by 1 unit, the miles one gallon can sustain decreases by 1.5 units. The impact of cylinders has decreased with the new variables added.

Looking at β_2 = -3.19 before wt, we know that holding other variables constant, when the weight of cars increases by 1 unit, the miles one gallon can sustain decreases by 3.19 units.

```r
model_2c <- lm(mpg ~ cyl + wt, data = mtcars)
tidy(model_2c)%>%kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	39.686262	1.7149840	23.140893	0.0000000
cyl	-1.507795	0.4146883	-3.635972	0.0010643
wt	-3.190972	0.7569065	-4.215808	0.0002220

2d. After adding the interaction term, the model now changes to:

$mpg_i = \beta_0 + \beta_1 * cyl_i + \beta_2 * wt_i + \beta_3 * cyl_i * wt_i + \epsilon_i$

By looking at the regression result, we can see that β_1 and β_2 are still negative, which means that the number of cylinders and the weight of the car are still negatively associated with the number of miles one gallon can sustain. However, the magnitude of the parameters changed. Meanwhile, β_0 is still positive, but the absolute value of β_0 also grows.

Theoretically, the interaction term is trying to capture how the two variables affect each other. We are imposing that the effect of number of cylinders is conditional on weight and vice versa. Here it just means that the negative effect of increasing either the number of cylinders or weight can be compromised by the increasing the other by looking at the positive parameter before the interaction term.

```r
model_2d <- lm(mpg ~ cyl + wt + I(cyl * wt), data = mtcars)
tidy(model_2d)%>%kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	54.3068062	6.127535	8.862749	0.0000000
cyl	-3.8032187	1.005028	-3.784193	0.0007472
wt	-8.6555590	2.320122	-3.730648	0.0008610
I(cyl * wt)	0.8083947	0.327322	2.469723	0.0198824

Question 3

3a. The population regression function is:

$wage_i = \beta_0 + \beta_1 * age_i + \beta_2 * (age_i)^2 + \epsilon_i$

By looking at the result, we can see β_1 = 5.29, and β_2 = -0.05. Note that β_1 is positive, and β_2 is negative, it means that when there are two ways that age can affect wage, and the direction depends on one which is stronger.

Therefore, it means that when age increases at the very beginning, the 5.29 * age is greater than -0.05 * $(age)^2$. As age grows, wage grows. However, when the situation changes after a certain point, as age grows, wage declines. It is hard to interpret the coefficient before the age squared alone as it is associated with the value of age. It will be clear if we take a partial derivative with respect to age.

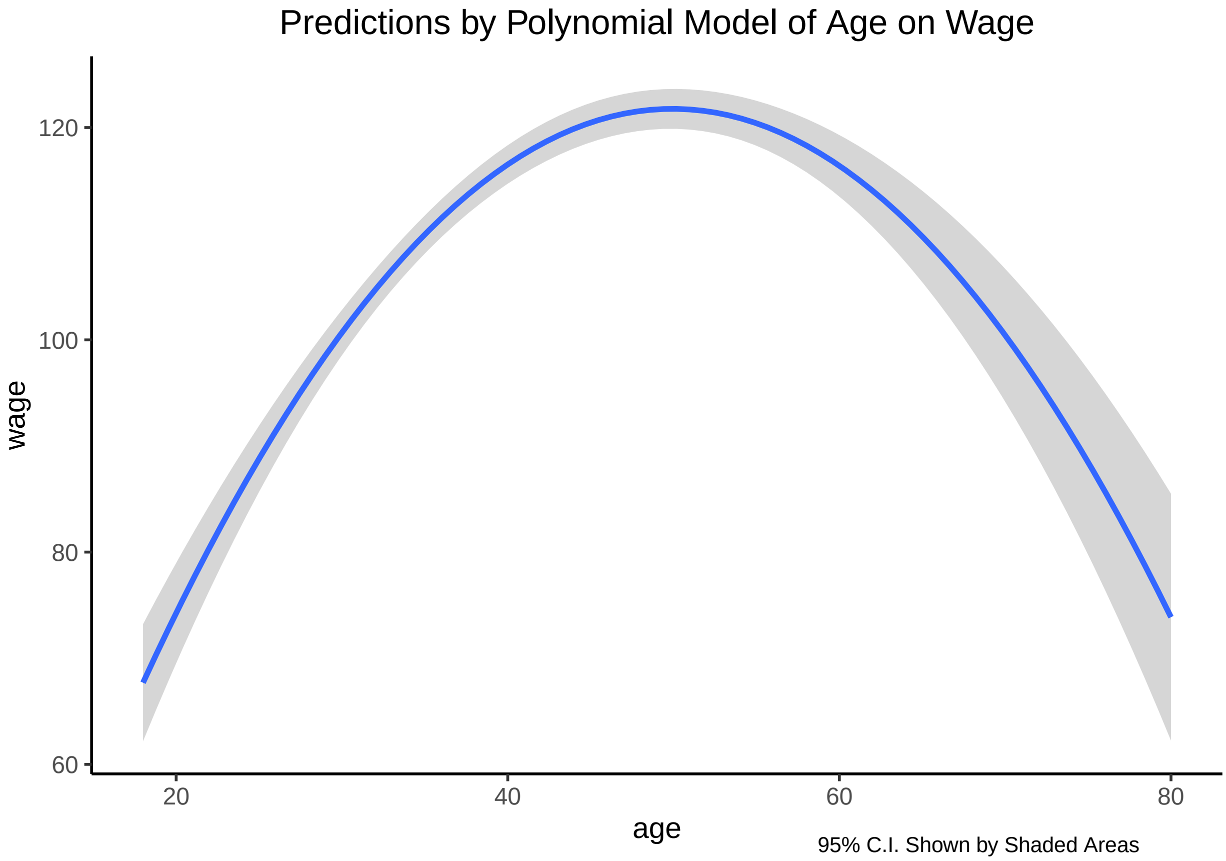
As for β_0, the interpretation for it is meaningless here, as wage at the age of 0 is normally not practical.

```r
model_3a <- lm(wage ~ age + I(age ^ 2), data = wage)
tidy(model_3a)%>%kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-10.4252243	8.1897803	-1.272955	0.2031326
age	5.2940300	0.3886886	13.620236	0.0000000
I(age^2)	-0.0530051	0.0044318	-11.960103	0.0000000

3b. Note that for the graph 95% is indicated by shaded areas as is marked in the tag of the graph.

```r
ggplot(wage, aes(x = age, y = wage)) +
  geom_smooth(formula = y~poly(x,2), method = "lm", se = TRUE, level = 0.95) +
  labs(title = "Predictions by Polynomial Model of Age on Wage",
       tag = "95% C.I. Shown by Shaded Areas")
```



3c. The estimated function is concave. When age increases at the very beginning, the 5.29 * age is greater than -0.05 * $(age)^2$. As age grows, wage grows. However, when the situation changes at around 50, as age grows, wage declines. The turning point is around 50 according to the graph.

By fitting a polynomial regression, we are assuming that the relationship between age and wage is very complex and definitely not linear.

3d. In this example, if we use a simple linear regression model, we will definitely conclude that wage grows as age grows, but according to common sense and the discussion above we know that it is not the case. By using polynomial regression model, we are actually finding the turning point of wage declining affected by age.

In general, polynomial regression has higher order of variable and greater flexibility statistically. Polynomial provides the best approximation of the relationship between the dependent and independent variable. A Broad range of function can be fit under it. On the other hand, when only the first order is allowed, linear model tries to simplify the relationship between Y and Xs.

Substantively, when relating the model to reality, it makes more sense that in most cases the relationship between two variables are not linear. Like the example in question 3. We all know that as a person ages, he or she will retire and the wage will definitely decline. However, it is worth noting that the result of linear regression is much more easier to interpret compared to that of the polynomial regression as the higher order coefficeint interpretation depends on the lower order change.