Intro to Machine Learning: Problem Set 1

Ipek Cinar

1/16/2020

Statistical and Machine Learning

Describe in 500-800 words the difference between supervised and unsupervised learning.

In today's world and technology, machine learning provides us the tools to analyze vast volumes of big data through the usage of data-driven models and algorithms. For instance, while analyzing a large dataset of speeches would require immense amounts of time, machine learning algorithms allows us to process the data at a much faster rate with the ability to automate along with (hopefully) good accuracy. While the process is complex withi itself, it is divided into two substantive/main areas: 1) supervised and 2) unsupervised machine learning.

Machine learning process begins with inputting training data into a selected algorithm. Although both methods are used to learn from given input data (X) and widely used in real applications, there are some distinct differences in their usage as well as implementations. The main difference emerges from the type of input data (known/labeled versus unknown/unlabeled) provided to the machine learning algorithm. In supervised learning we work with fully "labelled" data. That means, when we train our data, we already know all the correct answers and we create our model by "learning" from these correct answers. It is quite similar to learning a topic under supervision where someone is present judging whether you're getting the right answer. In supervised learning, we have our data with independent X's as input and their corresponding Y output values (i.e. X = 50 dog and 50 cat images and Y = dog or cat). We aim to find a model (mapping function) to predict Y value with the given X's (Y = f(X)) (i.e. given dog or cat image, predicting whether it is a dog or cat). In order to do that, generally we use some part of the data for training the model and other part for testing. During training, model rigorously learns from the correct answers in the training data. During testing we compare the model's predicted y-outputs with actual outputs and evaluate the accuracy of our model. Not surprisingly, we want to create a model where we predict the exact same values (with the actual values) and get the best accuracy (e.g. 100%).

But we also want this model to be a general, not specific for the trained dataset. As mentioned above, in supervised learning we need labelled data, therefore, supervised learning is best suited to problems where there is a set of available reference points or a ground truth with which to train the model. Thus, since supervised learning allows us to collect and/or produce data output from previous experiences, supervised learning is mostly used in classification and regression problems: classification problems ask the model to predict a discrete value, identifying the input data as the member of a particular class, or group (i.e. dog and cat example above) and regression problems look at continuous data to predict the Y values from given X's. Having said that, data with reference points or a ground truth aren't always available in real life scenarios and in that case we generally use unsupervised learning.

In unsupervised learning, we work with "unlabeled" data. Unlike the supervised learning, we only have input data (X's) and no corresponding output data (Y). Therefore, rather than doing a prediction, in unsupervised learning we try to model the underlying or hidden structure or unknown patterns or distribution in the data in order to learn more about the data. In other words, unsupervised machine learning allows us to find and explore all kinds of unknown patterns inherent in our data. These patterns and distributions can be used on finding features that could be useful for categorization.

Unsupervised learning is mostly used in clustering, association and anomaly detection problems. For example, let's say we have a country dataset with different independent metrics/features. We can use unsupervised learning on this dataset to analyze or discover patterns among countries or country groups. We can also use different unsupervised models with different initial parameters to see different patterns. Unlike the supervised learning, in unsupervised learning we use all dataset as our training dataset and all the input data is analyzed and labeled in real time during the learning. Since there is a no ground truth answers/clusters

to compare, in unsupervised learning either there is no validation or there are some statistical or intuitive approximates for the evaluation of the models. Thus, comparing to supervised learning, unsupervised learning can be considered and blamed as less accurate and trustworthy method. However, it is important to keep in mind that obtaining unlabeled data is in general much easier than gathering label data as clean and perfectly labeled datasets aren't easy to come by which requires than significant manual intervention.

Linear Regression

a)Predict miles per gallon (mpg) as a function of cylinders (cyl). What is the output and parameter values for your model.

```
library(caret)
# names(mtcars)
# head(mtcars)
# Simple Linear Regression 1 (Version 1)
lm_model_1_1 <- lm(mpg ~ cyl, data = mtcars)
summary(lm_model_1_1)
```

```
##
## Call:
## lm(formula = mpg ~ cyl, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.9814 -2.1185  0.2217  1.0717  7.5186
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.8846     2.0738   18.27  < 2e-16 ***
## cyl          -2.8758     0.3224   -8.92 6.11e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.206 on 30 degrees of freedom
## Multiple R-squared:  0.7262, Adjusted R-squared:  0.7171
## F-statistic: 79.56 on 1 and 30 DF,  p-value: 6.113e-10
```

```
# Simple Linear Regression 2 (Version 2)
 lm_model_1_2 <- train(mpg ~ cyl,
               data = mtcars,
               method = "lm")
# lm_model_1_2["finalModel"]
# Comparing lm_model_1 and lm_model_2 should reveal same intercept and coefficients as
# they both use linear regression to predict mpg as a function of cyl using
# linear regression, and indeed they do.
```

In simple linear regression, the model used to describe the relationship between a single dependent variable y and a single independent variable x is $y = \beta_0 + \beta_1 x + \varepsilon$. $\beta_0 and \beta_1$ are referred to as the model parameters (where ε is a probabilistic error term that accounts for the variability in y that cannot be explained by the linear relationship with x). Hence, in this model where we predict mpg as a function of cyl, parameter values are 1) the intercept (β_0) which equals 37.885 and 2) the coefficient for cyl (β_1) which equals -2.876. The output variable (also known as dependent variable) and its associated value in this the model is average predicted value for mpg (miles per gallon).

b) Write the statistical form of the simple model in the previous question (i.e., what is the population regression function?).

For a (simple) linear regression model where there is a single dependent variable and a single independent variable of interest, the population regression function is simply: $Y = \beta_0 + \beta_1 X$ swhere the β_0 represents the intercept coefficient while $\beta_1 x$ represents the slope coefficient on X.

Hence, in the case of our model (predicted in 1a above), the population regression function/statistical form of the simple model we have from the previous question is the following:

$$mpg = 37.885 + (-2.876) * cyl$$

c) Add vehicle weight (wt) to the specification. Report the results and talk about differences in coefficient size, effects, etc.

```
library(stargazer)
# Simple Linear Regression 3 (Version 1)
lm_model_3_1 <- lm(mpg~cyl + wt, data=mtcars)
summary(lm_model_3_1)
```

```
##
## Call:
## lm(formula = mpg ~ cyl + wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2893 -1.5512 -0.4684  1.5743  6.1004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  39.6863     1.7150  23.141  < 2e-16 ***
## cyl          -1.5078     0.4147  -3.636 0.001064 **
## wt           -3.1910     0.7569  -4.216 0.000222 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.568 on 29 degrees of freedom
## Multiple R-squared:  0.8302, Adjusted R-squared:  0.8185
## F-statistic: 70.91 on 2 and 29 DF,  p-value: 6.809e-12
```

```
# Simple Linear Regression 3 (Version 2)
# lm_model_3_2 <- train(mpg ~ cyl + wt,
              # data = mtcars,
              # method = "lm")
# View(lm_model_2)
# lm_model_3_2["finalModel"]
# Comparing lm_model_1 and lm_model_2 should reveal same intercept and coefficients as
# they both use linear regression to predict mpg as a function of cyl and wt using
# linear regression, and indeed they do.
```

As we can see from Table 1, our (linear regression) model which predicts mpg as a function of cyl and wt reveals the following information: - The intercept is 39.686. - The coefficient associated with cyl is -1.508 - The coefficient associated with wt is -3.191. In return, giving us the following statistical model: $mpg = 37.885 - 1.508 * (cyl) - 3.191 * (wt)$

Then, according to the model, we can conclude the following: -The coefficient associated with cyl (number of cylinders) indicates that for every additional unit increase in number of cylinders you can expect miles per gallon to decrease by an average of 1.508. -The coefficient associated with wt (weight) indicates that for every additional unit increase in the weight of the car (in lbs) you can expect miles per gallon to decrease by an average of 3.191. -The intercept indicates that when both number of cylinders and weight

associated with the car is 0, miles per gallon is predicted to be 39.686.

d) Interact weight and cylinders and report the results. What is the same or different? What are we theoretically asserting by including a multiplicative interaction term in the function?

```
# Simple Linear Regression 3 (Version 1)
lm_model_4_1 <- lm(mpg ~ cyl + wt + (cyl:wt), data=mtcars)
summary(lm_model_4_1)
```

```
##
## Call:
## lm(formula = mpg ~ cyl + wt + (cyl:wt), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2288 -1.3495 -0.5042  1.4647  5.2344
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  54.3068     6.1275   8.863 1.29e-09 ***
## cyl          -3.8032     1.0050  -3.784 0.000747 ***
## wt           -8.6556     2.3201  -3.731 0.000861 ***
## cyl:wt        0.8084     0.3273   2.470 0.019882 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.368 on 28 degrees of freedom
## Multiple R-squared:  0.8606, Adjusted R-squared:  0.8457
## F-statistic: 57.62 on 3 and 28 DF,  p-value: 4.231e-12
```

```
# We could have also written the above function as below, since the operator : designates
# the interaction between two variables, whereas * designates the interaction between
# the two variables, plus the main effects.
# lm_model_4_2 <- lm(mpg ~ cyl*wt, data=mtcars)
# lm_model_4_2
# We could also use stargazer to report regression results as a table.
# stargazer(lm_model_4_1, title="Results", header=FALSE, type='latex')
```

In contrast to the model we had in the previous question, we now included an interaction term in our model. Interaction terms indicate that the effect of one predictor depends on the value of another predictor. Hence, presence of a significant interaction indicates that the effect of one predictor variable (in our case either cyl or wt) on the outcome variable (in our case mpg) is different at different values of the other predictor variable (again in our case either cyl or wt.

As we can see from Table 2, including an interaction term changes the interpretation of all the coefficients (as their values change as well). If there were no interaction term, coefficient associated with cyl (lets denote as β_1) would be interpreted as the unique effect of number of cylinders on miles per gallin But the interaction means that the effect of number of cylinders on miles per gallon is different for different values of weight So the unique effect of number of cylinders on miles per gallon is not limited to β_1 but also depends on the values of coefficient associated with the interaction term (lets denote as β_3) and weight. The unique effect of number of cylinders is represented by everything that is multiplied by number of cylinders in the model: $\beta_1 + \beta_3 * weight$. β_1 is now interpreted as the unique effect of number of cylinders on mile sper gallon only when weight $= 0$. Since adding interaction term changed the values of β_1 and β_2, the effect of number of cylinders on mile sper gallon is now $-3.8032 + 0.8084 * weight$. The same logic applies when it comes to interpreting the results for weight.

Non-Linear Regression

a) Fit a polynomial regression, predicting wage as a function of a second order polynomial for age. Report the results and discuss the output (hint: there are many ways to fit polynomials in R, e.g., I, ^, poly(), etc.)

```
library(tidyverse)
data_wage <- read_csv("~/Desktop/Intro to Machine Learning/Problem Sets/PSet 1/wage_data.csv")
# First way to fit a polynomial regression, predicting wage as afunction of a second order
# polynomial for age:
poly_1 = lm(wage ~ poly(age, 2, raw = TRUE), data = data_wage)
# coef(summary(poly_1))
# Second way to fit a polynomial regression, predicting wage as afunction of a second order
# polynomial for age:
poly_2 = lm(wage ~ age + I(age ^2), data = data_wage)
# coef(poly_2)
summary(poly_2)
```

```
##
## Call:
## lm(formula = wage ~ age + I(age^2), data = data_wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -99.126 -24.309  -5.017  15.494 205.621
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -10.425224   8.189780  -1.273    0.203
## age           5.294030   0.388689  13.620   <2e-16 ***
## I(age^2)     -0.053005   0.004432 -11.960   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.99 on 2997 degrees of freedom
## Multiple R-squared:  0.08209,    Adjusted R-squared:  0.08147
## F-statistic:   134 on 2 and 2997 DF,  p-value: < 2.2e-16
```

```
# The first + second way are essentialy  same, they are just different ways to fit polynomials.
#Reporting regression results as a table
```

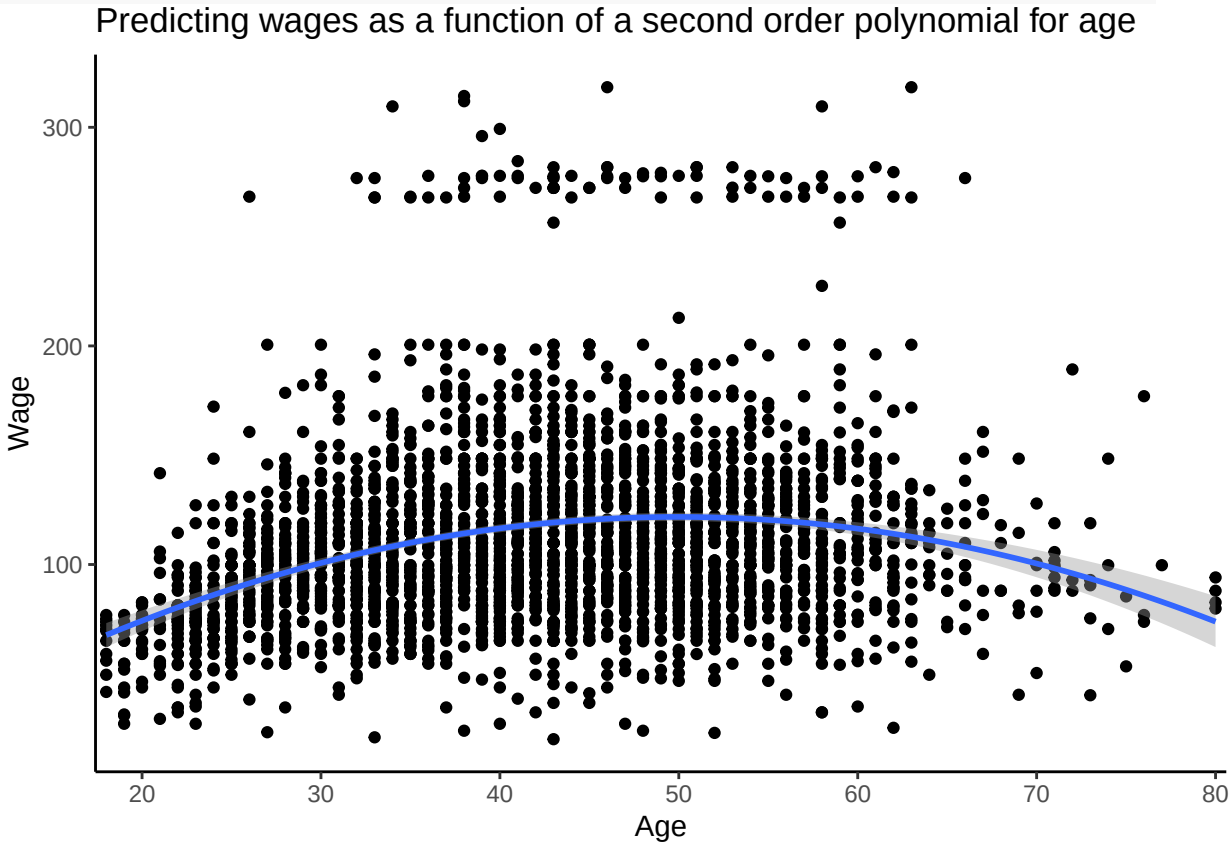
Given the results above, the polynomial regression results in the following statistical model: $wage = -10.425224 + 5.294030 * (age) - 0.053005 * (age^2)$. The first derivative of this function would then tell us how predicted wages change as age increases. $\frac{\partial w}{\partial (age)} = 5.294030 - (2 * 0.053005) * age$. Since the coefficient on age squared is negative, the function has a maximum value and this maximum value occurs at: $(5.294030)/(2 * 0.053005) = 49.939$

Up to (almost) 50 years of age, predicted wages increase as age increases but at a diminishing rate. After 50 years of age, wages decline at an increasing rate. For example, at age = 25, predicted wages rise at the rate of $5.294030 - (25 * 0.10601) = 2.64378$ whereas wages are predicted to rise by just $5.294030 - (40 * 0.10601) = 1.05363$ at the age of 40. After age of 50, wages are not rising at all, and start to fall as age goes beyond 50 years.

b) Plot the function with 95% confidence interval bounds.

```
library(ggplot2)
p <- ggplot(data_wage, aes(x = age, y = wage)) + geom_point()
# print(p) #this is safety check
```

```
p <- p + stat_smooth(method = "lm", level = 0.95, formula = y ~ x + I(x^2), size = 1)+
  theme_classic() + scale_x_discrete("Age", limits = c(20,30,40,50,60,70,80)) +
  ylab("Wage") + ggtitle("Predicting wages as a function of a second order polynomial for age")
print(p)
```



c) Describe the output. What do you see substantively? What are we asserting by fitting a polynomial regression?

As discussed in more detail above (section b), we can confirm from the plot above that up to age of 50 years, predicted wages increase as age increases (but at a diminishing rate). After 50 years of age, wages decline at an increasing rate. By fitting a polynomial regression (rather than linear), we are asserting that the straight line of a linear regression is unable to capture the patterns in our data, and we need to increase the complexity of the model to arrive at better predictions. Rather than a linear line, the curve we are fitting becomes quadratic in nature (as can be seen from the plot above).

Regarding the confidence interval, as we can see from the plot, the confidence interval gets wider as the age progresses (especially after age of 60, it continues to get wider as the age increases). What does this mean? In general, confidence interval describes the uncertainty that is inherent wihthin the estimate, and describes a range of values within which we can be reasonably sure to expect that the true effect actually lies within that range. A 95% confidence interval is a range of values that you can be 95% certain contains the true mean of the population (which is inherently not the same as a range that contains 95% of the values). Therefore, if the confidence interval is relatively narrow, the effect size is known more precisely than a confidence interval that is wider. Hence, as we can see from our own plot, fter the age of 60, the uncertainty is greater in our wage prediction.

d) How does a polynomial regression differ both statistically and substantively from a linear regression (feel free to also generalize to discuss broad differences between non-linear and linear regression)?

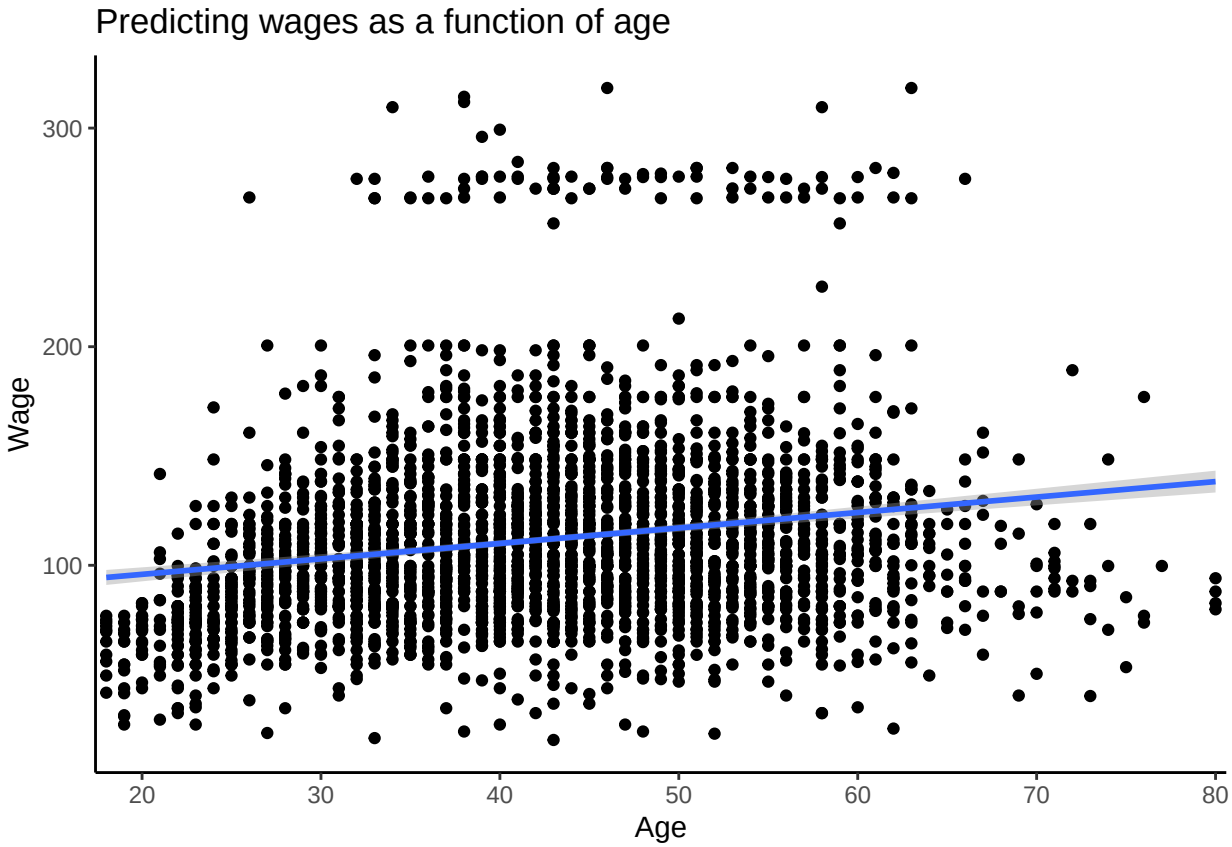
In order to understand how the application of polynomial regression differs from linear regression to data, let us actually fit a linear regression to our data first to both evaluate

```
linear_model = lm(wage ~ age, data = data_wage)
summary(linear_model)
```

```
##
## Call:
## lm(formula = wage ~ age, data = data_wage)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -100.265  -25.115   -6.063   16.601  205.748
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 81.70474    2.84624   28.71   <2e-16 ***
## age          0.70728    0.06475   10.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.93 on 2998 degrees of freedom
## Multiple R-squared:  0.03827,    Adjusted R-squared:  0.03795
## F-statistic: 119.3 on 1 and 2998 DF,  p-value: < 2.2e-16
```

Fitting a linear regression to our data reveals the following model: $wage = 81.70474 + 0.70728 * (age)$. Meaning that, given a year of increase in the age, average wage is expected to increase by 0.70728 in value. However, is this a good fit? Let us see a plot. As we can see from the plot below, the straight line is unable to capture the patterns in the data. In order to overcome this, we need to increase the complexity of our model and hence generate a higher order equation where we can add powers to our linear model; changing our previous linear model from $wage = \beta_0 + \beta_1 * (age)$ to $wage = \beta_0 + \beta_1 * (age) + \beta_2 * (age^2)$.Polynomial regression is a form of linear regression where higher order powers (2nd, 3rd or higher) of an independent variable are also included. How does this model capture the patterns in our data? It is indeed quite clear from the plot we have in section b of the question that the quadratic curve we obtained from applying a polynomial regression data is able to fit the data better than the linear line we have above.

```
p <- ggplot(data_wage, aes(x = age, y = wage)) + geom_point()
p <- p + stat_smooth(method = "lm", level = 0.95, formula = y ~ x, size = 1)+
  theme_classic()+ scale_x_discrete("Age", limits = c(20,30,40,50,60,70,80)) +
  ylab("Wage") + ggtitle("Predicting wages as a function of age")
print(p)
```



However when using polynomial regression (which is also a nonlinear model) to solve the problem of underfitting with linear regresison, we need to be careful. While the curvature allowed by the polynomial regression becomes better at fitting and catching the patterns in the data, we run the risk of overfitting this time if our model ends up picking up on the noise (or known as outliers) in the data given that nonlinear regression is much more sensitive to the outliers and hence the presence of one or two outliers in the data can seriously affect the results of a nonlinear analysis.

In addition, when it comes to the interpretation of coefficients from the non-linear statistical model, the results are not as straightforwards as in the linear coase since the assumption of independent variables being independent from another is violated under polynomical (also non-linear) regression. Rather than saying a unit increase in one of the independent variables leads to a either increase/decrese in the dependent variable in the amount of (associated coefficient with that independen variable) in the case of a linear regression, we need to compute the predicted value of $\frac{\Delta Y(dependent)}{\Delta X(independent)}$ at different values of X to interpret the estimated regression function. (For an example see section c on how the coefficients from the second order polynomical regression predicting wage as a function of age were intepreted - More presicely, we saw how the predicted wage was rising at a higher/increasing rate at the age of 25 than at the age of 40).