Assignment 1

Statistical Learning

Prepared by: Saeedreza Zouashkiani

Professor: Vahid Pourahmadi

Date: October 25, 2020

Note. The codes are written in jupyter notebook, but exported as .py and .pdf files and are all in the assignment directory.

1) Q1 & Q2 of chapter 2.

Q1. For each of parts (a) through (d), indicate whether we would generally expect the performance of a flexible statistical learning method to be better or worse than an inflexible method. Justify your answer.
(a) The sample size *n* is extremely large, and the number of predictors *p* is small.

Better. A more flexible model can fit the data better.
(b) The number of predictors *p* is extremely large, and the number of observations *n* is small.

Worse. The flexible model will overfit the data.
(c) The relationship between the predictors and response is highly non-linear.

Better. With more degrees of freedom, a more flexible method would be a better fit.
(d) The variance of the error terms, i.e. $\sigma^2 = Var(\epsilon)$, is extremely high.

Worse. Flexible methods would fit to the noise and increase the variance.

Q2. Explain whether each scenario is a classification or regression problem, and indicate whether we are most interested in inference or prediction. Finally, provide *n* and *p*.
(a) We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in understanding which factors affect CEO salary.

Regression. Inference. n=500: firms in the US, p=3: profit, number of employees, industry.
(b) We are considering launching a new product and wish to know whether it will be a *success* or a *failure*. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.

Classification. Prediction. n=20: products, p=13: price charged for the product, market budget, competition price, and ten other variables.
(c) We are interested in predicting the % change in the US dollar in relation to the weekly changes in the world stock markets. Hence, we collect weekly data for all of 2012. For each week we record the % change in the dollar, the % change in the US market, the % change in the British market, and the % change in the German market.

Regression. Prediction. n=52 weeks of 2012, p=3: % change in US market, % change in British market, % change in German market.

2) Q5 & Q7 of chapter

Q5. Consider the fitted values that result from performing linear regression without an intercept. In this setting, the *i*th fitted value takes the form

$$\hat{y}_i = x_i\hat{\beta}$$

where

$$\hat{\beta} = (\sum_{i=1}^{n} x_i y_i)/(\sum_{i'=1}^{n} x_{i'}^2)$$

Show that we can write

$$\hat{y}_i = \sum_{i'=1}^{n} a_{i'} y_{i'}$$

What is $a_{i'}$?

Note: We interpret this result by saying that the fitted values from linear regression are linear combinations *of the response values.*

$$\hat{y}_i = x_i \hat{\beta} = x_i \frac{(\sum_{i'=1}^{n} x_{i'} y_{i'})}{(\sum_{k=1}^{n} x_k^2)} = \frac{(\sum_{i'=1}^{n} x_{i'} x_i y_i)}{(\sum_{k=1}^{n} x_k^2)} = \sum_{i'=1}^{n} \left(\frac{x_{i'} x_i}{(\sum_{k=1}^{n} x_k^2)} \right) y_i$$

$$a_{i'} = \frac{x_{i'} x_i}{(\sum_{k=1}^{n} x_k^2)}$$

Q7. It is claimed in the text that in the case of simple linear regression of *Y* onto *X*, the R^2 statistic (3.17) is equal to the square of the correlation between *X* and *Y* (3.18). Prove that this is the case. For simplicity, you may assume that $\bar{x} = \bar{y} = 0$.

$$R^2 = \frac{TSS - RSS}{TSS} = \frac{\sum_{i=1}^{n}(y_i - \bar{y}) - \sum_{i=1}^{n}(y_i - \hat{y}_i)}{\sum_{i=1}^{n}(y_i - \bar{y})} \quad [1]$$

By simplifying the numerator:

$$A = \sum_{i=1}^{n}(y_i - \bar{y}) - \sum_{i=1}^{n}(y_i - \hat{y}_i)$$

$$= \sum_{i=1}^{n}[(y_i - \bar{y}) - (y_i - \hat{y}_i)][(y_i - \bar{y}) + (y_i - \hat{y}_i)]$$

$$= \sum_{i=1}^{n}(\hat{y}_i - \bar{y})\,(2y_i - \hat{y}_i - \bar{y}) \quad [2]$$

By using the formula of $\hat{\beta}_0$ and implementing that into $\hat{y}_i$:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad [3]$$

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i = \bar{y} - \hat{\beta}_1 \bar{x} + \hat{\beta}_1 x_i = \bar{y} + \hat{\beta}_1 (x_i - \bar{x}) \quad [4]$$

Put [3],[4] into [2]:

$$\sum_{i=1}^{n}(\hat{y}_i - \bar{y})\,(2y_i - \hat{y}_i - \bar{y})$$

$$= \sum_{i=1}^{n} \left(\bar{y} + \hat{\beta}_1(x_i - \bar{x}) - \bar{y}\right)\left(2y_i - \bar{y} - \hat{\beta}_1(x_i - \bar{x}) - \bar{y}\right)$$

$$= \sum_{i=1}^{n} \hat{\beta}_1(x_i - \bar{x})\left[2(y_i - \bar{y}) - \hat{\beta}_1(x_i - \bar{x})\right]$$

$$= \hat{\beta}_1 \sum_{i=1}^{n} (x_i - \bar{x})\left[2(y_i - \bar{y}) - \hat{\beta}_1(x_i - \bar{x})\right]$$

$$= \hat{\beta}_1 \left[2\sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y}) - \hat{\beta}_1 \sum_{i=1}^{n} (x_i - \bar{x})^2\right]$$

$$= \hat{\beta}_1 \left[2\sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y}) - \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})\right]$$

$$= \hat{\beta}_1 \left[\sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y})\right]$$

$$= \frac{[\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y})]^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad [5]$$

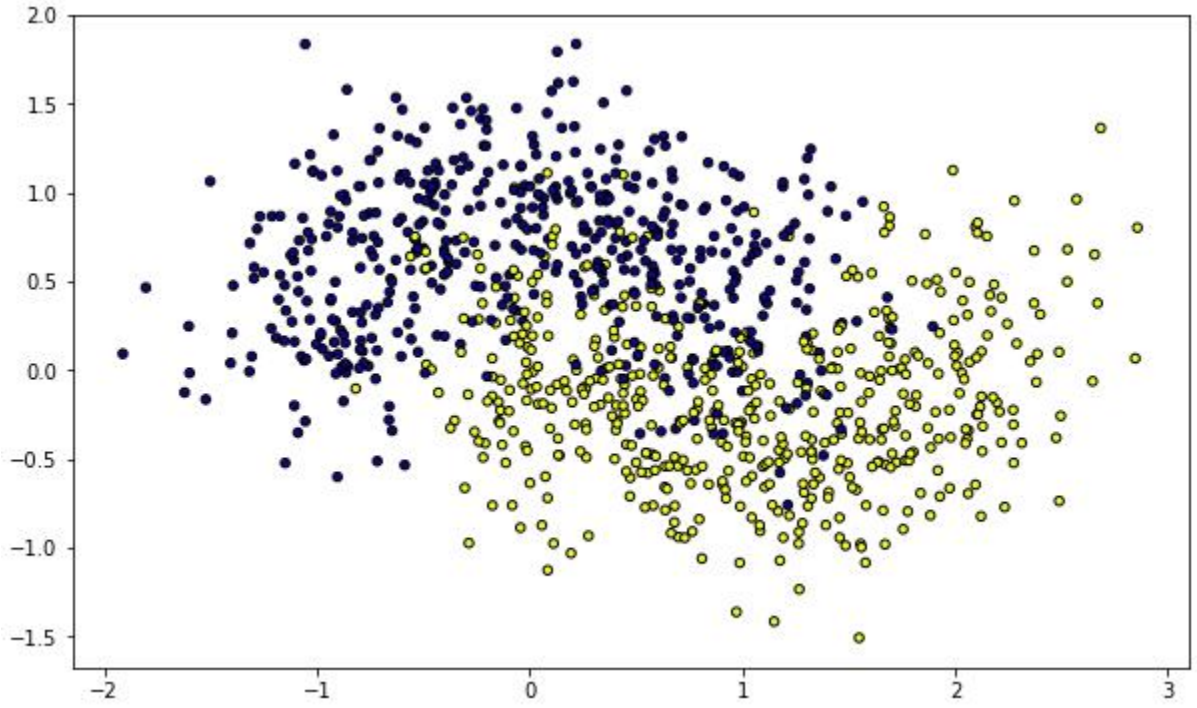
Put [5] into [1]:

$$R^2 = \frac{[\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y})]^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2} = r^2 = [Cor(X, Y)]^2$$

3) Bias-variance tradeoff I:

(a) Create a synthetic dataset (with both features and targets). Use the `make_moons` module with the parameter `noise=0.35` to generate 1000 random samples.

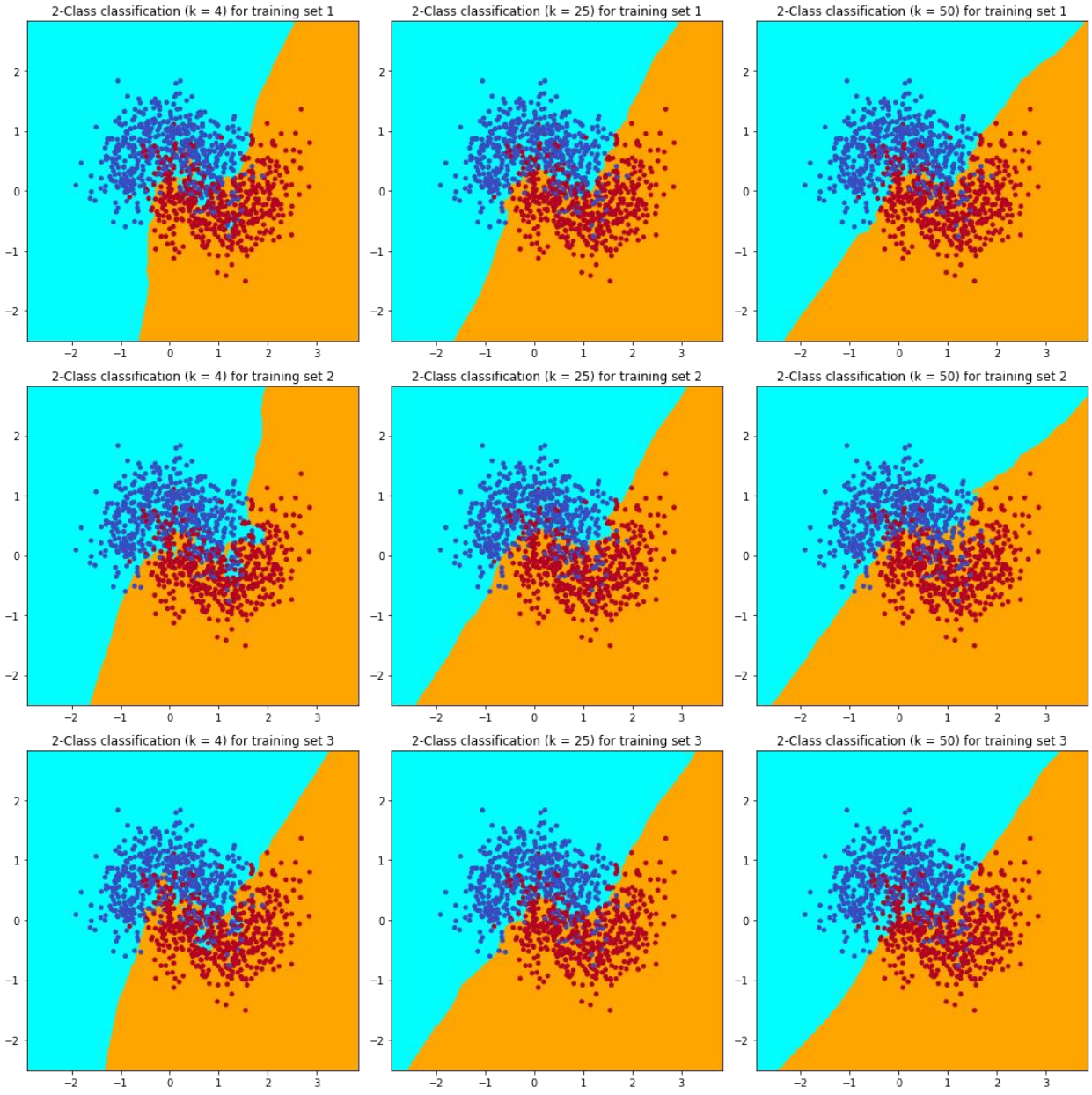
(b) Scatterplot your random samples with each class in a different color.



(c) Create 3 different data subsets by selecting 150 of the 1000 data points at random. For each of these 150-sample datasets, fit three k-Nearest Neighbor classifiers with: $k \in \{4, 25, 50\}$. This will result in 9 combinations (3 datasets, with 3 trained classifiers).

This was done using the `train_test_split` from `sklearn.model_selection` module with a `train_size` parameter of 0.15. It could also be done sampling the data randomly by first shuffling the data.

(d) For each combination of dataset trained classifier, in a 3-by-3 grid, plot the decision boundary (similar in style to Figure 2.15 from *Introduction to Statistical Learning*). Each column should represent a different value of *k* and each row should represent a different dataset.



(e) What do you notice about the difference between the rows and the columns. Which decision boundaries appear to best separate the two classes of data? Which decision boundaries vary the most as the data change?

For a small k (k=4) the model fits rigidly on the training set and for a large k (k=50) the model is very flexible and the decision boundary is almost a straight line. The decision boundary for k=4 as seen in the figure varies the most as the model is desperately trying to fit every point to the model.

(f) Explain the bias-variance tradeoff using the example of the plots you made in this exercise.

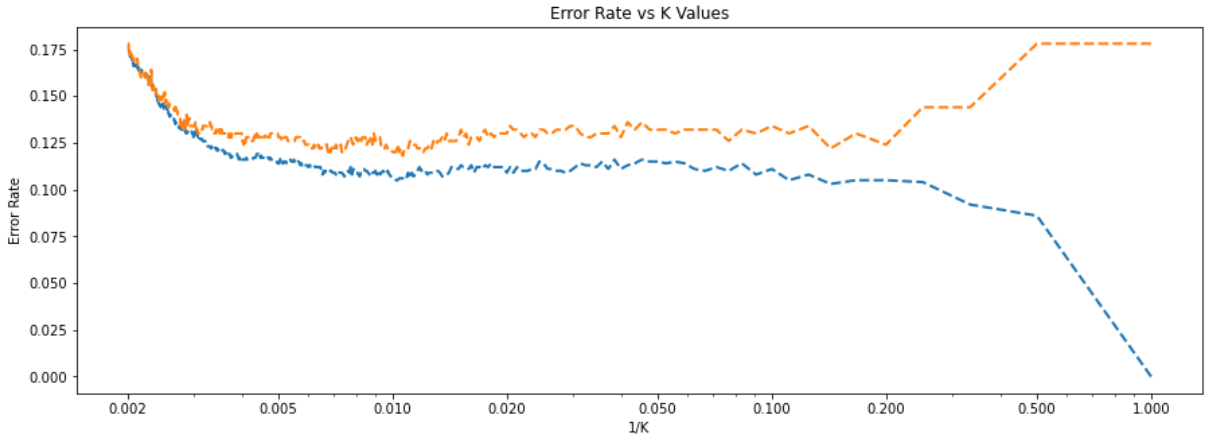
In k-NN classification, to make a new prediction, the model looks at its nearest k neighbors and predicts based on those neighbors. For a small k this means that the model will fit even the noise and the outliers

in the model, resulting in an overfit model and higher variance but low bias. As for large k's the training error will increase (high bias), but the test error might decrease (low variance).

4) Model Selection: which "k" leads to the best model?

(a) Use `make_moons` with same parameters of part "a" to create a new set of 500 random samples. Consider this as the test set while the 1000 samples of part "a" is your train dataset

(b) Use train dataset to train a kNN classifier for k = 1, 2, …, 500. For each "k" compute the test and train error and make a figure like Fig. 2.17 of the ISLR book.



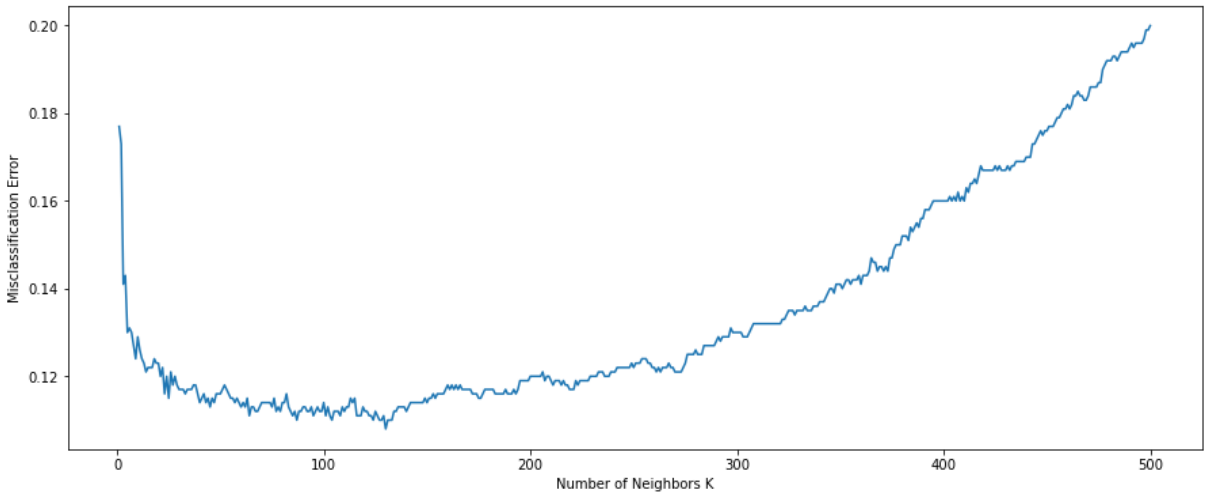
(c) What values of k represent high bias and which represent high variance?

The plot shows the train and test error as the flexibility i.e. 1/K increases. The lower the flexibility i.e. a larger k, will result in a high bias and underfit model, and the higher the flexibility i.e. a smaller k, will result in a high variance model.

(d) What is the optimal value of k and why

There is a couple of ways to approach this problem. One is deciding the optimal value of K based on the figure from the previous part. As seen in the figure smallest and largest k mean in respect an overfit and underfit model, since small k has small train error vs large test error, and large k has large test and train error. We have to choose some K in between. The test and train error between the values of around 0.003 and 0.2 for 1/K doesn't vary much, so we might as well choose a small k due to simplicity which will be about 22,23.

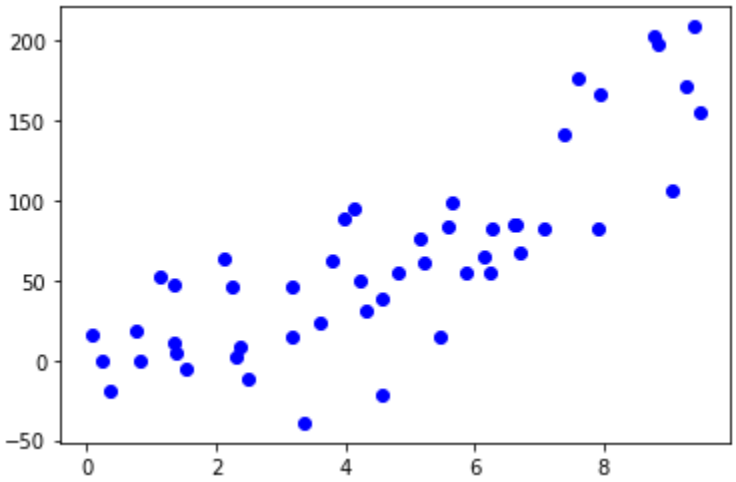
The second way is to use cross validation, in which we use the training set and split in an arbitrary number of folds called k-fold and train the model on (k-1) folds and test it on the remaining fold. After that we average the error to find the final accuracy. An optimal value of 130 was found for K using this method. Note that the misclassification error for K=23 using this method is 0.115 which is about 7% higher than the optimal value which is K=130 at 0.107. So, any K between (20,140) would be acceptable.



5) Regression model:

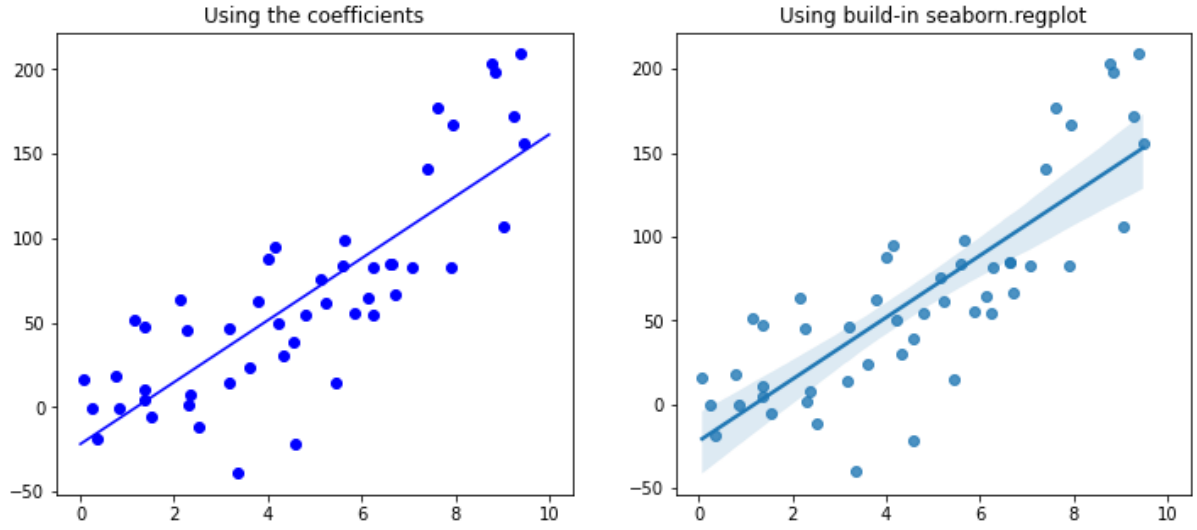
you can *use scikit-learn* `LinearRegression` *module.*

(a) You have the Train and test data below. Visualize them using a scatter plot.



Scatterplot of train and test data

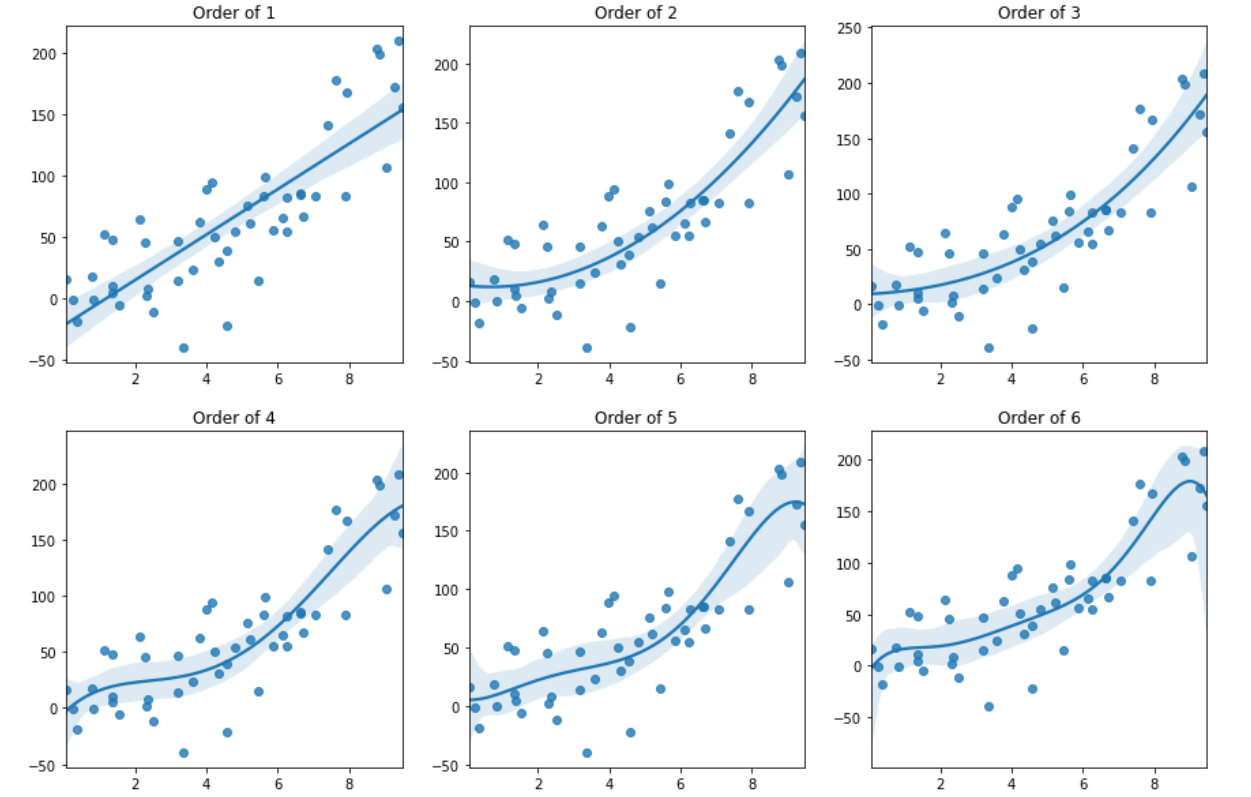
(b) Fit the best linear regression model for the training data. Report the model coefficients and both the R^2 value and mean square error for the fit of that model for the training data.




```
The coefficient is [18.30320685] and the intercept is -21.73078292905422
The mean squared error is: 1052.5853662498014
The R2 score is: 0.7014590913812251
```

(c) To improve the model, first look at the scatter plot of the data and decide what type of nonlinear parameter might be helpful. Use extended version of the linear model, i.e., $y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_nx_n$ where x_n could be any function of x, like $\frac{1}{x}, \log(x), x^2$ .Find the best parameters and also report the R^2 and mean square error of the fit for the training data.

Ans: We can change the order of the polynomial in the `seaborn.regplot` function to see which one better estimates the model. Considering only up to order 6 of the polynomial

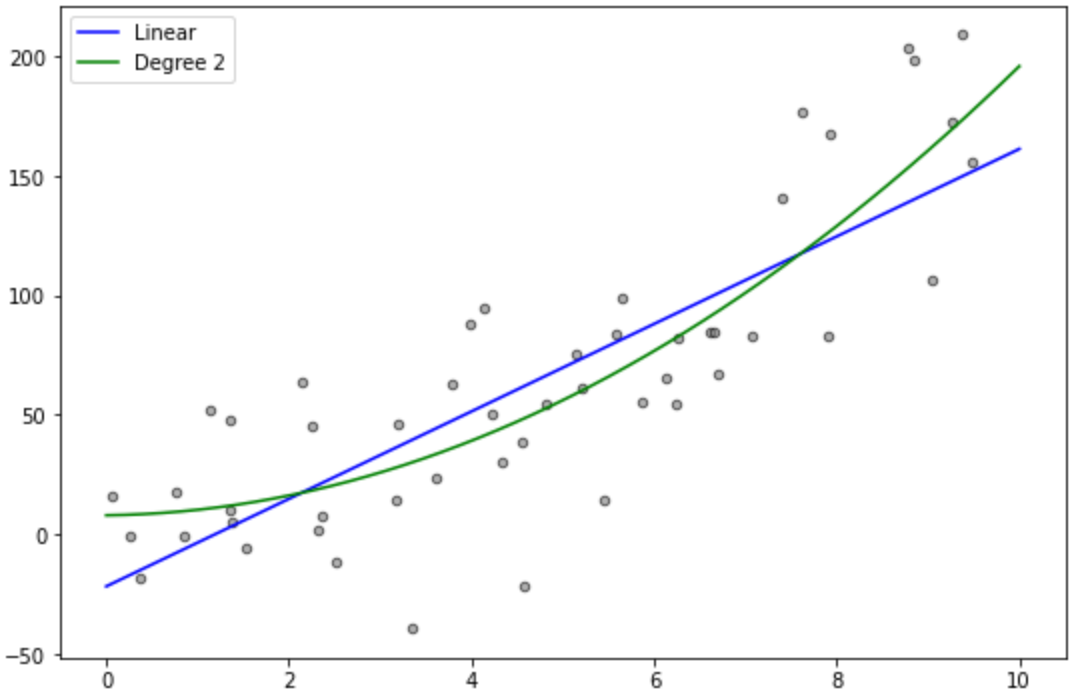


Looks like order of 2 is a good fit. So, we add a new feature i.e. x^2, then we fit the model on the new training set, and report the mean square error and R^2 score.

```
The coefficient is [0.47100686 1.83208191] and the intercept is
8.007337461589657
The mean squared error is: 884.797759660905
The R2 score is: 0.7490480719353505
```

The mean square error decreased by about 10 percent and the R2 also improved

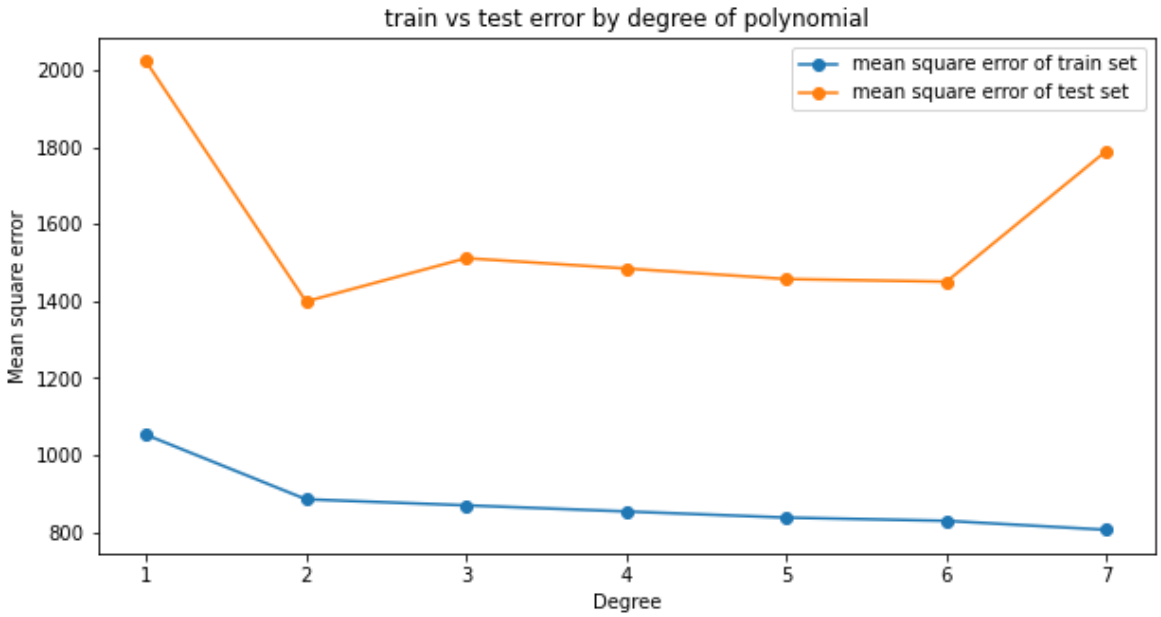
(d) Add the two curves of model in (b) and (c) to the scatter plot of part (a). The figure will be similar to Fig. 3.8 of ISLR.



(e) Use both model and apply them to the test data and estimate the R^2 and mean square error of the test dataset.

```
Mean squared error of test dataset for linear model is 2023.3121088887128 and
for non-linear degree of 2 is 1398.8817580143948
R2 score of test dataset for linear model is 0.5556465885794163 and for non-
linear degree of 2 is 0.6927820089560344
```

(f) Which models perform better on the training data, and which on the test data?



As seen in the plot the second order polynomial is a good fit since the test error increases after that, meaning that the model is overfitting the data. The 2^{nd} order polynomial has lower test error compared to other degrees of polynomials. But the train error decreases as we increase the order of the polynomial which is obvious because we are fitting the model on each point rather than the whole model (We stick with up to 7 order polynomials because after the order of 11 the polynomial fit is not a good idea since the least-squares fit is badly conditioned. This implies that the best fit is not well-defined due to numerical error).