Homework 2

PSTAT 131/231

Aarti Garaye

Linear Regression and KNN

For this assignment, we will be working with a data set from the UCI (University of California, Irvine) Machine Learning repository (see website here). The full data set consists of $4,177$ observations of abalone in Tasmania. (Fun fact: Tasmania supplies about 25% of the yearly world abalone harvest.)

Figure 1: *Inside of an abalone shell.*

The age of an abalone is typically determined by cutting the shell open and counting the number of rings with a microscope. The purpose of this data set is to determine whether abalone age (**number of rings + 1.5**) can be accurately predicted using other, easier-to-obtain information about the abalone.

The full abalone data set is located in the `\data` subdirectory. Read it into *R* using `read_csv()`. Take a moment to read through the codebook (`abalone_codebook.txt`) and familiarize yourself with the variable definitions.

Make sure you load the `tidyverse` and `tidymodels`!

Adding the necessary libraries and loading the data into R.

```
set.seed(123)

library(tidymodels)
library(tidyverse)
library(readr)
library(dplyr)
library(knitr)
library(kableExtra)
library(caret)
library(kknn)
```

```
library(yardstick)

abalone <- read_csv("/Users/aarti/Downloads/homework-2/data/abalone.csv")
```

Question 1

Your goal is to predict abalone age, which is calculated as the number of rings plus 1.5. Notice there currently is no `age` variable in the data set. Add `age` to the data set.

Assess and describe the distribution of `age`.

Solution: To add the column `age` we can use the mutate function and add 1.5 to the values of the `rings` variable. To check the distribution of `age`, we can fit a histogram and see the overall structure.

```
abalone <- abalone %>%
  mutate(age = rings + 1.5)
kable(head(abalone))
```

type	longest_shell	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	rings	age
M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	16.5
M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	8.5
F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	10.5
M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	11.5
I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	8.5
I	0.425	0.300	0.095	0.3515	0.1410	0.0775	0.120	8	9.5

```
hist(abalone$age, main = "Histogram of Age of Abalones",
     xlab = "Age", ylab = "Count")
```

As mentioned in the caption, at a first glance there the distribution of `age` of abalones seems to be fairly normal. We should've expected this since there are 4177 observations. By the central limit theorem, a large n should warrant for convergence towards a normal distribution. The mean age of the Abalones seem to be somewhere around 10-11 years.

If we want to explore normality further we can perform the `qqnorm` and see whether the residuals are aligning close to the line or the statistical method of testing Normal distribution, the Shapiro-Wilk test.

We could also perform the Kolmogorov Smirnoff test to check whether, with respect to the the empirical distribution Donsker's invariance principle, the distribution of age — which is unknown without assumptions — converges to a Gaussian process.

```
qqnorm(abalone$age)
qqline(abalone$age)
```

The residuals don't follow the line in a spiral-like pattern, which suggests that there is some deviation from the normal distribution in the right tail and that the data is a little skewed to the left. We would have missed this from the histogram but the qqplot shows us the skewness. However, this is nothing to be worried about when we are trying to fit a regression model since the assumption about normality is for the residuals and not the predictor variable itself.

In either case, since we have conflicting results, we would like to test the normality using the Statistical test, Shapiro Wilk.

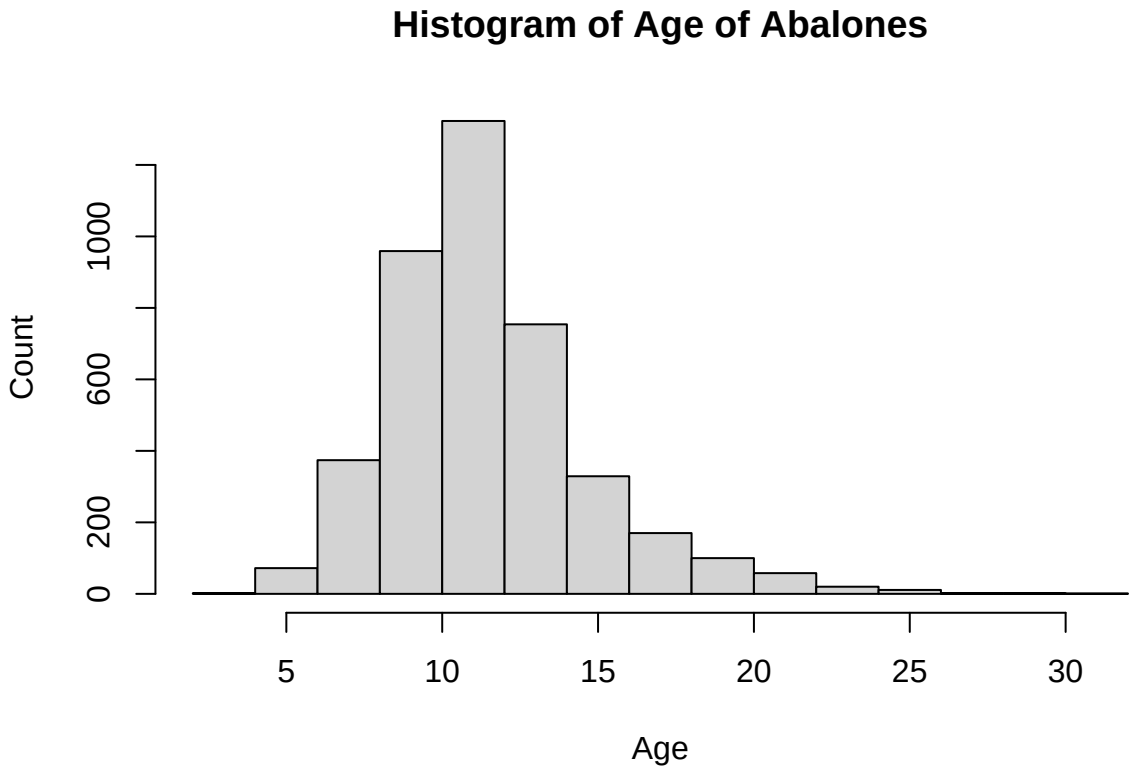

Figure 2: The Histogram of age of abalones. From the structure of theb histogram, the distribution of age looks fairly normal with the mean centered somewhere around 9 to 11 years in age.

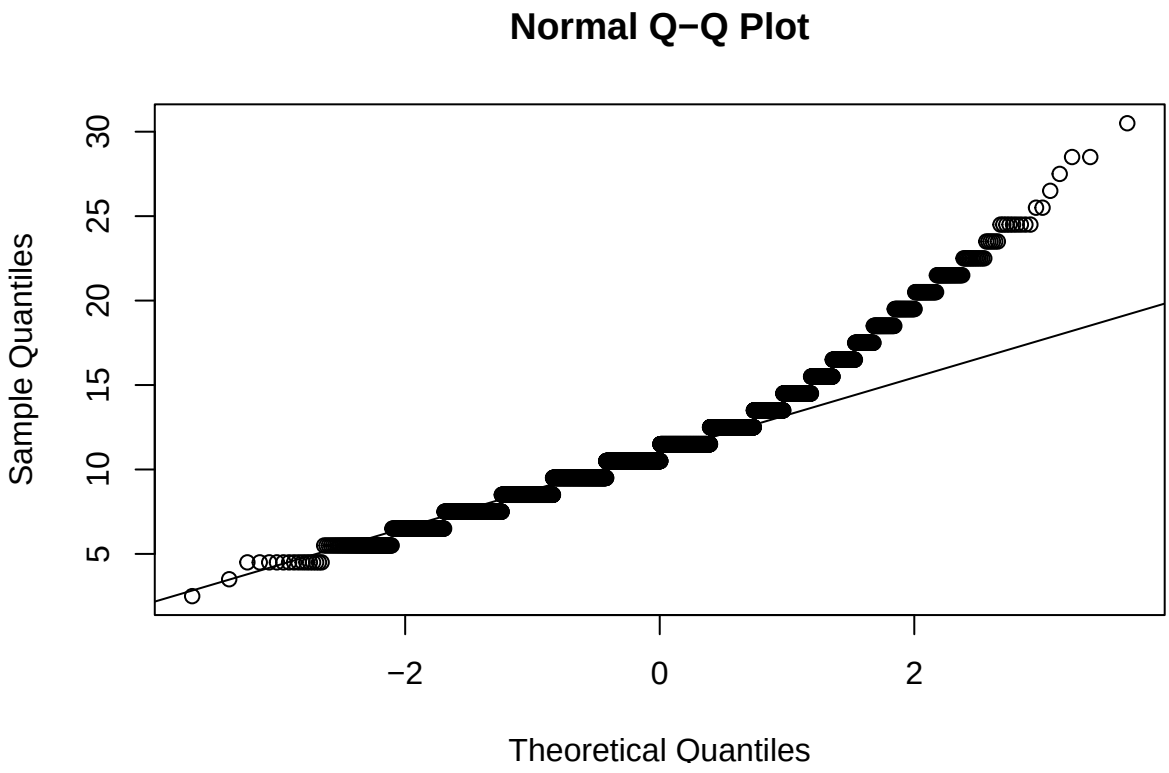

Figure 3: The qqplot of age of Abalones. The qqplot suggests that there is some deviation from the normal distribution as the theoretical quantiles cross 1.

```r
abalone_shapiro <- shapiro.test(abalone$age)

abalone_shapiro_df <- data.frame(
  Statistic = abalone_shapiro$statistic,
  Pvalue = abalone_shapiro$p.value
)

kable(abalone_shapiro_df, caption =
        "Shapiro Wilk test results for the age of the Abalones.")
```

Table 2: Shapiro Wilk test results for the age of the Abalones.

	Statistic	Pvalue
W	0.9311545	0

Since the p-value is small, the `kable` function round is to 0. In any case, the p value is less than our alpha significance level of 0.05 suggesting that we have sufficient information to reject the null and say that `age` follows a normal distribution.

Question 2

Split the abalone data into a training set and a testing set. Use stratified sampling. You should decide on appropriate percentages for splitting the data.

Remember that you'll need to set a seed at the beginning of the document to reproduce your results.

Solution: To split the data into the training set and the testing set using stratified sampling, we can stratify based on a the `rings` variable. The popular choice is a 70/30 split for training and testing, thus, I will be doing the same.

```
abalone_splits <- initial_split(abalone, prop = 0.80,
                                strata = age)
abalone_train <- training(abalone_splits)
abalone_test <- testing(abalone_splits)
```

Above code splits the `Abalone` dataset in 80/20 split where 80% of the data is in the training set, and 20% of the data is in the testing set. Note, that we have already set seed in the introductory code chunk where all the necessary libraries and the data is loaded.

Question 3

Using the **training** data, create a recipe predicting the outcome variable, `age`, with all other predictor variables. Note that you **should not** include `rings` to predict `age`. *Explain why you shouldn't use* ***`rings`*** *to predict* ***`age`***.

Steps for your recipe:

1. dummy code any categorical predictors
2. create interactions between
    - `type` and `shucked_weight`,
    - `longest_shell` and `diameter`,
    - `shucked_weight` and `shell_weight`
3. center all predictors, and
4. scale all predictors.

You'll need to investigate the `tidymodels` documentation to find the appropriate step functions to use.

Solution: Remember that we derived `age` from `rings`. They both are dependent. This violates the assumption of independence and would make the model trivial and unrealistic. Below is the steps created for the recipe to predict age using all other predictors.

```
simple_abalone_recipe <- recipe(age ~ ., data = abalone_train) %>%
  step_rm(rings) %>%
  step_dummy(all_nominal_predictors())

prep(simple_abalone_recipe) %>%
  bake(new_data = abalone_train) %>%
  head() %>%
  kable(caption =
          "Data when gender is dummy coded. Note that female is the reference level.")
```

Table 3: Data when gender is dummy coded. Note that female is the reference level.

longest_shell	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	age	type_I	type_M
0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5	0	1
0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5	1	0
0.355	0.280	0.085	0.2905	0.0950	0.0395	0.115	8.5	1	0
0.365	0.295	0.080	0.2555	0.0970	0.0430	0.100	8.5	0	1
0.465	0.355	0.105	0.4795	0.2270	0.1240	0.125	9.5	0	1
0.450	0.355	0.105	0.5225	0.2370	0.1165	0.145	9.5	0	0

Now we would like to create the interaction terms specified in the question, center all predictors, and scale them.

Table 4: Data with dummy coding, interaction terms, centering all predictors, and scaling all predictors.

longest_shell	diameter	height	whole_weight	shucked_weight	viscera_weight	shell_weight	age	type_I	type_M	type_I_x_shucked_weight	type_M_x_shucked_weight	longest_shell_x_diameter	shucked_weight_x_shell_weight
-1.4528364	-1.4441444	-1.2835325	-1.2277480	-1.1663850	-1.2033059	-1.2145007	8.5	-0.6831563	1.307030	-0.5289765	-0.2407409	-1.4070522	-0.8872545
-1.6204058	-1.5454804	-1.5452540	-1.2697328	-1.2114505	-1.2858363	-1.3232456	8.5	1.4633556	-0.764864	0.2613046	-0.6418631	-1.4986145	-0.9044726
-1.4109441	-1.2921404	-1.4143932	-1.0946254	-1.1866644	-1.2858363	-0.8882662	8.5	1.4633556	-0.764864	0.3098693	-0.6418631	-1.3362511	-0.8538720
-1.3271594	-1.1401364	-1.5452540	-1.1663068	-1.1776513	-1.2537412	-0.9970111	8.5	-0.6831563	1.307030	-0.5289765	-0.2508194	-1.2481490	-0.8641987
-0.4893126	-0.5321203	-0.8909502	-0.7075459	-0.5917995	-0.5109677	-0.8157697	9.5	-0.6831563	1.307030	-0.5289765	0.2732599	-0.6370237	-0.7067698
-0.6149896	-0.5321203	-0.8909502	-0.6194802	-0.5467339	-0.5797430	-0.6707766	9.5	-0.6831563	-0.764864	-0.5289765	-0.6418631	-0.6937178	-0.6562746

```
abalone_recipe <- recipe(age ~ ., data = abalone_train) %>%
  step_rm(rings) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(terms = ~ starts_with("type_"):shucked_weight) %>%
  step_interact(terms = ~ longest_shell:diameter) %>%
  step_interact(terms = ~ shucked_weight:shell_weight) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

prep(abalone_recipe) %>%
  bake(new_data = abalone_train) %>%
  head() %>%
  kable(caption =
          "Data with dummy coding, interaction terms,
        centering all predictors, and scaling all predictors.",
        "latex", booktabs = T) %>%
  kable_styling(latex_options = c("scale_down"))
```

Now we have our recipe ready for predicting `age`.

Question 4

Create and store a linear regression object using the `"lm"` engine.

Solution: Below is the code for creating and storing the linear regression object using the `lm` engine.

```
abalone_lm_model <- linear_reg() %>%
  set_engine("lm")
```

We set up a workflow. This step might seem unnecessary now, with only one model and one recipe, but it can make life easier when you are trying out a series of models or several different recipes later on.

Question 5

Create and store a KNN object using the `"kknn"` engine. Specify `k = 7`.

Solution: Below is the code for creating and storing the KNN object using the `kknn` engine where `k = 7`.

```
abalone_knn_model <- nearest_neighbor(neighbors = 7) %>%
  set_engine("kknn") %>%
  set_mode("regression")
```

Question 6

Now, for each of these models (linear regression and KNN):

1. set up an empty workflow,
2. add the model, and
3. add the recipe that you created in Question 3.

Note that you should be setting up two separate workflows.

Fit both models to the training set.

Solution: Below is the code for setting the workflow for the linear regression model.

```
abalone_lm_workflow <- workflow() %>%
  add_model(abalone_lm_model) %>%
  add_recipe(abalone_recipe)
```

Now we would like to do the same for the KNN model

```
abalone_knn_workflow <- workflow() %>%
  add_model(abalone_knn_model) %>%
  add_recipe(abalone_recipe)
```

Now we would like to fit each model. Let's start with fitting the linear regression model

```
abalone_lm_fit <- fit(abalone_lm_workflow, abalone_train)

abalone_lm_fit %>%
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 14 x 5
##    term                          estimate std.error statistic  p.value
##    <chr>                            <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)                     11.4      0.0373   306.    0
##  2 longest_shell                    0.351    0.286      1.23  2.19e- 1
##  3 diameter                         1.76     0.317      5.55  3.06e- 8
##  4 height                           0.627    0.101      6.18  7.17e-10
##  5 whole_weight                     4.57     0.394     11.6   1.73e-30
##  6 shucked_weight                  -4.27     0.255    -16.8   1.42e-60
##  7 viscera_weight                  -0.976    0.158     -6.18  7.26e-10
##  8 shell_weight                     1.48     0.218      6.77  1.47e-11
##  9 type_I                          -1.06     0.115     -9.23  4.77e-20
## 10 type_M                          -0.273    0.103     -2.65  8.16e- 3
## 11 type_I_x_shucked_weight          0.610    0.0869     7.02  2.59e-12
## 12 type_M_x_shucked_weight          0.294    0.109      2.70  7.06e- 3
## 13 longest_shell_x_diameter        -2.43     0.414     -5.87  4.65e- 9
## 14 shucked_weight_x_shell_weight    0.121    0.203      0.595 5.52e- 1
```

The intercept tells us that when all the predictors and the interaction terms are zero the estimated age of an abalone is around 11.437. And the other values in the estimate column is saying that one unit increase in

that particular predictor while keeping all the other predictors at the same rate would result in an increase (if positive) or decrease (if negative) in the age of the abalone.

Now we would fit the k-nearest neighbor model

```
abalone_knn_fit <- fit(abalone_knn_workflow, abalone_train)

abalone_knn_fit %>%
  extract_fit_parsnip()
```

```
## parsnip model object
##
##
## Call:
## kknn::train.kknn(formula = ..y ~ ., data = data, ks = min_rows(7,     data, 5))
##
## Type of response variable: continuous
## minimal mean absolute error: 1.648287
## Minimal mean squared error: 5.506757
## Best kernel: optimal
## Best k: 7
```

We fit both models to the training data set, noe we want to see how well they do on our testing data set.

Question 7

Use your linear regression `fit()` object to predict the age of a hypothetical female abalone with longest_shell = 0.50, diameter = 0.10, height = 0.30, whole_weight = 4, shucked_weight = 1, viscera_weight = 2, and shell_weight = 1.

Solution: Note that the equation of the linear regression could be given as:

$$\widehat{Y} = 11.4374251 + 0.3514627X_1 + 1.7617258X_2 + 0.6267087X_3 + 4.5691032X_4 + -4.2652767X_5 + -0.9755606X_6 + 1.4773657X_7 + -$$

Were X_1 is the longest_shell, X_2 is the diameter, X_3 is the height, X_4 is the whole_weight, X_5 is the shucked_weight, X_6 is the viscera_weight, X_7 is the shell_weight, X_8 is the type_I (type infant), and X_9 is the type_M (type male). So if we just plug them in we would get the expected age of the abalone. But that would take manual calculation, what we can rather do is use the `predict()` function.

```
new_abalone <- tibble(
  type = "F",
  longest_shell = 0.50,
  diameter = 0.10,
  height = 0.30,
  whole_weight = 4,
  shucked_weight = 1,
  viscera_weight = 2,
  shell_weight = 1,
  rings = 0
)

predict(abalone_lm_fit, new_abalone)
```

```
## # A tibble: 1 x 1
##   .pred
##   <dbl>
## 1  23.6
```

The predicted age of the given abalone with according to the linear regression model is 23.6321648193124.

Question 8

Now you want to assess your models' performance. To do this, use the `yardstick` package:

1. Create a metric set that includes R^2, RMSE (root mean squared error), and MAE (mean absolute error).
2. Use `augment()` to create a tibble of your model's predicted values from the **testing data** along with the actual observed ages (these are needed to assess your model's performance).
3. Finally, apply your metric set to the tibble, report the results, and interpret the *R^2* value.

Repeat these steps once for the linear regression model and for the KNN model.

Solution: Before we do that we could like to predict it for the training data set and see how far off our models are. Follow the code below to see the predictions for the linear regression models.

```
abalone_train_res <- predict(abalone_lm_fit,
                             new_data = abalone_train %>% select(-age))
abalone_train_res %>%
  head() %>%
  kable(caption =
          "Predicted values when we fit the linear
        regression model in the training set")
```

Table 5: Predicted values when we fit the linear regression model in the training set

.pred
9.610447
7.995828
9.576090
10.339264
9.943774
10.831503

```
# Combining the predicted values in the data set
abalone_train_res <- bind_cols(abalone_train_res,
                               abalone_train %>% select(age))
abalone_train_res %>%
  head() %>%
  kable(caption =
          "Predicted vs. the actual age when we fit
        the linear regression model in the training set.")
```

Table 6: Predicted vs. the actual age when we fit the linear regression model in the training set.

.pred	age
9.610447	8.5
7.995828	8.5
9.576090	8.5

.pred	age
10.339264	8.5
9.943774	9.5
10.831503	9.5

As we can see the predicted values are a little off. It's fairly clear that the model didn't do very well. If it predicted every observation accurately, the dots would form a straight line.

Now we can calculate the metrics for the testing dataset.

```
abalone_test_res <- augment(abalone_lm_fit, new_data = abalone_test)

abalone_metrics <- metric_set(rsq, rmse, mae)
abalone_metrics(abalone_test_res, truth = age,
                estimate = .pred) %>%
  kable(caption = "The metrics for the linear
        regression model when fit for the testing data.")
```

Table 7: The metrics for the linear regression model when fit for the testing data.

.metric	.estimator	.estimate
rsq	standard	0.5483193
rmse	standard	2.1883188
mae	standard	1.5481919

From the table above we can see that the R^2 value is 0.5483193, which means our linear regression model only able to explain 54.83% of the variance. This means that only 54.83% of the variance is depended on our predictors variables and the interaction terms we fit our model on. In simpler terms, it shows what percentage of the changes in one variable can be explained by changes in another variable. Generally if R^2 is 80% or above, the model is considered to be a good model for prediction. Clearly our model is not doing a very good job. Let's see how the KNN model does.

```
abalone_train_knn <- predict(abalone_knn_fit, new_data = abalone_train %>% select(-age))
abalone_train_knn <- bind_cols(abalone_train_knn, abalone_train %>% select(age))

abalone_metrics(abalone_train_knn, truth = age, estimate = .pred) %>%
  kable(caption = "Metrics of the KNN model on the training set.")
```

Table 8: Metrics of the KNN model on the training set.

.metric	.estimator	.estimate
rsq	standard	0.8096888
rmse	standard	1.4529122
mae	standard	1.0211404

However, we are not interested in the training metrics, we want to look for the testing metric.

```r
abalone_test_res2 <- augment(abalone_knn_fit, new_data = abalone_test)

abalone_metrics(abalone_test_res2, truth = age, estimate = .pred) %>%
  kable(caption = "Metrics of the KNN model on the testing set.")
```

Table 9: Metrics of the KNN model on the testing set.

.metric	.estimator	.estimate
rsq	standard	0.4886653
rmse	standard	2.3252337
mae	standard	1.6563690

Note how the R^2 value went drastically down from the training set to the testing set. It could be due to that the KNN model is overfitting which means we need to increase the number of K. However, in this case the R^2 is 0.4886653 which means that 48.87% of the variability could be explained in this model. This is not a very good model to predict the age of the abalone.

Question 9

Which model performed better on the testing data? Explain why you think this might be. Are you surprised by any of your results? Why or why not?

Solution: The linear regression model performed better on the testing data. We know that a higher R^2 means a better model, the linear regression has a little bit of better R^2.

KNN is a local and non-parametric method which means that I can expect it be work the best when we find the optimal hyperparameter, k. Since we tried with only one k value, there is a good chance of improving our model. Furthermore, we know KNN struggles with higher dimensional predictors, and in our case we had a lot of them.

Furthermore, based on the types of variables, it was reasonable to expect that age would grow somewhat linearly so the linear regression makes a little more sense. More than that, we knew that KNN would struggle because this was a very high-dimensional data to fit in that model. So, I would say I'm not very surprised by the results, if anything I would've expected the R^2 of the testing data when we fit the KNN model to be even lower.

Required for 231 Students

In lecture, we presented the general bias-variance tradeoff, which takes the form:

$$E[(y_0 - \hat{f}(x_0))^2] = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon)$$

where the underlying model $Y = f(X) + \epsilon$ satisfies the following:

- ϵ is a zero-mean random noise term and X is non-random (all randomness in Y comes from ϵ);
- (x_0, y_0) represents a test observation, independent of the training set, drawn from the same model;
- $\hat{f}(.)$ is the estimate of f obtained from the training set.

Question 10 Which term(s) in the bias-variance tradeoff above represent the reducible error? Which term(s) represent the irreducible error?

Solution: In the bias-variance tradeoff the reducible error is the

$$Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2$$

because this is the term we can allegedly make zero and then we would have the equation $Y = f(X) + \epsilon$ such that our model is perfectly capturing the $f(X)$ where that "function" is the relationship in the actual population. In statistical learning f is some unkown function of the predictors and thus the only randomness in Y is coming from the random noise, represented by the ϵ. Thus, we can summarize statistical learning in essence that it refers to a set of approaches for estimating f.

Then, the irreducible error in the bias-variance tradeoff would be the

$$Var(\epsilon)$$

because no matter how close our estimated function, $\hat{f}$, to the real function f would be there would still be some error from the random noise in the population. No matter how well we estimate f, we cannot reduce the error introduced by ϵ. Thus the $Var(\epsilon)$ is the irreducible error.

Question 11 Using the bias-variance tradeoff above, demonstrate that the expected test error is always at least as large as the irreducible error.

Solution: We know that the (x_0, y_0) represents a test observation, independent of the training set. In order to have a measure of the performance of our model $\hat{f}(X)$ on approximating map $X \to Y$, we define the error function that measures the distance between the target Y and the estimated value, $\widehat{Y} = \hat{f}(X)$ which we desire to minimize.

An usual error term choice is the mean squared error MSE, which is the squared (direction insensitive) residuals, averaged over all sampled data pairs $(x_i, y_i), i = 0, 1, ..., n$ This average is our approximation of the expected value of this residuals and can be written as

$$\mathbb{E}\left[\left(y_i - \hat{f}(x_i)\right)^2\right]$$

which is our test mean squared error. In simpler terms it refers to the average test MSE that we would obtain if we repeatedly estimated f using a large number of training sets and tested each at x_0.

Note, variance is inherently a non-negative quantity and the squared bias is also non-negative. Now the bias-variance tradeoff equation tells us that minimizing the expected test error, we need low bias and low variance. So, we can see that the minimized would still mean that the expected test MSE is equal to the irreducible error term, $Var(\epsilon)$. Hence, it can never lie below $Var(\epsilon)$.

Question 12 Prove the bias-variance tradeoff.

Hints:

- use the definition of $Bias(\hat{f}(x_0)) = E[\hat{f}(x_0)] - f(x_0)$;
- reorganize terms in the expected test error by adding and subtracting $E[\hat{f}(x_0)]$

Solution: *Proof:* Note,

$$\mathbb{E}[(y_0 - \hat{f}(x_0))^2] = \mathbb{E}\left[\left(y_0 - f(x_0) + f(x_0) - \hat{f}(x_0)\right)^2\right]$$

$$= \mathbb{E}\left[(y_0 - f(x_0))^2\right] + \mathbb{E}\left[\left(f(x_0) = \hat{f}(x_0)\right)^2\right] + 2\mathbb{E}\left[\left(f(x_0) - \hat{f}(x_0)\right)(y_0 - f(x_0))\right]$$

Note, $y_0 = f(x_0) + \epsilon$, which gives

$$\mathbb{E}\left[(f(x_0) + \epsilon - f(x_0))\right] + 2\mathbb{E}\left[y_0 f(x_0) - f(x_0)^2 - \hat{f}(x_0)y_0 + \hat{f}(x_0)f(x_0)\right]$$

$$\mathbb{E}[\epsilon] + \mathbb{E}\left[\left(f(x_0) - \hat{f}(x_0)\right)^2\right] + 2\left(-f(x_0)^2 + f(x_0)^2 - f(x_0)\mathbb{E}[\hat{f}(x_0)] + f(x_0)\mathbb{E}[\hat{f}(x_0)]\right)$$

Note,

$$Var(\epsilon) = \mathbb{E}[\epsilon^2] - \mathbb{E}[\epsilon]^2 = \mathbb{E}[\epsilon^2] - 0 = \mathbb{E}[\epsilon^2]$$

Gives,

$$Var(\epsilon) + \mathbb{E}\left[\left(f(x_0) - \hat{f}(x_0)\right)^2\right]$$

$$Var(\epsilon) + \mathbb{E}\left[\left(f(x_0) - \mathbb{E}[\hat{f}(x_0)] + \mathbb{E}[\hat{f}(x_0)] - \hat{f}(x_0)\right)\right]$$

$$Var(\epsilon) + \mathbb{E}\left[\mathbb{E}\left[(\hat{f}(x_0))\right] - f(x_0)\right]^2 + \mathbb{E}\left[\hat{f}(x_0) - \mathbb{E}[\hat{f}(x_0)]\right]^2$$

$$Var(\epsilon) + Bais^2[\hat{f}(x_0)] + Var(\hat{f}(x_0))\blacksquare$$