EASTERN MICHIGAN UNIVERSITY

MASTER'S THESIS

DEPARTMENT OF MATHEMATICS AND STATISTICS

Prostate Cancer: Multiple Logistic Regression

Author
JEFFREY OSIWALA

Supervisor
Prof. KHAIRUL ISLAM

December 3, 2020

1 Proposal

In a research study, a university medical center urology group was interested in the association between prostate-specific antigen (PSA) and a number of prognostic clinical measurements in men with advanced prostate cancer. Data were collected on 97 men who were about to undergo radical prostectomies. The data given has identifications numbers, and provides information on 8 other variables on each person. The 8 variables being: PSA Level, Cancer Volume, Weight, Age, Benign Prostatic Hyperplasia, Seminal Vesicle Invasion, Capsular Penetration, and Gleason Score.

With this available data set, I will carry out a complete logistic regression analysis by first creating a binary response variable Y, called high-grade-cancer, by letting Y=1 if Gleason Score equals 8, and Y=0 otherwise (i.e., if Gleason Score equals 6 or 7). Thus, the response of interest is high-grade-cancer (Y), and the pool of predictors include those previously mentioned.

My analysis will consider transformations of predictors, the inclusion of second-order predictors, analysis of residuals and influential observations, model selection, goodness of fit evaluation, and the development of an ROC curve. Additionally, I will discuss the determination of a prediction rule for determining whether the grade of disease is predicted to be high grade or not, model validation, and finally assess the strengths and weaknesses of my final model.

2 Rationale

Prostate Cancer is the most common cancer in American men. The American Cancer Society (ACS), a nationwide voluntary health organization, estimates 191,930 new cases of prostate cancer and over 33,000 deaths in year 2020 alone. Additionally, the typical cost of therapy to a prostate cancer patient is $2,800/month after diagnosis (primarily from surgery and subsequently from office visits). A reliable and well understood testing/screening procedure needs to be in place to support early detection, and to minimize these current and unforgiving metrics.

Research suggests that prostate cancer typically begins as a pre-cancerous condition, and these conditions are sometimes found when a man has an invasive prostate biopsy (the removal of small pieces of the prostate to look for cancer.) If prostate cancer is found early as a result of *screening*, it will probably be at an earlier and more treatable stage than if no screening were done. While this might seem like prostate cancer screening would always be a good thing, there are still issues surrounding screening procedures that make it unclear if the benefits outweigh the risks for most men.

For example, the popular PSA screening test is not 100% accurate. This test can sometimes have abnormal results even when a man does not have cancer (false-positive result), or normal results when a man does have cancer (false-negative result). Consequently, false-positive results can lead to some men to get prostate biopsies (with risks of pain, infection, and bleeding) when they do not have cancer, and false-negative results can give men a false sense of security even though they may actually have cancer.

Another important issue is that even if screening does detect prostate cancer, doctors often cannot tell if the cancer is truly dangerous and needs to be treated. Prostate cancer can grow so slowly that it may never cause a man problems in his lifetime, and some men who seek screening may be diagnosed with a prostate cancer that they would have never known about otherwise. It would never have led to their death, or even cause any symptoms. Finding a "disease" like this that would never cause problems is known as **overdiagnosis**.

The problem with overdiagnosis in prostate cancer is that many of the men might still be treated with either surgery or radiation, either because the doctor cannot be sure how quickly the cancer might grow or spread, or the man is uncomfortable knowing he has cancer and

is not receiving any treatment. The treatment of a cancer that would never have caused any problems is known as **overtreatment**, and the major downsides after surgery or radiation may include urinary, bowel, and/or sexual side effects that can seriously affect a man's quality of life. Thus, men and their doctors often struggle to decide if treatment is needed, or if the cancer can just be closely watched without being treated right away. Even when men are not treated right away, they still need regular blood PSA tests and prostate biopsies to determine if their need for treatment in the future.

For now, the ACS recommends that men thinking about getting tested for prostate cancer learn as much as they can so they can make informed decisions based on available information, discussions with their doctors, and their own views on the possible benefits, risks, and limits of prostate cancer screening. To combat and better navigate these difficulties, research needs to continue to grow the understanding of prostate cancer, and to build stronger predictive models which can improve the outlook of male lives, and also alleviate undo strain on the healthcare system.

3 Literature Review

3.1 Predictor Variables

An understanding of the predictor variables in this particular study can be seen as follows:

- **PSA Level**: Serum prostate-specific antigen level [mg/ml].

  -Prostate cancer can often be found early by testing for prostate-specific antigen (PSA) levels in a man's blood. However, the PSA test is not 100% accurate.
  - The chance of having prostate cancer increases as PSA level increases, but there is no set cutoff point that can tell for sure if a man does or does not have prostate cancer.

- **Cancer Volume**: Estimate of prostate cancer volume [cc].

  -Studies have suggested that inflammation of the prostate gland (prostatitis) may be linked to an increased risk of prostate cancer, but other studies have not found such a link.
  -Inflammation is often seen in samples of prostate tissue that also contain cancer. The link between the two it not clear, and it remains an active area of research.

- **Weight**: Prostate weight [gm].

  -As related to cancer volume, studies have suggested that inflammation (and an increase is prostate weight) may be linked to an increased risk of prostate cancer. This relationship remains an active area of research.

- **Age**: Age of patient [years].

  -Prostate cancer is rare in men younger than 40, but the chance of having prostate cancer rises rapidly after age 50. About 6 in 10 cases of prostate cancer are found in men older than 65.

- **Benign Prostatic Hyperplasia**: Amount of benign prostatic hyperplasia [cm^2].

  -BPH is a term used to describe common, benign type of prostate enlargement caused by an increased number of normal prostate cells. This condition is more common as men get older and is not currently known to be linked to cancer.

- **Seminal Vesicle Invasion**: Presence or absence of seminal vesicle invasion: 1 if yes; 0 otherwise.

-SVI is the presence of prostate cancer in the areolar connective tissue around the seminal vesicles and outside the prostate.

- **Capsular Penetration**: Degree of capsular penetration [cm].

  -Cancer that has reached the outer wall of an organ (i.e. the prostate) is referred to as capsular penetration. Conversely, if cancer is strictly confined to the organ itself it is called organ-confined cancer.

- **Gleason Score**: Pathologically determined grade of disease using total score of two patterns (summed scores were either 6, 7, or 8 with higher scores indicating worse prognosis).

  -A measure of how likely the cancer is to grow and spread quickly. This is typically determined by the results of the prostate biopsy, or surgery.

3.2 Related Research

Doctors are still studying if screening tests will lower the risk of death from prostate cancer. The most recent results from two large studies show conflicting evidence, and unfortunately did not offer clear answers.

The outcomes of both studies can be summarized as follows:

- Early results from a large study done in the United States found that annual screening with PSA and DRE (digital rectal exam - for a DRE, the doctor puts a gloved, lubricated finger into the rectum to feel the prostate gland) did detect more prostate cancers than in men not screened, but this screening did not lower the death rate from prostate cancer. However, questions have been raised about this study, because some men in the non-screening group actually were screened during the study, which may have affected the results.

- A European study did find a lower risk of death from prostate cancer with PSA screening (done about every 4 years), but the researchers estimated that roughly 781 men would need to be screened (and 27 cancers detected) to prevent one death from prostate cancer.

- Neither of these studies has shown that PSA screening helps men live longer overall (i.e. lowers the overall death rate).

Prostate cancer is often slow-growing, so the effects of screening in these studies might become more clear in coming years. Also, both of these studies are being continued to see if a longer follow-up will give clearer results.

4 Design and Analysis

To best model the dichotomous response variable, Y_HighGradeCancer, in the Prostate Cancer case study, I will employ a multiple logistic regression model, where 1 indicates high grade cancer and 0 indicates not high grade cancer.

In statistics, if $\pi = f(x)$ is a probability then $\frac{\pi}{1-\pi}$ is the corresponding *odds*, and the **logit** of the probability is the logarithm of the odds:

$$logit(\pi) = log(\frac{\pi}{1-\pi}) \tag{1}$$

Now, simple logistic regression means assuming that $\pi(x)$ is related to $\beta_0 + \beta_1 x$ (the *logit response function*) by the logit function. By equating $logit(\pi)$ to the logit response function, we

understand that the logarithm of the odds is a linear function of the predictor. In particular, the slope parameter β_1 is the change in the log odds associated with a one-unit increase in x. This implies that the odds itself changes by the multiplicative factor e^{β_1} when x increases by 1 unit.

$$log(\frac{\pi}{1-\pi}) = \beta_0 + \beta_1 x \tag{2}$$

From here, straightforward algebra will then show the Simple Linear Regression Model:

$$E[Y] = \pi(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}} \tag{3}$$

Next, this simple logistic regression model is easily extended to more than one predictor variable by inclusion of the following two vectors, in matrix notation:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} 1 \\ X_1 \\ X_2 \\ \vdots \\ X_{p-1} \end{bmatrix}$$

With this notation, the simple logistic response function (Eqn. 3) extends to the multiple logistic response function as follows:

$$E[Y] = \pi(\mathbf{X}) = \frac{exp(\mathbf{X}'\boldsymbol{\beta})}{1+exp(\mathbf{X}'\boldsymbol{\beta})} \tag{4}$$

Fitting the logistic regression to the sample data requires that the parameters β_0, β_1,$\cdots$, β_{p-1} be estimated. This will be done using the maximum likelihood technique provided within the statistical packages of both **R** and *Python*.

4.1 Data Transformations and Standardization

Variable transformation is an important technique to create robust models using logistic regression, and the appropriate transformations on continuous variables are necessary to optimize the model predictiveness. Because the predictors are linear in the log of the odds, it is often helpful to transform the continuous variables to create a more linear relationship.

The raw data collected contained several predictors with high skewness values. A few concerning features were determined to be PSA Level (skewness = 4.39), Cancer Volume (skewness = 2.18), and Weight (skewness = 7.46). As a prepossessing step to reduce skewness, I elected to transform these continuous predictor variables using the log-transformation, and standardize *all* the data on top of that. The standardization step was used to normalize the data, and did not affect any underlying distributions among the predictor variables.

The finalized data skewness is summarized directly below in Figure 1.

```
The skewness of PSALevel is: 0.0
The skewness of CancerVol is: -0.25
The skewness of Weight is: 1.21
The skewness of Age is: -0.83
The skewness of BenignProstaticHyperplasia is: 0.98
The skewness of SeminalVesicleInvasion is: 1.4
The skewness of CapsularPenetration is: 2.13
```

Figure 1: Finalized Skewness Values of Transformed Predictor Variables.

Additionally, I've included the histogram of PSA Level vs. Cancer Volume in Figure 2 - a helpful visual for the two predictors which carried the most significance through much of my analysis, as we soon shall see. Notice how the distributions exhibit no notable skewness, are quite symmetrical, and are centered on zero.

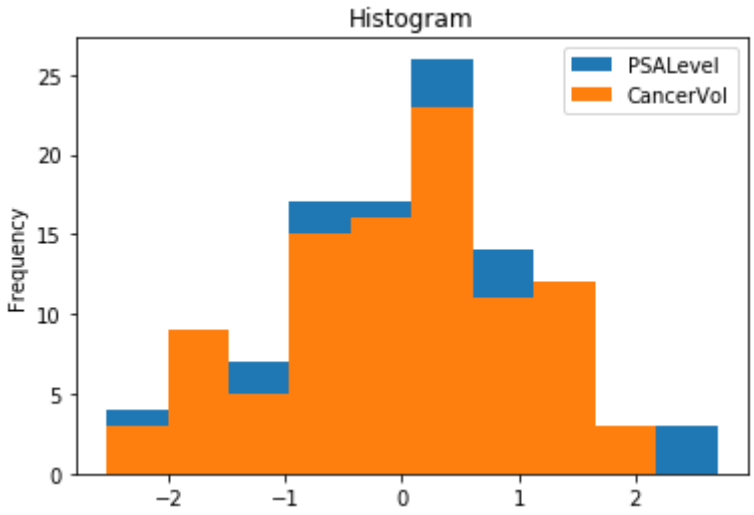


Figure 2: Finalized PSALevel vs. CancerVol Histogram.

4.2 Model Selection

The data of 97 individual men in the Prostate Cancer sample was split at 80% for train (model-building) and test (validation) sets. The training set is a random 76 observations and was used for fitting the model, and the remaining 21 cases were saved to serve as a validation data set. Figure 3 in columns 1-8 contains the variables, and shows a portion of the finalized and processed training data.

Obs	Y_HighGradeCancer	PSALevel	CancerVol	Weight	Age	BenignProstaticHyperplasia	SeminalVesicleInvasion	CapsularPenetration
1	0	-2.533700	-1.645747	-1.785921	-1.872101	-0.840562	0	-0.596573
2	0	-2.299250	-1.995368	-0.673281	-0.791989	-0.840562	0	-0.596573
3	0	-2.299250	-1.586043	-1.947772	1.368234	-0.840562	0	-0.596573
4	0	-2.299250	-2.174506	-0.754163	-0.791989	-0.840562	0	-0.596573
6	0	-1.488689	-2.046685	-0.855308	-1.872101	-0.840562	0	-0.596573
...	...	...	...	...	...	...	...	...
92	0	1.438825	1.006641	0.055045	-0.386947	0.438624	1	-0.596573
93	1	1.665361	1.262501	0.459679	0.558151	-0.840562	1	0.398013
94	1	1.918045	2.106830	0.500132	-2.682185	-0.840562	1	1.730425
96	1	2.615096	1.305144	0.237142	0.558151	0.737545	1	0.667795
97	1	2.702227	1.808328	0.641786	0.558151	-0.325658	1	4.232114

76 rows × 8 columns

Figure 3: Portion of Processed Model-Building Data Set - *Python* Dataframe.

4.2.1 Best Subsets Procedure

The procedure outlined here will help identify a group of subset models that give the best values of a specified criterion. This technique has been developed by time-saving algorithms which can find the most promising models, without having to evaluate all 2^{p-1} candidates. The use of the best subset procedure is based on the AIC_p criteria, where promising models will yield a relatively small value.

The minimized AIC_p stepwise output given by **R** is provided in Figure 4 below.

```
Step:  AIC=50.63
Y_HighGradeCancer ~ PSALevel + CancerVol

            Df Deviance    AIC
<none>           44.628 50.628
- PSALevel   1   48.123 52.123
- CancerVol  1   50.767 54.767

Call:  glm(formula = Y_HighGradeCancer ~ PSALevel + CancerVol, family = "binomial",
    data = training)

Coefficients:
(Intercept)     PSALevel    CancerVol
     -2.687        1.058        1.550

Degrees of Freedom: 75 Total (i.e. Null);  73 Residual
Null Deviance:      72.61
Residual Deviance: 44.63        AIC: 50.63
```

Figure 4: Full Linear Model AIC_p Best Subset Results - **R** Output.

In this procedure, I instructed **R** to iterate "backwards" through all 7 predictor variables and it was determined AIC_p was minimized for $p = 3$. In particular, the results reveal that the best two-predictor model for this criteria is based on *PSA Level* and *Cancer Volume*. The AIC_p was minimized to 50.63, with a Null Deviance equal to 72.61 and Residual Deviance equal to 44.63.

4.2.2 Model Fitting

A first-order multiple logistic regression model with two predictor variables was considered to be reasonable by §4.2.1:

$$\pi(\mathbf{X}) = \frac{exp(\mathbf{X}'\boldsymbol{\beta})}{1 + exp(\mathbf{X}'\boldsymbol{\beta})} = [1 + exp(-\mathbf{X}'\boldsymbol{\beta})]^{-1} \tag{5}$$

where:

$$\mathbf{X}'\boldsymbol{\beta} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \tag{6}$$

This model was then fit by the method of maximum likelihood to the data from the 76 random training cases. Results are summarized in the Figure 5 *Python* output below. Provided in the output are the estimated coefficients, their standard errors, z-scores, p-values, and the accompanying 95% confidence intervals.

```
                          Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.385
Dependent Variable: Y_HighGradeCancer AIC:              50.6278
Date:               2020-12-01 14:52  BIC:              57.6200
No. Observations:   76                Log-Likelihood:   -22.314
Df Model:           2                 LL-Null:          -36.307
Df Residuals:       73                LLR p-value:      8.3761e-07
Converged:          1.0000            Scale:            1.0000
No. Iterations:     8.0000
----------------------------------------------------------------
               Coef.   Std.Err.     z     P>|z|    [0.025   0.975]
----------------------------------------------------------------
const         -2.6867    0.6186  -4.3429  0.0000  -3.8992  -1.4742
PSALevel       1.0577    0.6198   1.7067  0.0879  -0.1570   2.2725
CancerVol      1.5502    0.6859   2.2599  0.0238   0.2058   2.8945
================================================================
```

Figure 5: Maximum Likelihood Estimates of Logistic Regression Function - *Python* Output.

Thus, the estimated logistic response function is:

$$\hat{\pi} = [1 + exp(-2.6867 + 1.0577X_1 + 1.5502X_2)]^{-1} \tag{7}$$

Note: Although the PSALevel predictor is not of 5% significance (p-value=0.0879), I did find it necessary to maintain it within the model. When removed, the Residual Deviance score of 44.628 from Figure 4 rose to a value of 48.123. Therefore, I've deemed it significant, and a valuable and impactful variable to achieve high model accuracy, and have not removed it from this subset of predictors.

With the estimated logistic regression equation now developed, it is left to consider second-order options and make adjustments if required, analyze the residuals and influential observations, test goodness of fit, apply a prediction rule for new observations, and finally apply the final model to the validation data and evaluate the results.

4.2.3 Geometric Interpretation

When fitting a standard multiple logistic regression model with two predictors, the estimated regression shape is an S-shaped surface in three-dimensional space. Figure 6 displays a three-dimensional plot of the estimated logistic response function that depicts the relationship between the diagnosis of high grade prostate cancer (Y, the binary outcome) and two continuous predictors, PSA Level (X_1) and Cancer Volume (X_2).

This surface increases in an approximately linear fashion with increasing values of PSA Level and Cancer Volume, but levels off and is nearly horizontal for very small and large values of these predictors.

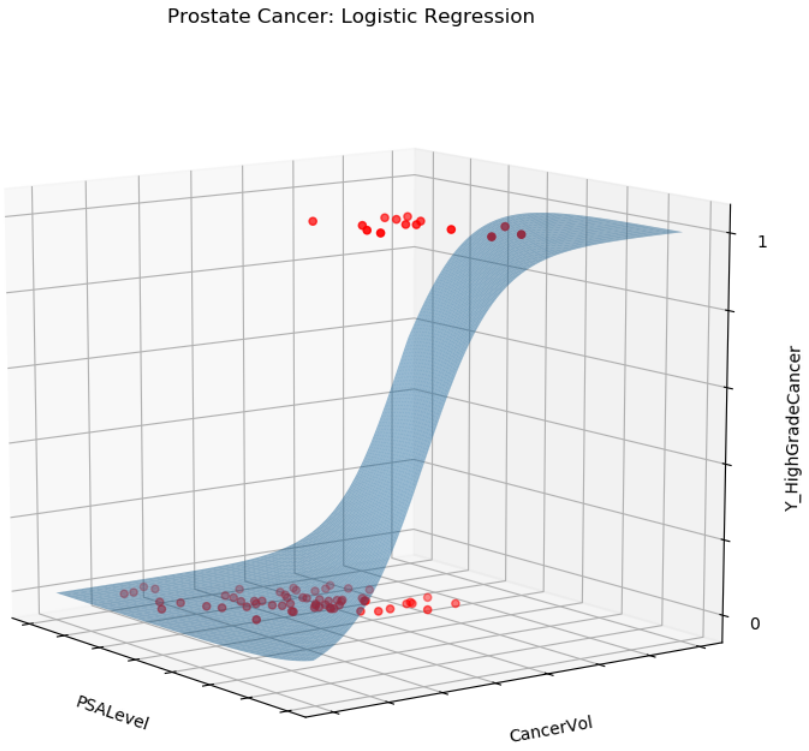


Figure 6: Three-Dimensional Fitted Logistic Response Surface.

4.2.4 Second-Order Predictors

Occasionally, the first-order logistic model may not provide a sufficient fit to the data, and the inclusion of higher-order predictors may be considered. I'll conclude my model development stage by attempting to fit the Prostate Cancer data to a *polynomial logistic* regression model of the second order, and analyze the results.

For simplicity, a 2^{nd}-order polynomial model in *two* predictors has a logit response function as:

$$logit(\pi) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 \tag{8}$$

and can be extended to more predictors by the inclusion of additional variables, their coefficients, and accompanying cross terms. Please recall, the Prostate Cancer data set considers 7 predictors.

In many situations the true regression function has one or more peaks or valleys, and in such cases a polynomial function can provide a satisfactory approximation. However, a polynomial fit was not successful here, as indicated by non-significant p-values across all predictors, at 5% significance (Figure 7).

```
Coefficients:
                                    Estimate Std. Error z value Pr(>|z|)
(Intercept)                           -3.044      1.069  -2.848  0.00439 **
poly(PSALevel, 2)1                     9.569      9.112   1.050  0.29365
poly(PSALevel, 2)2                     9.873      9.378   1.053  0.29243
poly(CancerVol, 2)1                   10.207     13.825   0.738  0.46034
poly(CancerVol, 2)2                    2.399      9.534   0.252  0.80135
poly(Weight, 2)1                     -20.609     18.220  -1.131  0.25800
poly(Weight, 2)2                     -18.912     18.242  -1.037  0.29987
poly(Age, 2)1                          2.823      5.365   0.526  0.59883
poly(Age, 2)2                          6.732      4.524   1.488  0.13680
poly(BenignProstaticHyperplasia, 2)1   8.187      7.551   1.084  0.27826
poly(BenignProstaticHyperplasia, 2)2   7.962      7.334   1.086  0.27765
SeminalVesicleInvasion1               -1.045      1.204  -0.868  0.38547
poly(CapsularPenetration, 2)1          2.485      4.076   0.610  0.54217
poly(CapsularPenetration, 2)2         -7.931      4.368  -1.815  0.06945 .
```

Figure 7: Logistic Regression Fit for Second-Order Model - **R** Output.

Additionally, my preliminary scatter plot analysis did not indicate any reason to believe a polynomial fit would be suitable in this study. For example, PSALevel was a major focus of this study and I've provided the scatter plot below in Figure 8. Additional scatter plots are provided in the Appendix, §7.1.

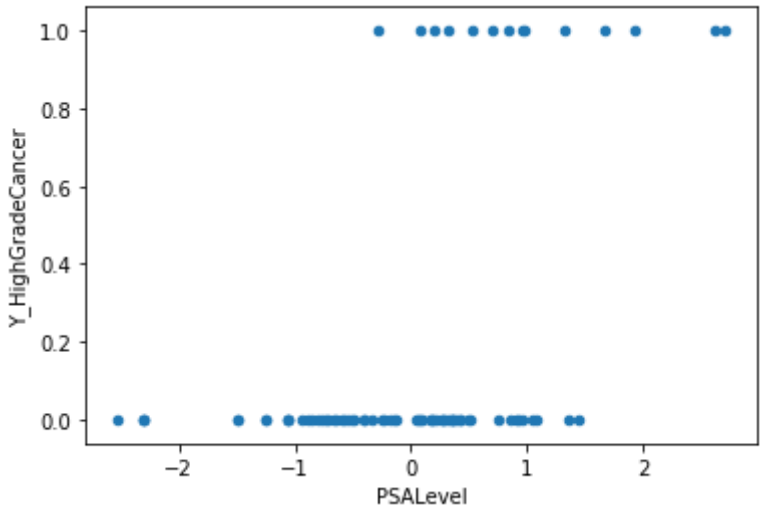


Figure 8: PSALevel vs Y_HighGradeCancer Scatterplot - Train Data.

Without evidence to be concerned of successfully fitting a model with second-order predictors, I will move forward with my analysis of the previously developed multiple logistic linear regression model.

4.3 Analysis of Residuals

In this section I will discuss the analysis of residuals and the identification of any influential observations for logistic regression. Due to the nature of logistic regression, and the fact that non-constant variance is always present in this setting, I will focus only on the detection of model inadequacy.

4.3.1 Logistic Regression Residuals

If the logistic regression model is correct, then $E[Y_i] = \pi_i$ and it follows that:

$$E[Y_i - \hat{\pi}_i] = E[e_i] = 0 \tag{9}$$

This suggests that if the model is correct, a lowess smooth of the plot of residuals against the linear predictor $\hat{\pi}_i'$ should result in approximately a horizontal line with zero intercept. Any significant departure from this suggests that the model may be inadequate. Shown in Figure 9 are the Pearson residuals plotted against the linear predictor, with the lowess smooth superimposed.

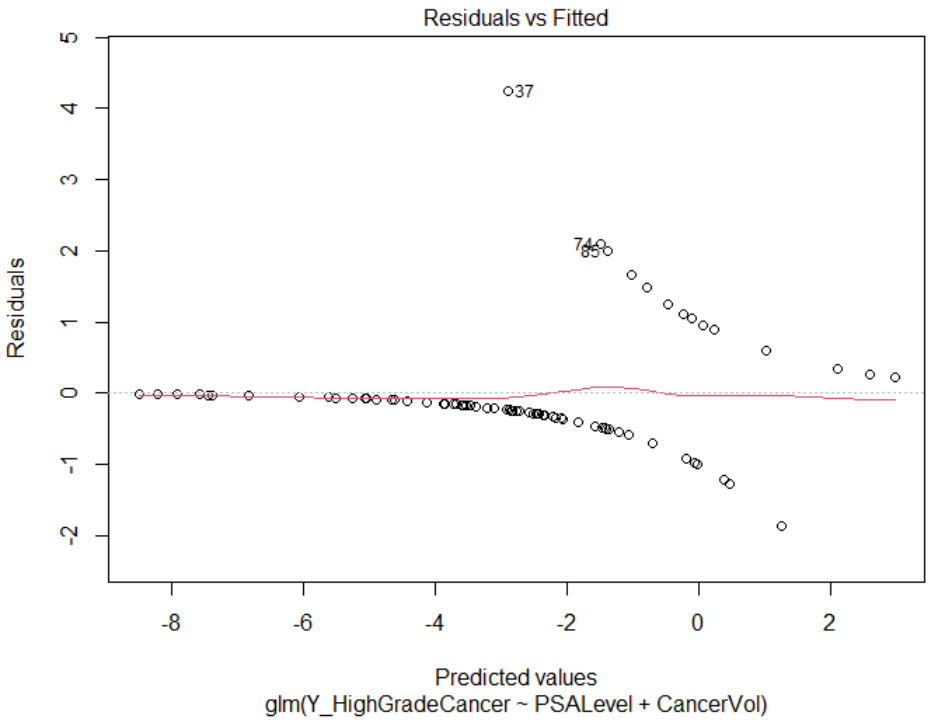


Figure 9: Pearson Residual Plot with Lowess Smooth.

Looking at the plot, the lowess smooth adequately approximates a line having zero slope and zero intercept, and I conclude that no significant model inadequacy is apparent.

4.3.2 Influential Observations

To aid in the identification of influential observations, I will use the **Cook's Distance** statistic, D_i, which measures the standardized change in the linear predictor $\hat{\pi}_i$ when the ith case is deleted. Cook's distances are listed in the **R** Appendix §7.2 for a portion of the Prostate Cancer testing data.

The plot of distances in Figure 10 identifies observation 90 as being the most outlying in the X space, and therefore potentially influential - observations 37 and 91 also read relatively high values. Observation 90 was temporarily deleted and the logistic regression fit was obtained. The results were not particularly different from those obtained from the full test set, and the observation was retained. **Note**: I additionally and temporarily removed observations 37 and 91 and obtained a fit to the updated final model. The results were not particularly different, and those records were also retained. Thus, no changes to the model are yet necessary.

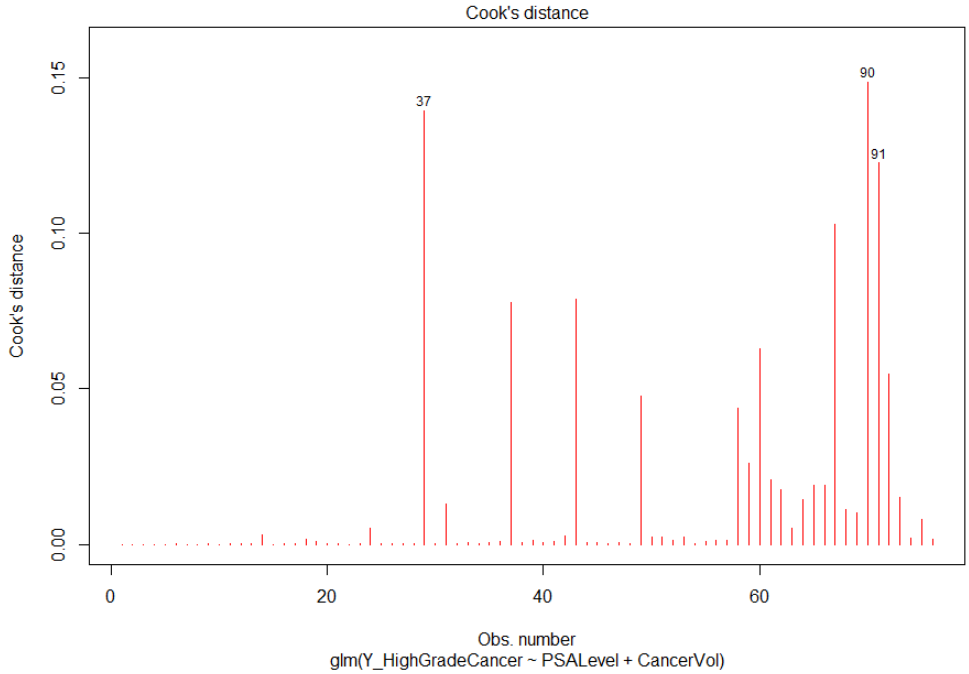


Figure 10: Index Plot of Cook's Distances.

4.4 Goodness Of Fit Evaluation

The appropriateness of the fitted logistic regression model needs to be examined before it is accepted for use. In particular, we need to examine whether the estimated response function for the data is monotonic and sigmoidal in shape, as are logistic response functions. Here I will employ the Hosmer-Lemeshow test, which is useful for unreplicated data sets, as is the Prostate Cancer data. The test can detect major departures from a logistic response function, and the alternatives of interest are as follows:

$$\begin{aligned} H_0 &: E[Y] = [1 + exp(-\mathbf{X}'\boldsymbol{\beta})]^{-1} \\ H_1 &: E[Y] \neq [1 + exp(-\mathbf{X}'\boldsymbol{\beta})]^{-1} \end{aligned} \tag{10}$$

4.4.1 Hosmer-Lemeshow

The Hosmer-Lemeshow Goodness of Fit procedure consists of grouping that data into classes with similar fitted values $\hat{\pi}_i$, with approximately the same number of cases in each class. Once the groups are formed, the Hosmer-Lemeshow goodness of fit statistic is calculated by using the Pearson chi-square test statistic of observed and expected frequencies. The test statistic is known to be well approximated by the chi-square distribution with $c - 2$ degrees of freedom.

$$\chi^2 = \sum_{j=1}^{c} \sum_{k=0}^{1} \frac{(O_{jk} - E_{jk})^2}{E_{jk}} \tag{11}$$

The output from **R** using 5 groups is shown in Figure 11 below.

```
                          yhat0      yhat1 y0 y1
[0.000206,0.00973] 15.941605 0.05839507 16  0
(0.00973,0.0521]   14.592690 0.40730980 15  0
(0.0521,0.103]     13.898096 1.10190404 14  1
(0.103,0.333]      11.889917 3.11008275 11  4
(0.333,0.951]       5.677692 9.32230835  6  9
    Hosmer and Lemeshow goodness of fit (GOF) test

data:  logit_red$y, fitted(logit_red)
X-squared = 0.83815, df = 3, p-value = 0.8403
```

Figure 11: Hosmer-Lemshow Goodness of Fit Test for Logistic Regression Function.

Large values of the test statistic X^2 indicate that the logistic response function is not appropriate. The decision rule for testing the alternatives (Eqn. 10) when controlling the level of significance at α therefore is:

$$\begin{aligned} &\text{If } X^2 \leq \chi^2(1-\alpha; c-p), \text{ conclude } H_0 \\ &\text{If } X^2 > \chi^2(1-\alpha; c-p), \text{ conclude } H_1 \end{aligned} \tag{12}$$

Thus, for $\alpha = 0.05$ and $c - 2 = 5 - 2 = 3$, we require $\chi^2(0.95; 3) = 7.81$. Since $X^2 = 0.838 \leq 7.81$, we conclude H_0, that the logistic response function is appropriate. The p-value of the test is 0.8403.

4.5 Development of ROC Curve

Multiple logistic regression is often employed for making predictions for new observations. The *receiver operating characteristic* (ROC) *curve* plots $P(\hat{Y} = 1|Y = 1)$ as a function of $1 - P(\hat{Y} = 0|Y = 0)$ and is an effective way to graphically display prediction rule information, and possible cutoff points.

The "True Positive" y-axis on an ROC curve is also known as *sensitivity*, and the "False Positive" x-axis is 1-*specificity*. Figure 12 below exhibits the ROC curve for my model (Eqn. 7) for all possible cut points between 0 and 1.

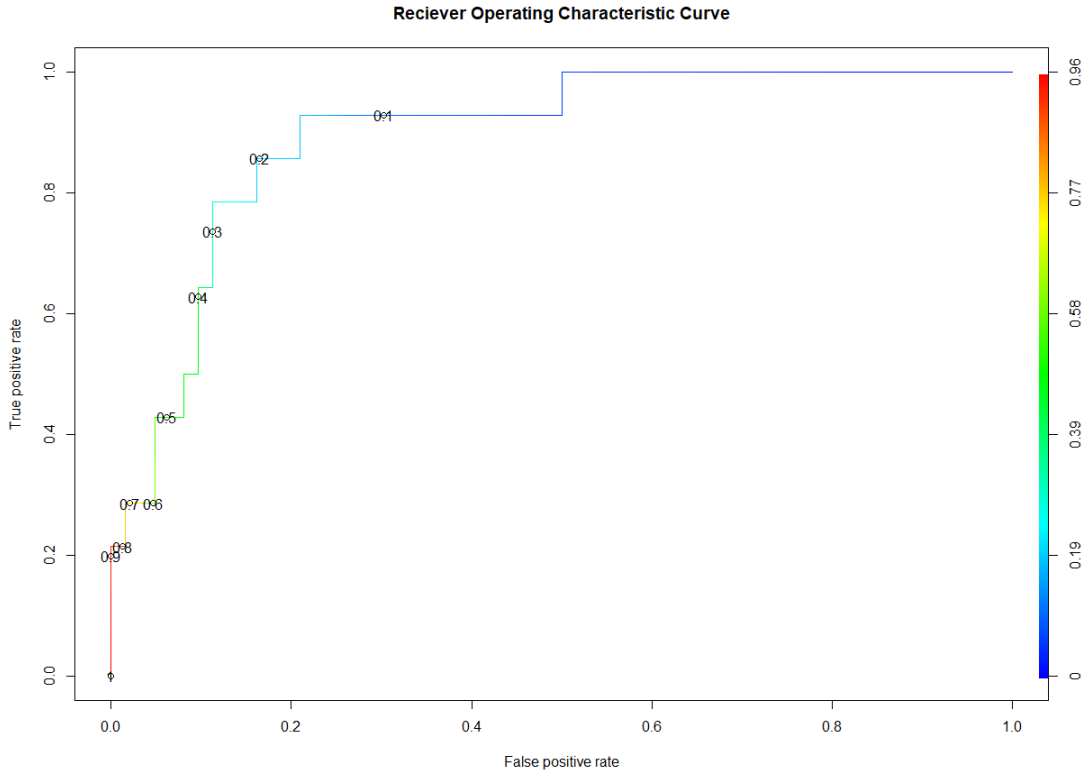


Figure 12: ROC Curve.

4.5.1 Prediction Rule

In the training data set (which represented a random 80% of the 97 provided observations), there were 14 men who were observed as high grade cancer patients; hence the estimated proportion of persons who had high grade cancer is $14/76 = 0.184$. This proportion can be used as the starting point in the search for the best cutoff in the prediction rule.

Thus, if $\hat{\pi}_h$ represents a newly fitted observation, my first prediction rule investigated is:

$$\text{Predict 1 if } \hat{\pi}_h \geq 0.184\text{; predict 0 if } \hat{\pi}_h < 0.184 \tag{13}$$

The Confusion Matrix of Table 1 below provides a summary of the number of correct and incorrect classifications based on the initial prediction rule (Eqn. 13). Of the 62 men without high grade cancer, 13 would be incorrectly predicted to have high grade cancer, or an error rate of 21.0%. Furthermore, of the 14 persons with high grade cancer, 1 would be incorrectly predicted to not have high grade cancer, or 7.1%. Altogether, $13 + 1 = 14$ of the 76 predictions would be incorrect, so that the prediction error rate for the rule is $14/76 = 0.184$ or 18.4%. Coincidentally, the model exactly matches our training set proportions with the current prediction rule.

Prediction Rule Eqn. 13			
True Classification	$\hat{Y} = 0$	$\hat{Y} = 1$	Total
$Y = 0$	49	13	62
$Y = 1$	1	13	14
Total	50	26	76

Table 1: Classification based on Logistic Response Function Eqn. 7 and Prediction Rule Eqn. 13.

With this baseline understood, it is straightforward to choose a stronger cutoff point in utilizing the ROC curve of Figure 12. As detailed above, the false-positive rate is not ideal at 21.0% - there are too many cases where a man may opt for additional screening and treatment, even invasive actions, because he believes he has high grade prostate cancer. It will be wise to now reference the ROC curve to better choose a prediction cutoff, while also not significantly disturbing the false-negative accuracy for the worse.

Looking at Figure 12, a step occurs at 0.20 and I use this value for my new cutoff candidate. Thus, my updated prediction rule is stated as follows:

$$\text{Predict 1 if } \hat{\pi}_h \geq 0.20\text{; predict 0 if } \hat{\pi}_h < 0.20 \tag{14}$$

and the effects of this change can be summarized by the Confusion Matrix in Table 2 below.

Prediction Rule Eqn. 14			
True Classification	$\hat{Y} = 0$	$\hat{Y} = 1$	Total
$Y = 0$	52	10	62
$Y = 1$	2	12	14
Total	54	226	76

Table 2: Classification based on Logistic Response Function Eqn. 7 and Prediction Rule Eqn. 14.

Here, of the 62 men without high grade cancer, 10 are incorrectly predicted, or an error rate of 16.1%. Continuing, of the 14 men with high grade cancer, 2 would be incorrectly predicted, or an error rate of 14.3%. Altogether, updated prediction rule (Eqn. 14) now provides a total error rate of $12/76 = 0.158$ or 15.8%. Thus, the model accuracy has now increased with a significantly better false-positive rate, which is intended to reduce unnecessary financial stress across the healthcare economy.

4.6 Model: Strengths and Weaknesses

4.6.1 Strengths

-The two predictors which build the final logistic model are PSA Level and Cancer Volume, and they both are adequately correlated with the dependent (outcome) variable Y_HighGradeCancer - their correlation values are 0.489 and 0.493, respectively. In fact, they are more correlated to the dependent variable than any other predictors of the data set. A consumable heat-map version of a correlation matrix is provided by Figure 13 below, with a color legend given in the upper-left corner.

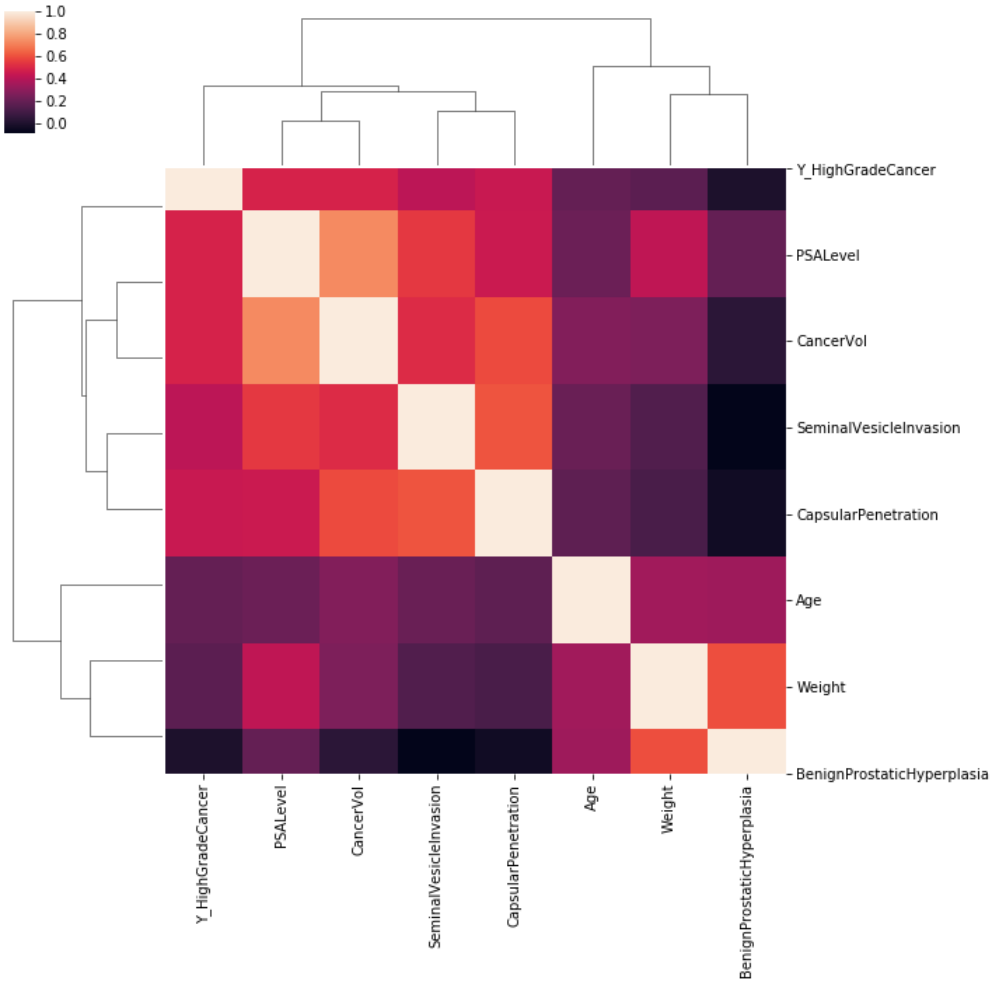


Figure 13: Correlation Heatmap - Train Data

-The final model has an excellent accuracy score of 84.2% against training data. If this model also performs well in the validation step, then deploying such a model in use could possibly help non high grade cancer men be categorized as such, and thus not pursue unnecessary invasive testing and not inflate costs within the healthcare system. Also, by properly identifying those men who are high grade cancer patients, treatment and a plan can be devised sooner, as well as doing so by only use of PSA Level and Cancer Volume information, and not invasive testing.

4.6.2 Weaknesses

-An initial concern while building this model occurs at the level of the provided raw data, namely the existence of multicollinearity. Figure 14 below is the Correlation Matrix of the two final predictors which built the final logistic model: PSA Level and Cancer Volume.

	PSALevel	CancerVol
PSALevel	1.000000	0.737585
CancerVol	0.737585	1.000000

Figure 14: PSALevel vs. CancerVol Correlation Matrix - Train Data

As shown, PSA Level and Cancer Volume have a mild correlation value of 0.738 in the full data set. One primary danger in designing models with multicollinearity is that small changes to the input data can lead to large changes in the model, which can further lead to over-fitting. Therefore, this logistic model may be considered mildly "noisy", sensitive, and not particularly robust.

-The Goodness of Fit Evaluation of §4.4 deserves some concern regarding the Pearson chi-square test. As described previously, the Hosmer-Lemeshow procedure was utilized to determine a goodness of fit, and the test statistic is known to be well approximated by the chi-square distribution with $c - 2$ degrees of freedom (Eqn. 11). However, in view of the **R** output (Figure 11) with 5 groupings, the expected values (y_1) returned were: 0, 0, 1, 4, 9. Because many values are less than 5, and two of the expected values equal 0, the conditions for a chi-square test may be voided, and it may not be an appropriate test procedure here. At the very lest, the results of the Hosmer-Lemeshow test should be accepted carefully.

-The final model may produce high false-negative rates. In view of Figure 15-A we see that the first prediction cutoff of 0.184 produced 1 count of false-negatives (7.1% error rate), and an overall model accuracy of 81.6%. After ROC analysis the final prediction cutoff I've employed is 0.20. By Figure 15-B this rule produces 2 counts of false-negatives (14.3% error rate), and an overall model accuracy of 84.2%. Thus the overall model accuracy has improved 2.6 percentage points by correctly predicting more non high grade cancer cases, but has concurrently doubled the false-negative rate. Because correctly identifying high grade cancer patients may be considered most important, this arrangement may be a downfall of the final model.

```
TRAINING DATA
Prediction Rule: 0.184
Confusion Matrix:

           PredictedValue
ActualValue FALSE TRUE
          0    49   13
          1     1   13

The calculated error is: 18.4 %
The calculated accuracy is: 81.6 %
```

(a) Confusion Matrix - 0.184 Cutoff Rule.

```
TRAINING DATA
Prediction Rule: 0.2
Confusion Matrix:

           PredictedValue
ActualValue FALSE TRUE
          0    52   10
          1     2   12

The calculated error is: 15.8 %
The calculated accuracy is: 84.2 %
```

(b) Confusion Matrix - 0.20 Cutoff Rule

Figure 15: Classification Based on Logistic Response Function (Eqn. 7) and Prediction Rules (Eqn. 13) and (Eqn. 14).

5 Conclusion

5.1 Model Validation

The reliability of the chosen model, prediction rule, and prediction error rate from the training data is examined by now applying the prediction rule to the validation data set (i.e. the remaining 20% of data). As I will show, the new prediction error rate is about the same as that for the model-building data set, and gives a reliable indication of the predictive ability of the fitted logistic regression model and the chosen prediction rule. If the new and unseen data had lead to a considerably higher prediction error rate, then the fitted logistic regression model and the chosen prediction rule would not predict new observations well.

In my Prostate Cancer logistics model, the fitted logistic regression function (Eqn. 7) based on the model-building data set:

$$\hat{\pi} = [1 + exp(-2.6867 + 1.0577X_1 + 1.5502X_2)]^{-1}$$

was used to calculate estimated probabilities $\hat{\pi}_h$ for the validation data set. The chosen prediction rule (Eqn. 14):

$$\text{Predict 1 if } \hat{\pi}_h \geq 0.20\text{; predict 0 if } \hat{\pi}_h < 0.20$$

was then applied to these estimated probabilities. The percent prediction error rates are summarized in Figure 16 and Table 3 below:

```
TESTING DATA
Prediction Rule: 0.2
Confusion Matrix:

            PredictedValue
ActualValue FALSE TRUE
          0    13    1
          1     2    5

The calculated error is: 14.3 %
The calculated accuracy is: 85.7 %
```

Figure 16: Confusion Matrix - Validation Data

Disease Status		
With High Grade Cancer	Without High Grade Cancer	Total
28.6%	7.1%	14.3%

Table 3: Percent Prediction Error Rates - Validation Data

Note that the total prediction error rate of 14.3% is approximately equal to, or very similar to, the 15.8% error rate based on the model-building data set. Therefore the latter is a reliable indicator of the predictive capability of the fitted logistic regression model and the chosen prediction rule. The accuracy is seen to be 85.7%.

5.2 Final Remarks

The primary purpose of this study was to assess the strength of the association between each of the predictor variables with the response variable, the predictable nature of PSA Level, and the probability of a man having been diagnosed with high grade prostate cancer over low grade. We can now examine the odds ratios of the fitted model (Eqn. 7) to help address these questions.

The interpretation for multiple logistic regression is that the estimated odds ratio for the predictor variable X_k assumes that all other predictor variables are held constant. In view of the fitted model (Eqn. 7) the estimated coefficients are: $\hat{\beta}_0 = -2.6867$, $\hat{\beta}_1 = 1.0577$, and $\hat{\beta}_2 = 1.5502$. Therefore we can see, for instance, that the odds of a man being diagnosed with high grade prostate cancer increase by about 5.8% for each additional score of PSA Level, for a given Cancer Volume. This means each unit increase of PSA Level increases the odds of said diagnosis by 5.8%. Similarly, the odds of a man being diagnosed with high grade prostate cancer increase by 55.0% for each unit increase in cancer volume.

Thus, these calculated odds ratios suggest that Cancer Volume has a significantly larger association to the outcome (a diagnosis of high grade prostate cancer) than PSA Level. However, PSA Level proved more significant than all other predictors in my analysis of §4.2, was

used to achieve 85.7% accuracy against validation data in §5.1, and is both a cost-effective and noninvasive screening procedure for prostate cancer grade classification.

Lastly, because this study is observational by nature (all 97 selected men were predetermined to have been diagnosed with prostate cancer), we must be careful about the scope of inferences we draw. Since the data are observational, the result cannot be used as proof that high grade patients test with higher PSA Level; the possibility of confounding variables cannot be excluded. Furthermore, since the individuals were not said be be drawn at random from the population of men about to undergo radical prostectomies, inference to a broader population is not justified.

6 References

- Devore, J. (2012). *Probability and Statistics for Engineering and the Sciences* (8th ed.). N.p.: Brooks/Cole.

- DOI: 10.1200/JCO.2019.37.7_suppl.116 *Journal of Clinical Oncology* Vol 37, no. 7_suppl (March 01, 2019) 116-116.

- Kutner, M. H., Nachtsheim, C. J., & Neter, J. (2004). *Applied Linear Regression Models* (4th ed.). N.p.: McGraw-Hill/Irwin.

- Prostate Cancer: Prostate Cancer Information and Overview. (n.d.). Retrieved November 23, 2020, from https://www.cancer.org/cancer/prostate-cancer.html

- Ramsey, F. L., & Schafer, D. W. (2013). *The Statistical Sleuth: A Course in Methods of Data Analysis* (3rd ed.). N.p.: Brooks/Cole Cengage Learning.

7 Appendix

7.1 Python

1.0-jo-extract-prostate-data

November 23, 2020

1 Extract Prostate Cancer Data From Textbook URL

[1]:
```
import urllib.request
import os.path
```

[2]:
```
# open connection to URL
webUrl = 'http://www.cnachtsheim-text.csom.umn.edu/Kutner/
 ↪Appendix%20C%20Data%20Sets/APPENC05.txt'
response = urllib.request.urlopen(webUrl)
data = response.read()

# decode bytes to string
data = data.decode('utf-8')
data = data.replace('\r', '')
```

[3]:
```
# write to .txt file and store
filename = 'APPENC05'
out_filename = '../data/raw/' + filename + '.txt'
if not os.path.isdir(out_filename):
    output_file = open(out_filename, 'w')
    output_file.write(data)
    output_file.close()
```

2.0-jo-data-exploration

November 23, 2020

1 Exploring and Processing Data

[1]:
```
# imports
import pandas as pd
import numpy as np
import os
```

[2]:
```
# allow plots/visuals to exist inline within this workbook
%matplotlib inline
```

1.1 Import Data

[3]:
```
# set path to raw data
raw_data_path = os.path.join(os.path.pardir, 'data', 'raw')
data_file_path = os.path.join(raw_data_path, 'APPENC05.txt')
```

[4]:
```
# read the default .txt file and print it
f = open(data_file_path, 'r')
print(f.read(500)) # print the first 500 characters
f.close()
```

```
  1     0.651    0.5599    15.959    50    0.0000    0    0.0000    6
  2     0.852    0.3716    27.660    58    0.0000    0    0.0000    7
  3     0.852    0.6005    14.732    74    0.0000    0    0.0000    7
  4     0.852    0.3012    26.576    58    0.0000    0    0.0000    6
  5     1.448    2.1170    30.877    62    0.0000    0    0.0000    6
  6     2.160    0.3499    25.280    50    0.0000    0    0.0000    6
  7     2.160    2.0959    32.137    64    1.8589    0    0.0000    6
```

[5]:
```
# create pandas dataframe with column headers
cols = [
    'Obs', 'PSALevel','CancerVol', 'Weight',
    'Age', 'BenignProstaticHyperplasia', 'SeminalVesicleInvasion',
    'CapsularPenetration', 'GleasonScore'
    ]

df = pd.read_fwf(data_file_path, names=cols, index_col='Obs')
```

1.2 Basic Structure

By instruction of the case study, I will create a new binary response variable Y, called high-grade cancer, by letting Y=1 if Gleason score equals 8, and Y=0 otherwise (i.e., if Gleason score equals 6 or 7). Let the new field be called "Y_HighGradeCancer" within the dataset. - Note: Y_HighGradeCancer and SeminalVesicleInvasion are binary indicator variables.

[6]:
```
df['Y_HighGradeCancer'] = np.where(df['GleasonScore'] == 8, 1, 0)
```

[7]:
```
# use .head() to view the first 5 rows
df.head()
```

[7]:

Obs	PSALevel	CancerVol	Weight	Age	BenignProstaticHyperplasia	SeminalVesicleInvasion	CapsularPenetration	GleasonScore	Y_HighGradeCancer
1	0.651	0.5599	15.959	50	0.0	0	0.0	6	0
2	0.852	0.3716	27.660	58	0.0	0	0.0	7	0
3	0.852	0.6005	14.732	74	0.0	0	0.0	7	0
4	0.852	0.3012	26.576	58	0.0	0	0.0	6	0
5	1.448	2.1170	30.877	62	0.0	0	0.0	6	0

[8]:
```
# use .tail() to view the last 5 rows
df.tail()
```

[8]:

Obs	PSALevel	CancerVol	Weight	Age	BenignProstaticHyperplasia	SeminalVesicleInvasion	CapsularPenetration	GleasonScore
93	80.640	16.9455	48.424	68	0.0000	1	1.0	7
94	107.770	45.6042	49.402	44	0.0000	1	1.0	7
95	170.716	18.3568	29.964	52	0.0000	1	1.0	7
96	239.847	17.8143	43.380	68	4.7588	1	1.0	7
97	265.072	32.1367	52.985	68	1.5527	1	1.0	7

```
Obs
93                        1               3.7434            8
94                        1               8.7583            8
95                        1              11.7048            8
96                        1               4.7588            8
97                        1              18.1741            8

     Y_HighGradeCancer
Obs
93                   1
94                   1
95                   1
96                   1
97                   1
```

[9]:
```
# use .info() to get basic information about the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 97 entries, 1 to 97
Data columns (total 9 columns):
PSALevel                      97 non-null float64
CancerVol                     97 non-null float64
Weight                        97 non-null float64
Age                           97 non-null int64
BenignProstaticHyperplasia    97 non-null float64
SeminalVesicleInvasion        97 non-null int64
CapsularPenetration           97 non-null float64
GleasonScore                  97 non-null int64
Y_HighGradeCancer             97 non-null int32
dtypes: float64(5), int32(1), int64(3)
memory usage: 7.2 KB
```

[10]:
```
# filter rows based on condition
gleason_8 = len(df.loc[df.GleasonScore == 8, :])
gleason_not8 = len(df.loc[df.GleasonScore != 8, :])
print(f'Count of high-grade cancer: {gleason_8}')
print(f'Count of non high-grade cancer: {gleason_not8}')
```

```
Count of high-grade cancer: 21
Count of non high-grade cancer: 76
```

[11]:
```
# Y binary column: proportions
df.Y_HighGradeCancer.value_counts(normalize=True)
```

[11]:
```
0    0.783505
1    0.216495
Name: Y_HighGradeCancer, dtype: float64
```

1.3 Summary Statistics

[12]:
```
# use .describe() to view summary statistics for all numerical columns
df.describe()
```

[12]:

	PSALevel	CancerVol	Weight	Age	\
count	97.000000	97.000000	97.000000	97.000000	
mean	23.730134	6.998682	45.491361	63.865979	
std	40.782925	7.880869	45.705053	7.445117	
min	0.651000	0.259200	10.697000	41.000000	
25%	5.641000	1.665300	29.371000	60.000000	
50%	13.330000	4.263100	37.338000	65.000000	
75%	21.328000	8.414900	48.424000	68.000000	
max	265.072000	45.604200	450.339000	79.000000	

	BenignProstaticHyperplasia	SeminalVesicleInvasion	\
count	97.000000	97.000000	
mean	2.534725	0.216495	
std	3.031176	0.413995	
min	0.000000	0.000000	
25%	0.000000	0.000000	
50%	1.349900	0.000000	
75%	4.758800	0.000000	
max	10.277900	1.000000	

	CapsularPenetration	GleasonScore	Y_HighGradeCancer
count	97.000000	97.000000	97.000000
mean	2.245367	6.876289	0.216495
std	3.783329	0.739619	0.413995
min	0.000000	6.000000	0.000000
25%	0.000000	6.000000	0.000000
50%	0.449300	7.000000	0.000000
75%	3.254400	7.000000	0.000000
max	18.174100	8.000000	1.000000

- PSALevel, CancerVol, Weight and Age appear to have high standard deviation values. This may provoke a standardized dataset, or transformations, further in my analysis.
- Skewness will need to be investigated.

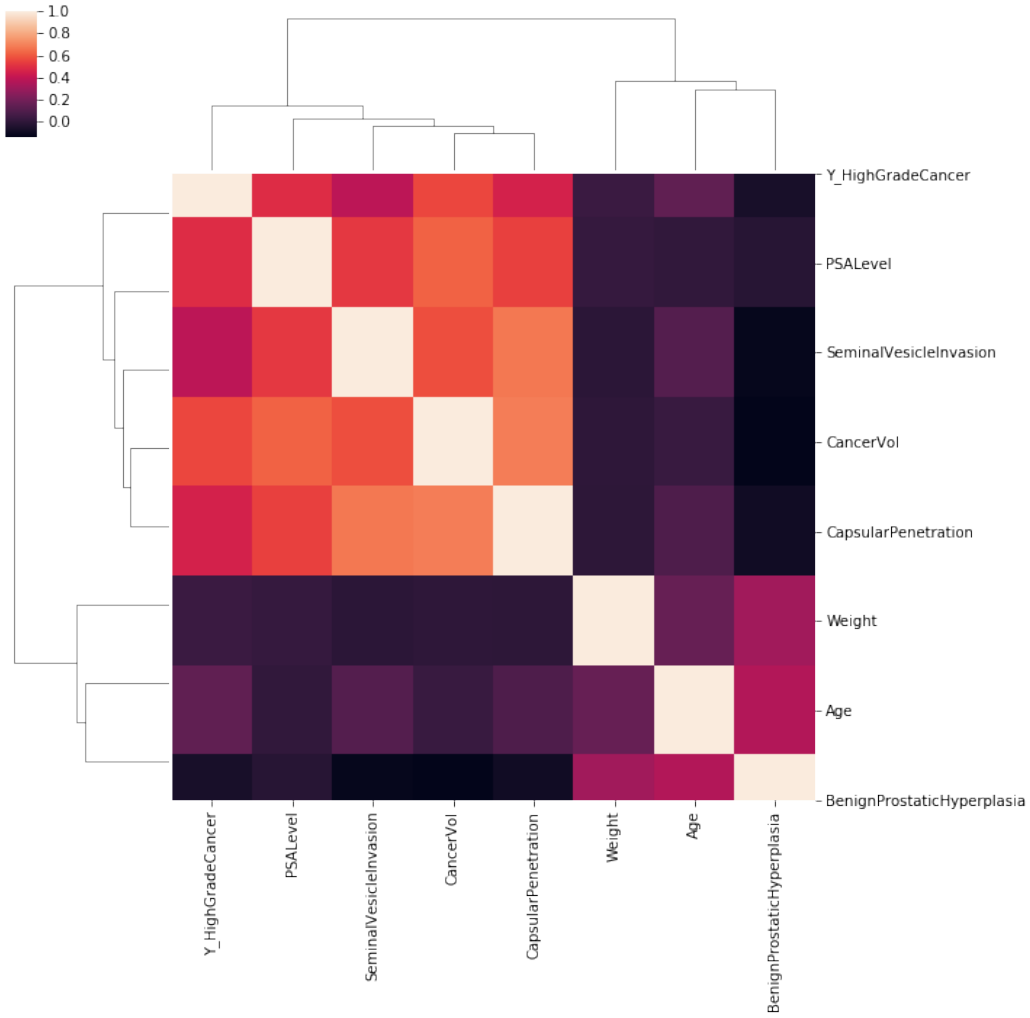
[13]:
```
# correlation matrix
# GleasonScore need not be considered
cols = [col for col in df.columns if col != 'GleasonScore']
df[cols].corr()
```

[13]:

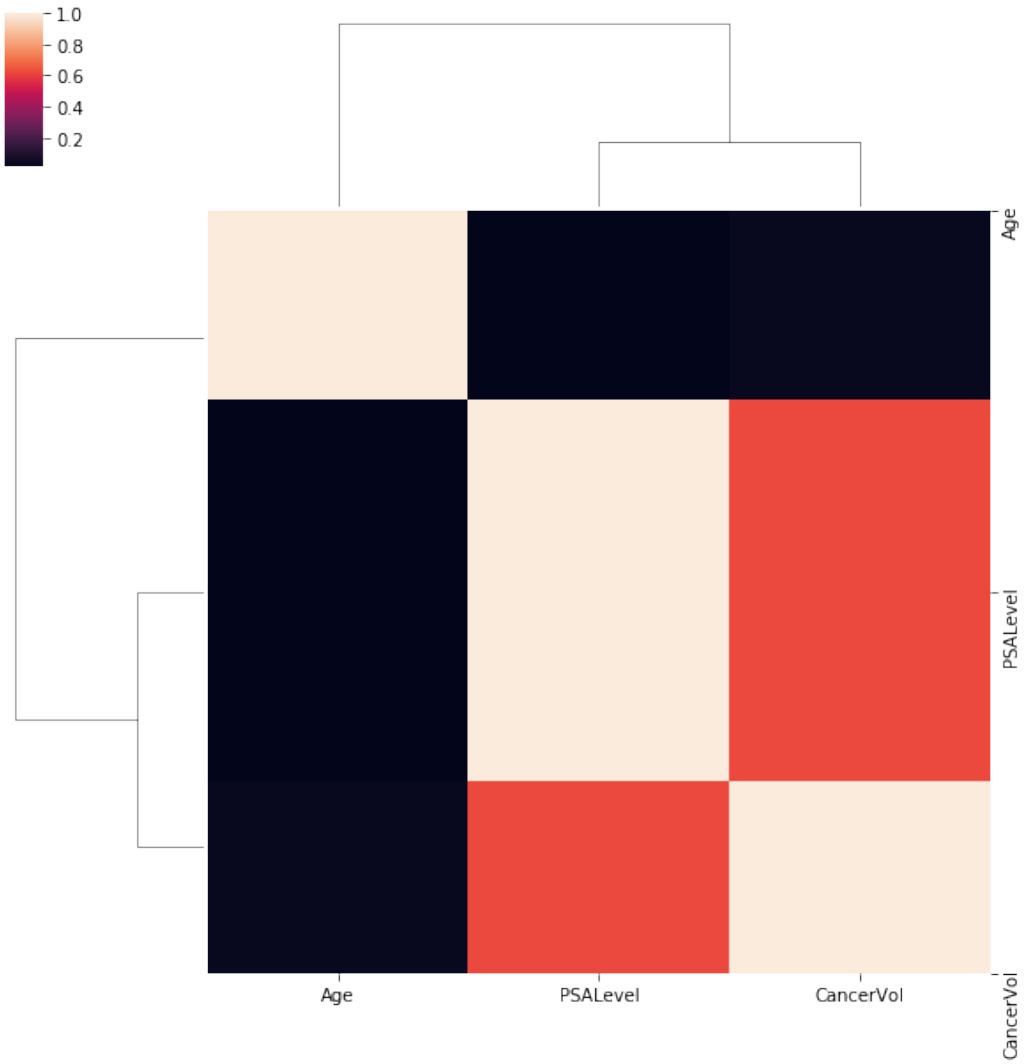
	PSALevel	CancerVol	Weight	Age	\
PSALevel	1.000000	0.624151	0.026213	0.017199	
CancerVol	0.624151	1.000000	0.005107	0.039094	
Weight	0.026213	0.005107	1.000000	0.164324	

```
Age                          0.017199   0.039094  0.164324  1.000000
BenignProstaticHyperplasia -0.016486  -0.133209  0.321849  0.366341
SeminalVesicleInvasion       0.528619   0.581742 -0.002410  0.117658
CapsularPenetration          0.550793   0.692897  0.001579  0.099555
Y_HighGradeCancer            0.497189   0.564645  0.039445  0.148074

                           BenignProstaticHyperplasia  \
PSALevel                                    -0.016486
CancerVol                                   -0.133209
Weight                                       0.321849
Age                                          0.366341
BenignProstaticHyperplasia                   1.000000
SeminalVesicleInvasion                      -0.119553
CapsularPenetration                         -0.083009
Y_HighGradeCancer                           -0.058032

                           SeminalVesicleInvasion  CapsularPenetration  \
PSALevel                                 0.528619             0.550793
CancerVol                                0.581742             0.692897
Weight                                  -0.002410             0.001579
Age                                      0.117658             0.099555
BenignProstaticHyperplasia              -0.119553            -0.083009
SeminalVesicleInvasion                   1.000000             0.680284
CapsularPenetration                      0.680284             1.000000
Y_HighGradeCancer                        0.392231             0.463134

                           Y_HighGradeCancer
PSALevel                            0.497189
CancerVol                           0.564645
Weight                              0.039445
Age                                 0.148074
BenignProstaticHyperplasia         -0.058032
SeminalVesicleInvasion              0.392231
CapsularPenetration                 0.463134
Y_HighGradeCancer                   1.000000
```

[14]:
```
# let's import searborn to help visualize the correlation matrix
import seaborn as sns

sns.clustermap(df[cols].corr());
```



[15]:
```
# view a few terms more closely
sns.clustermap(df[['PSALevel', 'CancerVol', 'Age']].corr());
```



[16]:
```
# correlation of final two predictors
df[['PSALevel', 'CancerVol']].corr()
```

[16]:

	PSALevel	CancerVol
PSALevel	1.000000	0.624151
CancerVol	0.624151	1.000000

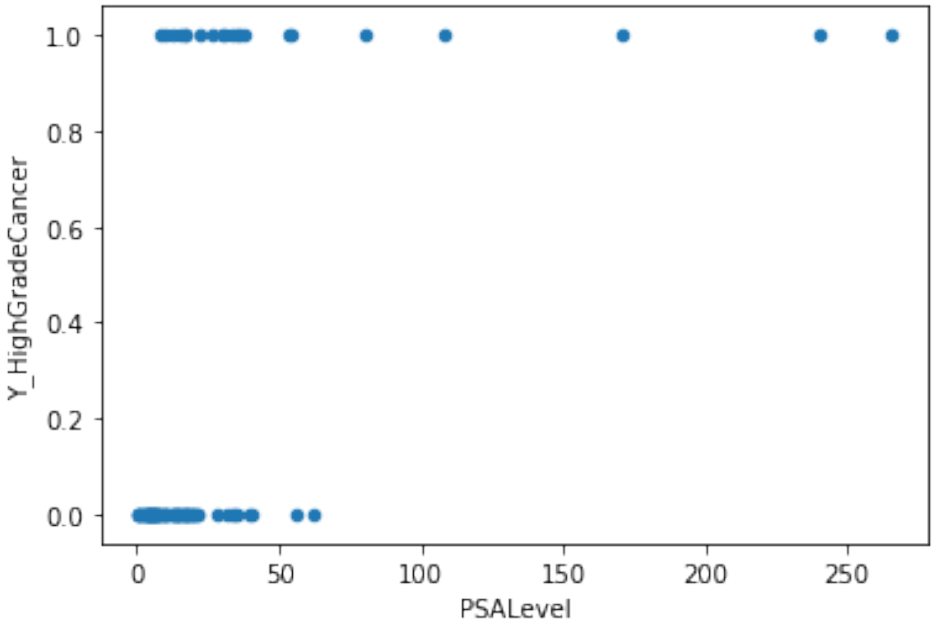
- PSALevel and CancerVol show a mild level of correlation: 0.624151

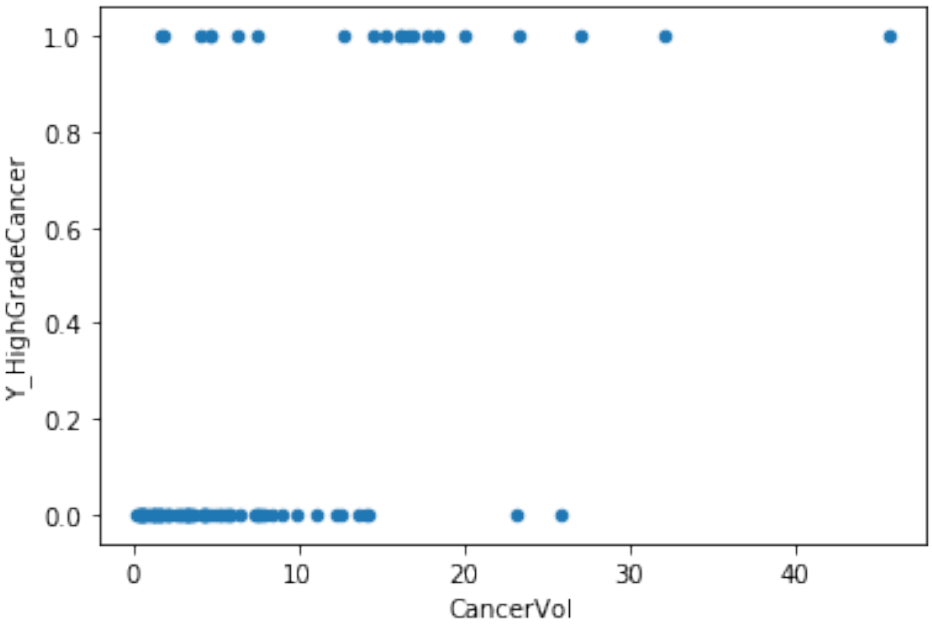
[17]:
```
# numerical features
# centrality measures
print(f'Mean Age: {round(df.Age.mean(), 2)}')
```

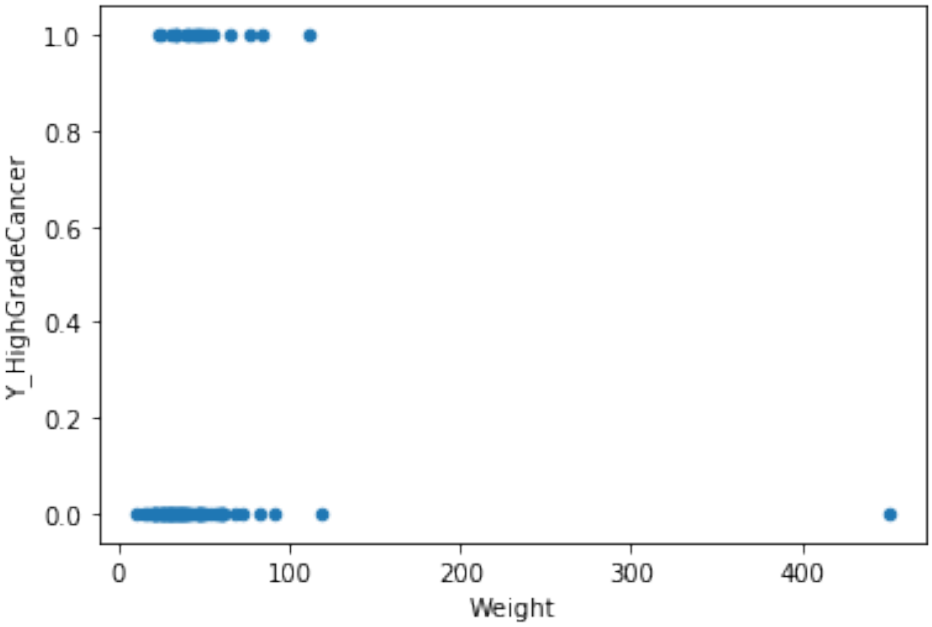
```
print(f'Median Age: {df.Age.median()}')
```

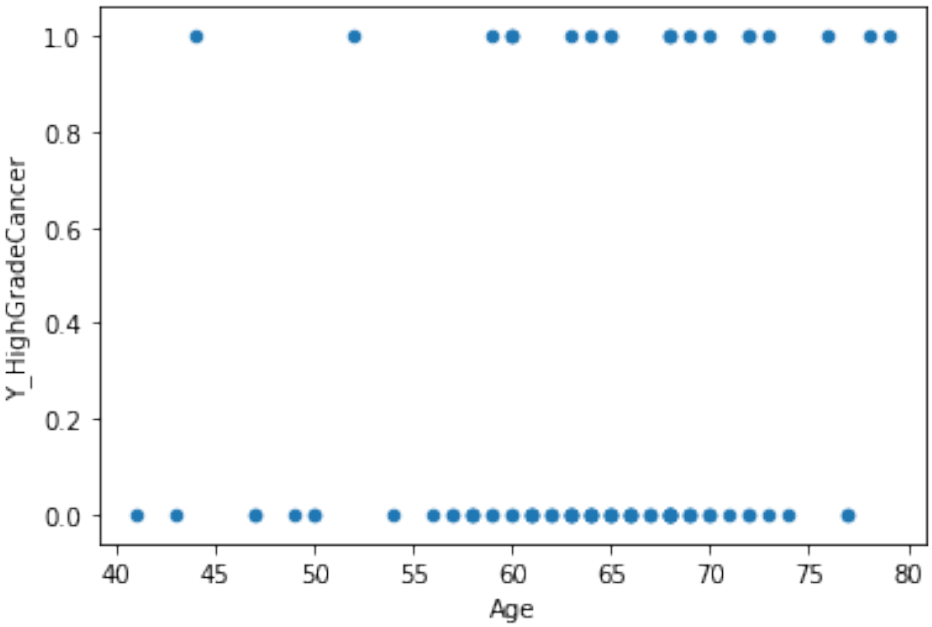
```
Mean Age: 63.87
Median Age: 65.0
```

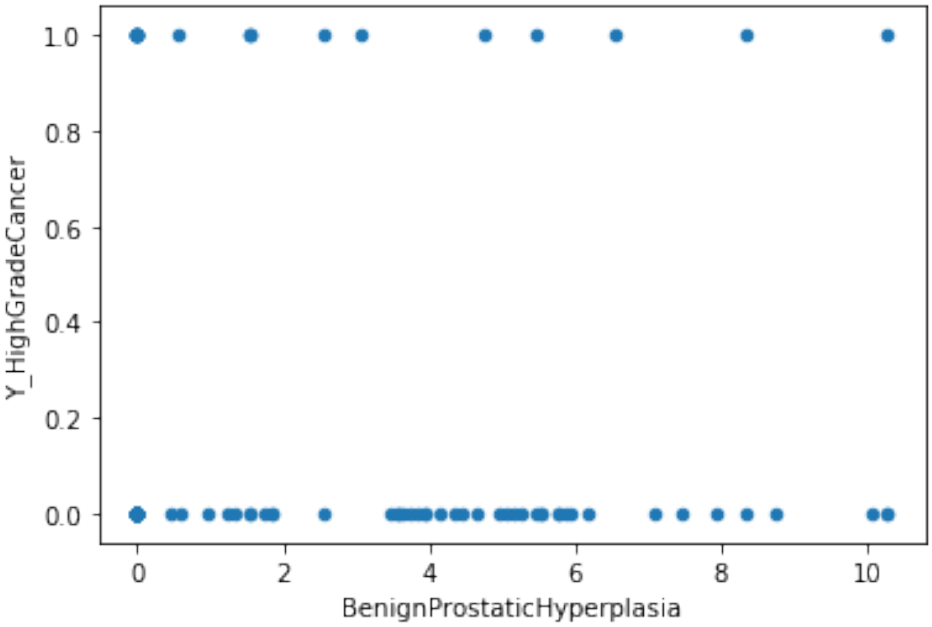
[18]:
```
# print all relevant scatter plots
# examine visuals for outliers
for col in cols:
    df[['Y_HighGradeCancer', col]].plot.scatter(x=col, y='Y_HighGradeCancer');
```

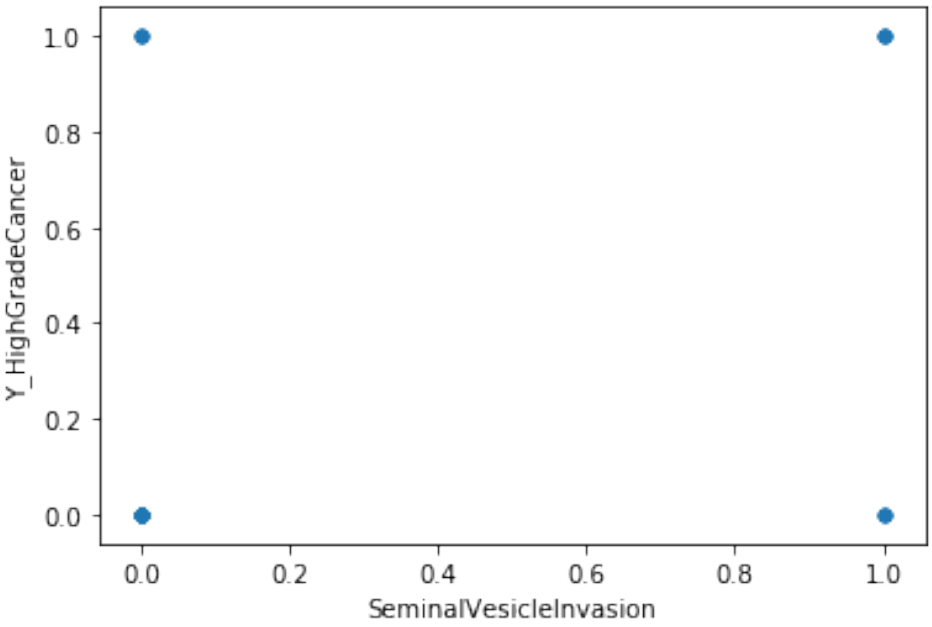


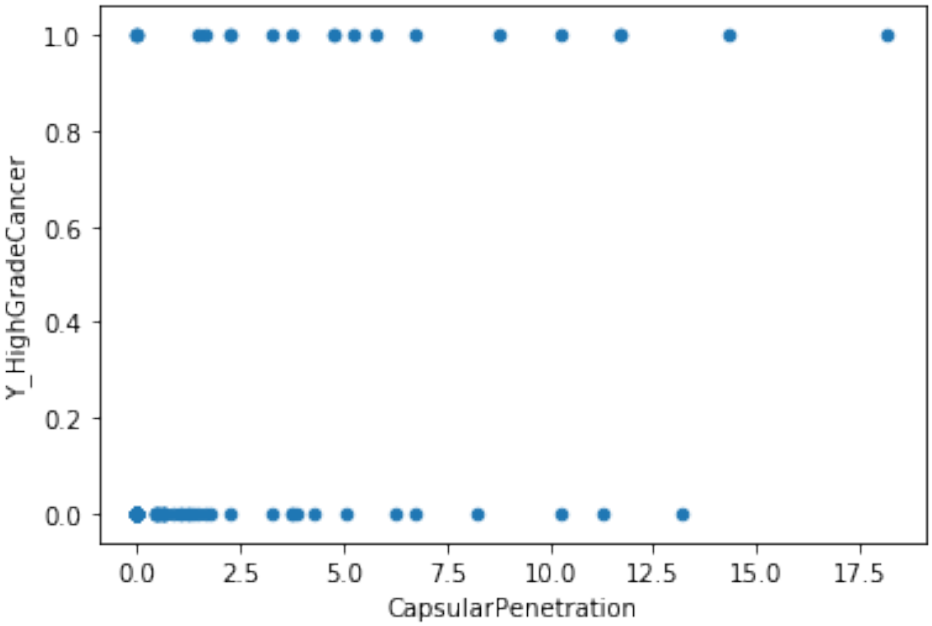

1.0
0.8
0.6
0.4
0.2
0.0
Y_HighGradeCancer
0
10
20
30
40
CancerVol

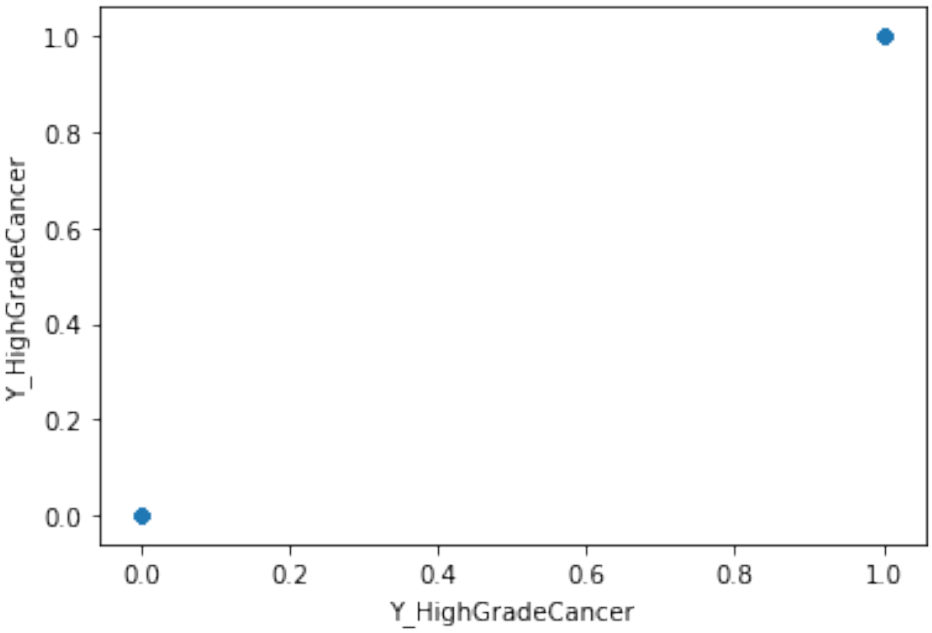

1.0
0.8
0.6
0.4
0.2
0.0
Y_HighGradeCancer
0
100
200
300
400
Weight


Y_HighGradeCancer
1.0
0.8
0.6
0.4
0.2
0.0
40
45
50
55
60
65
70
75
80
Age


Y_HighGradeCancer
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
BenignProstaticHyperplasia


1.0
0.8
0.6
0.4
0.2
0.0
Y_HighGradeCancer
0.0
0.2
0.4
0.6
0.8
1.0
SeminalVesicleInvasion


1.0
0.8
0.6
0.4
0.2
0.0
Y_HighGradeCancer
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
CapsularPenetration

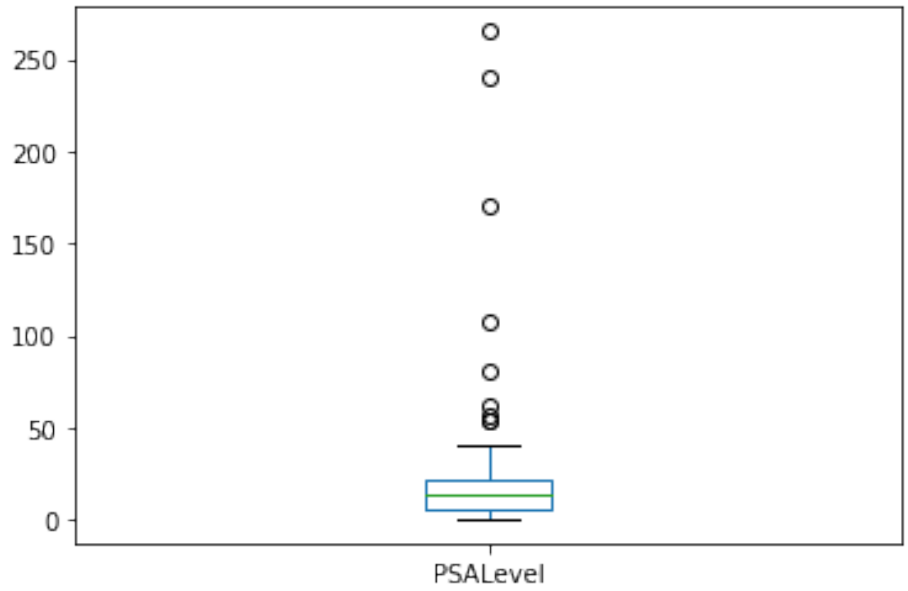


- PSALevel and CancerVol may be good predictors for Y_HighGradeCancer in a Logistic Regression model.
- Weight appears to contain an extreme outlier.
- CancerVol may present two outliers, when Y_HighGradeCancer = 0.
- Via a priori knowledge, Age may be a valuable predictor of Prostate Cancer, so I will retain it in my onging analysis.

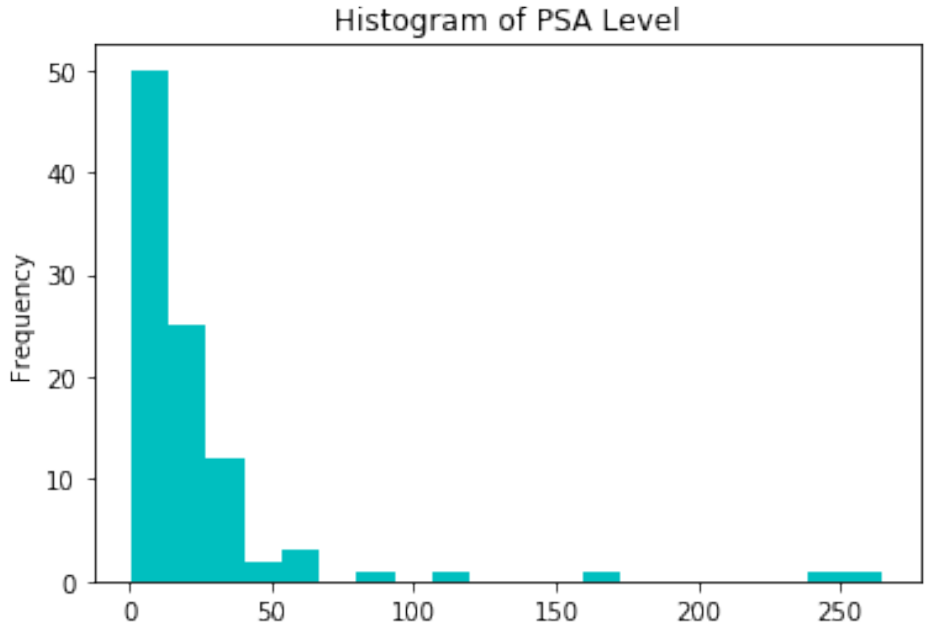
1.4 Distributions

1.4.1 PSALevel

[19]:
```
# box-whisker plot
df.PSALevel.plot(kind='box');
```



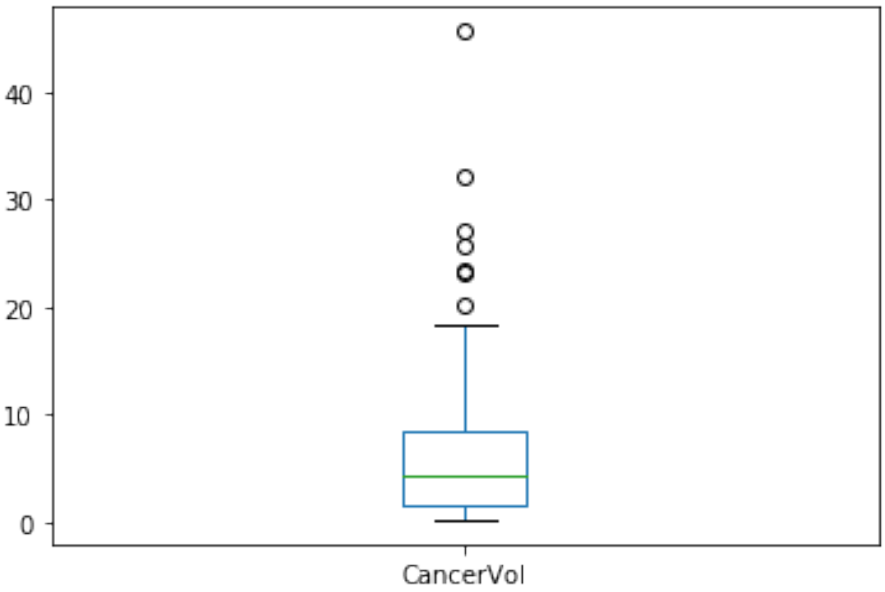
[20]:
```
# use hist to create histogram
df.PSALevel.plot(kind='hist', title='Histogram of PSA Level', color='c',␣
↪bins=20);
```



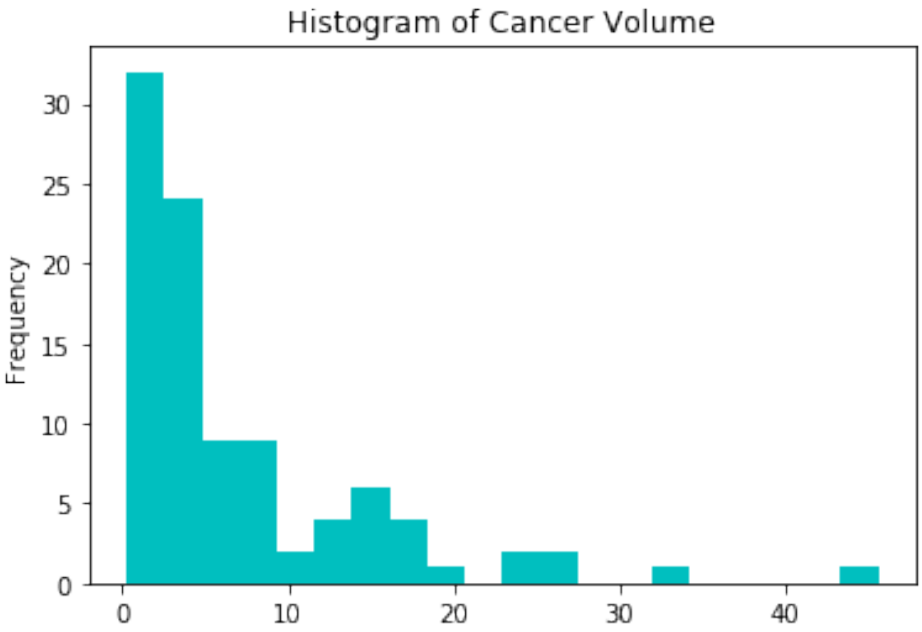
- PSALevel shows high positive skewness.

1.4.2 CancerVol

[21]:
```
df.CancerVol.plot(kind='box');
```



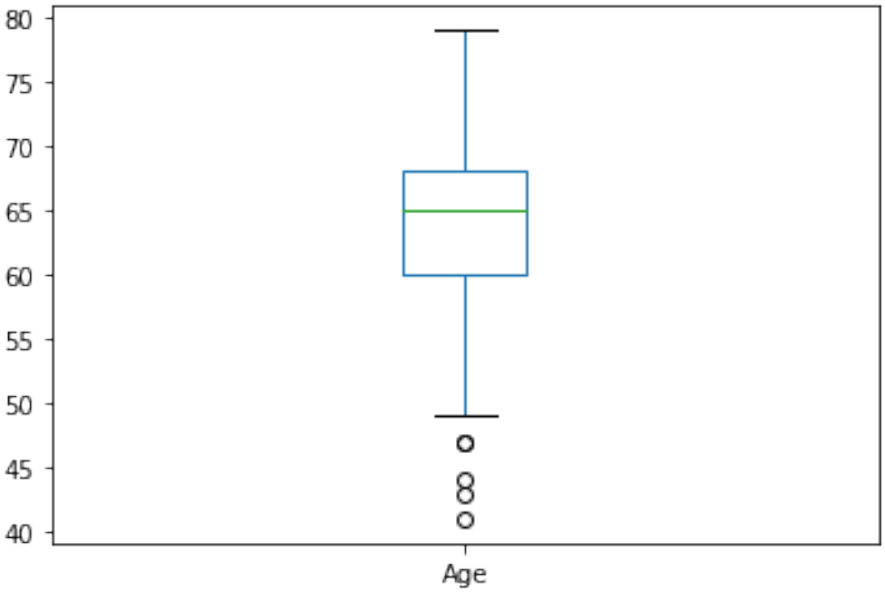
[22]:
```
df.CancerVol.plot(kind='hist', title='Histogram of Cancer Volume', color='c',␣
↪bins=20);
```



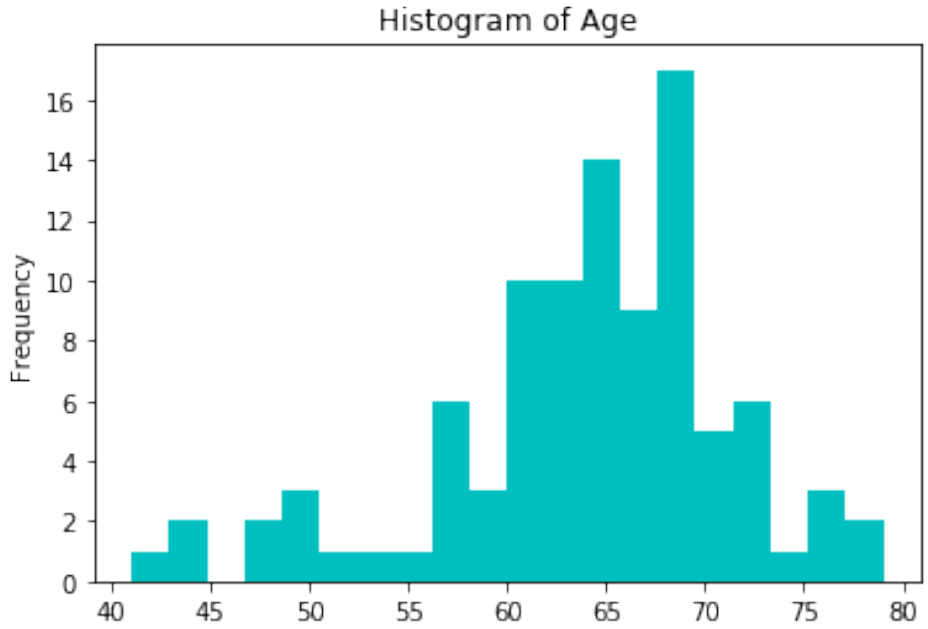
- CancerVol shows high positive skewness.

1.4.3 Age

[23]:
```
df.Age.plot(kind='box');
```

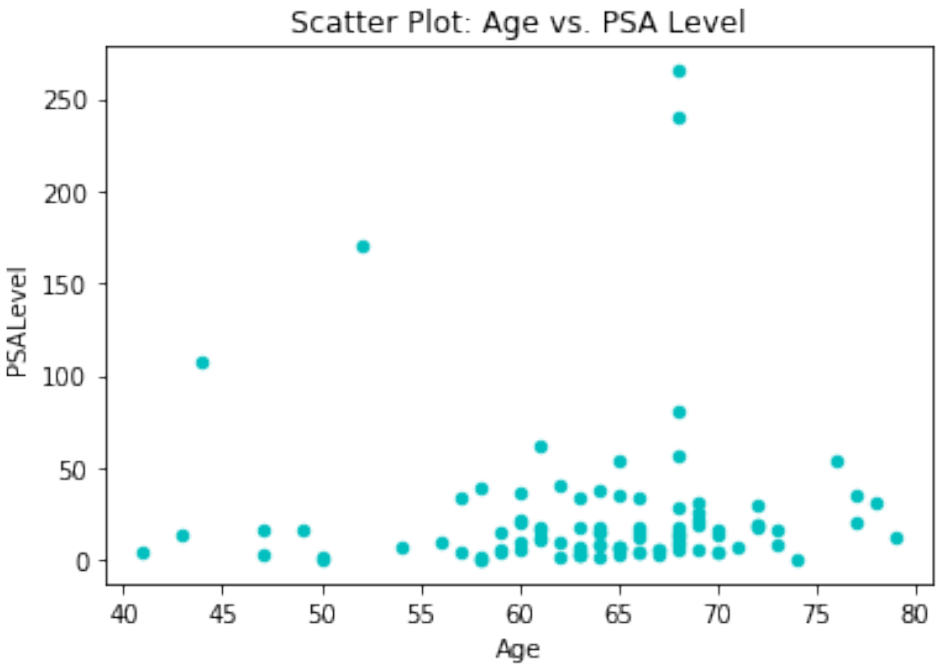


[24]:
```
df.Age.plot(kind='hist', title='Histogram of Age', color='c', bins=20);
```

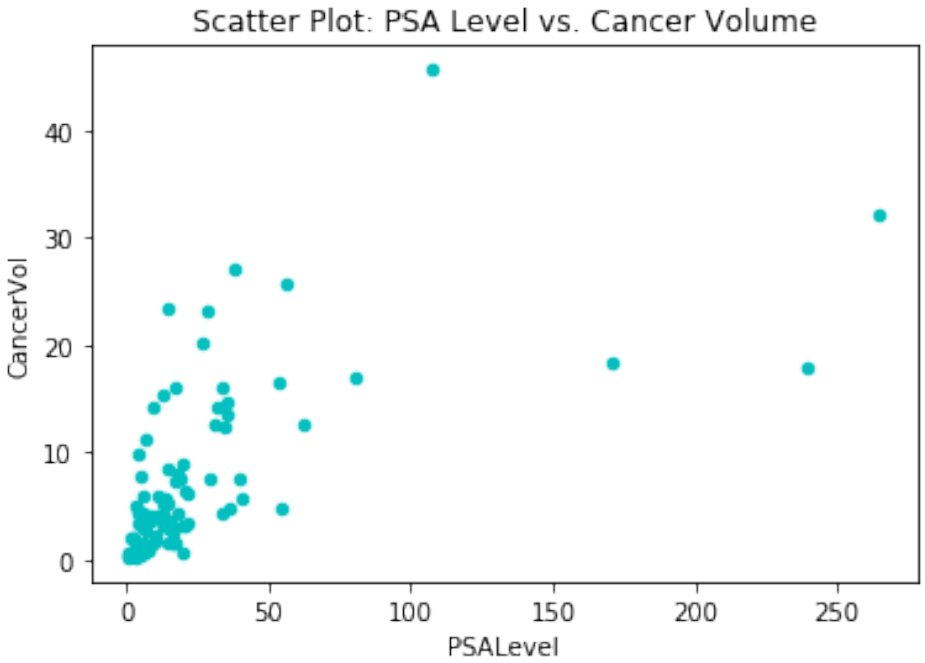


1.4.4 Bi-variate Interactions

[25]:
```
# use scatter plot for bi-variate distribution
df.plot.scatter(x='Age', y='PSALevel', color='c', title='Scatter Plot: Age vs.␣
↪PSA Level');
```

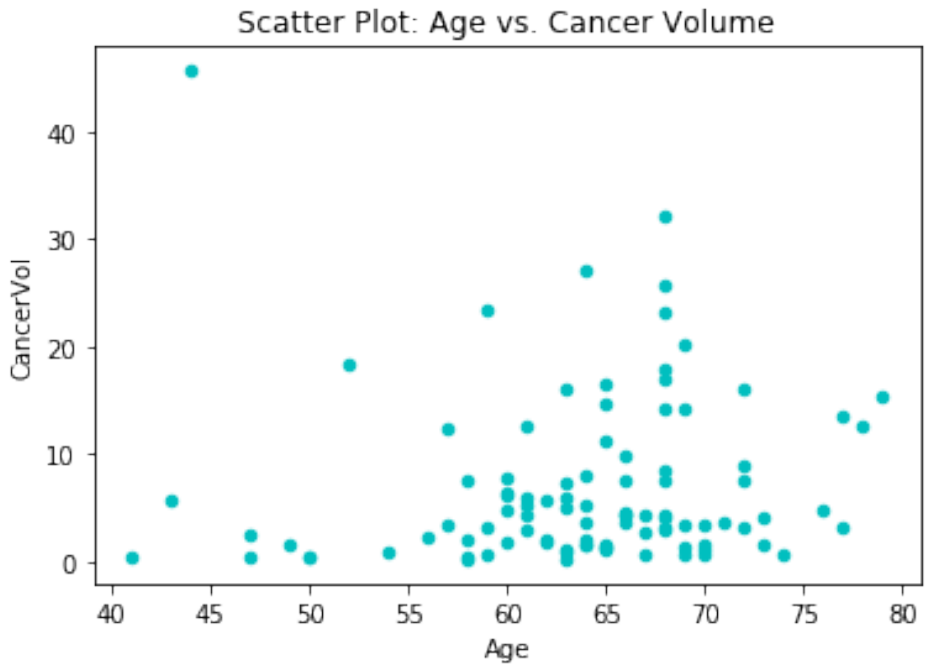


[26]:
```
df.plot.scatter(x='PSALevel', y='CancerVol', color='c', title='Scatter Plot: PSA Level vs. Cancer Volume');
```

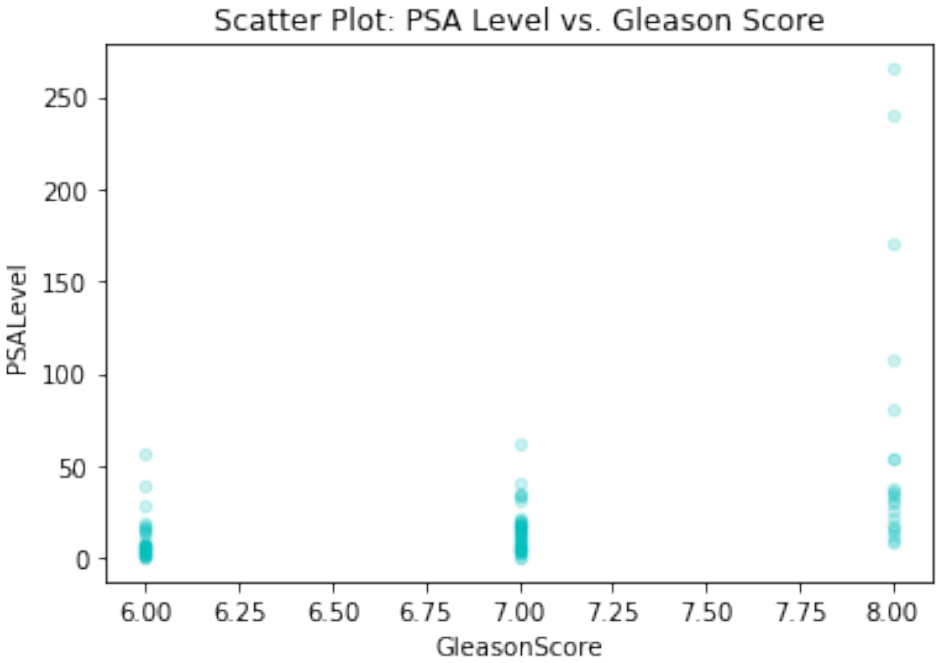


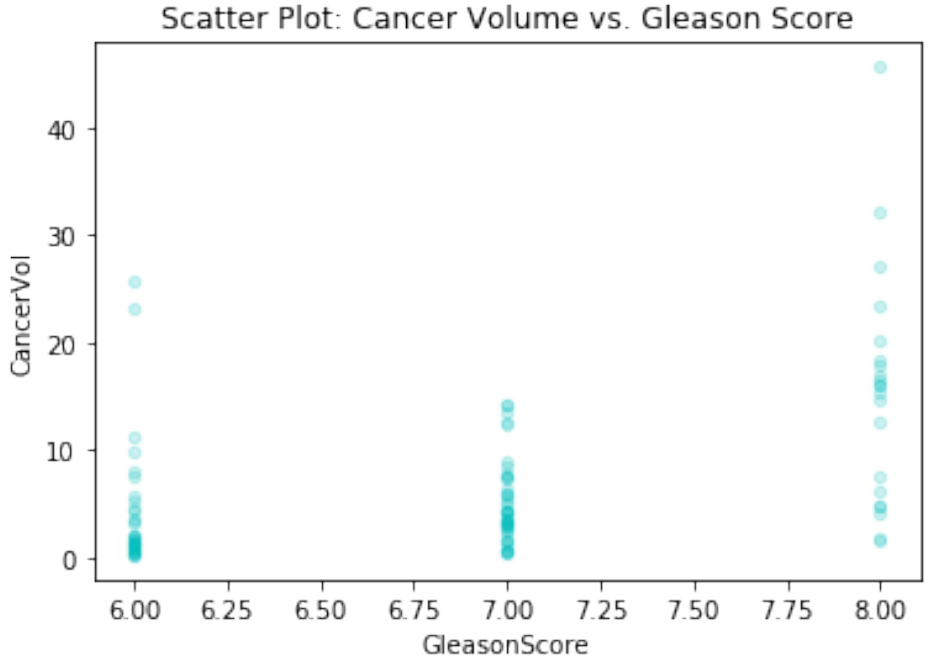
- PSA Level & Cancer Volume display a mild level of correlation

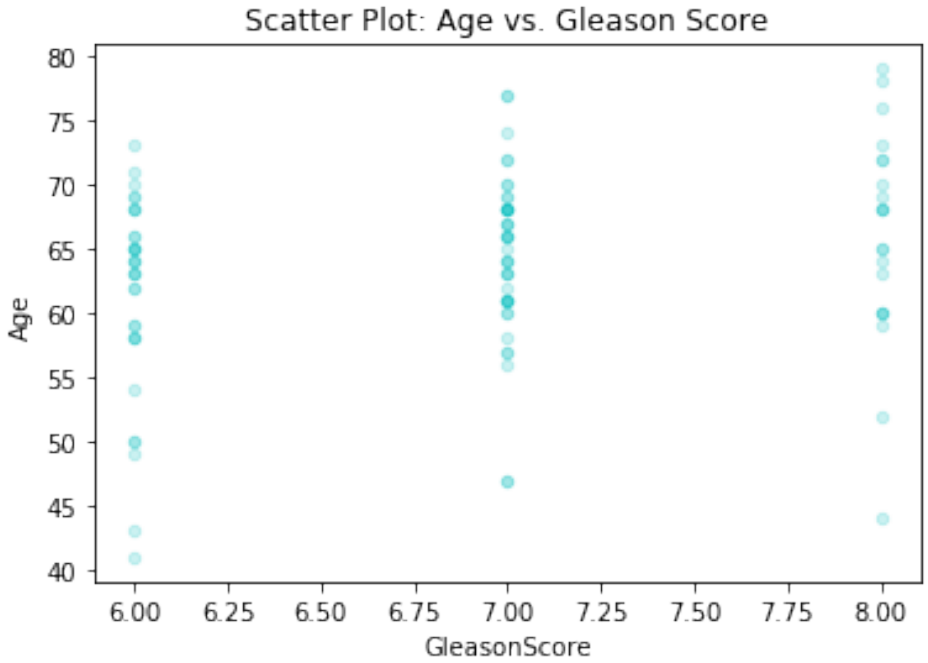
[27]:
```
df.plot.scatter(x='Age', y='CancerVol', color='c', title='Scatter Plot: Age vs.␣
↪Cancer Volume');
```



[28]:
```
# plot PSALevel, CancerVol, and Age against GleasonScore
df.plot.scatter(x='GleasonScore', y='PSALevel', color='c', alpha=0.2,␣
↪title='Scatter Plot: PSA Level vs. Gleason Score');
df.plot.scatter(x='GleasonScore', y='CancerVol', color='c', alpha=0.2,␣
↪title='Scatter Plot: Cancer Volume vs. Gleason Score');
df.plot.scatter(x='GleasonScore', y='Age', color='c', alpha=0.2, title='Scatter␣
↪Plot: Age vs. Gleason Score');
```


Scatter Plot: PSA Level vs. Gleason Score
PSALevel
250
200
150
100
50
0
6.00
6.25
6.50
6.75
7.00
7.25
7.50
7.75
8.00
GleasonScore


Scatter Plot: Cancer Volume vs. Gleason Score
CancerVol
40
30
20
10
0
6.00
6.25
6.50
6.75
7.00
7.25
7.50
7.75
8.00
GleasonScore



- High levels of PSALevel and/or CancerVol may suggest GleasonScore = 8.

[29]:
```
# calculate skewness for all columns in dataframe
for label, content in df.items():
    print(f'The skewness of {label} is: {round(content.skew(), 2)}')
```

```
The skewness of PSALevel is: 4.39
The skewness of CancerVol is: 2.18
The skewness of Weight is: 7.46
The skewness of Age is: -0.83
The skewness of BenignProstaticHyperplasia is: 0.98
The skewness of SeminalVesicleInvasion is: 1.4
The skewness of CapsularPenetration is: 2.13
The skewness of GleasonScore is: 0.2
The skewness of Y_HighGradeCancer is: 1.4
```

- PSALevel, CancerVol, and Weight are showing high skew values, and may require transformations in my analysis.

1.5 Working With Outliers

1.5.1 PSA Level

[30]:
```
# calculate IQR and find upper outlier fence

PSALevel_Q1 = np.percentile(df.PSALevel, 25)
PSALevel_Q2 = np.percentile(df.PSALevel, 50)
PSALevel_Q3 = np.percentile(df.PSALevel, 75)
PSALevel_IQR = PSALevel_Q3 - PSALevel_Q1  # inner quartile range
PSALevel_upper_fence = PSALevel_Q3 + 1.5 * PSALevel_IQR
print(f'The upper boundry for outliers in PSALevel is:␣
 ↪{round(PSALevel_upper_fence, 2)}')

# show relevant outlier data
df.loc[df.PSALevel >= PSALevel_upper_fence, :]
```

```
The upper boundry for outliers in PSALevel is: 44.86
```

[30]:

Obs	PSALevel	CancerVol	Weight	Age	BenignProstaticHyperplasia	SeminalVesicleInvasion	CapsularPenetration	GleasonScore	Y_HighGradeCancer
89	53.517	16.6099	112.168	65	0.0000	1	11.7048	8	1
90	54.055	4.7588	40.447	76	2.5600	1	2.2479	8	1
91	56.261	25.7903	60.340	68	0.0000	0	0.0000	6	0
92	62.178	12.5535	39.646	61	3.8574	1	0.0000	7	1
93	80.640	16.9455	48.424	68	0.0000	1	3.7434	8	1
94	107.770	45.6042	49.402	44	0.0000	1	8.7583	8	1
95	170.716	18.3568	29.964	52	0.0000	1	11.7048	8	1
96	239.847	17.8143	43.380	68	4.7588	1	4.7588	8	1
97	265.072	32.1367	52.985	68	1.5527	1	18.1741	8	1

```
92                    0
93                    1
94                    1
95                    1
96                    1
97                    1
```

1.5.2 Cancer Volume

[31]:
```
# calculate IQR and find upper outlier fences (mild and extreme)
# consider only where Y_HighGradeCancer == 0

CancerVol_Q1 = np.percentile(df.loc[df.Y_HighGradeCancer == 0, :]['CancerVol'],␣
↪25)
CancerVol_Q2 = np.percentile(df.loc[df.Y_HighGradeCancer == 0, :]['CancerVol'],␣
↪50)
CancerVol_Q3 = np.percentile(df.loc[df.Y_HighGradeCancer == 0, :]['CancerVol'],␣
↪75)
CancerVol_IQR = CancerVol_Q3 - CancerVol_Q1  # inner quartile range
CancerVol_mild_upper_fence = CancerVol_Q3 + 1.5 * CancerVol_IQR
CancerVol_extreme_upper_fence = CancerVol_Q3 + 2.0 * CancerVol_IQR
print(f'The upper boundry for MILD OUTLIERS in CancerVol is:␣
↪{round(CancerVol_mild_upper_fence, 2)}')
print(f'The upper boundry for EXTREME OUTLIERS in CancerVol is:␣
↪{round(CancerVol_extreme_upper_fence, 2)}')

df.loc[(df.CancerVol > CancerVol_extreme_upper_fence) & (df.Y_HighGradeCancer␣
↪== 0) ]
```

```
The upper boundry for MILD OUTLIERS in CancerVol is: 12.55
The upper boundry for EXTREME OUTLIERS in CancerVol is: 14.77
```

[31]:
```
     PSALevel  CancerVol  Weight  Age  BenignProstaticHyperplasia  \
Obs
76     28.219    23.1039   26.05   68                      0.9512
91     56.261    25.7903   60.34   68                      0.0000

     SeminalVesicleInvasion  CapsularPenetration  GleasonScore  \
Obs
76                        1              11.2459             6
91                        0               0.0000             6

     Y_HighGradeCancer
Obs
76                   0
91                   0
```

1.5.3 Age

[32]:
```
# calculate IQR and find lower outlier fence

Age_Q1 = np.percentile(df.Age, 25)
Age_Q2 = np.percentile(df.Age, 50)
Age_Q3 = np.percentile(df.Age, 75)
Age_IQR = Age_Q3 - Age_Q1 # inner quartile range
Age_lower_fence = Age_Q1 -  1.5 * Age_IQR
print(f'The lower boundry for outliers in Age is: {Age_lower_fence}')

df.loc[df.Age < Age_lower_fence]
```

```
The lower boundry for outliers in Age is: 48.0
```

[32]:

Obs	PSALevel	CancerVol	Weight	Age	BenignProstaticHyperplasia	SeminalVesicleInvasion	CapsularPenetration	GleasonScore	Y_HighGradeCancer
9	2.858	0.4584	34.467	47	0.0	0	0.0000	7	0
19	4.759	0.5712	26.311	41	0.0	0	0.0000	6	0
49	13.330	5.7546	33.115	43	0.0	0	0.0000	6	0
57	16.281	2.6379	17.637	47	0.0	0	1.6487	7	0
94	107.770	45.6042	49.402	44	0.0	1	8.7583	8	1

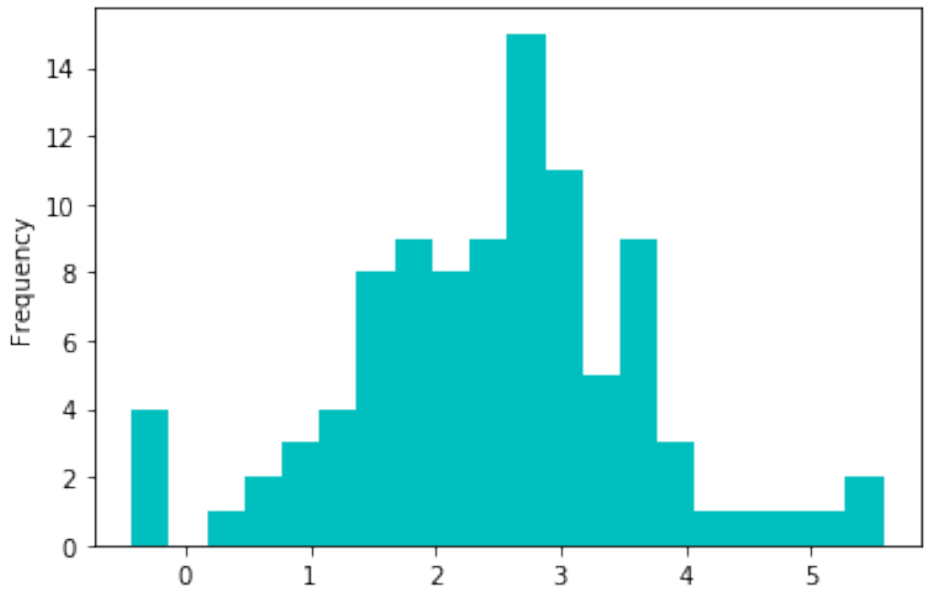
1.6 Transformations

- This section is for investigation/analysis purposes only. I may or may not include transformations in the finalized processed dataset.
- Considering only PSALevel, CancerVol, and Weight at this time.

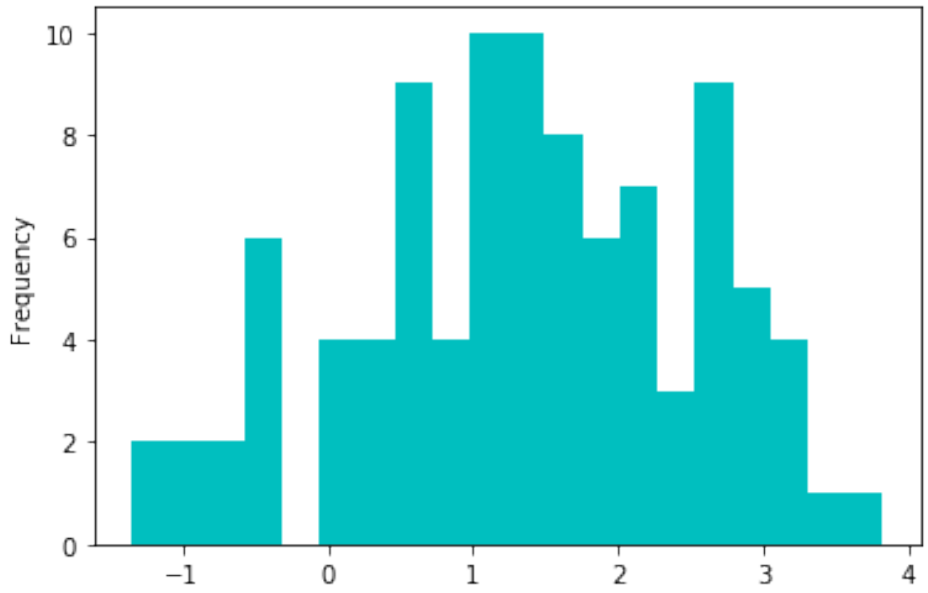
[33]:
```
# try log transformations to reduce skewness
log_PSALevel = np.log(df.PSALevel)
log_CancerVol = np.log(df.CancerVol)
```

```
log_Weight = np.log(df.Weight)
```

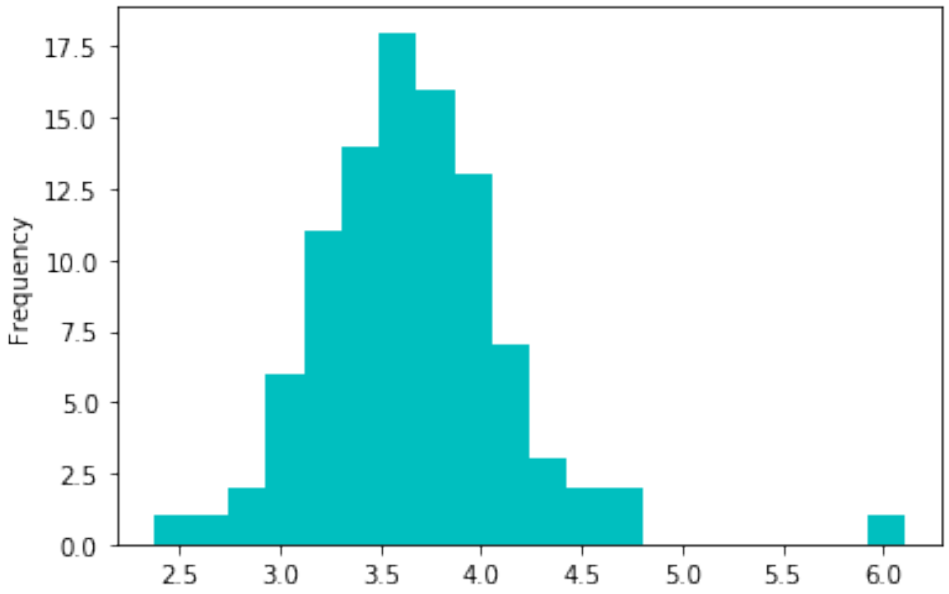
[34]:
```
# histogram of log PSALevel
log_PSALevel.plot(kind='hist', color='c', bins=20);
```



[35]:
```
# histogram of log CancerVol
log_CancerVol.plot(kind='hist', color='c', bins=20);
```



[36]:
```
# histogram of log Weight
log_Weight.plot(kind='hist', color='c', bins=20);
```



[37]:
```
# print original skew values
print(f'Original PSALevel skewness: {round(df.PSALevel.skew(), 2)}')
print(f'Original CancerVol skewness: {round(df.CancerVol.skew(), 2)}')
print(f'Original Weight skewness: {round(df.Weight.skew(), 2)}')

# print transformed skew values
print(f'Log Transformed PSALevel skewness: {round(log_PSALevel.skew(), 2)}')
print(f'Log Transformed CancerVol skewness: {round(log_CancerVol.skew(), 2)}')
print(f'Log Transformwed Weight skewness: {round(log_Weight.skew(), 2)}')
```

```
Original PSALevel skewness: 4.39
Original CancerVol skewness: 2.18
Original Weight skewness: 7.46
Log Transformed PSALevel skewness: 0.0
Log Transformed CancerVol skewness: -0.25
Log Transformwed Weight skewness: 1.21
```

- Log transformations have dramatically improved PSALevel and CancerVol skewness.
- I will include the transformed fields within final processed dataset.

1.7 Drop, Modify, and Reorder Columns

GleasonScore can now be removed from the dataset, as it will not be considered as a predictor of Y_HighGradeCancer. Let's also move the response variable to the 1st column, for ease of use durring model building.

[38]:
```
# remove GleasonScore from dataset and assign to new "df_trimmed" dataframe
df_trimmed = df.drop(columns=['GleasonScore'], axis=1)
```

[39]:
```
# reorder columns
cols = [col for col in df_trimmed.columns if col != 'Y_HighGradeCancer']
cols = ['Y_HighGradeCancer'] + cols
df_trimmed = df_trimmed[cols]
df_trimmed.head()
```

[39]:

Obs	Y_HighGradeCancer	PSALevel	CancerVol	Weight	Age	BenignProstaticHyperplasia	SeminalVesicleInvasion	CapsularPenetration
1	0	0.651	0.5599	15.959	50	0.0	0	0.0
2	0	0.852	0.3716	27.660	58	0.0	0	0.0
3	0	0.852	0.6005	14.732	74	0.0	0	0.0
4	0	0.852	0.3012	26.576	58	0.0	0	0.0
5	0	1.448	2.1170	30.877	62	0.0	0	0.0

[40]:
```
df_trimmed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 97 entries, 1 to 97
Data columns (total 8 columns):
Y_HighGradeCancer             97 non-null int32
PSALevel                      97 non-null float64
CancerVol                     97 non-null float64
Weight                        97 non-null float64
Age                           97 non-null int64
BenignProstaticHyperplasia    97 non-null float64
SeminalVesicleInvasion        97 non-null int64
CapsularPenetration           97 non-null float64
dtypes: float64(5), int32(1), int64(2)
memory usage: 6.4 KB
```

1.8 Standardize DataFrame

- Variable transformation and standardization is an important technique used to create robust models using logistic regression.

[41]:
```
# import and create instance of standardization class from sklearn module
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

[42]:
```
# select columns which need to be standardized
# do not inclue Y_HighGradeCancer or SeminalVesicleInvasion (categorical␣
 ↪variables)
cols = [col for col in df_trimmed.columns if col not in ['Y_HighGradeCancer',␣
 ↪'SeminalVesicleInvasion']]
```

[43]:
```
# make a copy of trimmed dataframe
df_stand = df_trimmed.copy()
```

[44]:
```
# apply log transformations to both PSALevel and CancerVol
df_stand['PSALevel'] = np.log(df_stand.PSALevel)
df_stand['CancerVol'] = np.log(df_stand.CancerVol)
df_stand['Weight'] = np.log(df_stand.Weight)
```

[45]:
```
# standardize the dataframe
df_stand[cols] = scaler.fit_transform(df_stand[cols])
```

[46]:
```
# the standardized features should now have mean=0 and sd=1
df_stand.describe()
```

[46]:

	Y_HighGradeCancer	PSALevel	CancerVol	Weight
count	97.000000	9.700000e+01	9.700000e+01	9.700000e+01
mean	0.216495	7.783007e-17	-2.403576e-16	-5.013172e-16
std	0.413995	1.005195e+00	1.005195e+00	1.005195e+00
min	0.000000	-2.533700e+00	-2.302583e+00	-2.595287e+00
25%	0.000000	-6.522705e-01	-7.161288e-01	-5.518528e-01
50%	0.000000	9.701907e-02	8.555117e-02	-6.629801e-02
75%	0.000000	5.065387e-01	6.655015e-01	4.596790e-01
max	1.000000	2.702227e+00	2.106830e+00	4.971231e+00

	Age	BenignProstaticHyperplasia	SeminalVesicleInvasion
count	9.700000e+01	9.700000e+01	97.000000
mean	3.433679e-16	6.409535e-17	0.216495
std	1.005195e+00	1.005195e+00	0.413995
min	-3.087227e+00	-8.405624e-01	0.000000
25%	-5.219612e-01	-8.405624e-01	0.000000
50%	1.531086e-01	-3.929102e-01	0.000000
75%	5.581506e-01	7.375452e-01	0.000000
max	2.043304e+00	2.567782e+00	1.000000

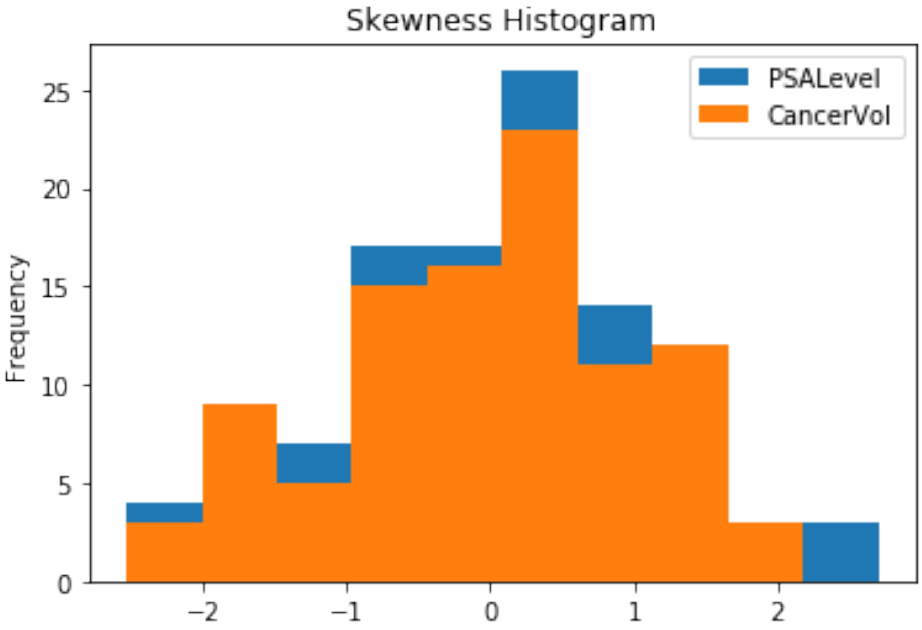
```
       CapsularPenetration
count         9.700000e+01
mean          1.281907e-16
std           1.005195e+00
min          -5.965729e-01
25%          -5.965729e-01
50%          -4.771981e-01
75%           2.680906e-01
max           4.232114e+00
```

Sanity check… As a final measure, let's examine skew values of the final trimmed and transformed dataset:

[47]:
```
# print skewness for all columns in trimmed and transformed dataset
for label, content in df_stand.items():
    if label != 'Y_HighGradeCancer':
        print(f'The skewness of {label} is: {round(content.skew(), 2)}')
```

```
The skewness of PSALevel is: 0.0
The skewness of CancerVol is: -0.25
The skewness of Weight is: 1.21
The skewness of Age is: -0.83
The skewness of BenignProstaticHyperplasia is: 0.98
The skewness of SeminalVesicleInvasion is: 1.4
The skewness of CapsularPenetration is: 2.13
```

[48]:
```
# visualize skewness of a few impactful feautures
df_stand[['PSALevel', 'CancerVol']].plot(kind='hist', title='Skewness␣
 ↪Histogram');
```



1.9 Save Processed Data

[49]:
```
# define paths
processed_data_path = os.path.join(os.path.pardir, 'data', 'processed')
write_data_path = os.path.join(processed_data_path, 'APPENC05.txt')
```

[50]:
```
# save data
df_stand.to_csv(write_data_path)
```

3.0-jo-building-predictive-model

December 3, 2020

1 Building Predictive Models

[1]:
```
import os
import pandas as pd
import numpy as np
import sklearn
```

1.1 Import Data

Train and test data were randomly split within R, using 0.80 ratio. The two dataframes were written to independent csv files, and will be brought into the Python notebook now.

[2]:
```
# set path to processed train/test data
processed_data_path = os.path.join(os.path.pardir, 'data', 'processed')
train_file_path = os.path.join(processed_data_path, 'train.txt')
test_file_path = os.path.join(processed_data_path, 'test.txt')
```

[3]:
```
df_train = pd.read_csv(train_file_path, index_col='Obs')
df_test = pd.read_csv(test_file_path, index_col='Obs')
```

[4]:
```
print('Train data:')
df_train.info()
print('\n')
print('Test data:')
df_test.info()
```

```
Train data:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 76 entries, 1 to 97
Data columns (total 9 columns):
Unnamed: 0                    76 non-null int64
Y_HighGradeCancer             76 non-null int64
PSALevel                      76 non-null float64
CancerVol                     76 non-null float64
Weight                        76 non-null float64
Age                           76 non-null float64
BenignProstaticHyperplasia    76 non-null float64
SeminalVesicleInvasion        76 non-null int64
```

```
CapsularPenetration           76 non-null float64
dtypes: float64(6), int64(3)
memory usage: 5.9 KB


Test data:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21 entries, 5 to 95
Data columns (total 9 columns):
Unnamed: 0                    21 non-null int64
Y_HighGradeCancer             21 non-null int64
PSALevel                      21 non-null float64
CancerVol                     21 non-null float64
Weight                        21 non-null float64
Age                           21 non-null float64
BenignProstaticHyperplasia    21 non-null float64
SeminalVesicleInvasion        21 non-null int64
CapsularPenetration           21 non-null float64
dtypes: float64(6), int64(3)
memory usage: 1.6 KB
```

[5]:
```
df_train.columns
```

[5]:
```
Index(['Unnamed: 0', 'Y_HighGradeCancer', 'PSALevel', 'CancerVol', 'Weight',
       'Age', 'BenignProstaticHyperplasia', 'SeminalVesicleInvasion',
       'CapsularPenetration'],
      dtype='object')
```

It looks like R appended an additional "Unnamed: 0" column, most likely related to indexing. I will remove that now.

[6]:
```
# drop the redudent columns (R auto-created an index column of its own); can be␣
 ↪seen in info() cell above
df_train = df_train.drop(columns='Unnamed: 0')
df_test = df_test.drop(columns='Unnamed: 0')
```

[7]:
```
# examine train set
df_train
```

[7]:

Obs	Y_HighGradeCancer	PSALevel	CancerVol	Weight	Age	\
1	0	-2.533700	-1.645747	-1.785921	-1.872101	
2	0	-2.299250	-1.995368	-0.673281	-0.791989	
3	0	-2.299250	-1.586043	-1.947772	1.368234	
4	0	-2.299250	-2.174506	-0.754163	-0.791989	
6	0	-1.488689	-2.046685	-0.855308	-1.872101	
..	…	…	…	…	…	
92	0	1.438825	1.006641	0.055045	-0.386947	

```
93                    1  1.665361   1.262501  0.459679  0.558151
94                    1  1.918045   2.106830  0.500132 -2.682185
96                    1  2.615096   1.305144  0.237142  0.558151
97                    1  2.702227   1.808328  0.641786  0.558151

     BenignProstaticHyperplasia  SeminalVesicleInvasion  CapsularPenetration
Obs
1                      -0.840562                       0            -0.596573
2                      -0.840562                       0            -0.596573
3                      -0.840562                       0            -0.596573
4                      -0.840562                       0            -0.596573
6                      -0.840562                       0            -0.596573
..                           …                       …                   …
92                      0.438624                       1            -0.596573
93                     -0.840562                       1             0.398013
94                     -0.840562                       1             1.730425
96                      0.737545                       1             0.667795
97                     -0.325658                       1             4.232114

[76 rows x 8 columns]
```

[8]:
```
# examine test set
df_test.head()
```

[8]:
```
     Y_HighGradeCancer  PSALevel  CancerVol    Weight       Age  \
Obs
5                    0 -1.837148  -0.511447 -0.450690 -0.251933
8                    0 -1.418947  -0.562625 -0.228166 -0.791989
14                   0 -0.983519   0.111131 -1.320605  0.423137
17                   0 -0.878912  -1.509353 -0.268658  0.828178
23                   0 -0.678455  -1.611706 -0.551853 -0.656975

     BenignProstaticHyperplasia  SeminalVesicleInvasion  CapsularPenetration
Obs
5                      -0.840562                       0            -0.596573
8                       0.706307                       0            -0.596573
14                     -0.840562                       0            -0.596573
17                      0.305380                       0            -0.450762
23                     -0.691566                       0            -0.596573
```

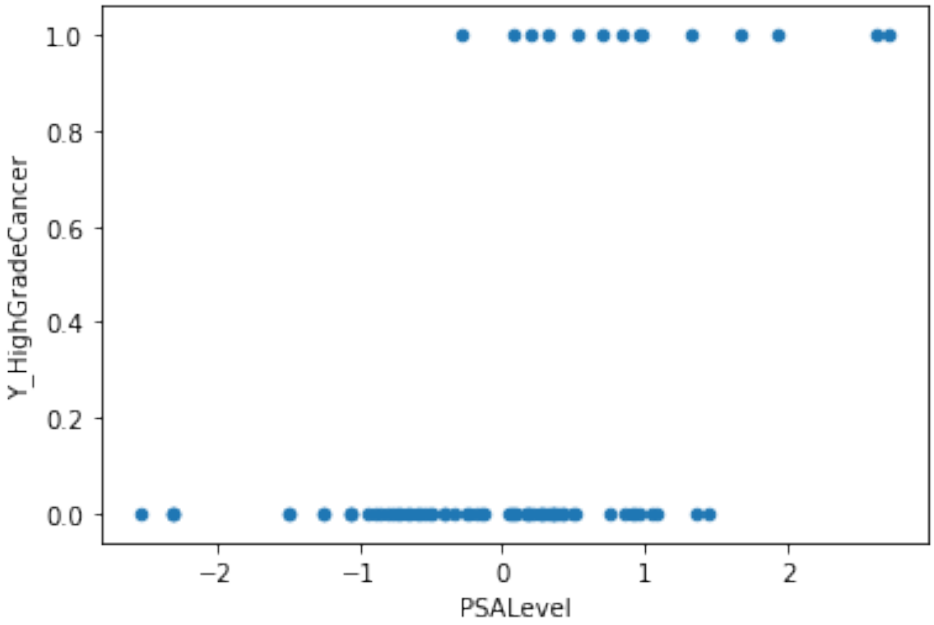
[9]:
```
# create a list which captures fields to ommit from model
skip = ['Y_HighGradeCancer'
        , 'Age'
        , 'Weight'
        , 'BenignProstaticHyperplasia'
        , 'SeminalVesicleInvasion'
        , 'CapsularPenetration'
```

```
        ]
cols_model = [col for col in df_train.columns if col not in skip]
cols_model
```

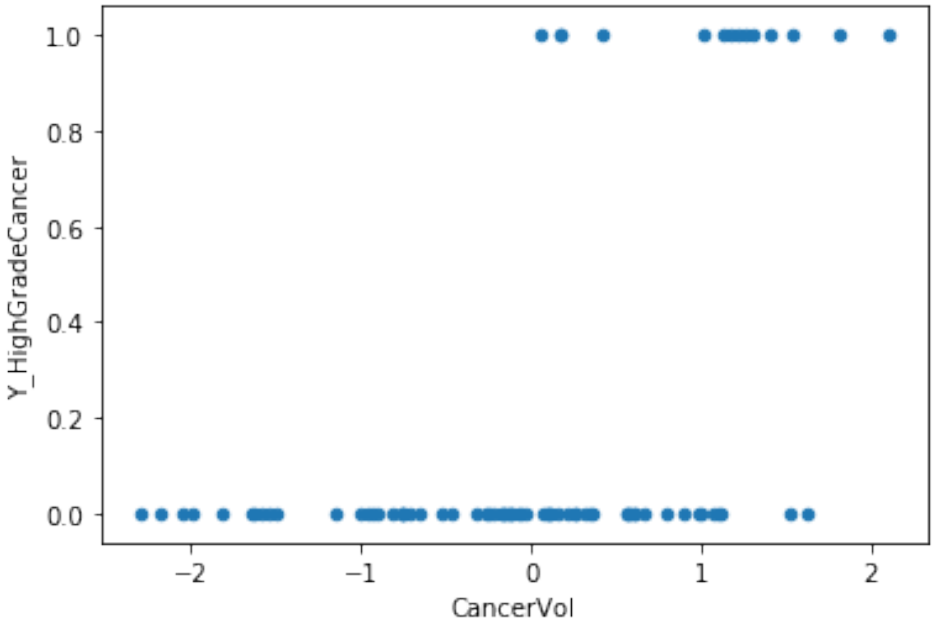
[9]:
```
['PSALevel', 'CancerVol']
```

1.2 Visuals

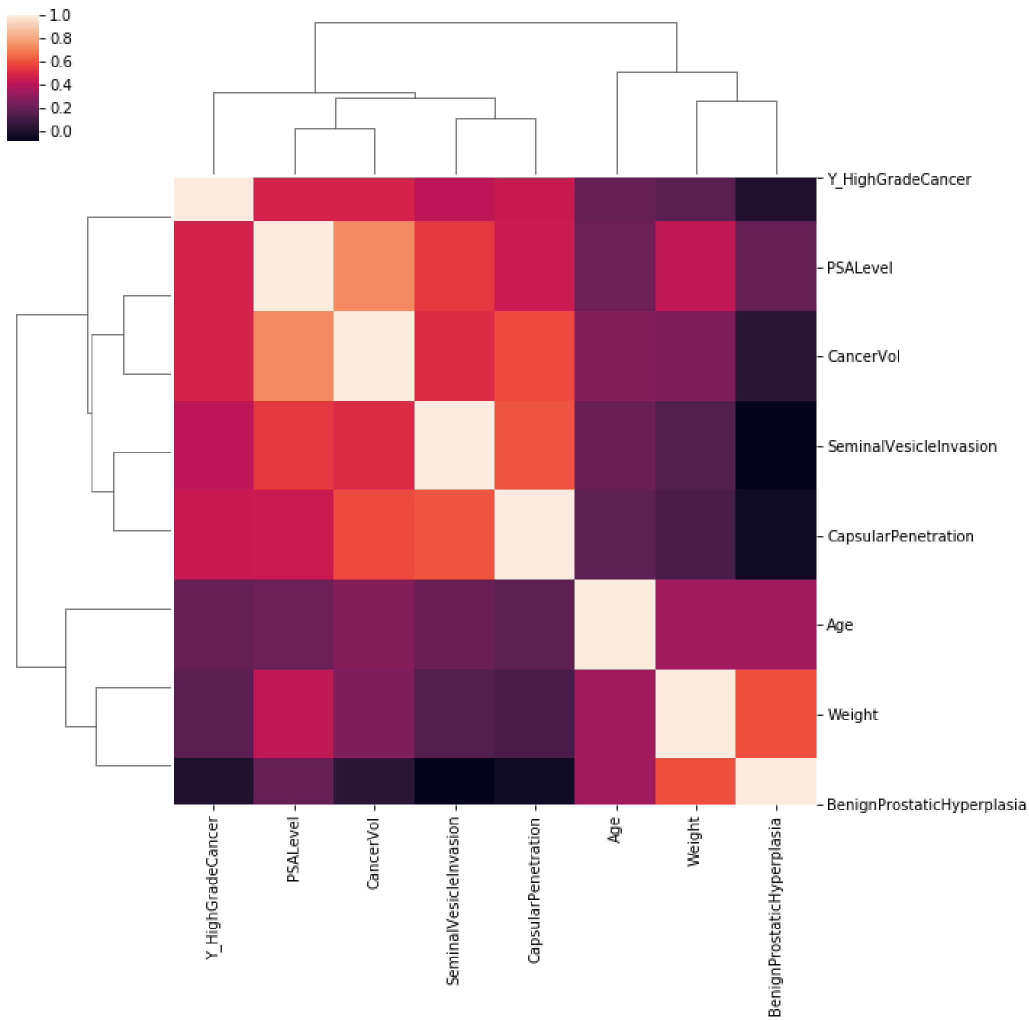
[11]:
```
df_train.plot.scatter(x='PSALevel', y='Y_HighGradeCancer');
```



[12]:
```
df_train.plot.scatter(x='CancerVol', y='Y_HighGradeCancer');
```



[13]:
```
# let's import searborn to help visualize the train data correlation matrix
import seaborn as sns

sns.clustermap(df_train.corr());
```



[14]:
```
df_train.corr()
```

[14]:

	Y_HighGradeCancer	PSALevel	CancerVol	Weight	\
Y_HighGradeCancer	1.000000	0.488609	0.492580	0.173835	
PSALevel	0.488609	1.000000	0.737585	0.427753	
CancerVol	0.492580	0.737585	1.000000	0.264202	
Weight	0.173835	0.427753	0.264202	1.000000	
Age	0.196961	0.217748	0.274467	0.350116	
BenignProstaticHyperplasia	0.000539	0.199778	0.044290	0.599560	
SeminalVesicleInvasion	0.420664	0.550701	0.515015	0.148291	
CapsularPenetration	0.452185	0.457590	0.593430	0.128845	

Age BenignProstaticHyperplasia \

```
Y_HighGradeCancer           0.196961                  0.000539
PSALevel                    0.217748                  0.199778
CancerVol                   0.274467                  0.044290
Weight                      0.350116                  0.599560
Age                         1.000000                  0.344029
BenignProstaticHyperplasia  0.344029                  1.000000
SeminalVesicleInvasion      0.209401                 -0.082420
CapsularPenetration         0.183055                 -0.035408

                            SeminalVesicleInvasion  CapsularPenetration
Y_HighGradeCancer                         0.420664             0.452185
PSALevel                                  0.550701             0.457590
CancerVol                                 0.515015             0.593430
Weight                                    0.148291             0.128845
Age                                       0.209401             0.183055
BenignProstaticHyperplasia               -0.082420            -0.035408
SeminalVesicleInvasion                    1.000000             0.611239
CapsularPenetration                       0.611239             1.000000
```

[15]:
```
# correlation of final two predictors
df_train[['PSALevel', 'CancerVol']].corr()
```

[15]:
```
            PSALevel  CancerVol
PSALevel    1.000000   0.737585
CancerVol   0.737585   1.000000
```

1.3 Data Preperation

Because R has already prepared the training and test sets, I will manually assign the split data to appropriate variables now.

[16]:
```
# train-test split
X_train = df_train.loc[:, cols_model]
y_train = df_train['Y_HighGradeCancer']
X_test = df_test.loc[:, cols_model]
y_test = df_test['Y_HighGradeCancer']
```

[17]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(76, 2) (76,)
(21, 2) (21,)
```

[18]:
```
# average survival in train and test sets
print(f'Mean y in train set: {round(np.mean(y_train), 3)}')
print(f'Mean y in test set: {round(np.mean(y_test), 3)}')
```

```
Mean y in train set: 0.184
Mean y in test set: 0.333
```

1.4 Baseline Model

Developing a basline model: - Here, I will feed the dummy model training data, and sklearn will determine the most frequent classification within the Y_HighGradeCancer field (via prior analysis we know this to value to be 0). Because Y_HighGradeCancer = 0 most freqently, the model will be designed to predict 0 on every single observation. - After the design of the baseline model, I will implement it on both the training and testing data, and calculate accuracy scores and confusion matrixes for good measure. - Subsequent model fittings can therefore be compared to the baseline model.

[19]:
```
# import function
from sklearn.dummy import DummyClassifier
```

[20]:
```
# create model
# because mean y in train = 0.184 (shown above), this "most frequent" model␣
↪will predict y=0 for all test observations
model_dummy = DummyClassifier(strategy='most_frequent', random_state=0)
```

[21]:
```
# train model
model_dummy.fit(X_train, y_train)
```

[21]:
```
DummyClassifier(constant=None, random_state=0, strategy='most_frequent')
```

[22]:
```
# run dummy_model with training data
print(f'Score for baseline model (TRAINING): {round(model_dummy.score(X_train,␣
↪y_train), 2)}')

# run dummy_model with testing data
print(f'Score for baseline model (TESTING): {round(model_dummy.score(X_test,␣
↪y_test), 2)}')
```

```
Score for baseline model (TRAINING): 0.82
Score for baseline model (TESTING): 0.67
```

[23]:
```
# performance metrics
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,␣
↪recall_score
```

[24]:
```
# training confusion matrix
print(f'Confusion matrix for baseline model (TRAINING): \n␣
↪{confusion_matrix(y_train, model_dummy.predict(X_train))} \n')

# testing confusion matrix
print(f'Confusion matrix for baseline model (TESTING): \n␣
↪{confusion_matrix(y_test, model_dummy.predict(X_test))}')
```

```
Confusion matrix for baseline model (TRAINING):
 [[62  0]
 [14  0]]
```

```
Confusion matrix for baseline model (TESTING):
 [[14  0]
 [ 7  0]]
```

1.5 Statsmodels Library

1.5.1 Full Logistics Model

[25]:
```
import statsmodels.api as sm
```

[26]:
```
X_model = sm.add_constant(X_train)
model = sm.Logit(y_train, X_model)
```

```
C:\Users\jaosi\Anaconda3\envs\datSci\lib\site-
packages\numpy\core\fromnumeric.py:2389: FutureWarning: Method .ptp is
deprecated and will be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

[27]:
```
results = model.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.293604
         Iterations 8
```

[28]:
```
### full model statistical output
print(results.summary2(alpha=0.05))
```

```
                          Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.385
Dependent Variable: Y_HighGradeCancer AIC:             50.6278
Date:               2020-12-03 15:26  BIC:             57.6200
No. Observations:   76                Log-Likelihood:  -22.314
Df Model:           2                 LL-Null:         -36.307
Df Residuals:       73                LLR p-value:     8.3761e-07
Converged:          1.0000            Scale:           1.0000
No. Iterations:     8.0000
-----------------------------------------------------------------
               Coef.   Std.Err.     z     P>|z|    [0.025   0.975]
-----------------------------------------------------------------
const         -2.6867    0.6186  -4.3429  0.0000  -3.8992  -1.4742
PSALevel       1.0577    0.6198   1.7067  0.0879  -0.1570   2.2725
CancerVol      1.5502    0.6859   2.2599  0.0238   0.2058   2.8945
=================================================================
```

[29]:
```
PSALevel_list = X_train['PSALevel'].tolist()
CancerVol_list = X_train['CancerVol'].tolist()
```

```
Y_HighGradeCancer_list = y_train.tolist()
```

[30]:
```
type(np.arange(1, 2, 0.5))
```

[30]:
```
numpy.ndarray
```

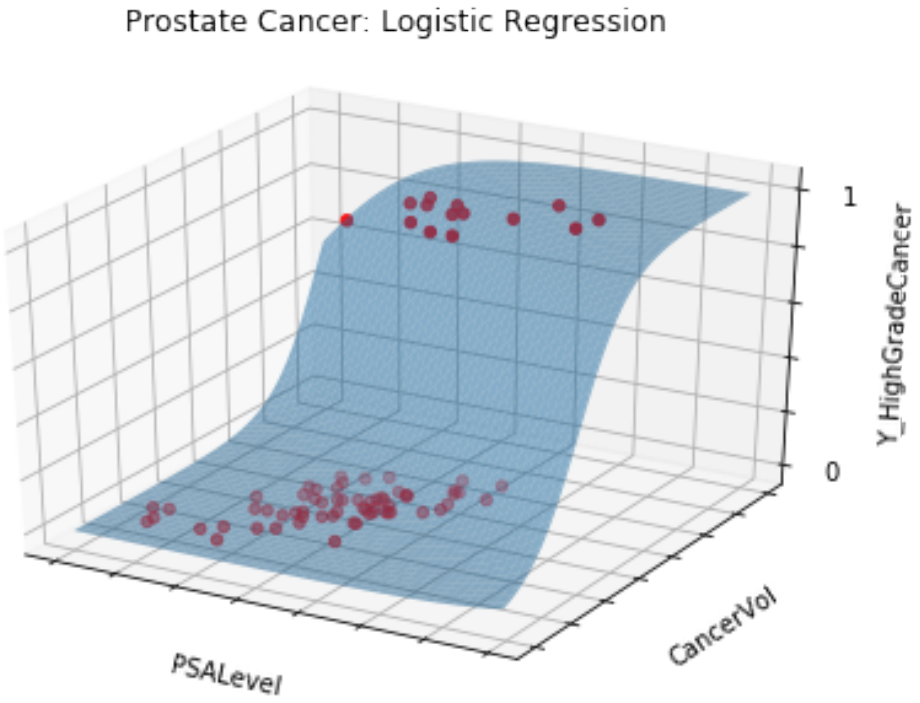
1.6 Advanced Visualizations Using Matplotlib

[31]:
```
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

1.6.1 Logistic Regression Plot

[32]:
```
%matplotlib inline

# bring in and store the coefficients of the fitted model
const_coeff, x1_coeff, x2_coeff = results.params

# define a sigmoid function of 2 variables
def sigmoid(x1, x2):
    func = 1.0 / (1.0 + math.exp(-(const_coeff + x1_coeff*x1 + x2_coeff*x2)))
    return func

# design plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
plt.tight_layout()
x = y = np.arange(-3.0, 4.0, 0.05)
X, Y = np.meshgrid(x, y)
zs = np.array([sigmoid(x,y) for x,y in zip(np.ravel(X), np.ravel(Y))])
Z = zs.reshape(X.shape)

# draw plots
ax.plot_surface(X, Y, Z, alpha=0.5)
ax.scatter(PSALevel_list, CancerVol_list, Y_HighGradeCancer_list, c='red',␣
 ↪marker='o')

# modify axes and labels
ax.set_xticklabels([])
ax.set_yticklabels([])
ax.set_zticklabels([0, 0, '', '', '', '', 1])
ax.set_xlabel('PSALevel')
ax.set_ylabel('CancerVol')
ax.set_zlabel('Y_HighGradeCancer')
ax.set_title('Prostate Cancer: Logistic Regression')
```

[32]: Text(0.5, 0.92, 'Prostate Cancer: Logistic Regression')



7.2 R

```
# set working directory
setwd("C:/Users/jaosi/Desktop/DS-Projects/graduate-project/prostate-cancer")

# load dataset
APPENC05 <- read.csv("./data/processed/APPENC05.txt")
mydata <- APPENC05
summary(mydata)
```

```
##       Obs     Y_HighGradeCancer    PSALevel          CancerVol
##  Min.   : 1   Min.   :0.0000    Min.   :-2.53370   Min.   :-2.30258
##  1st Qu.:25   1st Qu.:0.0000    1st Qu.:-0.65227   1st Qu.:-0.71613
##  Median :49   Median :0.0000    Median : 0.09702   Median : 0.08555
##  Mean   :49   Mean   :0.2165    Mean   : 0.00000   Mean   : 0.00000
##  3rd Qu.:73   3rd Qu.:0.0000    3rd Qu.: 0.50654   3rd Qu.: 0.66550
##  Max.   :97   Max.   :1.0000    Max.   : 2.70223   Max.   : 2.10683
##      Weight             Age          BenignProstaticHyperplasia
##  Min.   :-2.5953   Min.   :-3.0872   Min.   :-0.8406
##  1st Qu.:-0.5519   1st Qu.:-0.5220   1st Qu.:-0.8406
##  Median :-0.0663   Median : 0.1531   Median :-0.3929
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.4597   3rd Qu.: 0.5582   3rd Qu.: 0.7375
##  Max.   : 4.9712   Max.   : 2.0433   Max.   : 2.5678
##  SeminalVesicleInvasion CapsularPenetration
##  Min.   :0.0000         Min.   :-0.5966
##  1st Qu.:0.0000         1st Qu.:-0.5966
##  Median :0.0000         Median :-0.4772
##  Mean   :0.2165         Mean   : 0.0000
##  3rd Qu.:0.0000         3rd Qu.: 0.2681
##  Max.   :1.0000         Max.   : 4.2321
```

```
View(mydata)
names(mydata)
```

```
## [1] "Obs"                       "Y_HighGradeCancer"
## [3] "PSALevel"                  "CancerVol"
## [5] "Weight"                    "Age"
## [7] "BenignProstaticHyperplasia" "SeminalVesicleInvasion"
## [9] "CapsularPenetration"
```

```
# load packages
library(caTools)
library(ROCR)
library(ResourceSelection)
```

```
## Warning:  package 'ResourceSelection' was built under R version 4.0.3
## ResourceSelection 0.3-5 2019-07-22
```

```
library(car)
```

```
## Warning:  package 'car' was built under R version 4.0.3
## Loading required package:  carData
## Warning:  package 'carData' was built under R version 4.0.3
```

```
# declare SeminalVesicleInvasion a categorical variable (SeminalVesicleInvasion == [0, 1])
mydata$SeminalVesicleInvasion <- factor(mydata$SeminalVesicleInvasion)
```

```r
# create training and testing subsets
myseed <- 123
set.seed(myseed)
split <- sample.split(mydata, SplitRatio=0.8)
train <- subset(mydata, split=="TRUE")
test <- subset(mydata, split=="FALSE")

View(train)
View(test)

# write train & test datasets to CSV files
write.csv(train, "./data/processed/train.txt")
write.csv(test, "./data/processed/test.txt")


###################################
### Building Helpful Functions ###
###################################

freq <- function(data) {
  ### function requires one input parameter: data.
  ### this function will display the table of Y_HighGradeCancer counts (frequency table);
  # i.e. the counts of 0's and 1's in the input data.
  ### the function will then display the proportion of 0 to 1 (I already know via
  # previous analysis that the counts of 0's greatly outweigh the count of 1's).
  ### we can consider this proportion to be a "base accuracy" for model comparison;
  # i.e. if the model just predicted 0's (most frequent classification),
  # for all cases.

  name <- deparse(substitute(data))

  if (name=='train') {
    cat('TRAINING DATA\n')
  }
  else {
    cat('TESTING DATA\n')
  }

  freq_tab <- table(data$Y_HighGradeCancer)
  most_freq_prop <- round(sum(freq_tab[1])/sum(freq_tab), 4)
  less_freq_pop <- round(sum(freq_tab[2])/sum(freq_tab), 4)

  # print out both the table, and calculated base accuracy
  cat('Frequency Table:\n')
  print(freq_tab)
  cat('\nThe proportion of 0 to 1 is:', most_freq_prop, '\n')
  cat('The proportion of 1 to 0 is:', less_freq_pop)
}


accuracy <- function(model, data, val=0.50) {
  ### function requires three input parameters: model, data, and decision value boundry
  # (optional); defualt 50%.
```

```
  ### this function will first apply the fitted model and create classifications,
  # then compare to real values (which we know).
  ### the confusion matrix and accuracy score will output to the terminal.
  ### idealy we want the accuracy score to be greater than the base score calculated
  # previously (this indicates the logistic model is a better fit).
  ### decision boundry value may require analysis and adjustments/optimizations afterwards.

  name <- deparse(substitute(data))

  if (name=='train') {
    cat('TRAINING DATA\n')
  }
  else {
    cat('TESTING DATA\n')
  }

  res <- predict(model, data, type="response")
  tab <- table(ActualValue=data$Y_HighGradeCancer, PredictedValue=res>=val)
  err <- round((1-(sum(diag(tab))/sum(tab)))*100, 1)
  acc <- round(sum(diag(tab))/sum(tab)*100, 1)

  # print out confusion matrix, and calculated accuracy
  cat('Prediction Rule:', val, '\n')
  cat('Confusion Matrix:\n', '\n')
  print(tab)
  cat('\nThe calculated error is:', err, '%')
  cat('\nThe calculated accuracy is:', acc, '%')
}



############################
###                      ###
###    Model Fitting     ###
###                      ###
############################

################################################
### Second-Order Polynomial Logistic Models ###
################################################

# fit full second-order logistic model
logit_poly <- glm(Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(CancerVol, 2) +
                    poly(Weight, 2) + poly(Age, 2) + poly(BenignProstaticHyperplasia, 2) +
                    SeminalVesicleInvasion + poly(CapsularPenetration, 2),
                  data=train, family="binomial")
summary(logit_poly)

##
## Call:
## glm(formula = Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(CancerVol,
##     2) + poly(Weight, 2) + poly(Age, 2) + poly(BenignProstaticHyperplasia,
##     2) + SeminalVesicleInvasion + poly(CapsularPenetration, 2),
##     family = "binomial", data = train)
```

```
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.56947  -0.29292  -0.11807  -0.02792   2.70547
##
## Coefficients:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                         -3.1879     1.1911  -2.676  0.00744 **
## poly(PSALevel, 2)1                   8.8748     9.4300   0.941  0.34664
## poly(PSALevel, 2)2                   9.9300     9.7971   1.014  0.31079
## poly(CancerVol, 2)1                  9.0626    12.3092   0.736  0.46158
## poly(CancerVol, 2)2                  3.7797     8.9295   0.423  0.67209
## poly(Weight, 2)1                    -1.7984     8.8561  -0.203  0.83908
## poly(Weight, 2)2                   -22.1802    19.1499  -1.158  0.24676
## poly(Age, 2)1                        3.0558     5.5976   0.546  0.58513
## poly(Age, 2)2                        6.6241     4.6130   1.436  0.15102
## poly(BenignProstaticHyperplasia, 2)1   8.0033     7.2503   1.104  0.26966
## poly(BenignProstaticHyperplasia, 2)2   6.8035     6.3958   1.064  0.28745
## SeminalVesicleInvasion1             -0.9176     1.1728  -0.782  0.43397
## poly(CapsularPenetration, 2)1        2.4574     3.9676   0.619  0.53568
## poly(CapsularPenetration, 2)2       -7.9544     4.4302  -1.795  0.07258 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 72.613  on 75  degrees of freedom
## Residual deviance: 33.998  on 62  degrees of freedom
## AIC: 61.998
##
## Number of Fisher Scoring iterations: 8

step(logit_poly, direction="backward")

## Start:  AIC=62
## Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(CancerVol, 2) +
##     poly(Weight, 2) + poly(Age, 2) + poly(BenignProstaticHyperplasia,
##     2) + SeminalVesicleInvasion + poly(CapsularPenetration, 2)
##
##                                    Df Deviance    AIC
## - poly(CancerVol, 2)                2   35.756 59.756
## - poly(BenignProstaticHyperplasia, 2)  2   35.913 59.913
## - SeminalVesicleInvasion            1   34.644 60.644
## - poly(Weight, 2)                   2   36.698 60.698
## <none>                                  33.998 61.998
## - poly(CapsularPenetration, 2)      2   38.078 62.078
## - poly(Age, 2)                      2   38.511 62.511
## - poly(PSALevel, 2)                 2   40.072 64.072
##
## Step:  AIC=59.76
## Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(Weight, 2) + poly(Age,
##     2) + poly(BenignProstaticHyperplasia, 2) + SeminalVesicleInvasion +
##     poly(CapsularPenetration, 2)
##
```

```
##                                      Df Deviance    AIC
## - poly(BenignProstaticHyperplasia, 2)  2   37.400 57.400
## - poly(Weight, 2)                      2   38.191 58.191
## - SeminalVesicleInvasion               1   36.552 58.552
## <none>                                     35.756 59.756
## - poly(Age, 2)                         2   39.906 59.906
## - poly(CapsularPenetration, 2)         2   42.124 62.124
## - poly(PSALevel, 2)                    2   47.961 67.961
##
## Step:  AIC=57.4
## Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(Weight, 2) + poly(Age,
##     2) + SeminalVesicleInvasion + poly(CapsularPenetration, 2)
##
##                                Df Deviance    AIC
## - poly(Weight, 2)               2   38.338 54.338
## - SeminalVesicleInvasion        1   38.128 56.128
## <none>                              37.400 57.400
## - poly(Age, 2)                  2   43.065 59.065
## - poly(CapsularPenetration, 2)  2   43.734 59.734
## - poly(PSALevel, 2)             2   48.695 64.695
##
## Step:  AIC=54.34
## Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(Age, 2) + SeminalVesicleInvasion +
##     poly(CapsularPenetration, 2)
##
##                                Df Deviance    AIC
## - SeminalVesicleInvasion        1   38.900 52.900
## <none>                              38.338 54.338
## - poly(Age, 2)                  2   44.099 56.099
## - poly(CapsularPenetration, 2)  2   46.605 58.605
## - poly(PSALevel, 2)             2   51.230 63.230
##
## Step:  AIC=52.9
## Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(Age, 2) + poly(CapsularPenetration,
##     2)
##
##                                Df Deviance    AIC
## <none>                              38.900 52.900
## - poly(Age, 2)                  2   44.200 54.200
## - poly(CapsularPenetration, 2)  2   46.739 56.739
## - poly(PSALevel, 2)             2   51.888 61.888
##
## Call:  glm(formula = Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(Age,
##     2) + poly(CapsularPenetration, 2), family = "binomial", data = train)
##
## Coefficients:
##                   (Intercept)             poly(PSALevel, 2)1
##                        -2.604                         12.807
##            poly(PSALevel, 2)2                  poly(Age, 2)1
##                         7.005                          2.768
##                 poly(Age, 2)2  poly(CapsularPenetration, 2)1
##                         6.363                          4.741
## poly(CapsularPenetration, 2)2
```

```
##                       -8.417
##
## Degrees of Freedom: 75 Total (i.e. Null);  69 Residual
## Null Deviance:     72.61
## Residual Deviance: 38.9  AIC: 52.9
```

```
# use second-order reduced model setup for quick analysis of adding/removing predictors
logit_poly_red <- glm(Y_HighGradeCancer ~
                        poly(PSALevel, 2)
                      + poly(CancerVol, 2)
                      # + poly(Weight, 2)
                      # + poly(Age, 2)
                      # + poly(BenignProstaticHyperplasia, 2)
                      # + poly(SeminalVesicleInvasion, 2)
                      #  + poly(CapsularPenetration, 2)
                      , data=train, family="binomial")
summary(logit_poly_red)
```

```
##
## Call:
## glm(formula = Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(CancerVol,
##     2), family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.65643  -0.46122  -0.20861  -0.01794   2.41869
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -2.9802     1.2771  -2.334   0.0196 *
## poly(PSALevel, 2)1    9.0035     8.1120   1.110   0.2670
## poly(PSALevel, 2)2    0.6259     7.4678   0.084   0.9332
## poly(CancerVol, 2)1  17.9731    14.8783   1.208   0.2270
## poly(CancerVol, 2)2  -3.3028    10.0734  -0.328   0.7430
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 72.613  on 75  degrees of freedom
## Residual deviance: 44.511  on 71  degrees of freedom
## AIC: 54.511
##
## Number of Fisher Scoring iterations: 8
```

```
step(logit_poly_red, direction="backward")
```

```
## Start:  AIC=54.51
## Y_HighGradeCancer ~ poly(PSALevel, 2) + poly(CancerVol, 2)
##
##                     Df Deviance    AIC
## - poly(PSALevel, 2)   2   48.120 54.120
## <none>                    44.511 54.511
## - poly(CancerVol, 2)  2   50.767 56.767
##
```

```
## Step:  AIC=54.12
## Y_HighGradeCancer ~ poly(CancerVol, 2)
##
##                       Df Deviance    AIC
## <none>                     48.120 54.120
## - poly(CancerVol, 2)  2   72.613 74.613
##
## Call:  glm(formula = Y_HighGradeCancer ~ poly(CancerVol, 2), family = "binomial",
##     data = train)
##
## Coefficients:
##         (Intercept)  poly(CancerVol, 2)1  poly(CancerVol, 2)2
##             -2.6364              20.0802              -0.4743
##
## Degrees of Freedom: 75 Total (i.e. Null);  73 Residual
## Null Deviance:     72.61
## Residual Deviance: 48.12  AIC: 54.12

####################################
### First-Order Logistic Models ###
####################################

# fit full first-order logistic model
logit_full <- glm(Y_HighGradeCancer ~ PSALevel + CancerVol + Weight +
                    Age + BenignProstaticHyperplasia +
                    SeminalVesicleInvasion + CapsularPenetration, data=train,
                  family="binomial")
summary(logit_full)

##
## Call:
## glm(formula = Y_HighGradeCancer ~ PSALevel + CancerVol + Weight +
##     Age + BenignProstaticHyperplasia + SeminalVesicleInvasion +
##     CapsularPenetration, family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.66861  -0.39211  -0.18721  -0.02467   2.17445
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -2.84305    0.75774  -3.752 0.000175 ***
## PSALevel                    1.28155    0.74609   1.718 0.085856 .
## CancerVol                   1.40464    0.90103   1.559 0.119014
## Weight                     -0.17618    0.75535  -0.233 0.815567
## Age                         0.56784    0.43547   1.304 0.192245
## BenignProstaticHyperplasia  0.07823    0.54237   0.144 0.885320
## SeminalVesicleInvasion1    -0.38818    1.04075  -0.373 0.709164
## CapsularPenetration         0.25330    0.44939   0.564 0.572988
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

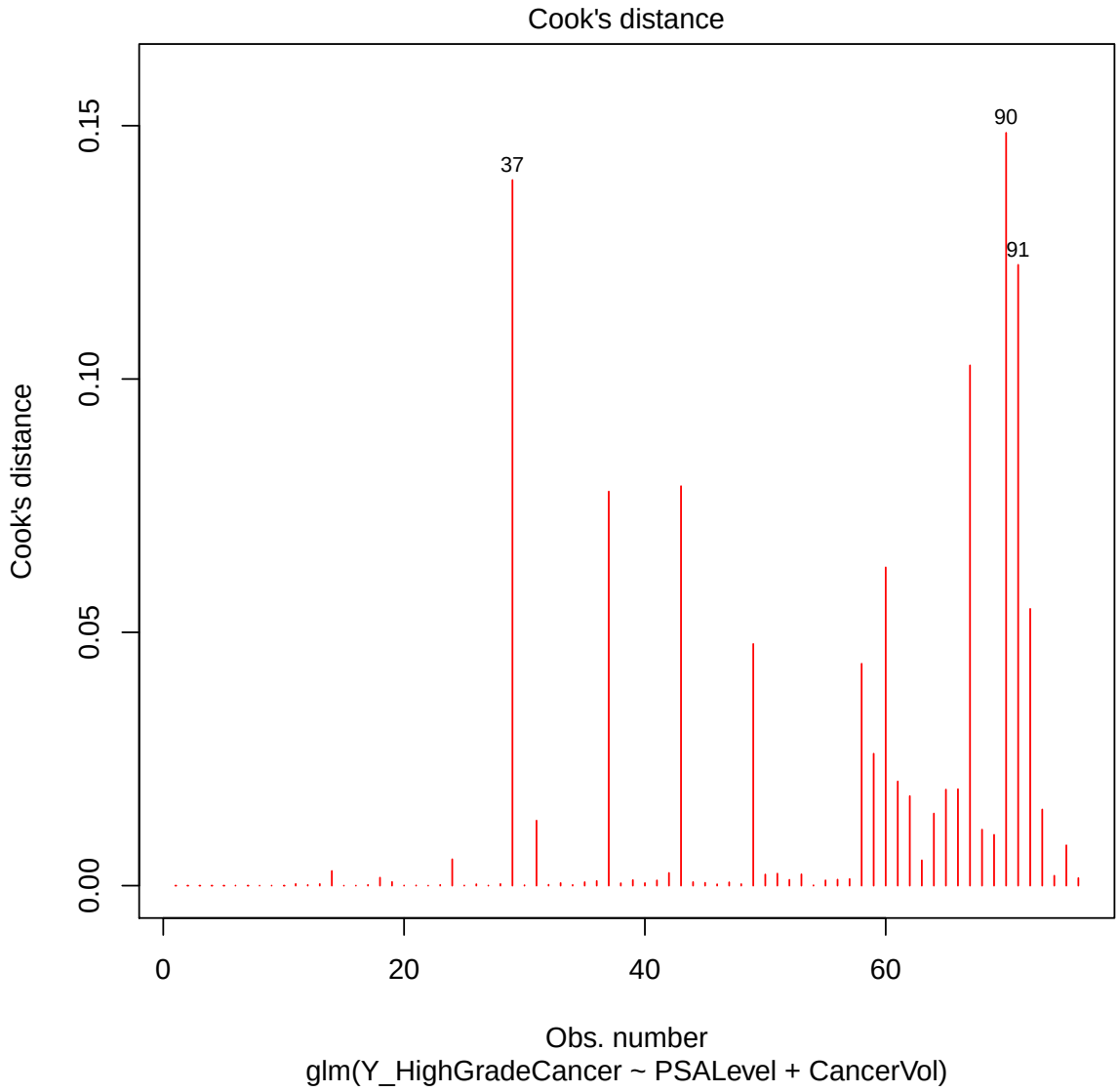
```
##     Null deviance: 72.613  on 75  degrees of freedom
## Residual deviance: 42.303  on 68  degrees of freedom
## AIC: 58.303
##
## Number of Fisher Scoring iterations: 7
```

```
step(logit_full, direction="backward")
```

```
## Start:  AIC=58.3
## Y_HighGradeCancer ~ PSALevel + CancerVol + Weight + Age + BenignProstaticHyperplasia +
##     SeminalVesicleInvasion + CapsularPenetration
##
##                              Df Deviance    AIC
## - BenignProstaticHyperplasia  1   42.324 56.324
## - Weight                      1   42.358 56.358
## - SeminalVesicleInvasion      1   42.444 56.444
## - CapsularPenetration         1   42.627 56.627
## - Age                         1   43.954 57.954
## <none>                            42.303 58.303
## - CancerVol                   1   45.225 59.225
## - PSALevel                    1   45.903 59.903
##
## Step:  AIC=56.32
## Y_HighGradeCancer ~ PSALevel + CancerVol + Weight + Age + SeminalVesicleInvasion +
##     CapsularPenetration
##
##                           Df Deviance    AIC
## - Weight                   1   42.358 54.358
## - SeminalVesicleInvasion   1   42.499 54.499
## - CapsularPenetration      1   42.695 54.695
## - Age                      1   44.044 56.044
## <none>                         42.324 56.324
## - CancerVol                1   45.389 57.389
## - PSALevel                 1   46.031 58.031
##
## Step:  AIC=54.36
## Y_HighGradeCancer ~ PSALevel + CancerVol + Age + SeminalVesicleInvasion +
##     CapsularPenetration
##
##                           Df Deviance    AIC
## - SeminalVesicleInvasion   1   42.522 52.522
## - CapsularPenetration      1   42.755 52.755
## - Age                      1   44.151 54.151
## <none>                         42.358 54.358
## - CancerVol                1   45.390 55.390
## - PSALevel                 1   46.059 56.059
##
## Step:  AIC=52.52
## Y_HighGradeCancer ~ PSALevel + CancerVol + Age + CapsularPenetration
##
##                         Df Deviance    AIC
## - CapsularPenetration    1   42.780 50.780
## - Age                    1   44.168 52.168
## <none>                       42.522 52.522
```

```
## - CancerVol            1   45.558 53.558
## - PSALevel             1   46.285 54.285
##
## Step:  AIC=50.78
## Y_HighGradeCancer ~ PSALevel + CancerVol + Age
##
##             Df Deviance    AIC
## - Age        1   44.628 50.628
## <none>           42.780 50.780
## - PSALevel   1   46.445 52.445
## - CancerVol  1   48.777 54.777
##
## Step:  AIC=50.63
## Y_HighGradeCancer ~ PSALevel + CancerVol
##
##             Df Deviance    AIC
## <none>           44.628 50.628
## - PSALevel   1   48.123 52.123
## - CancerVol  1   50.767 54.767
##
## Call:  glm(formula = Y_HighGradeCancer ~ PSALevel + CancerVol, family = "binomial",
##     data = train)
##
## Coefficients:
## (Intercept)     PSALevel    CancerVol
##      -2.687        1.058        1.550
##
## Degrees of Freedom: 75 Total (i.e. Null);  73 Residual
## Null Deviance:     72.61
## Residual Deviance: 44.63  AIC: 50.63

# use reduced model setup for quick analysis of adding/removing predictors
logit_red <- glm(Y_HighGradeCancer ~
                   PSALevel
                 + CancerVol
                 # + Weight
                 # + Age
                 # + BenignProstaticHyperplasia
                 # + SeminalVesicleInvasion
                 # + CapsularPenetration
                 , data=train, family="binomial")
summary(logit_red)

##
## Call:
## glm(formula = Y_HighGradeCancer ~ PSALevel + CancerVol, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.73560  -0.43637  -0.23378  -0.03521   2.42555
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  -2.6867     0.6186  -4.343 1.41e-05 ***
## PSALevel      1.0577     0.6198   1.707   0.0879 .
## CancerVol     1.5502     0.6859   2.260   0.0238 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 72.613  on 75  degrees of freedom
## Residual deviance: 44.628  on 73  degrees of freedom
## AIC: 50.628
##
## Number of Fisher Scoring iterations: 6

##########################################
###                                    ###
###    Model Checking and Validation   ###
###                                    ###
##########################################

#####################################################################
### Cook's Distance Diagnostics for Influential Observations ###
#####################################################################
plot(logit_red, pch=18, col="red", which=c(4))
```




```
myCDs <- sort(round(cooks.distance(logit_red), 5), decreasing=TRUE)
myCDs
```

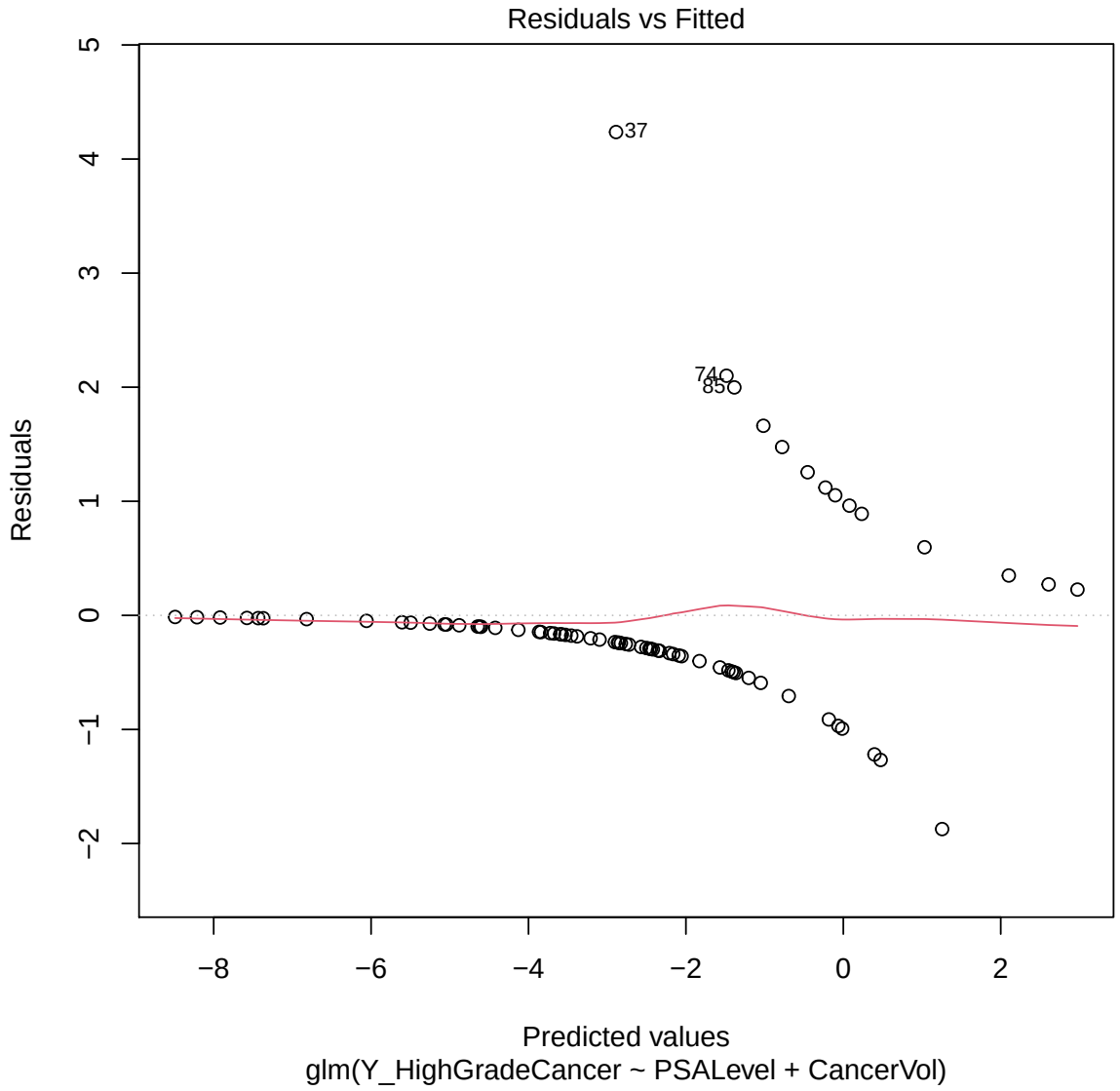
```
##      90      37      91      85      55      47      76      92      63      74
## 0.14859 0.13928 0.12254 0.10270 0.07886 0.07780 0.06282 0.05464 0.04771 0.04380
##      75      78      84      83      79      93      82      39      87      88
## 0.02604 0.02058 0.01905 0.01897 0.01768 0.01504 0.01424 0.01284 0.01107 0.01004
##      96      30      81      18      54      65      67      64      94      22
## 0.00797 0.00521 0.00500 0.00290 0.00251 0.00238 0.00224 0.00221 0.00196 0.00158
##      97      73      72      66      49      70      52      46      24      56
## 0.00149 0.00131 0.00119 0.00116 0.00110 0.00106 0.00105 0.00092 0.00073 0.00071
##      45      60      57      42      51      48      13      36      16      61
## 0.00069 0.00066 0.00057 0.00052 0.00052 0.00048 0.00034 0.00033 0.00032 0.00031
##      33      58      40      21      29      15      43      38      69      27
```

```
## 0.00029 0.00028 0.00018 0.00015 0.00015 0.00014 0.00014 0.00009 0.00008 0.00005
##      25      31      34       7      20      10      11      28       1       2
## 0.00004 0.00004 0.00003 0.00002 0.00002 0.00001 0.00001 0.00001 0.00000 0.00000
##       3       4       6       9      12      19
## 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
```

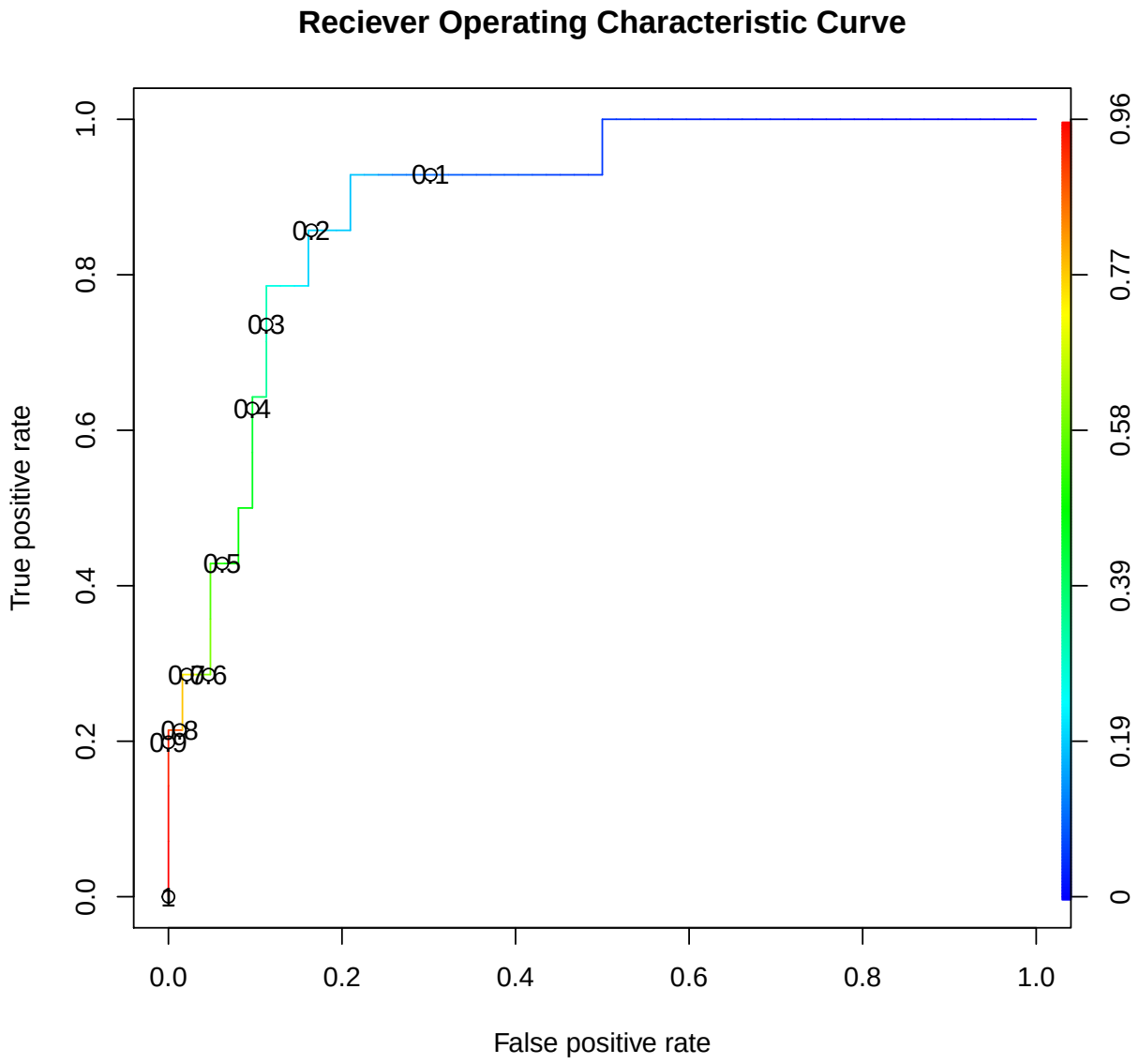
```
# drop rows for model building (influential observations)
# this step will be visited within Coook's Distance analysis
train_trim <- subset(train, Obs != 90
                     #  & Obs != 37
                     #  & Obs != 91
)

# view the trimmed data
View(train_trim)

# write train_trim dataset to csv file
write.csv(train_trim, "./data/processed/train_trim.txt")

# re-fit the logistic model
logit_red_trim <- glm(Y_HighGradeCancer ~
                        PSALevel
                      + CancerVol
                      # + Weight
                      # + Age
                      # + BenignProstaticHyperplasia
                      # + SeminalVesicleInvasion
                      # + CapsularPenetration
                      , data=train_trim, family="binomial")
summary(logit_red_trim)
```

```
##
## Call:
## glm(formula = Y_HighGradeCancer ~ PSALevel + CancerVol, family = "binomial",
##     data = train_trim)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.71156  -0.43371  -0.20855  -0.03378   2.46797
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.9030     0.6907  -4.203 2.63e-05 ***
## PSALevel      0.7495     0.6120   1.225   0.2207
## CancerVol     1.9077     0.7711   2.474   0.0134 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 69.170  on 74  degrees of freedom
## Residual deviance: 41.576  on 72  degrees of freedom
## AIC: 47.576
##
```

```
## Number of Fisher Scoring iterations: 6

# RESULT: the removal of these rows did not improve the model.
# continue forward with original logit_red model

##############################################
### Hosmer-Lemeshow Goodness of Fit Test ###
##############################################

gof <- hoslem.test(logit_red$y, fitted(logit_red), g=5) # choosing 5 groups
cbind(gof$expected, gof$observed)

##                         yhat0      yhat1 y0 y1
## [0.000206,0.00973] 15.941605 0.05839507 16  0
## (0.00973,0.0521]   14.592690 0.40730980 15  0
## (0.0521,0.103]     13.898096 1.10190404 14  1
## (0.103,0.333]      11.889917 3.11008275 11  4
## (0.333,0.951]       5.677692 9.32230835  6  9

gof

##
##  Hosmer and Lemeshow goodness of fit (GOF) test
##
## data:  logit_red$y, fitted(logit_red)
## X-squared = 0.83815, df = 3, p-value = 0.8403

######################
### vizualizations ###
######################

# Residuals vs. Fitted
# Normal Q-Q
# scale-location (Predicted Values vs. sqrt[Std. Pearson Residuals])
# Residuals vs. Leverage
# residualPlot(logit_red, type="pearson")
plot(logit_red, which=c(1))
```




```
### build and plot ROC curve ###
pred <- predict(logit_red, train, type="response")
ROCRPred <- prediction(pred, train$Y_HighGradeCancer)
ROCRPref <- performance(ROCRPred, "tpr", "fpr")
plot(ROCRPref, colorize=TRUE, print.cutoffs.at=seq(0.1, by=0.1),
     main="Reciever Operating Characteristic Curve")
```




```
###################################
### Accuracy Model Comparisons ###
###################################

### invoke functions ###
freq(train)

## TRAINING DATA
## Frequency Table:
##
##  0  1
## 62 14
##
## The proportion of 0 to 1 is: 0.8158
```

```
## The proportion of 1 to 0 is: 0.1842

accuracy(logit_red, train, 0.184) # starting point prediction rule

## TRAINING DATA
## Prediction Rule: 0.184
## Confusion Matrix:
##
##             PredictedValue
## ActualValue FALSE TRUE
##           0    49   13
##           1     1   13
##
## The calculated error is: 18.4 %
## The calculated accuracy is: 81.6 %

accuracy(logit_red, train, 0.20) # final prediction rule

## TRAINING DATA
## Prediction Rule: 0.2
## Confusion Matrix:
##
##             PredictedValue
## ActualValue FALSE TRUE
##           0    52   10
##           1     2   12
##
## The calculated error is: 15.8 %
## The calculated accuracy is: 84.2 %

#########################
###                   ###
###    Final Model    ###
###                   ###
#########################

# no changes have been made from the reduced model
logit_final <- logit_red
summary(logit_final)

##
## Call:
## glm(formula = Y_HighGradeCancer ~ PSALevel + CancerVol, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.73560  -0.43637  -0.23378  -0.03521   2.42555
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.6867     0.6186  -4.343 1.41e-05 ***
## PSALevel      1.0577     0.6198   1.707   0.0879 .
## CancerVol     1.5502     0.6859   2.260   0.0238 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 72.613  on 75  degrees of freedom
## Residual deviance: 44.628  on 73  degrees of freedom
## AIC: 50.628
##
## Number of Fisher Scoring iterations: 6

########################################
###                                  ###
###    Model Validation: Test Data   ###
###                                  ###
########################################

###################################
### Accuracy Model Comparisons ###
###################################

### invoke functions ###
freq(test)

## TESTING DATA
## Frequency Table:
##
##  0  1
## 14  7
##
## The proportion of 0 to 1 is: 0.6667
## The proportion of 1 to 0 is: 0.3333

accuracy(logit_final, test, 0.20)

## TESTING DATA
## Prediction Rule: 0.2
## Confusion Matrix:
##
##             PredictedValue
## ActualValue FALSE TRUE
##           0    13    1
##           1     2    5
##
## The calculated error is: 14.3 %
## The calculated accuracy is: 85.7 %
```