EASTERN MICHIGAN UNIVERSITY

MASTER'S RESEARCH

DEPARTMENT OF MATHEMATICS AND STATISTICS

Prostate Cancer: Multiple Logistic Regression

Author
JEFFREY OSIWALA

Supervisor
Prof. KHAIRUL ISLAM

October 23, 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 asses 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 things, 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, bowl, 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 test 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 growing 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 health care 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 of 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

(Eqn. X), 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 (1) 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*.

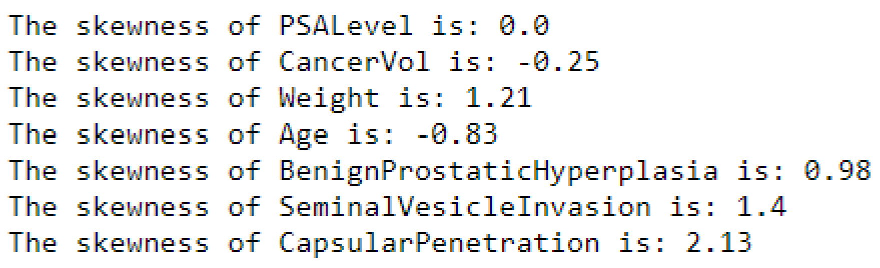
4.1 Data Transformations and Standardization

In modeling using logistic regression, the appropriate transformations on continuous variables are necessary to optimize the model predictiveness.

Variable transformation is an important technique to create robust models using logistic regression. 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, did not affect any underlying distributions, and was performed by using the following design:

The finalized data skewness is summarized directly below. Following, I've included the histogram of PSA Level vs. Cancer Volume vs. Age, a helpful visual for the three predictors which carried the most significance through much of my analysis, as we soon shall see.



```
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: insert caption here.

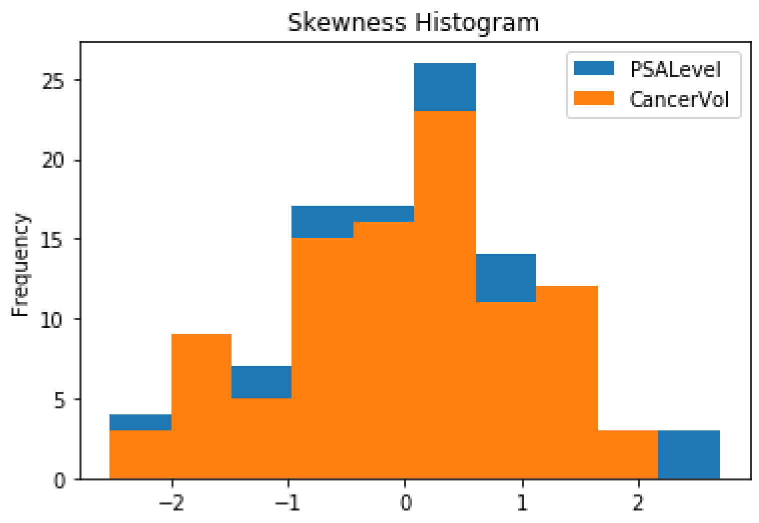


Figure 2: insert caption here.

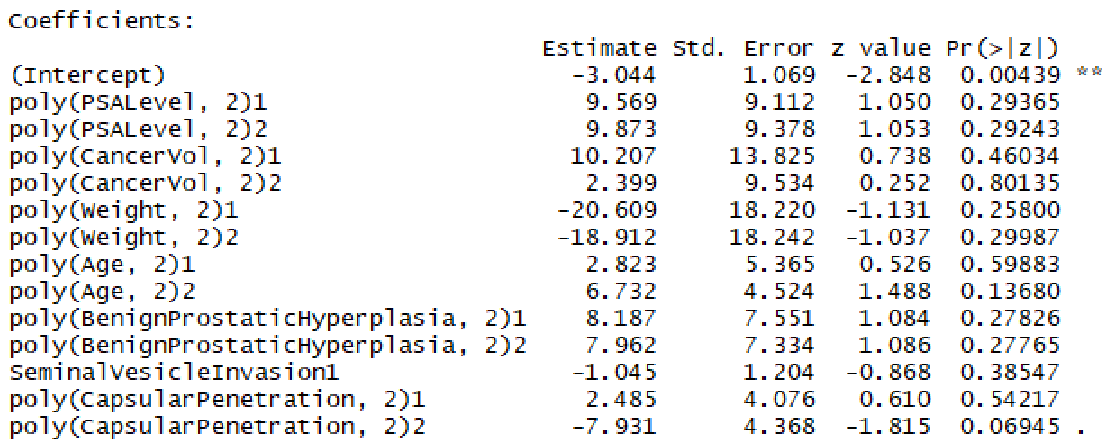
4.2 Second-Order Predictors

Occasionally, the first-order logistic model may not provide an adequate fit to the data, so I began my analysis by first attempting to fit the Prostate Cancer data to a *polynomial logistic* regression model. For simplicity, a 2^{nd}-order polynomial model in two predictors has a logit response function as:

$$logit(\pi) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 \tag{5}$$

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 XXX has one or more peaks or valleys, and in such cases a polynomial function can provide a satisfactory approximation to XXX. However, a polynomial fit was not successful here, as indicated by *non-significant* p-values across all predictors, at 5% significance. Additionally, my preliminary scatter plot analysis did not indicate any reason to believe a polynomial fit would be suitable in this study. FOR EXAMPLE, PSA LEVEL WAS A MAJOR FOCUS OF THIS STUDY, AND I'VE PROVIDED THE PLOT BELOW...ETC ETC... Without hesitation, I will move forward with the development of a multiple logistic *linear* regression model.



```
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 3: insert caption here

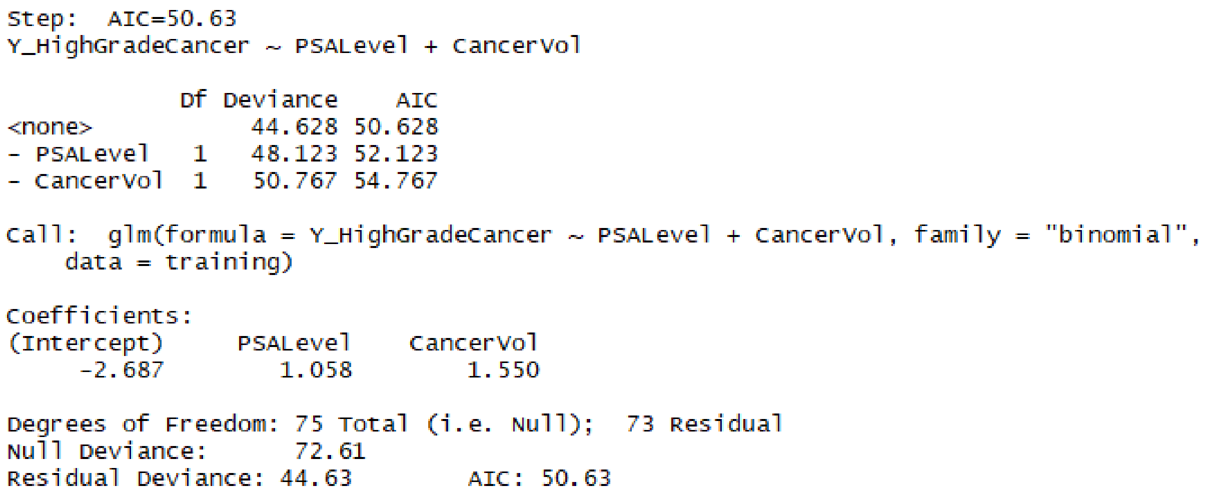
4.3 Model Selection

The data of 97 individual men in the Prostate Cancer sample was split at 80% for train and test 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. Table XXX in columns 1-X contains the variables... The primary purpose of the study was to asses the strength of the association between each of the predictor variables, the predictable nature of PSA Level, and the probability of a man having been diagnosed with high grade prostate cancer over low grade.

4.3.1 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 XXX 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: insert my caption here

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.

4.3.2 Model Fitting

A first-order multiple logistic regression model with two predictor variables was considered to be reasonable by §4.3.1:

$$\pi(\mathbf{X}) = \frac{exp('\boldsymbol{\beta})}{1 + exp(\mathbf{X}'\boldsymbol{\beta})} = [1 + exp(-\mathbf{X}'\boldsymbol{\beta})]^{-1} \tag{6}$$

where:

$$\mathbf{X}'\boldsymbol{\beta} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \tag{7}$$

This model was fitted by the method of maximum likelihood to the data from the 76 train cases. The results are summarized in Table XXX below. The estimated logistic response function is:

$$\hat{\pi} = [1 + exp(-2.6867 + 1.0577X_1 + 1.5502X_2)]^{-1} \tag{8}$$

Now, 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. We can see, for instance, that the odds of a man being diagnosed with high grade prostate cancer increase by about 105% for each additional score of PSA Level, for a given Cancer Volume. This means each unit increase of PSA Level approximately doubles the odds of said diagnosis.

With the estimated logistic regression equation now developed, it is left to analyze the residuals, test goodness of fit, adjust the model if required, and finally apply the final model to the validation data and discuss the results.

4.3.3 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 XXX displays a three-dimensional plot of a 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).

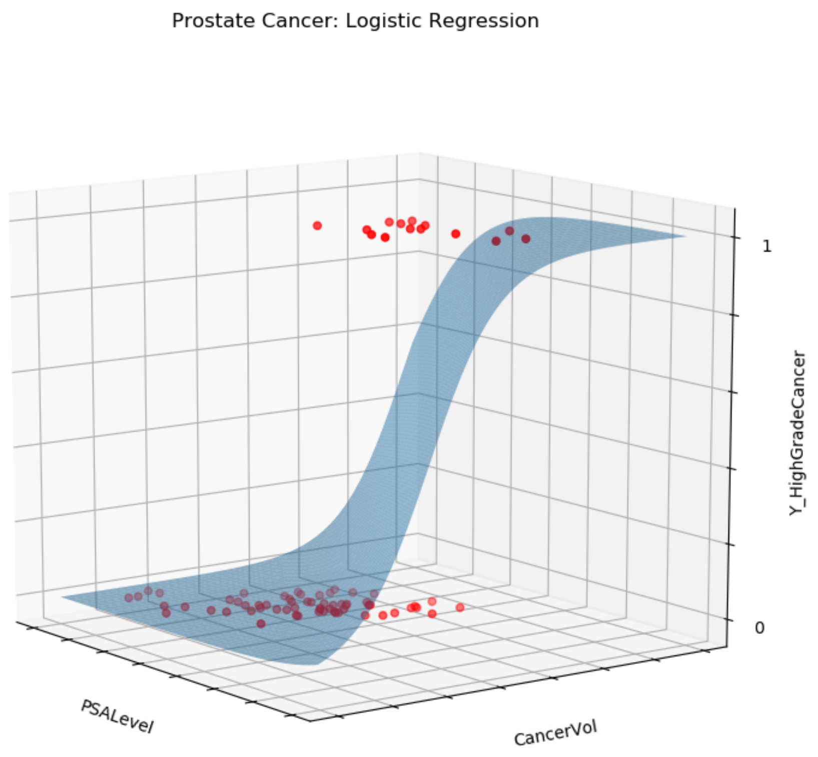


Figure 5: insert caption here.

4.4 Analysis of Residuals

In this section I will discuss the analysis of residuals and the identification of any influential cases 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.

4.4.1 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 approximately a horizontal line with zero intercept. Any significant departure from this suggests that the model may be inadequate. Shown in Figure XXX are the Pearson residuals plotted against the linear predictor, with the lowess smooth superimposed.

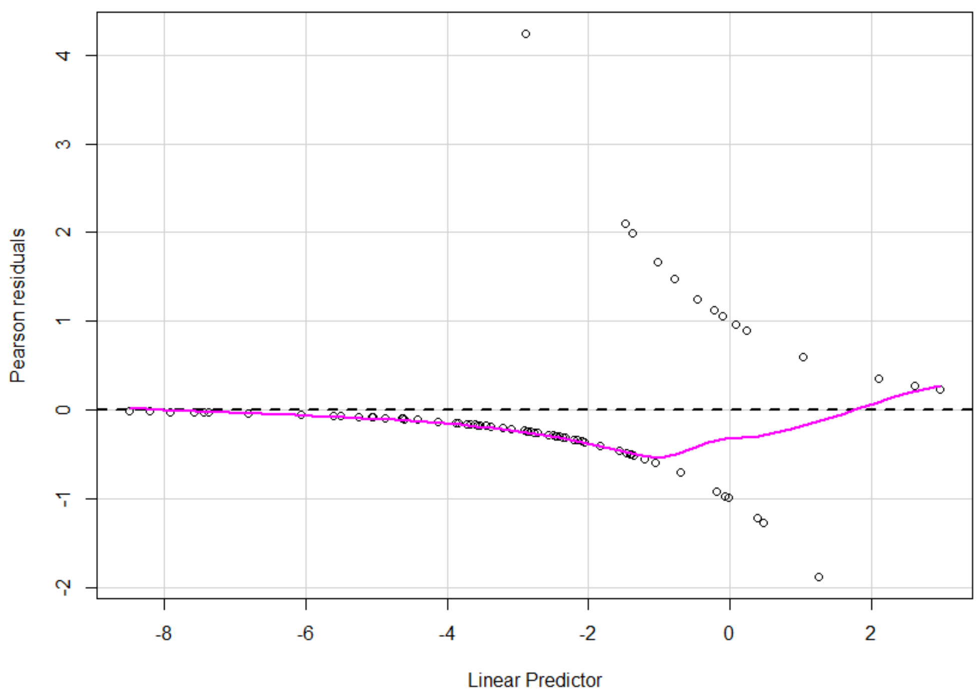


Figure 6: insert caption here

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.

4.4.2 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 Appendix for a portion of the Prostate Cancer testing data.

The plot of distances in Figure XXX 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.

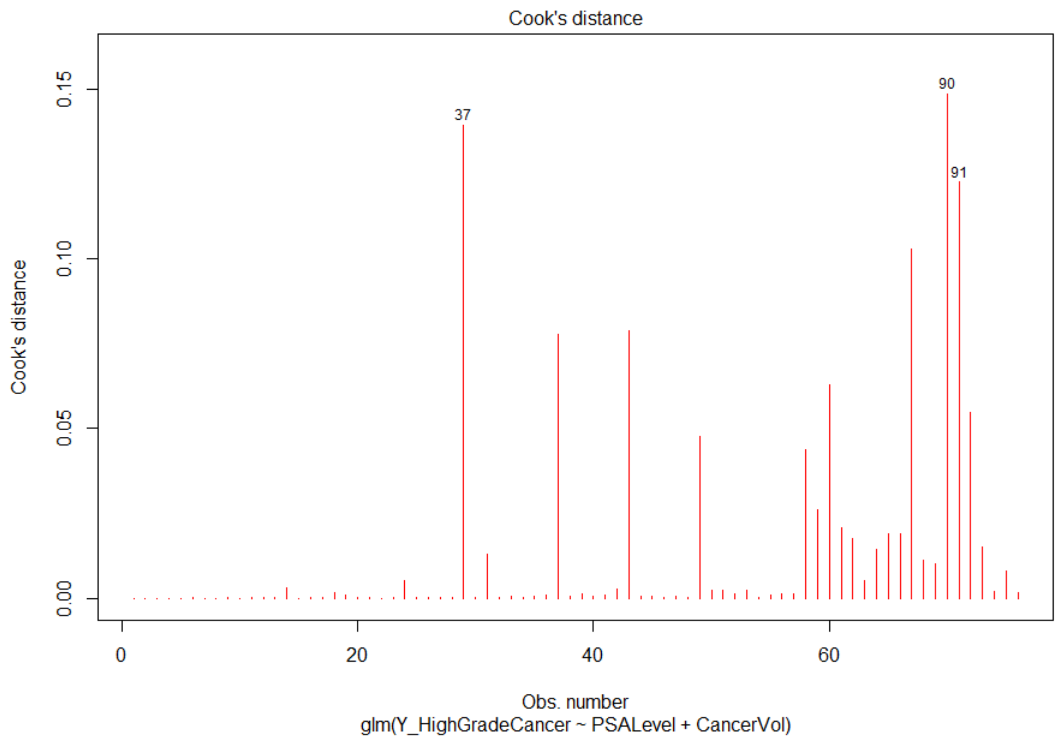


Figure 7: insert caption here

4.5 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:

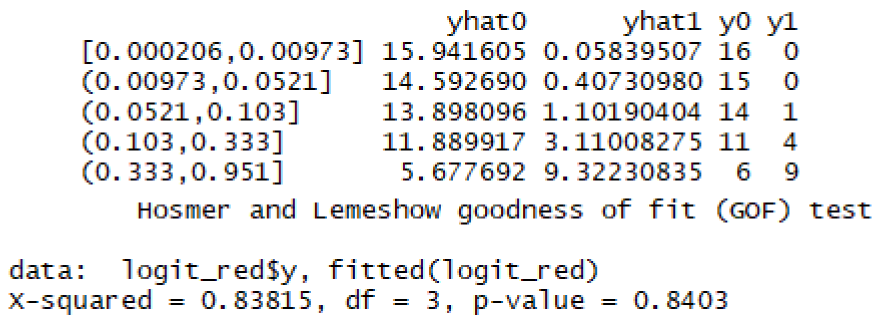
$$\begin{aligned} H_0 &: E[Y] = [1 + exp(-\mathbf{X}'\boldsymbol{\beta})]^{-1} \\ H_0 &: E[Y] \neq [1 + exp(-\mathbf{X}'\boldsymbol{\beta})]^{-1} \end{aligned} \tag{10}$$

4.5.1 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 XXX.



```
                          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 8: caption here

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

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

Thus, for $\alpha = 0.5$ and $c - 2 = 3$, we require $\chi^2(.95;3) = 7.81$. 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.

4.6 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 $1 - P(\hat{Y} = 0|Y = 0)$ 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 XXX below exhibits the ROC curve for model XXX for all possible cut points between 0 and 1.

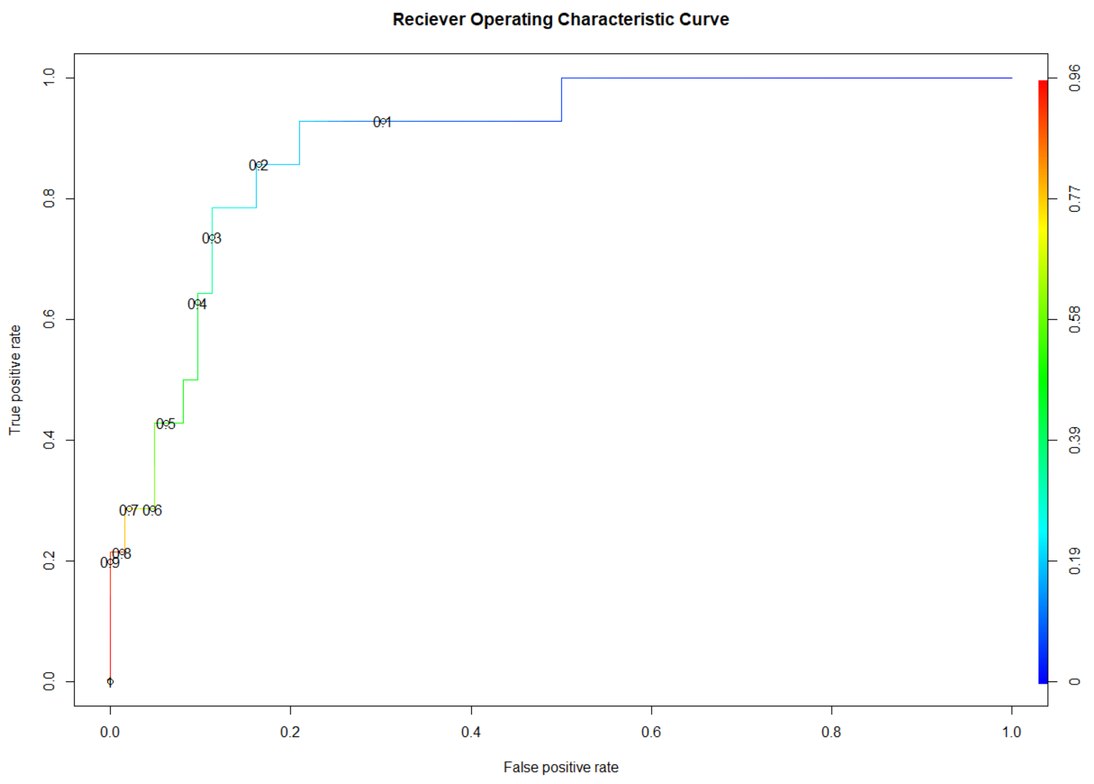


Figure 9: insert caption here

4.6.1 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 was:

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

Table XXX below provides a summary of the number of correct and incorrect classifications based on prediction rule XXX. 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 with rule XXX to not have it, or 7.1%. Altogether, $13 + 1 = 14$ of the 76 predictions would be incorrect, so that the prediction error rate for rule XXX is $14/76 = 0.184$ or 18.4%. Coincidentally, the model exactly matches our training set proportions with the current prediction rule.

Prediction Rule XXX			
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 XXX and Prediction Rules.

With this baseline understood, it is straightforward to choose a stronger cutoff point in utilizing the ROC curve of Figure XXX. 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 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.

Looking at Figure XXX I see a step occurring at 0.20, and use this value for my cutoff candidate. The effects of this change can be summarized in the Confusion Matrix table below.

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

Table 2: Classification based on Logistic Response Function XXX and Prediction Rules.

The model accuracy has now increased with a significantly better false-positive rate, thus intended to reduce the footprint across the health care economy.

Thus, my updated and final prediction rule is stated as follows:

$$\text{Predict 1 if } \hat{\pi}_h \geq 0.3\text{; predict 0 if } \hat{\pi}_h < 0.3 \tag{14}$$

4.7 Model: Strengths and Weaknesses

4.7.1 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 - their correlation values are 0.497 and 0.565, 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 XXX below, with a color legend given in the upper-left corner.

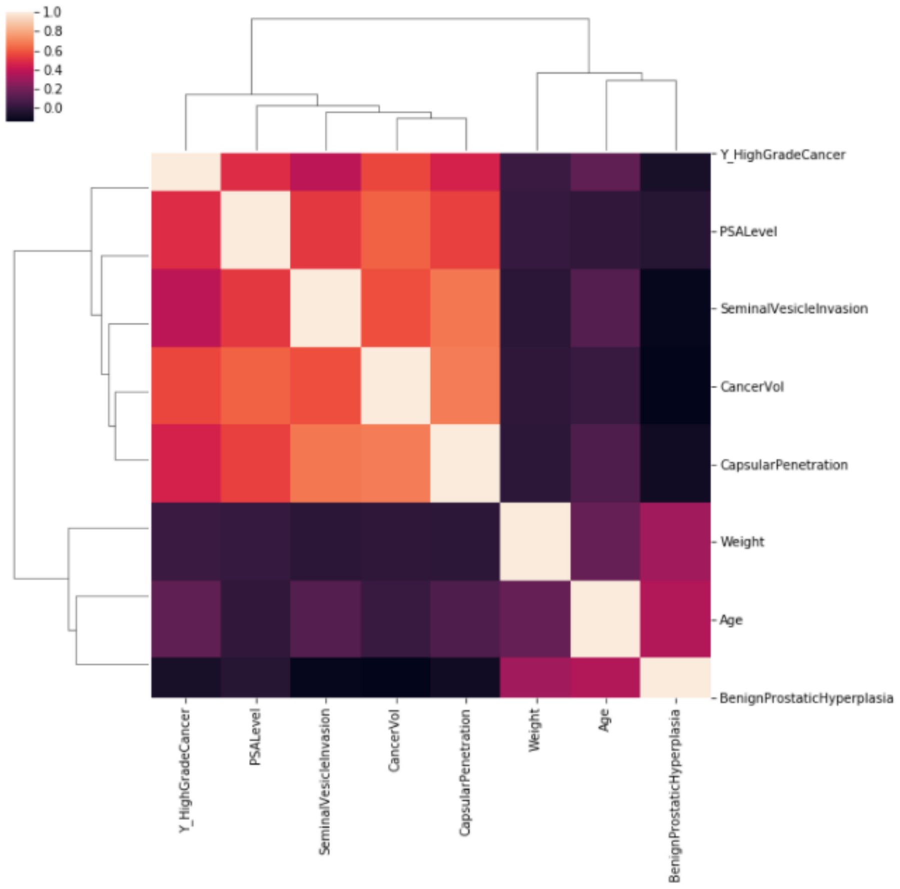
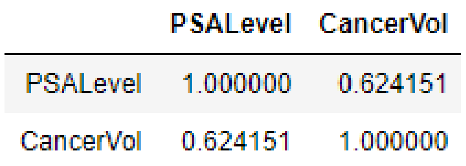


Figure 10: insert my caption here

-The final model has an excellent accuracy score of 84.21%. Deploying this model in use could 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, treatment and a plan can be devised sooner, as well as doing so by only PSA Level information, and not invasive testing.

4.7.2 Weaknesses

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



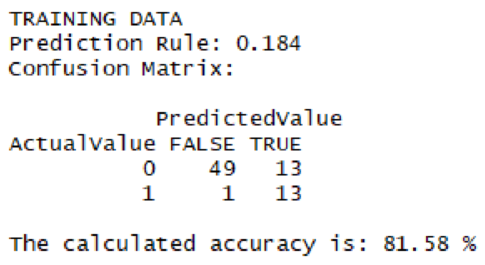
	PSALevel	CancerVol
PSALevel	1.000000	0.624151
CancerVol	0.624151	1.000000

Figure 11: insert my caption here

As shown, PSA Level and Cancer Volume have a mild correlation value of 0.624. 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 can further lead to over-fitting. Therefore, this logistic model may be considered mildly "noisy", sensitive, and not particularly robust.

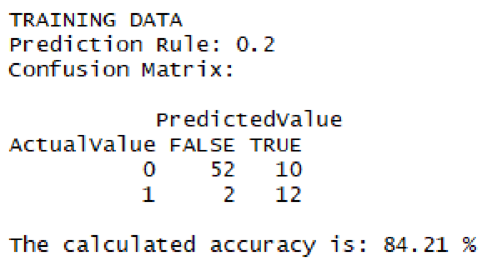
-The Goodness of Fit Evaluation of §4.5 deserves some concern regarding the Pearson chi-square test. As described previously, the Hosmer-Lemeshow procedure was utilized to determine a goodness of fit, and the test statistic is known to be well approximated by the chi-square distribution with $c-2$ degrees of freedom (Eqn. XXX). However, in view of the **R** output (Figure XXX) with 5 groupings, the expected values 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 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 XXX 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 81.58%. After ROC analysis the final prediction cutoff I've employed is 0.2, which by Figure XXX produces 2 counts of false-negatives (14.3% error rate), and an overall model accuracy of 84.21%. Thus the model accuracy has improved 2.63 percentage points by correctly predicting more non high grade cancer cases, but also doubling the false-negative rate. Because correctly identifying high grade cancer patients maybe 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 accuracy is: 81.58 %
```

(a) A subfigure



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

           PredictedValue
ActualValue FALSE TRUE
          0    52   10
          1     2   12

The calculated accuracy is: 84.21 %
```

(b) A subfigure

Figure 12: A figure with two subfigures

5 Conclusion

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 (FUNCTION XXX) based on the model-building data set:

$$\hat{\pi} = [1 + exp(-2.6867 + 1.0577X_1 + 1.5502X_2)]^{-1} \tag{15}$$

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

$$\text{Predict 1 if } \hat{\pi}_h \geq 0.2\text{; predict 0 if } \hat{\pi}_h < 0.2 \tag{16}$$

was then applied to these estimated probabilities. The percent prediction error rates were as follows:

Disease Status

With High Grade Cancer	Without High Grade Cancer	Total
14	49	62

Table 3: Classification based on Logistic Response Function XXX and Prediction Rules.

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 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

7 Appendix

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]:
```
     PSALevel  CancerVol  Weight  Age  BenignProstaticHyperplasia  \
Obs
1       0.651     0.5599  15.959   50                         0.0
2       0.852     0.3716  27.660   58                         0.0
3       0.852     0.6005  14.732   74                         0.0
4       0.852     0.3012  26.576   58                         0.0
5       1.448     2.1170  30.877   62                         0.0

     SeminalVesicleInvasion  CapsularPenetration  GleasonScore  \
Obs
1                         0                  0.0             6
2                         0                  0.0             7
3                         0                  0.0             7
4                         0                  0.0             6
5                         0                  0.0             6

     Y_HighGradeCancer
Obs
1                    0
2                    0
3                    0
4                    0
5                    0
```

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

[8]:
```
     PSALevel  CancerVol  Weight  Age  BenignProstaticHyperplasia  \
Obs
93     80.640    16.9455  48.424   68                      0.0000
94    107.770    45.6042  49.402   44                      0.0000
95    170.716    18.3568  29.964   52                      0.0000
96    239.847    17.8143  43.380   68                      4.7588
97    265.072    32.1367  52.985   68                      1.5527

     SeminalVesicleInvasion  CapsularPenetration  GleasonScore  \
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
```

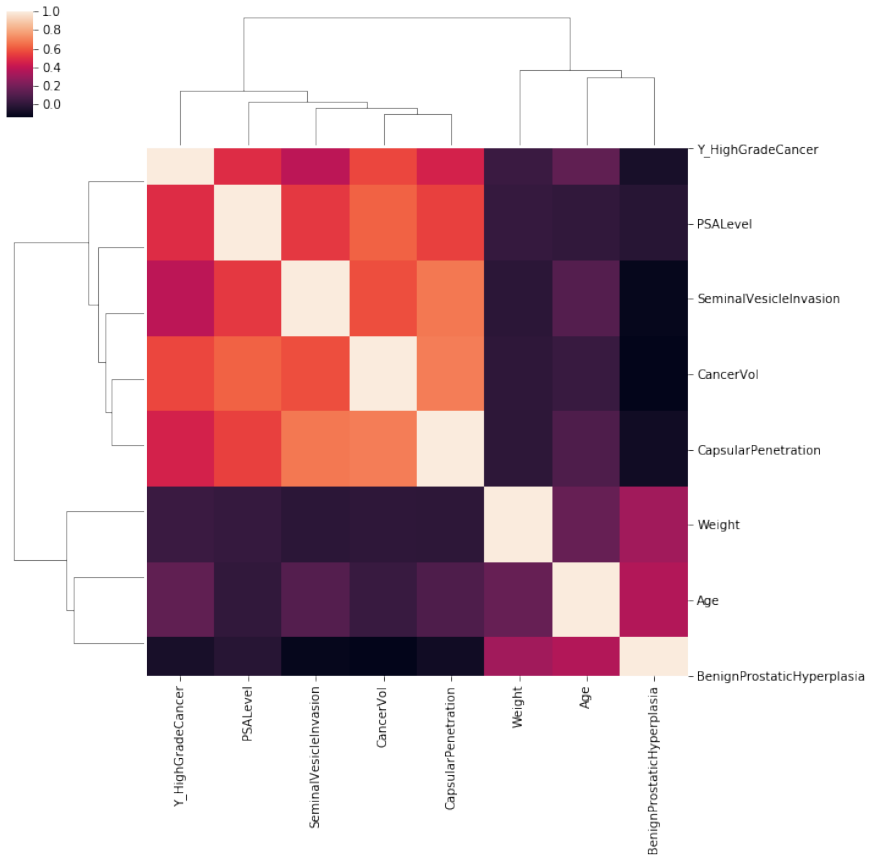
- PSALevel, CancerVol, Weight and Age appear to have high standard deviation values. This may provoke a standardized dataset, or transformations, further in my analysis.
- Skewness will need to be investigated.

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

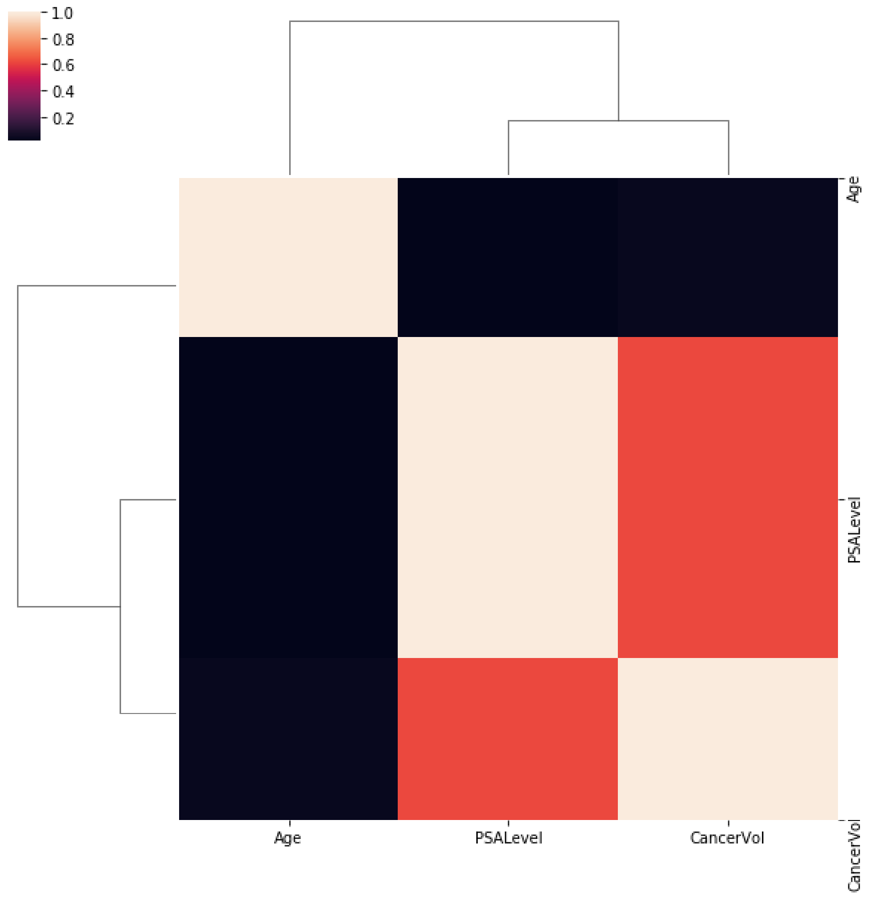
```
[13]:                             PSALevel  CancerVol    Weight       Age  \
PSALevel                    1.000000   0.624151  0.026213  0.017199
CancerVol                   0.624151   1.000000  0.005107  0.039094
Weight                      0.026213   0.005107  1.000000  0.164324
```

```
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]:
```python
# 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()
```

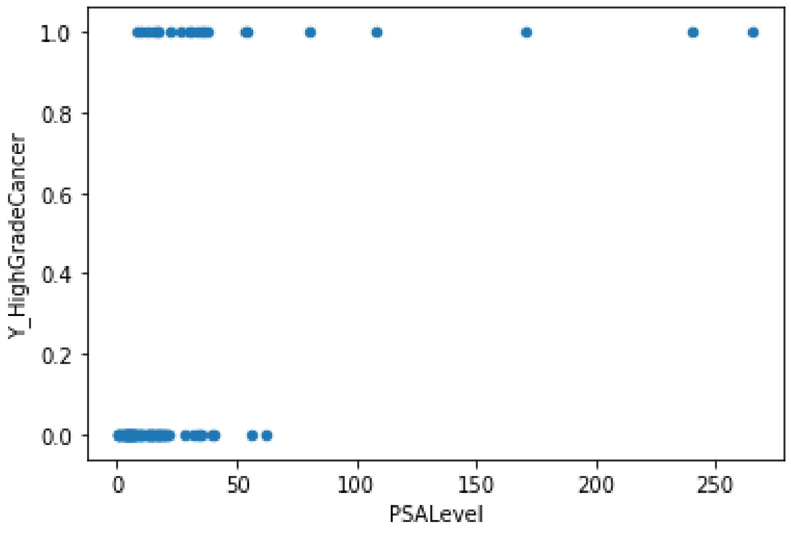
[16]:
```
           PSALevel  CancerVol
PSALevel   1.000000   0.624151
CancerVol  0.624151   1.000000
```

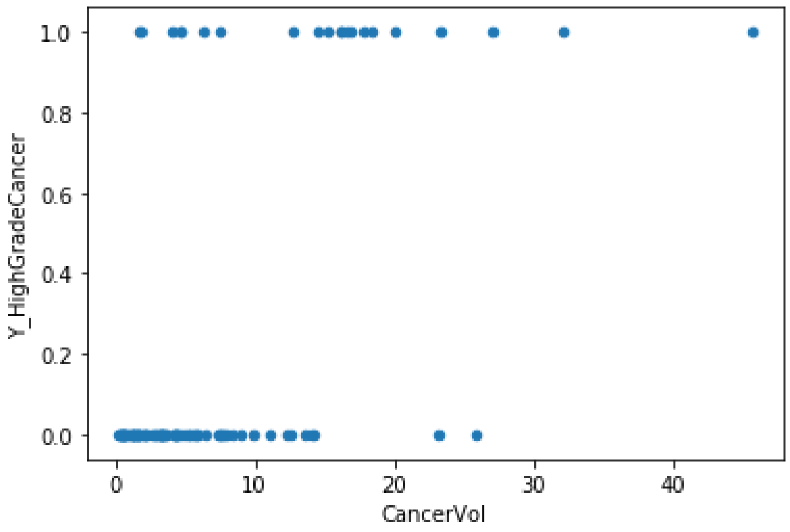
- PSALevel and CancerVol show a mild level of correlation: 0.624151

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

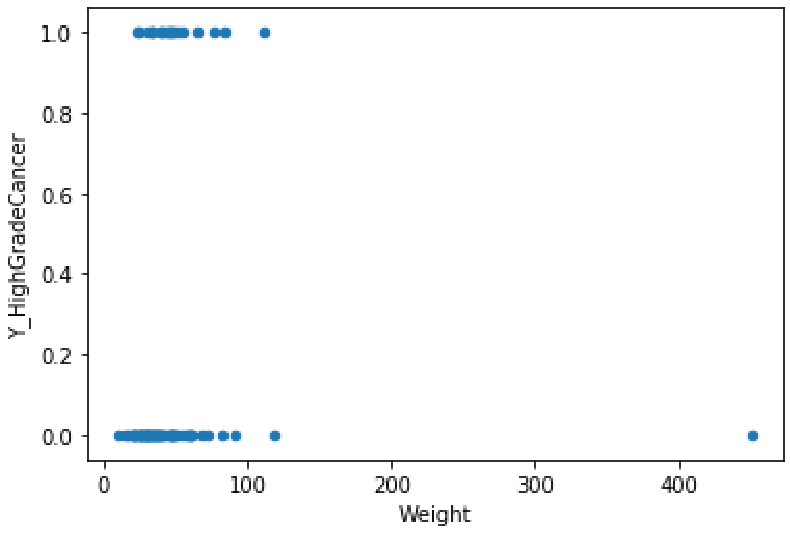
```
Mean Age: 63.87
Median Age: 65.0
```

```
[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');
```

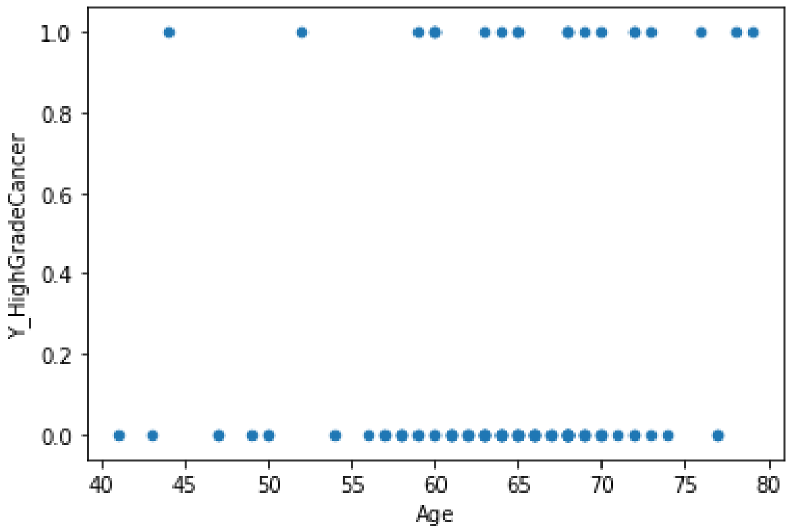




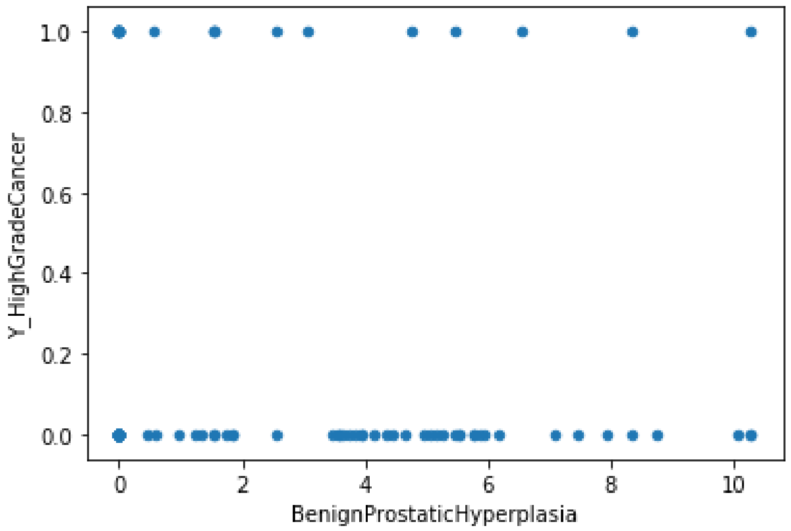
Y_HighGradeCancer
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
CancerVol



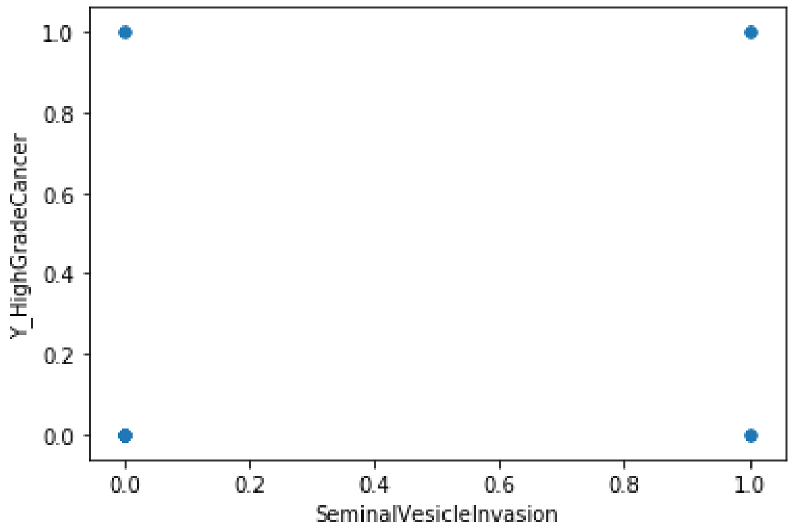
Y_HighGradeCancer
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
Weight



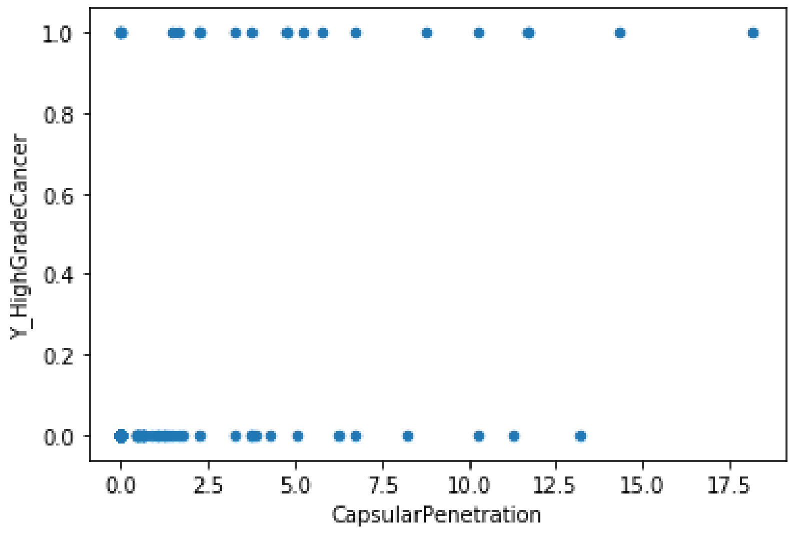
Y_HighGradeCancer
1.0
0.8
0.6
0.4
0.2
0.0
40
45
50
55
60
65
70
75
80
Age



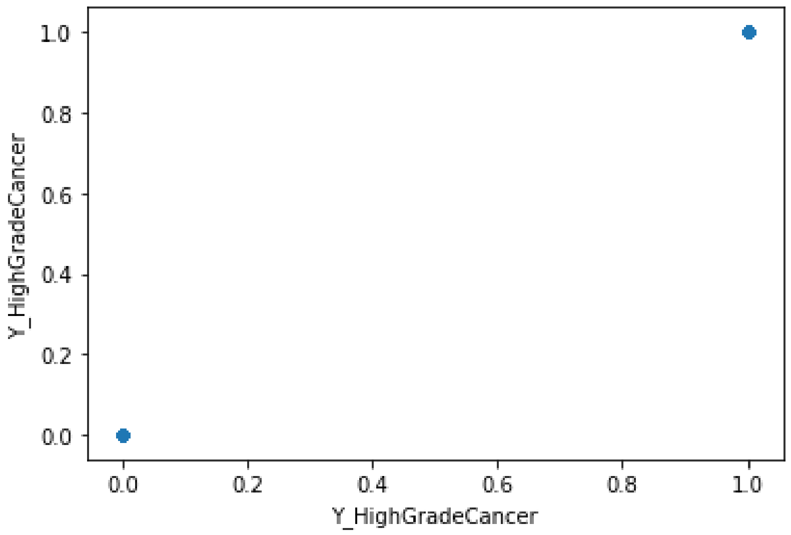
Y_HighGradeCancer
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
BenignProstaticHyperplasia



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

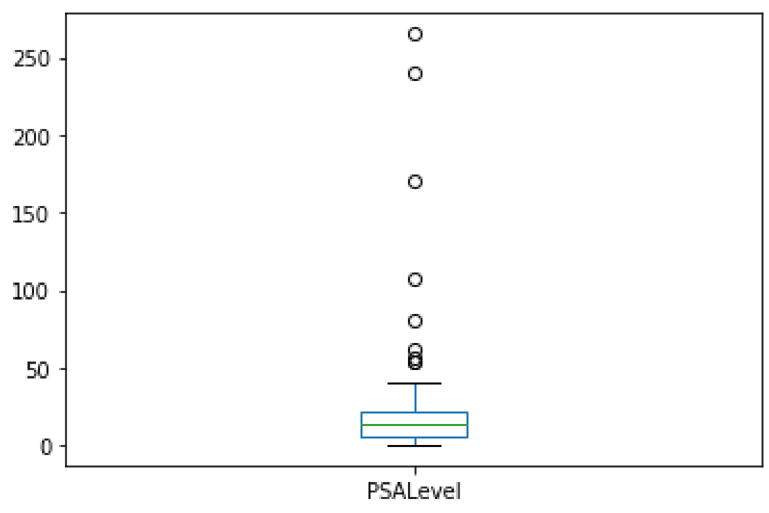


- PSALevel and CancerVol may be good predictors for Y_HighGradeCancer in a Logistic Regression model.
- Weight appears to contain an extreme outlier.
- CancerVol may present two outliers, when Y_HighGradeCancer = 0.
- Via a priori knowledge, Age may be a valuable predictor of Prostate Cancer, so I will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

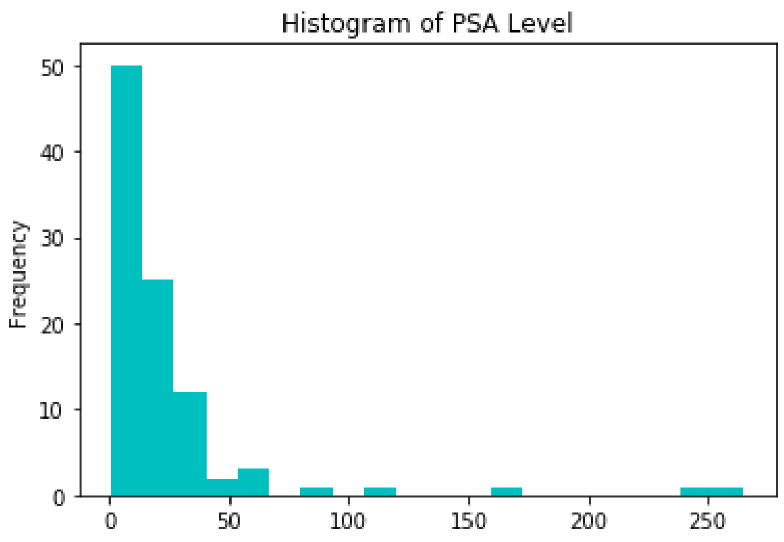
1.4 Distributions

1.4.1 PSALevel

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



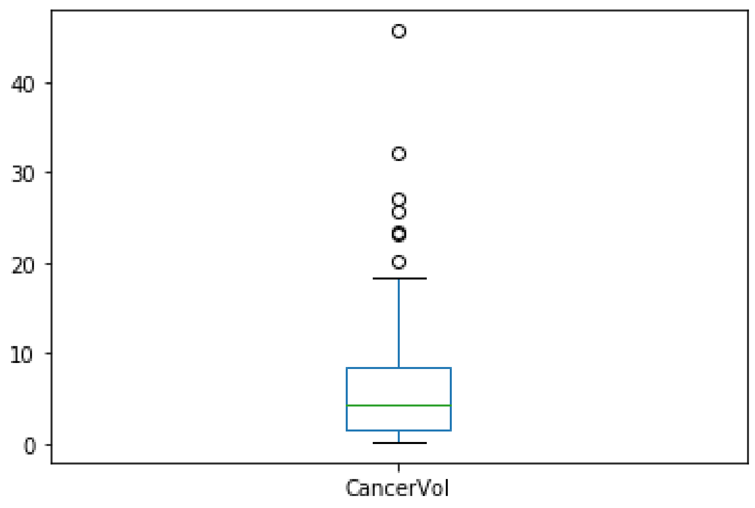
```
[20]: # use hist to create histogram
      df.PSALevel.plot(kind='hist', title='Histogram of PSA Level', color='c',␣
       ↪bins=20);
```



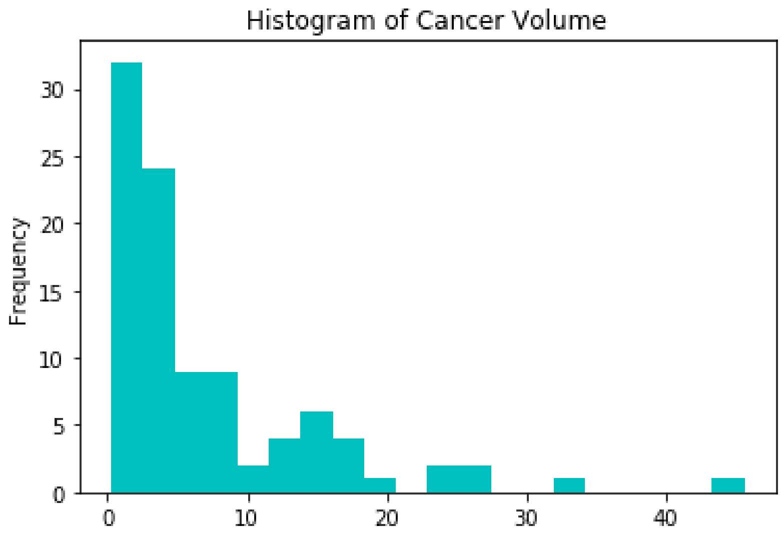
- PSALevel shows high positive skewness.

1.4.2 CancerVol

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



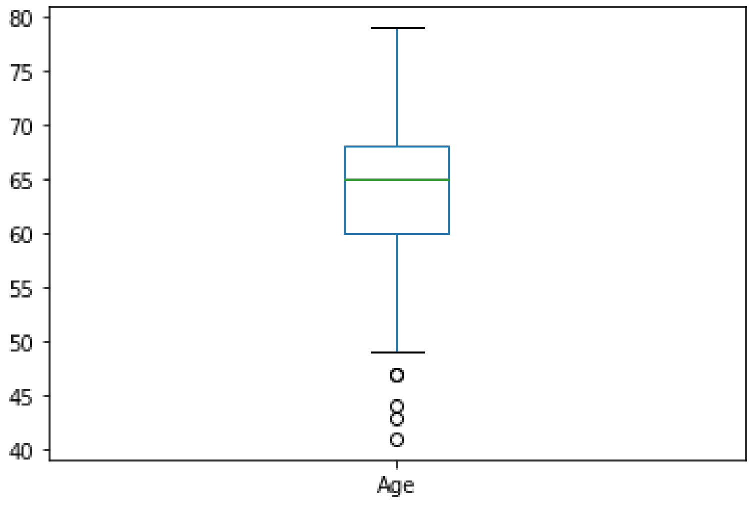
```
[22]: df.CancerVol.plot(kind='hist', title='Histogram of Cancer Volume', color='c',␣
      ↪bins=20);
```



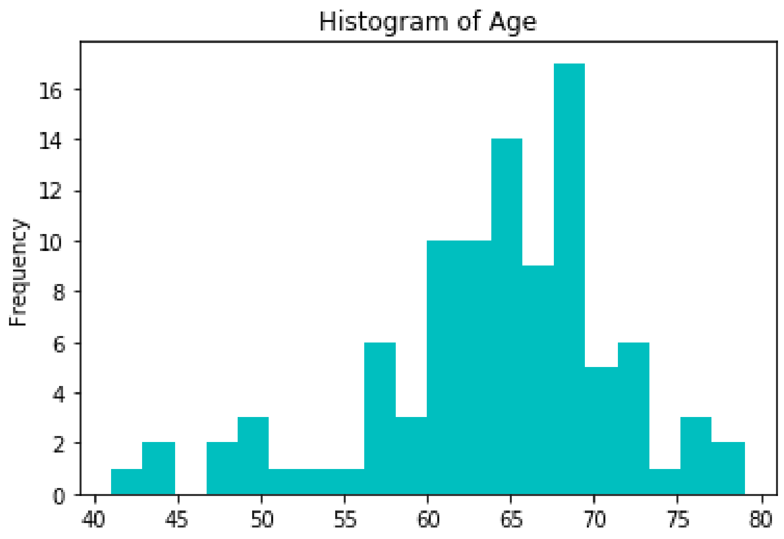
- CancerVol shows high positive skewness.

1.4.3 Age

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

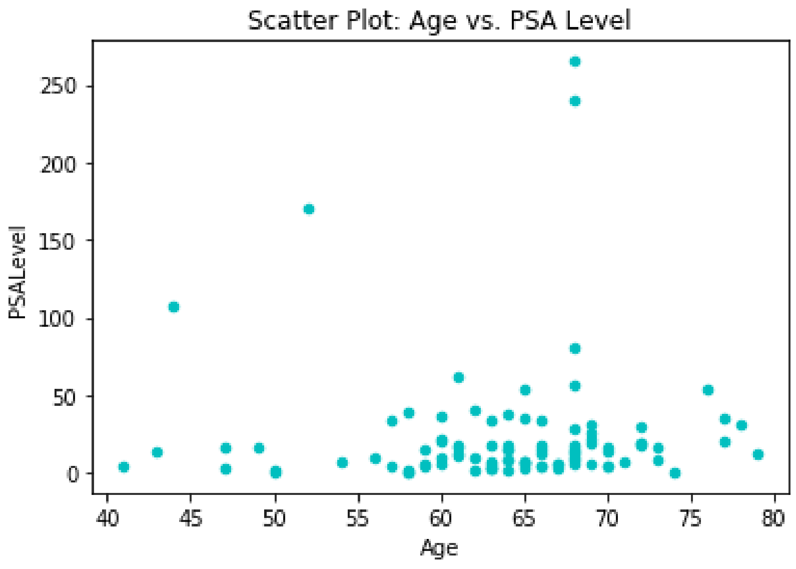


```
[24]: df.Age.plot(kind='hist', title='Histogram of Age', color='c', bins=20);
```

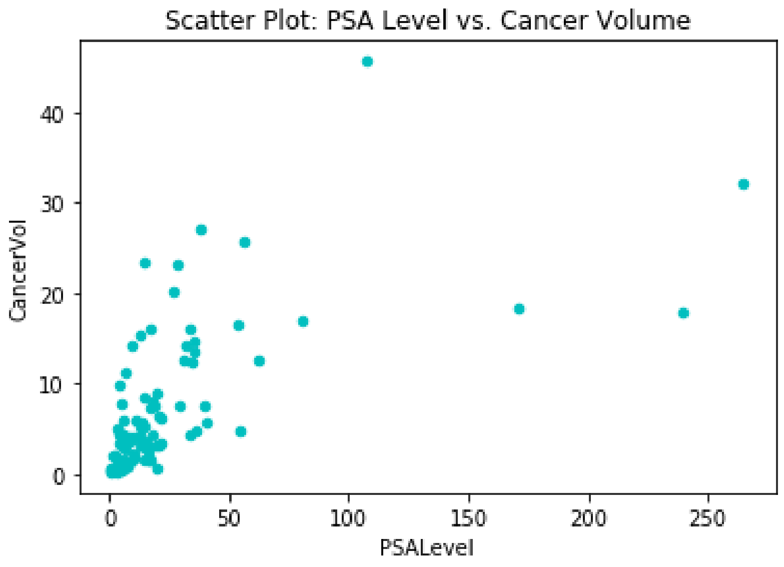


1.4.4 Bi-variate Interactions

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

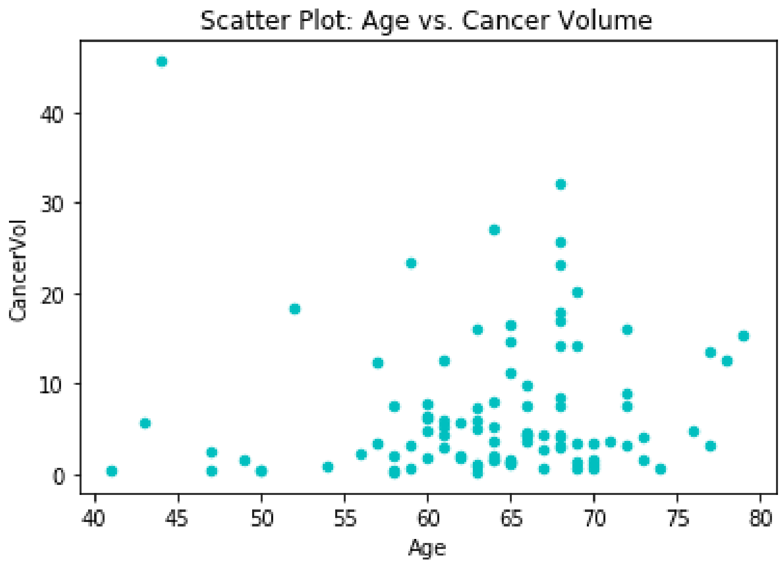


```
[26]: df.plot.scatter(x='PSALevel', y='CancerVol', color='c', title='Scatter Plot:␣
      ↪PSA Level vs. Cancer Volume');
```

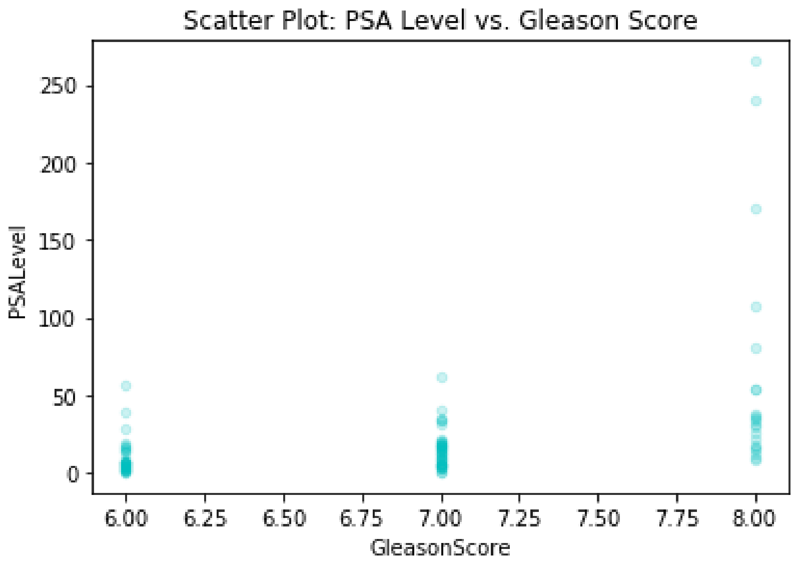


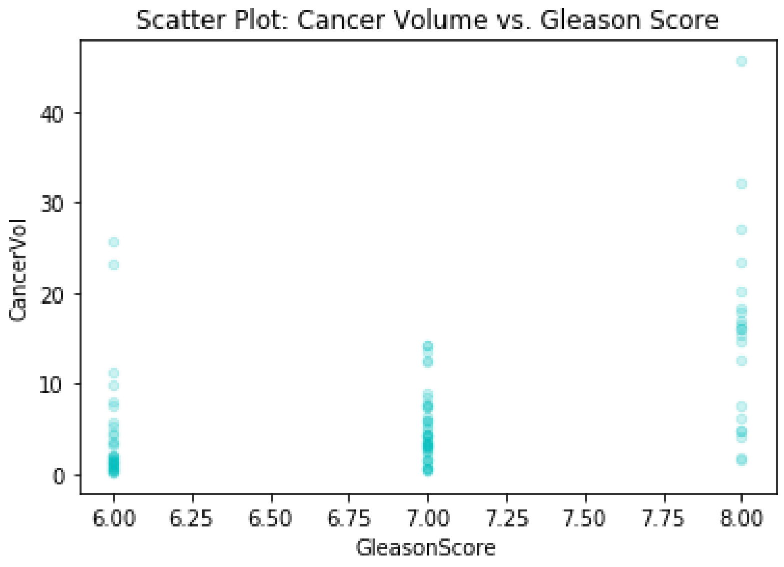
- PSA Level & Cancer Volume display a mild level of correlation

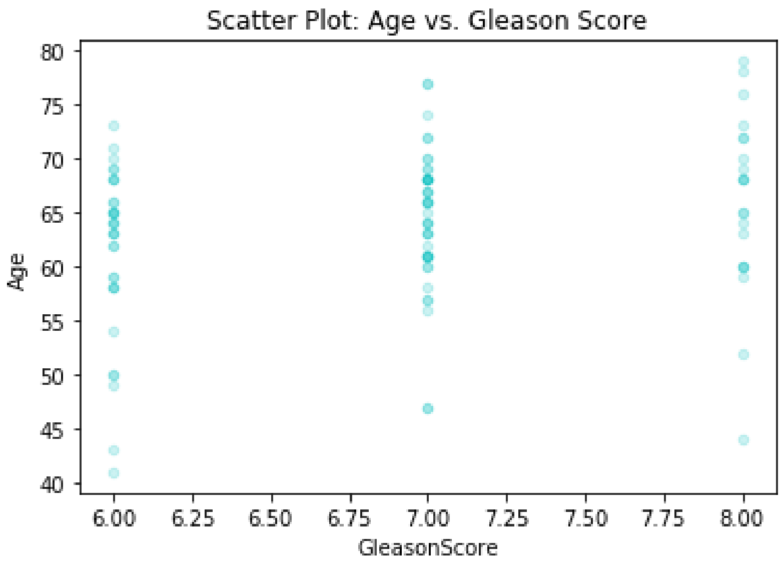
[27]:
```
df.plot.scatter(x='Age', y='CancerVol', color='c', title='Scatter Plot: Age vs.␣
↪Cancer Volume');
```



[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
0
50
100
150
200
250
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
0
10
20
30
40
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]:
```
     PSALevel  CancerVol   Weight  Age  BenignProstaticHyperplasia  \
Obs
89     53.517    16.6099  112.168   65                      0.0000
90     54.055     4.7588   40.447   76                      2.5600
91     56.261    25.7903   60.340   68                      0.0000
92     62.178    12.5535   39.646   61                      3.8574
93     80.640    16.9455   48.424   68                      0.0000
94    107.770    45.6042   49.402   44                      0.0000
95    170.716    18.3568   29.964   52                      0.0000
96    239.847    17.8143   43.380   68                      4.7588
97    265.072    32.1367   52.985   68                      1.5527

     SeminalVesicleInvasion  CapsularPenetration  GleasonScore  \
Obs
89                        1              11.7048             8
90                        1               2.2479             8
91                        0               0.0000             6
92                        1               0.0000             7
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
89                   1
90                   1
91                   0
```

```
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]:
```
     PSALevel  CancerVol  Weight  Age  BenignProstaticHyperplasia  \
Obs
9       2.858     0.4584  34.467   47                         0.0
19      4.759     0.5712  26.311   41                         0.0
49     13.330     5.7546  33.115   43                         0.0
57     16.281     2.6379  17.637   47                         0.0
94    107.770    45.6042  49.402   44                         0.0

     SeminalVesicleInvasion  CapsularPenetration  GleasonScore  \
Obs
9                         0               0.0000             7
19                        0               0.0000             6
49                        0               0.0000             6
57                        0               1.6487             7
94                        1               8.7583             8

     Y_HighGradeCancer
Obs
9                    0
19                   0
49                   0
57                   0
94                   1
```

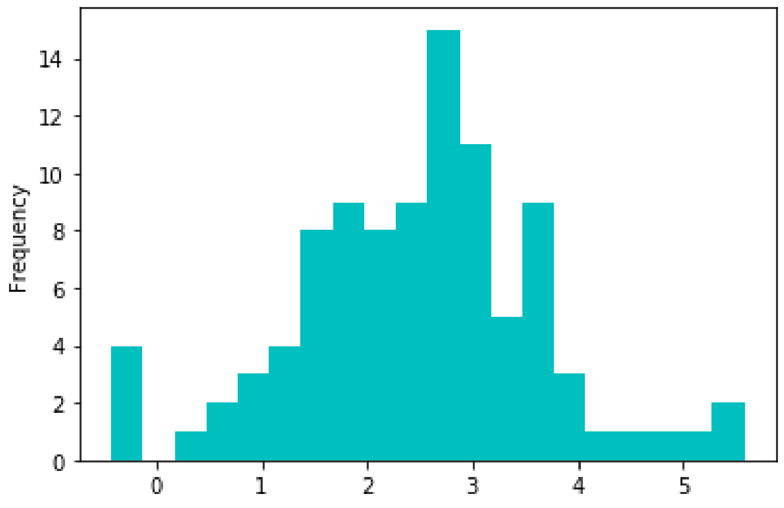
1.6 Transformations

- 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.

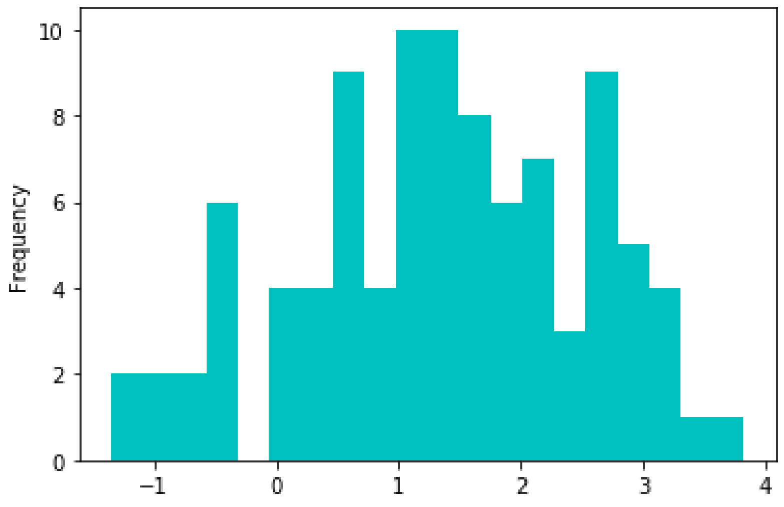
[33]:
```
# try log transformations to reduce skewness
log_PSALevel = np.log(df.PSALevel)
log_CancerVol = np.log(df.CancerVol)
```

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

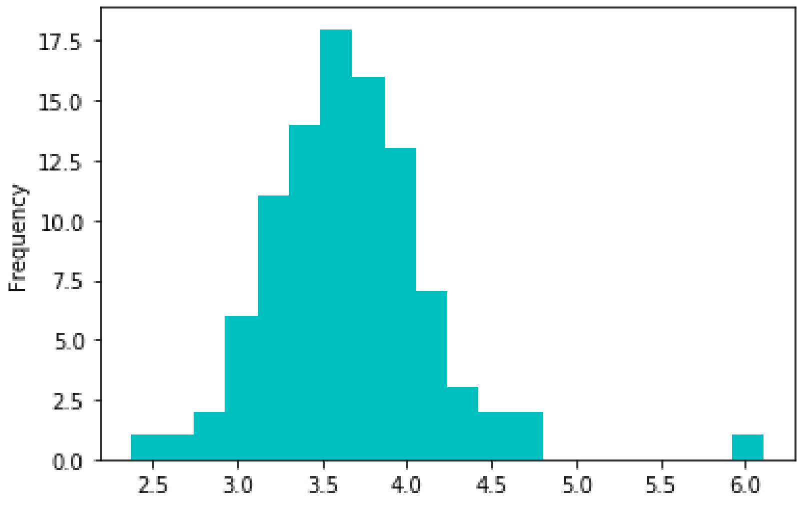
[34]:
```
# histogram of log PSALevel
log_PSALevel.plot(kind='hist', color='c', bins=20);
```



[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]:
```
     Y_HighGradeCancer  PSALevel  CancerVol  Weight  Age  \
Obs
1                    0     0.651     0.5599  15.959   50
2                    0     0.852     0.3716  27.660   58
3                    0     0.852     0.6005  14.732   74
4                    0     0.852     0.3012  26.576   58
5                    0     1.448     2.1170  30.877   62

     BenignProstaticHyperplasia  SeminalVesicleInvasion  CapsularPenetration
Obs
1                           0.0                       0                  0.0
2                           0.0                       0                  0.0
3                           0.0                       0                  0.0
4                           0.0                       0                  0.0
5                           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
```

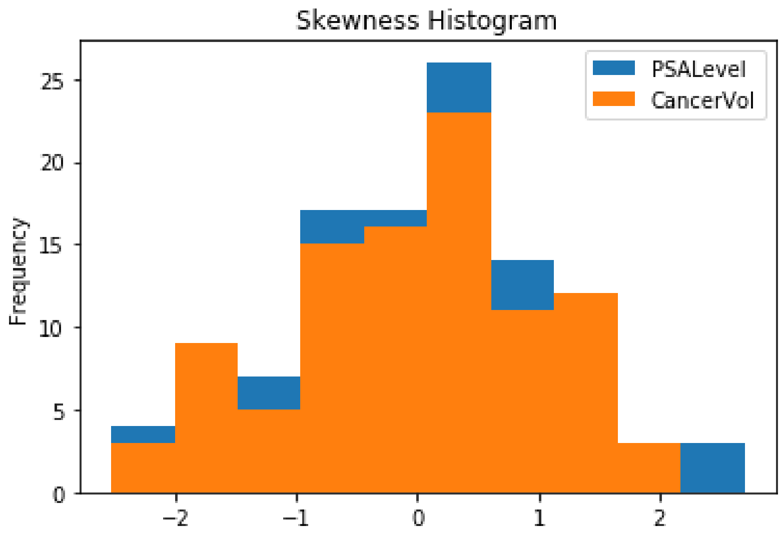
```
       CapsularPenetration
count         9.700000e+01
mean          1.281907e-16
std           1.005195e+00
min          -5.965729e-01
25%          -5.965729e-01
50%          -4.771981e-01
75%           2.680906e-01
max           4.232114e+00
```

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

November 23, 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.head()
```

```
[7]:      Y_HighGradeCancer  PSALevel  CancerVol    Weight       Age  \
     Obs
     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

          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
```

[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 Data Preperation

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

[10]:
```
# 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']
```

[11]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

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

[12]:
```
# 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
```

1.3 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.

[13]:
```
# import function
from sklearn.dummy import DummyClassifier
```

[14]:
```
# 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)
```

[15]:
```
# train model
model_dummy.fit(X_train, y_train)
```

[15]:
```
DummyClassifier(constant=None, random_state=0, strategy='most_frequent')
```

[16]:
```
# 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
```

[17]:
```
# performance metrics
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,␣
↪recall_score
```

[18]:
```
# 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]]
```

1.4 Statsmodels Library

1.4.1 Full Logistics Model

[19]:
```
import statsmodels.api as sm
```

[20]:
```
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)
```

[21]:
```
results = model.fit()
```

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

[22]:
```
### full model statistical output
print(results.summary2(alpha=0.10))
```

```
                          Results: Logit
==================================================================
Model:              Logit            Pseudo R-squared: 0.385
Dependent Variable: Y_HighGradeCancer AIC:              50.6278
Date:               2020-11-23 02:07 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|    [0.05    0.95]
-------------------------------------------------------------------
const          -2.6867    0.6186  -4.3429  0.0000  -3.7042  -1.6691
PSALevel        1.0577    0.6198   1.7067  0.0879   0.0383   2.0772
CancerVol       1.5502    0.6859   2.2599  0.0238   0.4219   2.6784
==================================================================
```

[23]:
```python
PSALevel_list = X_train['PSALevel'].tolist()
CancerVol_list = X_train['CancerVol'].tolist()
Y_HighGradeCancer_list = y_train.tolist()
```

[24]:
```python
type(np.arange(1, 2, 0.5))
```

[24]:
```
numpy.ndarray
```

1.5 Advanced Visualizations Using Matplotlib

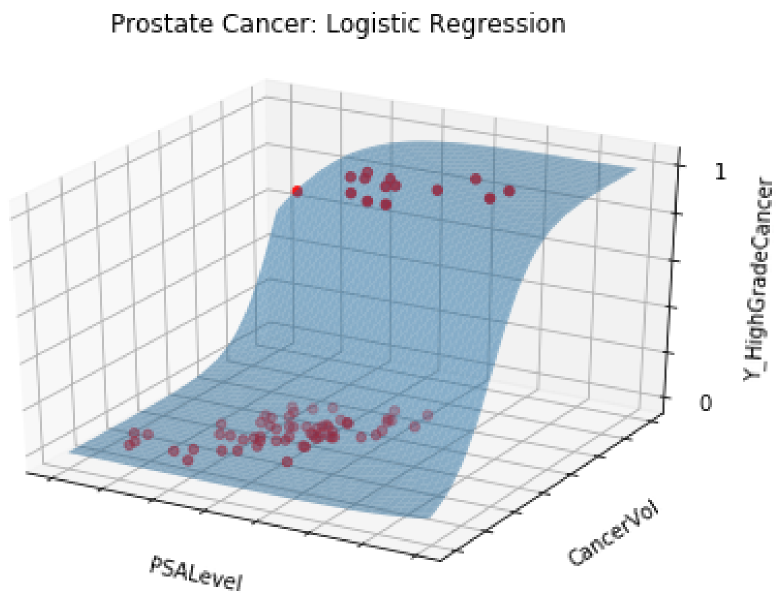
[25]:
```python
import math
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

1.5.1 Logistic Regression Plot

[26]:
```python
%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')
```

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


Prostate Cancer: Logistic Regression
Y_HighGradeCancer
PSALevel
CancerVol