

## 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

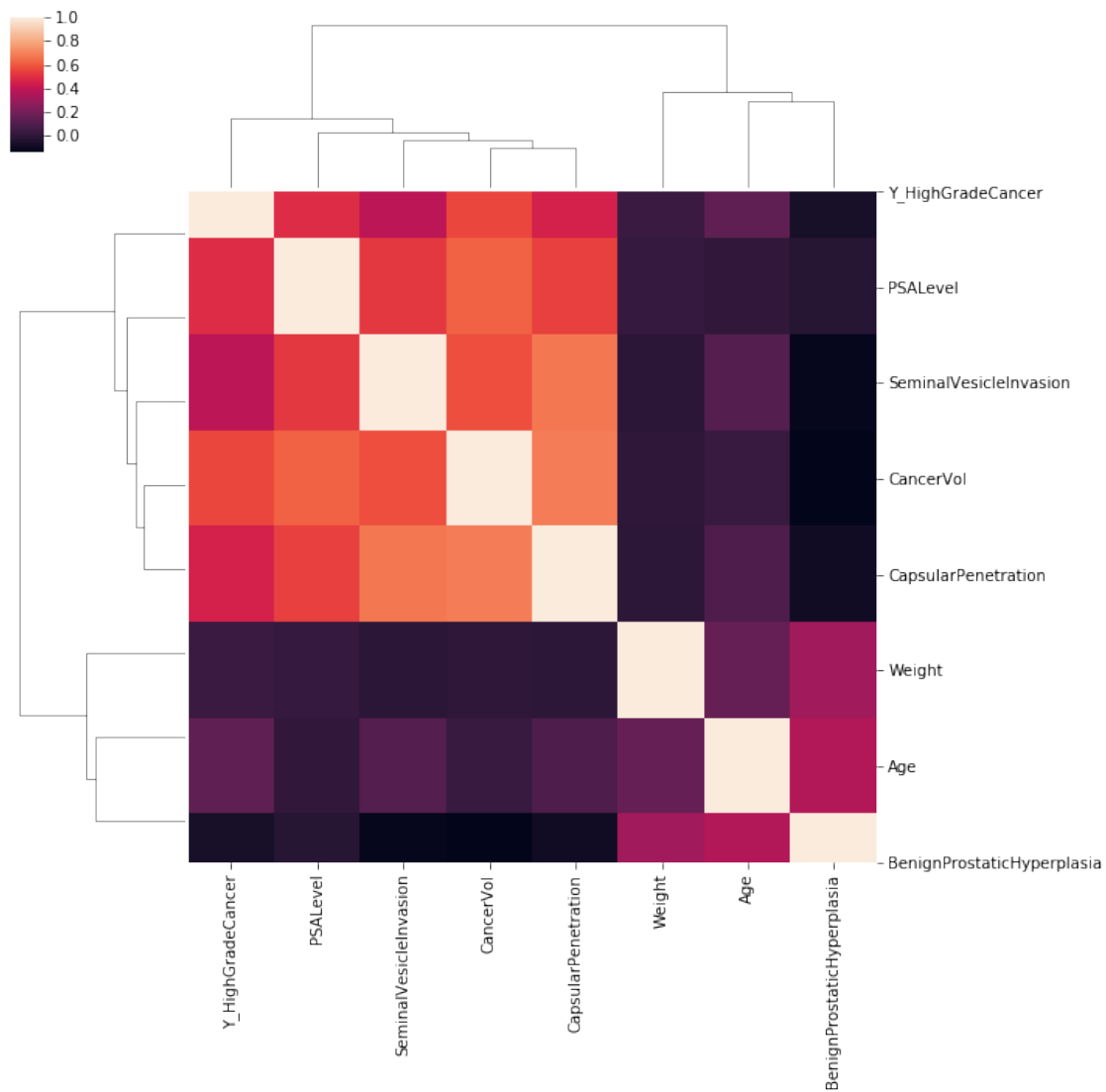
	BenignProstaticHyperplasia \
PSALevel	-0.016486
CancerVol	-0.133209
Weight	0.321849
Age	0.366341
BenignProstaticHyperplasia	1.000000
SeminalVesicleInvasion	-0.119553
CapsularPenetration	-0.083009
Y_HighGradeCancer	-0.058032

	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

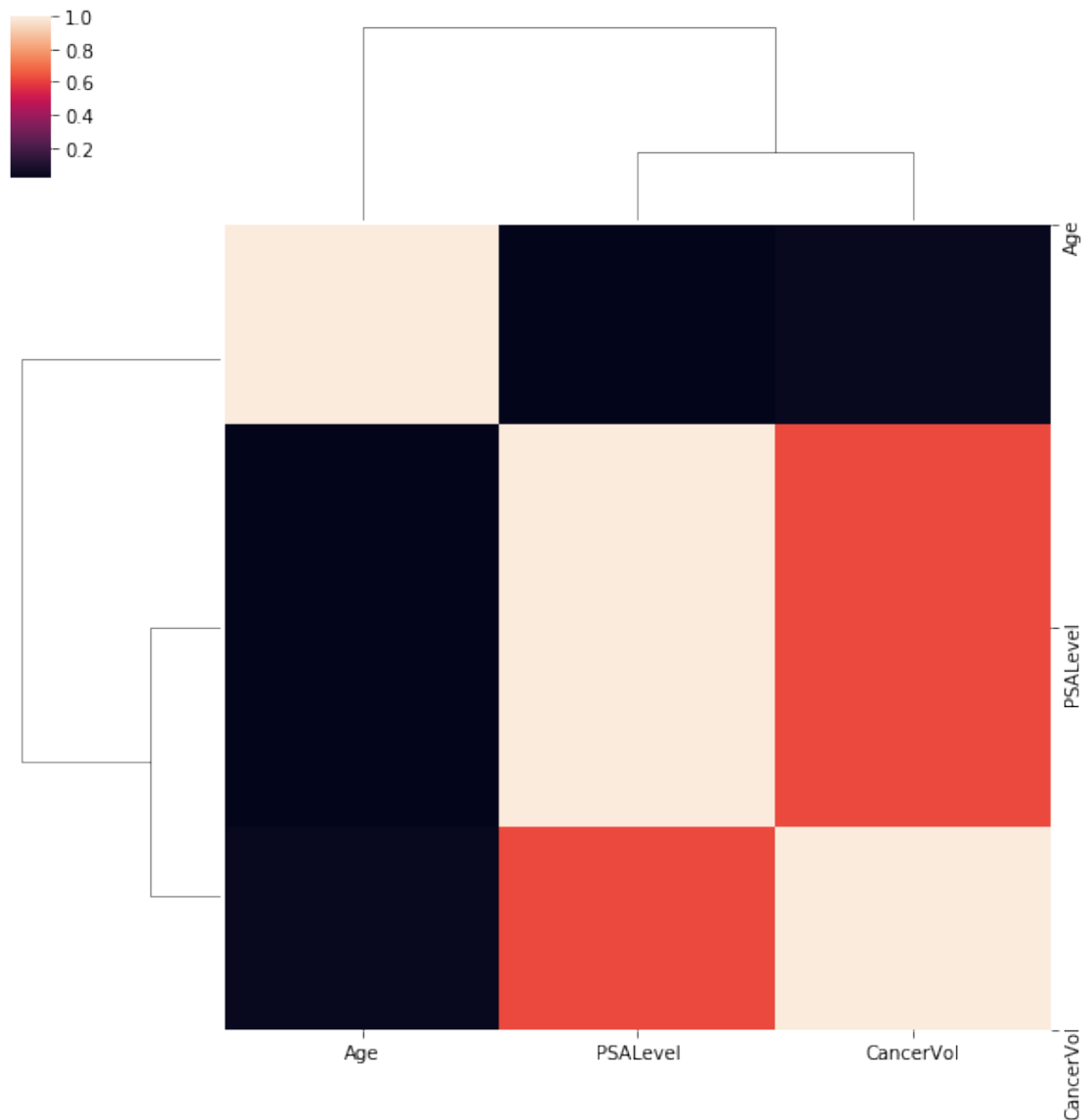
	Y_HighGradeCancer
PSALevel	0.497189
CancerVol	0.564645
Weight	0.039445
Age	0.148074
BenignProstaticHyperplasia	-0.058032
SeminalVesicleInvasion	0.392231
CapsularPenetration	0.463134
Y_HighGradeCancer	1.000000

```
[14]: # let's import seaborn 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

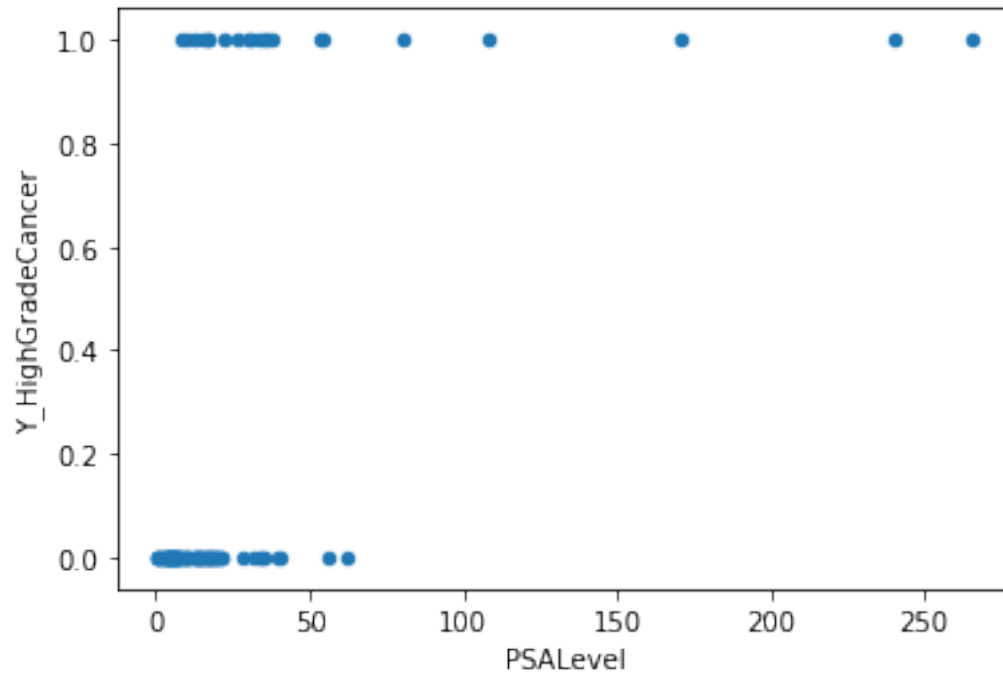
```
[17]: # numerical features
# centrality measures
print(f'Mean Age: {round(df.Age.mean(), 2)}')
```

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

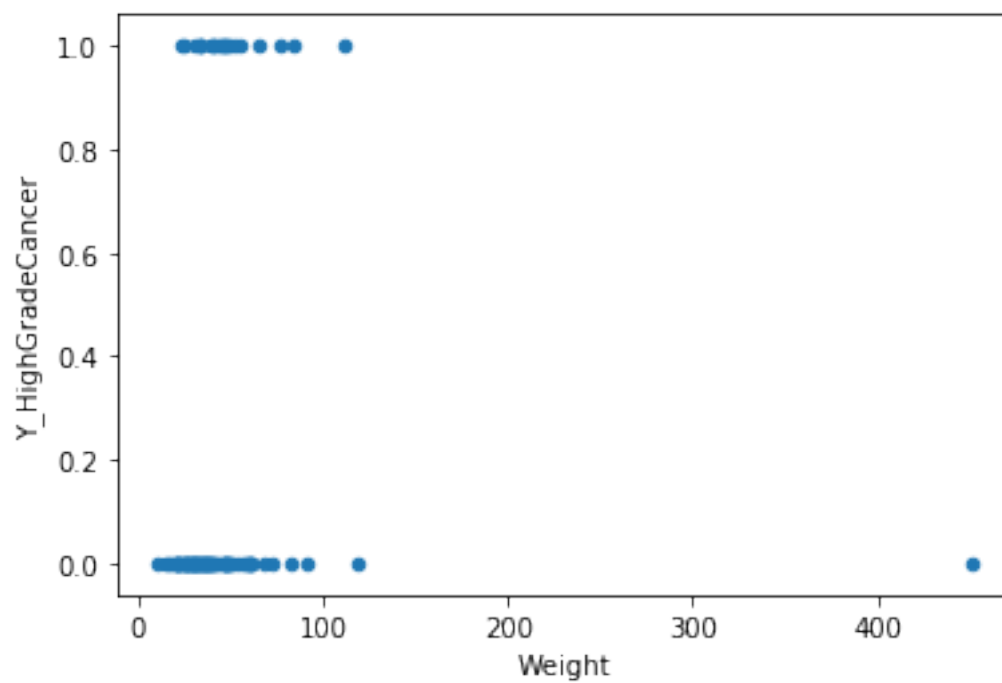
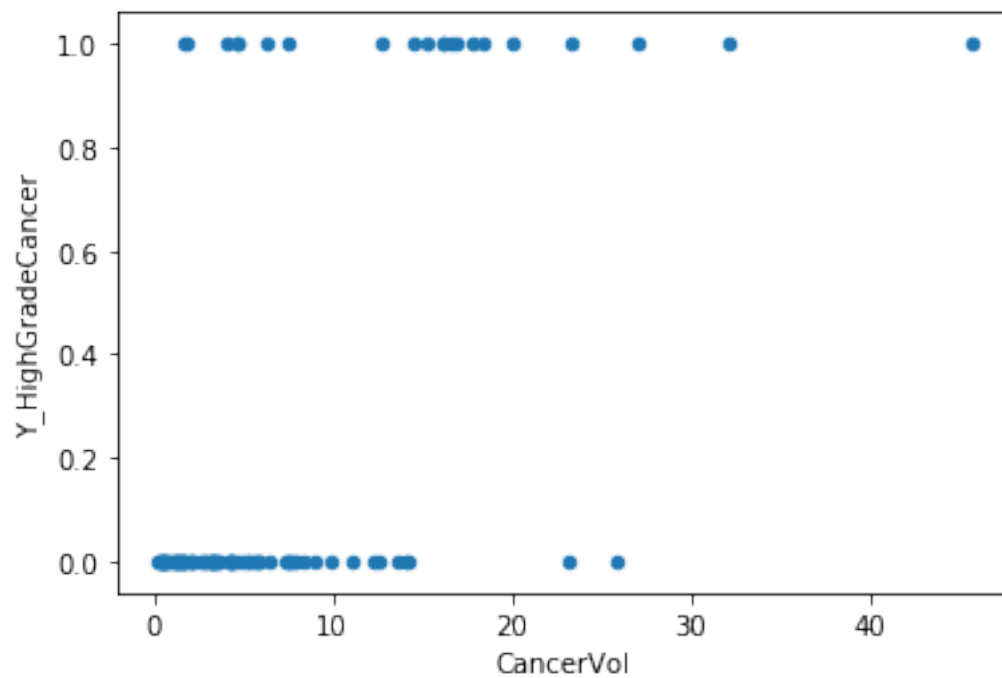
Mean Age: 63.87

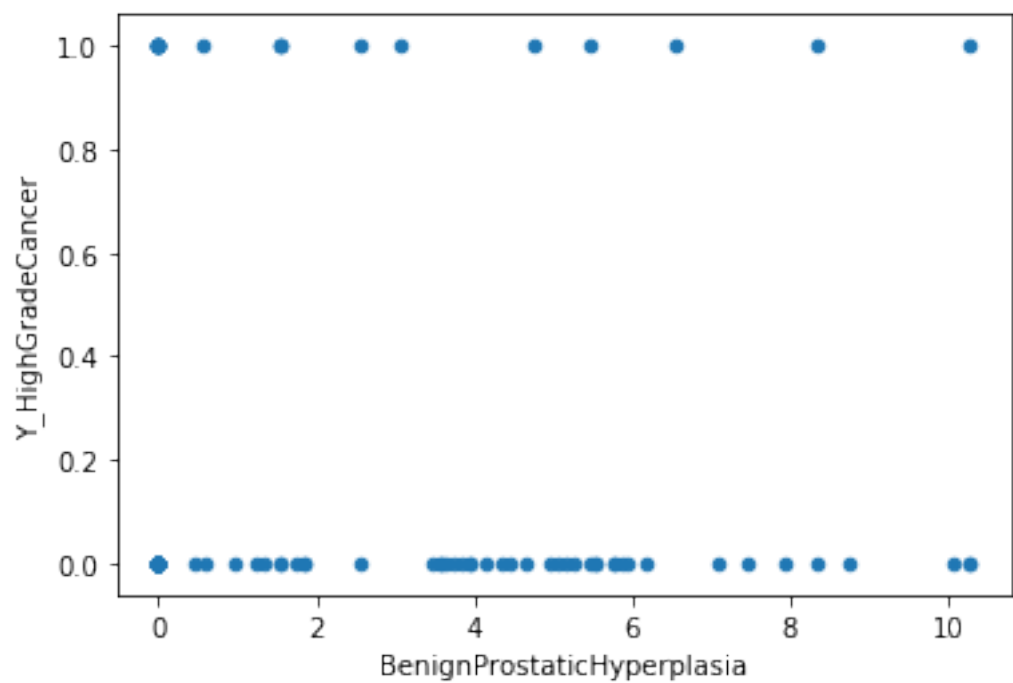
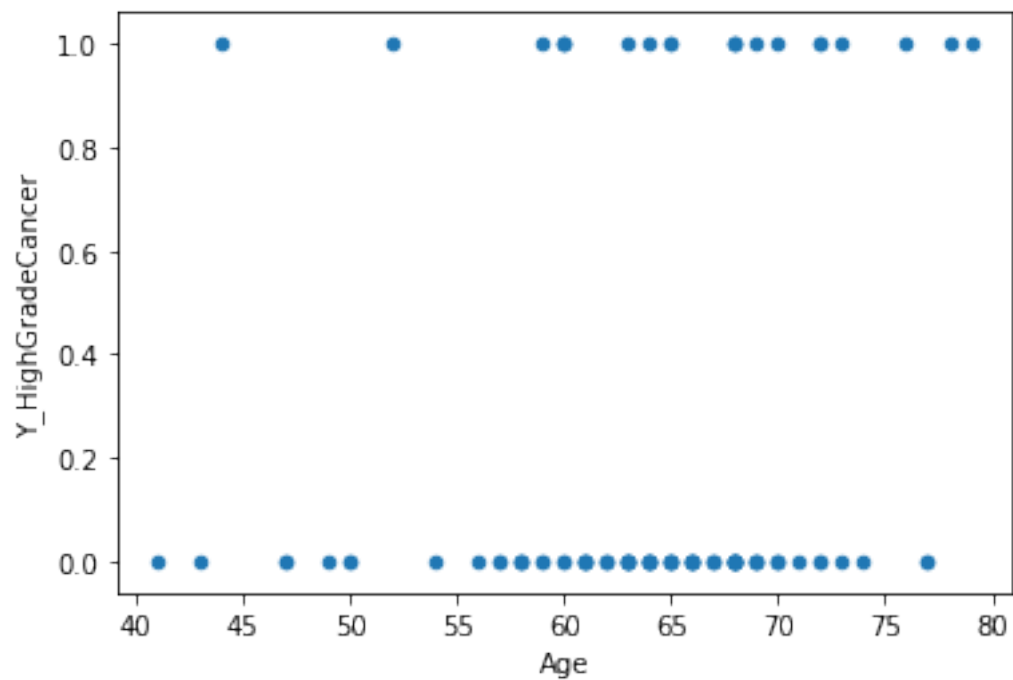
Median Age: 65.0

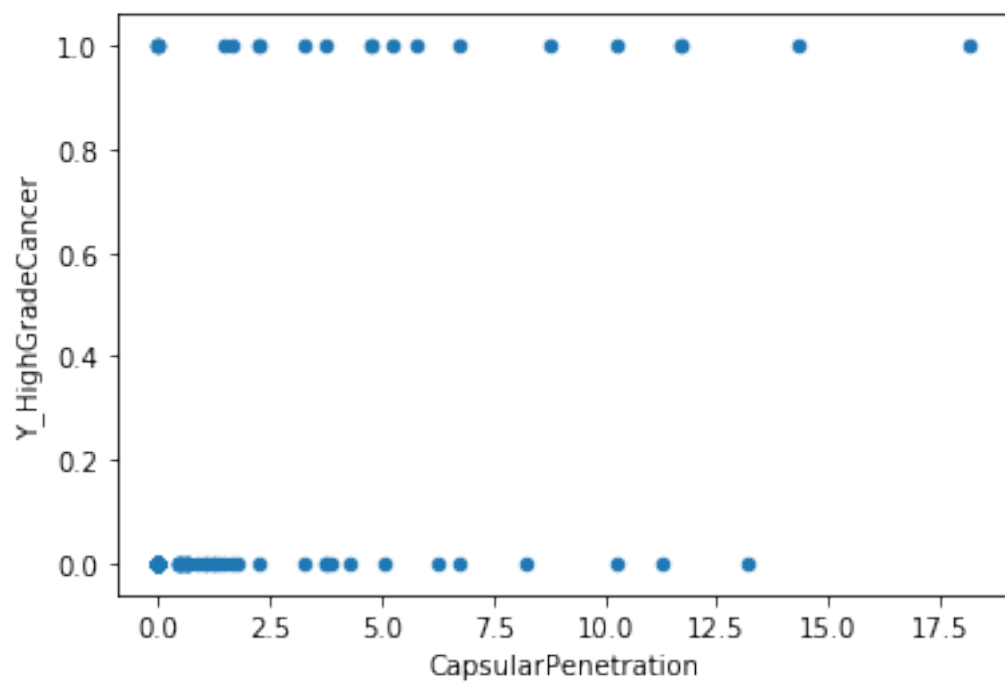
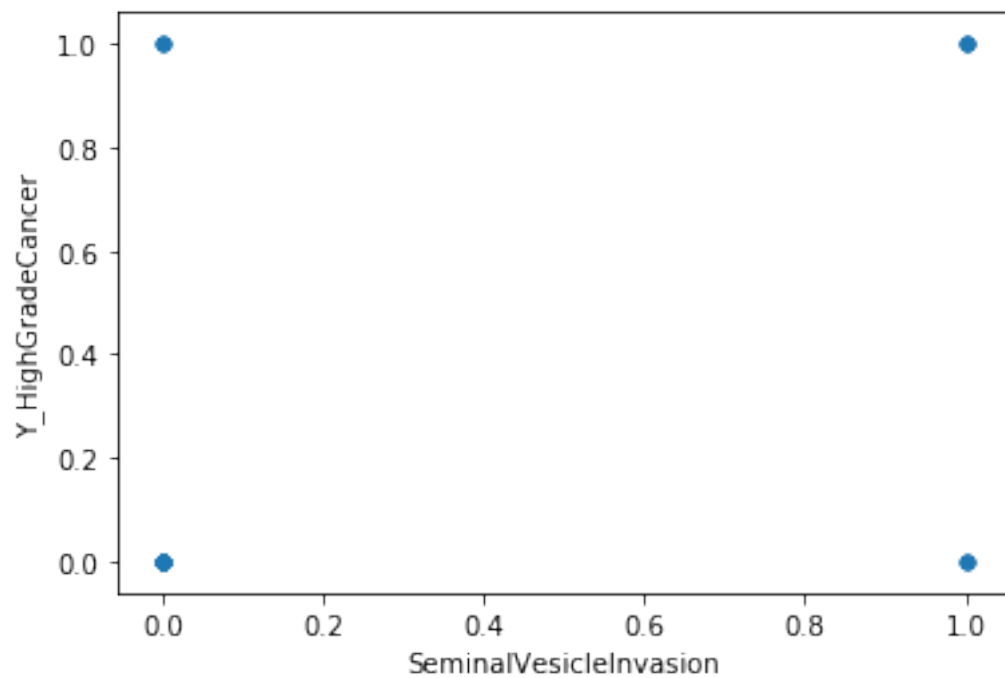
```
[18]: # print all relevant scatter plots
# examine visuals for outliers
for col in cols:
    df[['Y_HighGradeCancer', col]].plot.scatter(x=col, y='Y_HighGradeCancer');
```

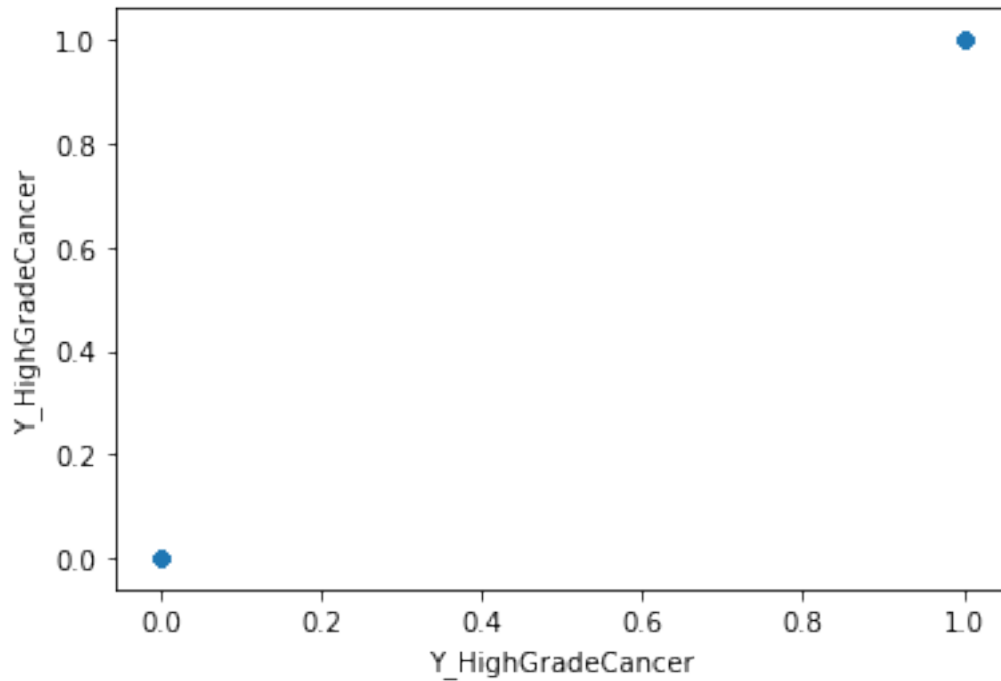










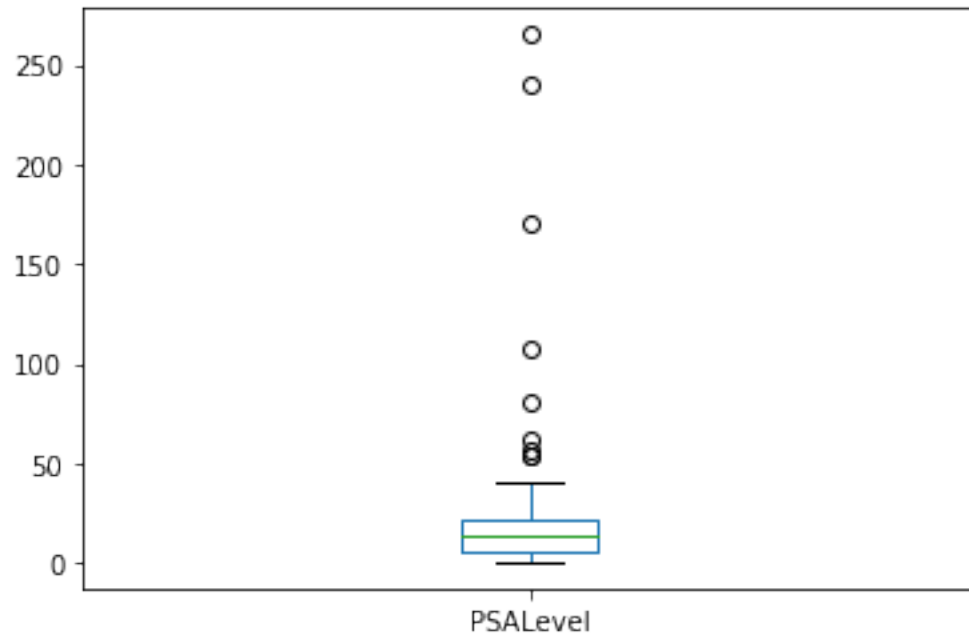


- PSALevel and CancerVol may be good predictors for Y\_HighGradeCancer in a Logistic Regression model.
- Weight appears to contain an extreme outlier.
- CancerVol may present two outliers, when Y\_HighGradeCancer = 0.
- Via a priori knowledge, Age may be a valuable predictor of Prostate Cancer, so I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

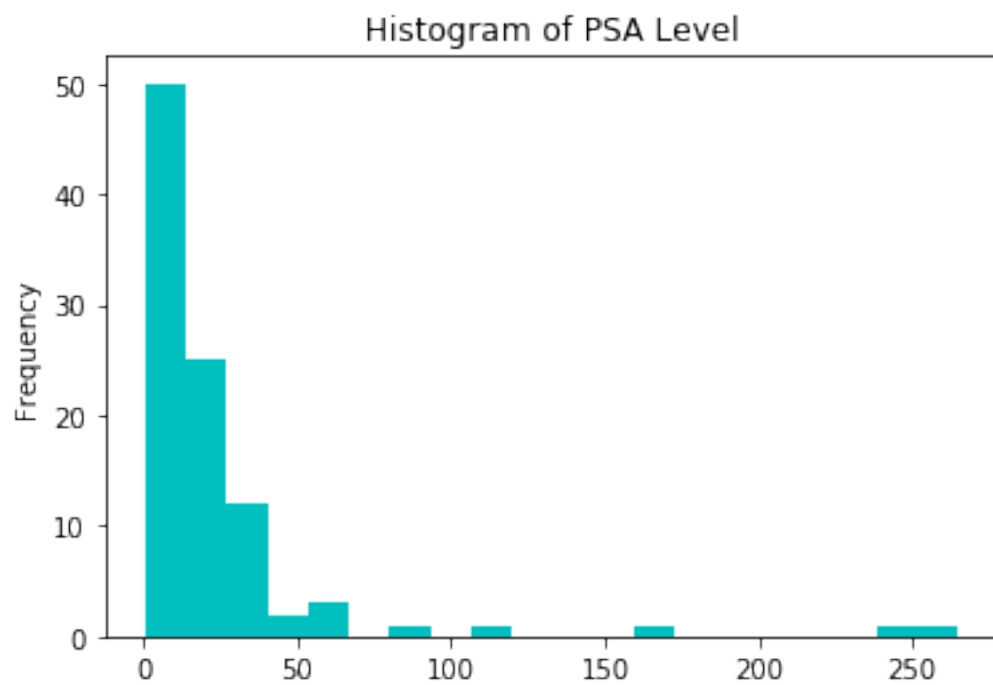
## 1.4 Distributions

### 1.4.1 PSALevel

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



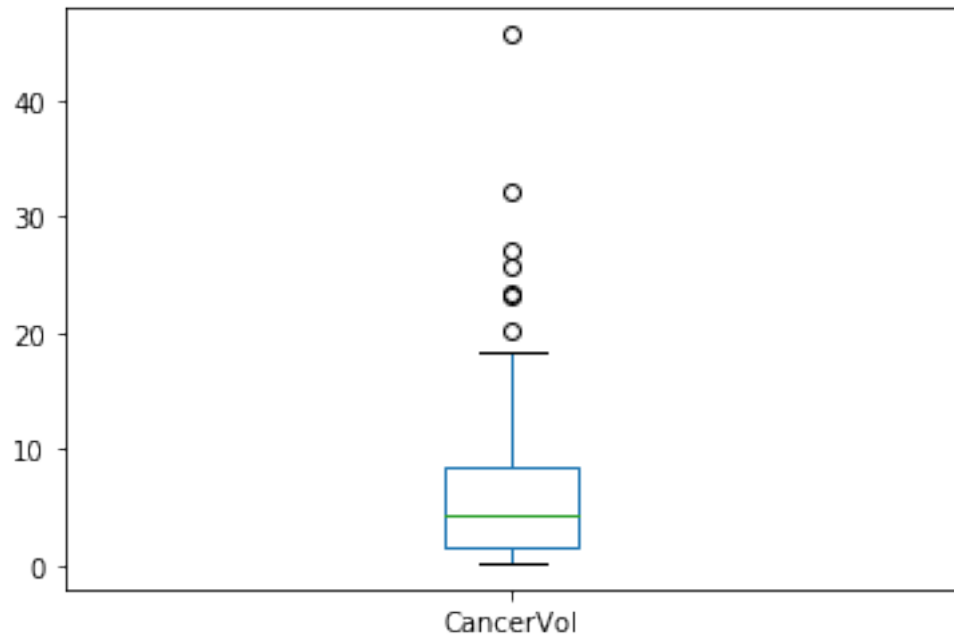
```
[20]: # use hist to create histogram
df.PSALevel.plot(kind='hist', title='Histogram of PSA Level', color='c',
    ↪bins=20);
```



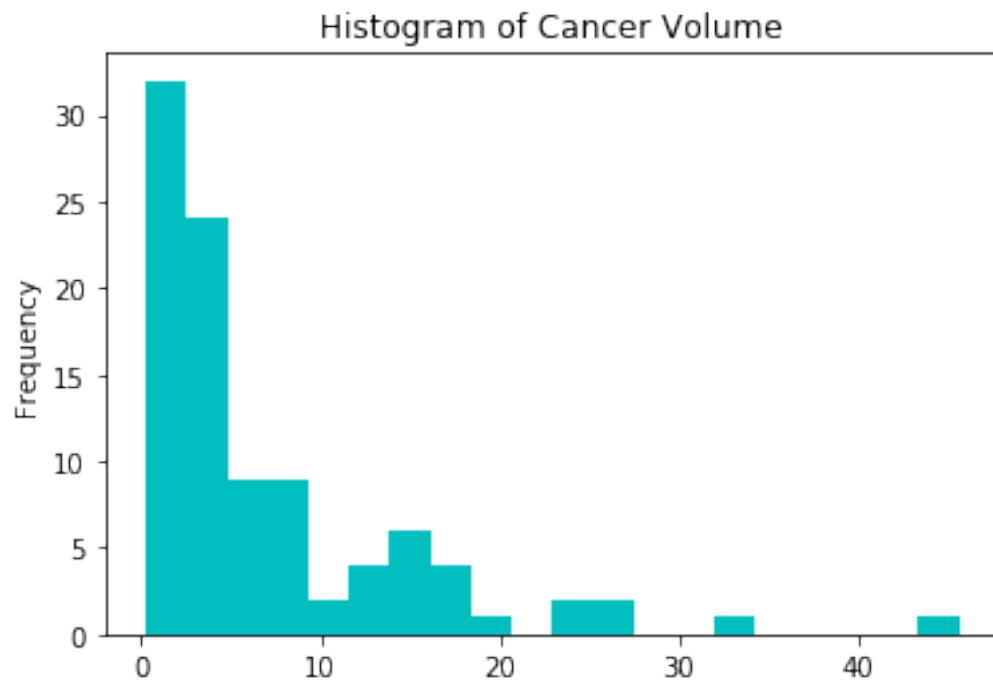
- PSALevel shows high positive skewness.

#### 1.4.2 CancerVol

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



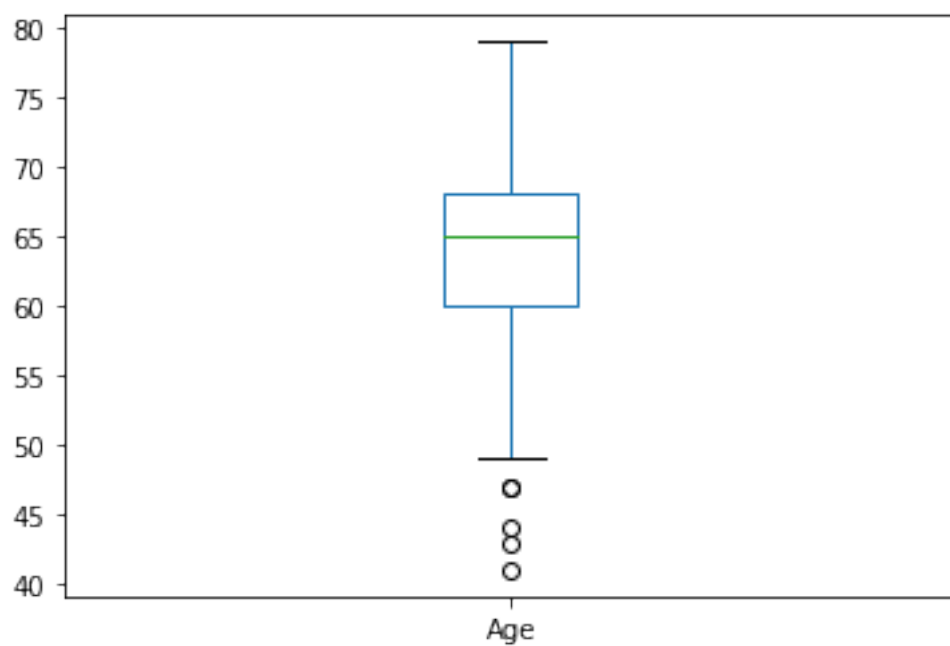
```
[22]: df.CancerVol.plot(kind='hist', title='Histogram of Cancer Volume', color='c',  
    ↪ bins=20);
```



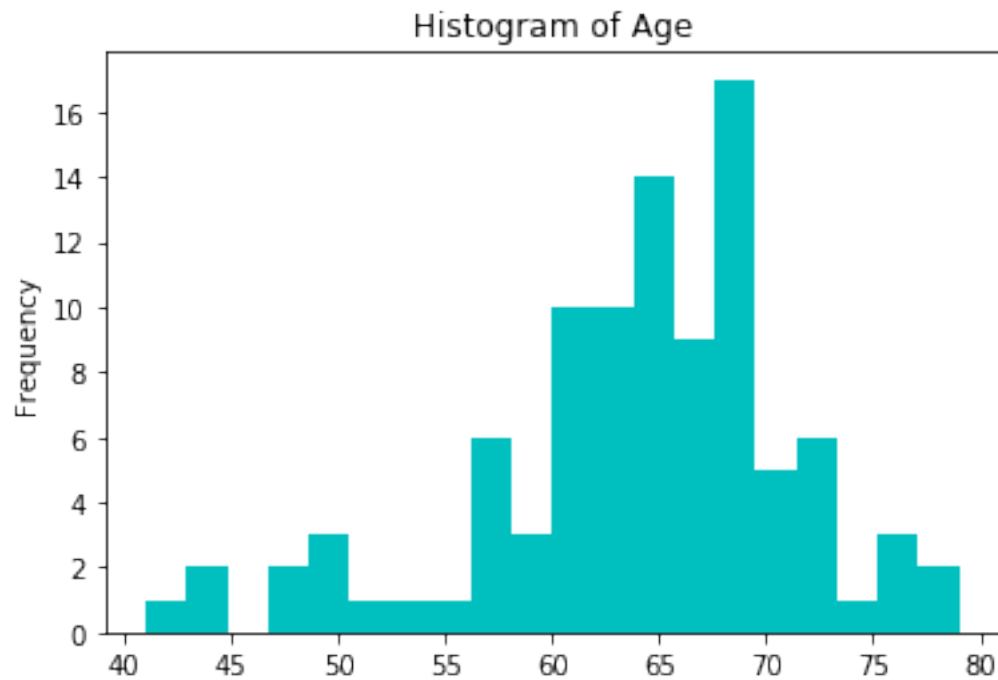
- CancerVol shows high positive skewness.

#### 1.4.3 Age

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



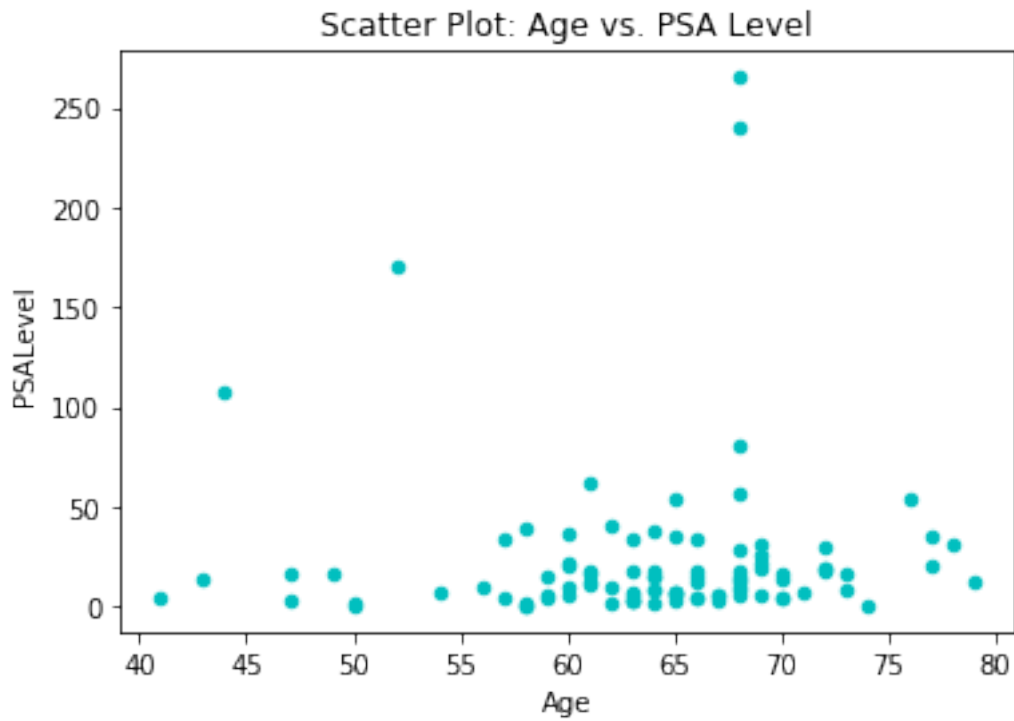
```
[24]: df.Age.plot(kind='hist', title='Histogram of Age', color='c', bins=20);
```



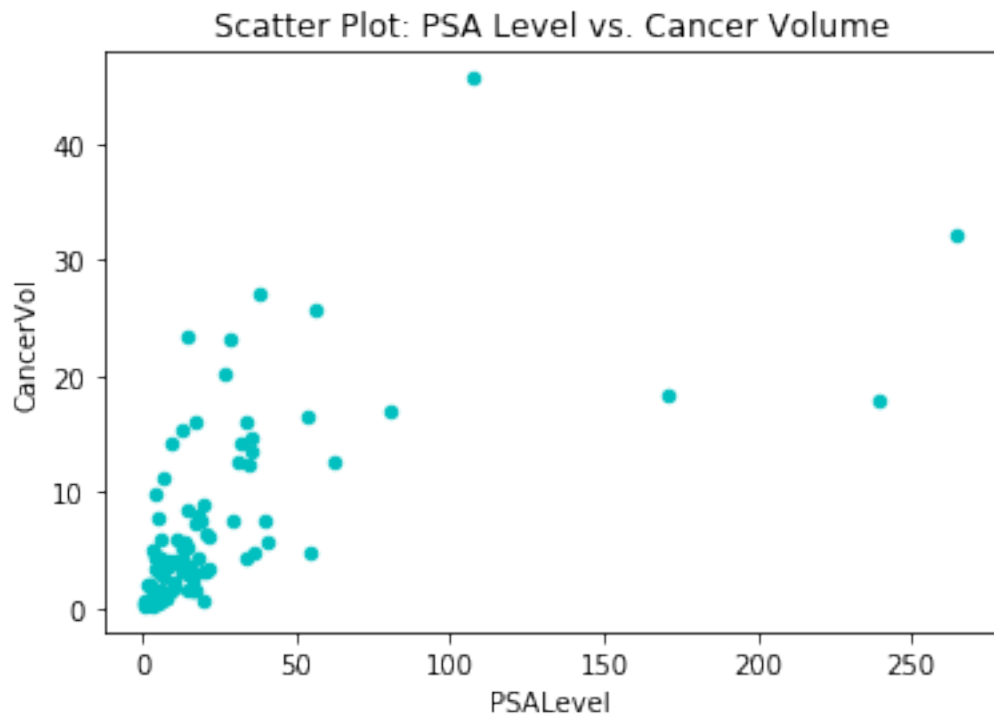
#### 1.4.4 Bi-variate Interactions

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



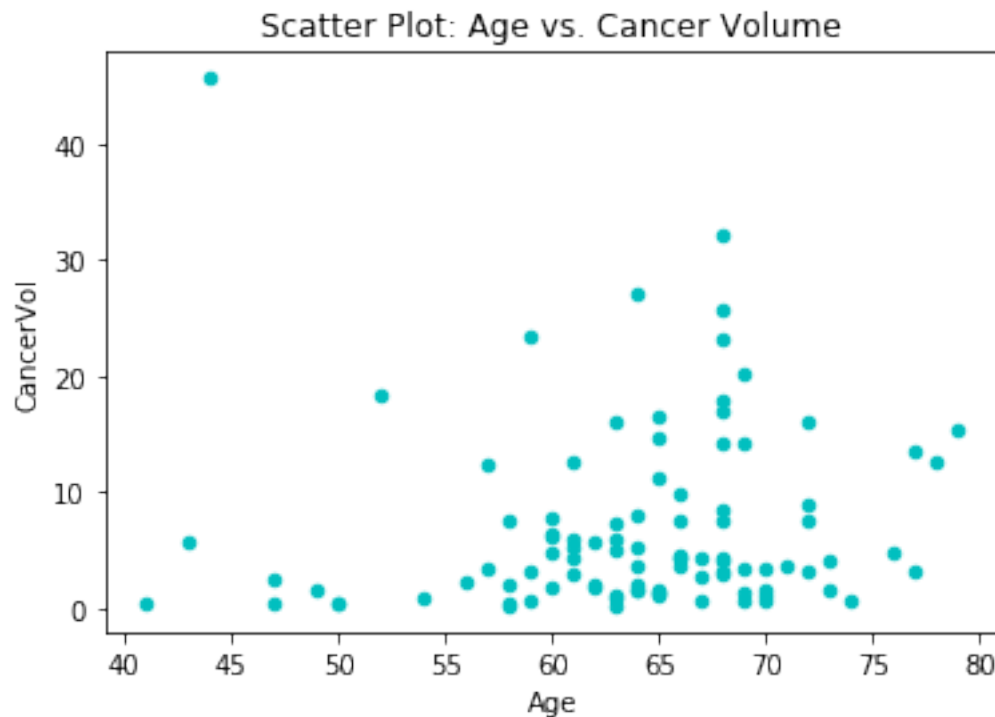


```
[26]: df.plot.scatter(x='PSALevel', y='CancerVol', color='c', title='Scatter Plot: ↪
      ↪PSA Level vs. Cancer Volume');
```

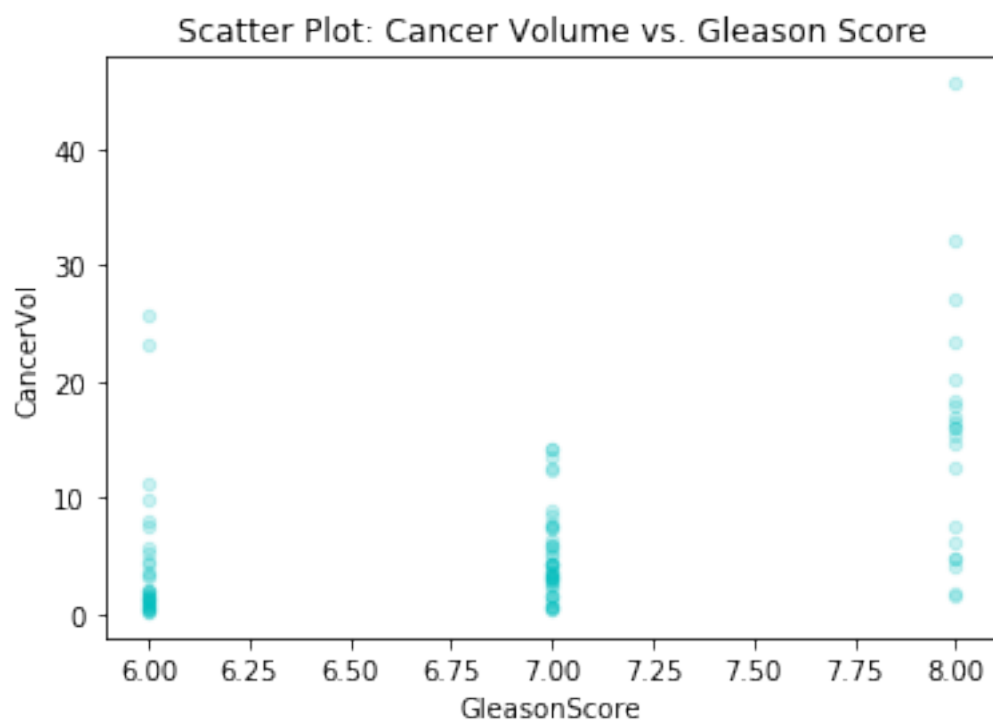
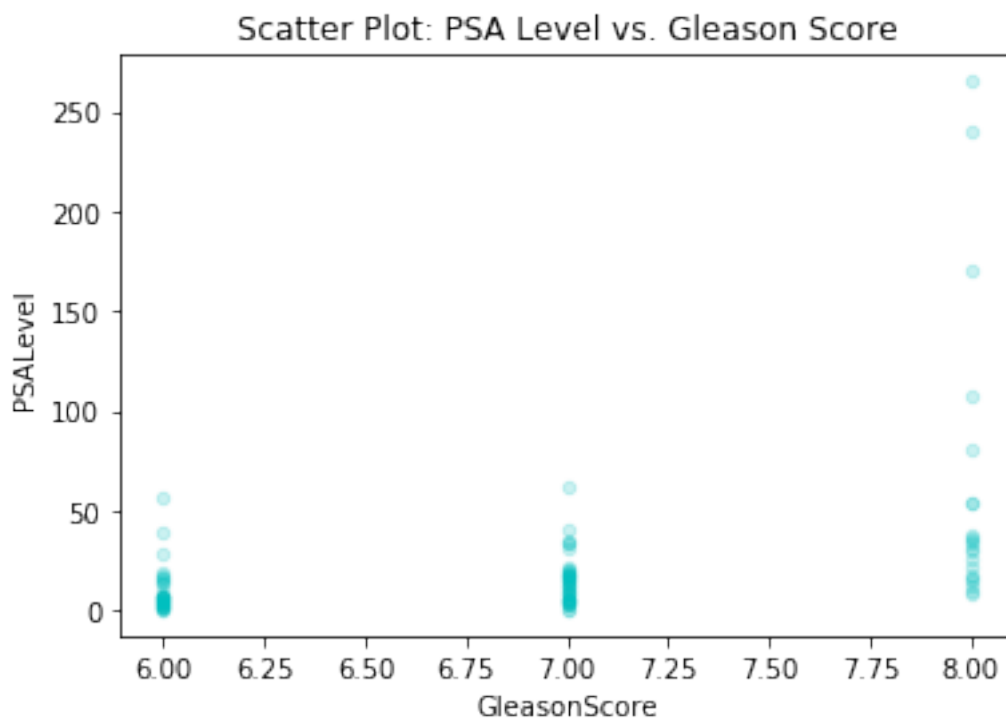


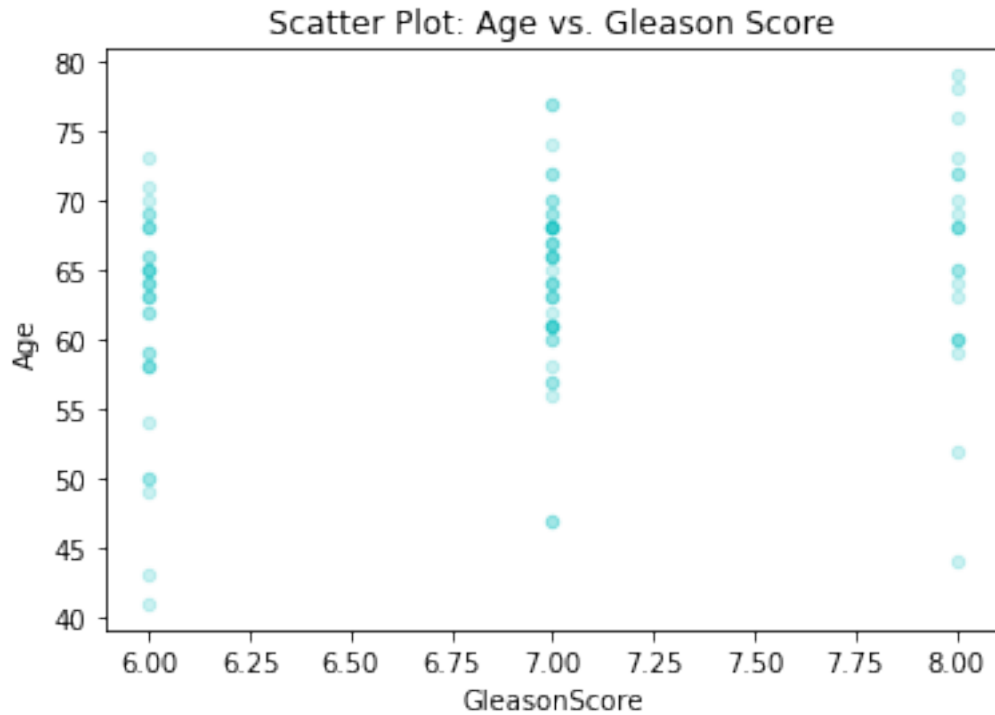
- PSA Level & Cancer Volume display a mild level of correlation

```
[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');
```





- 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

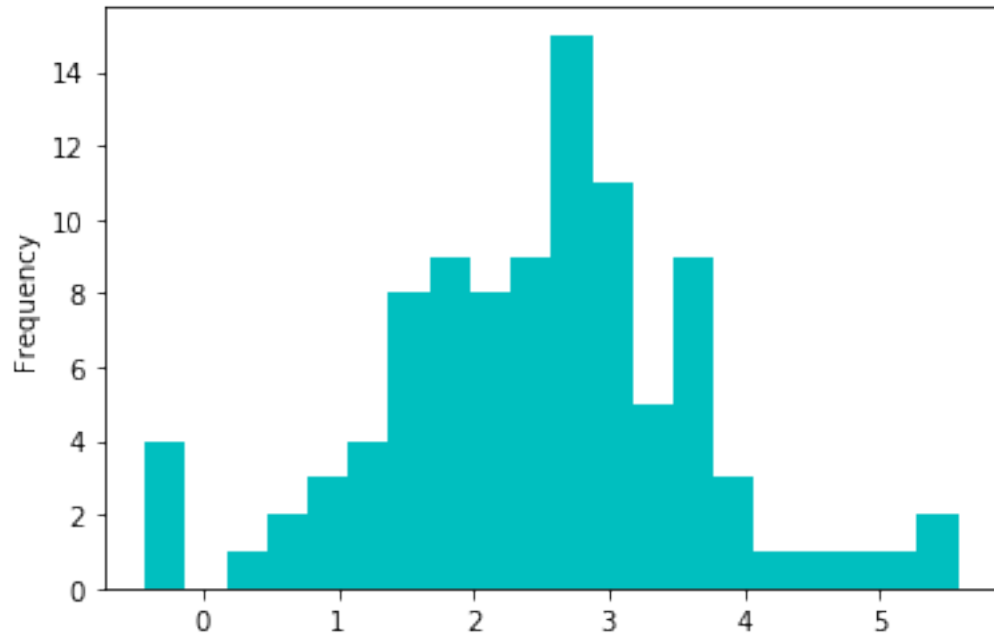
- This section is for investigation/analysis purposes only. I may or may not include transformations in the finalized processed dataset.
- Considering only PSALevel, CancerVol, and Weight at this time.

```
[33]: # try log transformations to reduce skewness

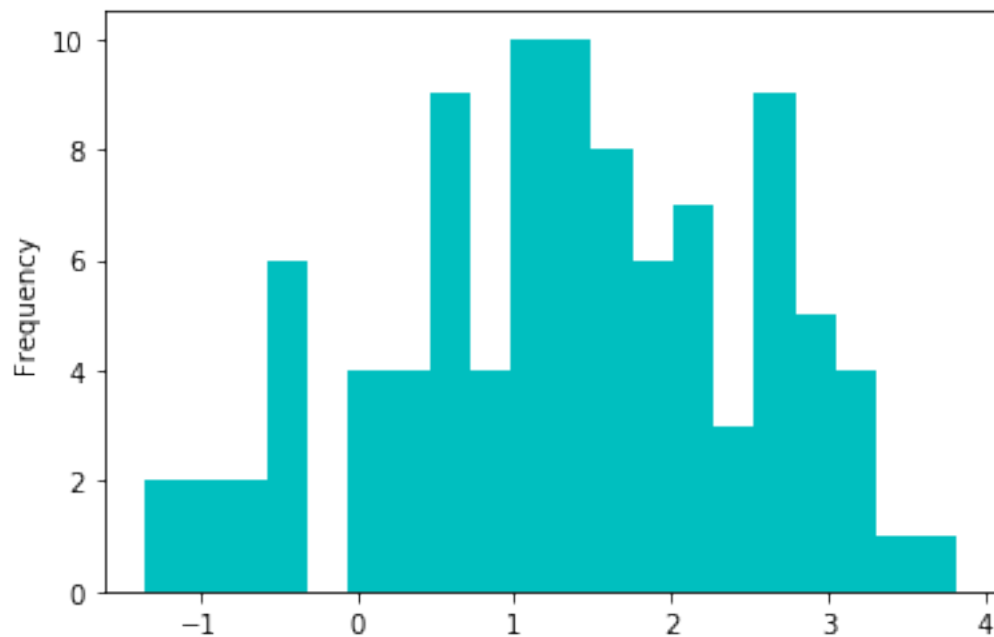
log_PSALevel = np.log(df.PSALevel)
log_CancerVol = np.log(df.CancerVol)
```

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

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

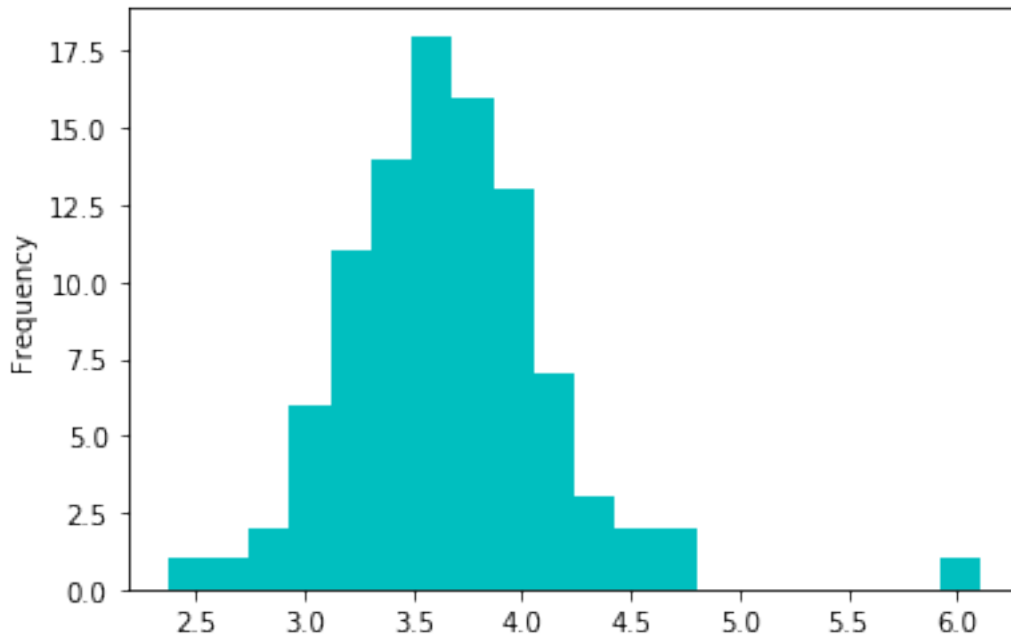


```
[35]: # histogram of log CancerVol  
log_CancerVol.plot(kind='hist', color='c', bins=20);
```





```
[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 Transformed 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 Transformed 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 during 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
2	0.0		0.0
3	0.0		0.0
4	0.0		0.0
5	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 include 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

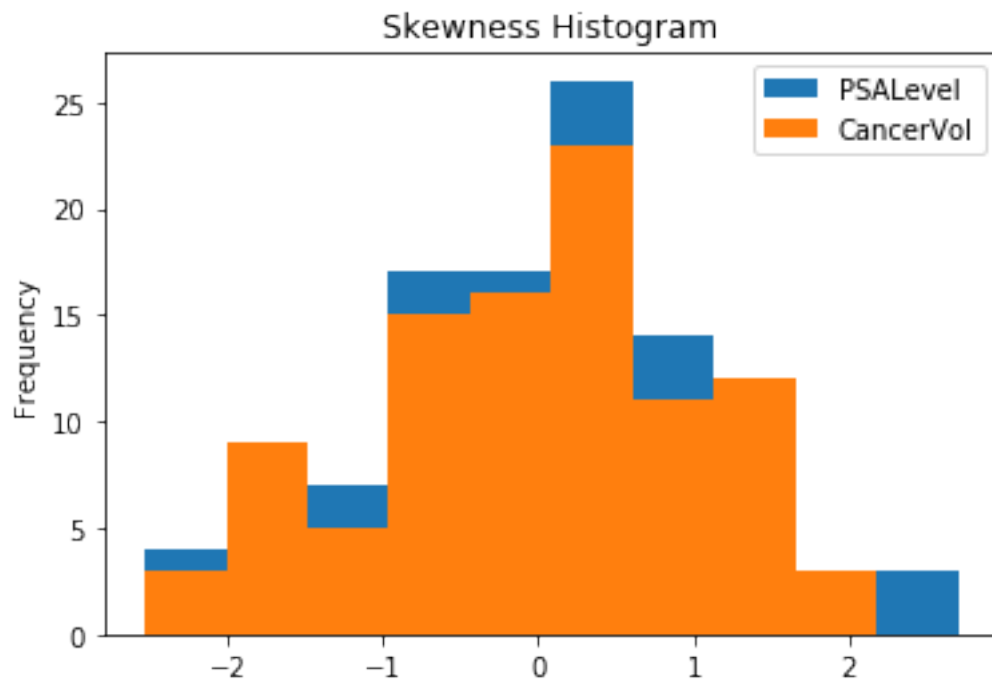
	CapsularPenetration
count	9.700000e+01
mean	1.281907e-16
std	1.005195e+00
min	-5.965729e-01
25%	-5.965729e-01
50%	-4.771981e-01
75%	2.680906e-01
max	4.232114e+00

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 features
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)
```