Data Description The Haberman's survival dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer.

Attribute Information:

Age of patient at time of operation (numerical) Patient's year of operation (year - 1900, numerical) Number of positive axillary nodes detected (numerical) Survival status (class attribute) 1 = the patient survived 5 years or longer 2 = the patient died within 5 years

1. Environment Configuration

In [1]:
```
# import necessary packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

In [2]:
```
# load the dataset
hamber = pd.read_csv('haberman.csv', header=None, names=['age', 'year_of_treatment', 'positive_lymph_nodes', 'survival_status_after_5_years'])
print(hamber.head())
```

```
   age  year_of_treatment  positive_lymph_nodes  survival_status_after_5_years
0   30                 64                     1                              1
1   30                 62                     3                              1
2   30                 65                     0                              1
3   31                 59                     2                              1
4   31                 65                     4                              1
```

2. Data Preparation

In [3]:
```
# (Q) how many data-points and features?
print (hamber.shape)
```

```
(306, 4)
```

In [4]:
```
#(Q) What are the column names in our dataset?
print (hamber.columns)
```

```
Index(['age', 'year_of_treatment', 'positive_lymph_nodes',
       'survival_status_after_5_years'],
      dtype='object')
```

In [5]:
```
#(Q) How many data points for each class are present?
#(or) How many cases for each status are present?

hamber["survival_status_after_5_years"].value_counts()
# balanced-dataset vs imbalanced datasets
#Hamber is an imbalanced dataset as the number of data points for each class is not equal.
```

Out[5]:
```
1    225
2     81
Name: survival_status_after_5_years, dtype: int64
```

In [6]:
```
print(hamber.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306 entries, 0 to 305
Data columns (total 4 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   age                             306 non-null    int64
 1   year_of_treatment               306 non-null    int64
 2   positive_lymph_nodes            306 non-null    int64
 3   survival_status_after_5_years   306 non-null    int64
dtypes: int64(4)
memory usage: 9.7 KB
None
```

Observations:

There are no missing values in this dataset. So there is no need to do data imputation. The datatype of 'survival_status_after_5_years' column is integer and the values of the column is not interpretable. Hence they should be mapped to 'yes' (survived after 5 years) and 'no' (not survived after 5 years)

In [7]:
```
# print the unique values of the target column
print(list(hamber['survival_status_after_5_years'].unique()))

# modify the target column values to be meaningful as well as categorical
hamber['survival_status_after_5_years'] = hamber['survival_status_after_5_years'].map({1:"yes", 2:"no"})
hamber['survival_status_after_5_years'] = hamber['survival_status_after_5_years'].astype('category')
print(hamber.head())
```

```
[1, 2]
   age  year_of_treatment  positive_lymph_nodes survival_status_after_5_years
0   30                 64                     1                           yes
1   30                 62                     3                           yes
2   30                 65                     0                           yes
3   31                 59                     2                           yes
4   31                 65                     4                           yes
```

3. High Level Statistics

In [8]:
```
print(hamber.describe())
```

```
              age  year_of_treatment  positive_lymph_nodes
count  306.000000         306.000000            306.000000
mean    52.457516          62.852941              4.026144
std     10.803452           3.249405              7.189654
min     30.000000          58.000000              0.000000
25%     44.000000          60.000000              0.000000
50%     52.000000          63.000000              1.000000
75%     60.750000          65.750000              4.000000
max     83.000000          69.000000             52.000000
```

In [9]:
```
print("Number of rows: " + str(hamber.shape[0]))
print("Number of columns: " + str(hamber.shape[1]))
print("Columns: " + ", ".join(hamber.columns))

print("Target variable distribution")
print(hamber.iloc[:,-1].value_counts())
print("*"*50)
print(hamber.iloc[:,-1].value_counts(normalize = True))
```

```
Number of rows: 306
Number of columns: 4
Columns: age, year_of_treatment, positive_lymph_nodes, survival_status_after_5
_years
Target variable distribution
yes    225
no      81
Name: survival_status_after_5_years, dtype: int64
**************************************************
yes    0.735294
no     0.264706
Name: survival_status_after_5_years, dtype: float64
```

Observations:

- The age of the patients vary from 30 to 83 with the median of 52.
- Although the maximum number of positive lymph nodes observed is 52, nearly 75% of the patients have less than 5 positive lymph nodes and nearly 25% of the patients have no positive lymph nodes
- The dataset contains only a small number of records (306).
- The target column is imbalanced with 73% of values are 'yes'

4. Objective

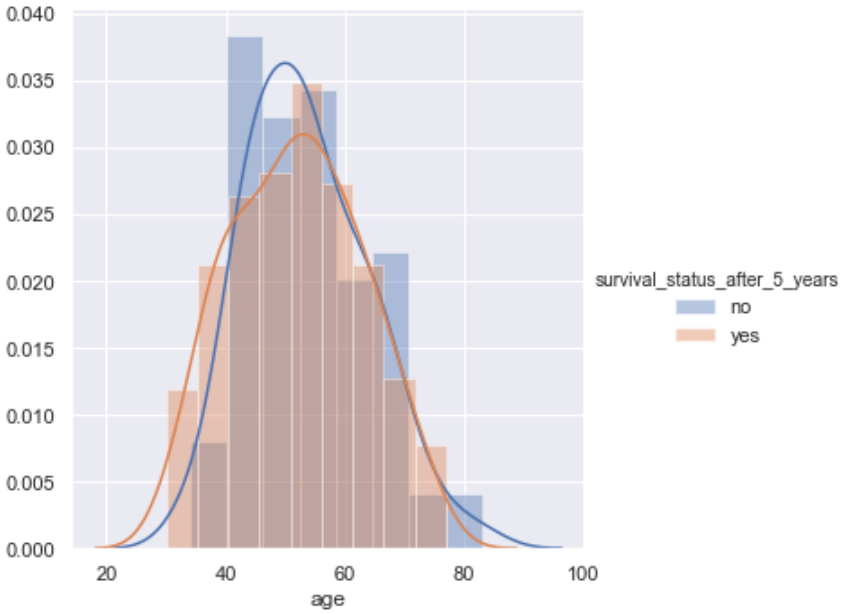
- To predict whether the patient will survive after 5 years or not based upon the patient's age, year of treatment and the number of positive lymph nodes

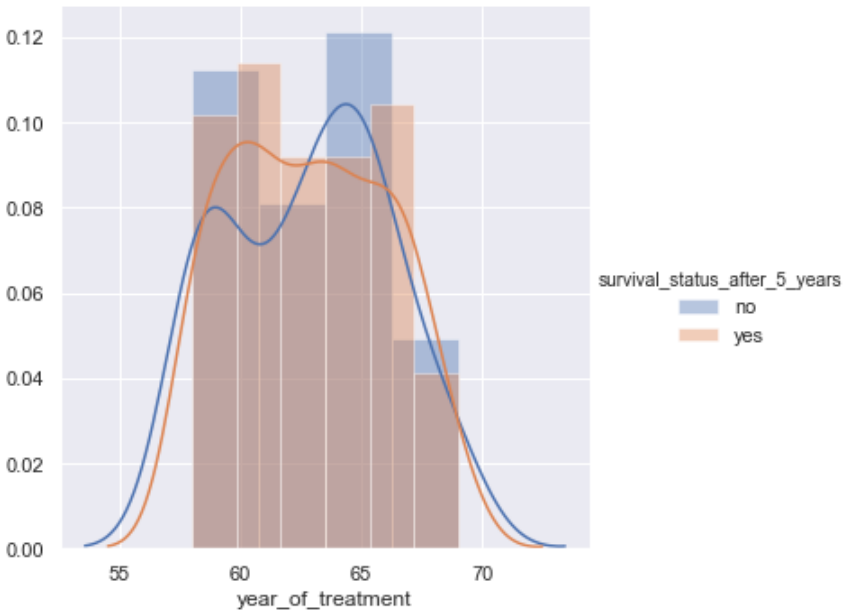
5. Univariate Analysis

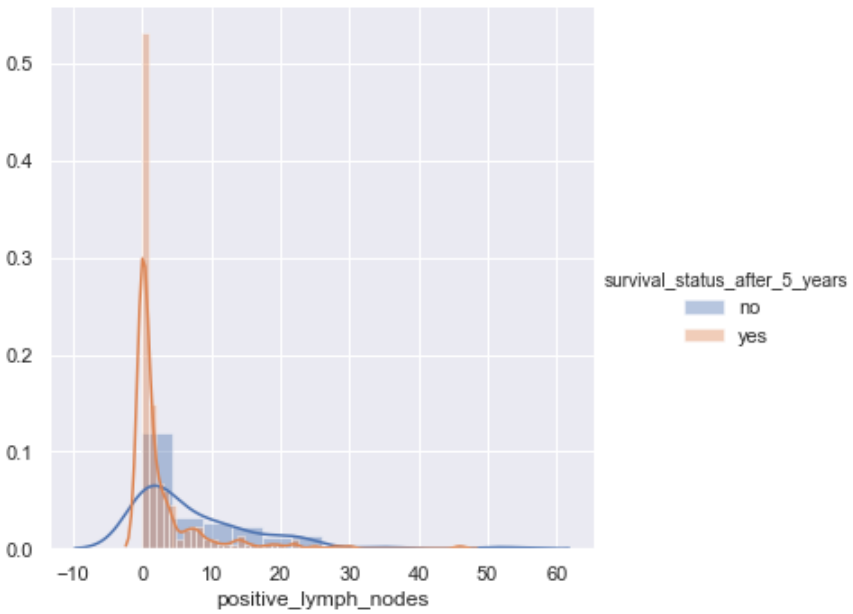
5.1 Distribution plots

In [10]:

```
"""
* Distribution plots are used to visually assess how the data points are distri
buted with respect to its frequency.
* Usually the data points are grouped into bins and the height of the bars repr
esenting each group increases with increase in the number of data points
lie within that group. (histogram)
* Probality Density Function (PDF) is the probabilty that the variable takes a
 value x. (smoothed version of the histogram)
* Kernel Density Estimate (KDE) is the way to estimate the PDF. The area under
 the KDE curve is 1.
* Here the height of the bar denotes the percentage of data points under the co
rresponding group
"""
for idx, feature in enumerate(list(hamber.columns)[:-1]):
    fg = sns.FacetGrid(hamber, hue='survival_status_after_5_years', height=5)
    fg.map(sns.distplot, feature).add_legend()
    plt.show()
```


0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
age
survival_status_after_5_years
no
yes

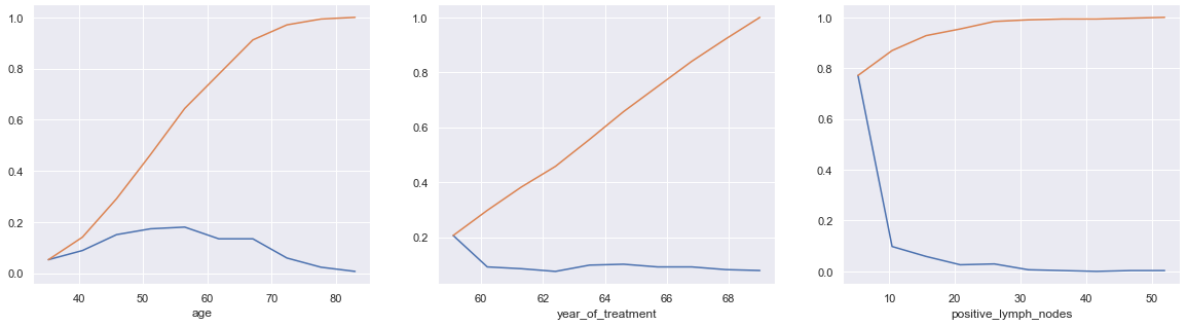

0.12
0.10
0.08
0.06
0.04
0.02
0.00
55
60
65
70
year_of_treatment
survival_status_after_5_years
no
yes


0.5
0.4
0.3
0.2
0.1
0.0
−10
0
10
20
30
40
50
60
positive_lymph_nodes
survival_status_after_5_years
no
yes

5.2 CDF

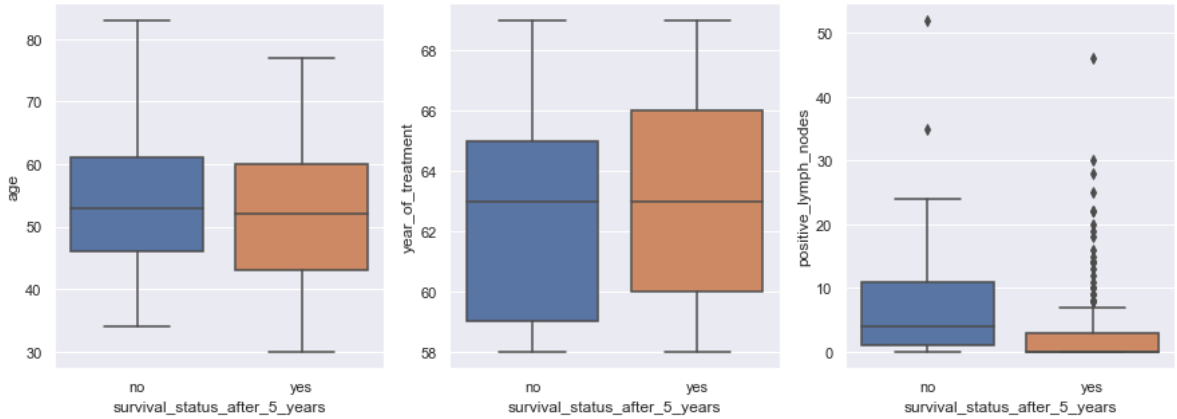
In [11]:
```
"""
The cumulative distribution function (cdf) is the probability that the variable
takes a value less than or equal to x.
"""
plt.figure(figsize=(20,5))
for idx, feature in enumerate(list(hamber.columns)[:-1]):
    plt.subplot(1, 3, idx+1)
    print("********* "+feature+" *********")
    counts, bin_edges = np.histogram(hamber[feature], bins=10, density=True)
    print("Bin Edges: {}".format(bin_edges))
    pdf = counts/sum(counts)
    print("PDF: {}".format(pdf))
    cdf = np.cumsum(pdf)
    print("CDF: {}".format(cdf))
    plt.plot(bin_edges[1:], pdf, bin_edges[1:], cdf)
    plt.xlabel(feature)
```

```
********* age *********
Bin Edges: [30.  35.3 40.6 45.9 51.2 56.5 61.8 67.1 72.4 77.7 83. ]
PDF: [0.05228758 0.08823529 0.1503268  0.17320261 0.17973856 0.13398693
 0.13398693 0.05882353 0.02287582 0.00653595]
CDF: [0.05228758 0.14052288 0.29084967 0.46405229 0.64379085 0.77777778
 0.91176471 0.97058824 0.99346405 1.        ]
********* year_of_treatment *********
Bin Edges: [58.  59.1 60.2 61.3 62.4 63.5 64.6 65.7 66.8 67.9 69. ]
PDF: [0.20588235 0.09150327 0.08496732 0.0751634  0.09803922 0.10130719
 0.09150327 0.09150327 0.08169935 0.07843137]
CDF: [0.20588235 0.29738562 0.38235294 0.45751634 0.55555556 0.65686275
 0.74836601 0.83986928 0.92156863 1.        ]
********* positive_lymph_nodes *********
Bin Edges: [ 0.   5.2 10.4 15.6 20.8 26.  31.2 36.4 41.6 46.8 52. ]
PDF: [0.77124183 0.09803922 0.05882353 0.02614379 0.02941176 0.00653595
 0.00326797 0.         0.00326797 0.00326797]
CDF: [0.77124183 0.86928105 0.92810458 0.95424837 0.98366013 0.99019608
 0.99346405 0.99346405 0.99673203 1.        ]
```



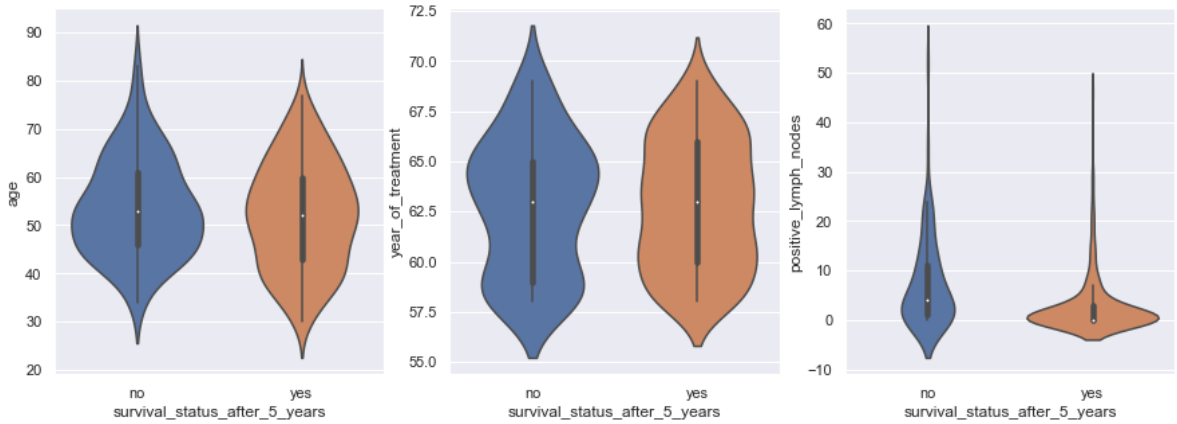
5.3 Box Plots

In [12]:
```
"""
Box plot takes a less space and visually represents the five number summary of
 the data points in a box.
The outliers are displayed as points outside the box.
1. Q1 - 1.5*IQR
2. Q1 (25th percentile)
3. Q2 (50th percentile or median)
4. Q3 (75th percentile)
5. Q3 + 1.5*IQR
Inter Quartile Range = Q3 -Q1
"""
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
for idx, feature in enumerate(list(hamber.columns)[:-1]):
    sns.boxplot( x='survival_status_after_5_years', y=feature, data=hamber, ax=
axes[idx])
plt.show()
```



5.4 Violin Plots

In [13]:
```
"""
Violin plot is the combination of box plot and probability density function.
"""
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
for idx, feature in enumerate(list(hamber.columns)[:-1]):
    sns.violinplot( x='survival_status_after_5_years', y=feature, data=hamber,
ax=axes[idx])
plt.show()
```

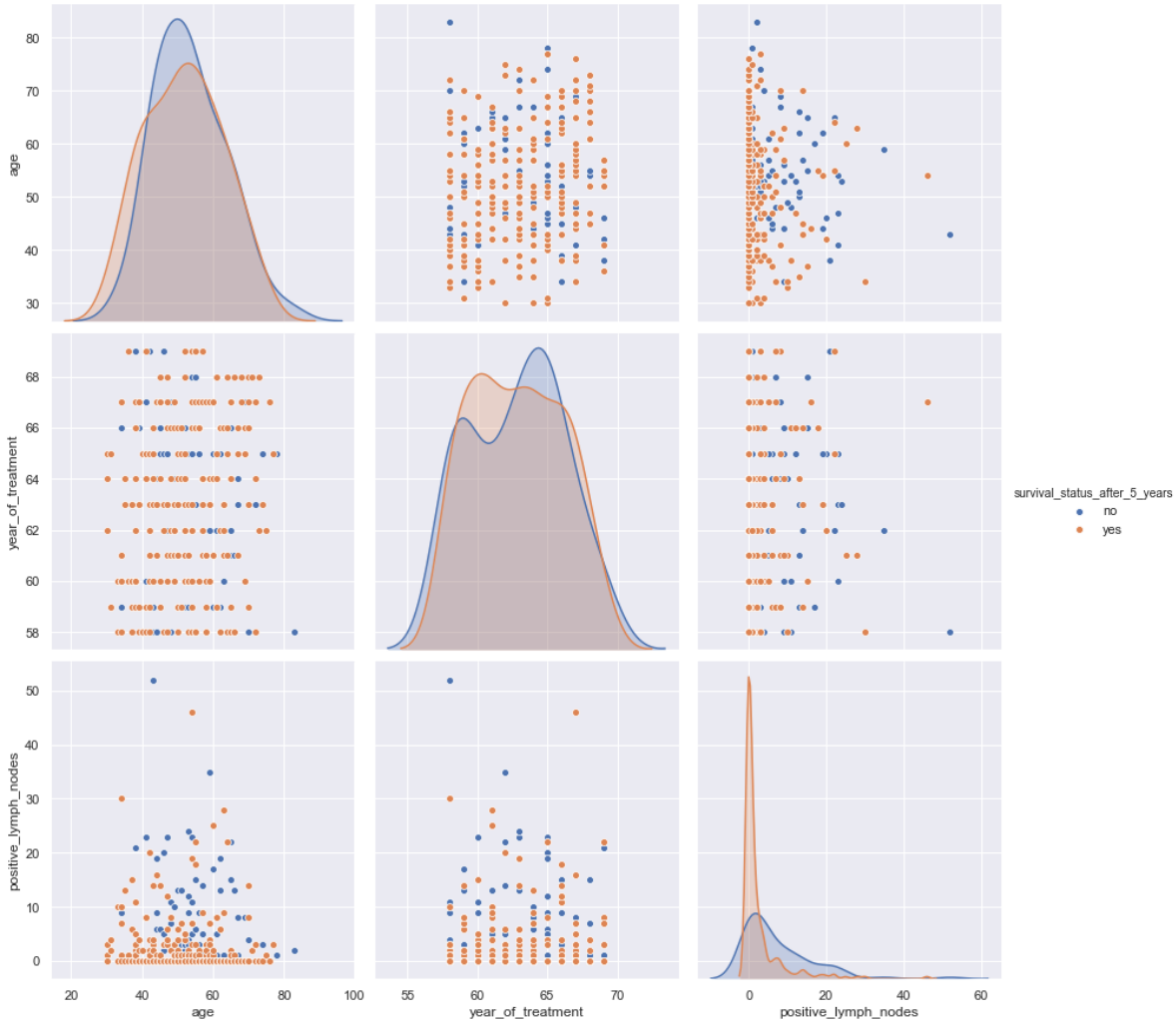


Observations

- The number of positive lymph nodes of the survivors is highly densed from 0 to 5. (#5.1)
- Almost 80% of the patients have less than or equal to 5 positive lymph nodes. (#5.2)
- The patients treated after 1966 have the slighlty higher chance to surive than the rest. The patients treated before 1959 have the slighlty lower chance to surive that the rest. (#5.3 and #5.4)

6. Multivariate Analysis

In [14]:
```
# pair plot
"""
Pair plot in seaborn plots the scatter plot between every two data columns in a
given dataframe.
It is used to visualize the relationship between two variables
"""
sns.pairplot(hamber, hue='survival_status_after_5_years', height=4)
plt.show()
```



Observations

- By scattering the data points between year_of_treatment and positive_lymph_nodes, we can see the better seperation between the two clases than other scatter plots.