Breast cancer signs and prediction

Chung-I Huang/ 2018-12-05

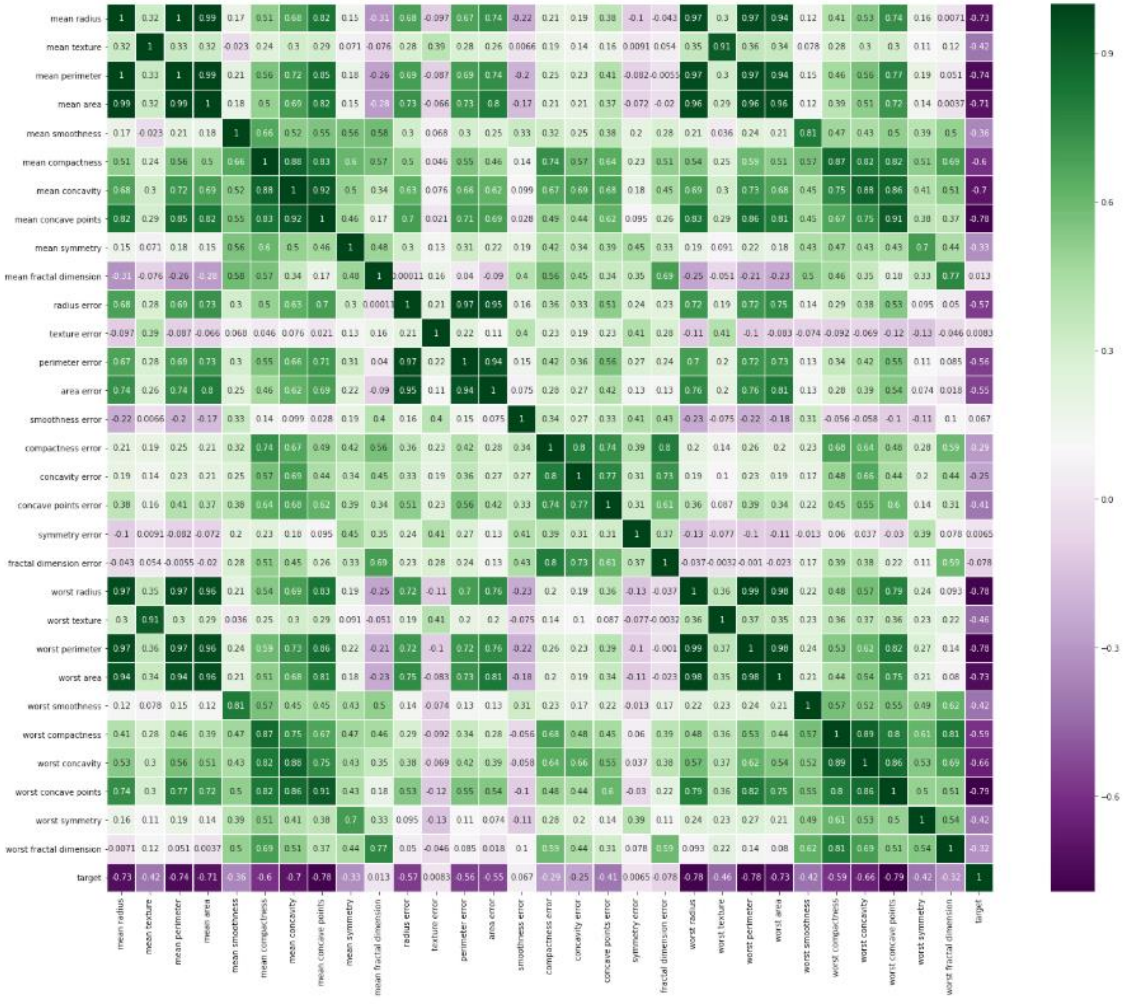
Abstract

I am using sklearn dataset's breast cancer data to figure out what symbols are more associated to breast cancer (benign or malignant)? The approaches I am applied for are DecisionTreeClassifier, KNN, and KMeans. The 1st & 2nd are belong to 'supervised' machine learning, and the 3rd is 'unsupervised'. Without a doubt, 1st & 2nd has a higher prediction result than the 3rd. In addition, there are 9 characteristics among total 31 have a high correlation to breast cancer (benign/malignant).

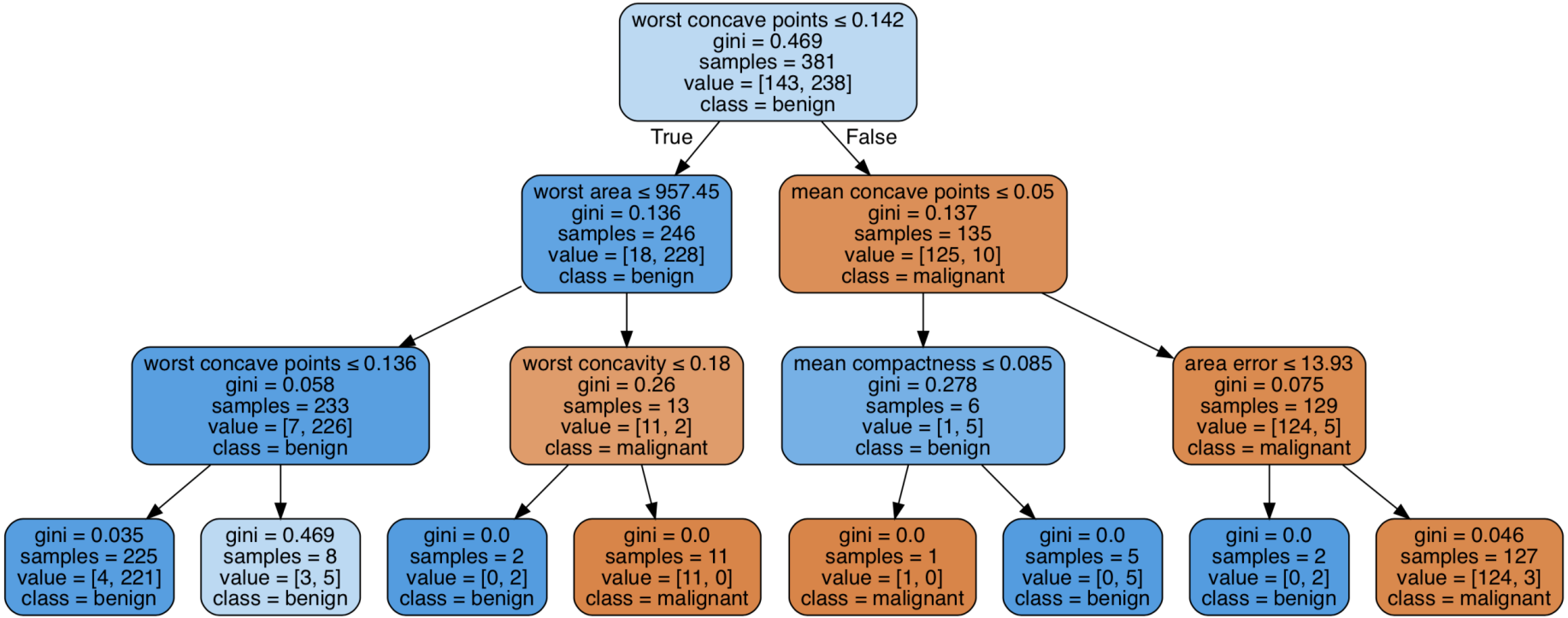
Motivation

To know what characters has a high chance led to breast cancer. Adopted ‘heatmap’ and ‘graphviz’ these two visualization tools (see next page). Thus, we can find out the high correlation factors and how does machine distinguish ‘benign’ and ‘malignant’ from listed characteristics.

Heatmap


mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
perimeter error
area error
smoothness error
compactness error
concavity error
concave points error
symmetry error
fractal dimension error
worst radius
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension
target
0.9
0.6
0.3
0.0
−0.3
−0.6

Graphviz


worst concave points ≤ 0.142
gini = 0.469
samples = 381
value = [143, 238]
class = benign
True
False
worst area ≤ 957.45
gini = 0.136
samples = 246
value = [18, 228]
class = benign
mean concave points ≤ 0.05
gini = 0.137
samples = 135
value = [125, 10]
class = malignant
worst concave points ≤ 0.136
gini = 0.058
samples = 233
value = [7, 226]
class = benign
worst concavity ≤ 0.18
gini = 0.26
samples = 13
value = [11, 2]
class = malignant
mean compactness ≤ 0.085
gini = 0.278
samples = 6
value = [1, 5]
class = benign
area error ≤ 13.93
gini = 0.075
samples = 129
value = [124, 5]
class = malignant
gini = 0.035
samples = 225
value = [4, 221]
class = benign
gini = 0.469
samples = 8
value = [3, 5]
class = benign
gini = 0.0
samples = 2
value = [0, 2]
class = benign
gini = 0.0
samples = 11
value = [11, 0]
class = malignant
gini = 0.0
samples = 1
value = [1, 0]
class = malignant
gini = 0.0
samples = 5
value = [0, 5]
class = benign
gini = 0.0
samples = 2
value = [0, 2]
class = benign
gini = 0.046
samples = 127
value = [124, 3]
class = malignant

Dataset(s)

Dataset -> https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

Since the dataset already in the sklearn.datasets, we can load it through typing the command of ‘from sklearn.datasets import load_breast_cancer’ in Juypter Notebook.

Data Preparation and Cleaning

First of all, check the data see if any row has null value on it by the command of ‘isnull().any()’. If it shows ‘True’, then use ‘dropna()’ function to clean the data.

In my case, I don’t see any null value. So, the data is good to go!

```
df.isnull().any()
```

```
mean radius                False
mean texture               False
mean perimeter             False
mean area                  False
mean smoothness            False
mean compactness           False
mean concavity             False
                           ...
worst smoothness           False
worst compactness          False
worst concavity            False
worst concave points       False
worst symmetry             False
worst fractal dimension    False
target                     False
Length: 31, dtype: bool
```

Research Question(s)

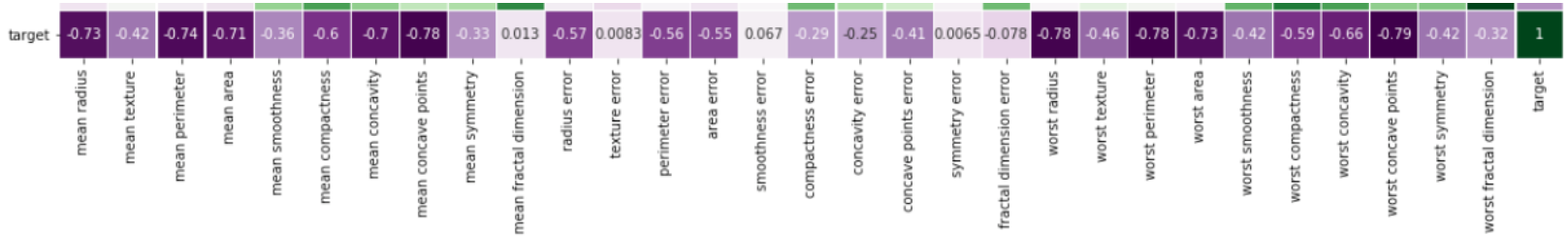
1. To figure out which characteristics have a high correlation to the breast cancer?
2. Compared two different supervised classification approaches, which one has a higher accuracy rate?
3. Using unsupervised method, then compare accuracy rate to the above two supervised methods.

Methods

Supervised approach: DecisionTreeClassifier, KNN

Unsupervised approach: KMeans

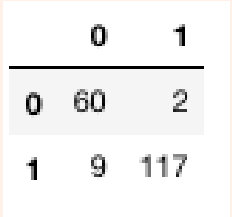
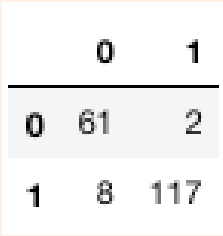
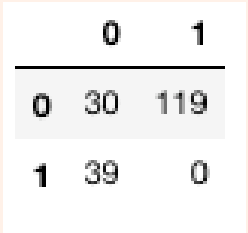
Findings & Conclusion



According to page 4 or above picture of the heatmap result, we noticed the darker purple color has a higher correlation to the breast cancer.

In page 5, we also found the decision tree using these darker purple signs to classify benign or malignant.

Findings & Conclusion

	DecisionTreeClassifier	KNN	KMeans
Accuracy rate	94.14%	94.68%	15.95%
Confusion matrix			

DecisionTreeClassifier and KNN has similar prediction result ~94%, but KMeans is obviously lower due to it is unsupervised approach.

Limitations

The data only has two results – benign and malignant, which is simple by using supervised data training.
However, I can foresee these two approaches won’t have that high accuracy rate (94%) when it has 3+ different results.

Acknowledgements

My friend suggested me to use KNN and KMeans approaches in this case due to our data is simple (only two labels - benign or malignant) and it’s belong to the classification question.

References

Data: sklearn.datasets

I did this presentation on my own.

Source: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.loa (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.loa

In [1]:

```
from sklearn.datasets import load_breast_cancer
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#load_breast_cancer()
pd.set_option('display.max_rows', 15)
pd.set_option('display.max_columns', 10)

breast_cancer = load_breast_cancer()
df = pd.DataFrame(breast_cancer["data"], columns = breast_cancer['feature_names'])
df['target'] = breast_cancer['target']
df.to_csv("breast_cancer.csv", encoding = 'utf-8', index = False)
df
```

Out[1]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	...	worst concavity	worst concave points	worst symmetry	wo frac dimens
0	17.99	10.38	122.80	1001.0	0.11840	...	0.7119	0.2654	0.4601	0.11
1	20.57	17.77	132.90	1326.0	0.08474	...	0.2416	0.1860	0.2750	0.08
2	19.69	21.25	130.00	1203.0	0.10960	...	0.4504	0.2430	0.3613	0.08
3	11.42	20.38	77.58	386.1	0.14250	...	0.6869	0.2575	0.6638	0.17
4	20.29	14.34	135.10	1297.0	0.10030	...	0.4000	0.1625	0.2364	0.07
5	12.45	15.70	82.57	477.1	0.12780	...	0.5355	0.1741	0.3985	0.12
6	18.25	19.98	119.60	1040.0	0.09463	...	0.3784	0.1932	0.3063	0.08
...	...	...	...	...	...	...	...	...	...	
562	15.22	30.62	103.40	716.9	0.10480	...	1.1700	0.2356	0.4089	0.14
563	20.92	25.09	143.00	1347.0	0.10990	...	0.6599	0.2542	0.2929	0.09
564	21.56	22.39	142.00	1479.0	0.11100	...	0.4107	0.2216	0.2060	0.07
565	20.13	28.25	131.20	1261.0	0.09780	...	0.3215	0.1628	0.2572	0.06
566	16.60	28.08	108.30	858.1	0.08455	...	0.3403	0.1418	0.2218	0.07
567	20.60	29.33	140.10	1265.0	0.11780	...	0.9387	0.2650	0.4087	0.12
568	7.76	24.54	47.92	181.0	0.05263	...	0.0000	0.0000	0.2871	0.07

569 rows × 31 columns

Data Cleaning - Whether any row has NULL value, if yes then diplay True

In [2]:

```
df.isnull().any()
```

Out[2]:

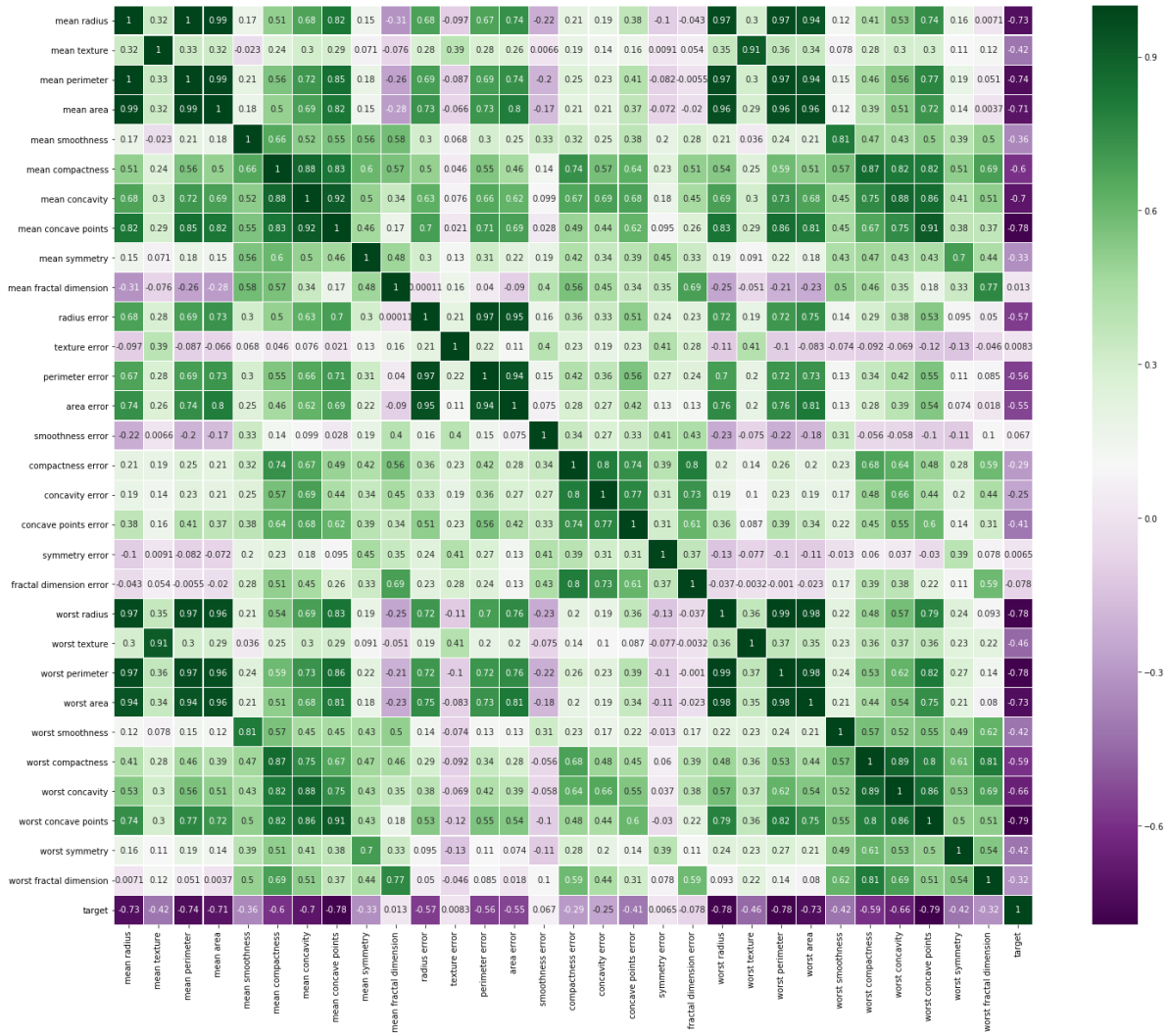
```
mean radius                False
mean texture               False
mean perimeter             False
mean area                  False
mean smoothness            False
mean compactness           False
mean concavity             False
                           ...
worst smoothness           False
worst compactness          False
worst concavity            False
worst concave points       False
worst symmetry             False
worst fractal dimension    False
target                     False
Length: 31, dtype: bool
```

In [3]:

```
%matplotlib inline

plt.figure(figsize=(25,20))
sns.heatmap(df.astype(float).corr(), cmap = 'PRGn', linewidths = 0.1, square = True,
```

Out[3]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a0da68f98>
```



Choose the color => https://matplotlib.org/2.0.2/examples/color/colormaps_reference.html (https://matplotlib.org/2.0.2/examples/color/colormaps_reference.html)

The coefficient scores below -0.7 are much related to the breast cancer.

In [4]:

```
df.columns.shape[0]
```

Out[4]:

```
31
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimensio
n',
       'radius error', 'texture error', 'perimeter error', 'area erro
r',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimension er
ror',
       'worst radius', 'worst texture', 'worst perimeter', 'worst are
a',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimens
ion',
       'target'],
      dtype='object')
```

Perform Test and Train split

I am going to adopt all of columns excluded target as Features to Predict breast cancer

In [6]:

```
from sklearn.model_selection import train_test_split
data_train, data_test, target_train, target_test = train_test_split(breast_cancer['d
                                                                    test_size = 0.3
```

Approach#1 DecisionTreeClassifier approach (Supervised)

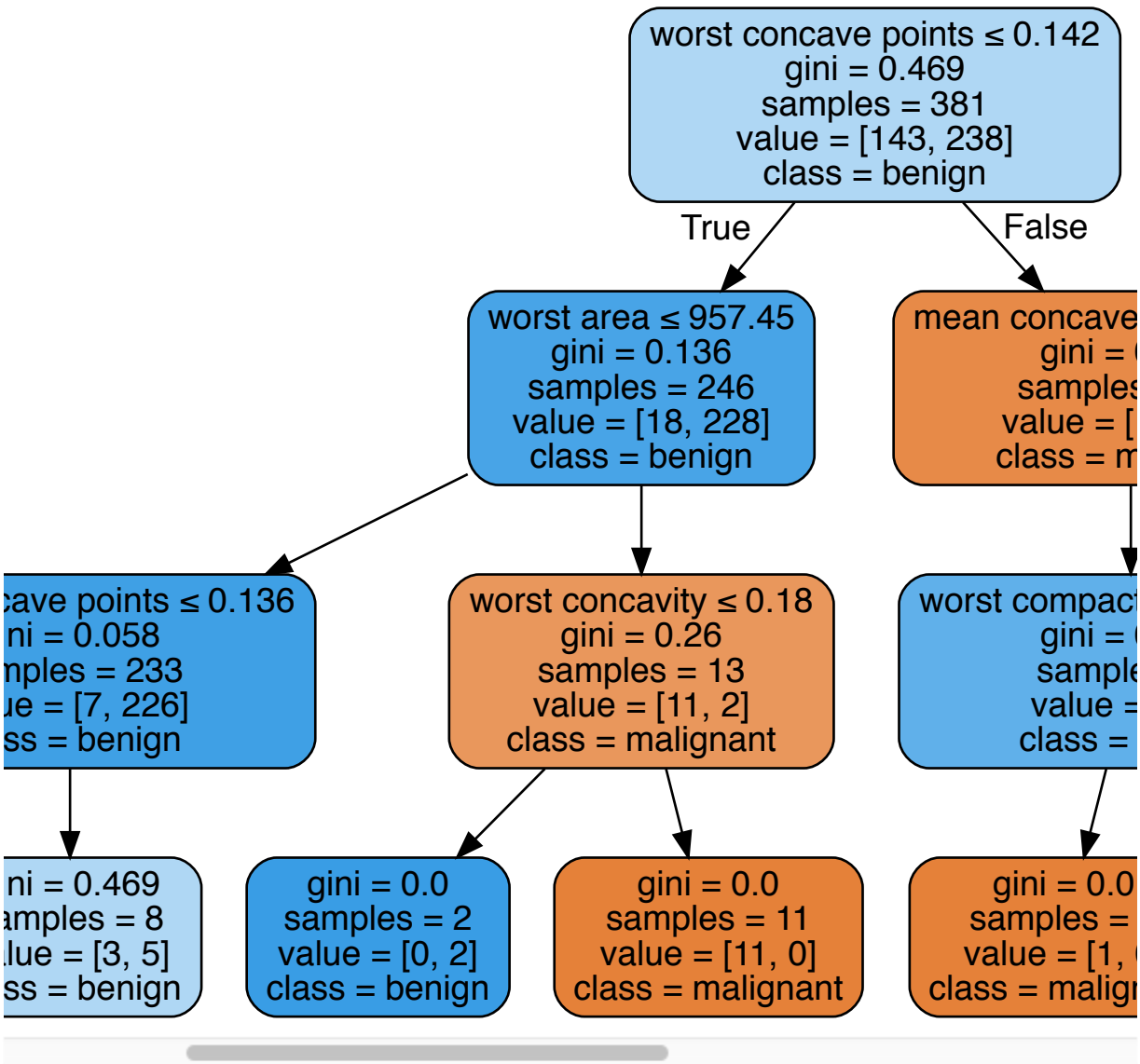
In [7]:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_depth = 3)
clf = clf.fit(data_train, target_train)
```

In [8]:

```
from sklearn.tree import export_graphviz
import graphviz
tree_pattern = export_graphviz(clf, out_file = None,
                               feature_names = breast_cancer['feature_names'],
                               class_names = breast_cancer['target_names'],
                               filled = True, rounded = True,
                               special_characters = True)
g = graphviz.Source(tree_pattern)
g.render('breast_cancer_graphviz') #Export to PDF
g
```

Out[8]:



Confusion Matrix for DecisionTreeClassifier

In [9]:

```
from sklearn.metrics import confusion_matrix
result_clf = confusion_matrix(clf.predict(data_test), target_test)
pd.DataFrame(result_clf)
```

Out[9]:

	0	1
0	60	2
1	9	117

Accuracy score for DecisionTreeClassifier

In [10]:

```
from sklearn.metrics import accuracy_score
print("Predict:", list(clf.predict(data_test)))
print("True:", list(target_test))
print('Accuracy Score:', accuracy_score(clf.predict(data_test), target_test) * 100,
```

```
Predict: [0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 1]
True: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 1]
Accuracy Score: 94.14893617021278 %
```

In [11]:

```
#or write in this way
accuracy_score(y_pred = clf.predict(data_test), y_true = target_test)
```

Out[11]:

```
0.9414893617021277
```

Approach#2 KNN algorithm (Supervised)

In [12]:

```
from sklearn.neighbors import KNeighborsClassifier

kclf = KNeighborsClassifier(n_neighbors = 3)
kclf = kclf.fit(data_train, target_train)
```

Cufsion matrix for KNN

In [13]:

```
from sklearn.metrics import confusion_matrix
result_kclf = confusion_matrix(kclf.predict(data_test), target_test)
pd.DataFrame(result_kclf)
```

Out[13]:

	0	1
0	61	2
1	8	117

Accuracy score for KNN

In [14]:

```
from sklearn.metrics import accuracy_score
print("Predict:", list(kclf.predict(data_test)))
print("True:", list(target_test))
print('Accuracy score:', accuracy_score(kclf.predict(data_test), target_test) * 100,
```

```
Predict: [0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1,
1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 1]
True: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 1]
Accuracy score: 94.68085106382979 %
```

Approach#3 KMeans algorithm (Unsupervised)

In [15]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 2)
kmeans.fit(data_train) # Due to it is unsupervised, don't need 'target_train'
```

Out[15]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

Cufsion matrix for KMeans

In [16]:

```
from sklearn.metrics import confusion_matrix
result_kmeans = confusion_matrix(kmeans.predict(data_test), target_test)
pd.DataFrame(result_kmeans)
```

Out[16]:

	0	1
0	30	119
1	39	0

Accuracy score for KMeans

In [17]:

```
print("Predict:", list(kmeans.predict(data_test)))
print("True:", list(target_test))
print('Accuracy score:', accuracy_score(kmeans.predict(data_test), target_test) * 1(
```

```
Predict: [1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0]
True: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 1]
Accuracy score: 15.957446808510639 %
```

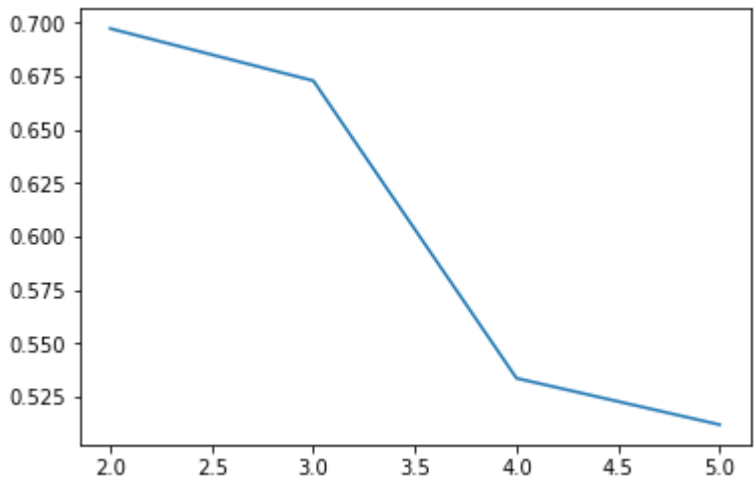
In [18]:

```
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
%matplotlib inline

scores = []
ks = []

for i in range (2,6):

    kmeans = KMeans(n_clusters= i)
    kmeans.fit(breast_cancer['data'])
    scores.append(silhouette_score(breast_cancer['data'], kmeans.labels_))
    ks.append(i)

print(scores)
print(ks)

plt.plot (ks, scores)
```

```
[0.6972646156059464, 0.6728663978657781, 0.533753360908508, 0.51205885
04057626]
[2, 3, 4, 5]
```

Out[18]:

```
[<matplotlib.lines.Line2D at 0x1a156d6a90>]
```



In this plot, we can notice machine judge there are two clusters (when k= 2~3 has a high value). This is make sense since we have benign and malignant in the classify.

Source: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.loa (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.loa

In [1]:

```
from sklearn.datasets import load_breast_cancer
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#load_breast_cancer()
pd.set_option('display.max_rows', 15)
pd.set_option('display.max_columns', 10)

breast_cancer = load_breast_cancer()
df = pd.DataFrame(breast_cancer["data"], columns = breast_cancer['feature_names'])
df['target'] = breast_cancer['target']
df.to_csv("breast_cancer.csv", encoding = 'utf-8', index = False)
df
```

Out[1]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	...	worst concavity	worst concave points	worst symmetry	wo frac dimens
0	17.99	10.38	122.80	1001.0	0.11840	...	0.7119	0.2654	0.4601	0.11
1	20.57	17.77	132.90	1326.0	0.08474	...	0.2416	0.1860	0.2750	0.08
2	19.69	21.25	130.00	1203.0	0.10960	...	0.4504	0.2430	0.3613	0.08
3	11.42	20.38	77.58	386.1	0.14250	...	0.6869	0.2575	0.6638	0.17
4	20.29	14.34	135.10	1297.0	0.10030	...	0.4000	0.1625	0.2364	0.07
5	12.45	15.70	82.57	477.1	0.12780	...	0.5355	0.1741	0.3985	0.12
6	18.25	19.98	119.60	1040.0	0.09463	...	0.3784	0.1932	0.3063	0.08
...	...	...	...	...	...	...	...	...	...	
562	15.22	30.62	103.40	716.9	0.10480	...	1.1700	0.2356	0.4089	0.14
563	20.92	25.09	143.00	1347.0	0.10990	...	0.6599	0.2542	0.2929	0.09
564	21.56	22.39	142.00	1479.0	0.11100	...	0.4107	0.2216	0.2060	0.07
565	20.13	28.25	131.20	1261.0	0.09780	...	0.3215	0.1628	0.2572	0.06
566	16.60	28.08	108.30	858.1	0.08455	...	0.3403	0.1418	0.2218	0.07
567	20.60	29.33	140.10	1265.0	0.11780	...	0.9387	0.2650	0.4087	0.12
568	7.76	24.54	47.92	181.0	0.05263	...	0.0000	0.0000	0.2871	0.07

569 rows × 31 columns

Data Cleaning - Whether any row has NULL value, if yes then diplay True

In [2]:

```
df.isnull().any()
```

Out[2]:

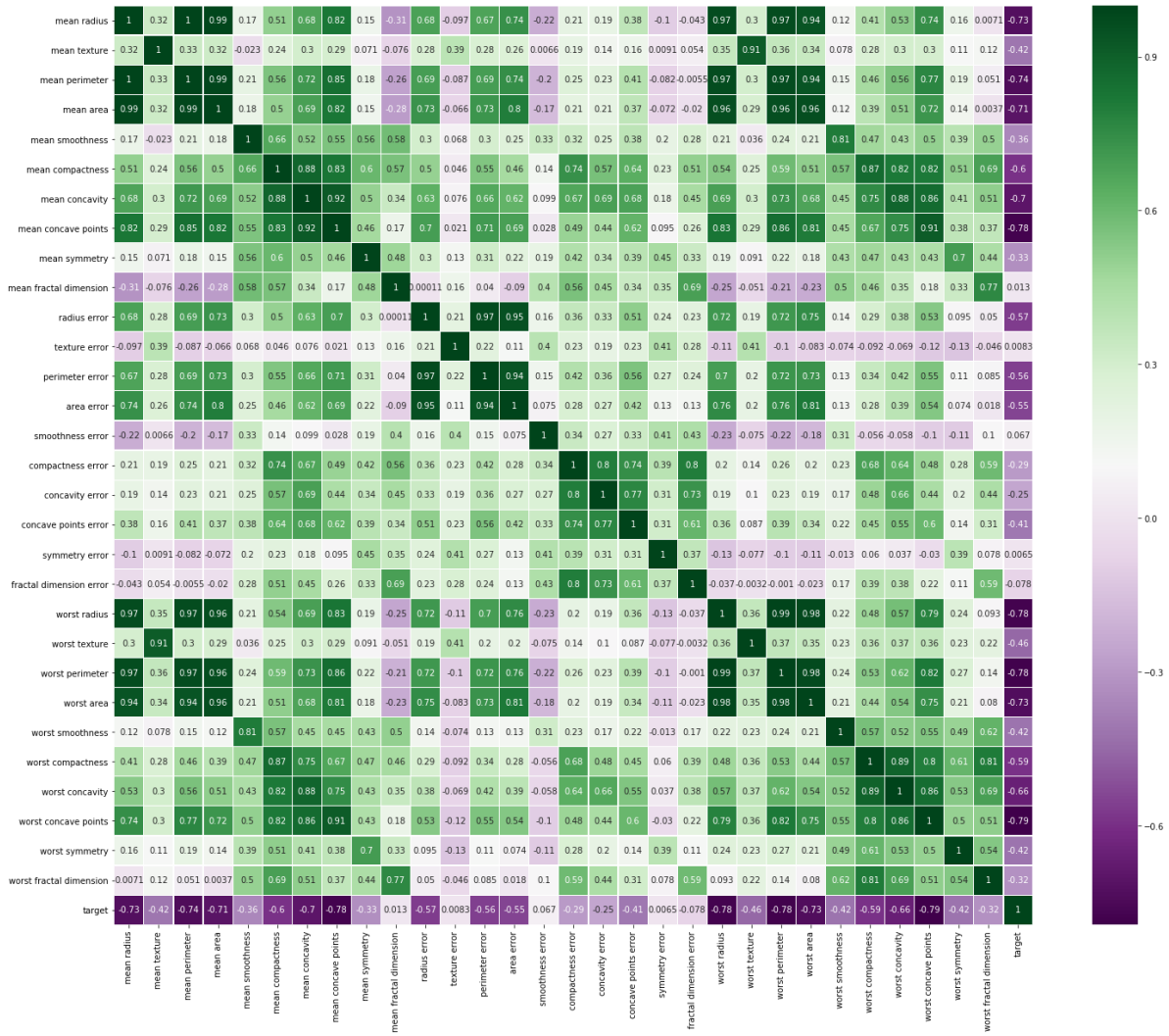
```
mean radius                False
mean texture               False
mean perimeter             False
mean area                  False
mean smoothness            False
mean compactness           False
mean concavity             False
                           ...
worst smoothness           False
worst compactness          False
worst concavity            False
worst concave points       False
worst symmetry             False
worst fractal dimension    False
target                     False
Length: 31, dtype: bool
```

In [3]:

```
%matplotlib inline

plt.figure(figsize=(25,20))
sns.heatmap(df.astype(float).corr(), cmap = 'PRGn', linewidths = 0.1, square = True,
```

Out[3]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a0da68f98>
```



Choose the color => https://matplotlib.org/2.0.2/examples/color/colormaps_reference.html (https://matplotlib.org/2.0.2/examples/color/colormaps_reference.html)

The coefficient scores below -0.7 are much related to the breast cancer.

In [4]:

```
df.columns.shape[0]
```

Out[4]:

```
31
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimensio
n',
       'radius error', 'texture error', 'perimeter error', 'area erro
r',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimension er
ror',
       'worst radius', 'worst texture', 'worst perimeter', 'worst are
a',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimens
ion',
       'target'],
      dtype='object')
```

Perform Test and Train split

I am going to adopt all of columns excluded target as Features to Predict breast cancer

In [6]:

```
from sklearn.model_selection import train_test_split
data_train, data_test, target_train, target_test = train_test_split(breast_cancer['d
                                                                    test_size = 0.3
```

Approach#1 DecisionTreeClassifier approach (Supervised)

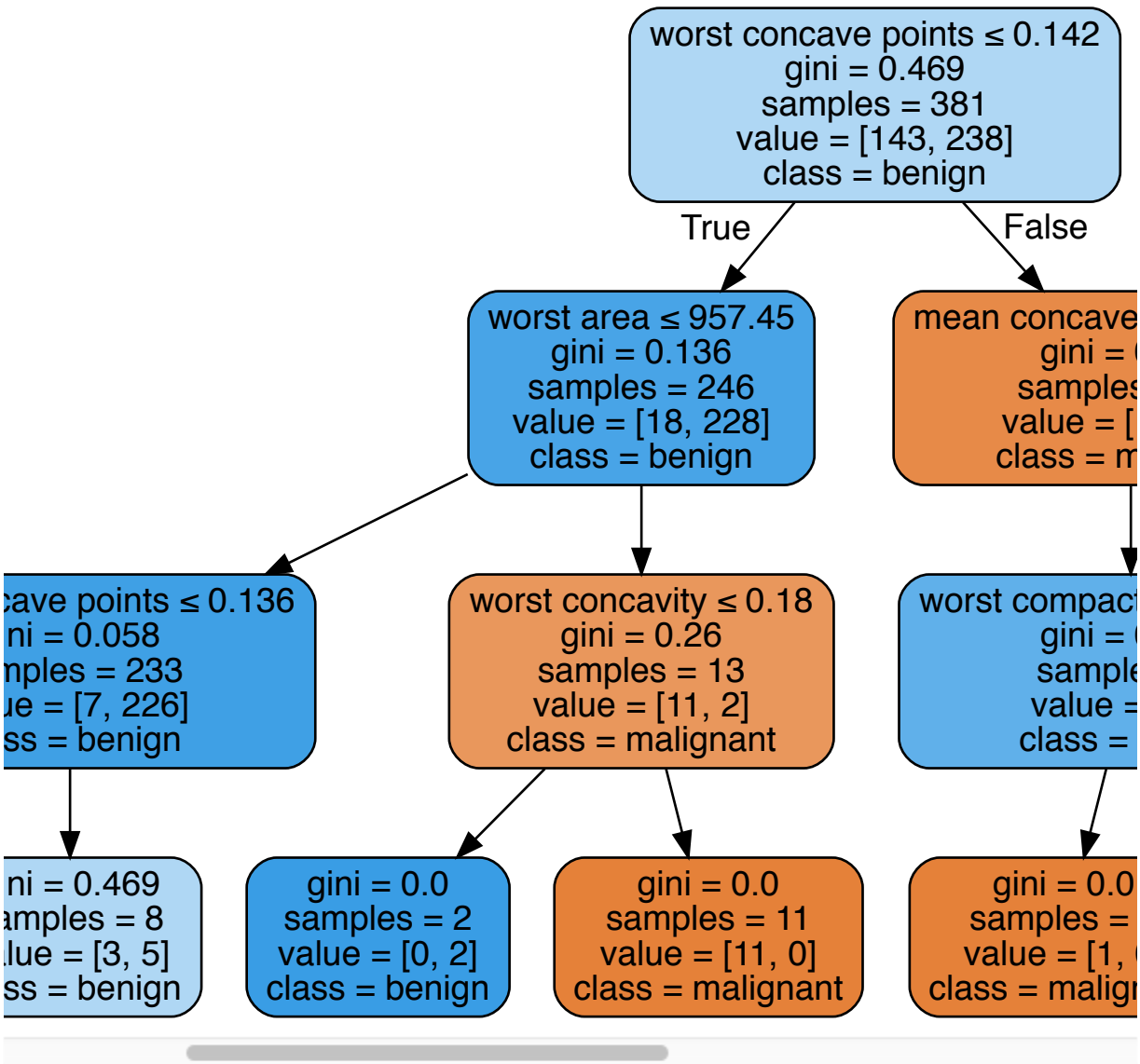
In [7]:

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_depth = 3)
clf = clf.fit(data_train, target_train)
```

In [8]:

```
from sklearn.tree import export_graphviz
import graphviz
tree_pattern = export_graphviz(clf, out_file = None,
                               feature_names = breast_cancer['feature_names'],
                               class_names = breast_cancer['target_names'],
                               filled = True, rounded = True,
                               special_characters = True)
g = graphviz.Source(tree_pattern)
g.render('breast_cancer_graphviz') #Export to PDF
g
```

Out[8]:



Confusion Matrix for DecisionTreeClassifier

In [9]:

```
from sklearn.metrics import confusion_matrix
result_clf = confusion_matrix(clf.predict(data_test), target_test)
pd.DataFrame(result_clf)
```

Out[9]:

	0	1
0	60	2
1	9	117

Accuracy score for DecisionTreeClassifier

In [10]:

```
from sklearn.metrics import accuracy_score
print("Predict:", list(clf.predict(data_test)))
print("True:", list(target_test))
print('Accuracy Score:', accuracy_score(clf.predict(data_test), target_test) * 100,
```

```
Predict: [0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 1]
True: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 1]
Accuracy Score: 94.14893617021278 %
```

In [11]:

```
#or write in this way
accuracy_score(y_pred = clf.predict(data_test), y_true = target_test)
```

Out[11]:

```
0.9414893617021277
```

Approach#2 KNN algorithm (Supervised)

In [12]:

```
from sklearn.neighbors import KNeighborsClassifier

kclf = KNeighborsClassifier(n_neighbors = 3)
kclf = kclf.fit(data_train, target_train)
```

Cufsion matrix for KNN

In [13]:

```
from sklearn.metrics import confusion_matrix
result_kclf = confusion_matrix(kclf.predict(data_test), target_test)
pd.DataFrame(result_kclf)
```

Out[13]:

	0	1
0	61	2
1	8	117

Accuracy score for KNN

In [14]:

```
from sklearn.metrics import accuracy_score
print("Predict:", list(kclf.predict(data_test)))
print("True:", list(target_test))
print('Accuracy score:', accuracy_score(kclf.predict(data_test), target_test) * 100,
```

```
Predict: [0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1,
1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 1, 1, 1]
True: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 1]
Accuracy score: 94.68085106382979 %
```

Approach#3 KMeans algorithm (Unsupervised)

In [15]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 2)
kmeans.fit(data_train) # Due to it is unsupervised, don't need 'target_train'
```

Out[15]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
    n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

Cufsion matrix for KMeans

In [16]:

```
from sklearn.metrics import confusion_matrix
result_kmeans = confusion_matrix(kmeans.predict(data_test), target_test)
pd.DataFrame(result_kmeans)
```

Out[16]:

	0	1
0	30	119
1	39	0

Accuracy score for KMeans

In [17]:

```
print("Predict:", list(kmeans.predict(data_test)))
print("True:", list(target_test))
print('Accuracy score:', accuracy_score(kmeans.predict(data_test), target_test) * 1(
```

```
Predict: [1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0]
True: [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1,
1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0, 1, 1, 1, 1]
Accuracy score: 15.957446808510639 %
```

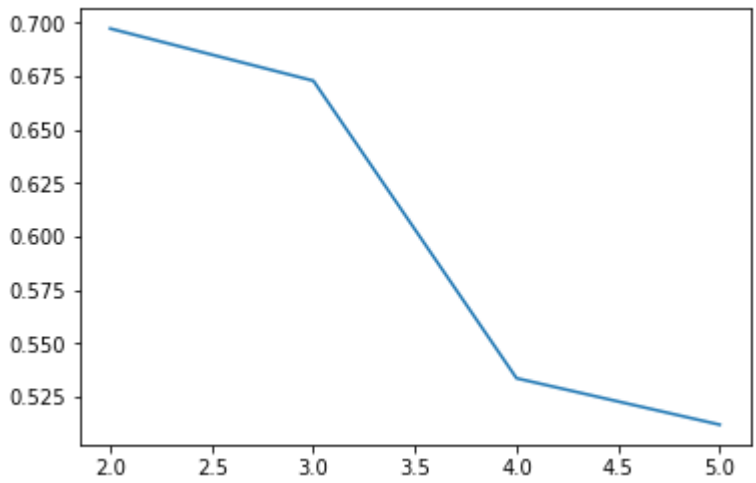
In [18]:

```
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
%matplotlib inline

scores = []
ks = []

for i in range (2,6):

    kmeans = KMeans(n_clusters= i)
    kmeans.fit(breast_cancer['data'])
    scores.append(silhouette_score(breast_cancer['data'], kmeans.labels_))
    ks.append(i)

print(scores)
print(ks)

plt.plot (ks, scores)
```

```
[0.6972646156059464, 0.6728663978657781, 0.533753360908508, 0.51205885
04057626]
[2, 3, 4, 5]
```

Out[18]:

```
[<matplotlib.lines.Line2D at 0x1a156d6a90>]
```



In this plot, we can notice machine judge there are two clusters (when k= 2~3 has a high value). This is make sense since we have benign and malignant in the classify.