Reading in Data

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import numpy as np
```

In [2]:
```
#reading in data.csv file
tissue_pd = pd.read_csv("project/data.csv")
```

In [3]:
```
#looking at first five rows of our data
tissue_pd.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	c point
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 33 columns

In [4]:
```
print(tissue_pd.shape)
print("569 rows and 33 columns")
```

```
(569, 33)
569 rows and 33 columns
```

Our project dataset is from the UCI Machine Learning Repository. Each row represents a patient and information around their breast tissue imaging, especially with respect to cell nuclei.

Missing Data?

In [5]:
```
#loop through columns in dataframe
#check for any NaN values
for col in tissue_pd.columns:
    print(col + ": " + str(tissue_pd[col].isnull().values.any()))
```

```
id: False
diagnosis: False
radius_mean: False
texture_mean: False
perimeter_mean: False
area_mean: False
smoothness_mean: False
compactness_mean: False
concavity_mean: False
concave points_mean: False
symmetry_mean: False
fractal_dimension_mean: False
radius_se: False
texture_se: False
perimeter_se: False
area_se: False
smoothness_se: False
compactness_se: False
concavity_se: False
concave points_se: False
symmetry_se: False
fractal_dimension_se: False
radius_worst: False
texture_worst: False
perimeter_worst: False
area_worst: False
smoothness_worst: False
compactness_worst: False
concavity_worst: False
concave points_worst: False
symmetry_worst: False
fractal_dimension_worst: False
Unnamed: 32: True
```

There are no missing values in our data!!

Checking for Duplicates

In [6]:
```
#checking for row-level duplicates in entire dataframe

print(tissue_pd.duplicated())

print("\nSum of duplicate categories: ")
print(tissue_pd.duplicated().sum())
```

```
0      False
1      False
2      False
3      False
4      False
       ...
564    False
565    False
566    False
567    False
568    False
Length: 569, dtype: bool

Sum of duplicate categories:
0
```

There are no row-level duplicates in our dataset!

Data Exploration

In [7]:
```
#looking at column types
tissue_pd.dtypes
```

Out[7]:
```
id                          int64
diagnosis                  object
radius_mean               float64
texture_mean              float64
perimeter_mean            float64
area_mean                 float64
smoothness_mean           float64
compactness_mean          float64
concavity_mean            float64
concave points_mean       float64
symmetry_mean             float64
fractal_dimension_mean    float64
radius_se                 float64
texture_se                float64
perimeter_se              float64
area_se                   float64
smoothness_se             float64
compactness_se            float64
concavity_se              float64
concave points_se         float64
symmetry_se               float64
fractal_dimension_se      float64
radius_worst              float64
texture_worst             float64
perimeter_worst           float64
area_worst                float64
smoothness_worst          float64
compactness_worst         float64
concavity_worst           float64
concave points_worst      float64
symmetry_worst            float64
fractal_dimension_worst   float64
Unnamed: 32               float64
dtype: object
```

There are three layers of our data: the mean (_mean), standard error (_se) and "worst" (_worst) which denotes the largest value i.e. the mean of the three largest values in the cell data.

Our target value in terms of logistic regression is the variable 'diagnosis'. It is currently denoted as an object type; however, for regression purposes, we will convert it to a numerical type with encoding towards the end of this analysis.

All of the other variables are floats, and they are rounded to 4 significant figures.

In [8]:
```
#converting id variable to type string
tissue_pd['id'] = tissue_pd['id'].astype(str)
```

In [9]:
```
#describing the data
tissue_pd.describe()
```

Out[9]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmet
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	56
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	

8 rows × 31 columns

From looking at the description of the numerical variables, I am going to remove the column 'Unnamed:32', as it does not have any values besides NaN. It will not be helpful in our analysis or modeling.

In [10]:
```
#dropping Unnamed column
tissue_pd = tissue_pd.drop(['Unnamed: 32'],axis='columns')
```

In [11]:
```
tissue_pd.head()
```

Out[11]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	c point
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

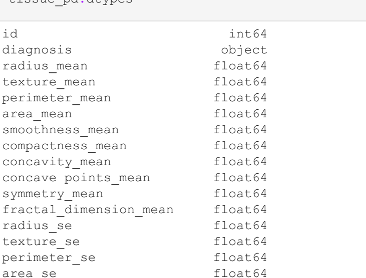
5 rows × 32 columns

I also want to look at the class distribution of our target variable 'diagnosis', as it will help us understand if we need to account for class imbalance in our modeling.

In [12]:
```
#looking at value counts of diagnosis variable
print(tissue_pd['diagnosis'].value_counts())
tissue_pd['diagnosis'].value_counts().sort_values().plot(kind = 'barh')
```

```
B    357
M    212
Name: diagnosis, dtype: int64
```

Out[12]:
```
<AxesSubplot:>
```



Out of the 569 patients that were screened for breast cancer, 357 were diagnosed with benign masses, meaning they do not have breast cancer. The other 212 patients were found to have malignant or cancerous tumors.

In terms of class distribution, they are pretty equal in sample size, and I would say that the classes are not imbalanced.

1/2 * 569 = 284.5

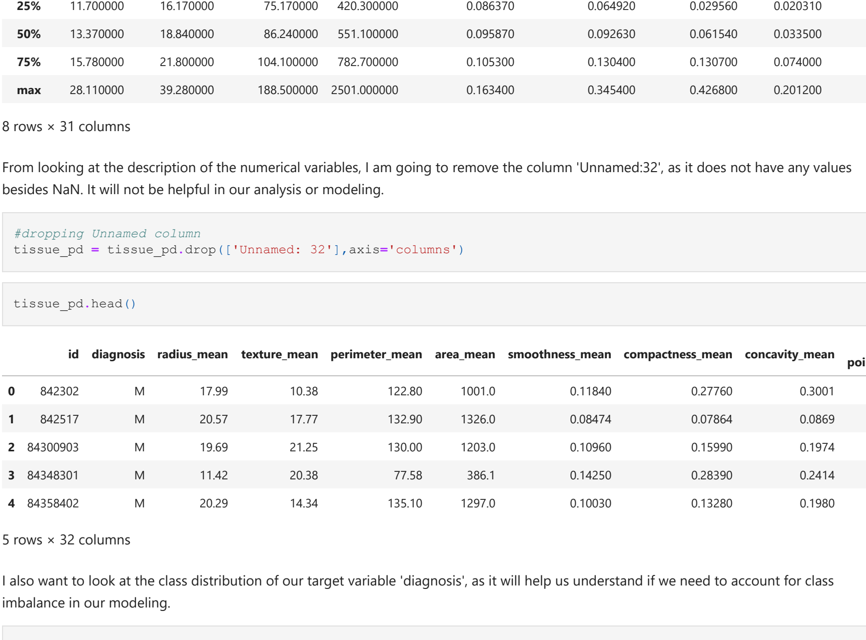
- Benign class is slightly above half
- Malignant class is slightly below half

Data Visualizations

Histograms of Numeric Variables

In [13]:
```
tissue_pd.hist(bins=10, figsize=(15, 10))
```

Out[13]:
```
array([[<AxesSubplot:title={'center':'radius_mean'}>,
        <AxesSubplot:title={'center':'texture_mean'}>,
        <AxesSubplot:title={'center':'perimeter_mean'}>,
        <AxesSubplot:title={'center':'area_mean'}>,
        <AxesSubplot:title={'center':'smoothness_mean'}>],
       [<AxesSubplot:title={'center':'compactness_mean'}>,
        <AxesSubplot:title={'center':'concavity_mean'}>,
        <AxesSubplot:title={'center':'concave points_mean'}>,
        <AxesSubplot:title={'center':'symmetry_mean'}>,
        <AxesSubplot:title={'center':'fractal_dimension_mean'}>],
       [<AxesSubplot:title={'center':'radius_se'}>,
        <AxesSubplot:title={'center':'texture_se'}>,
        <AxesSubplot:title={'center':'perimeter_se'}>,
        <AxesSubplot:title={'center':'area_se'}>,
        <AxesSubplot:title={'center':'smoothness_se'}>],
       [<AxesSubplot:title={'center':'compactness_se'}>,
        <AxesSubplot:title={'center':'concavity_se'}>,
        <AxesSubplot:title={'center':'concave points_se'}>,
        <AxesSubplot:title={'center':'symmetry_se'}>,
        <AxesSubplot:title={'center':'fractal_dimension_se'}>],
       [<AxesSubplot:title={'center':'radius_worst'}>,
        <AxesSubplot:title={'center':'texture_worst'}>,
        <AxesSubplot:title={'center':'perimeter_worst'}>,
        <AxesSubplot:title={'center':'area_worst'}>,
        <AxesSubplot:title={'center':'smoothness_worst'}>],
       [<AxesSubplot:title={'center':'compactness_worst'}>,
        <AxesSubplot:title={'center':'concavity_worst'}>,
        <AxesSubplot:title={'center':'concave points_worst'}>,
        <AxesSubplot:title={'center':'symmetry_worst'}>,
        <AxesSubplot:title={'center':'fractal_dimension_worst'}>]],
      dtype=object)
```



Most of the distributions for the numerical variables in the dataset lean towards a positive skew (majority of the points are at the lower value range).

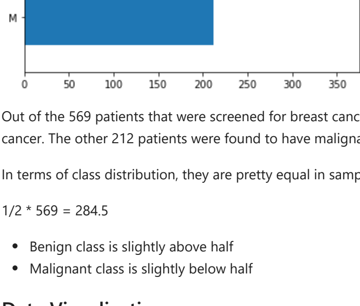
There are a few distributions that are relatively normal though: texture_mean, smoothness_mean, symmetry_mean, texture_worst and smoothness_worst. They are mostly centered around the mean of the column values, and follow a relatively similar Guassian distribution. There are also some variables which may contain outliers, and when those are assessed and possibly removed, we can look at their histograms again to assess normality.

Boxplots of Numerical Variables -- Assessing Outliers

In [14]:
```
#boxplot of perimeter_mean
sns.boxplot(tissue_pd['perimeter_mean'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

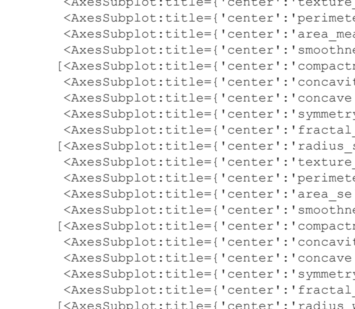
Out[14]:
```
<AxesSubplot:xlabel='perimeter_mean'>
```



In [15]:
```
#boxplot of concavepoints_mean
sns.boxplot(tissue_pd['concave points_mean'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

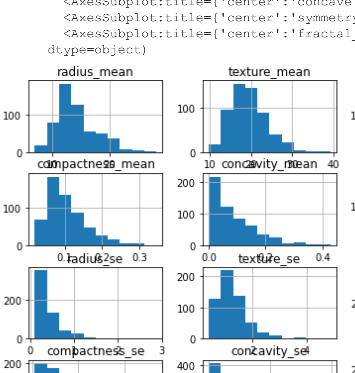
Out[15]:
```
<AxesSubplot:xlabel='concave points_mean'>
```



In [16]:
```
sns.boxplot(tissue_pd['fractal_dimension_mean'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

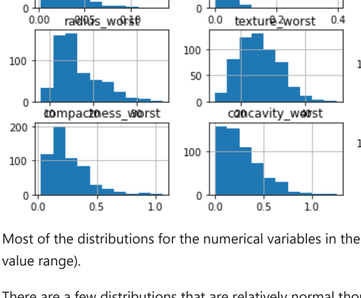
Out[16]:
```
<AxesSubplot:xlabel='fractal_dimension_mean'>
```



In [17]:
```
sns.boxplot(tissue_pd['texture_se'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

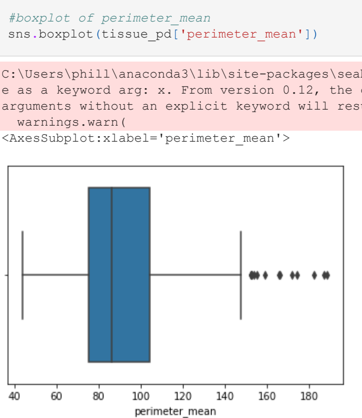
Out[17]:
```
<AxesSubplot:xlabel='texture_se'>
```



In [18]:
```
sns.boxplot(tissue_pd['area_worst'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

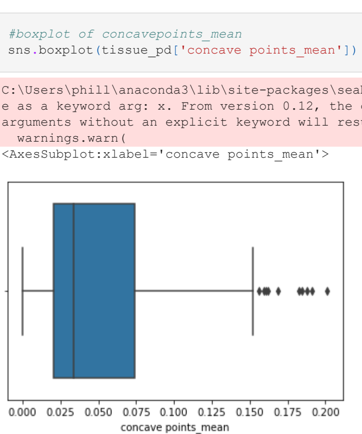
Out[18]:
```
<AxesSubplot:xlabel='area_worst'>
```



In [19]:
```
sns.boxplot(tissue_pd['compactness_worst'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[19]:
```
<AxesSubplot:xlabel='compactness_worst'>
```



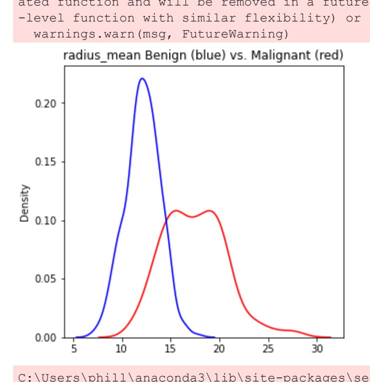
From looking over the boxplots, it is clear that there are some outlier points which would need to be further analyzed. However, when I was reading the original research paper from where this data came from, it mentioned that higher and lower value points for the various measures were usually indicative of malignancy in patients' breast masses.

Therefore, if we were to eliminate these points, it would disrupt our data and possibiliy minimize the number of malignant diagnoses in our target variable. I will not be removing them then.

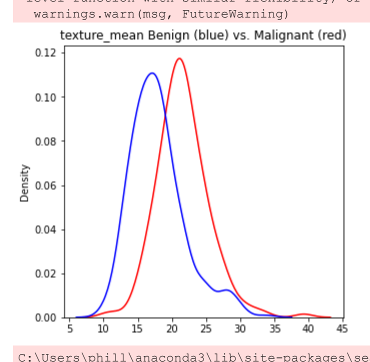
Visualizations with Target Variable

In [20]:
```
#separating dataframes into diagnosis categories
#malignant
tissue_malignant = tissue_pd.loc[tissue_pd['diagnosis'] == 'M']
#benign
tissue_benign = tissue_pd.loc[tissue_pd['diagnosis'] == 'B']

for column in tissue_pd.columns:
    if column not in  ['id', 'diagnosis']:
        plt.figure(figsize = (5, 5))

        #plotting columns from malignant vs. benign as distplot (distribution)
        sns.distplot(a = tissue_malignant[[column]], hist = False, color = 'red')
        sns.distplot(a = tissue_benign[[column]], hist = False, color = 'blue')

        plt.title(column + ' Benign (blue) vs. Malignant (red)')
        plt.show()
```

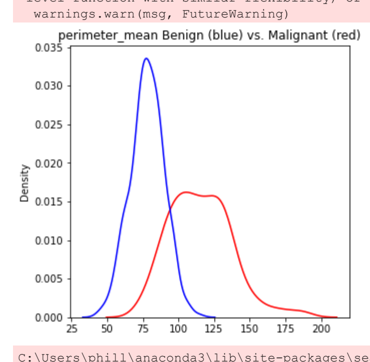
```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



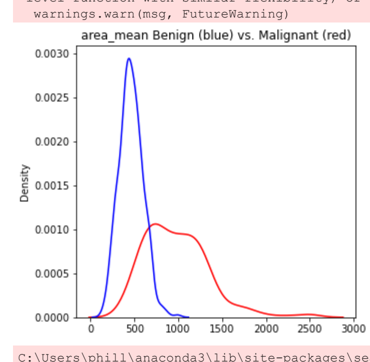

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



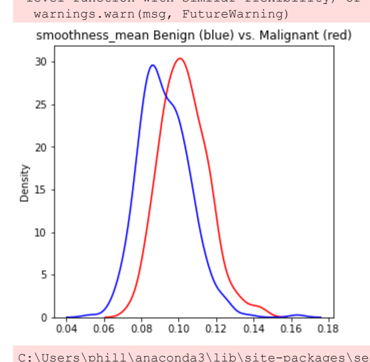

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



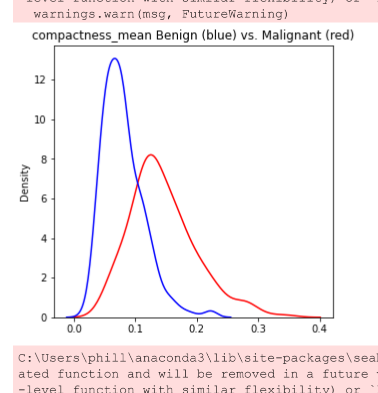

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



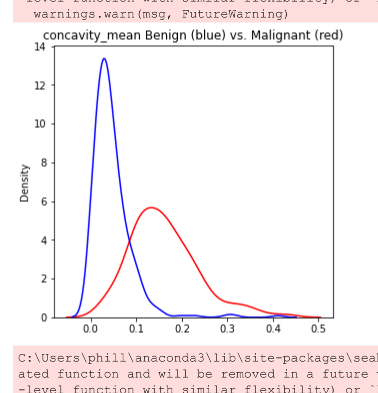

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



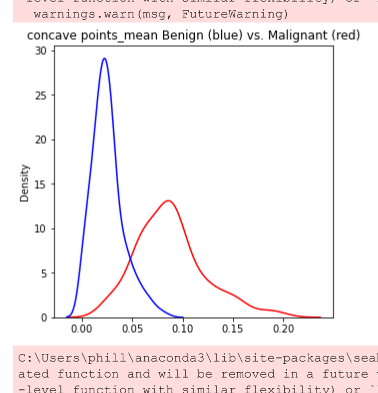

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



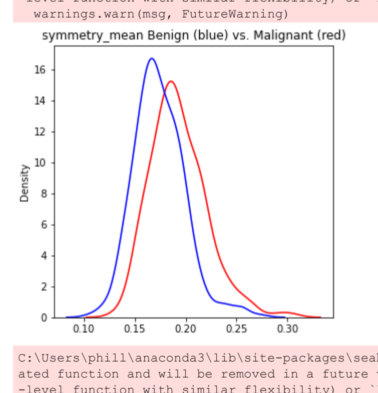

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



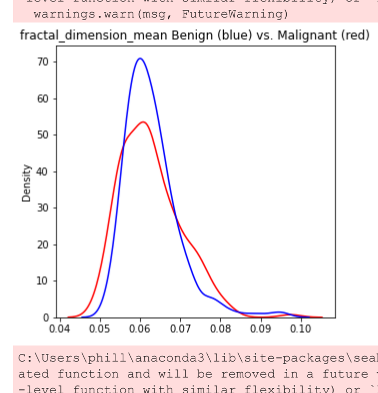

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



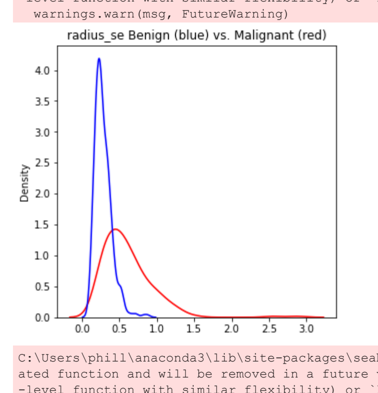

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



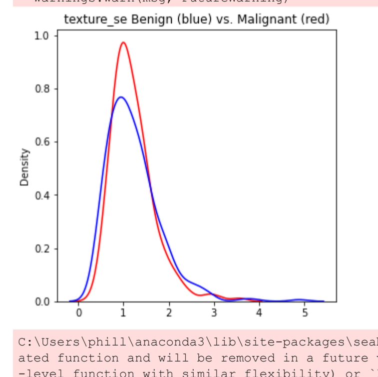

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



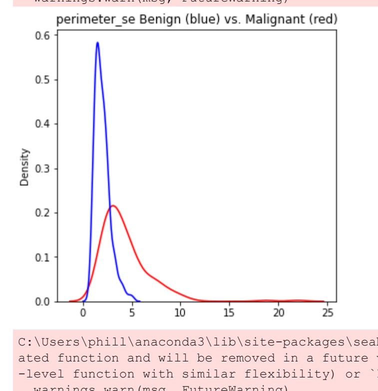

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



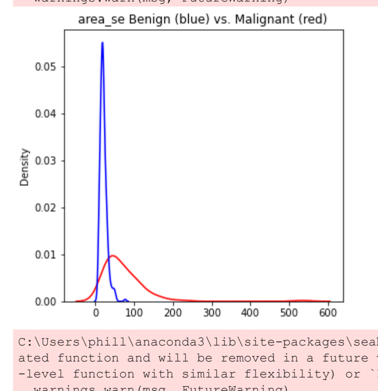

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



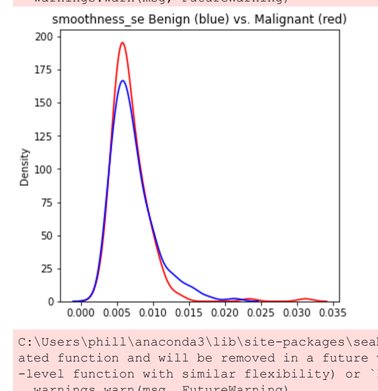

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



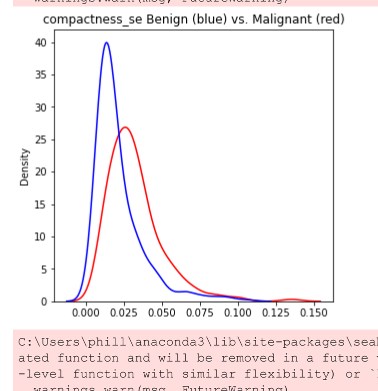

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



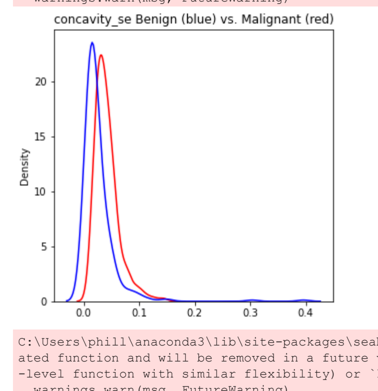

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



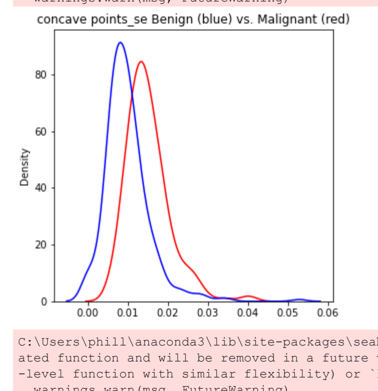

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



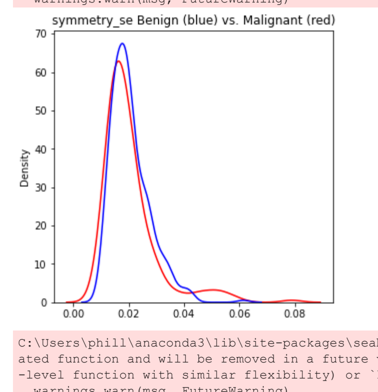

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



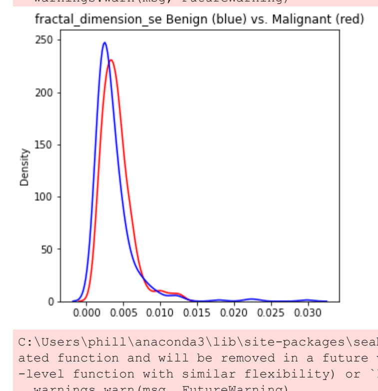

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



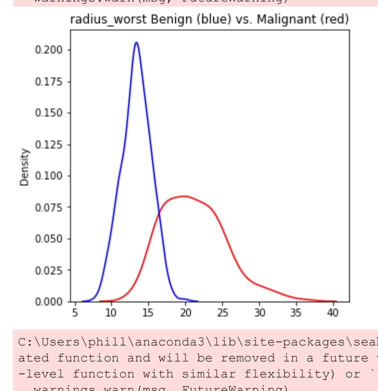

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



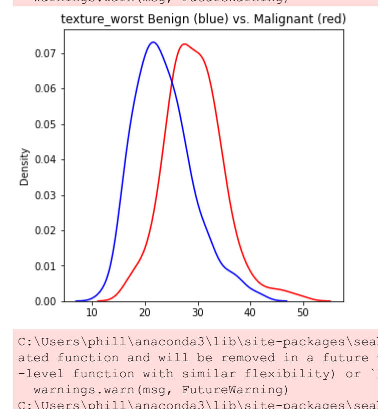

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



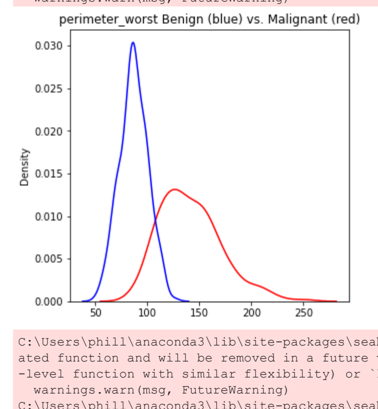

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



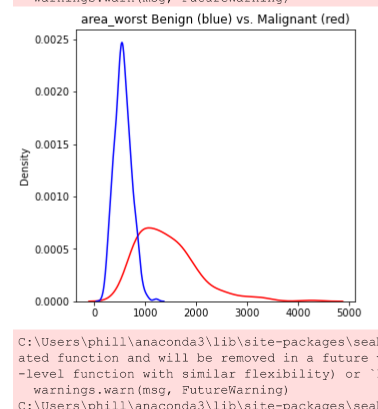

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



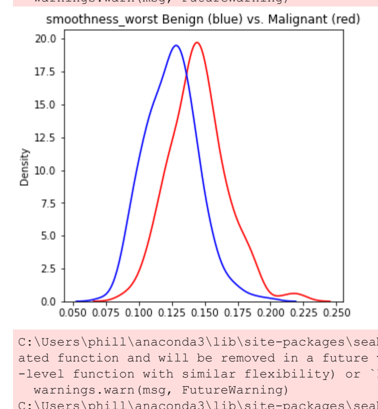

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



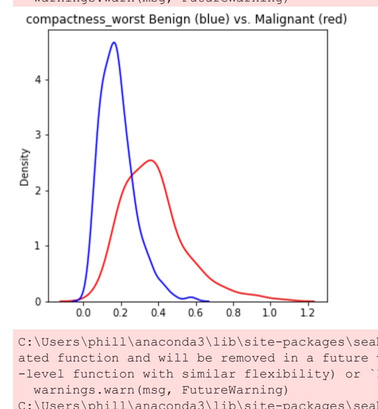

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



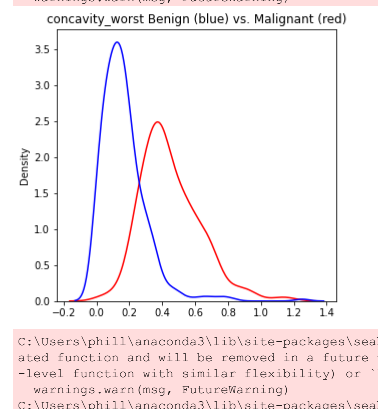

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



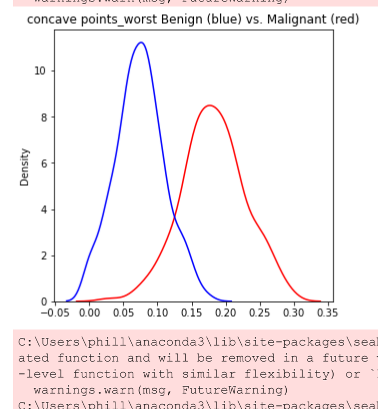

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



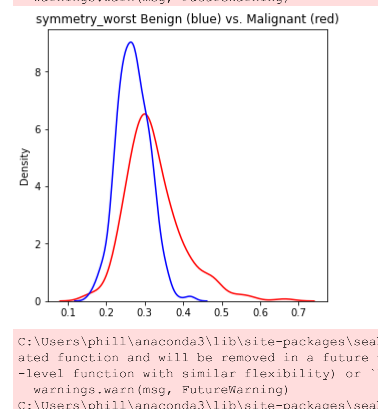

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



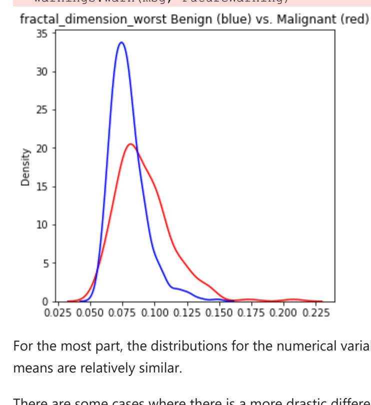

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```




```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```




```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\phill\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprec
ated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



For the most part, the distributions for the numerical variables between the diagnosis categories are relatively equal, i.e. they overlap. Their means are relatively similar.

There are some cases where there is a more drastic difference in the skew and distribution means, but there is nothing concerning that needs to be taken care of.

Encoding for Target Variable

In [21]:

```
#diagnosis
cat_cols = ['diagnosis']

#turning categories into their numerical counterparts using LabelEncoder
for var in cat_cols:
    number = LabelEncoder()
    tissue_pd[var+"cat"] = number.fit_transform(tissue_pd[var].astype('str'))
```

In [22]:

```
tissue_pd[['diagnosis','diagnosiscat']]
```

Out[22]:

	diagnosis	diagnosiscat
0	M	1
1	M	1
2	M	1
3	M	1
4	M	1
...	...	...
564	M	1
565	M	1
566	M	1
567	M	1
568	B	0

569 rows × 2 columns

In [23]:
```
tissue_pd[['diagnosis','diagnosiscat']].value_counts()
```
Out[23]:
```
diagnosis  diagnosiscat
B          0               357
M          1               212
dtype: int64
```
B = benign = 0

M = malignant = 1

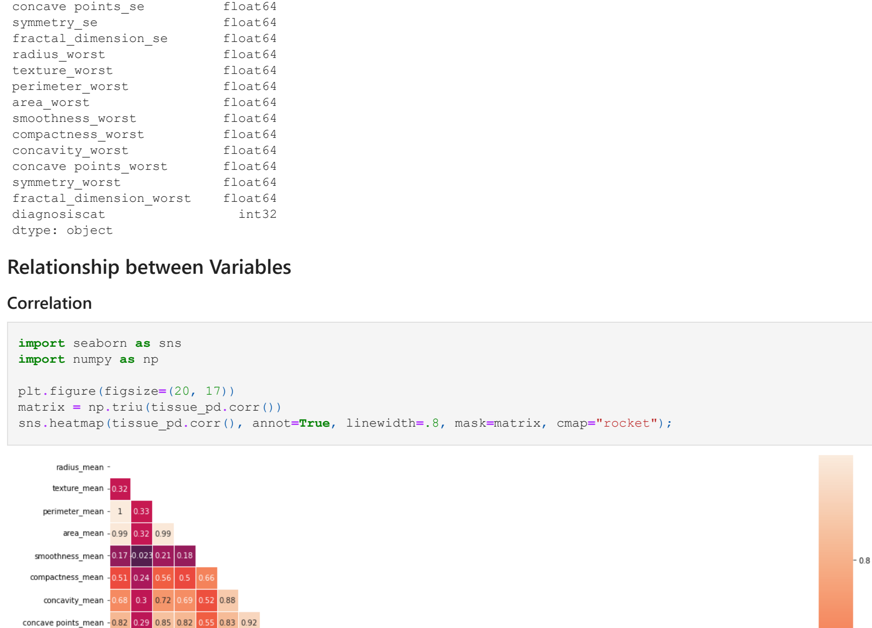
Now, we can remove the original 'diagnosis' column, since we'll be using the cat column for modeling purposes.

I am also going to remove the 'id' column before we move forward in our analysis, since it is not very useful for our data exploration or modeling.

In [24]:
```
#dropping diagnosis column
tissue_pd = tissue_pd.drop(['diagnosis','id'],axis='columns')
```
In [25]:
```
tissue_pd.dtypes
```
Out[25]:
```
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
diagnosiscat                 int32
dtype: object
```

Relationship between Variables

Correlation

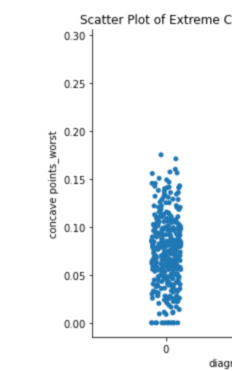
In [26]:
```
import seaborn as sns
import numpy as np

plt.figure(figsize=(20, 17))
matrix = np.triu(tissue_pd.corr())
sns.heatmap(tissue_pd.corr(), annot=True, linewidth=.8, mask=matrix, cmap="rocket");
```


There are quite a few relationships that feature high positive correlations between columns/variables.

We will have to investigate these to ensure that there is no collinearity in our model. Collinearity is when there is a correlation between predictor variables, such that they express a linear relationship in our regression model. When there is a correlation between predictor variables, it takes away from their ability to predict the outcome variable.

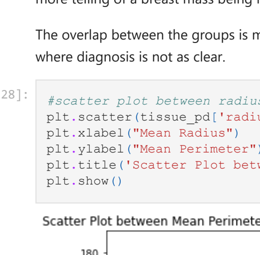
Some of these high positive correlations make sense given that some of these measures are derived from others in the dataset. For example, the perimeter of a circle is calculated by using the circle's radius ... 2*pi*radius.

Therefore, it may turn out that we don't need all of these measures in predicting our target variable since there are relationships between our predictors in how they get measured.

In [27]:
```
#scatter plot between concave points_worst & diagnosiscat
sns.catplot(x="diagnosiscat", y="concave points_worst", data=tissue_pd).set(title="Scatter Plot of Extreme Conc
```
Out[27]:
```
<seaborn.axisgrid.FacetGrid at 0x1f98d691700>
```


When looking at the categorical scatter plot for 'concave points_worst' vs. 'diagnosiscat', it can be seen that there is some overlap between the two diagnosis categoricals for the numerical measure. However, despite the overlap, it seems that higher values of concave points are more telling of a breast mass being malignant than benign.

The overlap between the groups is more in the middle of the range, which makes sense given that there is most likely some grey area where diagnosis is not as clear.

In [28]:
```
#scatter plot between radius and perimeter
plt.scatter(tissue_pd['radius_mean'], tissue_pd['perimeter_mean'])
plt.xlabel("Mean Radius")
plt.ylabel("Mean Perimeter")
plt.title('Scatter Plot between Mean Perimeter and Mean Radius of the Breast Masses')
plt.show()
```


A high (almost perfect) positive correlation at 0.998!

Using VIF for Feature Reduction

VIF or Variance-Inflation Factor will determine if collinearity exists among variables. In the below function, our bounday for feature removal based on VIF is set to 5.0, which would define whether any features are correlated with each other.

In [29]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
#features --> all variables except for diagnosiscat
x=tissue_pd.drop(['diagnosiscat'],axis=1)
#target variable --> diagnosiscat
y= tissue_pd['diagnosiscat']
```
In [30]:
```
#function for calculating the VIF score for each feature to determine if collinearity exists among variable
#finding values with large absolute values greater than 5
#function from https://medium.com/analytics-vidhya/feature-selection-techniques-2614b3b7efcd
def cal_vif(x):
    #threshold for VIF score
    thresh = 5
    output = pd.DataFrame()
    #number of columns
    k = x.shape[1]
    #calculate the VIF for the values in each column in the features
    vif = [variance_inflation_factor(x.values,i) for i in range(x.shape[1])]
    for i in range(1,k):
        print('Iteration no ',i)
        print(vif)
        #determine the feature with max VIF
        a = np.argmax(vif)
        print('Max vif is for variable no : ',a)
        #if max is not less than the threshold of 5.0, break and keep looking
        if(vif[a]<=thresh):
            break
        #else drop features and keep searching ... recursive
        if(i==1):
            output=x.drop(x.columns[a],axis=1)
            vif = [variance_inflation_factor(output.values,j)for j in range(output.shape[1])]
        elif(i>1):
            output = output.drop(output.columns[a],axis=1)
            vif = [variance_inflation_factor(output.values,j) for j in range(output.shape[1])]
    return(output)
```
In [31]:
```
selected_features = cal_vif(x)
selected_features.head()
```
```
Iteration no  1
[63306.17203588469, 251.04710784755466, 58123.58607886575, 1287.262339203524, 393.3981662424654, 200.9803539284
4344, 157.85504571106995, 154.2412682153576, 184.42655845792981, 629.6798743512951, 236.6657383805606, 24.67536
6821420543, 211.3963335352582, 72.46646814624141, 26.17024265526989, 44.919650577729094, 33.24409888132221, 53.
698656024401444, 37.17645208805795, 27.532630871224278, 9674.74260159131, 343.00438749768904, 4487.781269608321,
1138.7592521939323, 375.5971554566038, 132.88427640334123, 86.31036207218906, 148.67317987683427, 218.919805359
00996, 423.3967228281097]
Max vif is for variable no :  0
Iteration no  2
[250.98621003367228, 6937.433971823568, 1203.450804839675, 383.73348193052954, 118.91511263459785, 147.82643700
107937, 154.24120696552032, 182.88685591603095, 583.7647370674664, 235.34901585689187, 24.574000749094292, 202.
46795593263175, 72.16927132330092, 26.080058722769973, 44.915519460554684, 32.25716122629379, 53.1319706450126,
37.17472404694255, 27.292652573243558, 7573.943486033555, 342.54028857308043, 4189.475255886647, 1033.384833956
8924, 370.0860981925036, 128.83006643673826, 86.31030483624608, 148.67303309546642, 218.61063488049024, 422.406
10522502675]
Max vif is for variable no :  19
Iteration no  3
[248.21965797882058, 3901.901687119607, 882.1209697698505, 383.5985486450557, 114.65428972567135, 145.161895141
84506, 154.23870913583764, 178.65168402760412, 582.5929878671226, 121.23596422562281, 24.26729885711652, 177.15
501553486743, 58.6882323623932, 26.024460717870003, 44.852571447764504, 31.77458048510902, 52.27435659405065, 3
7.02841073997059, 26.950849697489957, 336.2148412371038, 3447.2223961910095, 571.2461629928926, 368.66638058940
435, 127.65108330062252, 86.25409282081111, 144.53971087229647, 213.82861419803842, 412.4922540214881]
Max vif is for variable no :  1
Iteration no  4
[246.0571329205338, 153.36052613324887, 381.8997535802432, 113.33438105184335, 144.95894610904634, 154.19931565
841853, 176.99418594074766, 557.5864435767686, 111.40985719884645, 24.01200555503391, 107.38889456876808, 52.48
879620041844, 25.46584174266479, 43.9879892081381, 31.197258134058856, 47.4606824188052, 36.67972301733117, 26.
2969150492185, 334.6108226675487, 668.3854404127386, 140.07451554484314, 368.4008028212317, 123.84128908703855,
85.7050294285516, 141.1401841893209, 211.469572804303, 411.60350719464356]
Max vif is for variable no :  19
Iteration no  5
[246.04322603965355, 83.90135786537718, 372.138153298496, 112.12707870230811, 144.91541799702716, 149.996245061
82326, 176.58922127189564, 508.08682464149285, 110.18904877728991, 23.625221271275194, 93.70634984934802, 35.95
6317956442575, 25.08372986787585, 43.287791764494635, 31.016624688143644, 47.4480767781785, 35.85486877059396,
26.017151653316795, 332.07541939040897, 76.95267223229258, 368.1154555277353, 123.5592172745276, 85.63818088475
915, 137.87771013644812, 208.9540396921155, 407.9663609053916]
Max vif is for variable no :  7
Iteration no  6
[242.10127746070486, 83.14445786198819, 324.46053109482114, 106.28326483027008, 142.8149004191429, 140.72471231
904638, 170.48479079984907, 109.30038592778098, 23.582202235086243, 91.03637812071628, 35.08353921194774, 25.06
912821478428, 41.72269757245191, 30.573093370269433, 47.37359814127844, 35.60548601772187, 24.729140171835102,
330.02372391012636, 76.63432662275619, 368.0533791867144, 105.42025062827265, 82.12397520599005, 137.4001620850
3013, 208.68244316540085, 190.62377696991288]
Max vif is for variable no :  19
Iteration no  7
[231.83708437667104, 82.1091540118899, 140.40526924826375, 106.26764486844908, 142.50270028870742, 137.01547980
765974, 167.19417521708195, 109.24402561336616, 22.874377809625415, 91.0323145479591, 35.03088812911213, 13.725
19606211277, 41.48238402033323, 30.57235196317352, 46.26013972264063, 34.23236849199917, 24.61930122444761, 30
9.54444960438434, 75.9860704842412, 105.31792988624267, 82.11382226682723, 126.13813492325608, 201.750876095906
02, 184.706777749357]
Max vif is for variable no :  17
Iteration no  8
[33.367214208242885, 79.44157701252354, 139.54167255709675, 106.06749162612384, 142.47784751716685, 136.3219755
448718, 156.85470228893126, 107.96361650926293, 12.447580396467746, 90.90702535280624, 34.85194873069072, 13.72
4792295177277, 41.434028936004566, 30.5266521395248, 44.97711436074712, 28.99988978183961, 24.607219285883797,
73.88163657828615, 105.31096332680872, 82.09369366376225, 120.7182274878607, 171.0062275818394, 184.67972071700
538]
Max vif is for variable no :  22
Iteration no  9
[32.13009426016344, 79.1285532838792, 98.16308547078732, 105.89476459348047, 142.41716726145484, 131.6305137965
9382, 155.4720304214451, 106.61473647084982, 12.44241887882756, 89.34110787700384, 34.845609008462134, 13.64463
7487347204, 39.033664795773596, 30.195514808672705, 44.83740313152744, 28.358229731273074, 12.900164271478465,
73.84171643152304, 92.00048117699147, 80.44781011844637, 120.59151611799894, 167.30971478504884]
Max vif is for variable no :  21
Iteration no  10
[31.77393129592127, 77.56996894006006, 97.02533194601766, 102.49033225890096, 142.29904340088856, 130.891272849
00542, 80.68109602547547, 106.4488930010305, 11.78339107305393, 89.30278999886823, 34.790995287872796, 13.41584
3606701372, 38.13139748318829, 29.51361350093057, 41.70499490776381, 14.477121122310594, 12.897678723297531, 7
1.23043057253574, 83.73624194379289, 80.06601064042769, 108.7720698543494]
Max vif is for variable no :  4
Iteration no  11
[31.682451621627802, 77.56445555938342, 88.98215432131022, 87.80817270392993, 56.347652567577406, 80.3945225539
4552, 104.99215955661566, 11.734238095799814, 88.99951449567969, 33.32013422777142, 13.141162261624869, 37.5082
3680844939, 22.338150736808608, 40.09333611432618, 14.477097781897545, 12.876679125099447, 70.32872425788786, 8
0.86536116895304, 53.57757249640807, 101.42124745600762]
Max vif is for variable no :  6
Iteration no  12
[31.54577630662006, 70.33014588863186, 88.20208064292635, 85.27089146937834, 53.71828898314472, 79.598710673675
67, 11.733113099801406, 46.36712636032097, 28.237511711650757, 13.082823181351941, 37.10443513719668, 22.255039
635672098, 40.077075208142524, 14.437480127483449, 12.390313510987534, 64.2399829932725, 80.79989535800676, 53.
3061323789551, 100.94649021325061]
Max vif is for variable no :  18
Iteration no  13
[31.422379439414993, 69.77137153239482, 86.99658368431041, 82.12511706996204, 39.28262824645959, 79.37813818851
046, 11.44077875638705, 45.48048708609361, 28.236515270432896, 12.915550873465158, 34.66124231759811, 21.192744
673390614, 23.747297340959015, 14.433150866043077, 12.39021890697013, 60.55961840470039, 67.9192439560803, 49.5
709817779003]
Max vif is for variable no :  2
Iteration no  14
[31.178527979951216, 68.06416415186425, 74.72314541276282, 38.35163651814377, 46.832291445745355, 11.4048645288
60662, 45.1192532535308, 28.196947633228287, 10.610302144505804, 30.644778222961037, 21.113316576166067, 23.006
880990841033, 14.36769820844603, 12.158571974020356, 60.2830453204909, 67.80843060136758, 49.55746551118662]
Max vif is for variable no :  2
Iteration no  15
[31.170855129255603, 67.47169344522399, 21.166226358370707, 41.11472390248302, 11.351261639512524, 43.991993963
545454, 27.949014607294, 10.551779169338836, 30.2643315206203, 20.946760077683894, 22.805909632990748, 14.36488
9036874546, 12.020198624953927, 59.608065018256454, 50.89341404598403, 47.42716301324604]
Max vif is for variable no :  1
Iteration no  16
[29.286358058908437, 18.85355454480084, 39.83517386996136, 11.239704497952681, 43.80459424927396, 27.9175269985
3354, 10.505313302080982, 29.439216449861398, 20.943134274421997, 22.805421345362305, 14.30110302434907, 11.779
946111072201, 23.86624465545877, 49.02308700997905, 47.351510827790555]
Max vif is for variable no :  13
Iteration no  17
[29.146601627553583, 18.834352979802553, 36.16286259917289, 11.026261527508575, 43.72833047786977, 27.820802888
88249, 10.361470490058048, 17.6179359760264, 12.783149013986316, 22.78496760956977, 14.209604629301607, 11.7405
17891719836, 23.7377537717185, 17.147157619910622]
Max vif is for variable no :  4
Iteration no  18
[29.065384453434593, 18.762893785653226, 36.0757931560618, 10.71946100189814, 7.0681231477979996, 10.1891698420
44262, 17.392890018081893, 12.258732617999728, 19.87086654611838, 13.708537103258227, 11.738689150476675, 23.22
8122342133048, 17.137049545563247]
Max vif is for variable no :  2
Iteration no  19
[23.709901129257826, 18.694802567455845, 10.614010519381022, 6.790823917119633, 9.57086183432995, 16.6339339577
8775, 11.582414907925102, 18.844003894137888, 9.383644442394147, 11.18442864611032, 22.594437592436297, 15.8716
97424889563]
Max vif is for variable no :  0
Iteration no  20
[15.73505173007278, 7.479449928876036, 6.164011942556995, 9.00126032784088, 16.59678918448984, 10.8517508751402
58, 18.16312090582923, 9.01873185573241, 11.15176371334831, 14.915564716727543, 13.484583340867935]
Max vif is for variable no :  6
Iteration no  21
[13.301158135648286, 7.387053184796467, 5.93767293353104, 8.415413255404443, 15.728368747925318, 8.061016185463
895, 8.94278375501195, 11.120758514043482, 14.35946152834649, 12.22440363507685]
Max vif is for variable no :  4
Iteration no  22
[13.117438144441206, 7.38653543972499, 5.871543698869255, 8.414719633040031, 7.623810630009888, 8.6763393327679
36, 7.732265068765098, 13.976625992787914, 11.178138029316086]
Max vif is for variable no :  7
Iteration no  23
[11.176654130350768, 7.10356618382093, 3.8581608934712355, 8.383688175734253, 6.992323779365014, 8.652297855142
976, 7.676302444267605, 9.853676377710157]
Max vif is for variable no :  0
Iteration no  24
[7.103564094635488, 2.1907615414061654, 8.347756771541066, 6.666504157385699, 8.648327614210078, 7.644681332828
403, 4.620457631474028]
Max vif is for variable no :  4
Iteration no  25
[5.817195921818661, 2.179316366339156, 6.987345633966846, 6.666419752137315, 7.551747392596707, 4.5586701739638
27]
Max vif is for variable no :  4
Iteration no  26
[5.7084092834529025, 2.156862544529822, 6.005778935127315, 4.2356537773736385, 4.558435525931816]
Max vif is for variable no :  2
Iteration no  27
[2.393834906513735, 2.1479539467716973, 4.0832796083399865, 4.553356915544563]
Max vif is for variable no :  3
```
Out[31]:

	texture_se	area_se	concavity_se	concavity_worst
0	0.9053	153.40	0.05373	0.7119
1	0.7339	74.08	0.01860	0.2416
2	0.7869	94.03	0.03832	0.4504
3	1.1560	27.23	0.05661	0.6869
4	0.7813	94.44	0.05688	0.4000

By utilizing the VIF of the variables, we were able to remove/drop variables which had collinearity with each other. The importance of dropping these allows for us to have features which provide unique and indepentent information about the variance within the given dataset.

Through these iterations, all features were dropped except for the following: texture_se, area_se, concavity_se and concavity_worst.

We will look to use these in our models.

Model Building

In [32]:
```
df_model = tissue_pd[["texture_se","area_se","concavity_se","concavity_worst","diagnosiscat"]]
```
In [33]:
```
df_model.head()
```
Out[33]:

	texture_se	area_se	concavity_se	concavity_worst	diagnosiscat
0	0.9053	153.40	0.05373	0.7119	1
1	0.7339	74.08	0.01860	0.2416	1
2	0.7869	94.03	0.03832	0.4504	1
3	1.1560	27.23	0.05661	0.6869	1
4	0.7813	94.44	0.05688	0.4000	1

In [34]:
```
#matrices of features
X = df_model.drop(labels='diagnosiscat',axis=1)
y = df_model['diagnosiscat']
col=X.columns
```
In [35]:
```
#train, test, split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state= 10)
```
In [36]:
```
#normalize the data
df_model = (X-X.mean())/(X.max()-X.min())
df_model = pd.concat([df_model,y],axis=1)
```
In [37]:
```
#check to see if the data is normalized ok
df_model.head(5)
```
Out[37]:

	texture_se	area_se	concavity_se	concavity_worst	diagnosiscat
0	-0.068855	0.211175	0.055142	0.351207	1
1	-0.106735	0.063024	-0.033570	-0.024432	1
2	-0.095022	0.100286	0.016228	0.142341	1
3	-0.013449	-0.024481	0.062415	0.331239	1
4	-0.096259	0.101052	0.063097	0.102086	1

In [38]:
```
#Label encoder
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
y  = pd.DataFrame(y)
```
In [39]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix

np.random.seed()
forest = RandomForestClassifier(n_estimators=1000)
fit = forest.fit(X_train,y_train)
accuracy = fit.score(X_test,y_test)
predict = fit.predict(X_test)
cmatrix = confusion_matrix(y_test,predict)

print('Accuracy of Random Forest: %s'% "{0:.2%}".format(accuracy))
```
```
Accuracy of Random Forest: 92.11%
```
In [40]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

K-Nearest Neighbor

Straight forward pattern recognition model which allows the testing of several k values and leaf sizes to determine the best performance

In [41]:
```
# Decide what k should be for KNN
knn = KNeighborsClassifier()

k_range = list(range(1, 30))
leaf_size = list(range(1,30))
weight_options = ['uniform', 'distance']
algorithm = ['auto', 'ball_tree', 'kd_tree', 'brute']
param_grid = {'n_neighbors': k_range, 'leaf_size': leaf_size, 'weights': weight_options, 'algorithm': algorithm
```
In [42]:
```
rand_knn = RandomizedSearchCV(knn, param_grid, cv=10, scoring="accuracy", n_iter=100, random_state=42)
rand_knn.fit(X,y.values.ravel())
```
Out[42]:
```
RandomizedSearchCV(cv=10, estimator=KNeighborsClassifier(), n_iter=100,
                   param_distributions={'algorithm': ['auto', 'ball_tree',
                                                      'kd_tree', 'brute'],
                                        'leaf_size': [1, 2, 3, 4, 5, 6, 7, 8, 9,
                                                      10, 11, 12, 13, 14, 15,
                                                      16, 17, 18, 19, 20, 21,
                                                      22, 23, 24, 25, 26, 27,
                                                      28, 29],
                                        'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8,
                                                        9, 10, 11, 12, 13, 14,
                                                        15, 16, 17, 18, 19, 20,
                                                        21, 22, 23, 24, 25, 26,
                                                        27, 28, 29],
                                        'weights': ['uniform', 'distance']},
                   random_state=42, scoring='accuracy')
```

Naive Bayes

Calculates the possibility of whether a data point belongs within a certain category or does not

In [43]:
```
gnb = GaussianNB()
gnb_scores = cross_val_score(gnb, X, y.values.ravel(), cv=10, scoring='accuracy')
```

Decision Tree Classifier

A decision tree is a supervised learning algorithm that performs strong in classification problems

In [44]:
```
dt_clf = DecisionTreeClassifier(random_state=42)

param_grid = {'max_features': ['auto', 'sqrt', 'log2'],
              'min_samples_split': sp_randint(2, 4),
              'min_samples_leaf': sp_randint(1, 4)}
```
In [45]:
```
rand_dt = RandomizedSearchCV(dt_clf, param_grid, cv=10, scoring="accuracy", n_iter=100, random_state=42)
rand_dt.fit(X,y)
```
Out[45]:
```
RandomizedSearchCV(cv=10, estimator=DecisionTreeClassifier(random_state=42),
                   n_iter=100,
                   param_distributions={'max_features': ['auto', 'sqrt',
                                                         'log2'],
                                        'min_samples_leaf': <scipy.stats._distn_infrastructure.rv_frozen object
at 0x000001F98ECF4550>,
                                        'min_samples_split': <scipy.stats._distn_infrastructure.rv_frozen objec
t at 0x000001F98D7A22E0>},
                   random_state=42, scoring='accuracy')
```

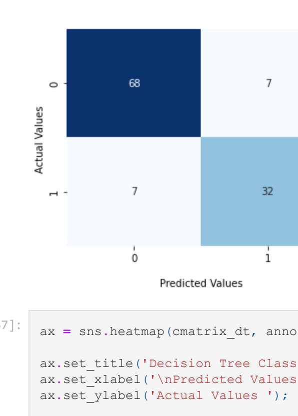
Support Vector Machine

A support vector machine (SVM) uses algorithms to train and classify data within degrees of polarity, which works well for complex data if a decision is needed beyond x/y

In [46]:
```
sv_clf = SVC(random_state=42)

param_grid = [
              {'C': [1, 10, 100, 1000],
               'kernel': ['linear']
              },
              {'C': [1, 10, 100, 1000],
               'gamma': [0.001, 0.0001],
               'kernel': ['rbf']
              },
 ]
```
In [47]:
```
grid_sv = GridSearchCV(sv_clf, param_grid, cv=10, scoring="accuracy")
grid_sv.fit(X,y.values.ravel())
```
Out[47]:
```
GridSearchCV(cv=10, estimator=SVC(random_state=42),
             param_grid=[{'C': [1, 10, 100, 1000], 'kernel': ['linear']},
                         {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001],
                          'kernel': ['rbf']}],
             scoring='accuracy')
```

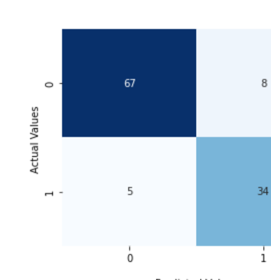
Random Forest Classifier

Expands beyond a decision tree by constructing multiple decision trees to remediate forcing a binary decision

In [48]:
```
rf_clf = RandomForestClassifier(random_state=42)

param_grid = {"max_depth": [3, None],
              "max_features":  sp_randint(1, 4),
              "min_samples_split": sp_randint(2, 11),
              "min_samples_leaf": sp_randint(1, 11),
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"]}
```
In [49]:
```
rand_rf = RandomizedSearchCV(rf_clf, param_distributions=param_grid, n_iter=100, random_state=42)
rand_rf.fit(X,y.values.ravel())
```
Out[49]:
```
RandomizedSearchCV(estimator=RandomForestClassifier(random_state=42),
                   n_iter=100,
                   param_distributions={'bootstrap': [True, False],
                                        'criterion': ['gini', 'entropy'],
                                        'max_depth': [3, None],
                                        'max_features': <scipy.stats._distn_infrastructure.rv_frozen object at
0x000001F98EE9FEE0>,
                                        'min_samples_leaf': <scipy.stats._distn_infrastructure.rv_frozen object
at 0x000001F98EE9F6D0>,
                                        'min_samples_split': <scipy.stats._distn_infrastructure.rv_frozen objec
t at 0x000001F990280B80>},
                   random_state=42)
```
In [50]:
```
#Summary of performance results
print("-"*100)
print("KNN Results")
print("Best Accuracy :",rand_knn.best_score_)
print("Best Parameters :",rand_knn.best_params_)
print("Best Estimator :",rand_knn.best_estimator_)
print("-"*100)
print("Naive Bayes Results")
print("Mean Accuracy :",gnb_scores.mean())
print("-"*100)
print("Decision Tree Classifier Results")
print("Best Accuracy :",rand_dt.best_score_)
print("Best Parameters :",rand_dt.best_params_)
print("Best Estimator :",rand_dt.best_estimator_)
print("-"*100)
print("Support Vector Machine Results")
print("Best Accuracy :",grid_sv.best_score_)
print("Best Parameters :",grid_sv.best_params_)
print("Best Estimator :",grid_sv.best_estimator_)
print("-"*100)
print("Random Forest Classification Results")
print("Best Accuracy :",rand_rf.best_score_)
print("Best Parameters :",rand_rf.best_params_)
print("Best Estimator :",rand_rf.best_estimator_)
print("-"*100)
```
```
----------------------------------------------------------------------------------------------------
KNN Results
Best Accuracy : 0.8928571428571429
Best Parameters : {'weights': 'distance', 'n_neighbors': 9, 'leaf_size': 29, 'algorithm': 'ball_tree'}
Best Estimator : KNeighborsClassifier(algorithm='ball_tree', leaf_size=29, n_neighbors=9,
                     weights='distance')
----------------------------------------------------------------------------------------------------
Naive Bayes Results
Mean Accuracy : 0.9121867167919799
----------------------------------------------------------------------------------------------------
Decision Tree Classifier Results
Best Accuracy : 0.9156641604010025
Best Parameters : {'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 3}
Best Estimator : DecisionTreeClassifier(max_features='auto', min_samples_split=3,
                       random_state=42)
----------------------------------------------------------------------------------------------------
Support Vector Machine Results
Best Accuracy : 0.9455513784461151
Best Parameters : {'C': 1000, 'kernel': 'linear'}
Best Estimator : SVC(C=1000, kernel='linear', random_state=42)
----------------------------------------------------------------------------------------------------
Random Forest Classification Results
Best Accuracy : 0.9350256171401956
Best Parameters : {'bootstrap': True, 'criterion': 'gini', 'max_depth': None, 'max_features': 3, 'min_samples_l
eaf': 1, 'min_samples_split': 2}
Best Estimator : RandomForestClassifier(max_features=3, random_state=42)
----------------------------------------------------------------------------------------------------
```
In [57]:
```
fit_knn = rand_knn.fit(X_train,y_train)
predict_knn = fit_knn.predict(X_test)
cmatrix_knn = confusion_matrix(y_test,predict_knn)

fit_gnb = gnb.fit(X_train,y_train)
predict_gnb = fit_gnb.predict(X_test)
cmatrix_gnb = confusion_matrix(y_test,predict_gnb)

fit_dt = rand_dt.fit(X_train,y_train)
predict_dt = fit_dt.predict(X_test)
cmatrix_dt = confusion_matrix(y_test,predict_dt)

fit_sv = grid_sv.fit(X_train,y_train)
predict_sv = fit_sv.predict(X_test)
cmatrix_sv = confusion_matrix(y_test,predict_sv)

fit_rf = rand_rf.fit(X_train,y_train)
predict_rf = fit_rf.predict(X_test)
cmatrix_rf = confusion_matrix(y_test,predict_rf)
```
In [58]:
```
#rows = actual, col = pred

#compute tp, tp_and_fn and tp_and_fp w.r.t all classes

#knn
tp_and_fn_knn = cmatrix_knn.sum(1)
tp_and_fp_knn = cmatrix_knn.sum(0)
tp_knn = cmatrix_knn.diagonal()

precision_knn = tp_knn / tp_and_fp_knn
recall_knn = tp_knn / tp_and_fn_knn

#naive bayes

tp_and_fn_gnb = cmatrix_gnb.sum(1)
tp_and_fp_gnb = cmatrix_gnb.sum(0)
tp_gnb = cmatrix_gnb.diagonal()

precision_gnb = tp_gnb / tp_and_fp_gnb
recall_gnb = tp_gnb / tp_and_fn_gnb

#decision tree
tp_and_fn_dt = cmatrix_dt.sum(1)
tp_and_fp_dt = cmatrix_dt.sum(0)
tp_dt = cmatrix_dt.diagonal()

precision_dt = tp_dt / tp_and_fp_dt
recall_dt = tp_dt / tp_and_fn_dt

#support vector
tp_and_fn_sv = cmatrix_sv.sum(1)
tp_and_fp_sv = cmatrix_sv.sum(0)
tp_sv = cmatrix_sv.diagonal()

precision_sv = tp_sv / tp_and_fp_sv
recall_sv = tp_sv / tp_and_fn_sv

#random forest

tp_and_fn_rf = cmatrix_rf.sum(1)
tp_and_fp_rf = cmatrix_rf.sum(0)
tp_rf = cmatrix_rf.diagonal()

precision_rf = tp_rf / tp_and_fp_rf
recall_rf = tp_rf / tp_and_fn_rf
```
In [62]:
```
print("-"*100)
print("KNN Confusion Matrix")
print(cmatrix_knn)
print('Precision: ',precision_knn)
print('Recall: ',recall_knn)
print("-"*100)
print("Gaussian Naive Bayes Confusion Matrix")
print(cmatrix_gnb)
print('Precision: ',precision_gnb)
print('Recall: ',recall_gnb)
print("-"*100)
print("Decision Tree Confusion Matrix")
print(cmatrix_dt)
print('Precision: ',precision_dt)
print('Recall: ',recall_dt)
print("-"*100)
print("Support Vector Machine Confusion Matrix")
print(cmatrix_sv)
print('Precision: ',precision_sv)
print('Recall: ',recall_sv)
print("-"*100)
print("Random Forest Confusion Matrix")
print(cmatrix_rf)
print('Precision: ',precision_rf)
print('Recall: ',recall_rf)
print("-"*100)
```
```
----------------------------------------------------------------------------------------------------
KNN Confusion Matrix
[[68  7]
 [ 7 32]]
Precision:  [0.90666667 0.82051282]
Recall:  [0.90666667 0.82051282]
----------------------------------------------------------------------------------------------------
Gaussian Naive Bayes Confusion Matrix
[[69  6]
 [ 6 33]]
Precision:  [0.92       0.84615385]
Recall:  [0.92       0.84615385]
----------------------------------------------------------------------------------------------------
Decision Tree Confusion Matrix
[[67  8]
 [ 5 34]]
Precision:  [0.93055556 0.80952381]
Recall:  [0.89333333 0.87179487]
----------------------------------------------------------------------------------------------------
Support Vector Machine Confusion Matrix
[[72  3]
 [ 2 37]]
Precision:  [0.97297297 0.925     ]
Recall:  [0.96       0.94871795]
----------------------------------------------------------------------------------------------------
Random Forest Confusion Matrix
[[71  4]
 [ 4 35]]
Precision:  [0.94666667 0.8974359 ]
Recall:  [0.94666667 0.8974359 ]
----------------------------------------------------------------------------------------------------
```
In [65]:
```
import seaborn as sns

ax = sns.heatmap(cmatrix_knn, annot=True, cmap='Blues')

ax.set_title('KNN Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Display the visualization of the Confusion Matrix.
plt.show()
```


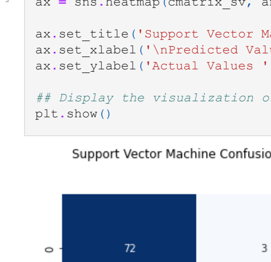
In [67]:
```
ax = sns.heatmap(cmatrix_dt, annot=True, cmap='Blues')

ax.set_title('Decision Tree Classifier Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Display the visualization of the Confusion Matrix.
plt.show()
```

Decision Tree Classifier Confusion Matrix



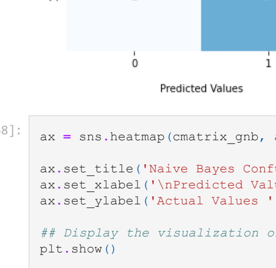
In [69]:
```python
ax = sns.heatmap(cmatrix_sv, annot=True, cmap='Blues')

ax.set_title('Support Vector Machine Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Display the visualization of the Confusion Matrix.
plt.show()
```

Support Vector Machine Confusion Matrix



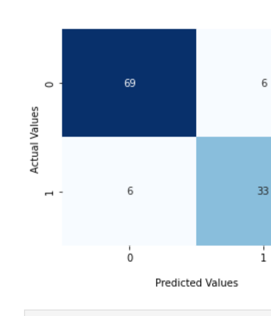
In [68]:
```python
ax = sns.heatmap(cmatrix_gnb, annot=True, cmap='Blues')

ax.set_title('Naive Bayes Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Display the visualization of the Confusion Matrix.
plt.show()
```

Naive Bayes Confusion Matrix



In [70]:
```python
ax = sns.heatmap(cmatrix_rf, annot=True, cmap='Blues')

ax.set_title('Random Forest Confusion Matrix\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Display the visualization of the Confusion Matrix.
plt.show()
```

Random Forest Confusion Matrix



Paper Information

Curvature

- deviations from a circle are problematic for malignancy
- can lead to quickly-dividing cells which usually points to cancer
- high or low pints are concerning

Larger values typically indicate a higher likelihood of malignancy

Radius

- averaging the length of the radial line segments defined by the centroid of the snake and the individual snake points

Perimeter

- total distance between snake points

Area

Compactness

- permiter and area are combined
- increases with the irregularity of boundaries

Smoothness

Concavity

- boundary inside of chords

Concave Points

Symmetry

Fractal Dimension

- higher value corresponds to a less regular contour ... higher probability of malignancy

Texture

- variance of the gray scale intensities

Extreme values are the most intuitively useful for the problem at hand, since only a few malignant cells may occur in a given sample