Breast Cancer EDA and Classification

In [4]:

```
# Ignoring Warnings
import warnings
warnings.filterwarnings('ignore')
```

In [5]:

```
# Importing Necessary Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:

```
# Downloading Dataset
df = pd.read_csv("git.csv")
```

In [7]:

```
# First Look at the data
df.head()
```

Out[7]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 33 columns

In [5]:

```
df.columns
```

Out[5]:

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mea
n',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_s
e',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_s
e',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

There are total 32 columns in this dataset

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [7]:

```
# Column which is unnamed
df['Unnamed: 32']
```

Out[7]:

```
0     NaN
1     NaN
2     NaN
3     NaN
4     NaN
       ..
564   NaN
565   NaN
566   NaN
567   NaN
568   NaN
Name: Unnamed: 32, Length: 569, dtype: float64
```

In [8]:

```
# Dropping COlumn
df = df.drop("Unnamed: 32", axis=1)
```

In [9]:

```
df.head()
```

Out[9]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 32 columns

In [10]:

```
df.columns
```

Out[10]:

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mea
n',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_s
e',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_s
e',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

Patient ID has no correlation with cancer so dropping column id

In [11]:

```
df.drop('id', axis=1, inplace=True)
# df = df.drop('id', axis=1)
```

In [12]:

```
df.columns
```

Out[12]:

```
Index(['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mea
n',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_s
e',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_s
e',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

In [13]:

```
type(df.columns)
```

Out[13]:

```
pandas.core.indexes.base.Index
```

In [14]:

```
# Taking list of remaining columns
l = list(df.columns)
print(l)
```

```
['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mea
n', 'smoothness_mean', 'compactness_mean', 'concavity_mean', 'concave poin
ts_mean', 'symmetry_mean', 'fractal_dimension_mean', 'radius_se', 'texture
_se', 'perimeter_se', 'area_se', 'smoothness_se', 'compactness_se', 'conca
vity_se', 'concave points_se', 'symmetry_se', 'fractal_dimension_se', 'rad
ius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smoothness_
worst', 'compactness_worst', 'concavity_worst', 'concave points_worst', 's
ymmetry_worst', 'fractal_dimension_worst']
```

In [15]:

```
features_mean = l[1:11]

features_se = l[11:21]

features_worst = l[21:]
```

In [16]:

```
print(features_mean)
```

```
['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness
_mean', 'compactness_mean', 'concavity_mean', 'concave points_mean', 'symm
etry_mean', 'fractal_dimension_mean']
```

In [17]:

```
print(features_se)
```

```
['radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se', 'c
ompactness_se', 'concavity_se', 'concave points_se', 'symmetry_se', 'fract
al_dimension_se']
```

In [18]:

```
print(features_worst)
```

```
['radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', 'smooth
ness_worst', 'compactness_worst', 'concavity_worst', 'concave points_wors
t', 'symmetry_worst', 'fractal_dimension_worst']
```

In [19]:

```
df.head(2)
```

Out[19]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
0	M	17.99	10.38	122.8	1001.0	0.11840	
1	M	20.57	17.77	132.9	1326.0	0.08474	

2 rows × 31 columns

In [20]:

```
df['diagnosis'].unique()
# M= Malignant, B= Benign
```

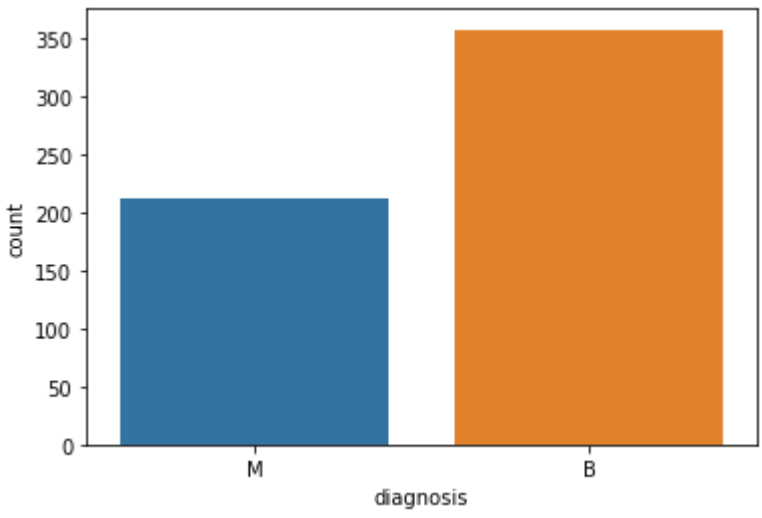
Out[20]:

```
array(['M', 'B'], dtype=object)
```

In [21]:

```
ax = sns.countplot(df['diagnosis'],label="Count")       # M = 212, B = 357
B, M = df['diagnosis'].value_counts()
print('Number of Benign: ',B)
print('Number of Malignant : ',M)
```

```
Number of Benign:  357
Number of Malignant :  212
```



In [22]:

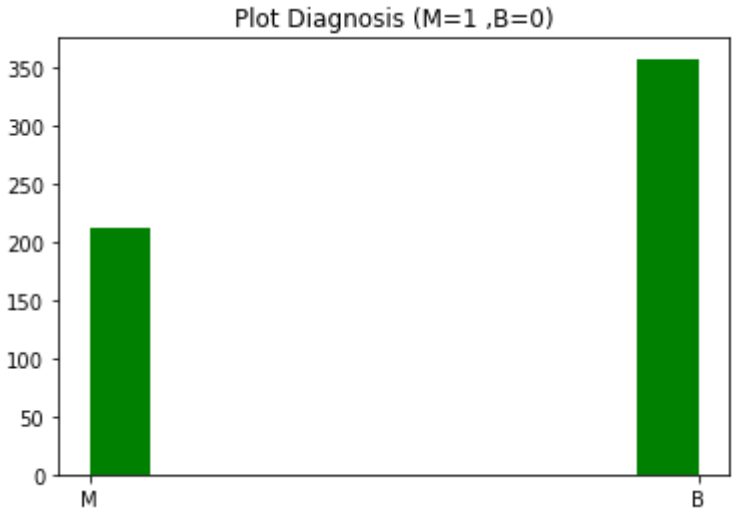
```
df.shape
```

Out[22]:

```
(569, 31)
```

In [23]:

```
plt.hist(df['diagnosis'],color='g')
plt.title('Plot Diagnosis (M=1 ,B=0)')
plt.show()
```



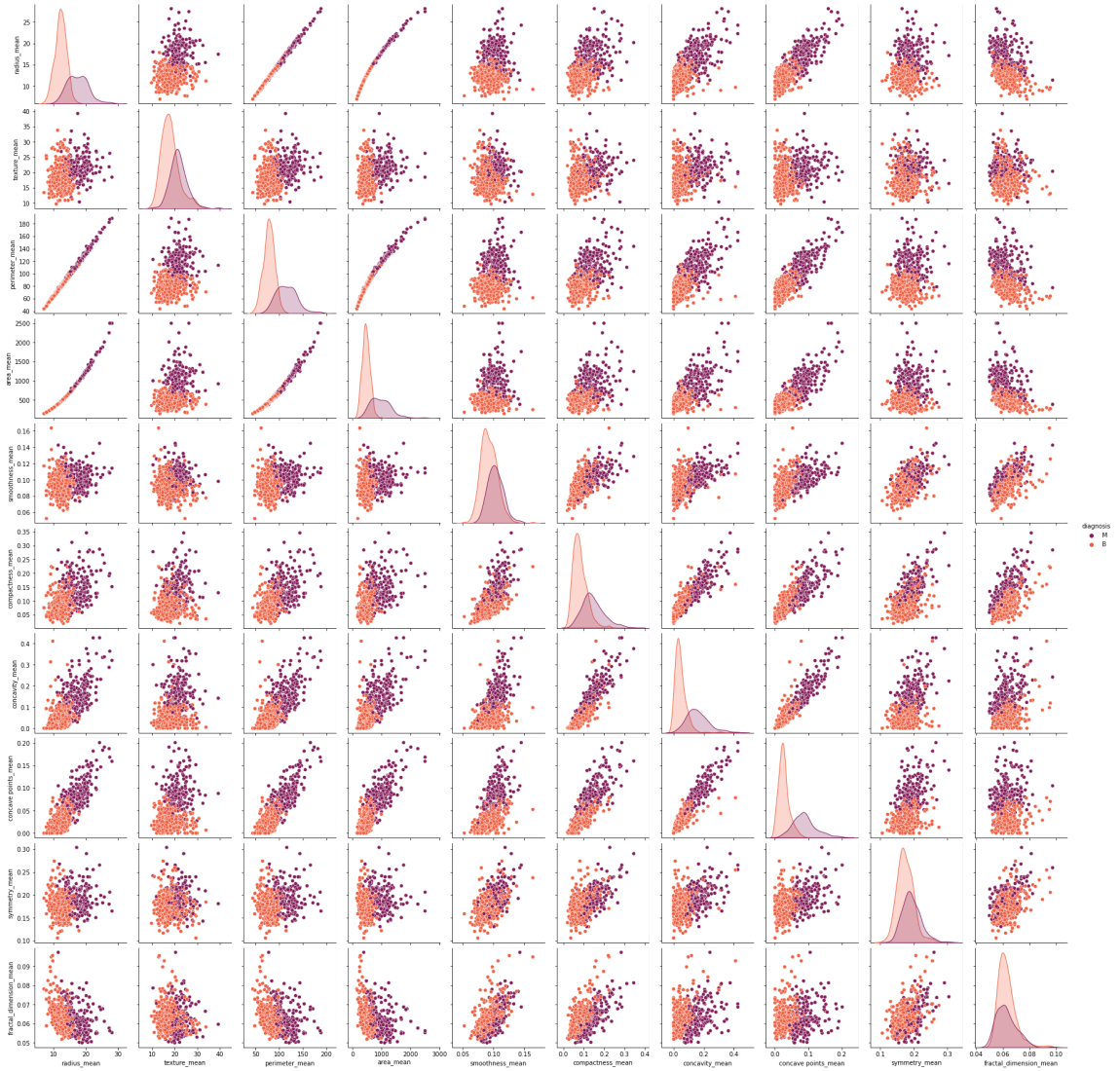
In [24]:

```
# generate a scatter plot matrix with the "mean" columns
cols = ['diagnosis',
        'radius_mean',
        'texture_mean',
        'perimeter_mean',
        'area_mean',
        'smoothness_mean',
        'compactness_mean',
        'concavity_mean',
        'concave points_mean',
        'symmetry_mean',
        'fractal_dimension_mean']

sns.pairplot(data=df[cols], hue='diagnosis', palette='rocket')
```

Out[24]:

```
<seaborn.axisgrid.PairGrid at 0x7f139e100100>
```



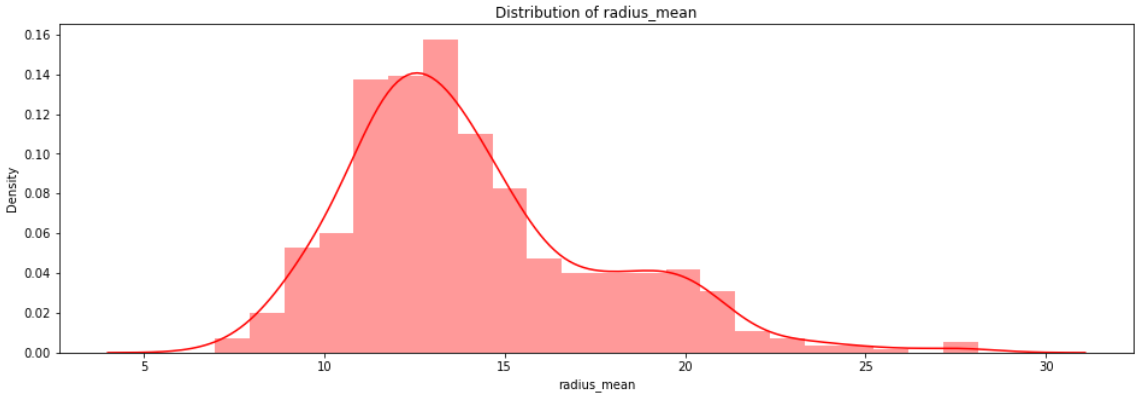
Distribution of radius_mean

In [25]:

```
f, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["radius_mean"], color="red",ax = axes)
plt.title("Distribution of radius_mean")
```

Out[25]:

```
Text(0.5, 1.0, 'Distribution of radius_mean')
```



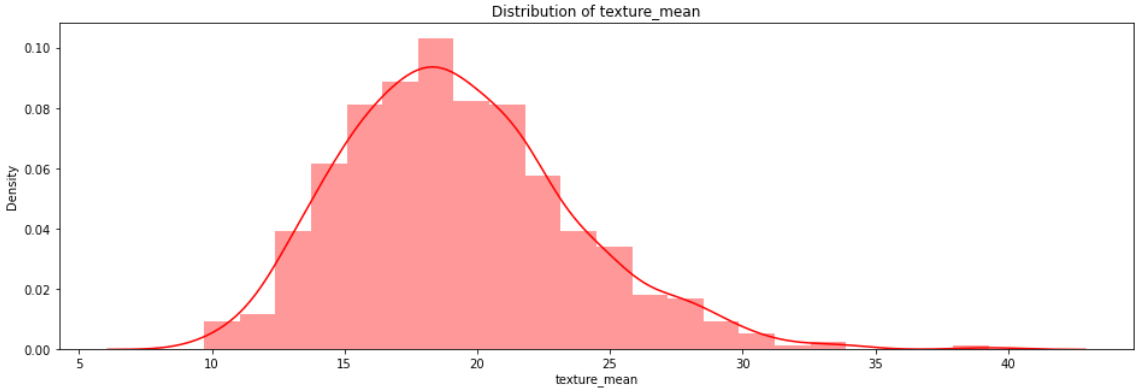
Distribution of texture_mean

In [26]:

```
f1, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["texture_mean"], color="red",ax = axes)
plt.title("Distribution of texture_mean")
```

Out[26]:

```
Text(0.5, 1.0, 'Distribution of texture_mean')
```



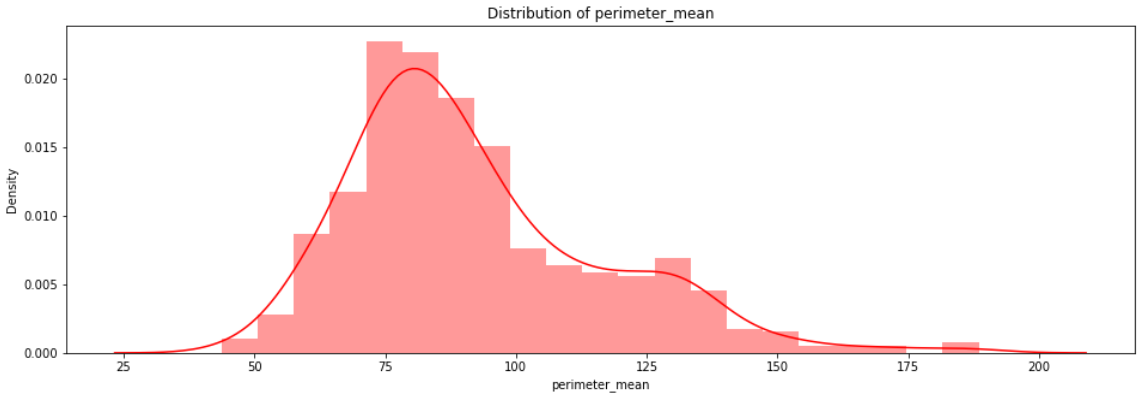
Distribution of perimeter_mean

In [27]:

```
f2, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["perimeter_mean"], color="red",ax = axes)
plt.title("Distribution of perimeter_mean")
```

Out[27]:

```
Text(0.5, 1.0, 'Distribution of perimeter_mean')
```



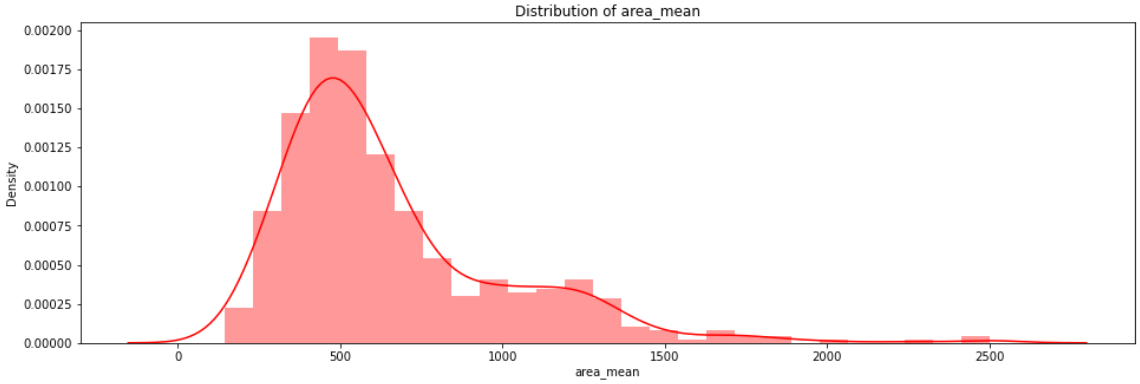
Distribution of area_mean

In [28]:

```
f3, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["area_mean"], color="red",ax = axes)
plt.title("Distribution of area_mean")
```

Out[28]:

```
Text(0.5, 1.0, 'Distribution of area_mean')
```



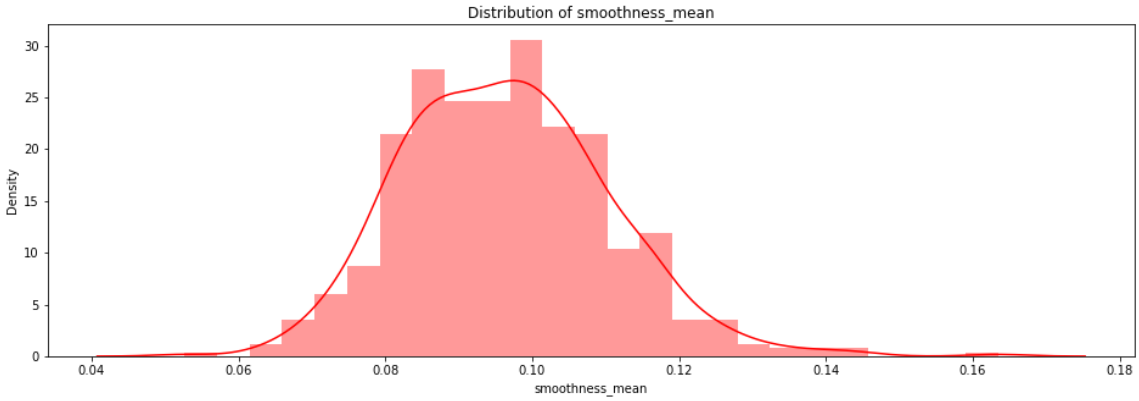
Distribution of smoothness_mean

In [29]:

```
f4, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["smoothness_mean"], color="red",ax = axes)
plt.title("Distribution of smoothness_mean")
```

Out[29]:

```
Text(0.5, 1.0, 'Distribution of smoothness_mean')
```



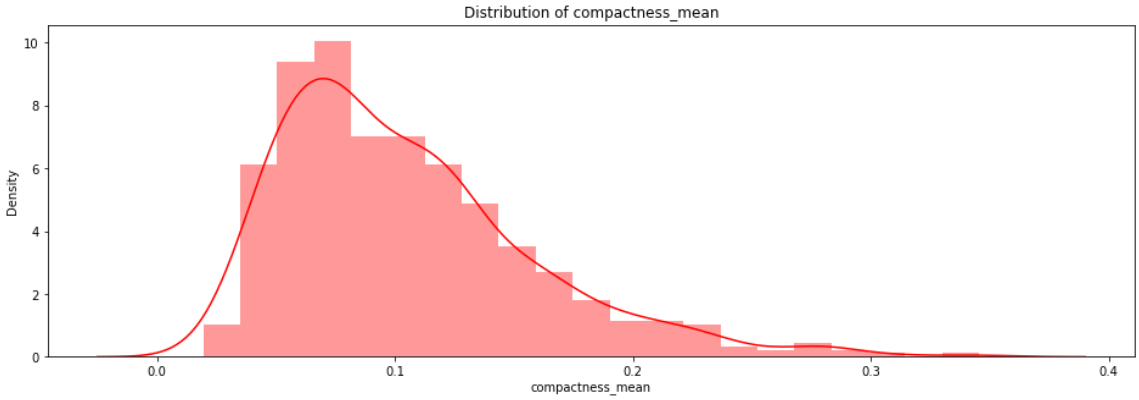
Distribution of compactness_mean

In [30]:

```
f5, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["compactness_mean"], color="red",ax = axes)
plt.title("Distribution of compactness_mean")
```

Out[30]:

```
Text(0.5, 1.0, 'Distribution of compactness_mean')
```



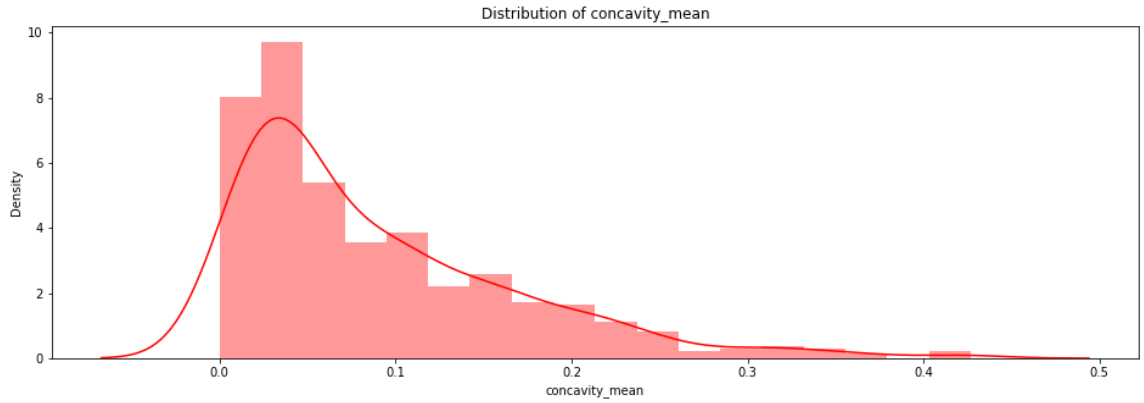
Distribution of concavity_mean

In [31]:

```
f6, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["concavity_mean"], color="red",ax = axes)
plt.title("Distribution of concavity_mean")
```

Out[31]:

```
Text(0.5, 1.0, 'Distribution of concavity_mean')
```



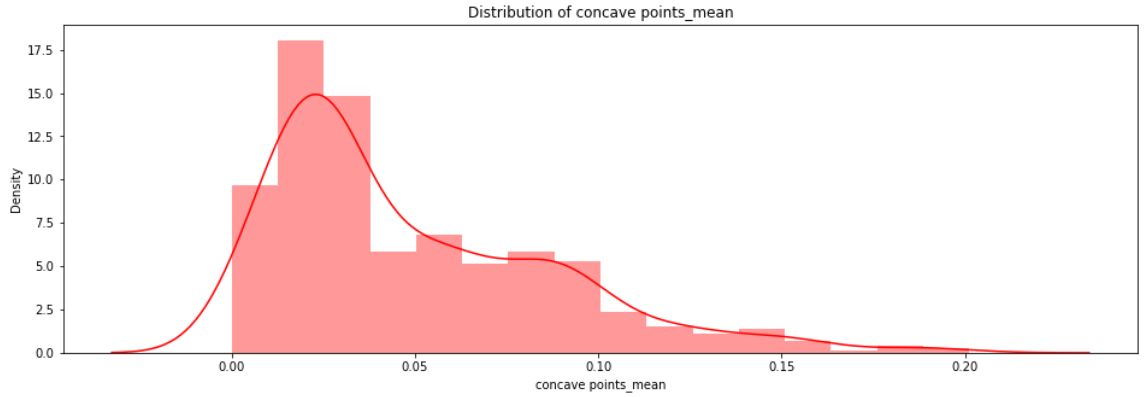
Distribution of concave points_mean

In [32]:

```
f7, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["concave points_mean"], color="red",ax = axes)
plt.title("Distribution of concave points_mean")
```

Out[32]:

```
Text(0.5, 1.0, 'Distribution of concave points_mean')
```



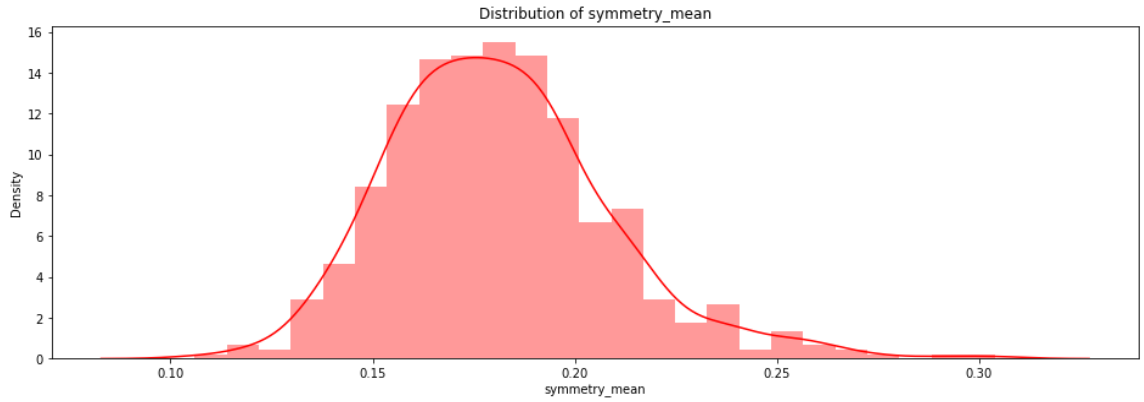
Distribution of symmetry_mean

In [33]:

```
f8, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["symmetry_mean"], color="red",ax = axes)
plt.title("Distribution of symmetry_mean")
```

Out[33]:

```
Text(0.5, 1.0, 'Distribution of symmetry_mean')
```



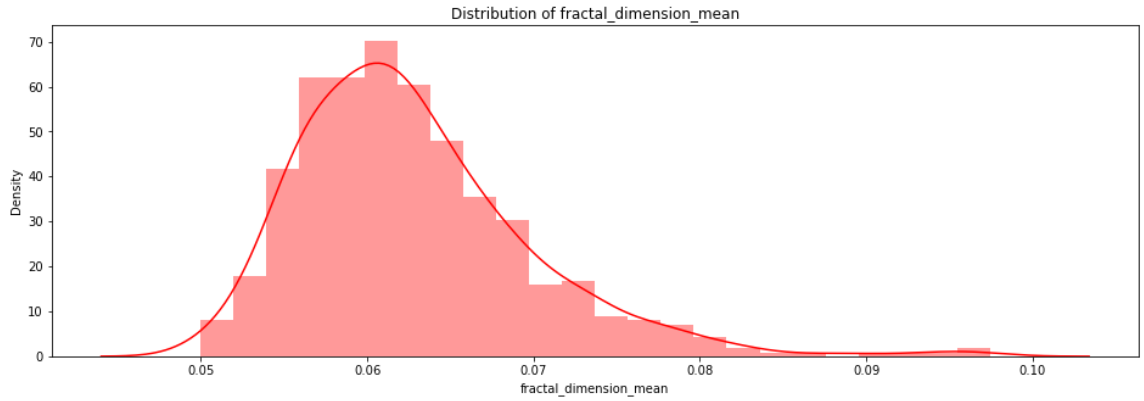
Distribution of fractal_dimension_mean

In [34]:

```
f9, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["fractal_dimension_mean"], color="red",ax = axes)
plt.title("Distribution of fractal_dimension_mean")
```

Out[34]:

```
Text(0.5, 1.0, 'Distribution of fractal_dimension_mean')
```



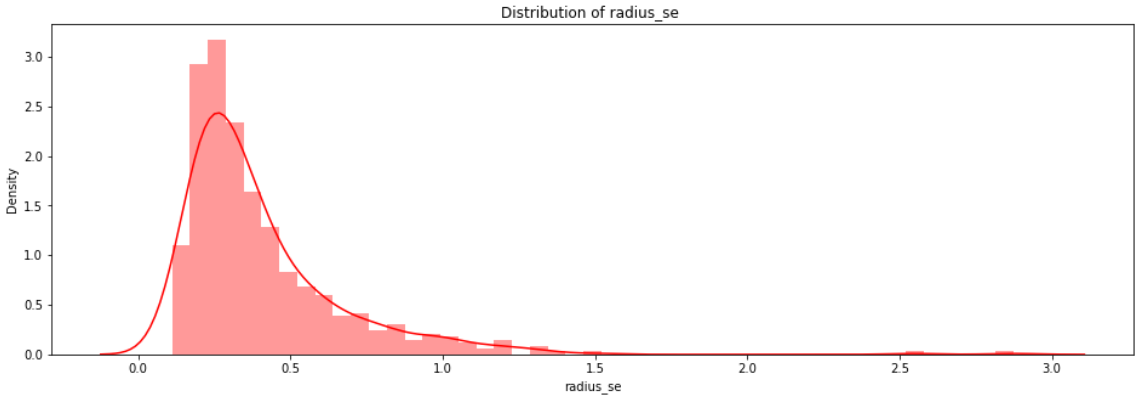
Distribution of radius_se

In [35]:

```
f11, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["radius_se"], color="red",ax = axes)
plt.title("Distribution of radius_se ")
```

Out[35]:

```
Text(0.5, 1.0, 'Distribution of radius_se ')
```



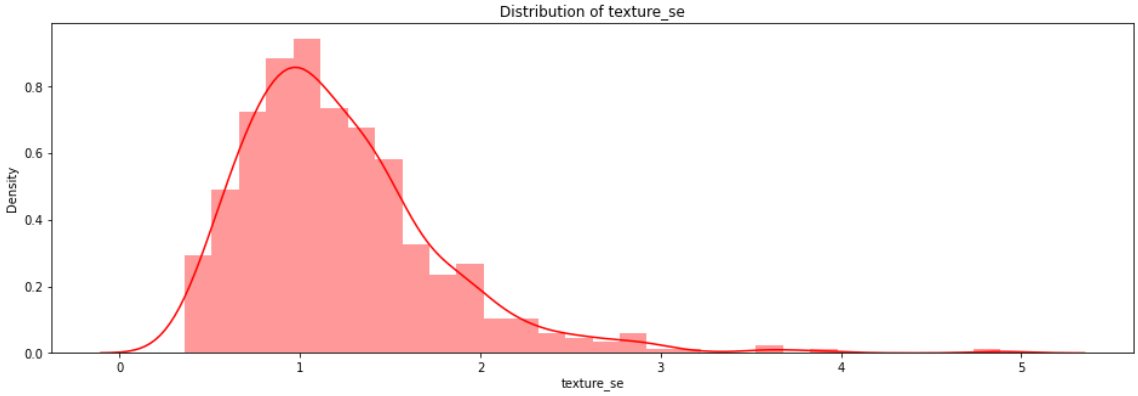
Distribution of texture_se

In [36]:

```
f12, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["texture_se"], color="red",ax = axes)
plt.title("Distribution of texture_se")
```

Out[36]:

```
Text(0.5, 1.0, 'Distribution of texture_se')
```



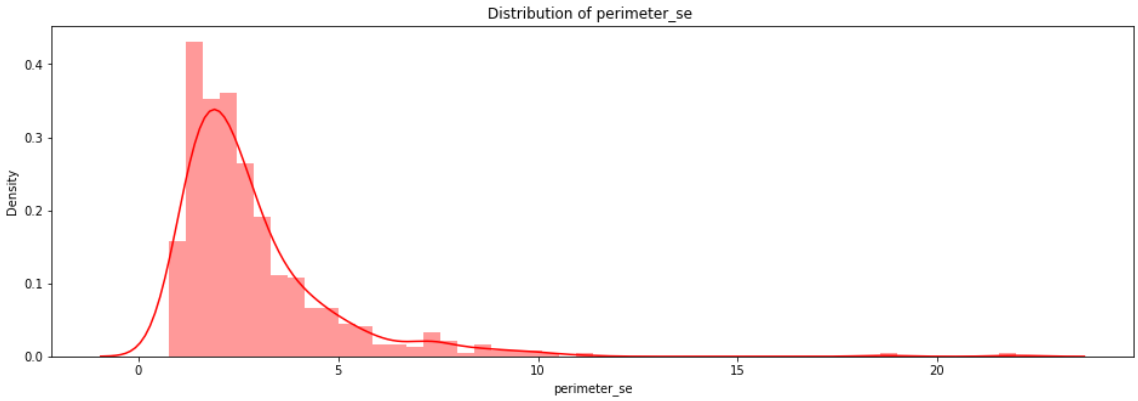
Distribution of perimeter_se

In [37]:

```
f13, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["perimeter_se"], color="red",ax = axes)
plt.title("Distribution of perimeter_se ")
```

Out[37]:

```
Text(0.5, 1.0, 'Distribution of perimeter_se ')
```



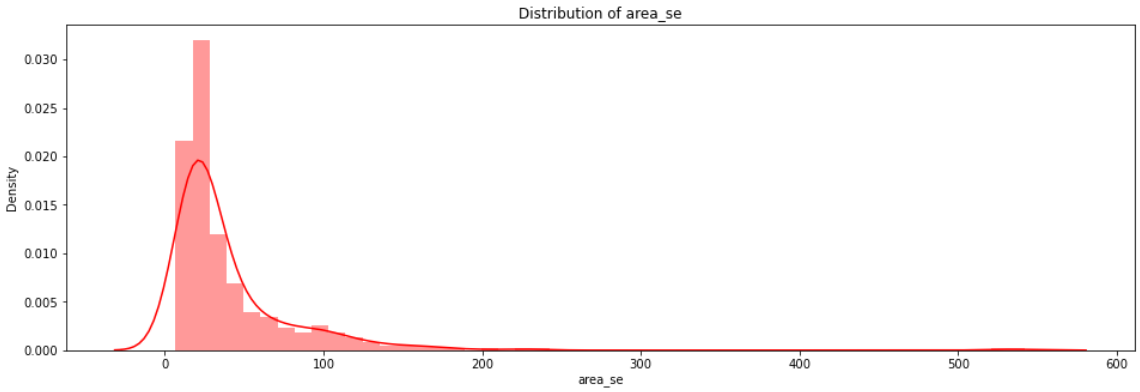
Distribution of area_se

In [38]:

```
f14, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["area_se"], color="red",ax = axes)
plt.title("Distribution of area_se")
```

Out[38]:

```
Text(0.5, 1.0, 'Distribution of area_se')
```



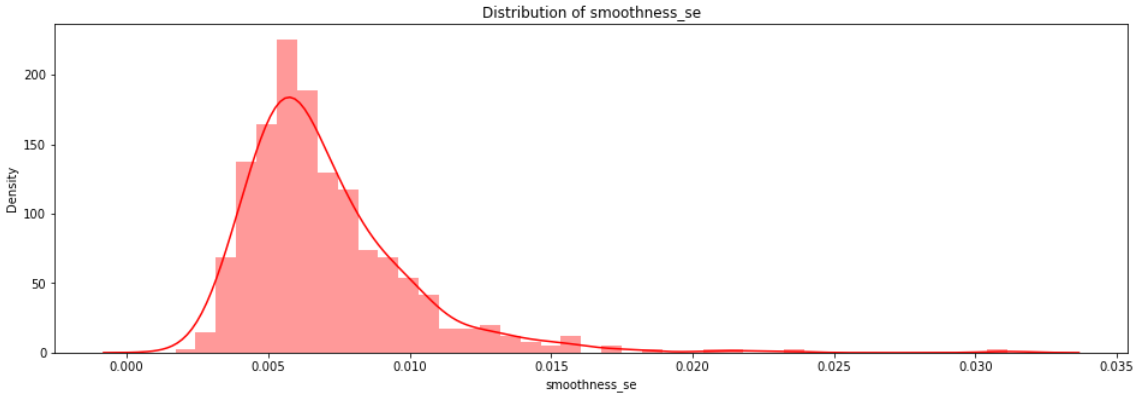
Distribution of smoothness_se

In [39]:

```
f15, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["smoothness_se"], color="red",ax = axes)
plt.title("Distribution of smoothness_se")
```

Out[39]:

```
Text(0.5, 1.0, 'Distribution of smoothness_se')
```



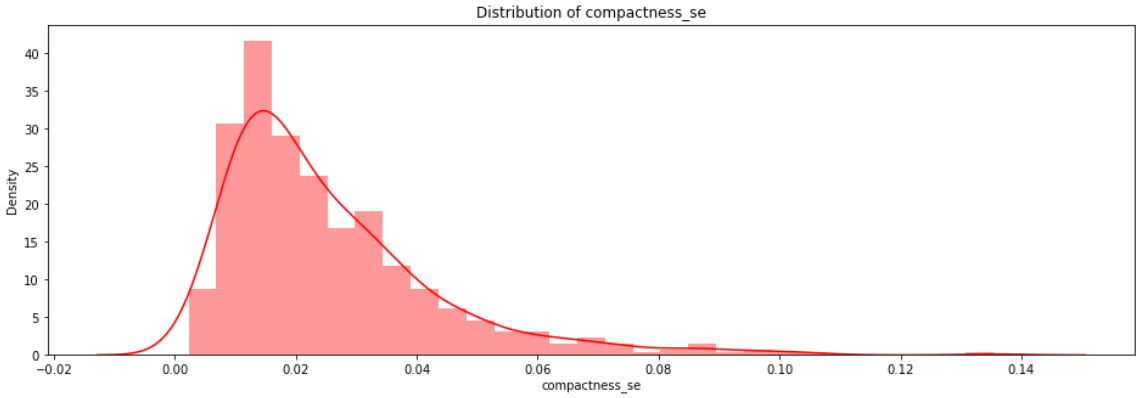
Distribution of compactness_se

In [40]:

```
f16, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["compactness_se"], color="red",ax = axes)
plt.title("Distribution of compactness_se")
```

Out[40]:

```
Text(0.5, 1.0, 'Distribution of compactness_se')
```



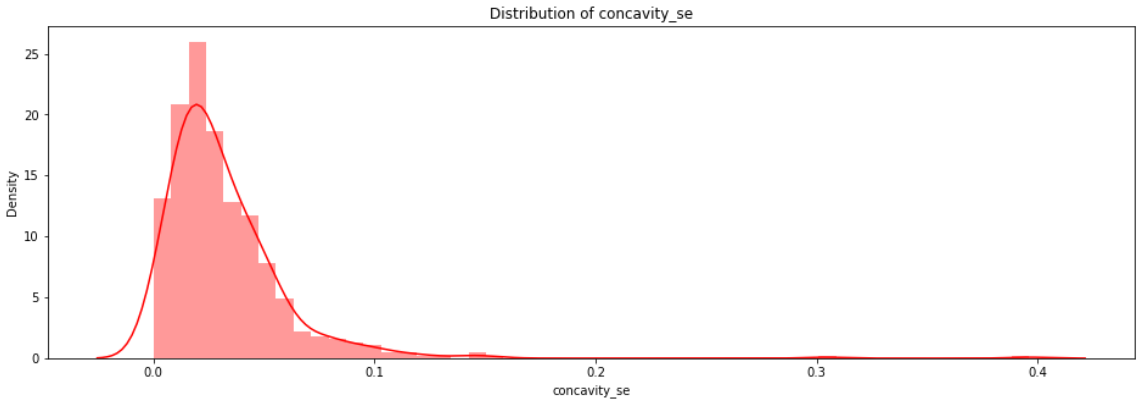
Distribution of concavity_se

In [41]:

```
f17, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["concavity_se"], color="red",ax = axes)
plt.title("Distribution of concavity_se")
```

Out[41]:

```
Text(0.5, 1.0, 'Distribution of concavity_se')
```



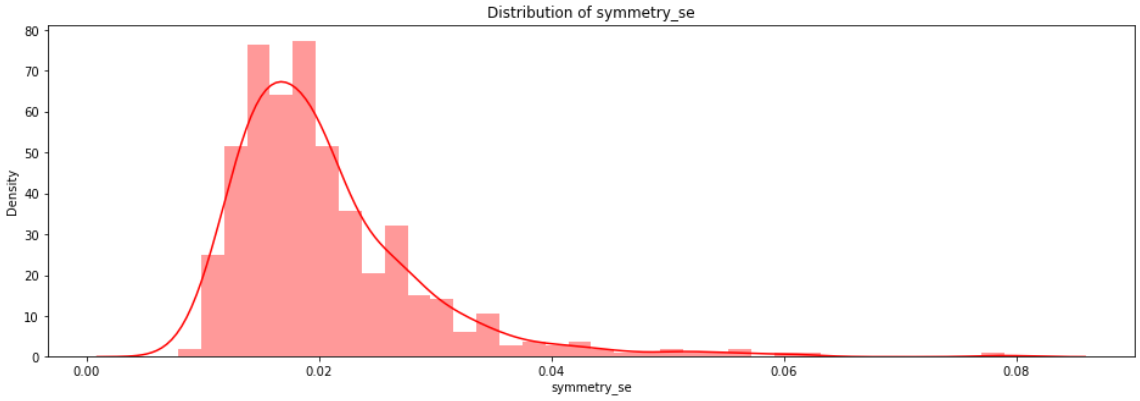
Distribution of symmetry_se

In [42]:

```
f18, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["symmetry_se"], color="red",ax = axes)
plt.title("Distribution of symmetry_se")
```

Out[42]:

```
Text(0.5, 1.0, 'Distribution of symmetry_se')
```



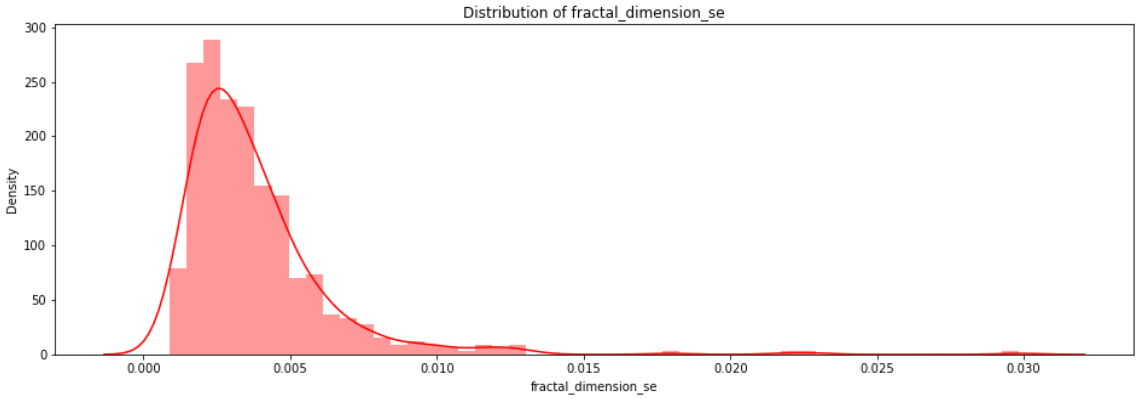
Distribution of fractal_dimension_se

In [43]:

```
f19, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["fractal_dimension_se"], color="red",ax = axes)
plt.title("Distribution of fractal_dimension_se")
```

Out[43]:

```
Text(0.5, 1.0, 'Distribution of fractal_dimension_se')
```



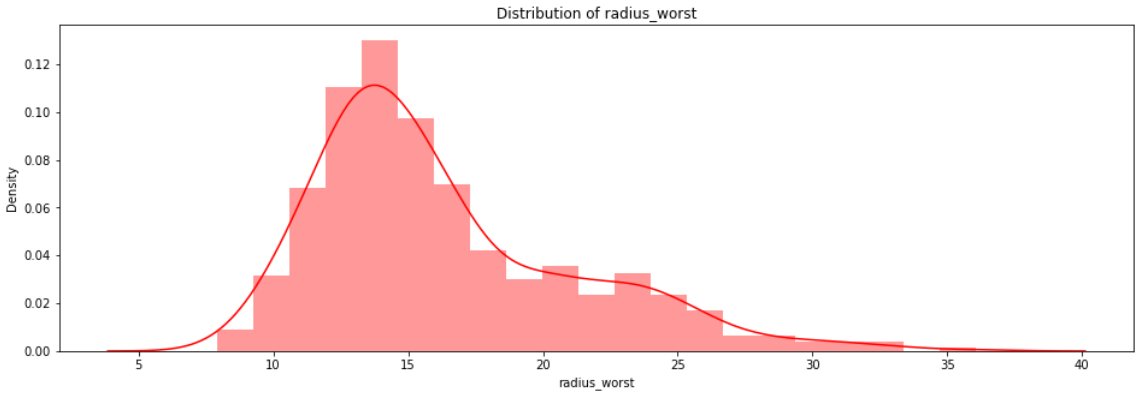
Distribution of radius_worst

In [44]:

```
f20, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["radius_worst"], color="red",ax = axes)
plt.title("Distribution of radius_worst")
```

Out[44]:

```
Text(0.5, 1.0, 'Distribution of radius_worst')
```



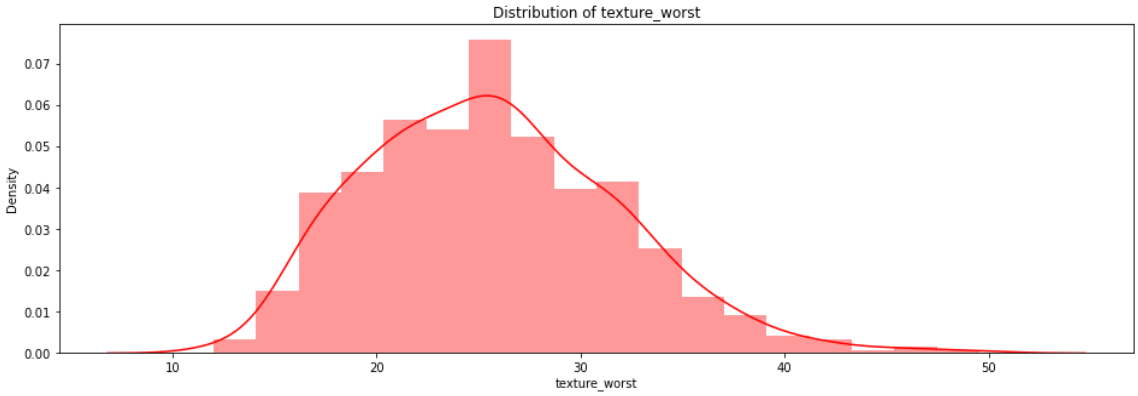
Distribution of texture_worst

In [45]:

```
f21, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["texture_worst"], color="red",ax = axes)
plt.title("Distribution of texture_worst")
```

Out[45]:

```
Text(0.5, 1.0, 'Distribution of texture_worst')
```



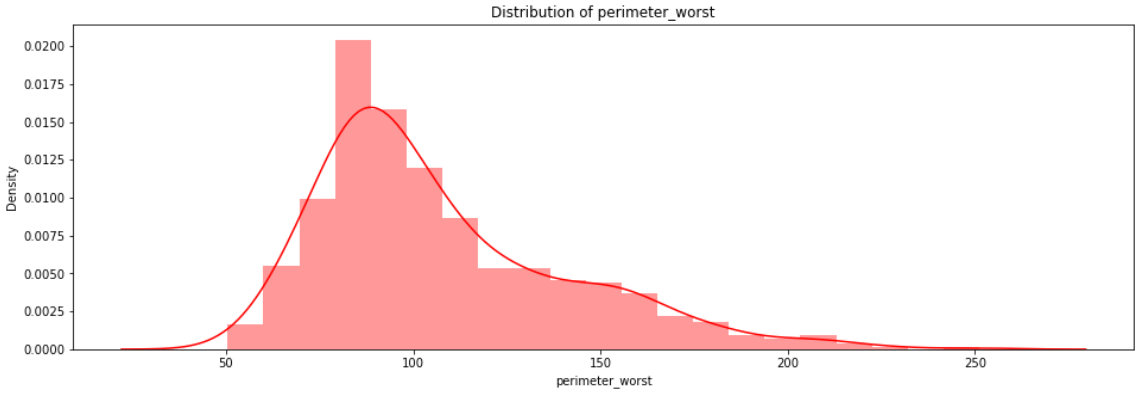
Distribution of perimeter_worst

In [46]:

```
f22, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["perimeter_worst"], color="red",ax = axes)
plt.title("Distribution of perimeter_worst")
```

Out[46]:

```
Text(0.5, 1.0, 'Distribution of perimeter_worst')
```



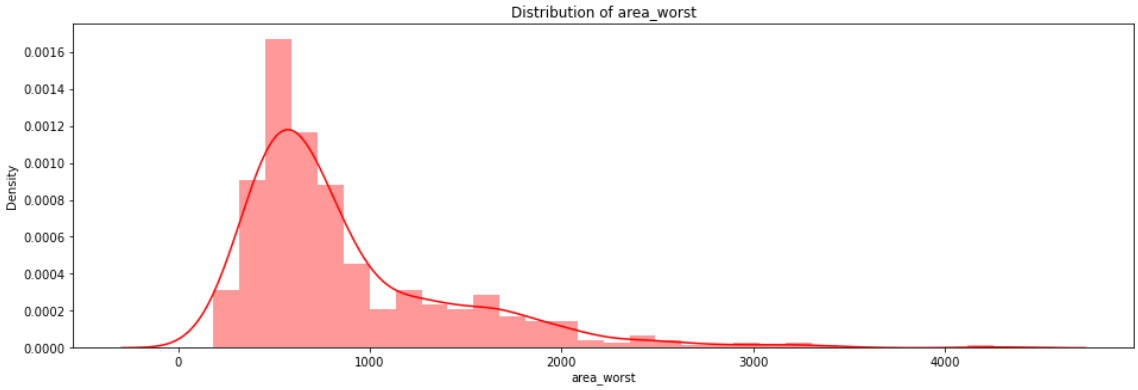
Distribution of area_worst

In [47]:

```
f23, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["area_worst"], color="red",ax = axes)
plt.title("Distribution of area_worst")
```

Out[47]:

```
Text(0.5, 1.0, 'Distribution of area_worst')
```



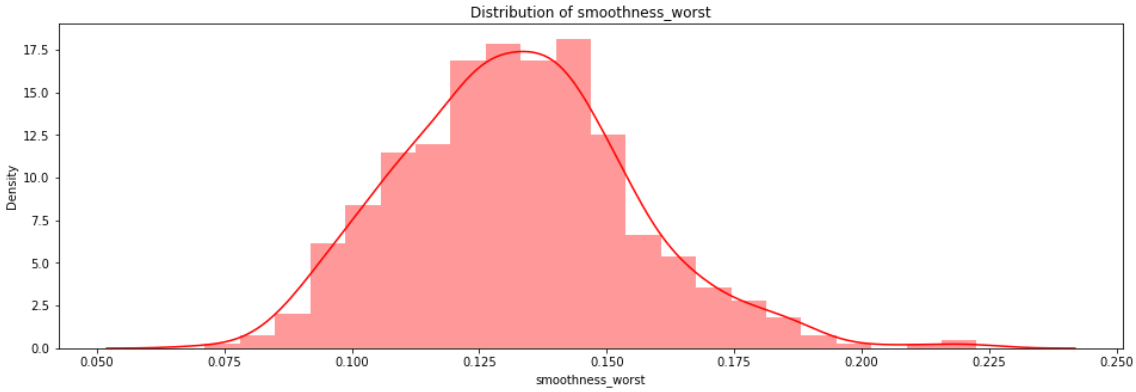
Distribution of smoothness_worst

In [48]:

```
f24, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["smoothness_worst"], color="red",ax = axes)
plt.title("Distribution of smoothness_worst")
```

Out[48]:

```
Text(0.5, 1.0, 'Distribution of smoothness_worst')
```



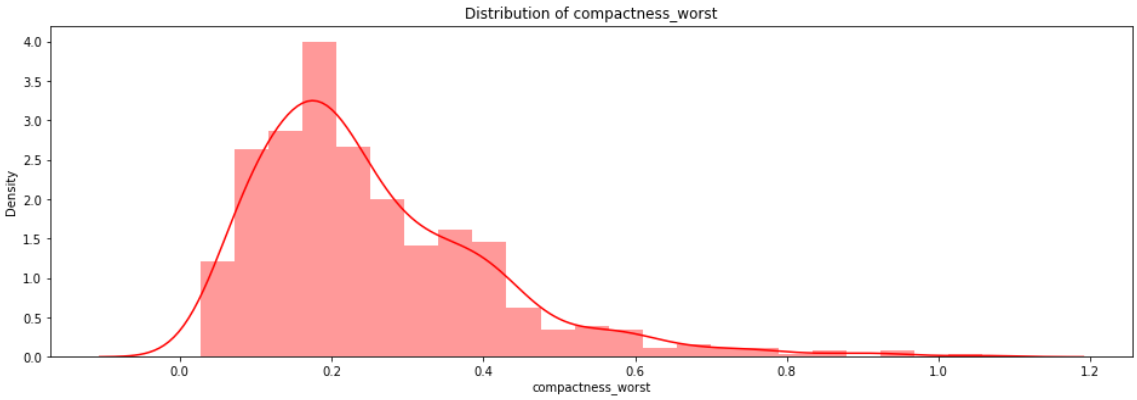
Distribution of compactness_worst

In [49]:

```
f25, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["compactness_worst"], color="red",ax = axes)
plt.title("Distribution of compactness_worst")
```

Out[49]:

```
Text(0.5, 1.0, 'Distribution of compactness_worst')
```



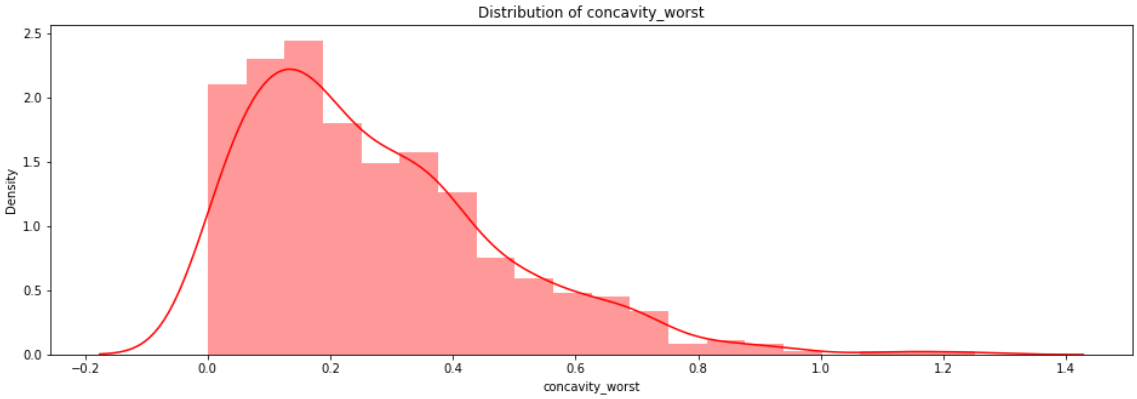
Distribution of concavity_worst

In [50]:

```
f26, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["concavity_worst"], color="red",ax = axes)
plt.title("Distribution of concavity_worst")
```

Out[50]:

```
Text(0.5, 1.0, 'Distribution of concavity_worst')
```



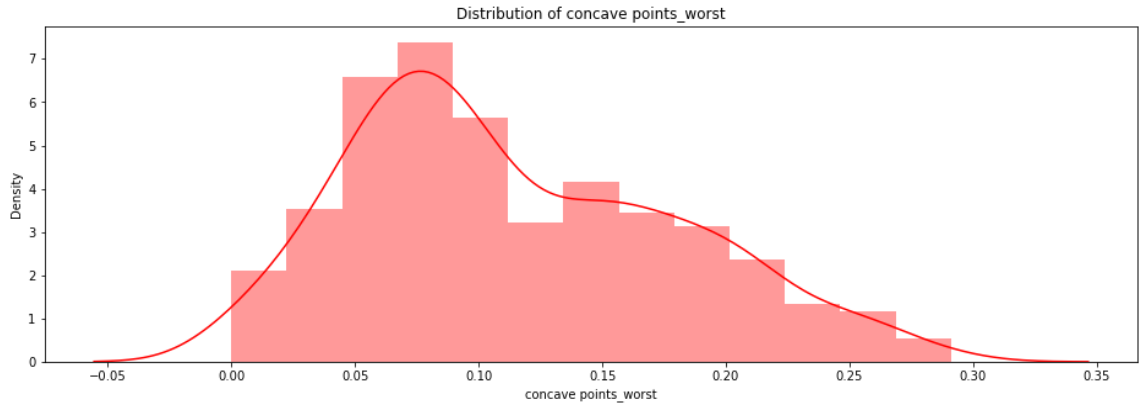
Distribution of concave points_worst

In [51]:

```
f27, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["concave points_worst"], color="red",ax = axes)
plt.title("Distribution of concave points_worst")
```

Out[51]:

```
Text(0.5, 1.0, 'Distribution of concave points_worst')
```



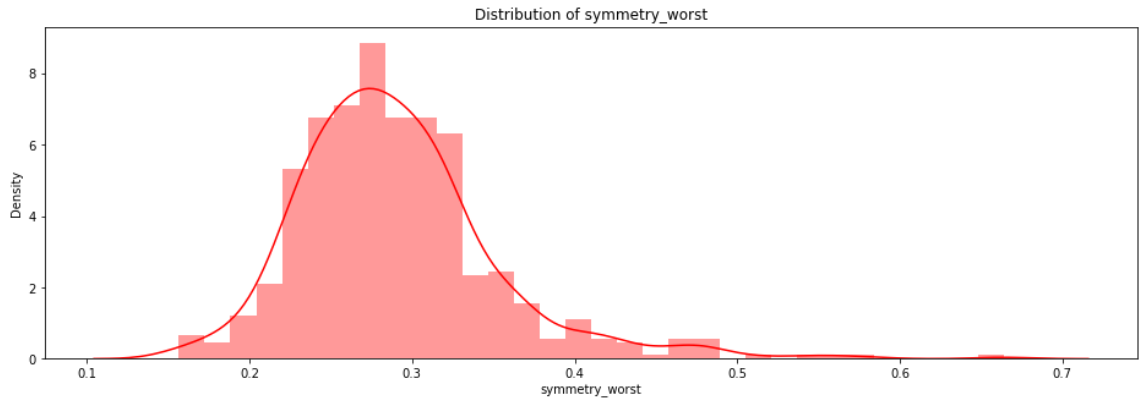
Distribution of symmetry_worst

In [52]:

```
f28, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["symmetry_worst"], color="red",ax = axes)
plt.title("Distribution of symmetry_worst")
```

Out[52]:

```
Text(0.5, 1.0, 'Distribution of symmetry_worst')
```



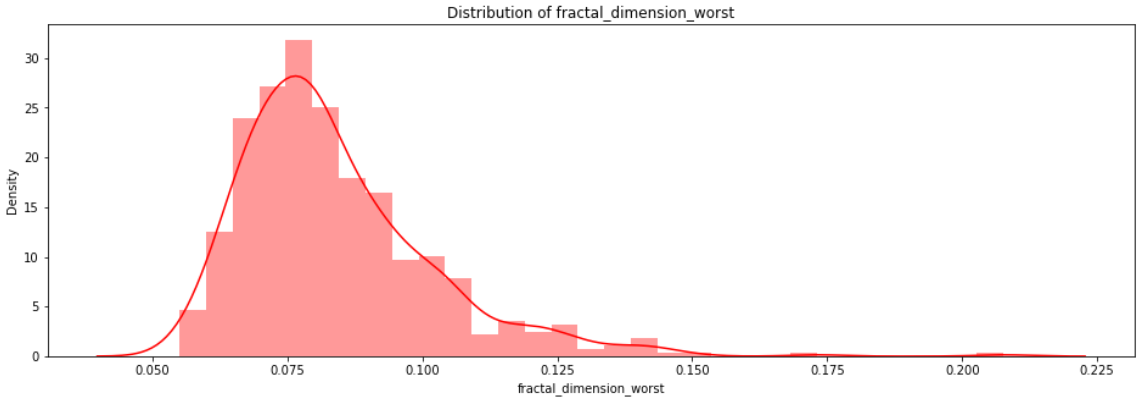
Distribution of fractal_dimension_worst

In [53]:

```
f29, axes = plt.subplots(1,1, figsize = (16, 5))
g1 = sns.distplot(df["fractal_dimension_worst"], color="red",ax = axes)
plt.title("Distribution of fractal_dimension_worst")
```

Out[53]:

```
Text(0.5, 1.0, 'Distribution of fractal_dimension_worst')
```



Explore the data

Now lets Try to find a correlation between columns

In [54]:

```
# Summary of all the numeric columns
df.describe()
```

Out[54]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compac
count	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	14.127292	19.289649	91.969033	654.889104	0.096360	
std	3.524049	4.301036	24.298981	351.914129	0.014064	
min	6.981000	9.710000	43.790000	143.500000	0.052630	
25%	11.700000	16.170000	75.170000	420.300000	0.086370	
50%	13.370000	18.840000	86.240000	551.100000	0.095870	
75%	15.780000	21.800000	104.100000	782.700000	0.105300	
max	28.110000	39.280000	188.500000	2501.000000	0.163400	

8 rows × 30 columns

In [55]:

```
len(df.columns)
```

Out[55]:

```
31
```

In [56]:

```
print("Any missing sample in data set:",df.isnull().values.any(), "\n")
```

```
Any missing sample in data set: False
```

In [57]:

```
#replace missing value with mean if exists
df = df.replace([np.inf, -np.inf], np.nan)
df= df.fillna(df.mean())
df
```

Out[57]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	M	17.99	10.38	122.80	1001.0	0.11840
1	M	20.57	17.77	132.90	1326.0	0.08474
2	M	19.69	21.25	130.00	1203.0	0.10960
3	M	11.42	20.38	77.58	386.1	0.14250
4	M	20.29	14.34	135.10	1297.0	0.10030
...	...	...	...	...	...	...
564	M	21.56	22.39	142.00	1479.0	0.11100
565	M	20.13	28.25	131.20	1261.0	0.09780
566	M	16.60	28.08	108.30	858.1	0.08455
567	M	20.60	29.33	140.10	1265.0	0.11780
568	B	7.76	24.54	47.92	181.0	0.05263

569 rows × 31 columns

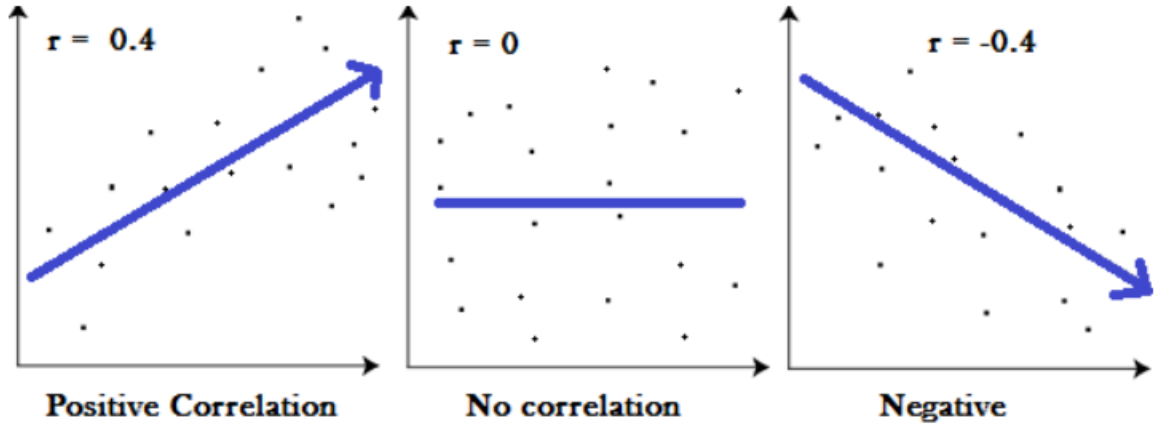
Correlation in Data

In this work, I try to measure correlation in data using Correlation coefficients.

Correlation coefficientsare used to measure how strong a relationship is between two variables.Correlation coefficient formulas are used to find how strong a relationship is between data. The formulas return a value between -1 and 1, where:

1 indicates a strong positive relationship. -1 indicates a strong negative relationship.

A result of zero indicates no relationship at all.



In [58]:

```
# Correlation Plot
corr = df.corr()
corr
```

Out[58]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
radius_mean	1.000000	0.323782	0.997855	0.987357	(
texture_mean	0.323782	1.000000	0.329533	0.321086	-(
perimeter_mean	0.997855	0.329533	1.000000	0.986507	(
area_mean	0.987357	0.321086	0.986507	1.000000	(
smoothness_mean	0.170581	-0.023389	0.207278	0.177028	1
compactness_mean	0.506124	0.236702	0.556936	0.498502	(
concavity_mean	0.676764	0.302418	0.716136	0.685983	(
concave points_mean	0.822529	0.293464	0.850977	0.823269	(
symmetry_mean	0.147741	0.071401	0.183027	0.151293	(
fractal_dimension_mean	-0.311631	-0.076437	-0.261477	-0.283110	(
radius_se	0.679090	0.275869	0.691765	0.732562	(
texture_se	-0.097317	0.386358	-0.086761	-0.066280	(
perimeter_se	0.674172	0.281673	0.693135	0.726628	(
area_se	0.735864	0.259845	0.744983	0.800086	(
smoothness_se	-0.222600	0.006614	-0.202694	-0.166777	(
compactness_se	0.206000	0.191975	0.250744	0.212583	(
concavity_se	0.194204	0.143293	0.228082	0.207660	(
concave points_se	0.376169	0.163851	0.407217	0.372320	(
symmetry_se	-0.104321	0.009127	-0.081629	-0.072497	(
fractal_dimension_se	-0.042641	0.054458	-0.005523	-0.019887	(
radius_worst	0.969539	0.352573	0.969476	0.962746	(
texture_worst	0.297008	0.912045	0.303038	0.287489	(
perimeter_worst	0.965137	0.358040	0.970387	0.959120	(
area_worst	0.941082	0.343546	0.941550	0.959213	(
smoothness_worst	0.119616	0.077503	0.150549	0.123523	(
compactness_worst	0.413463	0.277830	0.455774	0.390410	(
concavity_worst	0.526911	0.301025	0.563879	0.512606	(
concave points_worst	0.744214	0.295316	0.771241	0.722017	(
symmetry_worst	0.163953	0.105008	0.189115	0.143570	(
fractal_dimension_worst	0.007066	0.119205	0.051019	0.003738	(

30 rows × 30 columns

In [59]:

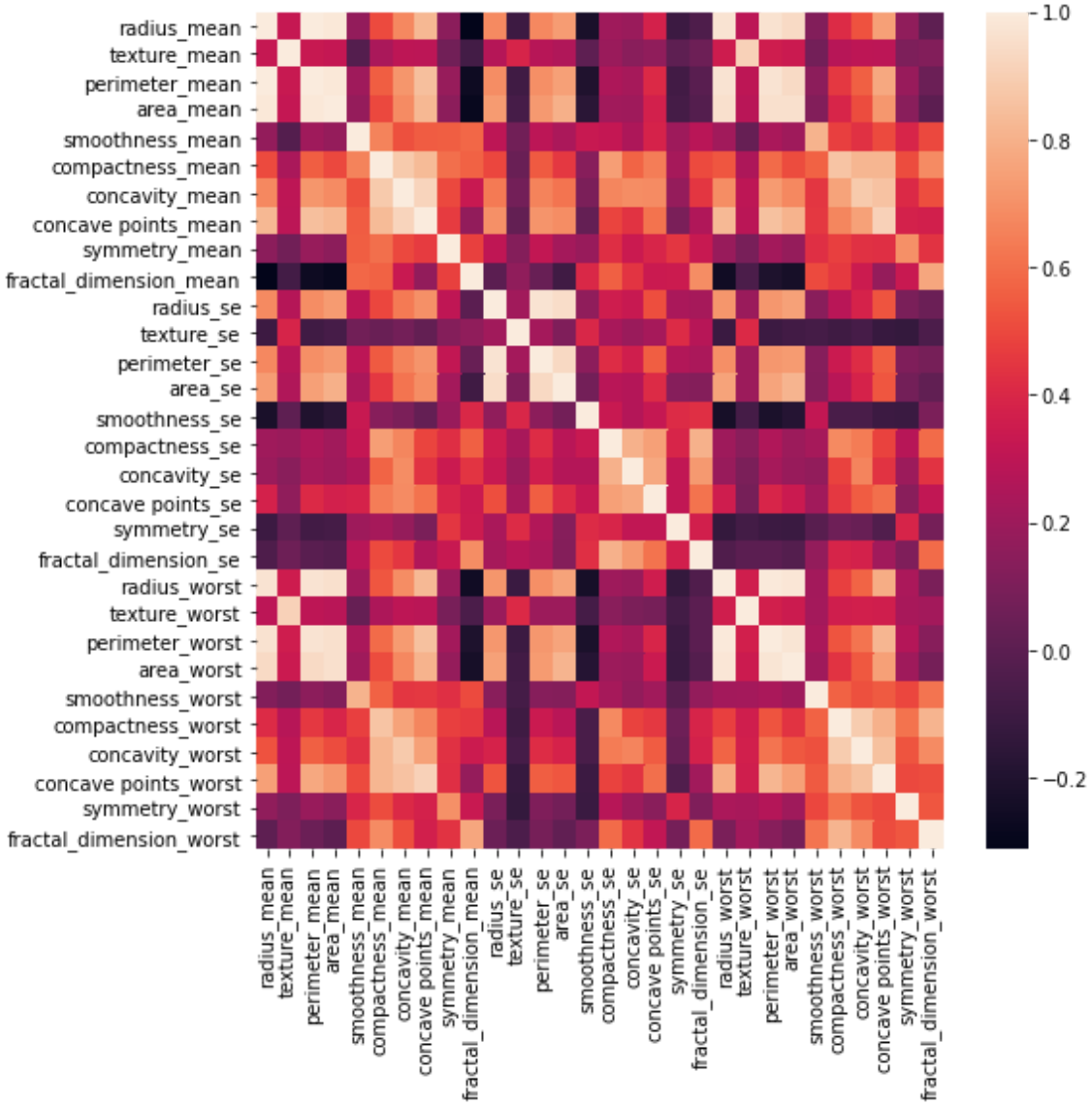
```
corr.shape
```

Out[59]:

```
(30, 30)
```

In [70]:

```
# HeatMap
plt.figure(figsize=(8,8))
sns.heatmap(corr);
```



In [71]:

```
df.head()
```

Out[71]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
0	M	17.99	10.38	122.80	1001.0	0.11840	
1	M	20.57	17.77	132.90	1326.0	0.08474	
2	M	19.69	21.25	130.00	1203.0	0.10960	
3	M	11.42	20.38	77.58	386.1	0.14250	
4	M	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 31 columns

In [72]:

```
df['diagnosis'] = df['diagnosis'].map({'M':1, 'B':0})
```

In [73]:

```
df.head()
```

Out[73]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
0	1	17.99	10.38	122.80	1001.0	0.11840	
1	1	20.57	17.77	132.90	1326.0	0.08474	
2	1	19.69	21.25	130.00	1203.0	0.10960	
3	1	11.42	20.38	77.58	386.1	0.14250	
4	1	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 31 columns

In [74]:

```
df['diagnosis'].unique()
```

Out[74]:

```
array([1, 0])
```

In [75]:

```
X = df.drop('diagnosis', axis=1)
X.head()
```

Out[75]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

5 rows × 30 columns

In [76]:

```
y = df['diagnosis']
y.head()
```

Out[76]:

```
0    1
1    1
2    1
3    1
4    1
Name: diagnosis, dtype: int64
```

In [77]:

```
# Spilting dataset into train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [78]:

```
df.shape
```

Out[78]:

```
(569, 31)
```

In [79]:

```
X_train.shape
```

Out[79]:

```
(398, 30)
```

In [80]:

```
X_test.shape
```

Out[80]:

```
(171, 30)
```

In [81]:

```
y_train.shape
```

Out[81]:

```
(398,)
```

In [82]:

```
y_test.shape
```

Out[82]:

```
(171,)
```

In [83]:

```
X_train.head(1)
```

Out[83]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactne
484	15.73	11.28	102.8	747.2	0.1043	

1 rows × 30 columns

In [84]:

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

In [85]:

```
X_train
```

Out[85]:

```
array([[ 0.41762049, -1.8674916 ,  0.4081951 , ...,  0.41198288,
        -0.56093931, -0.17099845],
       [ 0.11133713, -0.9750259 ,  0.12162204, ...,  0.32894168,
         0.37152857, -0.0176723 ],
       [ 0.43167019,  0.75356295,  0.52641158, ...,  1.29070969,
         0.56461535,  2.10736518],
       ...,
       [-0.12188782,  0.94287385, -0.04999568, ...,  1.25900306,
         2.38245678,  1.2357165 ],
       [ 0.30803287, -1.19588863,  0.39596581, ...,  1.83274222,
         2.75136242,  0.75578463],
       [-0.76536387, -1.04489064, -0.77111621, ..., -0.32965048,
        -0.39296951, -0.36633177]])
```

Machine Learning Models

1) Logistic Regression

In [86]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

Out[86]:

```
LogisticRegression()
```

In [87]:

```
y_pred = lr.predict(X_test)
```

In [88]:

```
y_pred
```

Out[88]:

```
array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0])
```

In [89]:

```
y_test
```

Out[89]:

```
424    0
143    0
202    1
146    1
561    0
      ..
535    1
513    0
509    1
533    1
454    0
Name: diagnosis, Length: 171, dtype: int64
```

In [90]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9824561403508771
```

In [91]:

```
lr_acc = accuracy_score(y_test, y_pred)
print(lr_acc)
```

```
0.9824561403508771
```

In [92]:

```
results = pd.DataFrame()
results
```

Out[92]:

—

In [93]:

```
tempResults = pd.DataFrame({'Algorithm':['Logistic Regression Method'], 'Accuracy':[lr_a
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[93]:

	Algorithm	Accuracy
0	Logistic Regression Method	0.982456

2) Decision Tree Classifier

In [94]:

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
```

Out[94]:

```
DecisionTreeClassifier()
```

In [95]:

```
y_pred = dtc.predict(X_test)
y_pred
```

Out[95]:

```
array([0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0])
```

In [96]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9239766081871345
```

In [97]:

```
dtc_acc = accuracy_score(y_test, y_pred)
print(dtc_acc)
```

```
0.9239766081871345
```

In [98]:

```
tempResults = pd.DataFrame({'Algorithm':['Decision tree Classifier Method'], 'Accuracy':
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[98]:

	Algorithm	Accuracy
0	Logistic Regression Method	0.982456
0	Decision tree Classifier Method	0.923977

3) Random Forest Classifier

In [99]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
```

Out[99]:

```
RandomForestClassifier()
```

In [100]:

```
y_pred = rfc.predict(X_test)
y_pred
```

Out[100]:

```
array([0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0])
```

In [101]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9649122807017544
```

In [102]:

```
rfc_acc = accuracy_score(y_test, y_pred)
print(rfc_acc)
```

```
0.9649122807017544
```

In [103]:

```
tempResults = pd.DataFrame({'Algorithm':['Random Forest Classifier Method'], 'Accuracy':
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[103]:

	Algorithm	Accuracy
0	Logistic Regression Method	0.982456
0	Decision tree Classifier Method	0.923977
0	Random Forest Classifier Method	0.964912

4) Support Vector Classifier

In [104]:

```
from sklearn import svm
svc = svm.SVC()
svc.fit(X_train,y_train)
```

Out[104]:

```
SVC()
```

In [105]:

```
y_pred = svc.predict(X_test)
y_pred
```

Out[105]:

```
array([0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0])
```

In [106]:

```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))
```

```
0.9824561403508771
```

In [107]:

```
svc_acc = accuracy_score(y_test, y_pred)
print(svc_acc)
```

```
0.9824561403508771
```

In [108]:

```
tempResults = pd.DataFrame({'Algorithm':['Support Vector Classifier Method'], 'Accuracy'
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

Out[108]:

	Algorithm	Accuracy
0	Logistic Regression Method	0.982456
0	Decision tree Classifier Method	0.923977
0	Random Forest Classifier Method	0.964912
0	Support Vector Classifier Method	0.982456

We are getting maximum accuracy in Logistic Regression Model