ASSIGNMENT 2

- **Name : Achal Rajesh Mate**
- **Roll No : 2203541**
- **Enroll No : MITU20BTCSD001**
- **Branch : CSE**
- **Class : TY CSE Is - 3**
- **Guided By : Prof Nagesh Jadhav Sir**

Title:- Download any dataset from UCI or Data.org or from any data repositories and perform the basic data pre-processing steps using Python/R

Objectives:-
1. Learn to pre-process dataset
2. Learn to use pandas and sklearn

Theory:

1. Description of basic data preprocessing steps.(Data cleaning, Standardization,Normalization, data splitting)

Basic steps

- **Step 1 :** Import the libraries
- **Step 2 :** Import the data-set
- **Step 3 :** Check out the missing values
- **Step 4 :** See the Categorical Values
- **Step 5 :** Splitting the data-set into Training and Test Set

Data cleaning:

- The main aim of Data Cleaning is to identify and remove errors & duplicate data,in order to create a reliable dataset.
- This improves the quality of the training data for analytics and enables accurate decision-making.
- Needless to say, data cleansing is a time-consuming process and most data scientists spend an enormous amount of time in enhancing the quality of the data. However, there are various methods to identify and classify data for data cleansing.
- There are mainly two distinct techniques, namely Qualitative and Quantitative techniques to classify data errors.
- Qualitative techniques involve rules, constraints, and patterns to identify errors.
- On the other hand, Quantitative techniques employ statistical techniques to identify errors in the trained data.

Normalisation:

- Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.
- Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.
  - When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
  - On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
  - If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

Standardisation:

- Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation.
- This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.
- Here's the formula for standardization:

$$X' = \frac{X - \mu}{\sigma}$$

μ is the mean of the feature values and σ is the standard deviation of the feature values. Note that in this case, the values are not restricted to a particular range.

Splitting the Dataset

- X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.2,random_state=0)
- Split arrays or matrices into random train and test subsets
- Quick utility that wraps input validation and next(ShuffleSplit().split(X, y)) and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.
- Imputer(missing_values='NaN', strategy='mean', axis=0)
- Imputation transformer for completing missing values.

pandas.read_csv()

- Read a comma-separated values (csv) file into DataFrame.
- Also supports optionally iterating or breaking of the file into chunks.

Dataset used and its attributes

DataSet Link : https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic) (https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic))
Dataset Information : Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Features Explanations:

- Number of Instances: 569
- Number of Attributes: 30 numeric, predictive attributes and the class

Attribute Information:

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)
- The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each data, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.
- Missing Values: 569
- Class Distribution:
  - 212 - Malignant,
  - 357 - Benign Depending on the types of cells in a tumor, it can be:
- Benign - The tumor doesn’t contain cancerous cells.
- Malignant - The tumor contains cancerous cells.

Import All Necessary Library

In [1]:
```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
from sklearn import preprocessing
from scipy.stats import norm
from sklearn.model_selection import train_test_split
```

Load the DataSet

In [5]:
```
df = pd.read_csv("Breast Cancer Data.csv")
```

In [6]:
```
df.head()
```

Out[6]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conc points_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10

5 rows × 33 columns

View First 5 Rows

In [7]:
```
df.head()
```

Out[7]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conc points_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10

5 rows × 33 columns

Set Option to View all Rows and Columns

In [8]:
```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

Dimensions of the Dataset

In [9]:
```
df.shape
```

Out[9]:
```
(569, 33)
```

Dataset contains 569 instances with 33 rows

Concise Summary

In [10]:
```
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
```

From above result diagnosis is the feature in object datatype and Unnamed: 32 feature contains all now value lets check by using pandas function

Check the Missing Data

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave points_mean         0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
fractal_dimension_worst     0
Unnamed: 32               569
dtype: int64
```

From the above observation Unnamed: 32 column contains all the Null values
So it would be better to drop the column

Drop the Column Containing Missing Value

In [12]:
```
df.drop('Unnamed: 32',axis = 1,inplace = True)
```

Recheck the Missing Value is present or not

In [13]:
```
df.isnull().sum().sum()
```

Out[13]:
```
0
```

Statistical Summary of data

In [14]:
```
df.describe()
```

Out[14]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concav points_mea
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.0000(
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.0489
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.0388(
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.0000(
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.0203
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.0335(
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.0740(
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.2012(

Columns of the dataset

In [15]:
```
df.columns
```

Out[15]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       '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')
```

Check The types of values of Diagnosis Present In dataset

In [16]:
```
df.diagnosis.unique()
```

Out[16]:
```
array(['M', 'B'], dtype=object)
```

Count number of Malignant (M) or Benign (B) Cells

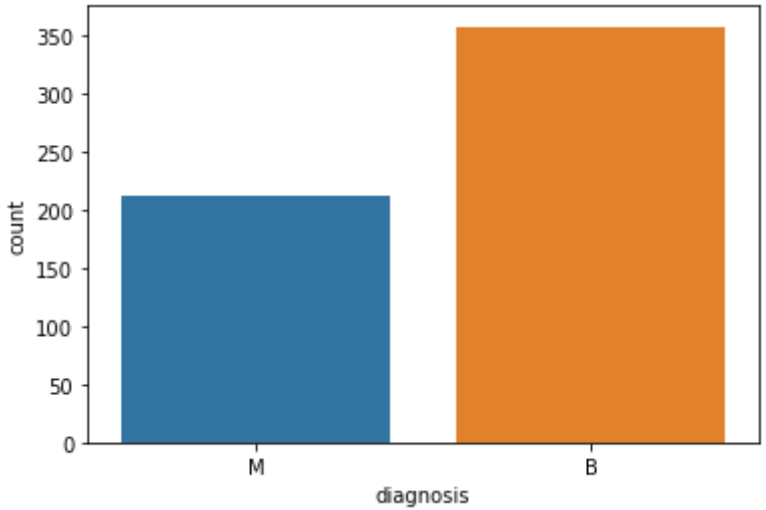
In [17]:
```
df['diagnosis'].value_counts()
```

Out[17]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

From above result we can say that M=Maligant is of 212 instance which is postive prediction
B= Benign is of 357 instance which is negative prediction

In [18]:
```
sns.countplot('diagnosis',data=df,label ="Diagnosis")
```

Out[18]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='count'>
```



Dataset Contain Maximum Number of Negative Predictions

Univatiate Analysis

Check the outlier in the dataset

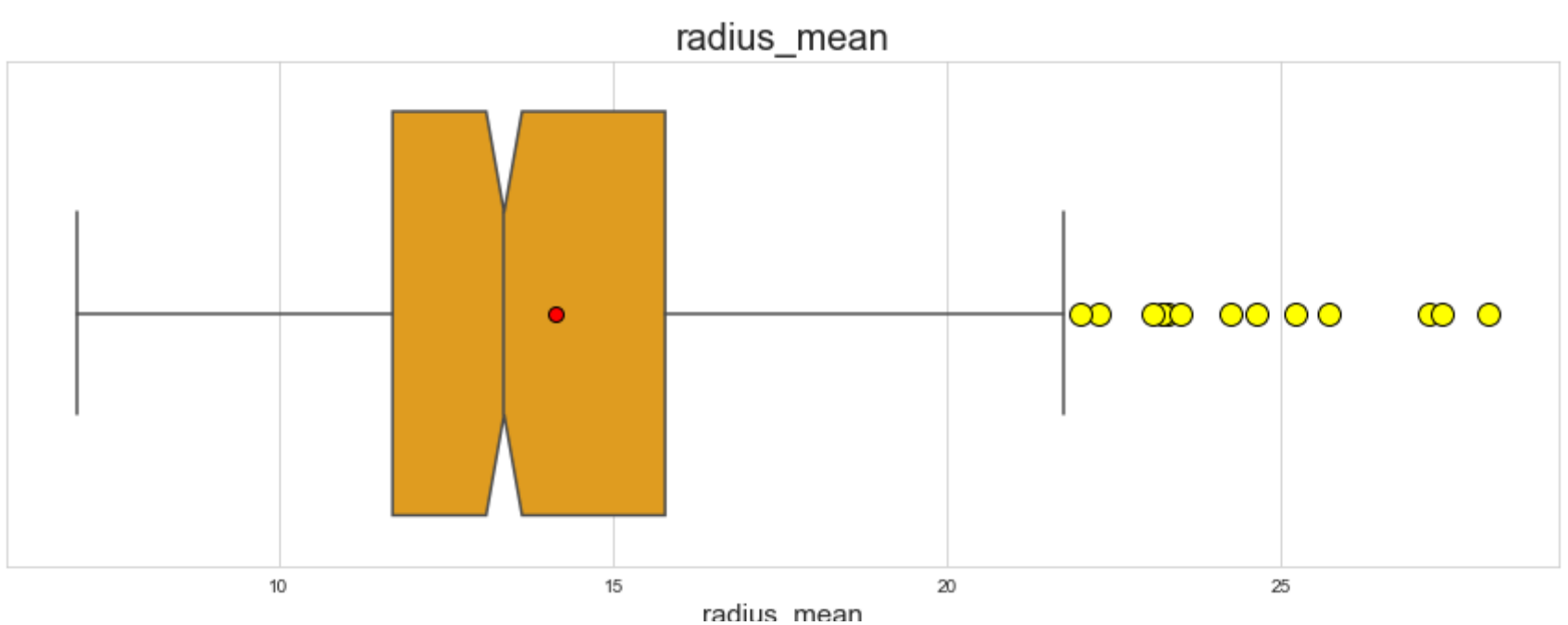
In [19]:
```
data = ['radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       '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']
```

In [20]:
```
def box_plot_wiskers(feature):
    sns.set_style('whitegrid')
    out_marker = dict(markerfacecolor='yellow',markeredgecolor = 'black', marker='o', markersize = 12)
    plt.figure(figsize=(15,5))
    plt.title(feature ,fontsize = 20)
    plt.xlabel(feature , fontsize = 15)



    boxplot = sns.boxplot(df[feature],color='orange',
                          showmeans=True,
                          notch = True,
                          flierprops=out_marker,
                          meanprops={ "marker":"o",
                                     "markerfacecolor":"red",
                                     "markeredgecolor":"black",
                                     "markersize":"8"})
```

In [21]:
```
for clm in data:
    box_plot_wiskers(clm)
```



In [ ]:

In [22]:

```
for i in df.columns[3:]:
    Q1,Q3 = np.quantile(df[i],[0.25,0.75])
    IQR = Q3-Q1
    lower_bound = Q1 - (1.5*IQR)
    if lower_bound < 0:  # as in whole dataset there is no -ve value hence we set lowerbound to 0
        lower_bound=0
    upper_bound = Q3 + (1.5*IQR)
    print(i)
    print('Lower Bound =',np.round(lower_bound,2),' Upper Bound =',np.round(upper_bound,2))
    print('min value =',df[i].min(), ' max value =', df[i].max())

    if df[i].min() < lower_bound:
        print('negative Outliers',len(df[(df[i]<lower_bound)]))

    if df[i].max() > upper_bound:
        print('positive Outliers', len(df[(df[i]>upper_bound)]))




    print('='*50)
```

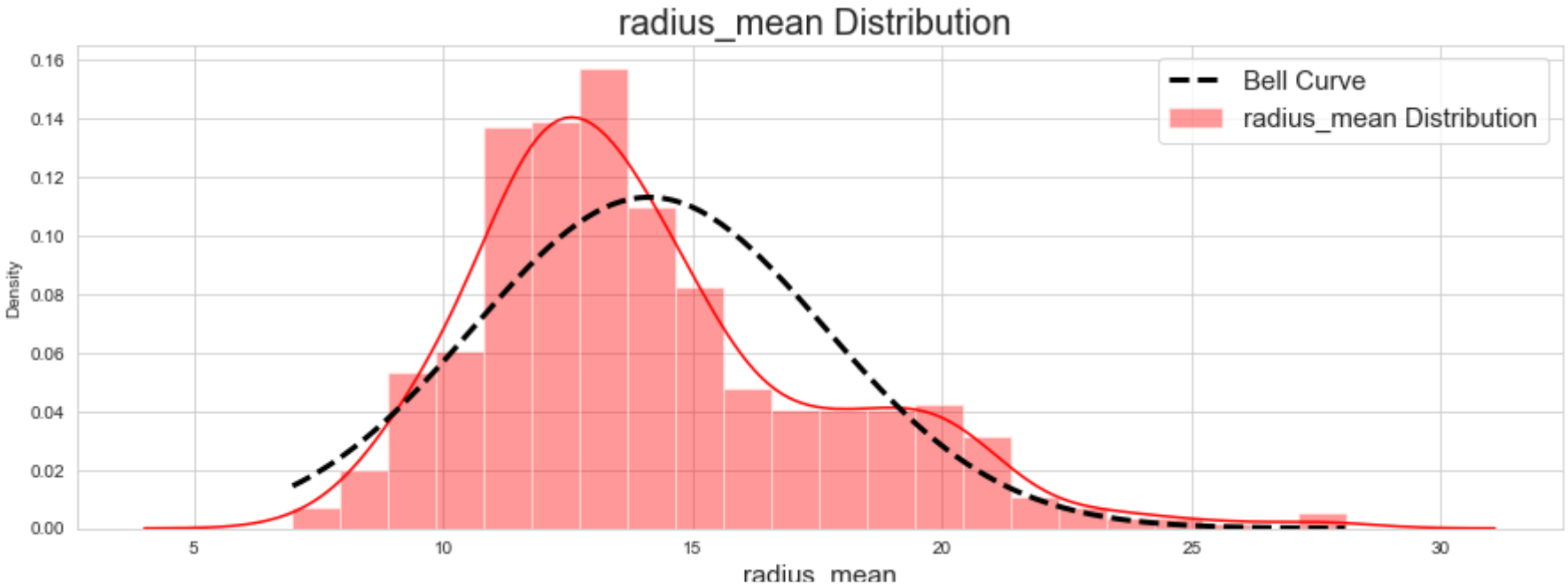
```
texture_mean
Lower Bound = 7.73  Upper Bound = 30.24
min value = 9.71  max value = 39.28
positive Outliers 7
==================================================
perimeter_mean
Lower Bound = 31.78  Upper Bound = 147.49
min value = 43.79  max value = 188.5
positive Outliers 13
==================================================
area_mean
Lower Bound = 0  Upper Bound = 1326.3
min value = 143.5  max value = 2501.0
positive Outliers 25
==================================================
smoothness_mean
Lower Bound = 0.06  Upper Bound = 0.13
```

```
min value = 0.05263  max value = 0.1634
negative Outliers 1
```

Data Distribution of Each feature

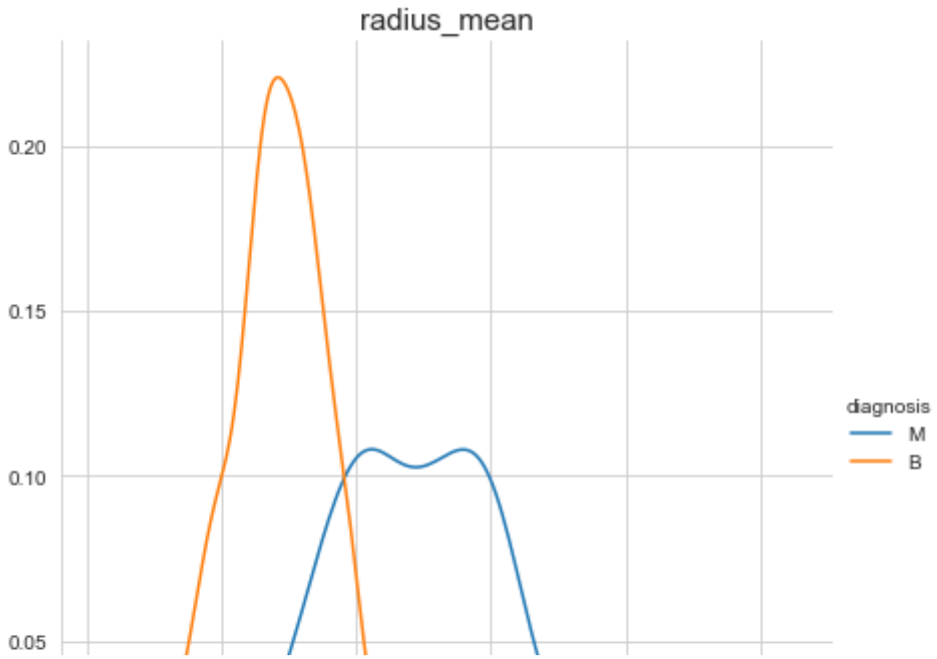
In [23]:
```
for clm in df.columns[2:]:
    sns.set_style('whitegrid')
    plt.figure(figsize=(15,5))
    plt.title(clm+' Distribution ',fontsize = 20)
    plt.xlabel(clm , fontsize = 15)
    sns.distplot(df[clm],color='r' , label=clm+' Distribution')

    rang = np.arange(df[clm].min(),df[clm].max(),0.1)
    plt.plot(rang, norm.pdf(rang,df[clm].mean(),df[clm].std()),color='black', ls='--',lw =3,label = 'Bell Curve')
    plt.legend(fontsize= 15)
```



Distribution density plot KDE (kernel density estimate)

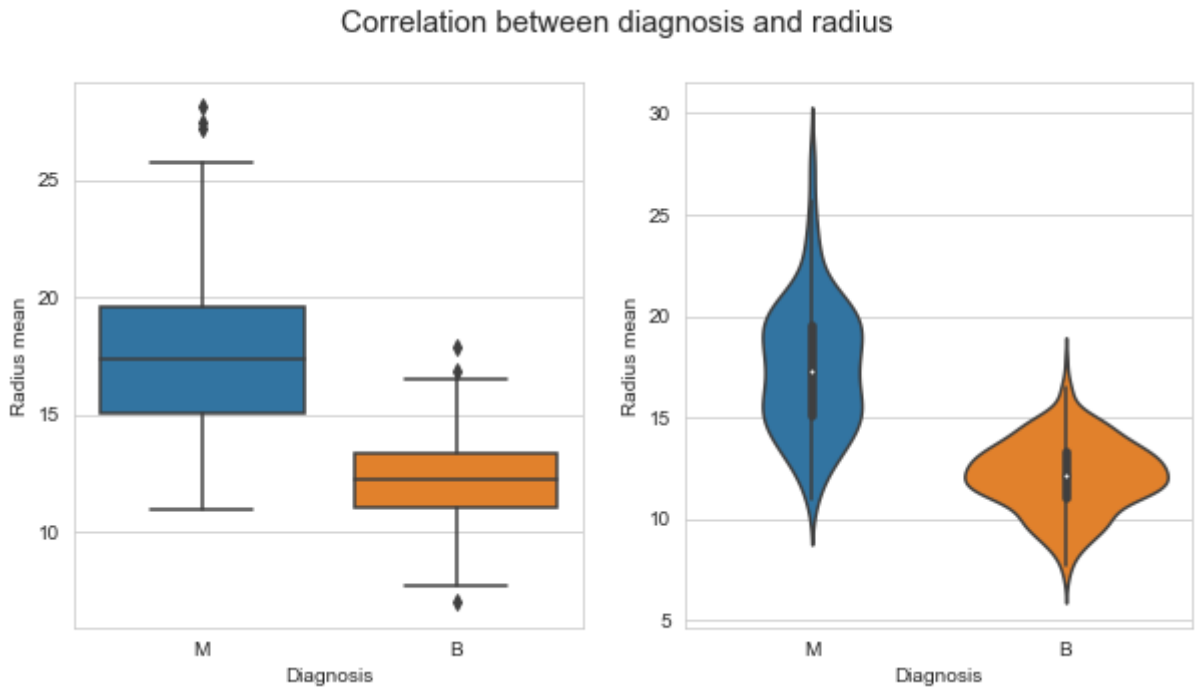
In [24]:
```
for clm in data:
    sns.FacetGrid(df, hue="diagnosis", height=6).map(sns.kdeplot, clm).add_legend()
    plt.title(clm,fontsize=15)
    plt.show()
```



Plotting Bivariate relations between features and diagnosis

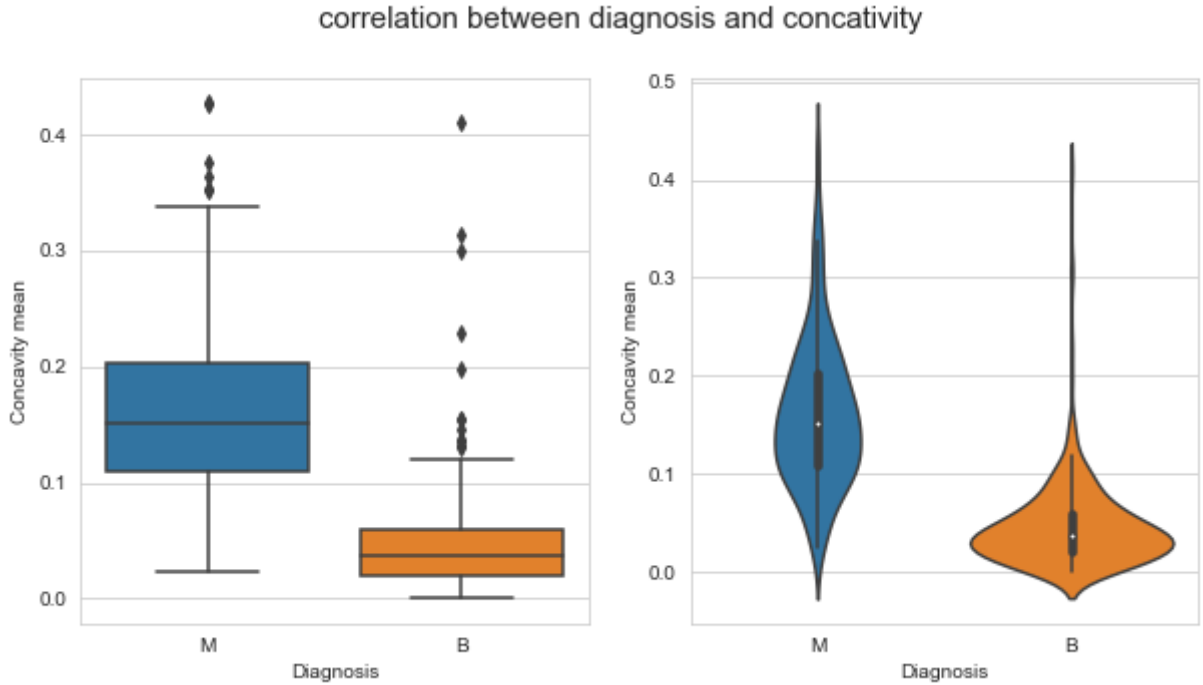
In [25]:

```
plt.figure(figsize=(10,5))
plt.suptitle("Correlation between diagnosis and radius",fontsize=15)
plt.subplot(1,2,1)
sns.boxplot(x="diagnosis", y="radius_mean", data=df)
plt.xlabel('Diagnosis',fontsize=10)
plt.ylabel("Radius mean",fontsize=10)
plt.subplot(1,2,2)
sns.violinplot(x="diagnosis", y="radius_mean", data=df)
plt.xlabel('Diagnosis',fontsize=10)
plt.ylabel("Radius mean",fontsize=10)
plt.show()
```



In [26]:
```
# Plotting

plt.figure(figsize=(10,5))
plt.suptitle("correlation between diagnosis and concativity",fontsize=15)
plt.subplot(1,2,1)
sns.boxplot(x="diagnosis", y="concavity_mean", data=df)
plt.xlabel('Diagnosis',fontsize=10)
plt.ylabel("Concavity mean",fontsize=10)
plt.subplot(1,2,2)
sns.violinplot(x="diagnosis", y="concavity_mean", data=df)
plt.xlabel('Diagnosis',fontsize=10)
plt.ylabel("Concavity mean",fontsize=10)
plt.show()
```



More outliers are present in 'B' in correlation with concativity than 'M' type of diagnosis

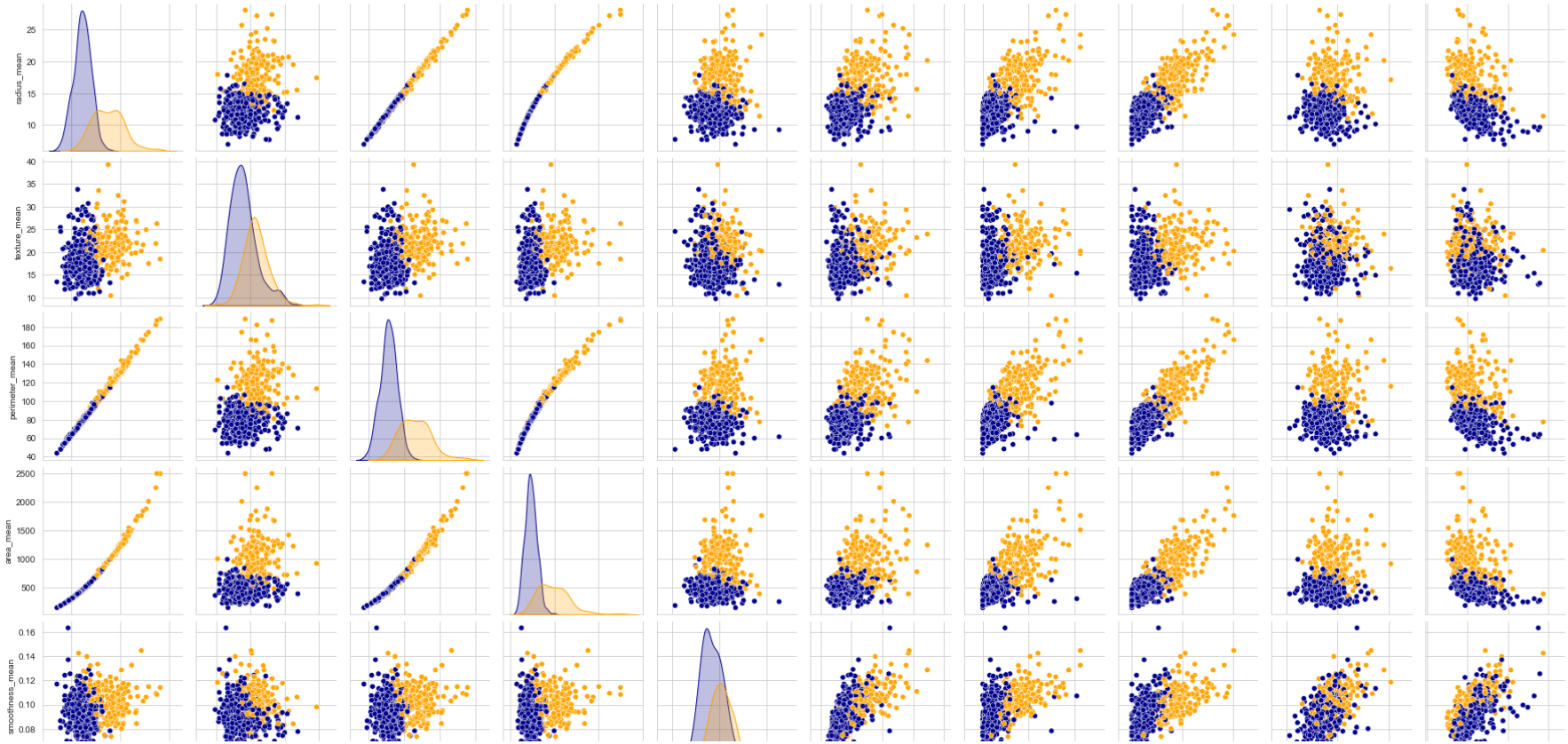
Plotting Multivariate relations between each pair of features hue = "diagnosis"

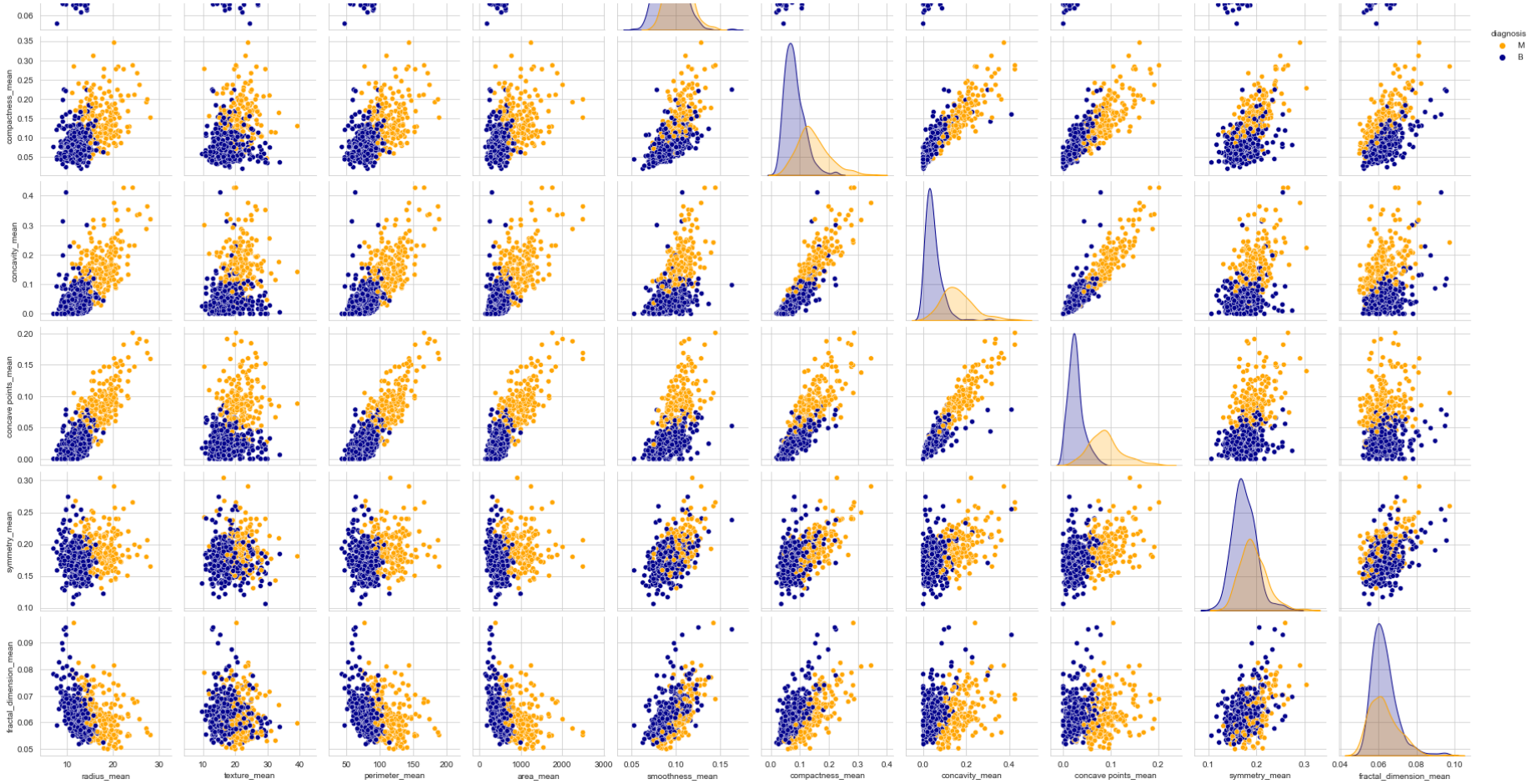
PairPlot between Mean Features with Diagnosis

In [27]:
```
df_mean = df[['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_mean, hue='diagnosis', palette=('Orange','DarkBlue'))
```

Out[27]: <seaborn.axisgrid.PairGrid at 0x1ca11ac7b50>



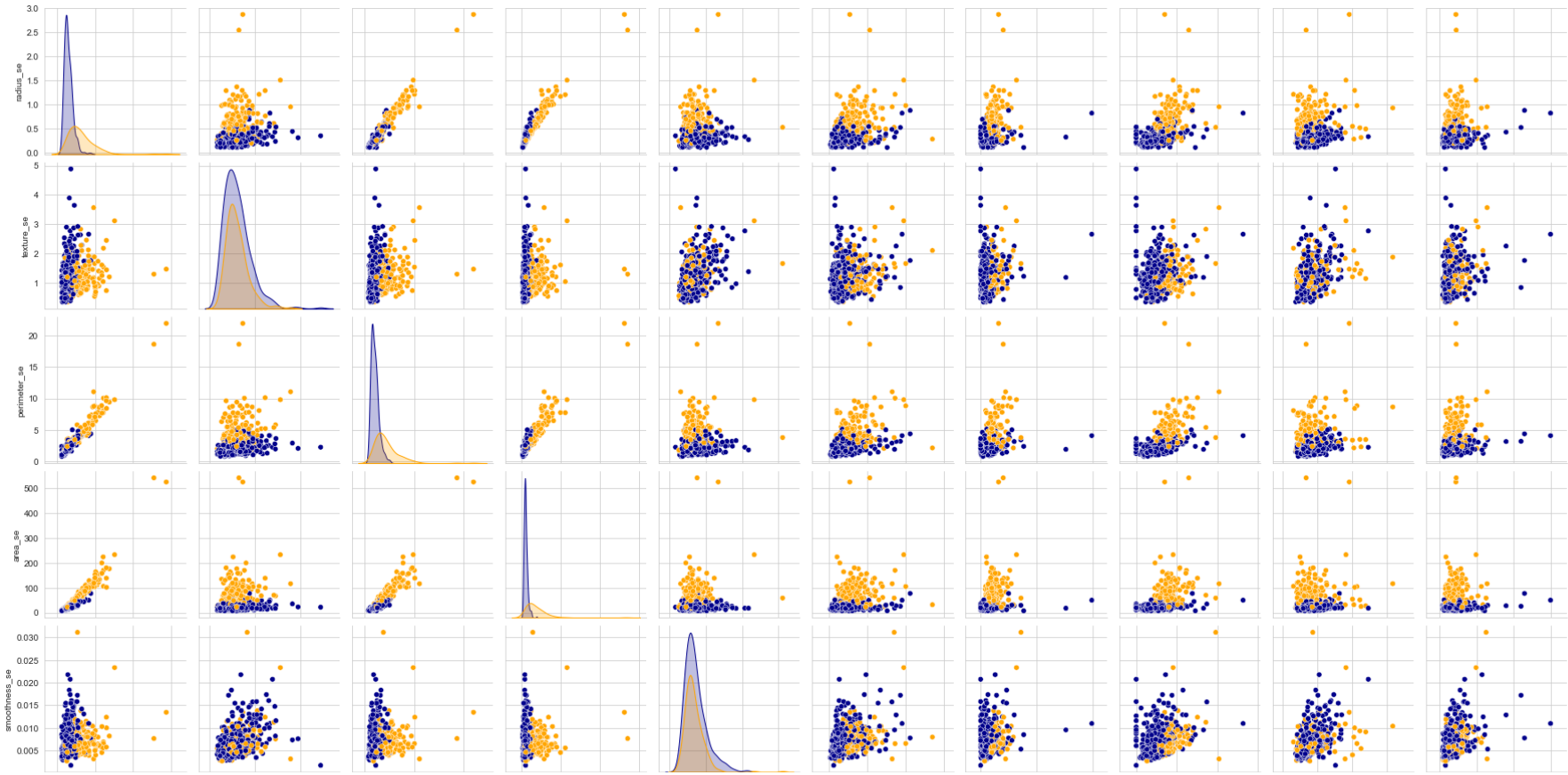


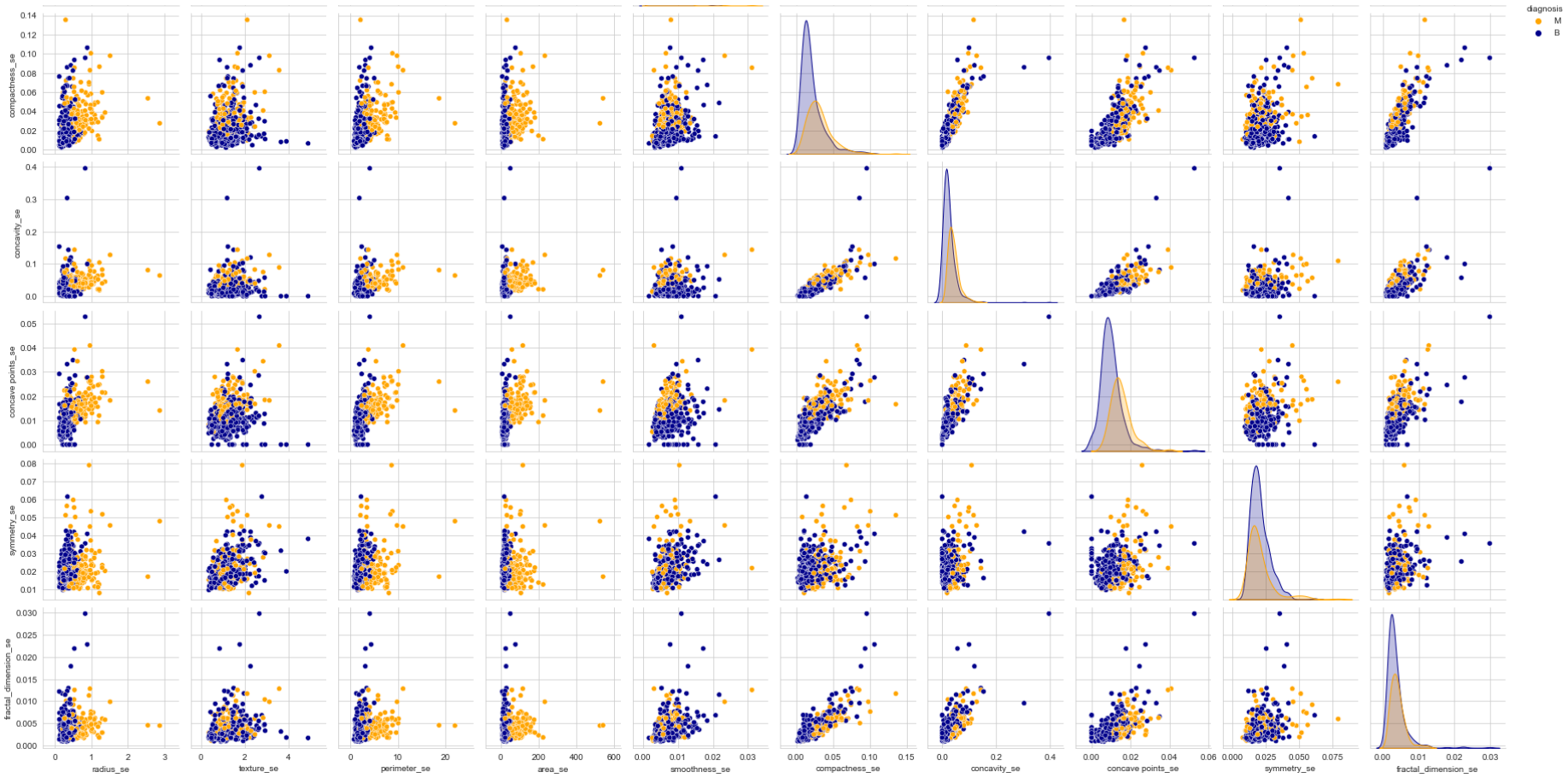
From the above pair plot we can say that data is highly correlated with ecah other

In [28]:
```
### Pairplot between Squared Error Features with Diagnosis
```

In [29]:
```
df_se = df[['diagnosis',
        'radius_se',
        'texture_se',
        'perimeter_se',
        'area_se',
        'smoothness_se',
        'compactness_se',
        'concavity_se',
        'concave points_se',
        'symmetry_se',
        'fractal_dimension_se']]
sns.pairplot(data=df_se, hue='diagnosis', palette=('Orange','DarkBlue'))
```

Out[29]: <seaborn.axisgrid.PairGrid at 0x1ca118b9340>





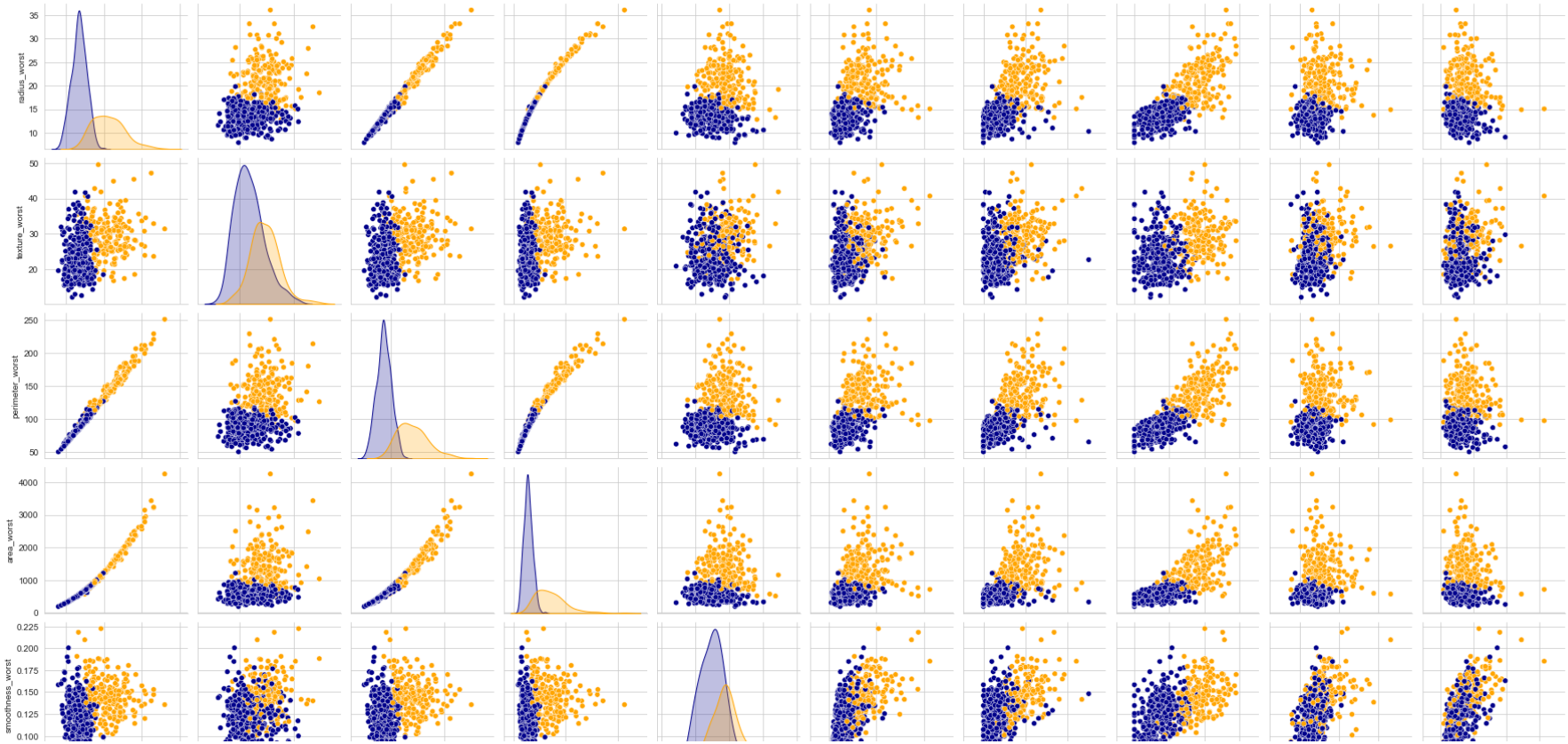
From the above pair plot we can say that data is some how correlated with ecah other

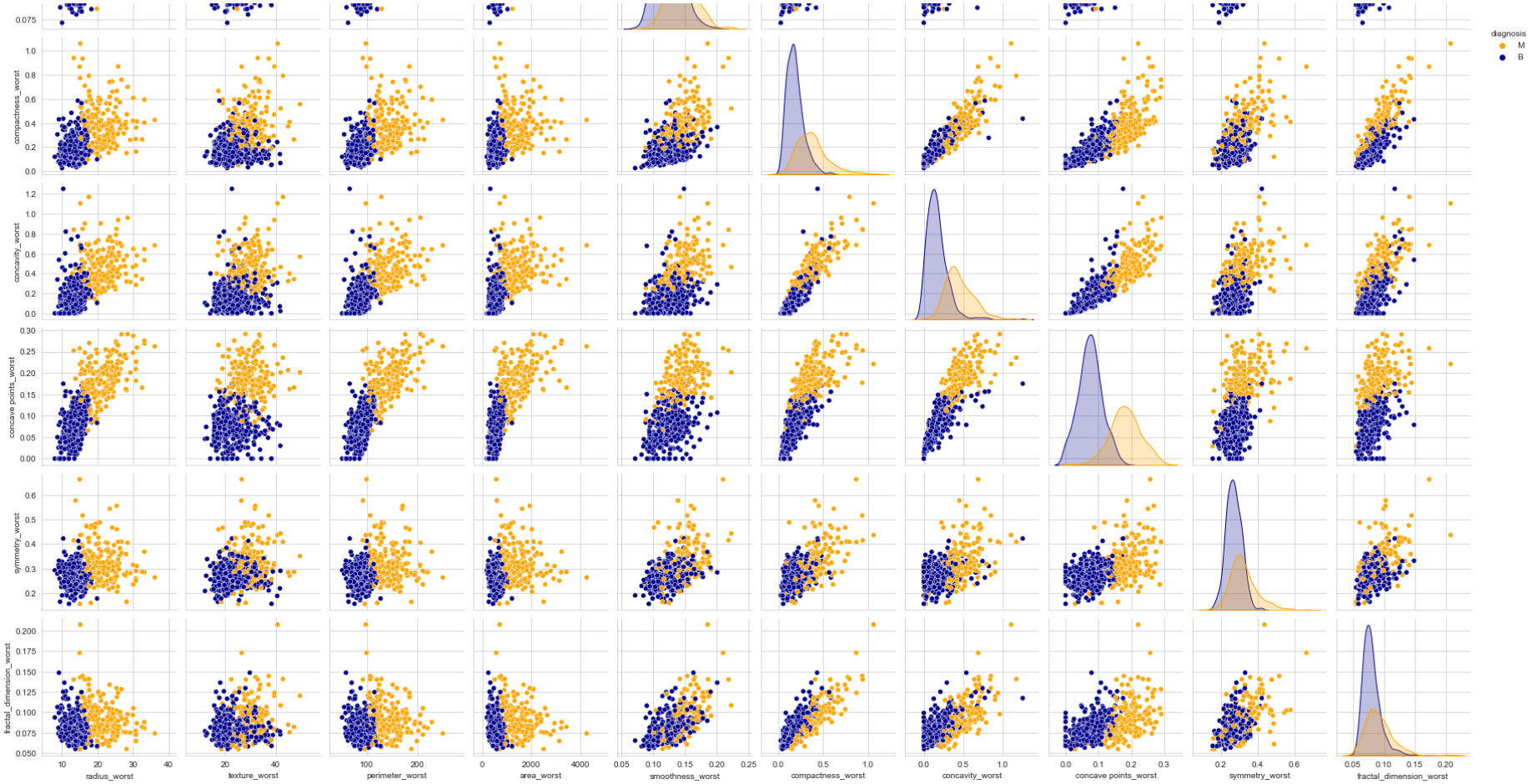
In [30]:
```
### Pairplot between  Worst Error Features with Diagnosis
```

In [31]:
```
df_worst = df[['diagnosis',
        'radius_worst',
        'texture_worst',
        'perimeter_worst',
        'area_worst',
        'smoothness_worst',
        'compactness_worst',
        'concavity_worst',
        'concave points_worst',
        'symmetry_worst',
        'fractal_dimension_worst']]

sns.pairplot(data=df_worst, hue='diagnosis', palette=('Orange','DarkBlue'))
```

Out[31]:
```
<seaborn.axisgrid.PairGrid at 0x1ca1b033cd0>
```



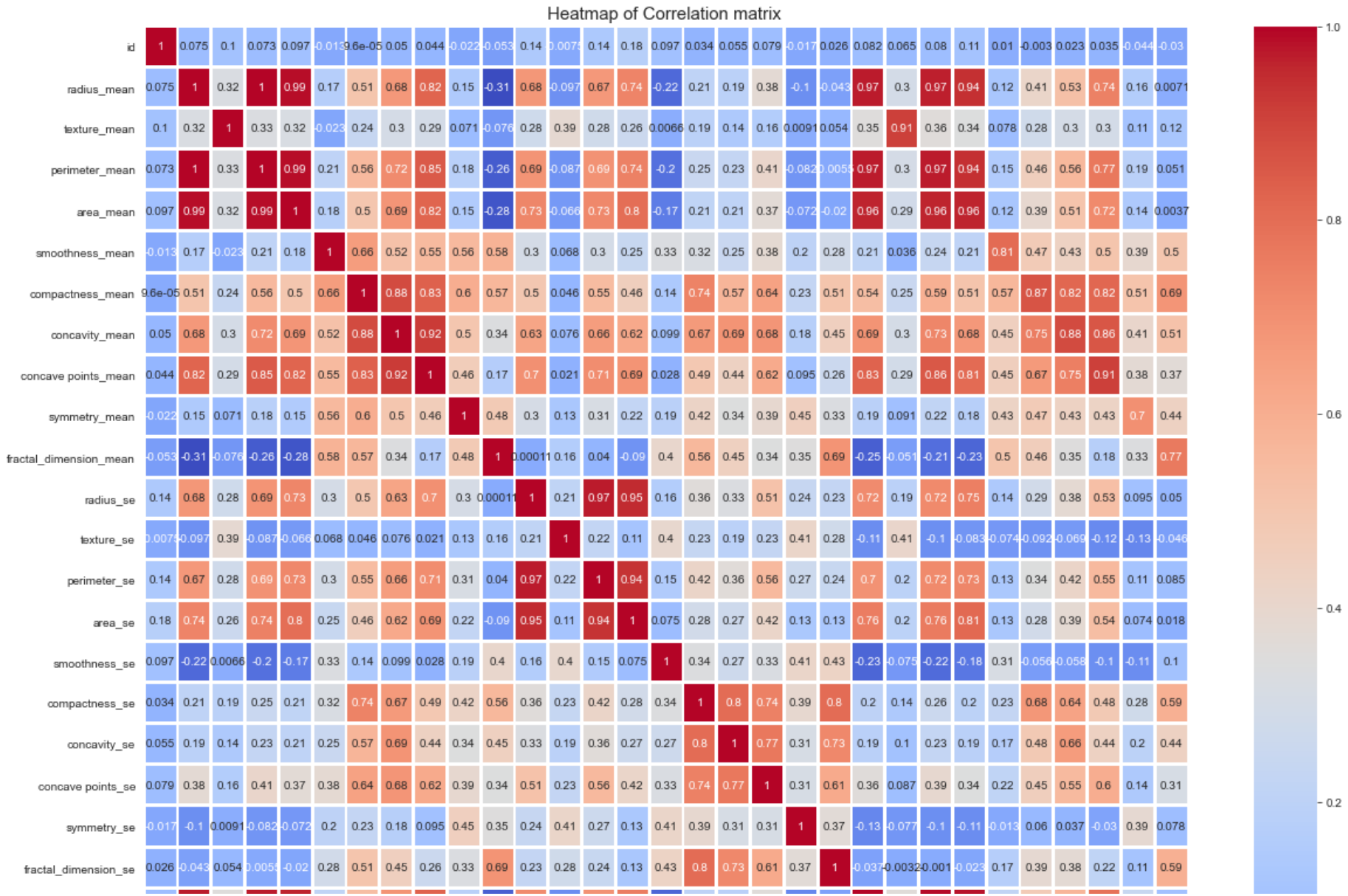


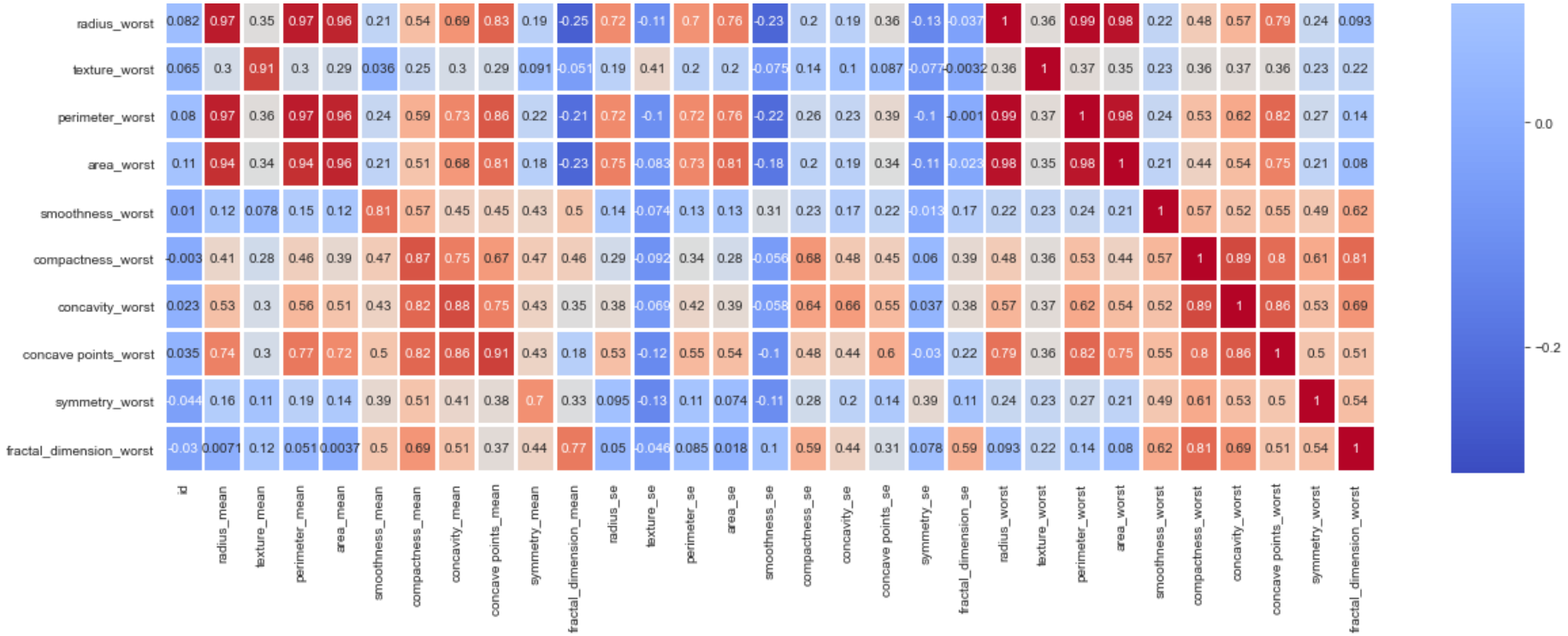
From aboove pairplots we can clearly say that data is highly colinear with each other

Correlation Between Data

In [33]:
```
plt.figure(figsize=(20,20))
plt.title("Heatmap of Correlation matrix ",fontsize=15)
sns.heatmap(df.corr(), annot = True, cmap ='coolwarm', linewidths=2)
```

Out[33]: <AxesSubplot:title={'center':'Heatmap of Correlation matrix '}>




radius_worst 0.082 0.97 0.35 0.97 0.96 0.21 0.54 0.69 0.83 0.19 -0.25 0.72 -0.11 0.7 0.76 -0.23 0.2 0.19 0.36 -0.13 -0.037 1 0.36 0.99 0.98 0.22 0.48 0.57 0.79 0.24 0.093
texture_worst 0.065 0.3 0.91 0.3 0.29 0.036 0.25 0.3 0.29 0.091 -0.051 0.19 0.41 0.2 0.2 -0.075 0.14 0.1 0.087 -0.077 -0.0032 0.36 1 0.37 0.35 0.23 0.36 0.37 0.36 0.23 0.22
perimeter_worst 0.08 0.97 0.36 0.97 0.96 0.24 0.59 0.73 0.86 0.22 -0.21 0.72 -0.1 0.72 0.76 -0.22 0.26 0.23 0.39 -0.1 -0.001 0.99 0.37 1 0.98 0.24 0.53 0.62 0.82 0.27 0.14
area_worst 0.11 0.94 0.34 0.94 0.96 0.21 0.51 0.68 0.81 0.18 -0.23 0.75 -0.083 0.73 0.81 -0.18 0.2 0.19 0.34 -0.11 -0.023 0.98 0.35 0.98 1 0.21 0.44 0.54 0.75 0.21 0.08
smoothness_worst 0.01 0.12 0.078 0.15 0.12 0.81 0.57 0.45 0.45 0.43 0.5 0.14 -0.074 0.13 0.13 0.31 0.23 0.17 0.22 -0.013 0.17 0.22 0.23 0.24 0.21 1 0.57 0.52 0.55 0.49 0.62
compactness_worst -0.003 0.41 0.28 0.46 0.39 0.47 0.87 0.75 0.67 0.47 0.46 0.29 -0.092 0.34 0.28 -0.056 0.68 0.48 0.45 0.06 0.39 0.48 0.36 0.53 0.44 0.57 1 0.89 0.8 0.61 0.81
concavity_worst 0.023 0.53 0.3 0.56 0.51 0.43 0.82 0.88 0.75 0.43 0.35 0.38 -0.069 0.42 0.39 -0.058 0.64 0.66 0.55 0.037 0.38 0.57 0.37 0.62 0.54 0.52 0.89 1 0.86 0.53 0.69
concave points_worst 0.035 0.74 0.3 0.77 0.72 0.5 0.82 0.86 0.91 0.43 0.18 0.53 -0.12 0.55 0.54 -0.1 0.48 0.44 0.6 -0.03 0.22 0.79 0.36 0.82 0.75 0.55 0.8 0.86 1 0.5 0.51
symmetry_worst -0.044 0.16 0.11 0.19 0.14 0.39 0.51 0.41 0.38 0.7 0.33 0.095 -0.13 0.11 0.074 -0.11 0.28 0.2 0.14 0.39 0.11 0.24 0.23 0.27 0.21 0.49 0.61 0.53 0.5 1 0.54
fractal_dimension_worst -0.03 0.0071 0.12 0.051 0.0037 0.5 0.69 0.51 0.37 0.44 0.77 0.05 -0.046 0.085 0.018 0.1 0.59 0.44 0.31 0.078 0.59 0.093 0.22 0.14 0.08 0.62 0.81 0.69 0.51 0.54 1
id
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
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
0.0
−0.2

In [34]:
```
df.corr(method='spearman')
```

Out[34]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
id	1.000000	0.043740	0.125809	0.043374	0.042829	-0.073225	-0.002247	0.000657
radius_mean	0.043740	1.000000	0.340956	0.997802	0.999602	0.148510	0.497578	0.645728
texture_mean	0.125809	0.340956	1.000000	0.348142	0.344145	0.024649	0.266499	0.342646
perimeter_mean	0.043374	0.997802	0.348142	1.000000	0.997068	0.182923	0.543925	0.681958
area_mean	0.042829	0.999602	0.344145	0.997068	1.000000	0.138053	0.488988	0.642557
smoothness_mean	-0.073225	0.148510	0.024649	0.182923	0.138053	1.000000	0.678806	0.518511
compactness_mean	-0.002247	0.497578	0.266499	0.543925	0.488988	0.678806	1.000000	0.896518
concavity_mean	0.000657	0.645728	0.342646	0.681958	0.642557	0.518511	0.896518	1.000000
concave points_mean	-0.003174	0.759702	0.306891	0.788629	0.755165	0.565172	0.848295	0.927352
symmetry_mean	-0.001102	0.120242	0.110130	0.150049	0.113928	0.542228	0.552203	0.446793
fractal_dimension_mean	-0.087017	-0.349931	-0.059303	-0.304891	-0.358425	0.588465	0.499195	0.258174
radius_se	-0.016157	0.550247	0.363621	0.560326	0.553388	0.334282	0.506582	0.575277
texture_se	0.032837	-0.144499	0.450720	-0.137578	-0.142469	0.091283	0.047766	0.051318
perimeter_se	0.002656	0.565520	0.386813	0.582789	0.568237	0.331360	0.583520	0.646199
area_se	0.002448	0.738077	0.395139	0.745824	0.741518	0.296059	0.539511	0.644344
smoothness_se	0.020673	-0.326385	0.037048	-0.311147	-0.327431	0.338692	0.127381	0.070321
compactness_se	0.037236	0.264904	0.263591	0.308620	0.260362	0.392455	0.817875	0.761230
concavity_se	0.029344	0.364555	0.287188	0.402277	0.362308	0.354730	0.772283	0.858306
concave points_se	0.016625	0.410576	0.238610	0.441996	0.406468	0.438826	0.732425	0.774656
symmetry_se	-0.028736	-0.241376	0.008945	-0.228187	-0.243507	0.150740	0.098388	0.022753
fractal_dimension_se	-0.020282	-0.008411	0.147605	0.032429	-0.012688	0.413429	0.621121	0.513593
radius_worst	0.025451	0.978604	0.366547	0.981244	0.979258	0.203453	0.542626	0.682316
texture_worst	0.077484	0.314911	0.909218	0.323109	0.318178	0.060645	0.255305	0.335866
perimeter_worst	0.026199	0.971555	0.375273	0.978980	0.971822	0.226345	0.592254	0.722424

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
area_worst	0.023137	0.978863	0.368335	0.980864	0.980264	0.191735	0.531590	0.676628
smoothness_worst	-0.080853	0.125789	0.101401	0.156611	0.119712	0.796085	0.578902	0.488775
compactness_worst	-0.011300	0.491357	0.290917	0.534565	0.485813	0.481384	0.901029	0.849985
concavity_worst	-0.013284	0.596043	0.339725	0.632106	0.593736	0.429107	0.837921	0.938543
concave points_worst	-0.012798	0.727265	0.319235	0.757526	0.723390	0.498868	0.825473	0.904938
symmetry_worst	-0.064977	0.174698	0.120693	0.199007	0.170860	0.393579	0.450333	0.383667
fractal_dimension_worst	-0.090637	0.044564	0.116144	0.088961	0.038758	0.511457	0.688986	0.541838

Data Preprocessing -

Feature elemination

In [35]:
```
# feature is not for our use as it consist of id of the patient
df.drop('id',axis=1,inplace= True)
```

In [36]:
```
df.isnull().sum().sum()
```

Out[36]:
```
0
```

Feature encoding

Converting Categorical features into numberic features

In [38]:
```
le = preprocessing.LabelEncoder()
df['diagnosis']=le.fit_transform(df['diagnosis'])
```

In [39]:
```
df['diagnosis'].unique()
```

Out[39]:
```
array([1, 0])
```

In [40]:
```
df.head()
```

Out[40]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symm
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

Now our data is ready to pass the model and perform ML building.

In [41]:
```
cancer_pos_rate = np.sum(df.diagnosis) / len(df.diagnosis) *100
print('Breast Cancer +ve rate - ',cancer_pos_rate)
```

```
Breast Cancer +ve rate -  37.258347978910365
```

The given dataset contains only 3.7 % of cancer +ve data which is balance data for building accurate model.

Spliting Dataset for Feature Selection of Model Building

In [42]:
```
data = df.copy()
data.head()
```

Out[42]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symm
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

In [43]:
```
data.shape,data.isnull().sum().sum()
```

Out[43]:
```
((569, 31), 0)
```

The process dataset contain 569 instance ,31 columns including target variable and zero null values

In [45]:
```
X =data.drop(['diagnosis'],axis = 1)
y = data['diagnosis']
```

In [46]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)
```

Data is split into train and testing part

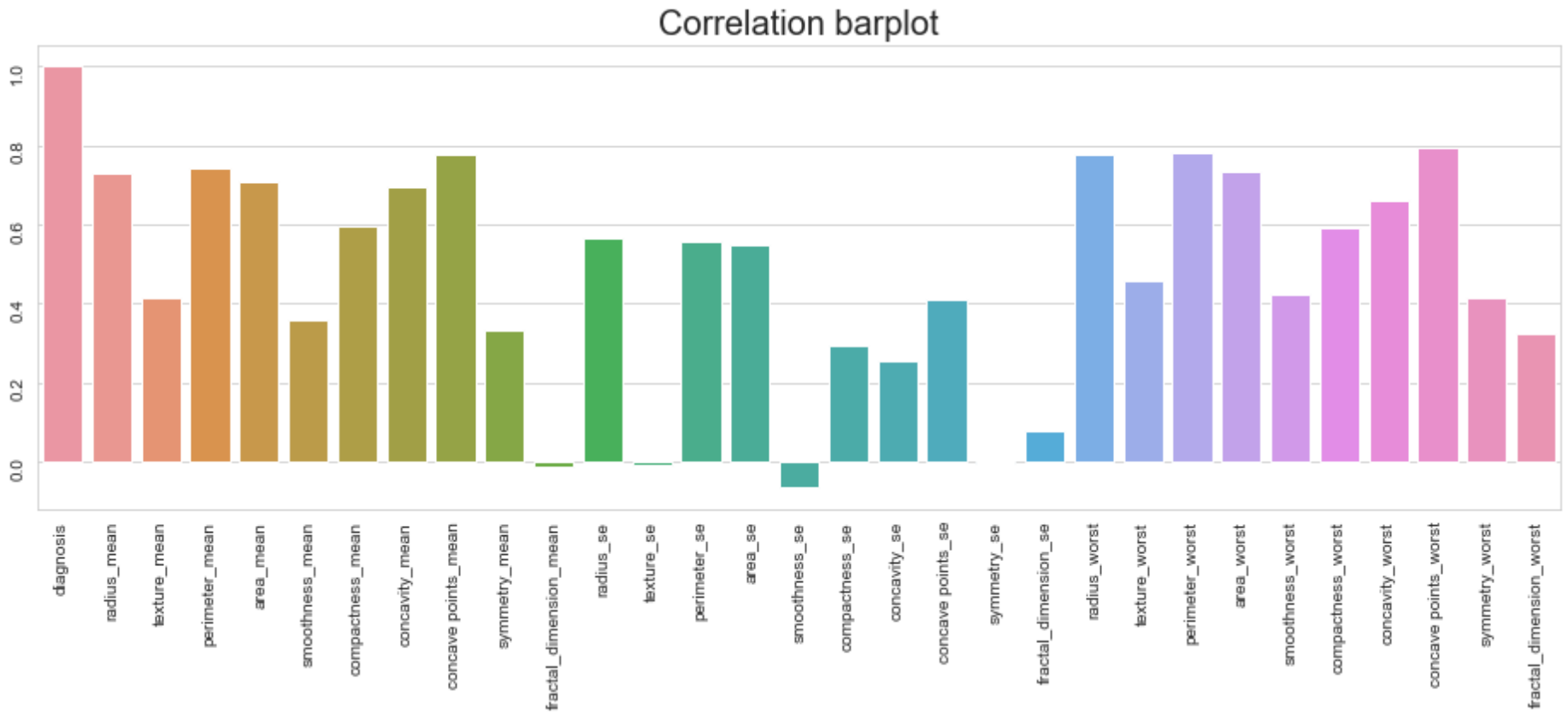
In [47]:
```
X_train.shape , X_test.shape, y_train.shape, y_test.shape
```

Out[47]:
```
((398, 30), (171, 30), (398,), (171,))
```

Calculating the feature correlated to Target variable

In [52]:
```
plt.figure(figsize = (16,5))
plt.title("Correlation barplot",fontsize= 20)
ax = sns.barplot(data.corrwith(df.diagnosis).index, data.corrwith(data.diagnosis))

ax.tick_params(labelrotation = 90)
```



from the above figure we can say that concave points_worst,radius_worst,perimeter_worst,concave points_means contributes toward target varibale more than other features

Conclusion

Thus we have successfully implemented pre-processing operations on a dataset

In [ ]: