In [90]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

In [2]:
```
data=pd.read_csv("wisc_bc_data.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poin
0	87139402	B	12.32	12.39	78.85	464.1	0.10280	0.06981	0.03987	
1	8910251	B	10.60	18.95	69.28	346.4	0.09688	0.11470	0.06387	
2	905520	B	11.04	16.83	70.92	373.2	0.10770	0.07804	0.03046	
3	868871	B	11.28	13.39	73.00	384.8	0.11640	0.11360	0.04635	
4	9012568	B	15.19	13.21	97.65	711.8	0.07963	0.06934	0.03393	

5 rows × 32 columns

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 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   points_mean        569 non-null    float64
 10  symmetry_mean      569 non-null    float64
 11  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  points_se          569 non-null    float64
 20  symmetry_se        569 non-null    float64
 21  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  points_worst       569 non-null    float64
 30  symmetry_worst     569 non-null    float64
 31  dimension_worst    569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [5]:
```
data.describe()
```

Out[5]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	points

count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.(
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.(
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.(
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.(
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.(
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.(
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.(
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.2

8 rows × 31 columns

In [6]:
```
data.columns
```

Out[6]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'points_mean', 'symmetry_mean', 'dimension_mean', 'radius_se',
       'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'points_se', 'symmetry_se',
       'dimension_se', 'radius_worst', 'texture_worst', 'perimeter_worst',
       'area_worst', 'smoothness_worst', 'compactness_worst',
       'concavity_worst', 'points_worst', 'symmetry_worst', 'dimension_worst'],
      dtype='object')
```

In [7]:
```
data.shape
```

Out[7]:
```
(569, 32)
```

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
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
points_mean              0
symmetry_mean            0
dimension_mean           0
radius_se                0
texture_se               0
perimeter_se             0
area_se                  0
smoothness_se            0
compactness_se           0
concavity_se             0
points_se                0
symmetry_se              0
dimension_se             0
radius_worst             0
texture_worst            0
perimeter_worst          0
area_worst               0
smoothness_worst         0
compactness_worst        0
concavity_worst          0
points_worst             0
symmetry_worst           0
dimension_worst          0
dtype: int64
```

NO MISSING VALUES

In [12]:
```
df=pd.get_dummies(data,columns=['diagnosis'])
```

In [13]:
```
df.sample()
```

```
df.sample()
```

Out[13]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	points_mean
544	8610862	20.18	23.97	143.7	1245.0	0.1286	0.3454	0.3754	0.1604

1 rows × 33 columns

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

Out[14]:
```
id                   906024.000000
radius_mean              13.370000
texture_mean             18.840000
perimeter_mean           86.240000
area_mean               551.100000
smoothness_mean           0.095870
compactness_mean          0.092630
concavity_mean            0.061540
points_mean               0.033500
symmetry_mean             0.179200
dimension_mean            0.061540
radius_se                 0.324200
texture_se                1.108000
perimeter_se              2.287000
area_se                  24.530000
smoothness_se             0.006380
compactness_se            0.020450
concavity_se              0.025890
points_se                 0.010930
symmetry_se               0.018730
dimension_se              0.003187
radius_worst             14.970000
texture_worst            25.410000
perimeter_worst          97.660000
area_worst              686.500000
smoothness_worst          0.131300
compactness_worst         0.211900
concavity_worst           0.226700
points_worst              0.099930
symmetry_worst            0.282200
dimension_worst           0.080040
diagnosis_B               1.000000
diagnosis_M               0.000000
dtype: float64
```

In [15]:
```
corr=data.corr()
```

In [16]:
```
corr
```

Out[16]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
id	1.000000	0.074626	0.099770	0.073159	0.096893	-0.012968	0.000096	0.050080
radius_mean	0.074626	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764
texture_mean	0.099770	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418
perimeter_mean	0.073159	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136
area_mean	0.096893	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983
smoothness_mean	-0.012968	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984
compactness_mean	0.000096	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121
concavity_mean	0.050080	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000
points_mean	0.044158	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391
symmetry_mean	-0.022114	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667
dimension_mean	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783
radius_se	0.143048	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925
texture_se	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218
perimeter_se	0.137331	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391
area_se	0.177742	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.617427
smoothness_se	0.096781	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564
compactness_se	0.033961	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722	0.670279
concavity_se	0.055239	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270

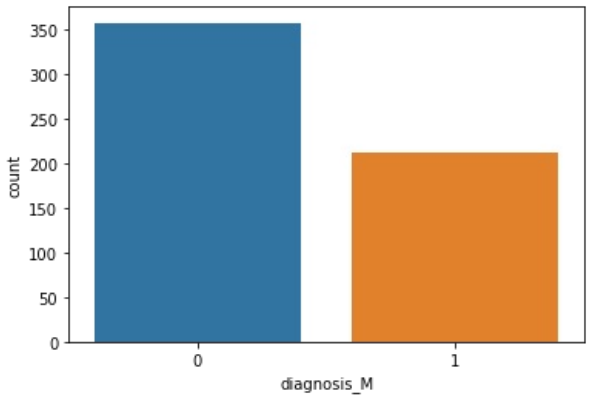
points_se	0.078768	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262	0.68326
symmetry_se	-0.017306	-0.104321	0.009127	-0.081629	-0.072497	0.200774	0.229977	0.17800
dimension_se	0.025725	-0.042641	0.054458	-0.005523	-0.019887	0.283607	0.507318	0.44930
radius_worst	0.082405	0.969539	0.352573	0.969476	0.962746	0.213120	0.535315	0.68823
texture_worst	0.064720	0.297008	0.912045	0.303038	0.287489	0.036072	0.248133	0.29987
perimeter_worst	0.079986	0.965137	0.358040	0.970387	0.959120	0.238853	0.590210	0.72956
area_worst	0.107187	0.941082	0.343546	0.941550	0.959213	0.206718	0.509604	0.67598
smoothness_worst	0.010338	0.119616	0.077503	0.150549	0.123523	0.805324	0.565541	0.44882
compactness_worst	-0.002968	0.413463	0.277830	0.455774	0.390410	0.472468	0.865809	0.75496
concavity_worst	0.023203	0.526911	0.301025	0.563879	0.512606	0.434926	0.816275	0.88410
points_worst	0.035174	0.744214	0.295316	0.771241	0.722017	0.503053	0.815573	0.86132
symmetry_worst	-0.044224	0.163953	0.105008	0.189115	0.143570	0.394309	0.510223	0.40946
dimension_worst	-0.029866	0.007066	0.119205	0.051019	0.003738	0.499316	0.687382	0.51493

31 rows × 31 columns

DATA VISUALIZATION

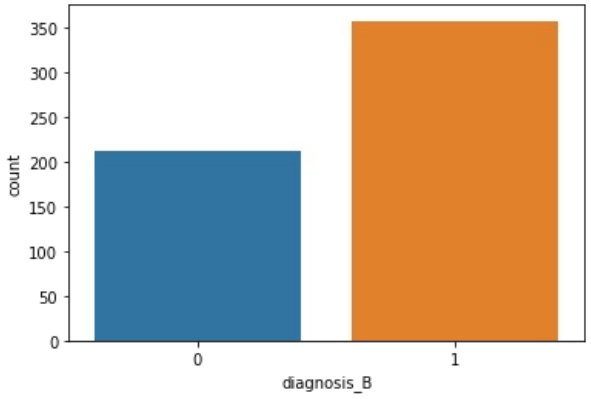
In [176…
```
sns.countplot(x='diagnosis_M',data=df)
```

Out[176…
```
<AxesSubplot:xlabel='diagnosis_M', ylabel='count'>
```



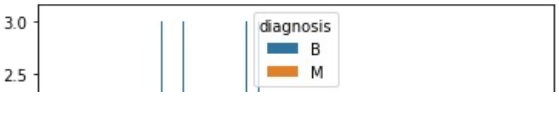
In [21]:
```
sns.countplot(df['diagnosis_B'])
```

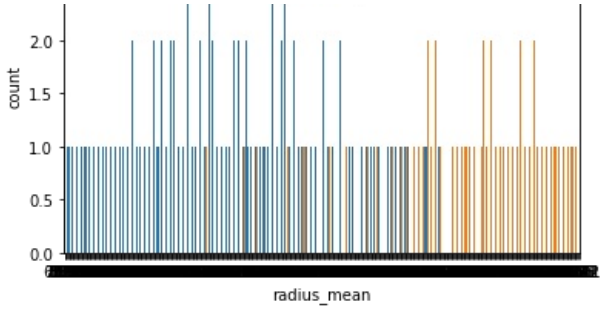
Out[21]:
```
<AxesSubplot:xlabel='diagnosis_B', ylabel='count'>
```



In [54]:
```
sns.countplot(x='radius_mean',hue='diagnosis',data=data)
```

Out[54]:
```
<AxesSubplot:xlabel='radius_mean', ylabel='count'>
```

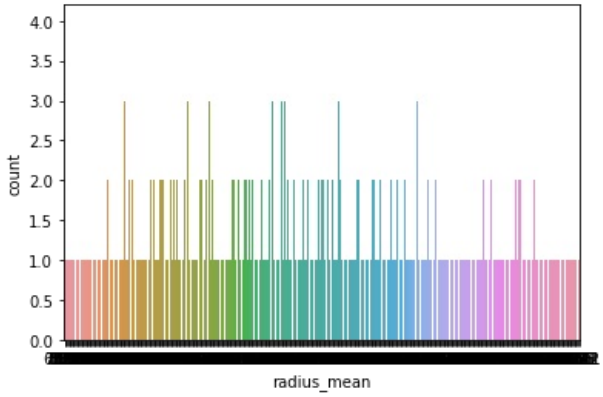




In [44]:
```
sns.countplot(data['radius_mean'])
```

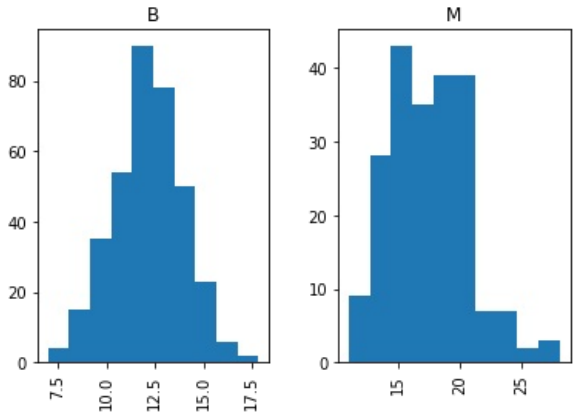
```
d:\Users\Lalithya\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.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

Out[44]:
```
<AxesSubplot:xlabel='radius_mean', ylabel='count'>
```



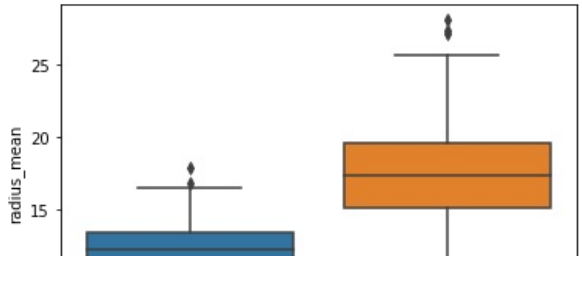
In [45]:
```
data.hist(by='diagnosis',column='radius_mean')
```

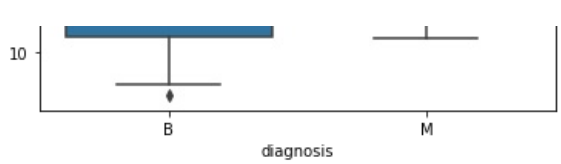
Out[45]:
```
array([<AxesSubplot:title={'center':'B'}>,
       <AxesSubplot:title={'center':'M'}>], dtype=object)
```



In [52]:
```
sns.boxplot(x='diagnosis',y='radius_mean',data=data)
```

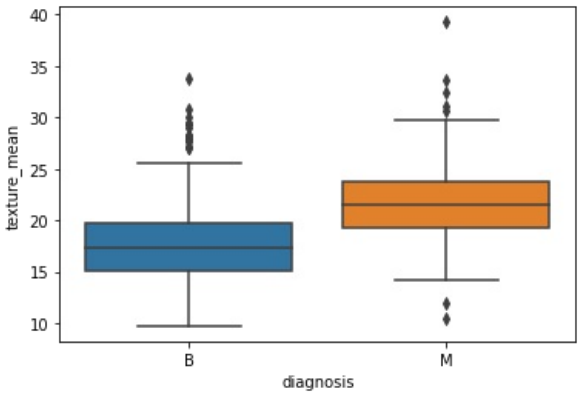
Out[52]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='radius_mean'>
```





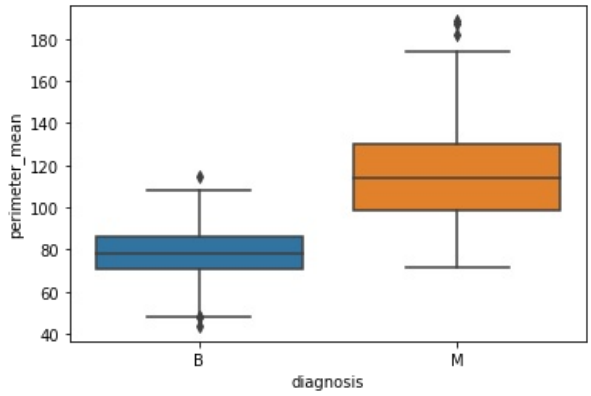
In [57]:
```
sns.boxplot(x='diagnosis',y='texture_mean',data=data)
```

Out[57]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='texture_mean'>
```



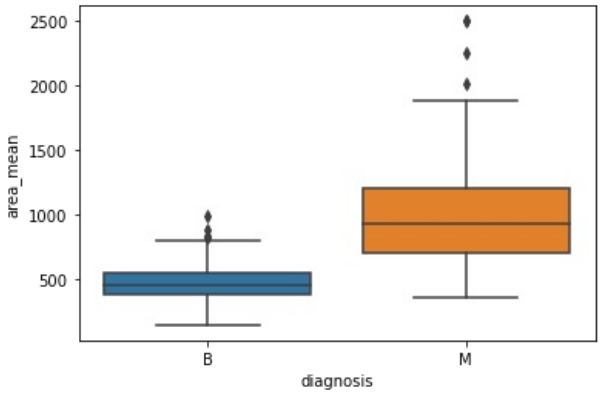
In [60]:
```
sns.boxplot(x='diagnosis',y='perimeter_mean',data=data)
```

Out[60]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='perimeter_mean'>
```



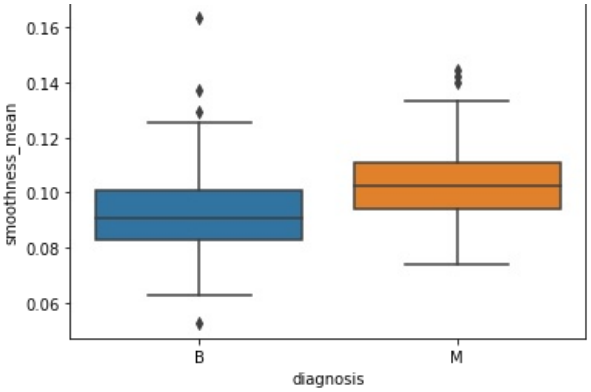
In [61]:
```
sns.boxplot(x='diagnosis',y='area_mean',data=data)
```

Out[61]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='area_mean'>
```



In [62]:
```
sns.boxplot(x='diagnosis',y='smoothness_mean',data=data)
```

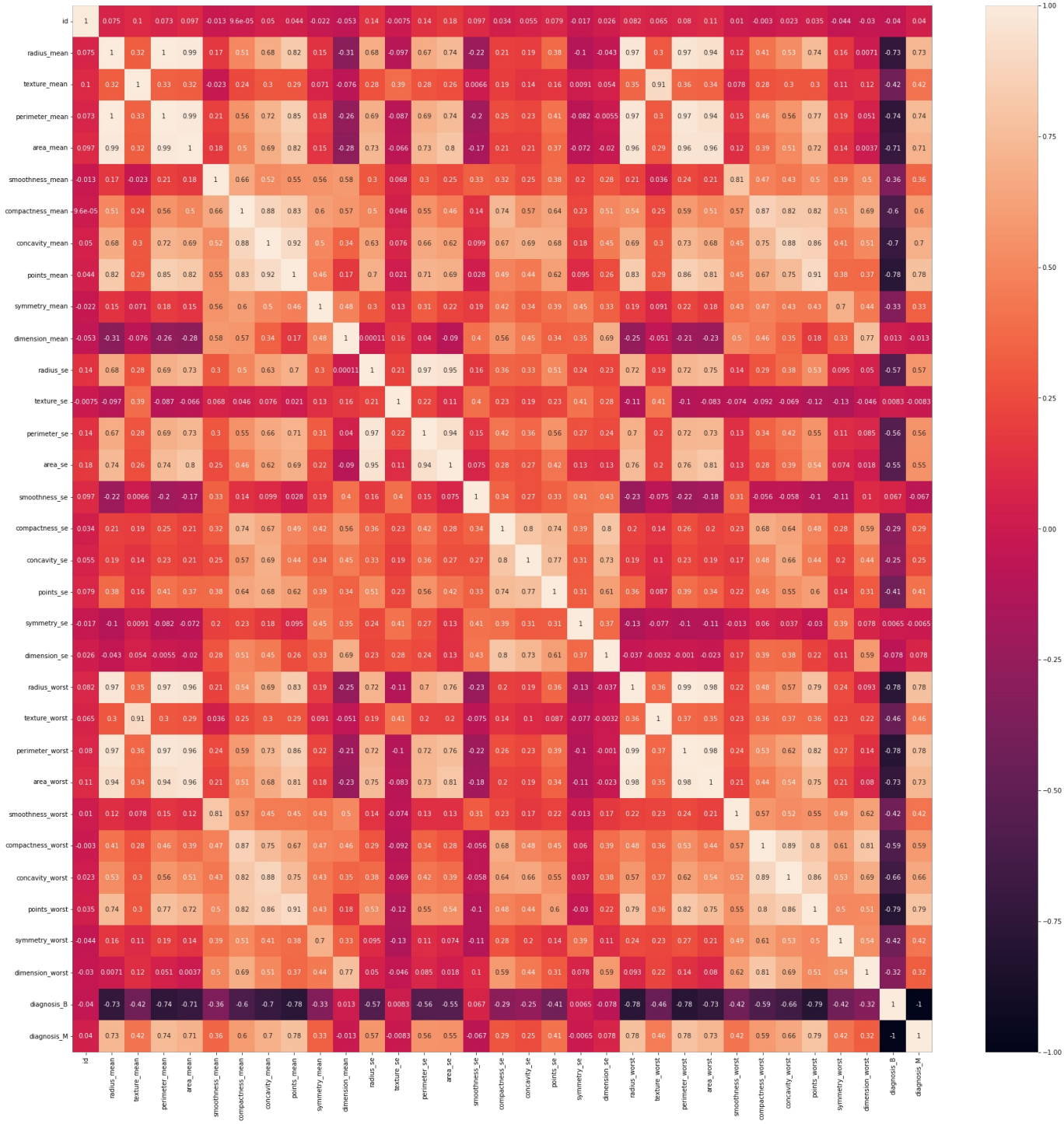
Out[62]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='smoothness_mean'>
```



In [76]:

```
plt.figure(figsize=(30, 30))
sns.heatmap(df.corr(), annot=True)
```

Out[76]: <AxesSubplot:>

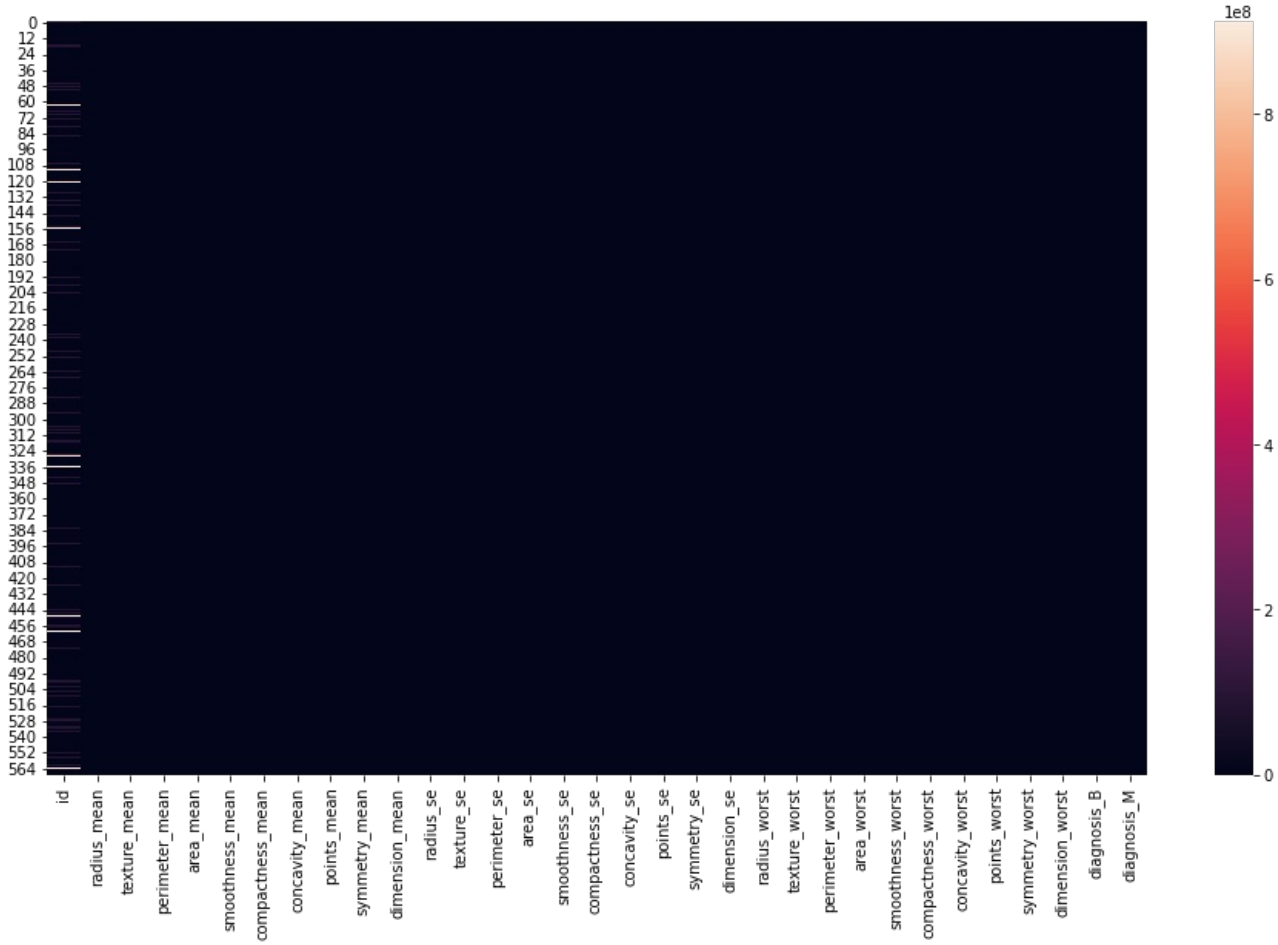


In [78]:

```
plt.figure(figsize=(16,9))
```

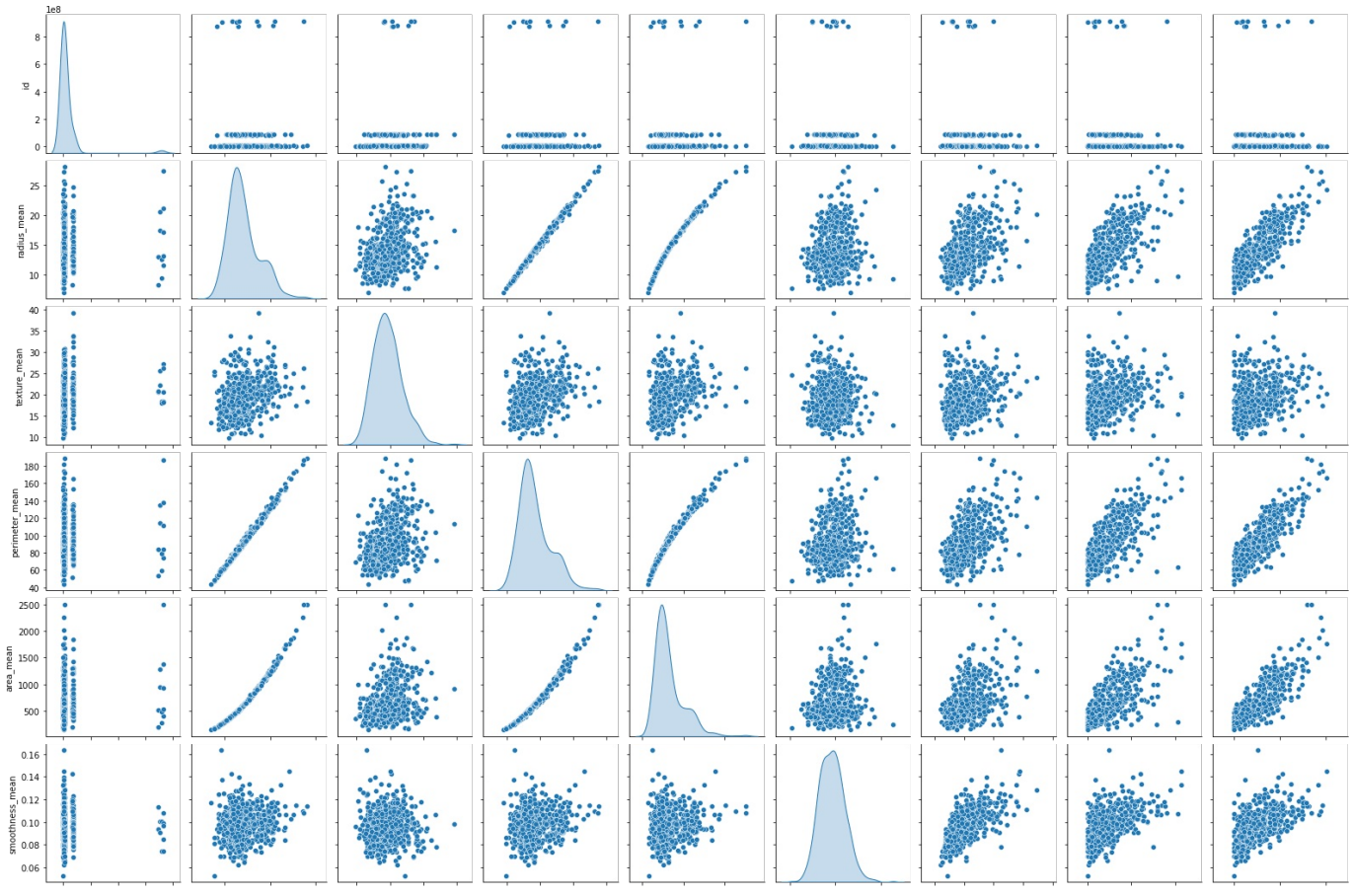
```
plt.figure(figsize=(10,5))
sns.heatmap(df)
```

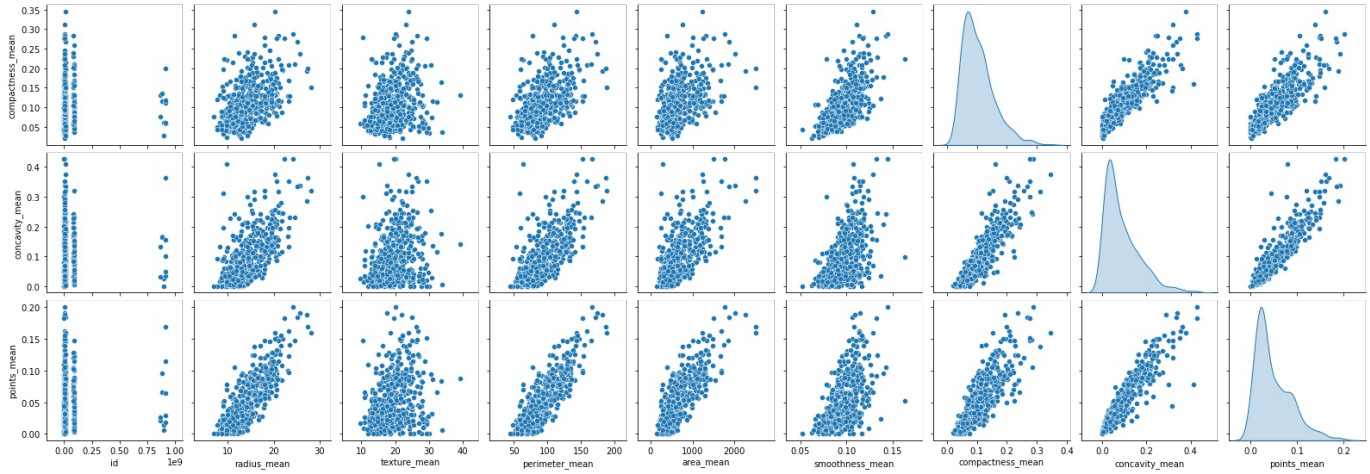
Out[78]: <AxesSubplot:>



In [80]:
```
df_attr = df.iloc[:, 0:9]
sns.pairplot(df_attr, diag_kind='kde')
```

Out[80]: <seaborn.axisgrid.PairGrid at 0x1c552970b50>





SPLITTING OF DATA

In [141…
```
x=df.drop(['diagnosis_B','diagnosis_M'],axis=1)
y=df['diagnosis_M']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.35,random_state=5)
```

In [142…
```
model=LogisticRegression()
model.fit(x_train,y_train)
```

Out[142…
```
LogisticRegression()
```

In [143…
```
dF=model.predict(x_test)
```

In [144…
```
accuracy_score(y_test,dF)
```

Out[144…
```
0.645
```

In [145…
```
accuracy_score(y_test,dF)
```

Out[145…
```
0.645
```

In [146…
```
dtree=DecisionTreeClassifier(criterion='gini',random_state=5)
```

In [147…
```
dtree.fit(x_train,y_train)
```

Out[147…
```
DecisionTreeClassifier(random_state=5)
```

In [148…
```
DecisionTreeClassifier(random_state=5)
```

Out[148…
```
DecisionTreeClassifier(random_state=5)
```

In [149…
```
print(dtree.score(x_train,y_train))
print(dtree.score(x_test,y_test))
```

```
1.0
0.945
```

FITTING THE MODEL

In [150…
```
dtree=DecisionTreeClassifier(criterion='entropy',random_state=5)
dtree.fit(x_train,y_train)
```

Out[150…
```
DecisionTreeClassifier(criterion='entropy', random_state=5)
```

In [151…
```
print(dtree.score(x_train,y_train))
print(dtree.score(x_test,y_test))
```

```
1.0
0.95
```

IT ISN'T A GOOD MODEL

In [160…
```
dtree=DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=5)
dtree.fit(x_train,y_train)
print(dtree.score(x_train,y_train))
print(dtree.score(x_test,y_test))
```

```
0.975609756097561
0.945
```

In [161…
```
bg=BaggingClassifier(n_estimators=50,base_estimator=dtree,random_state=5)
bg=bg.fit(x_train,y_train)
y_predict=bg.predict(x_test)
print(bg.score(x_test,y_test))
```

```
0.96
```

In [162…
```
ad=AdaBoostClassifier(n_estimators=50,random_state=5)
ad=ad.fit(x_train,y_train)
y_predict=ad.predict(x_test)
print(ad.score(x_test,y_test))
```

```
0.975
```

IT IS A GOOD MODEL

In [163…
```
gd=GradientBoostingClassifier(n_estimators=50,random_state=5)
gd=gd.fit(x_train,y_train)
y_predict=gd.predict(x_test)
print(gd.score(x_test,y_test))
```

```
0.965
```

In [174…
```
rf=RandomForestClassifier(n_estimators=75,random_state=5,max_features=3)
rf=rf.fit(x_train,y_train)
y_predict=rf.predict(x_test)
print(rf.score(x_test,y_test))
```

```
0.97
```

ADABOOSTCLASSIFIER IS BEST CLASSIFIER FOR THE GIVEN DATA