In [ ]:
```
# Start of the Project
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import seaborn as sns
import time
%matplotlib inline

#Import models from scikit learn module:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC

from sklearn import metrics
```

In [3]:
```
data = pd.read_csv("breastcancer.csv")
data.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
0	842302	M	17.99	10.38	122.80	1001.0	0.1184
1	842517	M	20.57	17.77	132.90	1326.0	0.0847
2	84300903	M	19.69	21.25	130.00	1203.0	0.1096
3	84348301	M	11.42	20.38	77.58	386.1	0.1425
4	84358402	M	20.29	14.34	135.10	1297.0	0.1003

5 rows × 33 columns

In [4]:
```
# Data visualisation and preprocessing
data.drop('id',axis=1,inplace=True) #dropping the 'id' column
data.drop('Unnamed: 32',axis=1,inplace=True)
print("Row, Col", data.shape)# (row,col)
```

```
Row, Col (569, 31)
```

In [5]:
```
data['diagnosis'] = data['diagnosis'].map({'M':1,'B':0})
data.head()
```

Out[5]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compact
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

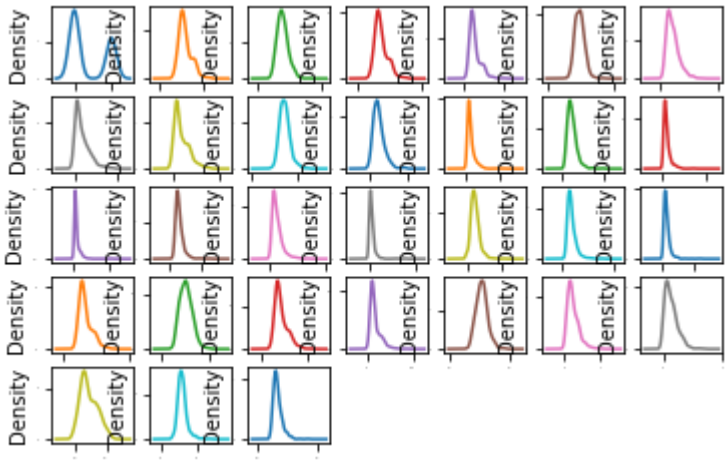
In [24]:
```
data.describe()
```

Out[24]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	c
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360	
std	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064	
min	0.000000	6.981000	9.710000	43.790000	143.500000	0.052630	
25%	0.000000	11.700000	16.170000	75.170000	420.300000	0.086370	
50%	0.000000	13.370000	18.840000	86.240000	551.100000	0.095870	
75%	1.000000	15.780000	21.800000	104.100000	782.700000	0.105300	
max	1.000000	28.110000	39.280000	188.500000	2501.000000	0.163400	

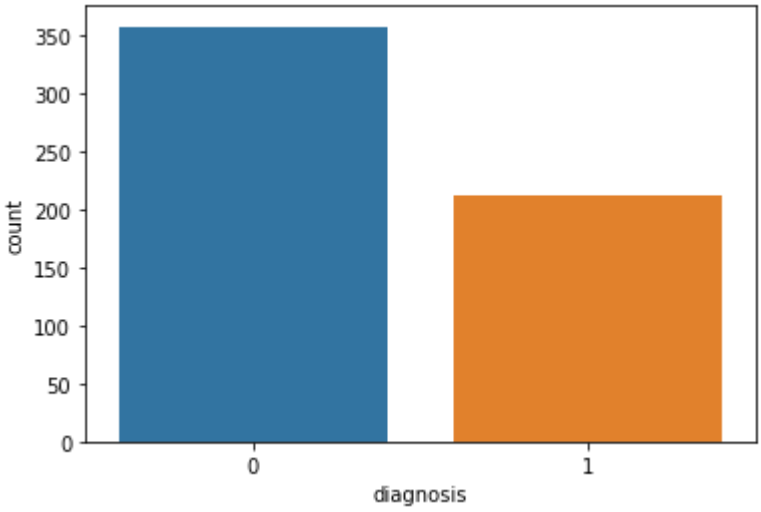
8 rows × 31 columns

In [6]:
```
data.plot(kind='density', subplots=True, layout=(5,7), sharex=False, legend=Fals
plt.show()
```

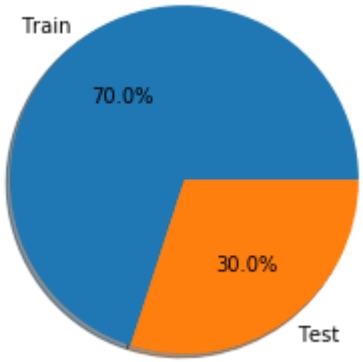


In [26]:
```
print(data.groupby('diagnosis').size())
sns.countplot(data['diagnosis'],label="Count")
plt.show()
```

```
diagnosis
0    357
1    212
dtype: int64
```

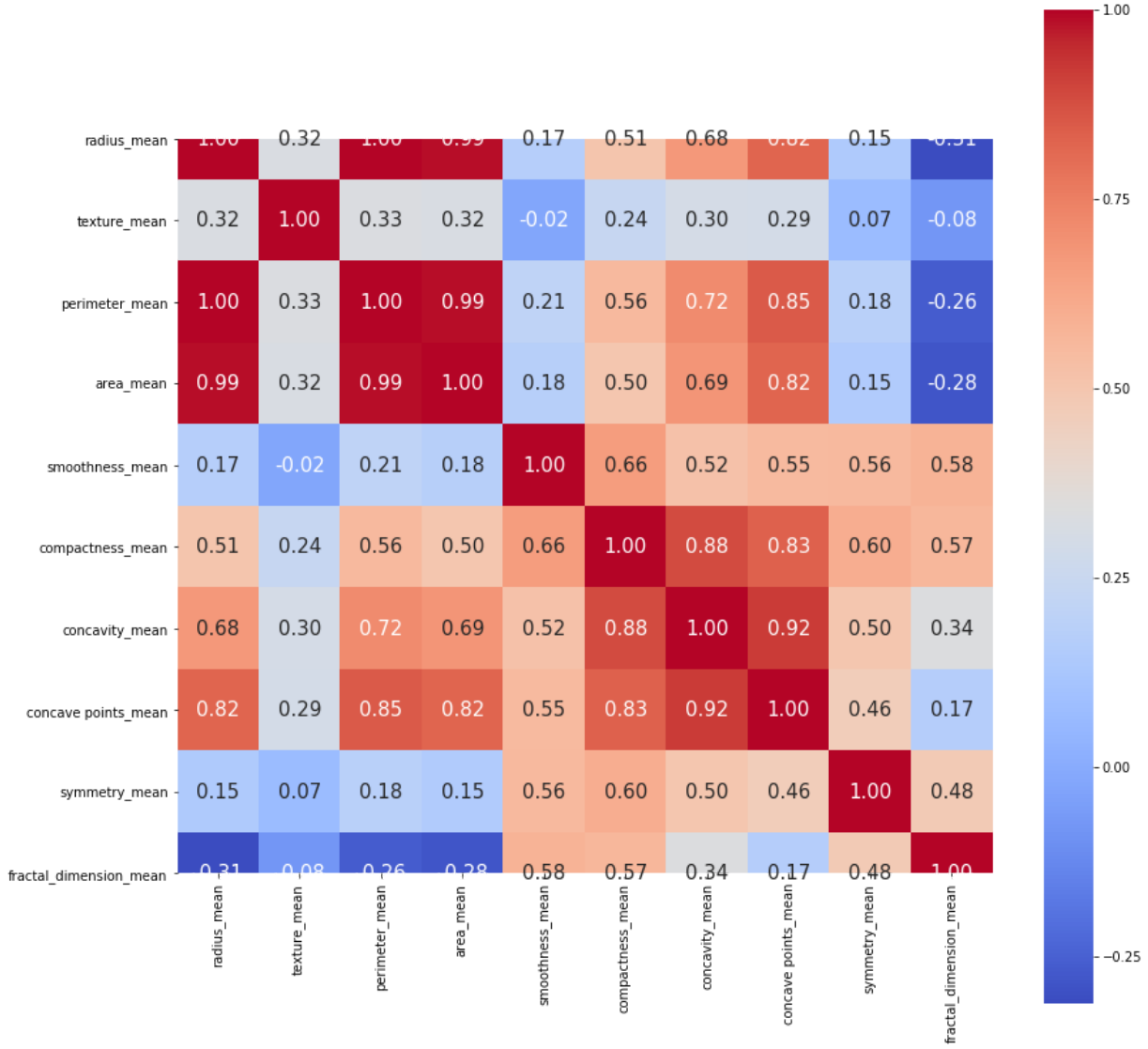


In [9]:
```
#split our data into train and test
traindf, testdf = train_test_split(data, test_size = 0.3)
labels = 'Train', 'Test'
plt.pie([70, 30], labels=labels, autopct='%1.1f%%', shadow=True)
plt.show()
print("Train set", traindf.shape)
print("Test set", testdf.shape)
```




```
Train set (398, 31)
Test set (171, 31)
```

In [10]:
```
features_mean= list(data.columns[1:11])
corr = data[features_mean].corr() # .corr is used for find corelation
plt.figure(figsize=(14,14))
sns.heatmap(corr, cbar = True,  square = True, annot=True, fmt= '.2f',annot_kws=
            xticklabels= features_mean, yticklabels= features_mean,
            cmap= 'coolwarm')
plt.show()
```



In [11]:
```
#Generic function for making a classification model and accessing the performance
from sklearn.model_selection import KFold
Y = data['diagnosis'].values
X = data.drop('diagnosis', axis=1).values
X_train, X_test, Y_train, Y_test = train_test_split (X, Y, test_size = 0.30, rand
def classification_model(model, data, predictors, outcome):
  #Fit the model:
  model.fit(data[predictors],data[outcome])

  #Make predictions on training set:
  predictions = model.predict(data[predictors])

  #Print accuracy
  accuracy = metrics.accuracy_score(predictions,data[outcome])
  print("Accuracy : %s" % "{0:.2%}".format(accuracy))
```

In [12]:
```
# Logistic Regression
predictor_var = ['texture_mean','perimeter_mean','smoothness_mean','compactness_
outcome_var='diagnosis'
model=LogisticRegression()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 89.20%

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

In [13]:
```
# Decisiontree Classifier
predictor_var = ['texture_mean','perimeter_mean','smoothness_mean','compactness_
model = DecisionTreeClassifier()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 100.00%
```

In [14]:
```
# SVM model
predictor_var = ['texture_mean','perimeter_mean','smoothness_mean','compactness_
model= SVC()
classification_model(model,traindf,predictor_var,outcome_var)
```

```
Accuracy : 93.72%

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarn
ing: The default value of gamma will change from 'auto' to 'scale' in version
0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or
'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [15]:
```
# Runtime of algorithms
Y = data['diagnosis'].values
X = data.drop('diagnosis', axis=1).values
X_train, X_test, Y_train, Y_test = train_test_split (X, Y, test_size = 0.30, rand
models_list = []
models_list.append(('LR', LogisticRegression()))
models_list.append(('DT', DecisionTreeClassifier()))
models_list.append(('SVM', SVC()))
num_folds = 3
results = []
names = []
for name, model in models_list:
    start = time.time()
    cv_results = cross_val_score(model, X_train, Y_train, cv=num_folds, scoring=
    end = time.time()
    results.append(cv_results)
    names.append(name)
    print( "%s:(run time: %f)" % (name, end-start))
```

```
LR:(run time: 0.012010)
DT:(run time: 0.010737)
SVM:(run time: 0.031914)

C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarn
ing: The default value of gamma will change from 'auto' to 'scale' in version
0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or
'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarn
ing: The default value of gamma will change from 'auto' to 'scale' in version
0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or
'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
C:\Users\ADITYA\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarn
ing: The default value of gamma will change from 'auto' to 'scale' in version
0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or
'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

In [ ]:
```
# End of the project
```