Marc Riley

Breast cancer detection

In [37]:
```
# import libraries used for EDA
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [38]:
```
#Load Data
df = pd.read_csv('data.csv')
```

In [39]:
```
# Examine the data
df.head()
```

Out[39]:

	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 [40]:
```
df.shape
```

Out[40]:
```
(569, 33)
```

In [41]:
```
# find missing values
for c in df.columns:
    miss = df[c].isnull().sum()
    if miss >0:
        print("{} has {} missing values".format(c,miss))
    else:
        print("{} column has no missing values!".format(c))
```

```
id column has no missing values!
diagnosis column has no missing values!
radius_mean column has no missing values!
texture_mean column has no missing values!
perimeter_mean column has no missing values!
area_mean column has no missing values!
smoothness_mean column has no missing values!
compactness_mean column has no missing values!
concavity_mean column has no missing values!
concave points_mean column has no missing values!
symmetry_mean column has no missing values!
fractal_dimension_mean column has no missing values!
radius_se column has no missing values!
texture_se column has no missing values!
perimeter_se column has no missing values!
area_se column has no missing values!
smoothness_se column has no missing values!
compactness_se column has no missing values!
concavity_se column has no missing values!
concave points_se column has no missing values!
symmetry_se column has no missing values!
fractal_dimension_se column has no missing values!
radius_worst column has no missing values!
texture_worst column has no missing values!
perimeter_worst column has no missing values!
area_worst column has no missing values!
smoothness_worst column has no missing values!
compactness_worst column has no missing values!
concavity_worst column has no missing values!
concave points_worst column has no missing values!
symmetry_worst column has no missing values!
fractal_dimension_worst column has no missing values!
Unnamed: 32 has 569 missing values
```

In [42]:
```
# remove'unnamed column'
df.drop(['Unnamed: 32', 'id'], axis=1,inplace=True)
```

In [43]:
```
df.head()
```

Out[43]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compac
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 [44]:
```
#replace M/B with 1/0
df['diagnosis'] = df['diagnosis'].map(
                    {'M':1 ,'B':0})
```

In [45]:
```
df.head()
```

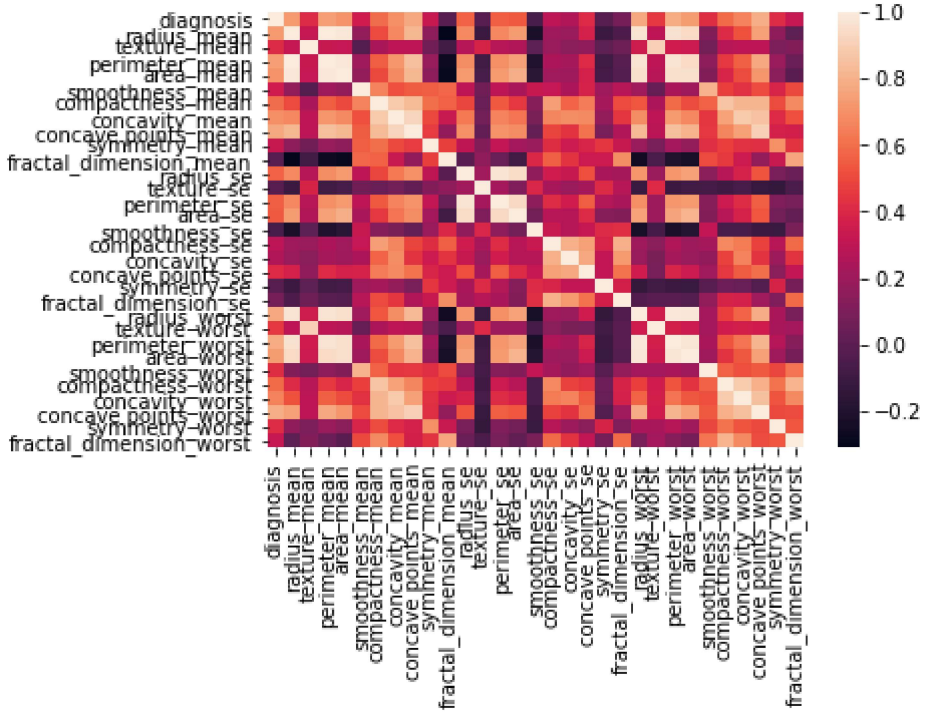
Out[45]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compac
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 [10]:
```
#Correlation test
corr = df.corr()
sns.heatmap(corr,
            xticklabels = corr.columns.values,
            yticklabels = corr.columns.values)
```

Out[10]: <AxesSubplot:>



In [11]:
```
#find how many malignant vs benign are in the data
Malignant = df[df["diagnosis"] == 1]
Benign = df[df["diagnosis"] == 0]
print(Malignant.shape)
print(Benign.shape)
```

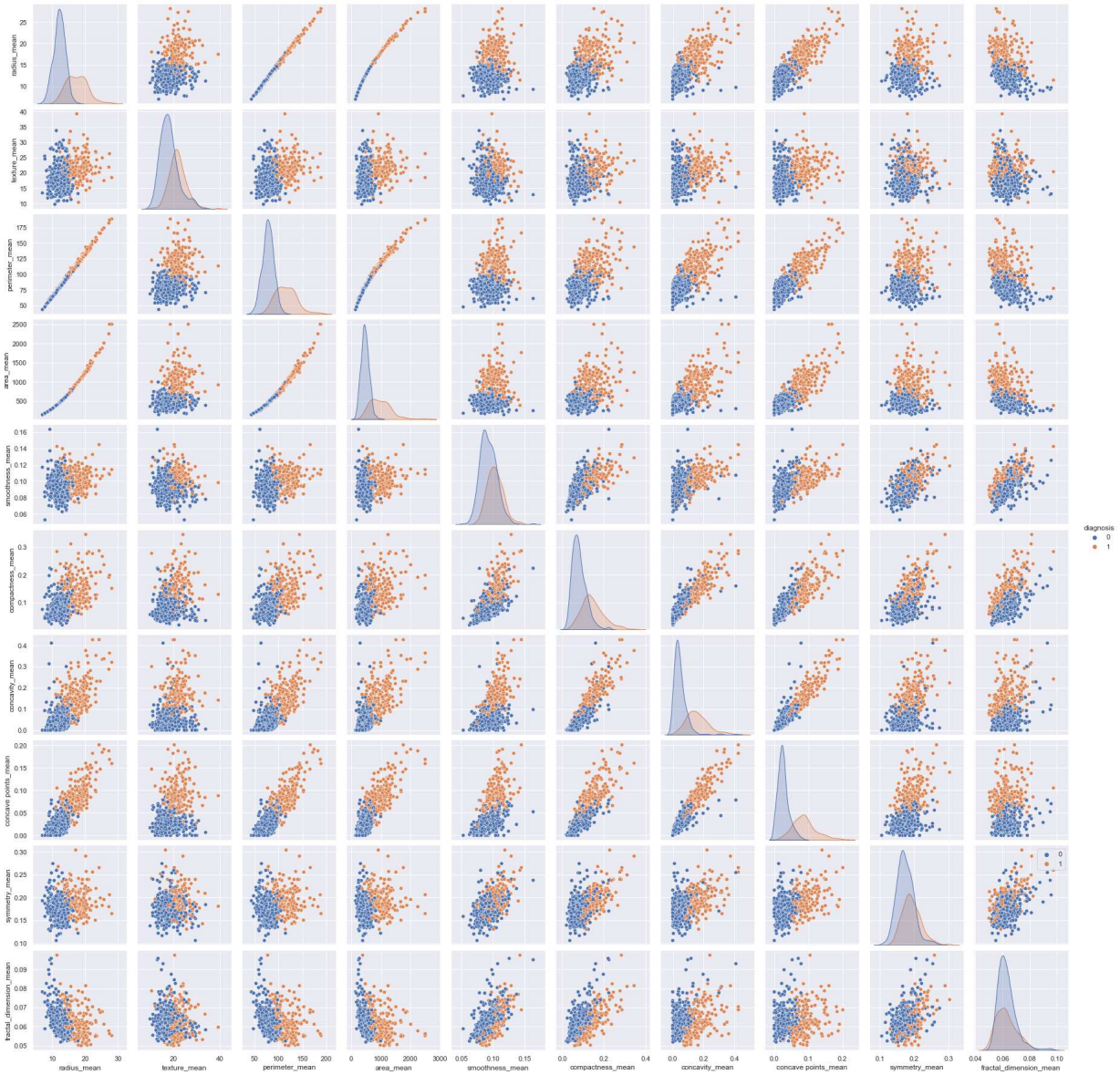
```
(212, 31)
(357, 31)
```

In [12]:
```
#find duplicates
df.duplicated().sum()
```

Out[12]: 0

In [13]:
```
#Plot the variables Malignant vs Benign using sns
sns.set()

plot = df.columns[:11]
sns.pairplot(df[plot], hue="diagnosis")
plt.legend()
plt.show()
```



In [14]:
```
# import Libraries
from sklearn.neighbors import KNeighborsClassifier
#ignore the warnings
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [15]:
```
# split the data
x = df.drop("diagnosis",axis = 1)
y = df.diagnosis
```

In [68]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33,random_state=
```

In [69]:
```
test_scores = []
train_scores = []
K=[]
for i in range(1,15):

    knn = KNeighborsClassifier(i)
    knn.fit(x_train,y_train)

    train_scores.append(knn.score(x_train,y_train))
    test_scores.append(knn.score(x_test,y_test))
    K.append(i)
```

In [72]:
```
r = { 'train':train_scores,
         'test':test_scores,
         'K':K}
results = pd.DataFrame(results,
        columns=['train','test', 'K'])
results
```

Out[72]:

	train	test	K
0	1.000000	0.909574	1
1	0.958005	0.920213	2
2	0.960630	0.930851	3
3	0.947507	0.925532	4
4	0.939633	0.930851	5
5	0.937008	0.914894	6
6	0.942257	0.930851	7
7	0.934383	0.925532	8
8	0.934383	0.941489	9
9	0.926509	0.925532	10
10	0.923885	0.941489	11
11	0.926509	0.925532	12
12	0.921260	0.946809	13
13	0.926509	0.930851	14

In [71]:
```
Model = KNeighborsClassifier(13)
Model.fit(x_train,y_train)

train_score = Model.score(x_train,y_train)
test_score = Model.score(x_test,y_test)
print( 'train_score ', train_score)
print( 'test_score ',test_score )
```

```
train_score  0.9212598425196851
test_score  0.9468085106382979
```

In [74]:
```
from sklearn.metrics import confusion_matrix, classification_report, precision_re
results = Model.predict(x_test)
print(classification_report(results,y_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.93      0.96       126
           1       0.87      0.98      0.92        62

    accuracy                           0.95       188
   macro avg       0.93      0.96      0.94       188
weighted avg       0.95      0.95      0.95       188
```

In [53]:
```
#Random Forest Classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report, precision_re
classifier=RandomForestClassifier()
classifier.fit(X_train,y_train)
results=classifier.predict(X_test)
print(classification_report(results,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97        90
           1       0.94      0.96      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.97       143
```

In [ ]:
```

```