vv7ciuuxx

November 9, 2023

#Imports

[62]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

1 DATA ONBORDING

[63]:
```
df=pd.read_csv('BREAST CANCER.csv')
```

[64]:
```
# DATAFRAME
df
```

[64]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
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	
..	…	…	…	…	…	…	
564	926424	M	21.56	22.39	142.00	1479.0	
565	926682	M	20.13	28.25	131.20	1261.0	
566	926954	M	16.60	28.08	108.30	858.1	
567	927241	M	20.60	29.33	140.10	1265.0	
568	92751	B	7.76	24.54	47.92	181.0	

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	\
0	0.11840	0.27760	0.30010	0.14710	
1	0.08474	0.07864	0.08690	0.07017	
2	0.10960	0.15990	0.19740	0.12790	
3	0.14250	0.28390	0.24140	0.10520	
4	0.10030	0.13280	0.19800	0.10430	
..	…	…	…	…	
564	0.11100	0.11590	0.24390	0.13890	
565	0.09780	0.10340	0.14400	0.09791	
566	0.08455	0.10230	0.09251	0.05302	

```
567          0.11780           0.27700         0.35140              0.15200
568          0.05263           0.04362         0.00000              0.00000

     …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0    …          17.33           184.60      2019.0           0.16220
1    …          23.41           158.80      1956.0           0.12380
2    …          25.53           152.50      1709.0           0.14440
3    …          26.50            98.87       567.7           0.20980
4    …          16.67           152.20      1575.0           0.13740
..   …            …                …          …                …
564  …          26.40           166.10      2027.0           0.14100
565  …          38.25           155.00      1731.0           0.11660
566  …          34.12           126.70      1124.0           0.11390
567  …          39.42           184.60      1821.0           0.16500
568  …          30.37            59.16       268.6           0.08996

     compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0              0.66560           0.7119                0.2654          0.4601
1              0.18660           0.2416                0.1860          0.2750
2              0.42450           0.4504                0.2430          0.3613
3              0.86630           0.6869                0.2575          0.6638
4              0.20500           0.4000                0.1625          0.2364
..                 …                …                     …               …
564            0.21130           0.4107                0.2216          0.2060
565            0.19220           0.3215                0.1628          0.2572
566            0.30940           0.3403                0.1418          0.2218
567            0.86810           0.9387                0.2650          0.4087
568            0.06444           0.0000                0.0000          0.2871

     fractal_dimension_worst  Unnamed: 32
0                    0.11890          NaN
1                    0.08902          NaN
2                    0.08758          NaN
3                    0.17300          NaN
4                    0.07678          NaN
..                       …            …
564                  0.07115          NaN
565                  0.06637          NaN
566                  0.07820          NaN
567                  0.12400          NaN
568                  0.07039          NaN

[569 rows x 33 columns]
```

[65]:
```
#EDA
```

[66]:
```
df_copy=df.copy()
```

[67]:
```
df.shape
```

[67]:
```
(569, 33)
```

[68]:
```
df.columns
```

[68]:
```
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', 'Unnamed: 32'],
      dtype='object')
```

[69]:
```
df.head()
```

[69]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
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	

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	\
0	0.11840	0.27760	0.3001	0.14710	
1	0.08474	0.07864	0.0869	0.07017	
2	0.10960	0.15990	0.1974	0.12790	
3	0.14250	0.28390	0.2414	0.10520	
4	0.10030	0.13280	0.1980	0.10430	

	…	texture_worst	perimeter_worst	area_worst	smoothness_worst	\
0	…	17.33	184.60	2019.0	0.1622	
1	…	23.41	158.80	1956.0	0.1238	
2	…	25.53	152.50	1709.0	0.1444	
3	…	26.50	98.87	567.7	0.2098	
4	…	16.67	152.20	1575.0	0.1374	

	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	\
0	0.6656	0.7119	0.2654	0.4601	
1	0.1866	0.2416	0.1860	0.2750	
2	0.4245	0.4504	0.2430	0.3613	
3	0.8663	0.6869	0.2575	0.6638	
4	0.2050	0.4000	0.1625	0.2364	

```
   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

[70]:
```
df.tail()
```

[70]:
```
          id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
564  926424         M        21.56         22.39          142.00     1479.0
565  926682         M        20.13         28.25          131.20     1261.0
566  926954         M        16.60         28.08          108.30      858.1
567  927241         M        20.60         29.33          140.10     1265.0
568   92751         B         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
564          0.11100           0.11590         0.24390              0.13890
565          0.09780           0.10340         0.14400              0.09791
566          0.08455           0.10230         0.09251              0.05302
567          0.11780           0.27700         0.35140              0.15200
568          0.05263           0.04362         0.00000              0.00000

     …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
564  …          26.40           166.10      2027.0           0.14100
565  …          38.25           155.00      1731.0           0.11660
566  …          34.12           126.70      1124.0           0.11390
567  …          39.42           184.60      1821.0           0.16500
568  …          30.37            59.16       268.6           0.08996

     compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
564            0.21130           0.4107                0.2216          0.2060
565            0.19220           0.3215                0.1628          0.2572
566            0.30940           0.3403                0.1418          0.2218
567            0.86810           0.9387                0.2650          0.4087
568            0.06444           0.0000                0.0000          0.2871

     fractal_dimension_worst  Unnamed: 32
564                  0.07115          NaN
565                  0.06637          NaN
566                  0.07820          NaN
567                  0.12400          NaN
568                  0.07039          NaN

[5 rows x 33 columns]
```

[71]:
```
df.iloc[100:201]
```

[71]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
100	862717	M	13.610	24.98	88.05	582.7	
101	862722	B	6.981	13.43	43.79	143.5	
102	862965	B	12.180	20.52	77.22	458.7	
103	862980	B	9.876	19.40	63.95	298.3	
104	862989	B	10.490	19.29	67.41	336.1	
..	…	…	…	…	…	…	
196	875938	M	13.770	22.29	90.63	588.9	
197	877159	M	18.080	21.84	117.40	1024.0	
198	877486	M	19.180	22.49	127.50	1148.0	
199	877500	M	14.450	20.22	94.49	642.7	
200	877501	B	12.230	19.56	78.54	461.0	

	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	\
100	0.09488	0.08511	0.08625	0.04489	
101	0.11700	0.07568	0.00000	0.00000	
102	0.08013	0.04038	0.02383	0.01770	
103	0.10050	0.09697	0.06154	0.03029	
104	0.09989	0.08578	0.02995	0.01201	
..	…	…	…	…	
196	0.12000	0.12670	0.13850	0.06526	
197	0.07371	0.08642	0.11030	0.05778	
198	0.08523	0.14280	0.11140	0.06772	
199	0.09872	0.12060	0.11800	0.05980	
200	0.09586	0.08087	0.04187	0.04107	

	…	texture_worst	perimeter_worst	area_worst	smoothness_worst	\
100	…	35.27	108.60	906.5	0.12650	
101	…	19.54	50.41	185.2	0.15840	
102	…	32.84	84.58	547.8	0.11230	
103	…	26.83	72.22	361.2	0.15590	
104	…	23.31	74.22	402.8	0.12190	
..	…	…	…	…	…	
196	…	34.01	111.60	806.9	0.17370	
197	…	24.70	129.10	1228.0	0.08822	
198	…	32.06	166.40	1688.0	0.13220	
199	…	30.12	117.90	1044.0	0.15520	
200	…	28.36	92.15	638.4	0.14290	

	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	\
100	0.19430	0.31690	0.11840	0.2651	
101	0.12020	0.00000	0.00000	0.2932	
102	0.08862	0.11450	0.07431	0.2694	
103	0.23020	0.26440	0.09749	0.2622	
104	0.14860	0.07987	0.03203	0.2826	

```
..                 …               …                   …              …
196            0.31220         0.38090             0.16730         0.3080
197            0.19630         0.25350             0.09181         0.2369
198            0.56010         0.38650             0.17080         0.3193
199            0.40560         0.49670             0.18380         0.4753
200            0.20420         0.13770             0.10800         0.2668

     fractal_dimension_worst  Unnamed: 32
100                  0.07397          NaN
101                  0.09382          NaN
102                  0.06878          NaN
103                  0.08490          NaN
104                  0.07552          NaN
..                       …            …
196                  0.09333          NaN
197                  0.06558          NaN
198                  0.09221          NaN
199                  0.10130          NaN
200                  0.08174          NaN

[101 rows x 33 columns]
```

[72]:
```
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
```

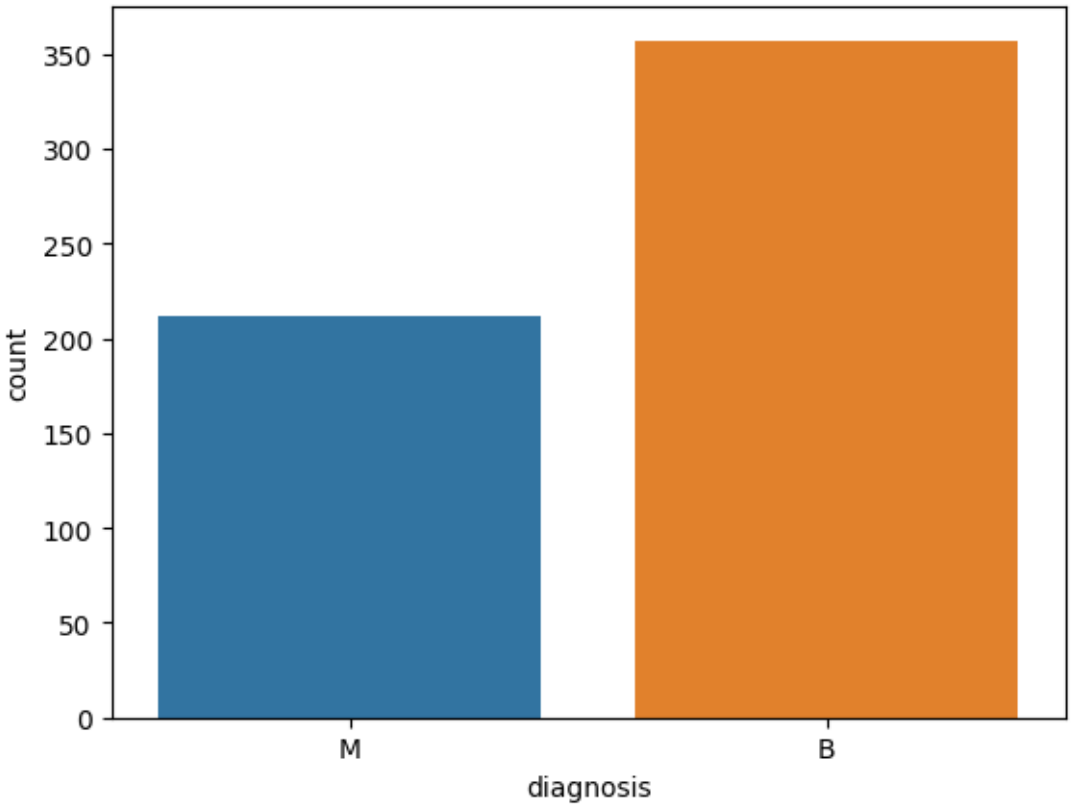
[73]:
```
df.describe().T.style.background_gradient(sns.color_palette("light:#5A9",␣
↪as_cmap=True))
```

[73]:
```
<pandas.io.formats.style.Styler at 0x7e3368dc5030>
```

#visualization

[74]:
```
sns.countplot(x='diagnosis',data=df)
```

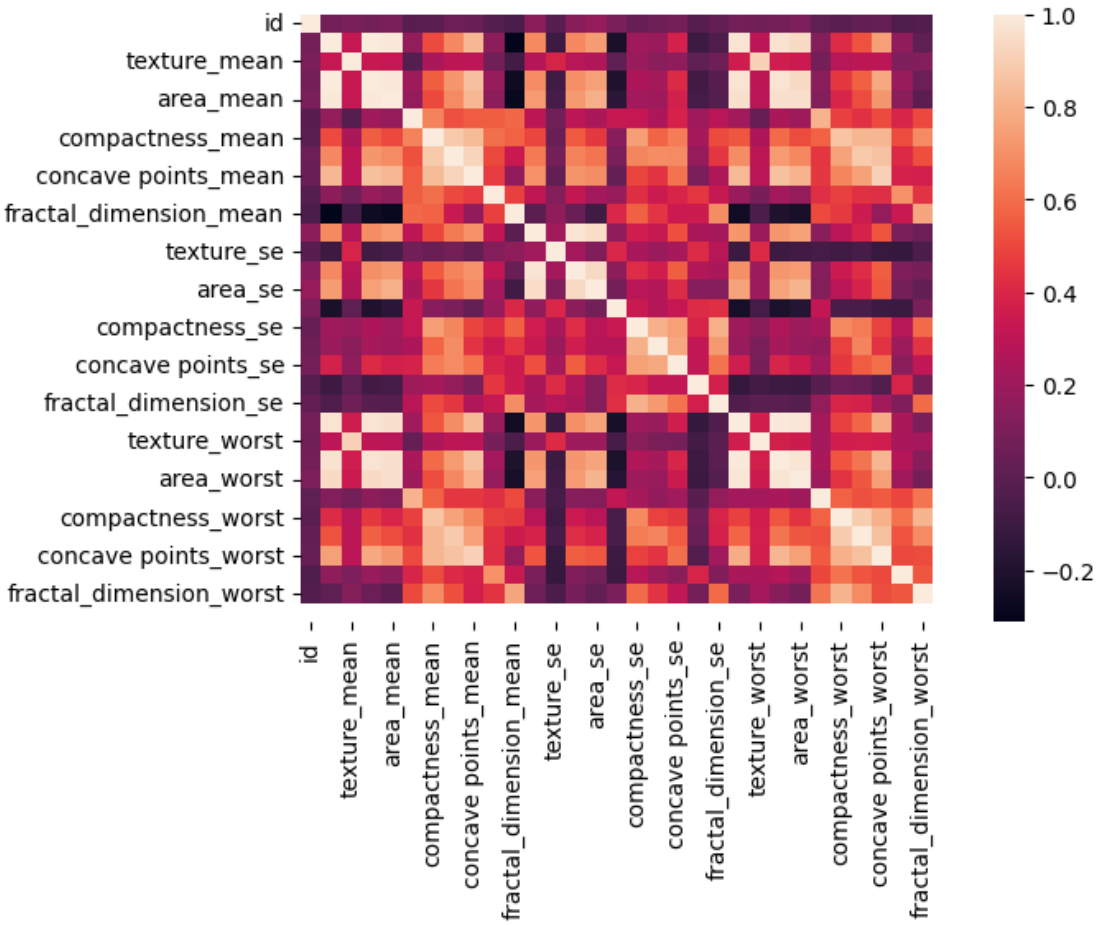
[74]:
```
<Axes: xlabel='diagnosis', ylabel='count'>
```



[75]:
```
sns.heatmap(df.corr())
```

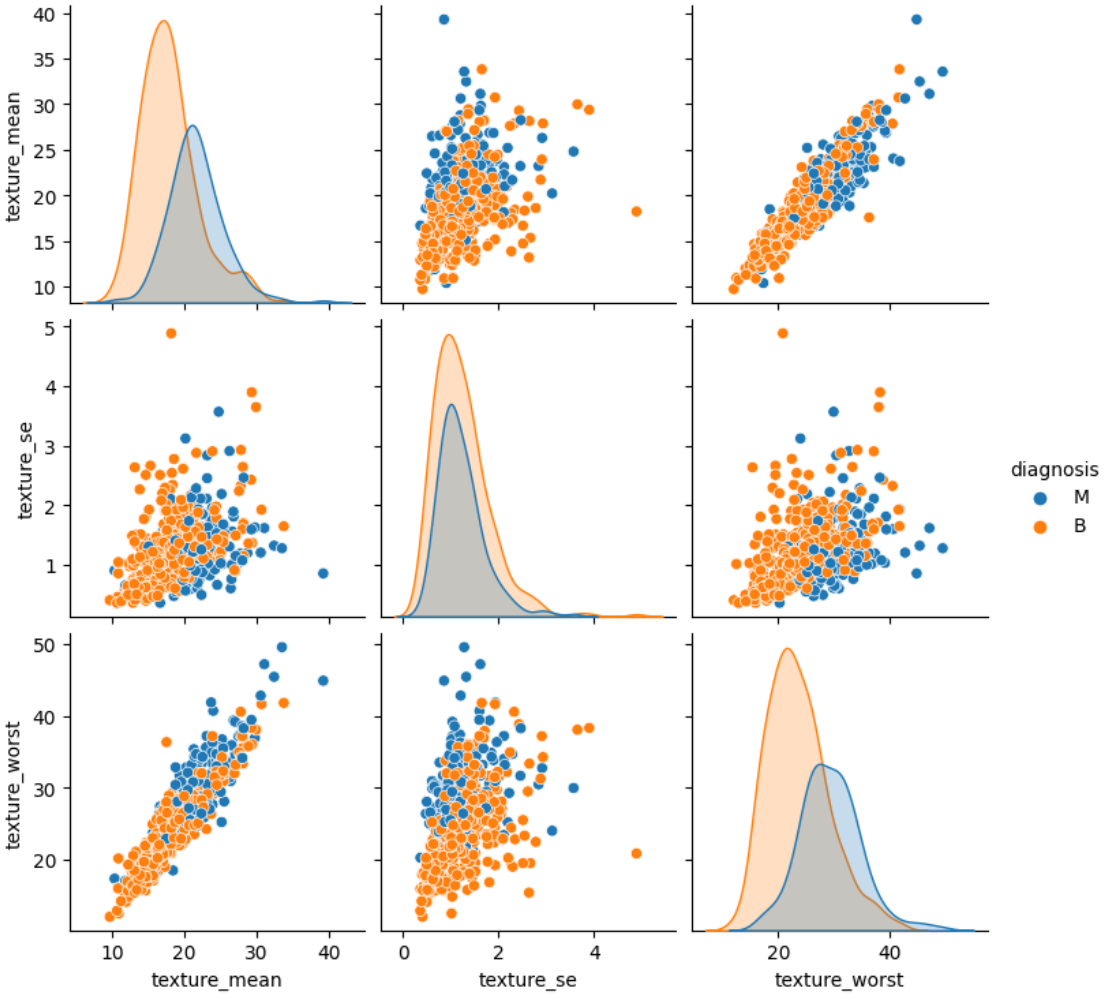
```
<ipython-input-75-aa4f4450a243>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df.corr())
```

[75]: <Axes: >



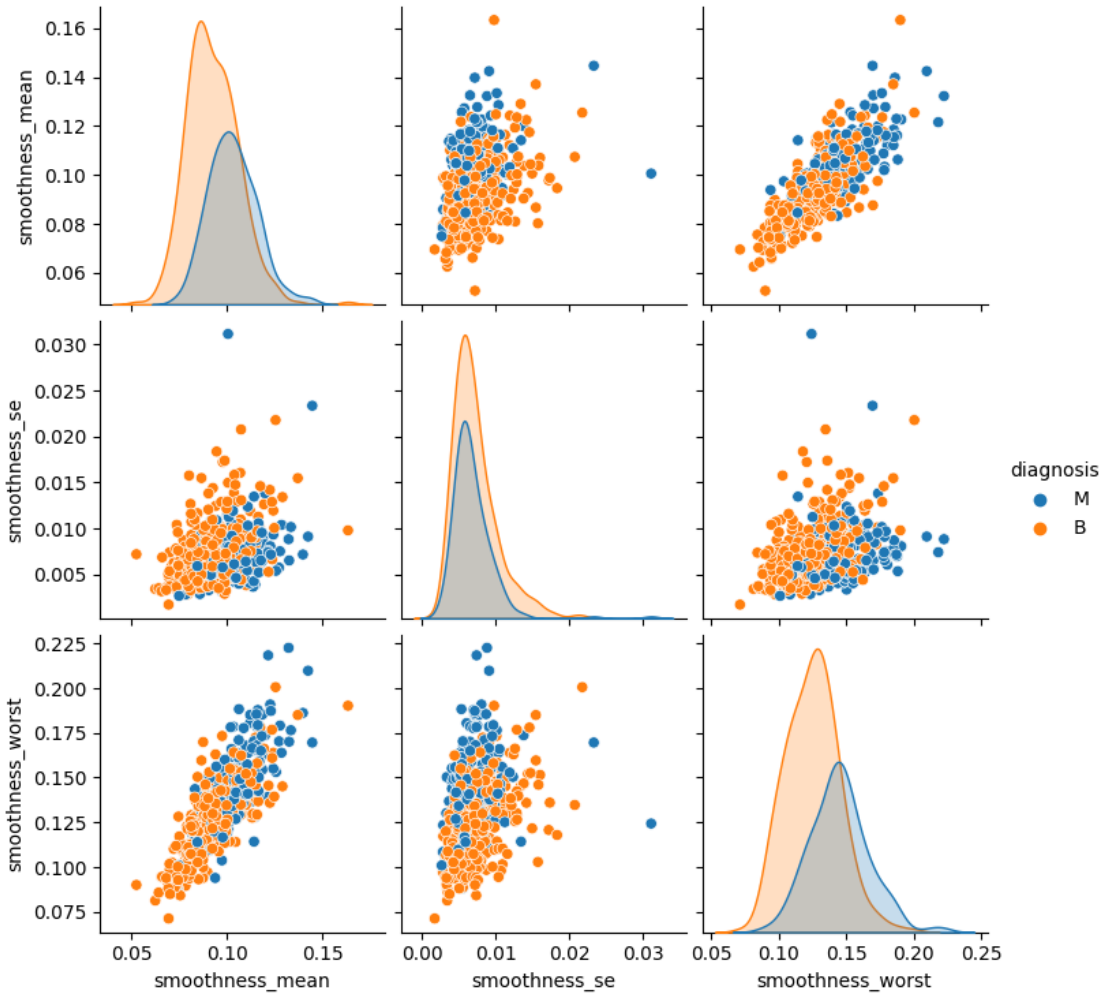
[76]:
```
texture=df[['texture_mean','texture_se','texture_worst','diagnosis']]
sns.pairplot(texture,hue='diagnosis')
```

[76]: <seaborn.axisgrid.PairGrid at 0x7e3368bce9e0>



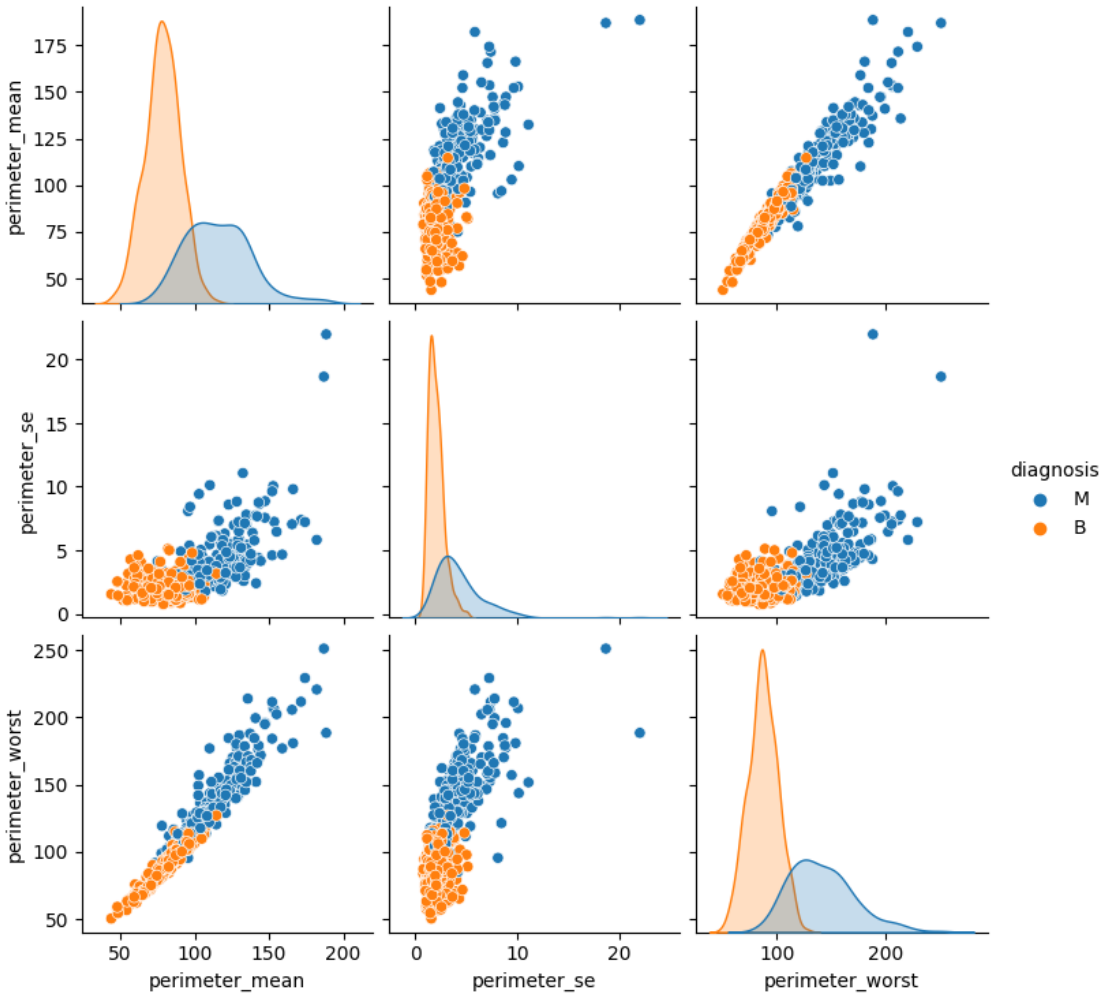
[77]:
```
smoothness=df[['smoothness_mean','smoothness_se','smoothness_worst','diagnosis']]
sns.pairplot(smoothness,hue='diagnosis')
```

[77]: <seaborn.axisgrid.PairGrid at 0x7e3368bce8c0>



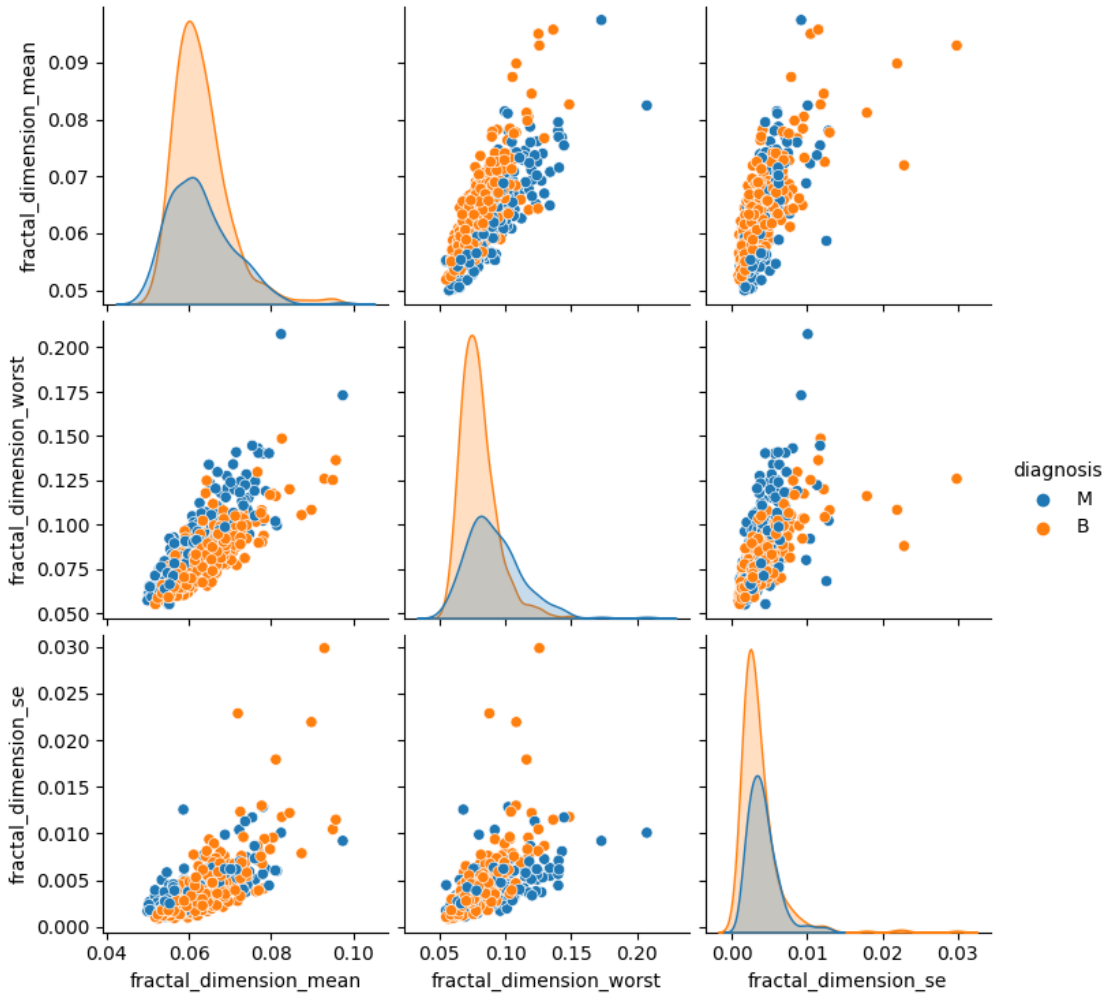
[78]:
```
perimeter=df[['perimeter_mean','perimeter_se','perimeter_worst','diagnosis']]
sns.pairplot(perimeter,hue='diagnosis')
```

[78]: <seaborn.axisgrid.PairGrid at 0x7e3368347820>



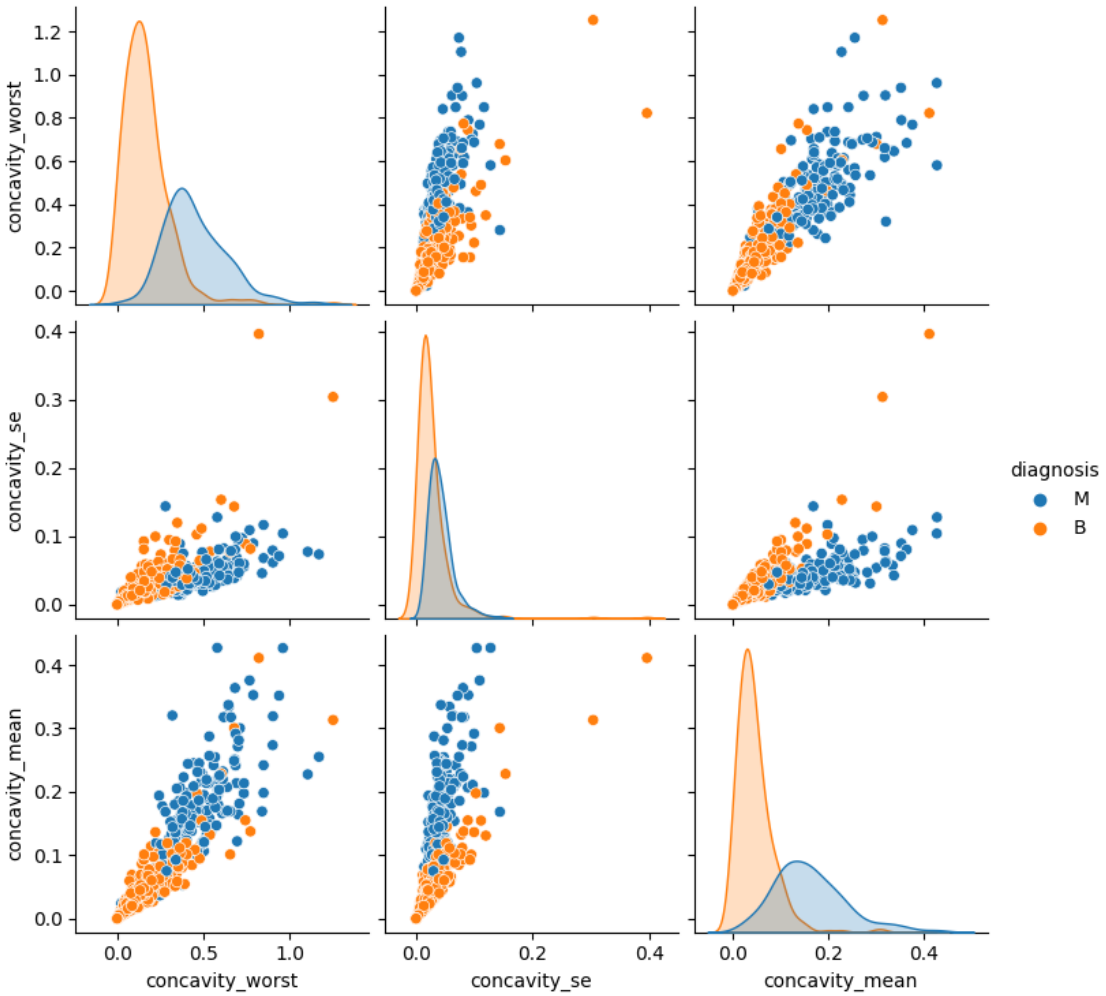
[79]:
```
fractal=df[['fractal_dimension_mean','fractal_dimension_worst','fractal_dimension_se','diagnos
sns.pairplot(fractal,hue='diagnosis')
```

[79]: <seaborn.axisgrid.PairGrid at 0x7e3368347460>



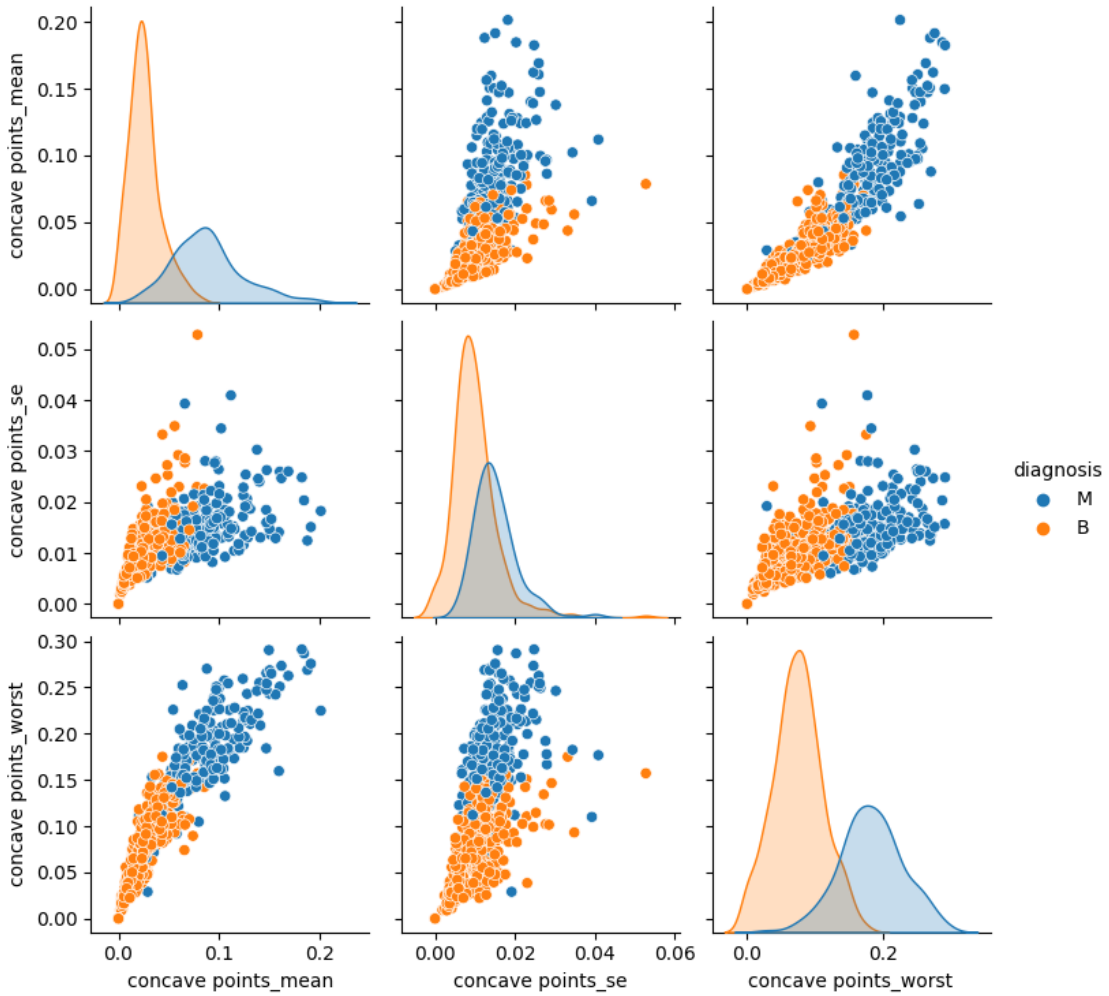
[80]:
```
concavity=df[['concavity_worst','concavity_se','concavity_mean','diagnosis']]
sns.pairplot(concavity,hue='diagnosis')
```

[80]: <seaborn.axisgrid.PairGrid at 0x7e3363b6fd90>



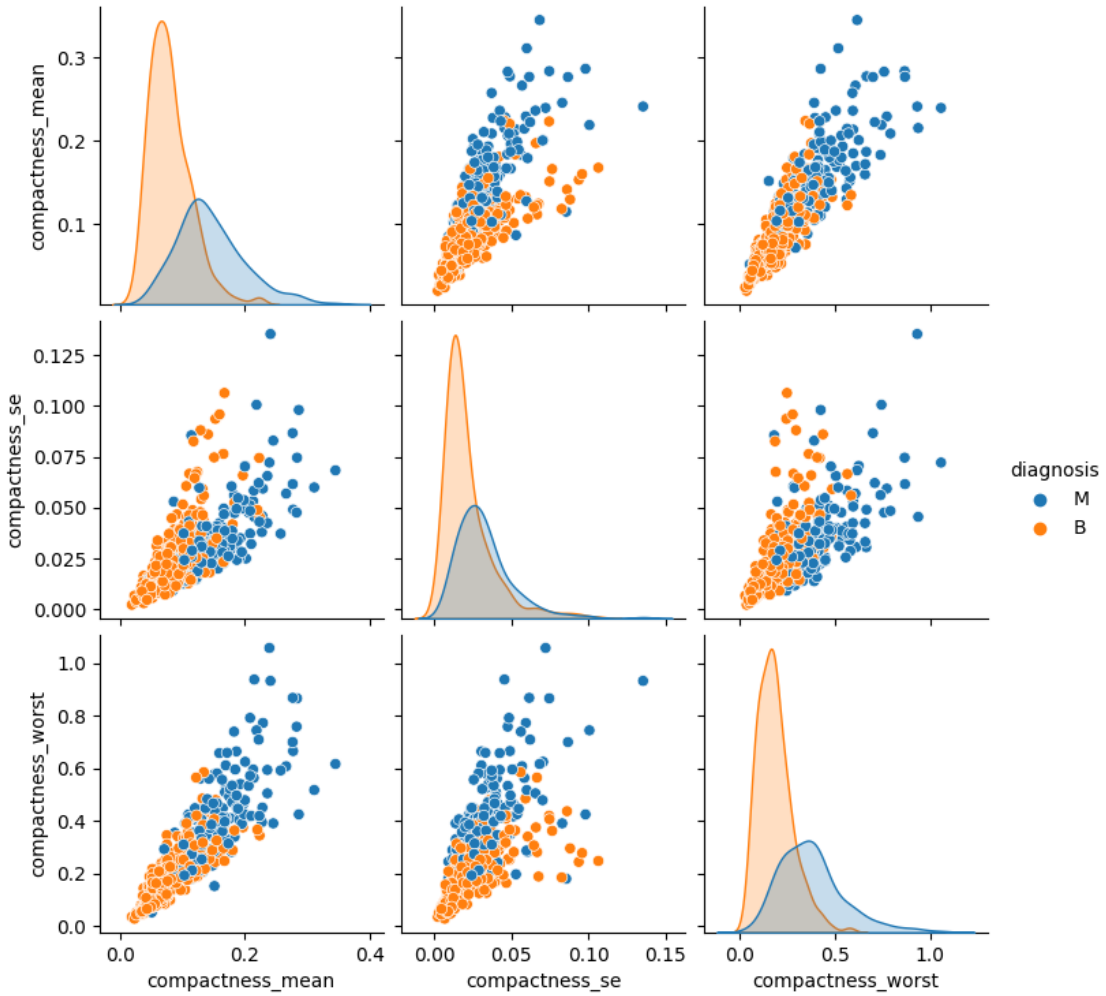
[81]:
```
concave=df[['concave points_mean','concave points_se','concave␣
↪points_worst','diagnosis']]
sns.pairplot(concave,hue='diagnosis')
```

[81]: <seaborn.axisgrid.PairGrid at 0x7e3363843430>



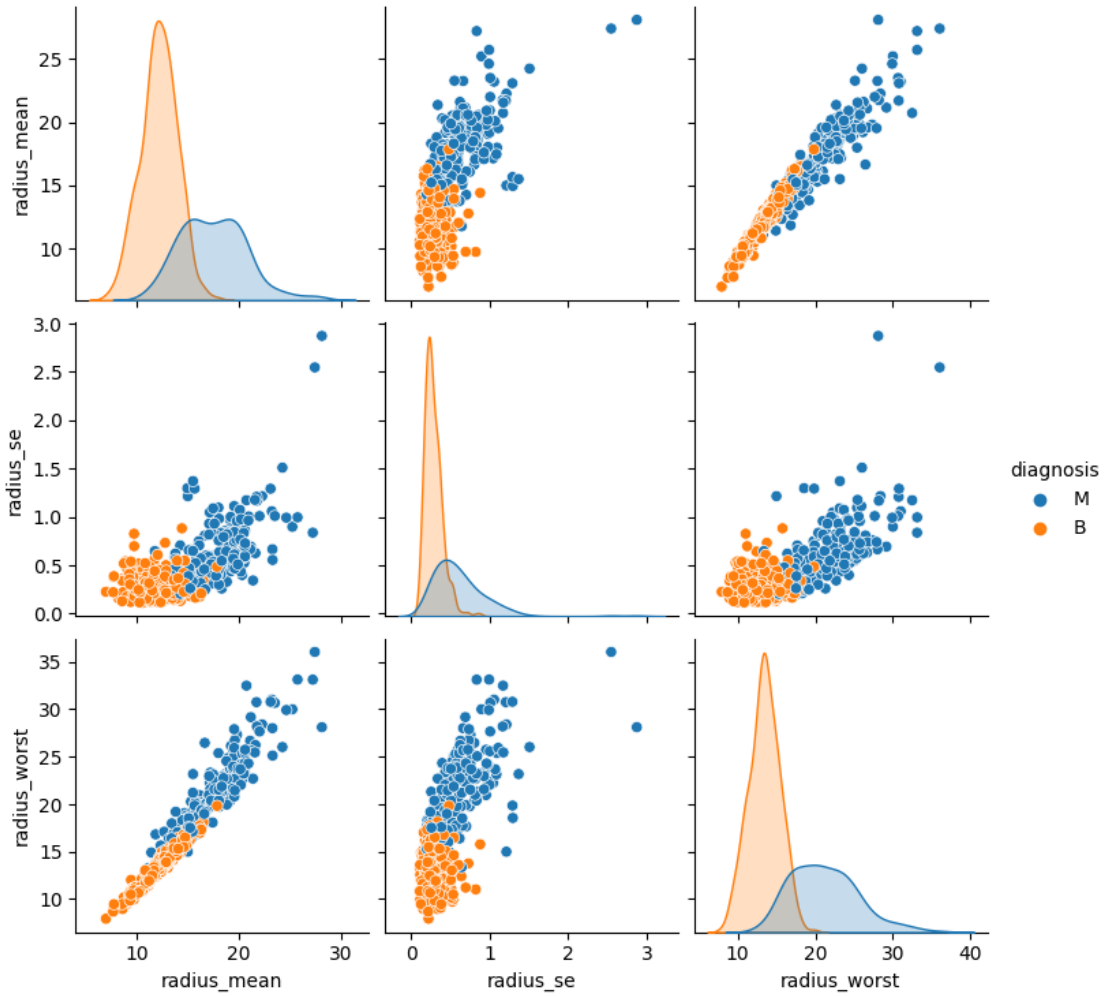
[82]:
```
compactness=df[['compactness_mean','compactness_se','compactness_worst','diagnosis']]
sns.pairplot(compactness,hue='diagnosis')
```

[82]: <seaborn.axisgrid.PairGrid at 0x7e3362f663e0>



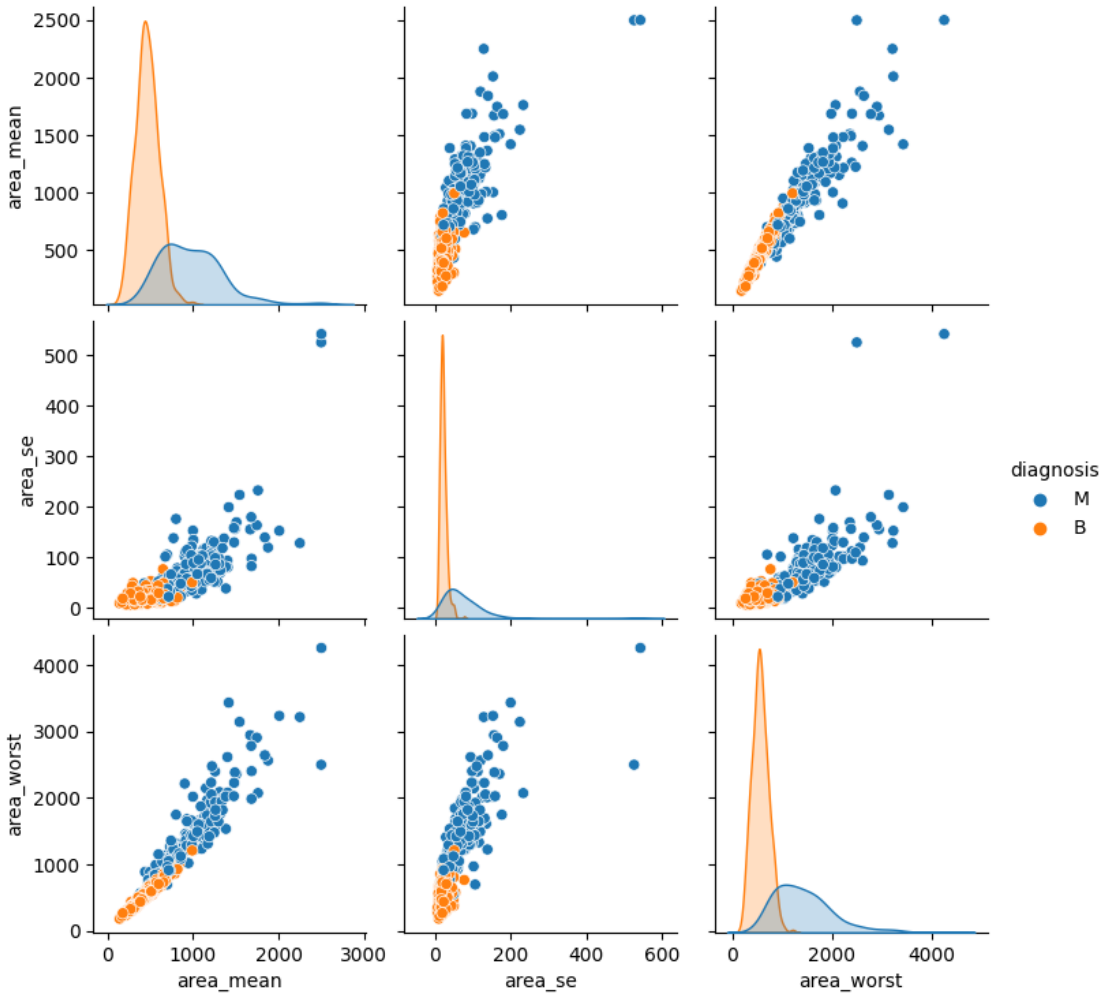
[83]:
```
radius=df[['radius_mean','radius_se','radius_worst','diagnosis']]
sns.pairplot(radius,hue='diagnosis')
```

[83]: <seaborn.axisgrid.PairGrid at 0x7e3362de29b0>



[84]:
```
area=df[['area_mean','area_se','area_worst','diagnosis']]
sns.pairplot(area,hue='diagnosis')
```

[84]: <seaborn.axisgrid.PairGrid at 0x7e336271cac0>



#DATA CLEANING

[85]:
```
df.isnull().sum()
```

[85]:
```
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
```

[86]:
```
df.drop(columns='Unnamed: 32',inplace=True)
```

[87]:
```
df.drop(columns='id',inplace=True)
```

[88]:
```
df.duplicated().sum()
```

[88]:
```
0
```

[89]:
```
df.columns
```

[89]:
```
Index(['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')
```

2 X and y split

[90]:
```
df.columns
```

[90]:
```
Index(['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')
```

[91]:
```
y=df['diagnosis']
x=df.drop('diagnosis',axis=1)
```

3 Labuel encoding

[92]:
```
from sklearn.preprocessing import LabelEncoder
lbe=LabelEncoder()
y=lbe.fit_transform(y)
```

4 TRAIN AND TEST

[93]:
```
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,random_state=124)
```

5 KNeighborsClassifier(KNN)

[94]:
```
from sklearn.neighbors import KNeighborsClassifier
k=3
knn=KNeighborsClassifier(n_neighbors=k)
# train
knn.fit(x_train,y_train)
# predict
y_pred_knn=knn.predict(x_test)
```

[95]:
```
# Evaluation KNN
```

[96]:
```
from sklearn.metrics import␣
↪confusion_matrix,accuracy_score,classification_report
```

```
cm_knn=confusion_matrix(y_test,y_pred_knn)
acc_knn=accuracy_score(y_test,y_pred_knn)
cr=classification_report(y_test,y_pred_knn)
print('confusion_matrix:',cm_knn)
print('accuracy_score:',acc_knn)
print('classification_report',cr)
# ERROR RATE AND CLASSIFICATION REPORT
print("Misclassification error rate:",round(np.mean(y_pred_knn!=y_test),3))
print(classification_report(y_test,y_pred_knn))
```

```
confusion_matrix: [[98  4]
 [10 59]]
accuracy_score: 0.9181286549707602
classification_report               precision    recall  f1-score   support

           0       0.91      0.96      0.93       102
           1       0.94      0.86      0.89        69

    accuracy                           0.92       171
   macro avg       0.92      0.91      0.91       171
weighted avg       0.92      0.92      0.92       171

Misclassification error rate: 0.082
              precision    recall  f1-score   support

           0       0.91      0.96      0.93       102
           1       0.94      0.86      0.89        69

    accuracy                           0.92       171
   macro avg       0.92      0.91      0.91       171
weighted avg       0.92      0.92      0.92       171
```

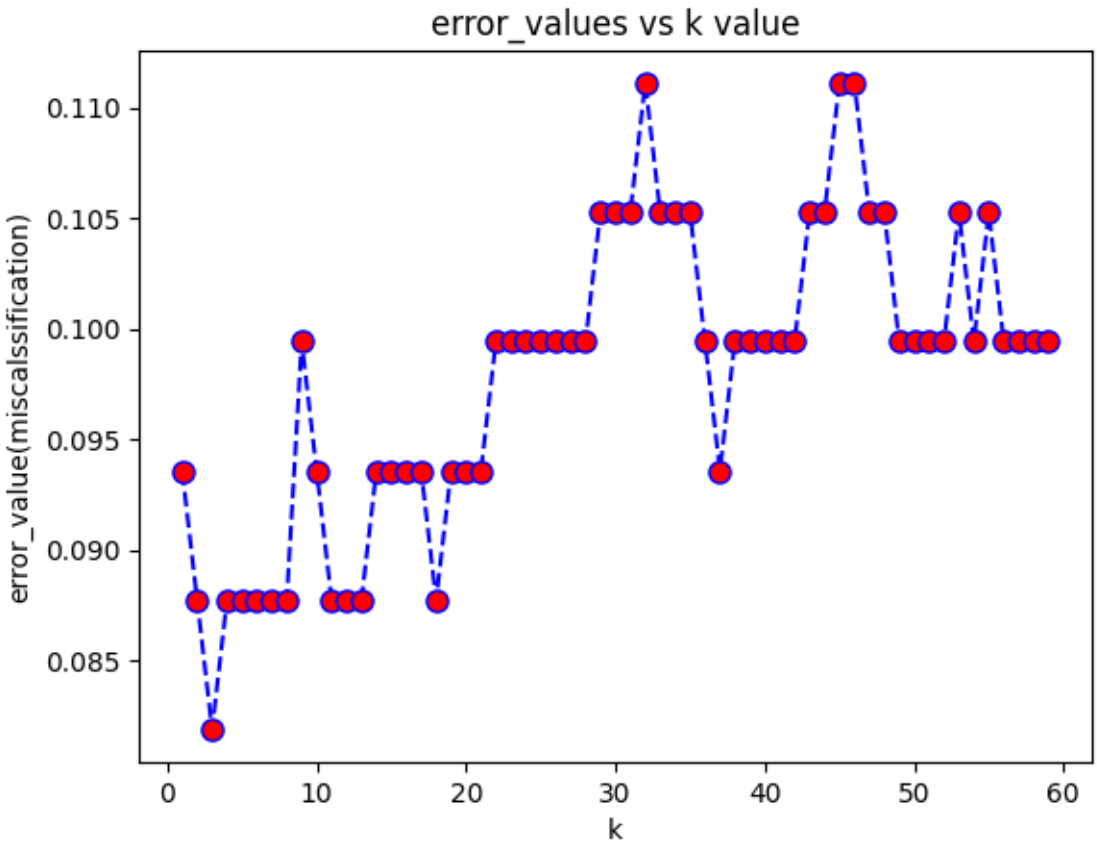
##Choosing 'k' by elbow **method**

[97]:
```
# ERROR VALUE
error_value=[]
for i in range(1,60):
  knn_i=KNeighborsClassifier(n_neighbors=i)
  knn_i.fit(x_train,y_train)
  y_pred_knn_i=knn_i.predict(x_test)
  error_value.append(np.mean(y_pred_knn_i!=y_test))
```

[98]:
```
# PLOT FOR ERROR VALUE
plt.
 ↪plot(range(1,60),error_value,color='blue',linestyle='dashed',marker='o',markerfacecolor='re
 ↪markersize=8)
plt.title('error_values vs k value')
```

```
plt.xlabel('k')
plt.ylabel("error_value(miscalssification)")
```

[98]: `Text(0, 0.5, 'error_value(miscalssification)')`



6 Support Vector Regression (SVR)

[99]:
```
from sklearn.svm import SVC
# create the SVM classifier
svm = SVC()
# Train the classifier on the training data
svm.fit(x_train, y_train)
# Make predictions on the testing data
y_pred_svm= svm.predict(x_test)
```

[100]:
```
# Evaluate the model
from sklearn.metrics import accuracy_score,confusion_matrix
accuracy_svm= accuracy_score(y_test, y_pred_svm)*100
print(f"Accuracy: {accuracy_svm}")
```

```
cm_svm= confusion_matrix(y_test,y_pred_svm)
print('Confusion',cm_svm)
# ERROR RATE AND CLASSIFICATION REPORT
print("Misclassification error rate:",round(np.mean(y_pred_svm!=y_test),3))
print(classification_report(y_test,y_pred_svm))
```

```
Accuracy: 90.05847953216374
Confusion [[101   1]
 [ 16  53]]
Misclassification error rate: 0.099
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       102
           1       0.98      0.77      0.86        69

    accuracy                           0.90       171
   macro avg       0.92      0.88      0.89       171
weighted avg       0.91      0.90      0.90       171
```

#Logistic Regression

[101]:
```
from sklearn.linear_model import LogisticRegression
#Logistic Regression model
lg=LogisticRegression()
# Train the model on the training data
lg.fit(x_train,y_train)
# Make predictions on the testing data
y_pred_lg=lg.predict(x_test)
# Evaluate the model
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[102]:
```
# Evaluate the model
from sklearn.metrics import␣
 ↪confusion_matrix,accuracy_score,f1_score,precision_score,recall_score,balanced_accuracy_sco
cm_lg = confusion_matrix(y_test,y_pred_lg)
print('Confusion',cm_lg)
accuracy_lg= accuracy_score(y_test,y_pred_lg)*100
```

```
print('Accuracy',accuracy_lg)
# ERROR RATE AND CLASSIFICATION REPORT
print("Misclassification error rate:",round(np.mean(y_pred_lg!=y_test),3))
print(classification_report(y_test,y_pred_lg))
```

```
Confusion [[96  6]
 [ 5 64]]
Accuracy 93.56725146198829
Misclassification error rate: 0.064
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       102
           1       0.91      0.93      0.92        69

    accuracy                           0.94       171
   macro avg       0.93      0.93      0.93       171
weighted avg       0.94      0.94      0.94       171
```

#Naive Bayes

[103]:
```
from sklearn.naive_bayes import MultinomialNB
# Multinomial Naive Bayes classifier
mnb=MultinomialNB()
# Train the model on the training data
mnb.fit(x_train,y_train)
# Make predictions on the testing data
y_pred_nb=mnb.predict(x_test)
# Evaluate the model
from sklearn.metrics import confusion_matrix,accuracy_score
cm_nb= confusion_matrix(y_test,y_pred_nb)
print('Confusion',cm_nb)
accuracy_nb= accuracy_score(y_test,y_pred_nb)*100
print('Accuracy',accuracy_nb)
```

```
Confusion [[97  5]
 [19 50]]
Accuracy 85.96491228070175
```

[104]:
```
# ERROR RATE AND CLASSIFICATION REPORT
print("Misclassification error rate:",round(np.mean(y_pred_nb!=y_test),3))
print(classification_report(y_test,y_pred_nb))
```

```
Misclassification error rate: 0.14
              precision    recall  f1-score   support

           0       0.84      0.95      0.89       102
           1       0.91      0.72      0.81        69
```

```
    accuracy                           0.86       171
   macro avg       0.87      0.84      0.85       171
weighted avg       0.87      0.86      0.86       171
```

7 Decision tree

[105]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
# train
dt.fit(x_train,y_train)
# predit
y_pred_dt=dt.predict(x_test)
```

[106]:
```
from sklearn.metrics import␣
 ↪confusion_matrix,accuracy_score,classification_report
cm_dt= confusion_matrix(y_test,y_pred_dt)
print('Confusion',cm_dt)
accuracy_dt= accuracy_score(y_test,y_pred_dt)*100
print('Accuracy',accuracy_dt)
# ERROR RATE AND CLASSIFICATION REPORT
print("Misclassification error rate:",round(np.mean(y_pred_dt!=y_test),3))
print(classification_report(y_test,y_pred_dt))
```

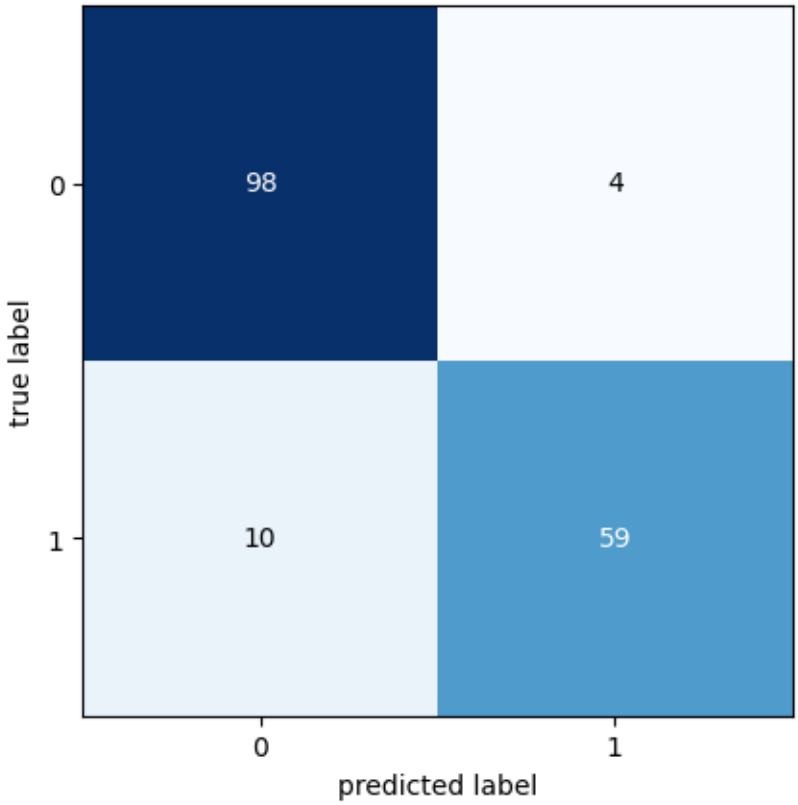
```
Confusion [[98  4]
 [ 9 60]]
Accuracy 92.39766081871345
Misclassification error rate: 0.076
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       102
           1       0.94      0.87      0.90        69

    accuracy                           0.92       171
   macro avg       0.93      0.92      0.92       171
weighted avg       0.92      0.92      0.92       171
```

8 Evaluation visualization

8.1 KNN

[107]:
```
from mlxtend.plotting import plot_confusion_matrix
plot_confusion_matrix(cm_knn)
```

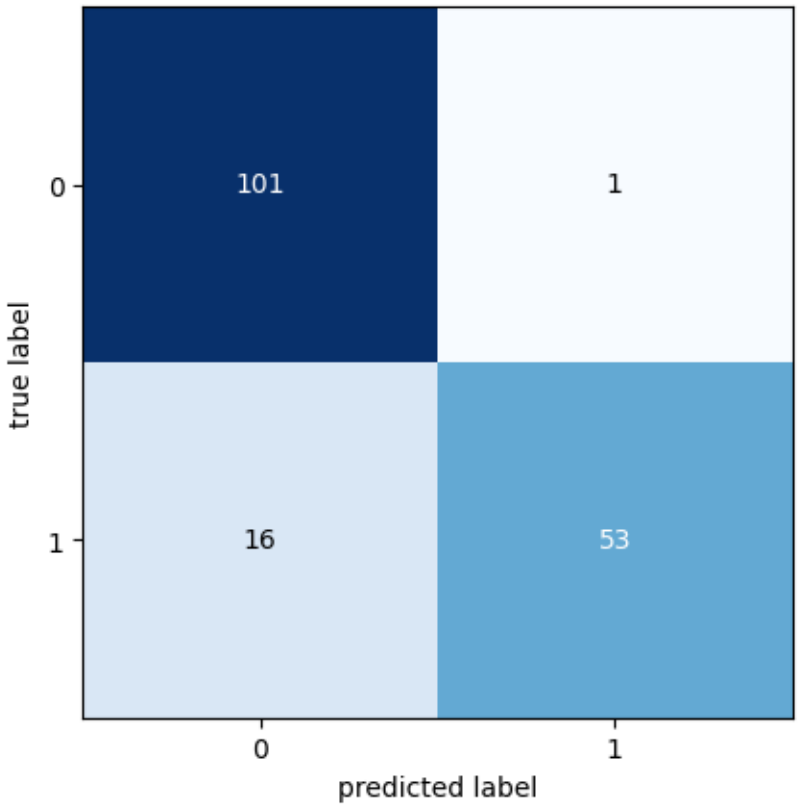
[107]: (<Figure size 640x480 with 1 Axes>,
<Axes: xlabel='predicted label', ylabel='true label'>)



8.2 SVM

[108]:
```
# confusion matrix plot
from mlxtend.plotting import plot_confusion_matrix
plot_confusion_matrix(cm_svm)
```

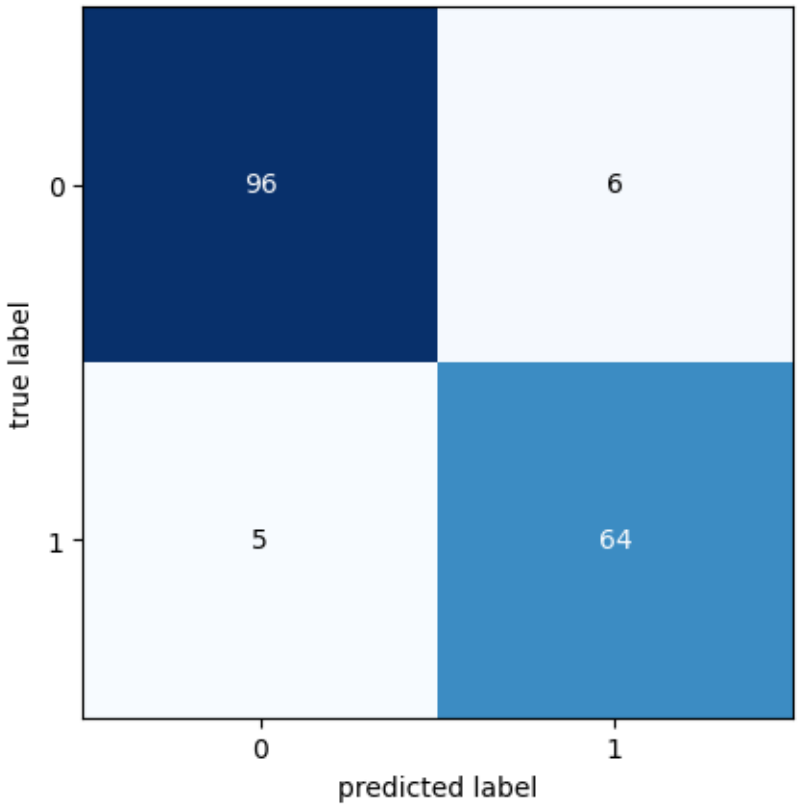
[108]: (<Figure size 640x480 with 1 Axes>,
<Axes: xlabel='predicted label', ylabel='true label'>)



##Logistic Regression

[109]:
```
# confusion matrix plot
from mlxtend.plotting import plot_confusion_matrix
plot_confusion_matrix(cm_lg)
```

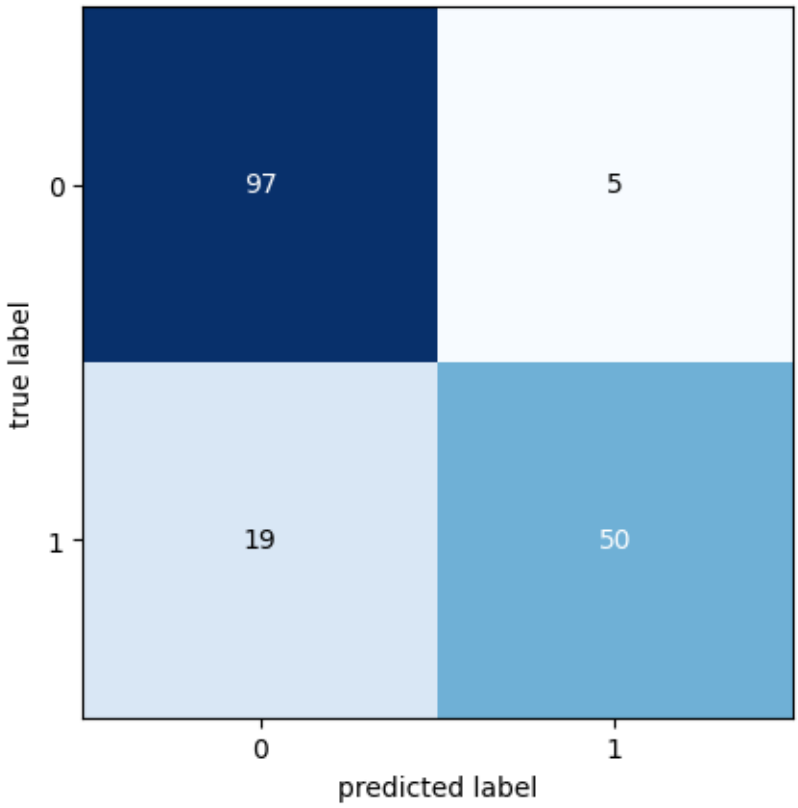
[109]:
```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='predicted label', ylabel='true label'>)
```



##Naive Bayes

[110]:
```
# confusion matrix plot
from mlxtend.plotting import plot_confusion_matrix
plot_confusion_matrix(cm_nb)
```

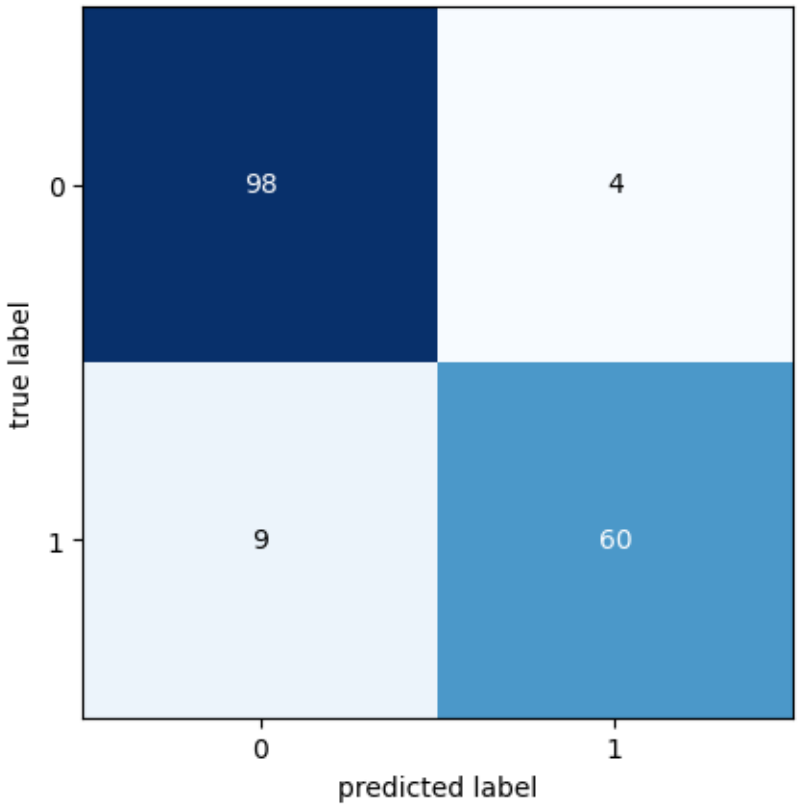
[110]:
```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='predicted label', ylabel='true label'>)
```



##Decision Tree

[111]:
```
# confusion matrix plot
from mlxtend.plotting import plot_confusion_matrix
plot_confusion_matrix(cm_dt)
```

[111]:
```
(<Figure size 640x480 with 1 Axes>,
 <Axes: xlabel='predicted label', ylabel='true label'>)
```



8.3 Model Evaluation data frame (using function)

[112]:
```
def train_evaluation_model(model,x_train,y_train,x_test,y_test):
  model.fit(x_train,y_train)
  pred=model.predict(x_test)
  accuracy = accuracy_score(y_test, pred)
  f1 = f1_score(y_test, pred)
  precision = precision_score(y_test, pred)
  recall = recall_score(y_test, pred)
  balanced_accuracy = balanced_accuracy_score(y_test, pred)
  error_val=round(np.mean(pred!=y_test),3)
  ev_df=pd.DataFrame([[accuracy, f1, precision, recall,␣
 ↪balanced_accuracy,error_val]],columns=('accuracy','f1','precision','recall','balanced_accur
  return ev_df
```

[113]:
```
result=train_evaluation_model(lg,x_train,y_train,x_test,y_test)
dt_results = train_evaluation_model(dt,x_train, y_train, x_test, y_test)
knn_results = train_evaluation_model(knn,x_train, y_train, x_test, y_test)
nb_results = train_evaluation_model(mnb,x_train, y_train, x_test, y_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
```

```
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[114]:
```
result.index=['Logistic Regression']
dt_results.index=['decision tree']
result = result.append(dt_results)
knn_results.index=['K-Nearest Neighbors']
nb_results.index=['Naive Bayes']
result = result.append(knn_results)
result = result.append(nb_results)
```

```
<ipython-input-114-97d3832c0520>:3: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
  result = result.append(dt_results)
<ipython-input-114-97d3832c0520>:6: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
  result = result.append(knn_results)
<ipython-input-114-97d3832c0520>:7: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use
pandas.concat instead.
  result = result.append(nb_results)
```

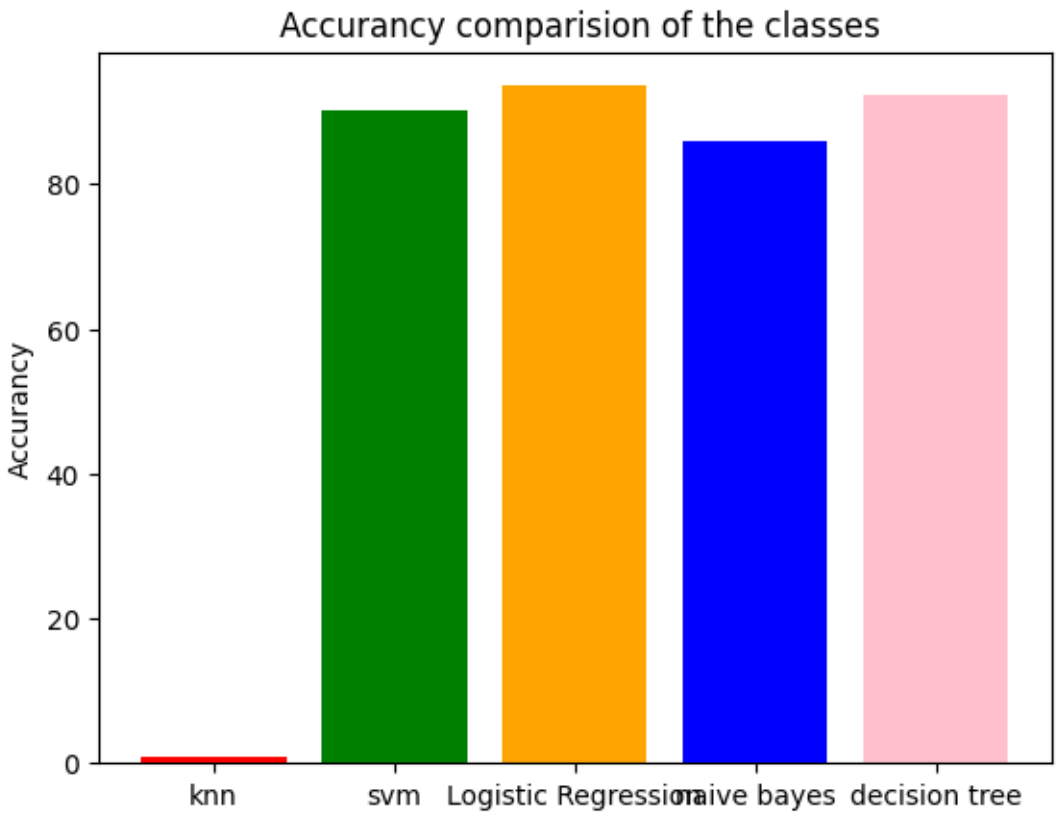
8.4 result visualization

[115]:
```
result.sort_values(by='f1',ascending=False).style.background_gradient(sns.
↪color_palette("Spectral", as_cmap=True))
```

[115]:
```
<pandas.io.formats.style.Styler at 0x7e3361e1b520>
```

[121]:
```
# Accurancy comparision of the classes
class_name = ('knn','svm','Logistic Regression','naive bayes','decision tree')
class_score=(acc_knn,accuracy_svm,accuracy_lg,accuracy_nb,accuracy_dt)
colors=('r','g','orange','b','pink')
plt.bar(class_name,class_score,color=colors)
plt.title(' Accurancy comparision of the classes')
plt.ylabel('Accurancy')
```

[121]:
```
Text(0, 0.5, 'Accurancy')
```


Accurancy comparision of the classes
Accurancy
0
20
40
60
80
knn
svm
Logistic Regressio
naive bayes
decision tree