21BAI1533 PCA and SVM

PCA

In [83]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [84]:
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/ir
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-widt
df=pd.read_csv(url, names=names)
```

In [85]:
```
df.head()
```

Out[85]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [86]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal-length  150 non-null    float64
 1   sepal-width   150 non-null    float64
 2   petal-length  150 non-null    float64
 3   petal-width   150 non-null    float64
 4   Class         150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [87]:
```
df.describe()
```

Out[87]:

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [88]:
```
x=df.drop('Class', 1)
y=df['Class']
```

```
/var/folders/72/3kxng2yd5yn203b626zw3kpc0000gn/T/ipykernel_17046/3
513278735.py:1: FutureWarning: In a future version of pandas all a
rguments of DataFrame.drop except for the argument 'labels' will b
e keyword-only.
  x=df.drop('Class', 1)
```

In [89]:
```
from sklearn.model_selection import train_test_split
```

In [90]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [91]:
```
from sklearn.preprocessing import StandardScaler
```

In [92]:
```
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [93]:
```
from sklearn.decomposition import PCA
```

In [94]:
```
pca = PCA()
```

In [95]:
```
x_train=pca.fit_transform(x_train)
x_test=pca.transform(x_test)
```

In [96]:
```
var=pca.explained_variance_ratio_
var
```

Out[96]:
```
array([0.73053554, 0.22968726, 0.03446579, 0.00531141])
```

In [97]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [98]:
```
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(x_train, y_train)
```

Out[98]:
```
RandomForestClassifier(max_depth=2, random_state=0)
```

In [100]:
```
y_pred = classifier.predict(x_test)
```

In [101]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

In [102]:
```
pca = PCA(n_components=1)
x_train = pca.fit_transform(x_train)
x_test = pca.transform(x_test)
```

In [103]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
```

```
[[15  0  0]
 [ 0 10  8]
 [ 0  0 12]]
0.8222222222222222
```

In [107]:
```
pca = PCA()
X_train = pca.fit_transform(x_train)
X_test = pca.transform(x_test)
```

In [108]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
print(accuracy_score(y_test, y_pred))
```

```
[[15  0  0]
 [ 0 10  8]
 [ 0  0 12]]
0.8222222222222222
```

SVM

In [156]:
```
from sklearn.datasets import load_breast_cancer
```

In [157]:
```
cancer = load_breast_cancer()
```

In [158]:
```
col_names = list(cancer.feature_names)
col_names.append('target')
df = pd.DataFrame(np.c_[cancer.data, cancer.target], columns=col_na
df.head()
```

Out[158]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	symr
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0

5 rows × 31 columns

In [159]:
```
print(cancer.target_names)
```

```
['malignant' 'benign']
```

In [160]:
```
df.describe()
```

Out[160]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	conc
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.08
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.07
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.00
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.02
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.06
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.13
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.42

8 rows × 31 columns

In [161]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    float64
dtypes: float64(31)
memory usage: 137.9 KB
```

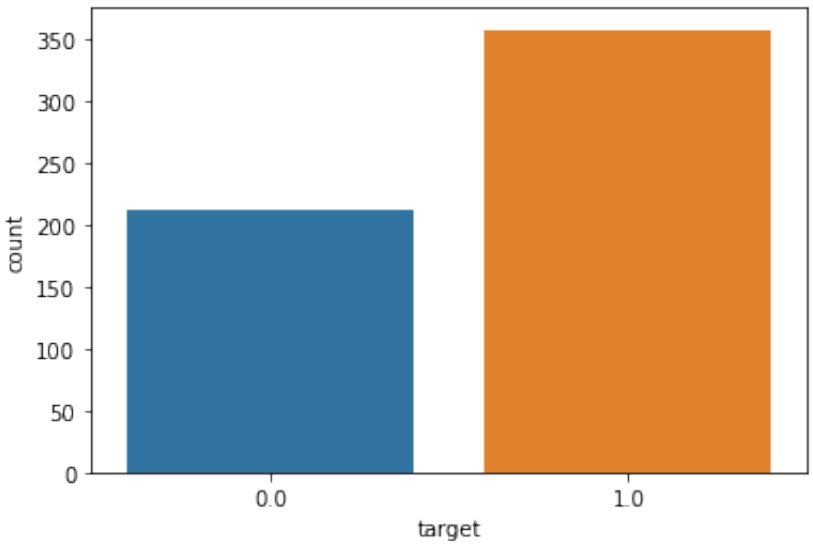
In [162]:
```
df.columns
```

Out[162]:
```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area
',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimen
sion',
       'radius error', 'texture error', 'perimeter error', 'area e
rror',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimensio
n error',
       'worst radius', 'worst texture', 'worst perimeter', 'worst
area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal di
mension',
       'target'],
      dtype='object')
```

In [163]:
```
#sns.pairplot(df,hue='target')
```

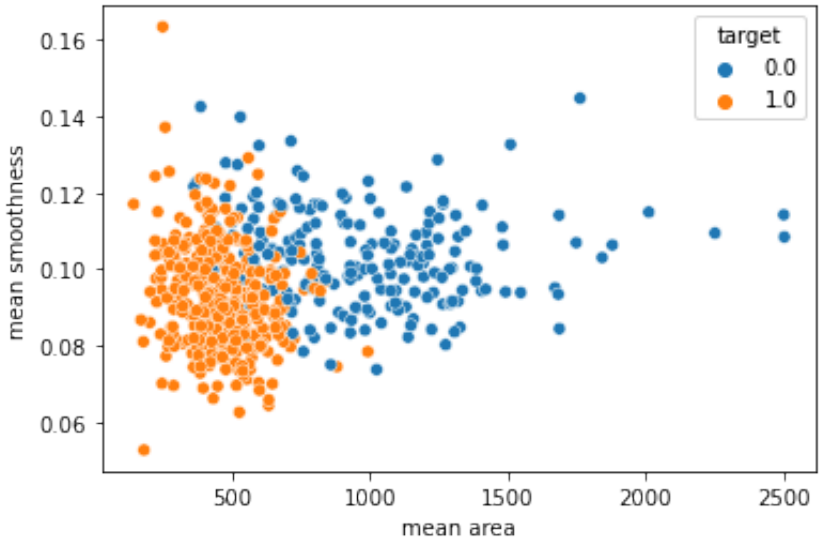
In [164]:
```
sns.countplot(x=df['target'], label = "Count")
```

Out[164]:
```
<AxesSubplot:xlabel='target', ylabel='count'>
```



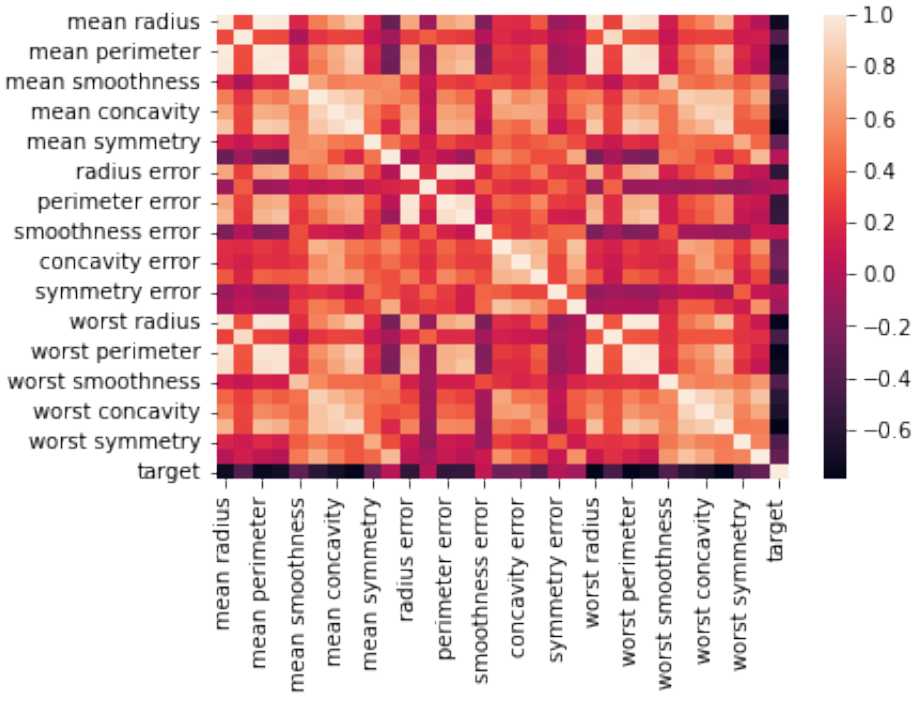
In [165]:
```
sns.scatterplot(x = 'mean area', y = 'mean smoothness', hue = 'targ
```

Out[165]: <AxesSubplot:xlabel='mean area', ylabel='mean smoothness'>



In [166]:
```
sns.heatmap(df.corr())
```

Out[166]: <AxesSubplot:>



In [167]:
```
from sklearn.model_selection import cross_val_score, train_test_spl
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

In [168]:
```
x=df.drop('target', axis=1)
y=df.target
```

In [169]:
```
x.shape
```

Out[169]: (569, 30)

In [170]:
```
y.shape
```

Out[170]: (569,)

In [171]:
```
,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

In [179]:
```
sc=StandardScaler()
```

In [180]:
```
sc.fit(df)
```

```
/Users/zeelmehta/opt/anaconda3/lib/python3.9/site-packages/sklearn
/utils/validation.py:1688: FutureWarning: Feature names only suppo
rt names that are all strings. Got feature names with dtypes: ['st
r', 'str_']. An error will be raised in 1.2.
  warnings.warn(
```

Out[180]: StandardScaler()

In [182]:
```
scaled=sc.transform(df)
```

```
/Users/zeelmehta/opt/anaconda3/lib/python3.9/site-packages/sklearn
/utils/validation.py:1688: FutureWarning: Feature names only suppo
rt names that are all strings. Got feature names with dtypes: ['st
r', 'str_']. An error will be raised in 1.2.
  warnings.warn(
```

In [184]:
```
from sklearn.decomposition import PCA
```

In [186]:
```
pca=PCA(n_components=2)
```

In [188]:
```
pca.fit(scaled)
```

Out[188]: PCA(n_components=2)

In [189]:
```
xpca=pca.transform(scaled)
```

In [190]:
```
scaled.shape
```

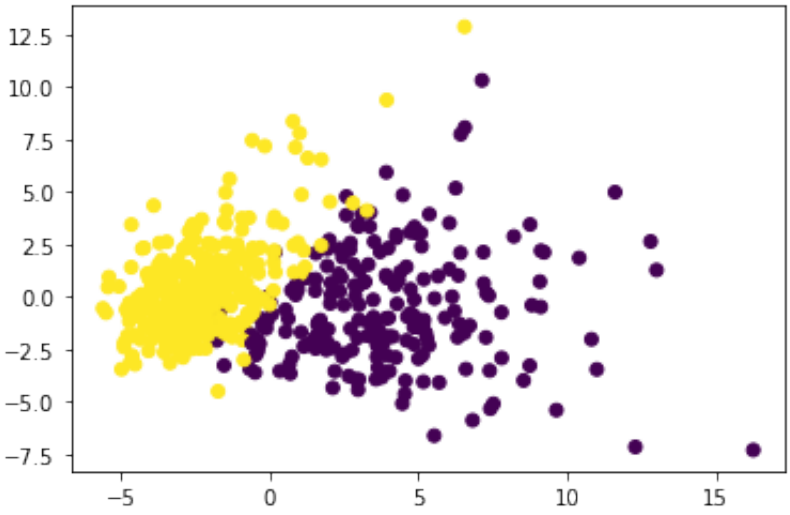
Out[190]: (569, 31)

In [191]:
```
xpca.shape
```

Out[191]: (569, 2)

In [194]:
```
plt.scatter(xpca[:,0],xpca[:,1],c=cancer['target'])
```

Out[194]:
```
<matplotlib.collections.PathCollection at 0x2e8bc9fa0>
```



In [55]:
```
pca.components_
```

Out[55]:
```
array([[-0.21890244, -0.10372458, -0.22753729, -0.22099499, -0.142
58969,
        -0.23928535, -0.25840048, -0.26085376, -0.13816696, -0.064
36335,
        -0.20597878, -0.01742803, -0.21132592, -0.20286964, -0.014
53145,
        -0.17039345, -0.15358979, -0.1834174 , -0.04249842, -0.102
56832,
        -0.22799663, -0.10446933, -0.23663968, -0.22487053, -0.127
95256,
        -0.21009588, -0.22876753, -0.25088597, -0.12290456, -0.131
78394],
       [ 0.23385713,  0.05970609,  0.21518136,  0.23107671, -0.186
11302,
        -0.15189161, -0.06016536,  0.0347675 , -0.19034877, -0.366
57547,
         0.10555215, -0.08997968,  0.08945723,  0.15229263, -0.204
43045,
        -0.2327159 , -0.19720728, -0.13032156, -0.183848  , -0.280
09203,
         0.21986638,  0.0454673 ,  0.19987843,  0.21935186, -0.172
30435,
        -0.14359317, -0.09796411,  0.00825724, -0.14188335, -0.275
33947]])
```

In [195]:
```
from sklearn.svm import SVC
```

In [196]:
```
svm=SVC()
```

In [197]:
```
svm.fit(x_train,y_train)
```

```
/Users/zeelmehta/opt/anaconda3/lib/python3.9/site-packages/sklearn
/utils/validation.py:1688: FutureWarning: Feature names only suppo
rt names that are all strings. Got feature names with dtypes: ['st
r_']. An error will be raised in 1.2.
  warnings.warn(
```

Out[197]:
```
SVC()
```

In [198]:
```
pred=svm.predict(x_test)
```

```
/Users/zeelmehta/opt/anaconda3/lib/python3.9/site-packages/sklearn
/utils/validation.py:1688: FutureWarning: Feature names only suppo
rt names that are all strings. Got feature names with dtypes: ['st
r_']. An error will be raised in 1.2.
  warnings.warn(
```

In [199]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [200]:
```
print(confusion_matrix(y_test,pred))
```

```
[[ 52  11]
 [  0 108]]
```

In [201]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.83      0.90        63
         1.0       0.91      1.00      0.95       108

    accuracy                           0.94       171
   macro avg       0.95      0.91      0.93       171
weighted avg       0.94      0.94      0.93       171
```