데이터마이닝 과제

2016170864 산업경영공학부 조동혁

In [1]:
```
import pandas as pd
from sklearn import datasets
load_df=datasets.load_breast_cancer()

data=pd.DataFrame(load_df.data)
feature=pd.DataFrame(load_df.feature_names)
data.columns=feature[0]
target=pd.DataFrame(load_df.target)
target.columns=['target']
df=pd.concat([data,target], axis=1)
print(df.shape)
df.head()
```

```
(569, 31)
```

Out[1]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	com
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	

5 rows × 31 columns

데이터는 Breast Cancer 데이터를 사용했습니다. 문제 정의는: 어떤 설명변수가 가장 Breast Cancer 에 영향을 많이 주는지 확인을 하는 것입니다.

In [2]:
```
# dictionary 키값 확인
print(df.keys())
```

```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimension error',
       'worst radius', 'worst texture', 'worst perimeter', 'worst area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimension',
       'target'],
      dtype='object')
```

In [3]:
```
#변수가 너무 많기에, 차원축소를 진행 해주었습니다
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

In [4]:
```
# 데이터 가져오기

from sklearn.datasets import load_breast_cancer
cancer=load_breast_cancer()
X = cancer.data
```

In [5]:
```
pd.DataFrame(X)
```

Out[5]:

	0	1	2	3	4	5	6	7	8	9	...	20	21	22	23	24	25	26	27	
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	25.380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.4
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	24.990	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.2
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	23.570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.3
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	14.910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.6
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	22.540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2

569 rows × 30 columns

key 값들을 확인해준결과, 데이터의 양 및 정보가 너무 많아서, PCA 로 차원축소를 하기로 결정했습니다.

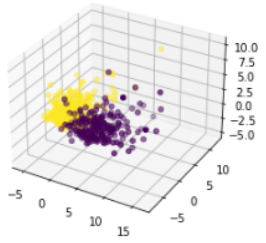
In [6]:
```
# Data Normalizing (Mean Centering)

from sklearn.preprocessing import StandardScaler
X_ = StandardScaler().fit_transform(X)
```

In [17]:
```
# PCA 수행 - 주성분 3개
pca = PCA(n_components=3)
pc = pca.fit_transform(X_)
```

In [18]:
```
# 카테고리정보 (음성 양성)
pc_y = np.c_[pc,y]
df = pd.DataFrame(pc_y,columns=['PC1','PC2','PC3','diagnosis'])
```

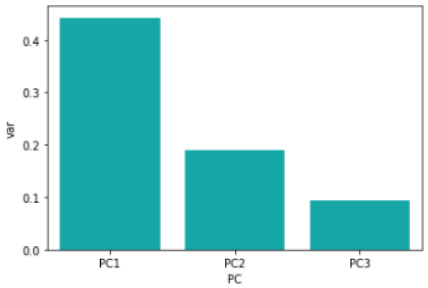
In [19]:
```
#Plotting

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df['PC1'], df['PC2'], df['PC3'], c=df['diagnosis'],  s=20)
```

Out[19]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x2b0033287c0>
```



우선, 변수들을 표준화 작업을 mean centering 용법으로 해주었습니다. 그 후에는 주성분을 3 개를 기준으로 계산을 해주었으며, 3 차원 공간에 plotting 을 해주었습니다.

In [30]:
```
# PCA 수행 - 주성분 3개
pca = PCA(n_components=3)
pc = pca.fit_transform(X_)

df_var = pd.DataFrame({'var':pca.explained_variance_ratio_,
             'PC':['PC1','PC2', 'PC3']})
sns.barplot(x='PC',y="var",
           data=df_var, color="c");
```



PCA 결과: 주성분 3 개가 합쳐서 대략 70%의 데이터를 설명해줄수 있었습니다.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn import datasets
from matplotlib import pyplot as plt
```

In [4]:
```
cancer = datasets.load_breast_cancer()
data=pd.DataFrame(cancer.data) ; data
feature=pd.DataFrame(cancer.feature_names) ; feature


data.columns=feature[0]
target=pd.DataFrame(cancer.target)
target.columns=['target']

df=pd.concat([data,target], axis=1)
df.head()
```

Out[4]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	com
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	

5 rows × 31 columns

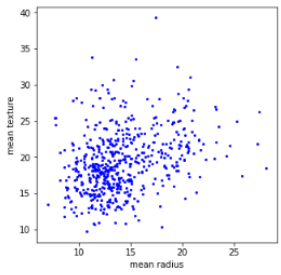
Breast Cancer 데이터의 군집화를 보기위해서 Clustering 작업을 진행 해주었습니다.

In [6]:
```
#2차원 공간에 가시화 시키기
df_f = df.copy()
```

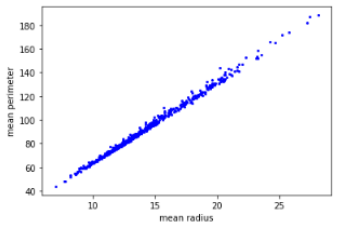
In [22]:
```
fig = plt.figure(figsize=(5,5))
X = df_f
for col_n in range(1,4):
    print(col_n)
    plt.plot( X.iloc[:,0] , X.iloc[:,col_n] , 'o' , markersize=2 , color='blue' )
    plt.xlabel(data.columns[0] )
    plt.ylabel(data.columns[col_n] )
    plt.show()
```

위 데이터들을 2 차원 공간에 mean texture, perimeter, area 를 기준으로 가시화 시켜주었습니다.

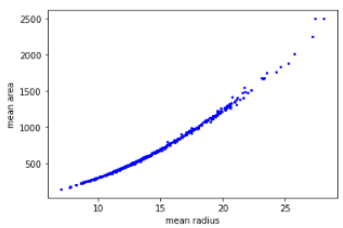
1



2

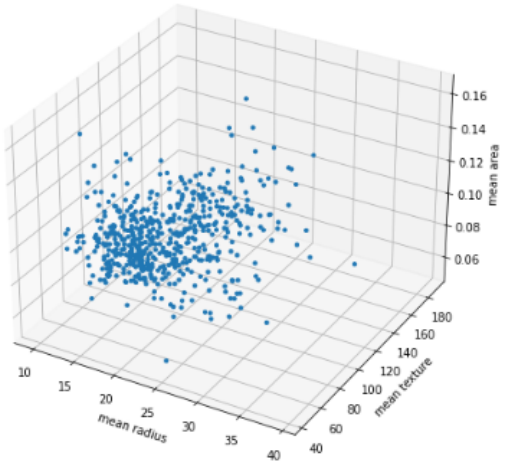


3



In [8]:
```
# 3차원 그리기
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')
X = df_f
# 3d scatterplot 그리기
print('X=',data.columns[0] ,'Y=', data.columns[1] , 'Z=',data.columns[3] )
ax.scatter( X.iloc[:,1] , X.iloc[:,2] , X.iloc[:,4]
           ,s=10
           ,cmap="orange"
           ,alpha=1
          )

ax.set_xlabel(data.columns[0] )
ax.set_ylabel(data.columns[1] )
ax.set_zlabel(data.columns[3] )

plt.show()
```

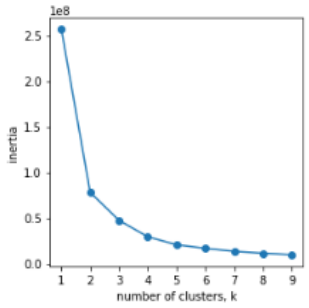
```
X= mean radius Y= mean texture Z= mean area
```



3 차원 공간상에서도 가시화를 시켜주었습니다. 사이즈는 10, 투명도는 1 로 설정해주었습니다. 2 차원, 3 차원 공간상에 가시화된 plot 들을 보니, 데이터가 군집화가 전혀 안되어 있는 모습을 볼수 있었습니다. K-means 를 사용해서 군집수를 찾고 다시 plotting 을 진행 해주었습니다.

In [9]:
```
from sklearn.cluster import KMeans


ks = range(1,10)
inertias = []

for k in ks:
    model = KMeans(n_clusters=k)
    model.fit(df_f)
    inertias.append(model.inertia_)


plt.figure(figsize=(4, 4))

plt.plot(ks, inertias, '-o')
plt.xlabel('number of clusters, k')
plt.ylabel('inertia')
plt.xticks(ks)
plt.show()
```

```
C:₩Users₩Administrator₩anaconda3₩lib₩site-packages₩sklearn₩cluster₩_kmeans.py:881: UserWarning: KMeans is known to have a memory
leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=3.
  warnings.warn(
```



k 개수를 늘렸더니, k=3 일때가 elbow point 인 것을 알수가 있었습니다.

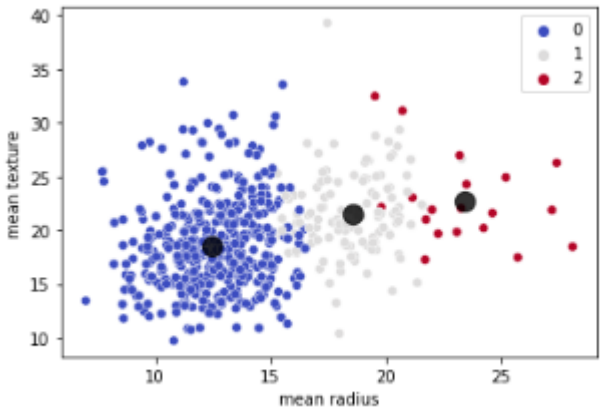
In [23]:
```
clust_model = KMeans(n_clusters = 3
                     , n_init=10
                     , max_iter=50
                     , random_state = 42)

clust_model.fit(df_f)


centers = clust_model.cluster_centers_
pred = clust_model.predict(df_f)
print(pd.DataFrame(centers))
print(pred[:10])
```

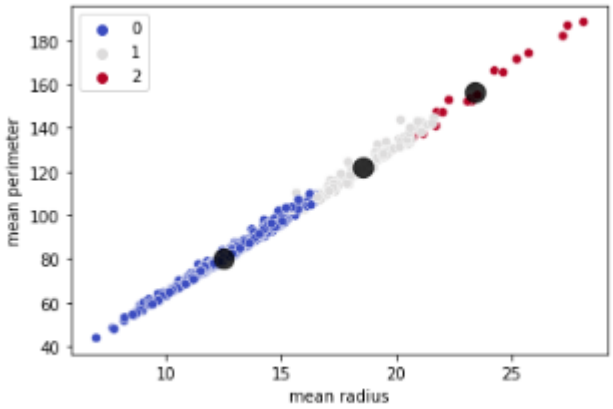
```
          0         1          2           3         4         5  ₩
0  12.475172  18.490117   80.576410   488.859207  0.094915  0.090775
1  18.528512  21.579091  122.283471  1074.812397  0.100261  0.141827
2  23.401579  22.762105  156.147368  1729.421053  0.104154  0.171922

          6         7         8         9   ...         21          22  ₩
0  0.061506  0.032939  0.177987  0.063573  ...  24.599044   91.227110
1  0.161977  0.092081  0.191677  0.060620  ...  28.774711  149.064463
2  0.239016  0.134858  0.185884  0.059145  ...  30.295263  203.073684

            23        24        25        26        27        28        29  ₩
0   609.272261  0.130029  0.222372  0.216940  0.090475  0.282925  0.083394
1  1546.471074  0.139227  0.346118  0.431358  0.182433  0.315463  0.086227
2  2765.842105  0.141511  0.389416  0.505995  0.227526  0.289853  0.081874

          30
0  0.827506
1  0.016529
2  0.000000

[3 rows x 31 columns]
[1 1 1 0 1 0 1 0 0 0]
```

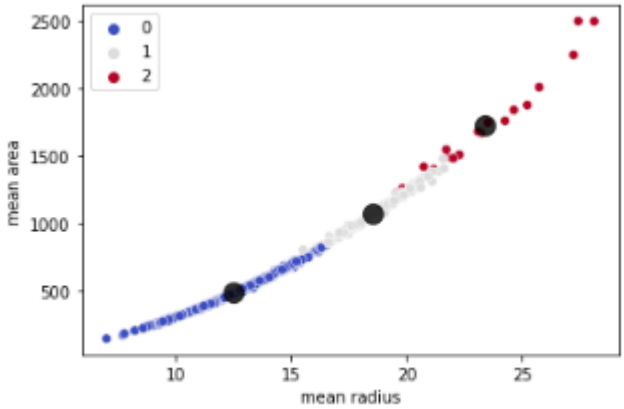
Cluster 개수 =3 으로 K-means 모델링을 진행 해주었습니다. 또한, 각 군집의 Center-value 를 가져왔습니다.

In [25]:
```
import seaborn as sns


X = clust_df


sns.scatterplot(x=X.iloc[:,0], y=X.iloc[:,1], data=df_f, hue=clust_model.labels_, palette='coolwarm')
plt.scatter(centers[:,0], centers[:,1], c='black', alpha=0.8, s=150)
plt.show()


sns.scatterplot(x=X.iloc[:,0], y=X.iloc[:,2], data=df_f, hue=clust_model.labels_, palette='coolwarm')
plt.scatter(centers[:,0], centers[:,2], c='black', alpha=0.8, s=150)
plt.show()

sns.scatterplot(x=X.iloc[:,0], y=X.iloc[:,3], data=df_f, hue=clust_model.labels_, palette='coolwarm')
plt.scatter(centers[:,0], centers[:,3], c='black', alpha=0.8, s=150)
plt.show()

#2차원으로 시각화하기
```

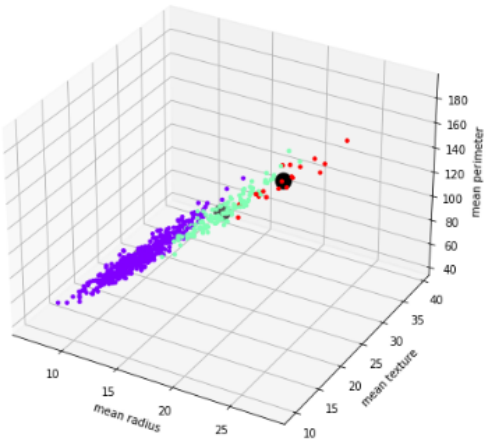
결과를 다시 plotting 을 해주었습니다. 각 검은 점들은 각 cluster 들의 중심값을 나타내주고 있는 모습입니다.



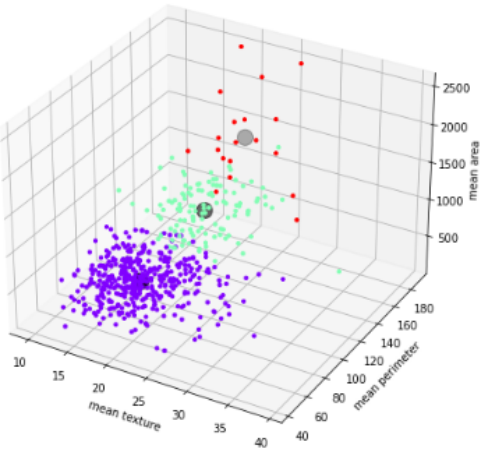




In [26]:
```
# 3차원으로 시각화하기
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')
X = clust_df


ax.scatter( X.iloc[:,0] , X.iloc[:,1] , X.iloc[:,2] , c = X.clust , s = 10 , cmap = "rainbow" , alpha = 1 )


ax.scatter(centers[:,0],centers[:,1],centers[:,2] ,c='black', s=200)

ax.set_xlabel(data.columns[0] )
ax.set_ylabel(data.columns[1] )
ax.set_zlabel(data.columns[2] )

plt.show()
```



In [27]:
```
# 3차원으로 시각화하기
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')
X = clust_df


ax.scatter( X.iloc[:,1] , X.iloc[:,2] , X.iloc[:,3] , c = X.clust , s = 10 , cmap = "rainbow" , alpha = 1 )


ax.scatter(centers[:,1],centers[:,2],centers[:,3] ,c='black', s=200)

ax.set_xlabel(data.columns[1] )
ax.set_ylabel(data.columns[2] )
ax.set_zlabel(data.columns[3] )

plt.show()
```



변수가 4 개지만, k=3 이기 때문에, 변수 4 개를 두번에 걸쳐서 3 차원에서 plotting 했습니다.

데이터를 가지고 이제 modeling 을 진행 해주었습니다.

In [11]:
```
from sklearn import datasets
import pandas as pd
```

In [12]:
```
cancer = datasets.load_breast_cancer()
cancer
```

Out [12]:
```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
```

In [13]:
```
print(cancer.keys())
```

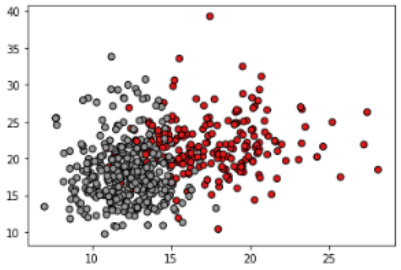
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

In [16]:
```
cancer['target']
```

Out [16]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

Dataset 을 다시 와서 load 를 해주었습니다. Dictionary 형태로 표현을 해주었습니다.

```
In [17]: X = cancer.data[:, :2]
         y = cancer.target

In [18]: import matplotlib.pyplot as plt
         plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor='k')
Out[18]: <matplotlib.collections.PathCollection at 0x263f6645340>
```




```
In [19]: from sklearn.neighbors import KNeighborsClassifier
         from sklearn.model_selection import train_test_split, cross_val_score


         x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=30)

         KNN_MODEL = KNeighborsClassifier()
         KNN_MODEL.fit(x_train, y_train)
         prediction = KNN_MODEL.predict(x_test)


         print("Training error       : {0:.3f}".format(KNN_MODEL.score(x_train, y_train)))
         print("Testing error        : {0:.3f}".format((prediction==y_test).mean()))



         print("cross_val_score      : {0:.3f}".format(cross_val_score(KNN_MODEL, X, y, cv=10).mean()))

Training error       : 0.910
Testing error        : 0.901
cross_val_score      : 0.873
```

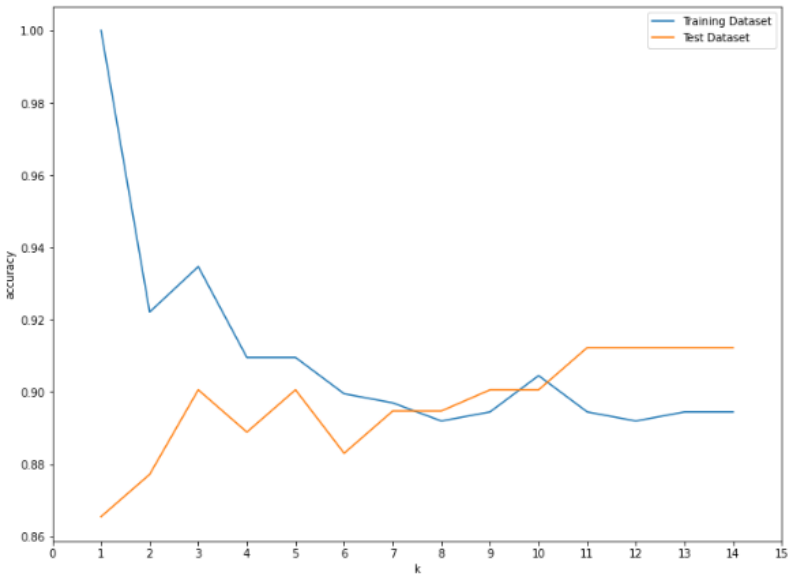
x 변수를 가져와야하지만, 전부 다 가져왔더니 실행이 너무 느려서, 두개의 변수를 가지고 와서 설정을 해주었습니다.

또한, 예측성능을 진행 해주었습니다. Test size 는 30%로 잡아주었고, training test 는 70%로 설정 해주었습니다. Training error 는 91%, testing error 는 90%가 나왔습니다.

In [20]:
```
train_acc = []
test_acc = []

for n in range(1,15):
    KNN_MODEL = KNeighborsClassifier(n_neighbors=n)
    KNN_MODEL.fit(x_train, y_train)
    prediction = KNN_MODEL.predict(x_test)
    train_acc.append(KNN_MODEL.score(x_train, y_train))
    test_acc.append((prediction==y_test).mean())
```

In [21]:
```
import numpy as np
plt.figure(figsize=(12, 9))
plt.plot(range(1, 15), train_acc, label='Training Dataset')
plt.plot(range(1, 15), test_acc, label='Test Dataset')
plt.xlabel("k")
plt.ylabel("accuracy")
plt.xticks(np.arange(0, 16, step=1))
plt.legend()
```

Out [21]:
```
<matplotlib.legend.Legend at 0x263f6a85a30>
```



마지막으로, Training accuracy, test accuracy 를 1~15 로 주면서 모델링을 해주었습니다. Test data 는 11~14 에서 높은 모습을 보여주고 있습니다. K 값을 11~14 로 잡고 모델링을 해주면 될거 같다는 결론을 지었습니다.

In [5]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
```

In [6]:
```
cancer=load_breast_cancer()
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target, test_size=0.2, random_state=24)
```

In [8]:
```
from sklearn.tree import DecisionTreeClassifier



DT_MODEL= DecisionTreeClassifier(random_state=0)
DT_MODEL.fit(X_train, y_train)



prediction=DT_MODEL.predict(X_test)
```

In [9]:
```


from sklearn.metrics import classification_report, confusion_matrix
CM=confusion_matrix(y_test, prediction)
CM_report=classification_report(y_test, prediction)
print('-'*15, 'Confusion Matrix','-'*15)
print(CM)

print('-'*15, 'Confusion Matrix2','-'*15)
import pandas as pd
CM_rename=pd.DataFrame(CM).rename(index={0:'실제값(0)',1:'실제값(1)',2:'실제값(2)'},columns={0:'예측값(0)',1:'예측값(1)',2:'예측값
print(CM_rename)
```

```
--------------- Confusion Matrix ---------------
[[36  5]
 [ 3 70]]
--------------- Confusion Matrix2 ---------------
       예측값(0)  예측값(1)
실제값(0)      36       5
실제값(1)       3      70
```

In [10]:
```


print('-'*20, '성능평가','-'*20)
print(CM_report)
```

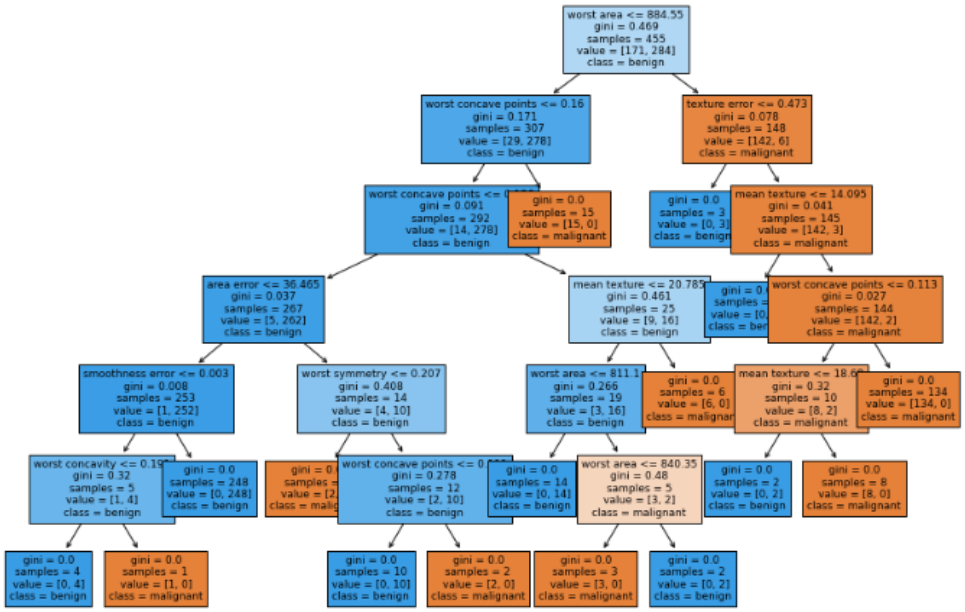
```
-------------------- 성능평가 --------------------
              precision    recall  f1-score   support

           0       0.92      0.88      0.90        41
           1       0.93      0.96      0.95        73

    accuracy                           0.93       114
   macro avg       0.93      0.92      0.92       114
weighted avg       0.93      0.93      0.93       114
```

의사결정나무 algorithm 을 DecisionTreeClassifier 를 사용해주었습니다. Test size 는 8:2 비율로 해주었습니다. Confusion matrix 및 prediction 을 진행 해주었습니다.

CM_report 를 통해서 성능평가를 해주었습니다.

In [12]:

```
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
plt.figure(figsize=(15,10))
plot_tree(DT_MODEL,feature_names=cancer.feature_names, class_names=cancer.target_names, filled=True, fontsize=9)
plt.show()
```

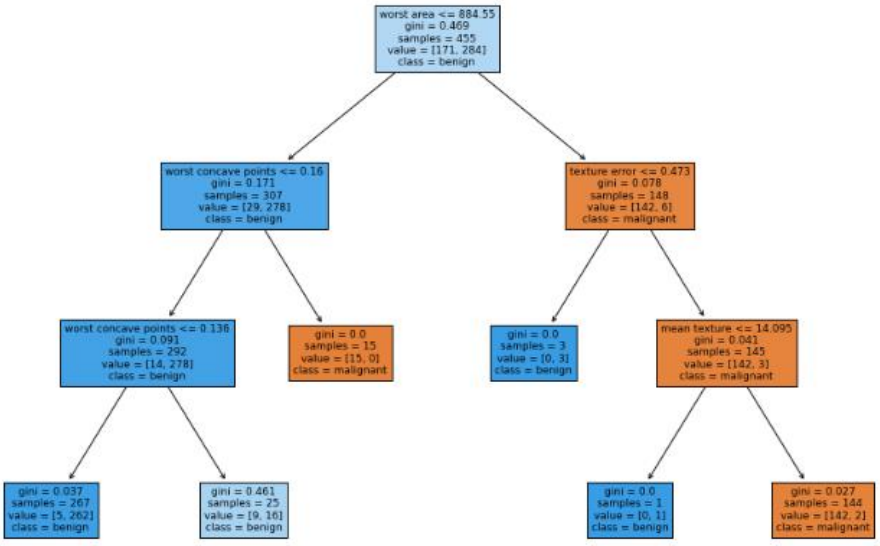


의사결정나무를 가시화 해주었습니다. 455 개의 샘플을 가지고 해주었습니다.

In [14]:

```
DT_MODEL_DEP3= DecisionTreeClassifier(max_depth=3, random_state=0)
DT_MODEL_DEP3.fit(X_train, y_train)

prediction_DEP3=DT_MODEL_DEP3.predict(X_test)

CM_DEP3=confusion_matrix(y_test, prediction_DEP3)
CM_report_DEP3=classification_report(y_test, prediction_DEP3)

print('-'*15, 'Confusion Matrix','-'*15)
print(CM_DEP3)
print('-'*20, '성능평가','-'*20)
print(CM_report_DEP3)

plt.figure(figsize=(15,10))
plot_tree(DT_MODEL_DEP3,feature_names=cancer.feature_names, class_names=cancer.target_names, filled=True, fontsize=9)
plt.show()
```

```
--------------- Confusion Matrix ---------------
[[35  6]
 [ 3 70]]
-------------------- 성능평가 --------------------
              precision    recall  f1-score   support

           0       0.92      0.85      0.89        41
           1       0.92      0.96      0.94        73

    accuracy                           0.92       114
   macro avg       0.92      0.91      0.91       114
weighted avg       0.92      0.92      0.92       114
```

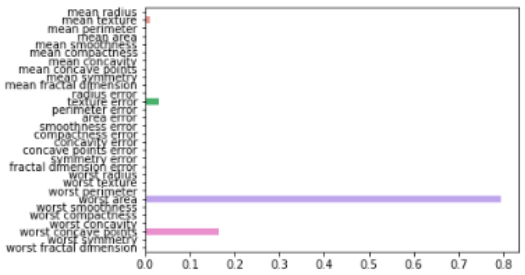


가지치기 진행할때는, Depth 설정은 3 으로 해주었습니다. 결과를 confusion matrix 로 보니까, 아까보다 한 개 더 정답을 맞추어 주었습니다.

In [22]:
```
# 변수 중요도

for name, value in zip(cancer.feature_names , DT_MODEL_DEP3.feature_importances_):
    print('{0} : {1:.3f}'.format(name, value))

# 변수 중요도 가시화

import seaborn as sns
sns.barplot(x=DT_MODEL_DEP3.feature_importances_ , y=cancer.feature_names)
```

```
mean radius : 0.000
mean texture : 0.010
mean perimeter : 0.000
mean area : 0.000
mean smoothness : 0.000
mean compactness : 0.000
mean concavity : 0.000
mean concave points : 0.000
mean symmetry : 0.000
mean fractal dimension : 0.000
radius error : 0.000
texture error : 0.030
perimeter error : 0.000
area error : 0.000
smoothness error : 0.000
compactness error : 0.000
concavity error : 0.000
concave points error : 0.000
symmetry error : 0.000
fractal dimension error : 0.000
worst radius : 0.000
worst texture : 0.000
worst perimeter : 0.000
worst area : 0.794
worst smoothness : 0.000
worst compactness : 0.000
worst concavity : 0.000
worst concave points : 0.166
worst symmetry : 0.000
worst fractal dimension : 0.000
```

Out [22]:
```
<AxesSubplot:>
```



마지막으로 변수의 중요도를 결정 해주었습니다. 결과로 봤을 때, worst area 가 가장 중요한 변수였으며, 그 다음으로는 worst concave pointsm texture error, mean texture 순서대로 높은 중요도를 보여주고 있습니다.