In [1]:
```
import io #modulo para realizar distintas operaciones en Python.
import pandas as pd #Librería de análisis de datos.
import matplotlib.pyplot as plt #Libraría de representación de gráficas.
import seaborn as sb #Herramienta de visualización de datos.
import numpy as np #Numpy facilita un largo set de tipos de datos numéricos pa
ra construir arrays.
from scipy.stats import norm #Librería con herramientas y algoritmos matemátic
os.
from scipy import stats
from sklearn import svm #Librería con algoritmos de aprendizaje
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import OneClassSVM
from sklearn.cluster import DBSCAN
from pylab import rcParams
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn import metrics
from sklearn.preprocessing import MinMaxScaler
```

Lectura de datos

In [2]:
```
#Read CSV
wdbc = pd.read_csv(
    'wdbc.data',
    encoding='utf-8',
    delim_whitespace=False,
    names=["id","diagnostic",'mean radius', 'mean texture', 'mean perimter',
'mean area', 'mean smoothnesss', 'mean compactness', 'mean concavity', 'mean c
oncave points', 'mean symetry','mean fractal dimension','se radius', 'se textu
re', 'se perimter', 'se area', 'se smoothnesss', 'se compactness', 'se concavi
ty', 'se concave points', 'se symetry','se fractal dimension''worst radius',
'worst texture', 'worst perimter', 'worst area', 'worst smoothnesss', 'worst c
ompactness', 'worst concavity', 'worst concave points', 'worst symetry','worst
fractal dimension'],
    index_col=False,
    header=None,
)
display(wdbc)
```

	id	diagnostic	mean radius	mean texture	mean perimter	mean area	mean smoothnesss	mean compactness	m conca
0	842302	M	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300
1	842517	M	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086
2	84300903	M	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197
3	84348301	M	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241
4	84358402	M	20.290	14.34	135.10	1297.0	0.10030	0.13280	0.198
5	843786	M	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157
6	844359	M	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112
7	84458202	M	13.710	20.83	90.20	577.9	0.11890	0.16450	0.093
8	844981	M	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185
9	84501001	M	12.460	24.04	83.97	475.9	0.11860	0.23960	0.227
10	845636	M	16.020	23.24	102.70	797.8	0.08206	0.06669	0.032
11	84610002	M	15.780	17.89	103.60	781.0	0.09710	0.12920	0.099
12	846226	M	19.170	24.80	132.40	1123.0	0.09740	0.24580	0.206
13	846381	M	15.850	23.95	103.70	782.7	0.08401	0.10020	0.099
14	84667401	M	13.730	22.61	93.60	578.3	0.11310	0.22930	0.212
15	84799002	M	14.540	27.54	96.73	658.8	0.11390	0.15950	0.163
16	848406	M	14.680	20.13	94.74	684.5	0.09867	0.07200	0.073
17	84862001	M	16.130	20.68	108.10	798.8	0.11700	0.20220	0.172
18	849014	M	19.810	22.15	130.00	1260.0	0.09831	0.10270	0.147
19	8510426	B	13.540	14.36	87.46	566.3	0.09779	0.08129	0.066
20	8510653	B	13.080	15.71	85.63	520.0	0.10750	0.12700	0.045
21	8510824	B	9.504	12.44	60.34	273.9	0.10240	0.06492	0.029
22	8511133	M	15.340	14.26	102.50	704.4	0.10730	0.21350	0.207
23	851509	M	21.160	23.04	137.20	1404.0	0.09428	0.10220	0.109
24	852552	M	16.650	21.38	110.00	904.6	0.11210	0.14570	0.152
25	852631	M	17.140	16.40	116.00	912.7	0.11860	0.22760	0.222
26	852763	M	14.580	21.53	97.41	644.8	0.10540	0.18680	0.142
27	852781	M	18.610	20.25	122.10	1094.0	0.09440	0.10660	0.149
28	852973	M	15.300	25.27	102.40	732.4	0.10820	0.16970	0.168
29	853201	M	17.570	15.05	115.00	955.1	0.09847	0.11570	0.098
...	...	...	...	...	...	...	...	...	
539	921362	B	7.691	25.44	48.34	170.4	0.08668	0.11990	0.092
540	921385	B	11.540	14.44	74.65	402.9	0.09984	0.11200	0.067
541	921386	B	14.470	24.99	95.81	656.4	0.08837	0.12300	0.100

	id	diagnostic	mean radius	mean texture	mean perimter	mean area	mean smoothnesss	mean compactness	m conca
542	921644	B	14.740	25.42	94.70	668.6	0.08275	0.07214	0.041
543	922296	B	13.210	28.06	84.88	538.4	0.08671	0.06877	0.029
544	922297	B	13.870	20.70	89.77	584.8	0.09578	0.10180	0.036
545	922576	B	13.620	23.23	87.19	573.2	0.09246	0.06747	0.029
546	922577	B	10.320	16.35	65.31	324.9	0.09434	0.04994	0.010
547	922840	B	10.260	16.58	65.85	320.8	0.08877	0.08066	0.043
548	923169	B	9.683	19.34	61.05	285.7	0.08491	0.05030	0.023
549	923465	B	10.820	24.21	68.89	361.6	0.08192	0.06602	0.015
550	923748	B	10.860	21.48	68.51	360.5	0.07431	0.04227	0.000
551	923780	B	11.130	22.44	71.49	378.4	0.09566	0.08194	0.048
552	924084	B	12.770	29.43	81.35	507.9	0.08276	0.04234	0.019
553	924342	B	9.333	21.94	59.01	264.0	0.09240	0.05605	0.039
554	924632	B	12.880	28.92	82.50	514.3	0.08123	0.05824	0.061
555	924934	B	10.290	27.61	65.67	321.4	0.09030	0.07658	0.059
556	924964	B	10.160	19.59	64.73	311.7	0.10030	0.07504	0.005
557	925236	B	9.423	27.88	59.26	271.3	0.08123	0.04971	0.000
558	925277	B	14.590	22.68	96.39	657.1	0.08473	0.13300	0.102
559	925291	B	11.510	23.93	74.52	403.5	0.09261	0.10210	0.111
560	925292	B	14.050	27.15	91.38	600.4	0.09929	0.11260	0.044
561	925311	B	11.200	29.37	70.67	386.0	0.07449	0.03558	0.000
562	925622	M	15.220	30.62	103.40	716.9	0.10480	0.20870	0.255
563	926125	M	20.920	25.09	143.00	1347.0	0.10990	0.22360	0.317
564	926424	M	21.560	22.39	142.00	1479.0	0.11100	0.11590	0.243
565	926682	M	20.130	28.25	131.20	1261.0	0.09780	0.10340	0.144
566	926954	M	16.600	28.08	108.30	858.1	0.08455	0.10230	0.092
567	927241	M	20.600	29.33	140.10	1265.0	0.11780	0.27700	0.351
568	92751	B	7.760	24.54	47.92	181.0	0.05263	0.04362	0.000

569 rows × 31 columns

In [3]:
```
#Separación de las columnas no utilizadas para crear un modelo
wdbc=wdbc.drop(['id', 'diagnostic'],axis=1)
```

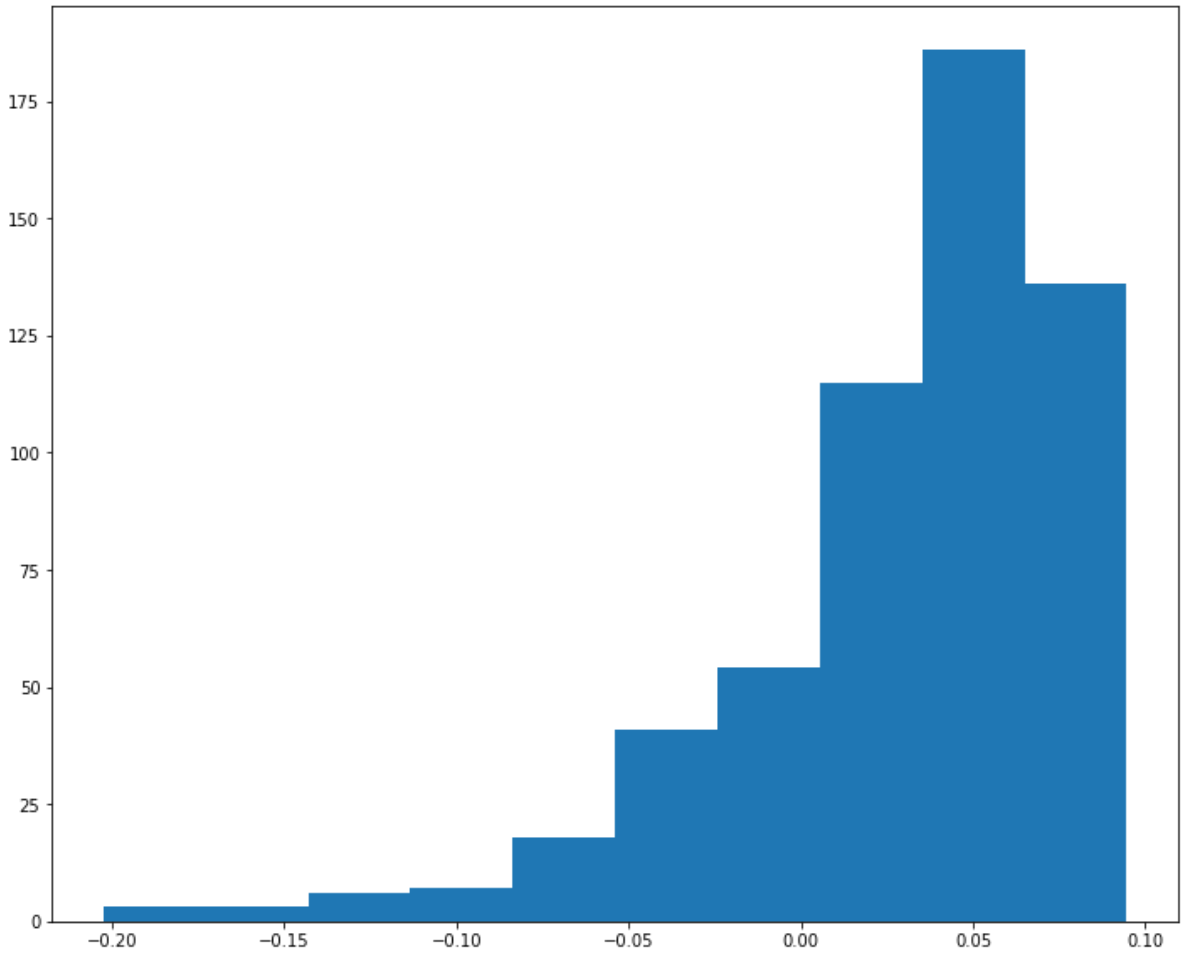
Identificación de Anomalías - Isolation Forest

In [4]:
```
#Modelo
model=IsolationForest(n_estimators=100, max_samples=256, contamination=0.2, ma
x_features=1.0, bootstrap=False, n_jobs=-1, random_state=2019,verbose=0, behav
iour="new")
model.fit(wdbc)
```

Out[4]:
```
IsolationForest(behaviour='new', bootstrap=False, contamination=0.2,
        max_features=1.0, max_samples=256, n_estimators=100, n_jobs=-1,
        random_state=2019, verbose=0)
```

In [5]:
```
#Resultado tras el modelo aplicado
scores=model.decision_function(wdbc)
plt.figure(figsize=(12,10))
plt.hist(scores,bins=10)
```

Out[5]:
```
(array([  3.,   3.,   6.,   7.,  18.,  41.,  54., 115., 186., 136.]),
 array([-0.20259888, -0.17287623, -0.14315358, -0.11343094, -0.08370829,
        -0.05398565, -0.024263  ,  0.00545965,  0.03518229,  0.06490494,
         0.09462758]),
 <a list of 10 Patch objects>)
```



In [6]:
```
#Labelizador según los resultados representados
predwdbc=model.predict(wdbc)
len(predwdbc)
wdbc["label"]=predwdbc
display(wdbc)
```

	mean radius	mean texture	mean perimter	mean area	mean smoothnesss	mean compactness	mean concavity	mean concave points	mear symetry
0	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300100	0.147100	0.241
1	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900	0.070170	0.181
2	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197400	0.127900	0.206
3	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241400	0.105200	0.259
4	20.290	14.34	135.10	1297.0	0.10030	0.13280	0.198000	0.104300	0.180
5	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157800	0.080890	0.208
6	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112700	0.074000	0.179
7	13.710	20.83	90.20	577.9	0.11890	0.16450	0.093660	0.059850	0.219
8	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185900	0.093530	0.235
9	12.460	24.04	83.97	475.9	0.11860	0.23960	0.227300	0.085430	0.203
10	16.020	23.24	102.70	797.8	0.08206	0.06669	0.032990	0.033230	0.152
11	15.780	17.89	103.60	781.0	0.09710	0.12920	0.099540	0.066060	0.184
12	19.170	24.80	132.40	1123.0	0.09740	0.24580	0.206500	0.111800	0.239
13	15.850	23.95	103.70	782.7	0.08401	0.10020	0.099380	0.053640	0.184
14	13.730	22.61	93.60	578.3	0.11310	0.22930	0.212800	0.080250	0.206
15	14.540	27.54	96.73	658.8	0.11390	0.15950	0.163900	0.073640	0.230
16	14.680	20.13	94.74	684.5	0.09867	0.07200	0.073950	0.052590	0.158
17	16.130	20.68	108.10	798.8	0.11700	0.20220	0.172200	0.102800	0.216
18	19.810	22.15	130.00	1260.0	0.09831	0.10270	0.147900	0.094980	0.158
19	13.540	14.36	87.46	566.3	0.09779	0.08129	0.066640	0.047810	0.188
20	13.080	15.71	85.63	520.0	0.10750	0.12700	0.045680	0.031100	0.196
21	9.504	12.44	60.34	273.9	0.10240	0.06492	0.029560	0.020760	0.181
22	15.340	14.26	102.50	704.4	0.10730	0.21350	0.207700	0.097560	0.252
23	21.160	23.04	137.20	1404.0	0.09428	0.10220	0.109700	0.086320	0.176
24	16.650	21.38	110.00	904.6	0.11210	0.14570	0.152500	0.091700	0.199
25	17.140	16.40	116.00	912.7	0.11860	0.22760	0.222900	0.140100	0.304
26	14.580	21.53	97.41	644.8	0.10540	0.18680	0.142500	0.087830	0.225
27	18.610	20.25	122.10	1094.0	0.09440	0.10660	0.149000	0.077310	0.169
28	15.300	25.27	102.40	732.4	0.10820	0.16970	0.168300	0.087510	0.192
29	17.570	15.05	115.00	955.1	0.09847	0.11570	0.098750	0.079530	0.173
...	...	...	...	...	...	...	...	...	..
539	7.691	25.44	48.34	170.4	0.08668	0.11990	0.092520	0.013640	0.203
540	11.540	14.44	74.65	402.9	0.09984	0.11200	0.067370	0.025940	0.181
541	14.470	24.99	95.81	656.4	0.08837	0.12300	0.100900	0.038900	0.187

	mean radius	mean texture	mean perimter	mean area	mean smoothnesss	mean compactness	mean concavity	mean concave points	mear symetry
542	14.740	25.42	94.70	668.6	0.08275	0.07214	0.041050	0.030270	0.184(
543	13.210	28.06	84.88	538.4	0.08671	0.06877	0.029870	0.032750	0.162{
544	13.870	20.70	89.77	584.8	0.09578	0.10180	0.036880	0.023690	0.162(
545	13.620	23.23	87.19	573.2	0.09246	0.06747	0.029740	0.024430	0.166{
546	10.320	16.35	65.31	324.9	0.09434	0.04994	0.010120	0.005495	0.188[
547	10.260	16.58	65.85	320.8	0.08877	0.08066	0.043580	0.024380	0.166[
548	9.683	19.34	61.05	285.7	0.08491	0.05030	0.023370	0.009615	0.158(
549	10.820	24.21	68.89	361.6	0.08192	0.06602	0.015480	0.008160	0.197{
550	10.860	21.48	68.51	360.5	0.07431	0.04227	0.000000	0.000000	0.166[
551	11.130	22.44	71.49	378.4	0.09566	0.08194	0.048240	0.022570	0.203(
552	12.770	29.43	81.35	507.9	0.08276	0.04234	0.019970	0.014990	0.153[
553	9.333	21.94	59.01	264.0	0.09240	0.05605	0.039960	0.012820	0.169[
554	12.880	28.92	82.50	514.3	0.08123	0.05824	0.061950	0.023430	0.156{
555	10.290	27.61	65.67	321.4	0.09030	0.07658	0.059990	0.027380	0.159[
556	10.160	19.59	64.73	311.7	0.10030	0.07504	0.005025	0.011160	0.179[
557	9.423	27.88	59.26	271.3	0.08123	0.04971	0.000000	0.000000	0.174[
558	14.590	22.68	96.39	657.1	0.08473	0.13300	0.102900	0.037360	0.145{
559	11.510	23.93	74.52	403.5	0.09261	0.10210	0.111200	0.041050	0.138[
560	14.050	27.15	91.38	600.4	0.09929	0.11260	0.044620	0.043040	0.153[
561	11.200	29.37	70.67	386.0	0.07449	0.03558	0.000000	0.000000	0.106(
562	15.220	30.62	103.40	716.9	0.10480	0.20870	0.255000	0.094290	0.212[
563	20.920	25.09	143.00	1347.0	0.10990	0.22360	0.317400	0.147400	0.214[
564	21.560	22.39	142.00	1479.0	0.11100	0.11590	0.243900	0.138900	0.172{
565	20.130	28.25	131.20	1261.0	0.09780	0.10340	0.144000	0.097910	0.175[
566	16.600	28.08	108.30	858.1	0.08455	0.10230	0.092510	0.053020	0.159(
567	20.600	29.33	140.10	1265.0	0.11780	0.27700	0.351400	0.152000	0.239[
568	7.760	24.54	47.92	181.0	0.05263	0.04362	0.000000	0.000000	0.158[

569 rows × 30 columns

Muestras Anómalas

In [7]:
```
#Muestras anómalas
outliers=wdbc.loc[wdbc["label"]==-1]
outliers_index=list(outliers.index)
print(outliers_index)
```

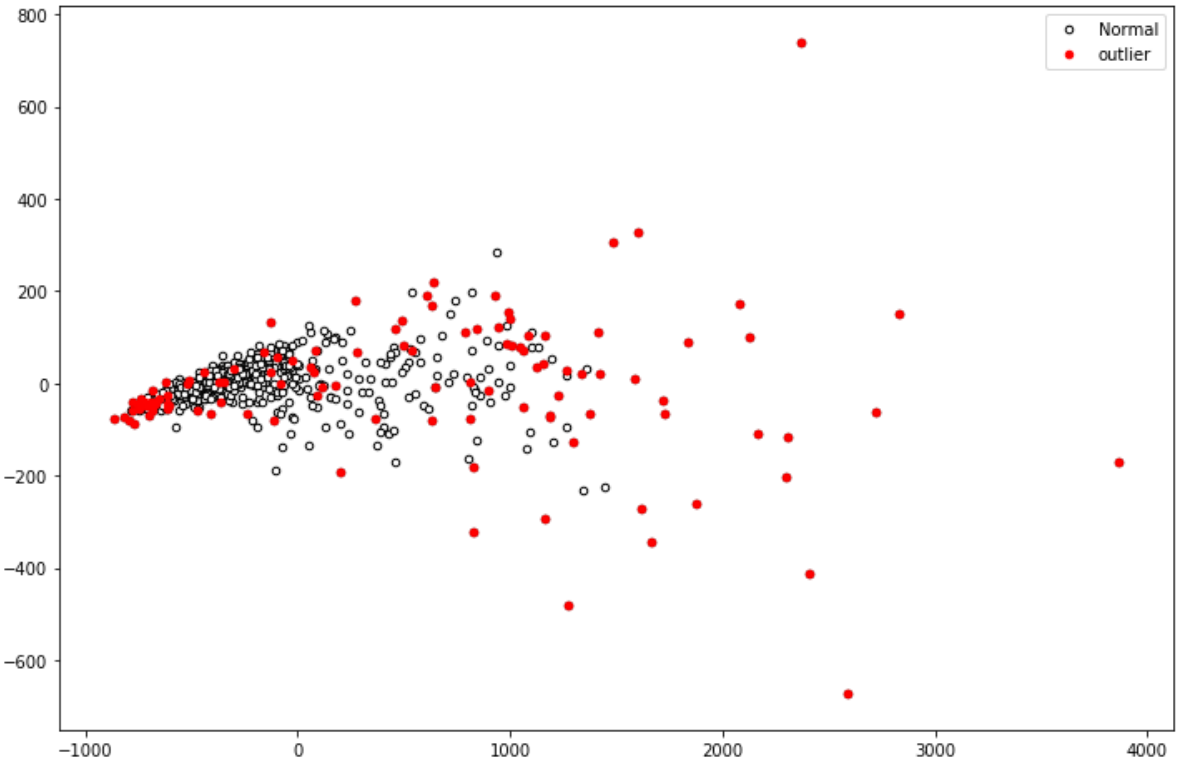
```
[0, 3, 4, 9, 12, 18, 22, 23, 24, 25, 33, 38, 42, 45, 59, 60, 61, 63, 68, 71,
72, 77, 78, 82, 83, 95, 101, 105, 108, 112, 116, 119, 122, 138, 140, 146, 15
1, 152, 161, 164, 175, 176, 178, 180, 181, 190, 192, 197, 202, 203, 210, 212,
213, 214, 219, 236, 237, 250, 252, 256, 257, 258, 259, 260, 265, 272, 288, 29
0, 300, 302, 307, 314, 318, 323, 339, 343, 351, 352, 366, 368, 369, 373, 376,
379, 382, 389, 391, 393, 400, 417, 424, 430, 461, 468, 473, 485, 489, 491, 50
3, 504, 505, 520, 521, 533, 538, 539, 557, 561, 562, 563, 564, 565, 567, 568]
```

In [8]:
```
#Número de muestras normales y muestras anómalas
print(wdbc["label"].value_counts())
```

```
 1    455
-1    114
Name: label, dtype: int64
```

Visualización

In [9]:
```
#Remover la etiqueta
wdbc=wdbc.drop(["label"],axis=1)
#Pasar las variables a dos dimensiones
pca=PCA(2)
pca.fit(wdbc)

#Array tras la transformación con PCA
principalDF=pd.DataFrame(data=pca.transform(wdbc), columns=["principal compone
nt 1", "principal component 2"])

#Valores
principalDF_values=principalDF.iloc[:,:].values

#Visualización
fig = plt.figure(figsize = (12,8))

b1=plt.scatter(principalDF_values[:,0],principalDF_values[:,1], c='white',s=20
,edgecolor='k', label="Normal")
b1=plt.scatter(principalDF_values[outliers_index,0],principalDF_values[outlier
s_index,1], s=20, c="red", label="outlier")
plt.legend(loc="upper right")

plt.show()
```



Comprobación resultados

In [10]:
```
wdbccomp = pd.read_csv(
    'wdbc.data',
    encoding='utf-8',
    delim_whitespace=False,
    names=["id","diagnostic",'mean radius', 'mean texture', 'mean perimter',
'mean area', 'mean smoothnesss', 'mean compactness', 'mean concavity', 'mean c
oncave points', 'mean symetry','mean fractal dimension','se radius', 'se textu
re', 'se perimter', 'se area', 'se smoothnesss', 'se compactness', 'se concavi
ty', 'se concave points', 'se symetry','se fractal dimension''worst radius',
'worst texture', 'worst perimter', 'worst area', 'worst smoothnesss', 'worst c
ompactness', 'worst concavity', 'worst concave points', 'worst symetry','worst
fractal dimension'],
    index_col=False,
    header=None,
)
wdbccomp["label"]=predwdbc
wdbccomp["label"][wdbccomp["label"].values==-1]="B"
wdbccomp["label"][wdbccomp["label"].values==1]="M"
wdbc_comparation=wdbccomp[["diagnostic","label"]]
wdbc_comparation
```

```
C:\Users\FedericoDamianEsteba\Anaconda3\lib\site-packages\ipykernel_launcher.
py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.
```

Out[10]:

	diagnostic	label
0	M	B
1	M	M
2	M	M
3	M	B
4	M	B
5	M	M
6	M	M
7	M	M
8	M	M
9	M	B
10	M	M
11	M	M
12	M	B
13	M	M
14	M	M
15	M	M
16	M	M
17	M	M
18	M	B
19	B	M
20	B	M
21	B	M
22	M	B
23	M	B
24	M	B
25	M	B
26	M	M
27	M	M
28	M	M
29	M	M
...	...	...
539	B	B
540	B	M
541	B	M
542	B	M

	diagnostic	label
543	B	M
544	B	M
545	B	M
546	B	M
547	B	M
548	B	M
549	B	M
550	B	M
551	B	M
552	B	M
553	B	M
554	B	M
555	B	M
556	B	M
557	B	B
558	B	M
559	B	M
560	B	M
561	B	B
562	M	B
563	M	B
564	M	B
565	M	B
566	M	M
567	M	B
568	B	B

569 rows × 2 columns

In [11]:
```
print(metrics.classification_report(wdbccomp["diagnostic"],wdbccomp["label"]))
print(metrics.confusion_matrix(wdbccomp["diagnostic"],wdbccomp["label"]),"\n")
```

```
              precision    recall  f1-score   support

           B       0.32      0.10      0.15       357
           M       0.29      0.63      0.40       212

   micro avg       0.30      0.30      0.30       569
   macro avg       0.31      0.37      0.28       569
weighted avg       0.31      0.30      0.25       569

[[ 36 321]
 [ 78 134]]
```

SVM

In [12]:
```
ocsvm=OneClassSVM(nu=0.25,gamma=0.05)
ocsvm.fit(wdbc)
predocsvm=ocsvm.predict(wdbc)
```

In [13]:
```
wdbc["labelsvm"]=predocsvm
display(wdbc)
#Muestras anómalas
outlierssvm=wdbc.loc[wdbc["labelsvm"]==-1]
outlierssvm_index=list(outlierssvm.index)
print(outlierssvm_index)

#Número de muestras normales y muestras anómalas
print(wdbc["labelsvm"].value_counts())
```

	mean radius	mean texture	mean perimter	mean area	mean smoothnesss	mean compactness	mean concavity	mean concave points	mear symetry
0	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300100	0.147100	0.241
1	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900	0.070170	0.181
2	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197400	0.127900	0.206
3	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241400	0.105200	0.259
4	20.290	14.34	135.10	1297.0	0.10030	0.13280	0.198000	0.104300	0.180
5	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157800	0.080890	0.208
6	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112700	0.074000	0.179
7	13.710	20.83	90.20	577.9	0.11890	0.16450	0.093660	0.059850	0.219
8	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185900	0.093530	0.235
9	12.460	24.04	83.97	475.9	0.11860	0.23960	0.227300	0.085430	0.203
10	16.020	23.24	102.70	797.8	0.08206	0.06669	0.032990	0.033230	0.152
11	15.780	17.89	103.60	781.0	0.09710	0.12920	0.099540	0.066060	0.184
12	19.170	24.80	132.40	1123.0	0.09740	0.24580	0.206500	0.111800	0.239
13	15.850	23.95	103.70	782.7	0.08401	0.10020	0.099380	0.053640	0.184
14	13.730	22.61	93.60	578.3	0.11310	0.22930	0.212800	0.080250	0.206
15	14.540	27.54	96.73	658.8	0.11390	0.15950	0.163900	0.073640	0.230
16	14.680	20.13	94.74	684.5	0.09867	0.07200	0.073950	0.052590	0.158
17	16.130	20.68	108.10	798.8	0.11700	0.20220	0.172200	0.102800	0.216
18	19.810	22.15	130.00	1260.0	0.09831	0.10270	0.147900	0.094980	0.158
19	13.540	14.36	87.46	566.3	0.09779	0.08129	0.066640	0.047810	0.188
20	13.080	15.71	85.63	520.0	0.10750	0.12700	0.045680	0.031100	0.196
21	9.504	12.44	60.34	273.9	0.10240	0.06492	0.029560	0.020760	0.181
22	15.340	14.26	102.50	704.4	0.10730	0.21350	0.207700	0.097560	0.252
23	21.160	23.04	137.20	1404.0	0.09428	0.10220	0.109700	0.086320	0.176
24	16.650	21.38	110.00	904.6	0.11210	0.14570	0.152500	0.091700	0.199
25	17.140	16.40	116.00	912.7	0.11860	0.22760	0.222900	0.140100	0.304
26	14.580	21.53	97.41	644.8	0.10540	0.18680	0.142500	0.087830	0.225
27	18.610	20.25	122.10	1094.0	0.09440	0.10660	0.149000	0.077310	0.169
28	15.300	25.27	102.40	732.4	0.10820	0.16970	0.168300	0.087510	0.192
29	17.570	15.05	115.00	955.1	0.09847	0.11570	0.098750	0.079530	0.173
...	...	...	...	...	...	...	...	...	..
539	7.691	25.44	48.34	170.4	0.08668	0.11990	0.092520	0.013640	0.203
540	11.540	14.44	74.65	402.9	0.09984	0.11200	0.067370	0.025940	0.181
541	14.470	24.99	95.81	656.4	0.08837	0.12300	0.100900	0.038900	0.187

	mean radius	mean texture	mean perimter	mean area	mean smoothnesss	mean compactness	mean concavity	mean concave points	mear symetry
542	14.740	25.42	94.70	668.6	0.08275	0.07214	0.041050	0.030270	0.184(
543	13.210	28.06	84.88	538.4	0.08671	0.06877	0.029870	0.032750	0.162{
544	13.870	20.70	89.77	584.8	0.09578	0.10180	0.036880	0.023690	0.162(
545	13.620	23.23	87.19	573.2	0.09246	0.06747	0.029740	0.024430	0.166-
546	10.320	16.35	65.31	324.9	0.09434	0.04994	0.010120	0.005495	0.188[
547	10.260	16.58	65.85	320.8	0.08877	0.08066	0.043580	0.024380	0.166[
548	9.683	19.34	61.05	285.7	0.08491	0.05030	0.023370	0.009615	0.158(
549	10.820	24.21	68.89	361.6	0.08192	0.06602	0.015480	0.008160	0.197(
550	10.860	21.48	68.51	360.5	0.07431	0.04227	0.000000	0.000000	0.166-
551	11.130	22.44	71.49	378.4	0.09566	0.08194	0.048240	0.022570	0.203(
552	12.770	29.43	81.35	507.9	0.08276	0.04234	0.019970	0.014990	0.153[
553	9.333	21.94	59.01	264.0	0.09240	0.05605	0.039960	0.012820	0.169-
554	12.880	28.92	82.50	514.3	0.08123	0.05824	0.061950	0.023430	0.156(
555	10.290	27.61	65.67	321.4	0.09030	0.07658	0.059990	0.027380	0.159-
556	10.160	19.59	64.73	311.7	0.10030	0.07504	0.005025	0.011160	0.179-
557	9.423	27.88	59.26	271.3	0.08123	0.04971	0.000000	0.000000	0.174-
558	14.590	22.68	96.39	657.1	0.08473	0.13300	0.102900	0.037360	0.145-
559	11.510	23.93	74.52	403.5	0.09261	0.10210	0.111200	0.041050	0.138[
560	14.050	27.15	91.38	600.4	0.09929	0.11260	0.044620	0.043040	0.153-
561	11.200	29.37	70.67	386.0	0.07449	0.03558	0.000000	0.000000	0.106(
562	15.220	30.62	103.40	716.9	0.10480	0.20870	0.255000	0.094290	0.212[
563	20.920	25.09	143.00	1347.0	0.10990	0.22360	0.317400	0.147400	0.214[
564	21.560	22.39	142.00	1479.0	0.11100	0.11590	0.243900	0.138900	0.172(
565	20.130	28.25	131.20	1261.0	0.09780	0.10340	0.144000	0.097910	0.175-
566	16.600	28.08	108.30	858.1	0.08455	0.10230	0.092510	0.053020	0.159(
567	20.600	29.33	140.10	1265.0	0.11780	0.27700	0.351400	0.152000	0.239-
568	7.760	24.54	47.92	181.0	0.05263	0.04362	0.000000	0.000000	0.158-

569 rows × 30 columns

```
[0, 3, 11, 12, 13, 14, 15, 17, 20, 23, 24, 25, 26, 29, 31, 32, 33, 34, 35, 3
6, 37, 38, 40, 42, 43, 44, 46, 47, 48, 49, 50, 51, 53, 56, 57, 58, 59, 60, 6
2, 63, 64, 65, 68, 69, 70, 74, 80, 81, 85, 89, 90, 92, 93, 96, 103, 105, 106,
109, 110, 112, 113, 114, 116, 117, 118, 120, 125, 126, 129, 130, 131, 132, 13
4, 136, 138, 142, 143, 145, 148, 150, 152, 153, 156, 157, 158, 159, 160, 161,
162, 164, 165, 167, 168, 171, 172, 174, 175, 177, 178, 183, 184, 186, 187, 19
0, 191, 193, 194, 197, 198, 199, 200, 201, 202, 203, 205, 207, 208, 209, 212,
218, 219, 220, 222, 223, 224, 226, 228, 229, 230, 232, 233, 238, 239, 240, 24
1, 242, 245, 246, 247, 251, 253, 258, 260, 261, 262, 263, 264, 265, 269, 271,
273, 274, 277, 279, 281, 285, 286, 287, 289, 291, 292, 295, 297, 298, 299, 30
5, 306, 308, 309, 310, 311, 312, 313, 318, 320, 322, 327, 330, 331, 332, 336,
338, 339, 341, 343, 346, 350, 355, 357, 358, 359, 360, 361, 362, 363, 364, 36
6, 367, 374, 376, 377, 378, 382, 385, 386, 394, 395, 398, 401, 402, 403, 404,
405, 407, 409, 410, 411, 413, 415, 417, 421, 425, 429, 431, 432, 436, 437, 43
9, 441, 443, 444, 445, 446, 449, 450, 452, 453, 454, 455, 458, 459, 460, 462,
463, 464, 467, 471, 475, 476, 479, 480, 483, 486, 488, 494, 495, 496, 502, 50
3, 508, 510, 513, 515, 516, 517, 521, 522, 525, 526, 533, 535, 536, 538, 545,
546, 548, 549, 550, 551, 552, 553, 554, 555, 559, 561, 563, 564]
-1    287
 1    282
Name: labelsvm, dtype: int64
```

In [14]:
```
wdbccomp2 = pd.read_csv(
    'wdbc.data',
    encoding='utf-8',
    delim_whitespace=False,
    names=["id","diagnostic",'mean radius', 'mean texture', 'mean perimter',
'mean area', 'mean smoothnesss', 'mean compactness', 'mean concavity', 'mean c
oncave points', 'mean symetry','mean fractal dimension','se radius', 'se textu
re', 'se perimter', 'se area', 'se smoothnesss', 'se compactness', 'se concavi
ty', 'se concave points', 'se symetry','se fractal dimension''worst radius',
'worst texture', 'worst perimter', 'worst area', 'worst smoothnesss', 'worst c
ompactness', 'worst concavity', 'worst concave points', 'worst symetry','worst
fractal dimension'],
    index_col=False,
    header=None,
)
wdbccomp2["labelsvm"]=predocsvm
#for row in wdbccomp2.index:
#    if (wdbccomp2["labelsvm"])==-1:
#        wdbccomp2["labelsvm"]=["B"]
#    if (wdbccomp2["labelsvm"])==1:
#        wdbccomp2["labelsvm"]=["M"]


#for row in range(len(wdbccomp2)):
#    if wdbccomp2["labelsvm"][row]==-1:
#        wdbccomp2['labelsvm'][row]="B"
#    elif wdbccomp2["labelsvm"][row]==1:
#        wdbccomp2['labelsvm'][row]="M"


wdbccomp2["labelsvm"][wdbccomp2["labelsvm"]==-1]="B"
wdbccomp2["labelsvm"][wdbccomp2["labelsvm"]==1]="M"
wdbc_comparation2=wdbccomp2[["diagnostic","labelsvm"]]
display(wdbc_comparation2)
```

```
C:\Users\FedericoDamianEsteba\Anaconda3\lib\site-packages\ipykernel_launcher.
py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
C:\Users\FedericoDamianEsteba\Anaconda3\lib\site-packages\ipykernel_launcher.
py:25: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
```

	diagnostic	labelsvm
0	M	B
1	M	M
2	M	M
3	M	B
4	M	M
5	M	M
6	M	M
7	M	M
8	M	M
9	M	M
10	M	M
11	M	B
12	M	B
13	M	B
14	M	B
15	M	B
16	M	M
17	M	B
18	M	M
19	B	M
20	B	B
21	B	M
22	M	M
23	M	B
24	M	B
25	M	B
26	M	B
27	M	M
28	M	M
29	M	B
...	...	...
539	B	M
540	B	M
541	B	M
542	B	M

	diagnostic	labelsvm
543	B	M
544	B	M
545	B	B
546	B	B
547	B	M
548	B	B
549	B	B
550	B	B
551	B	B
552	B	B
553	B	B
554	B	B
555	B	B
556	B	M
557	B	M
558	B	M
559	B	B
560	B	M
561	B	B
562	M	M
563	M	B
564	M	B
565	M	M
566	M	M
567	M	M
568	B	M

569 rows × 2 columns

In [15]:
```
print(metrics.classification_report(wdbccomp2["diagnostic"],wdbccomp2["labelsvm"]))
print(metrics.confusion_matrix(wdbccomp2["diagnostic"],wdbccomp2["labelsvm"]),"\n")
```

```
              precision    recall  f1-score   support

           B       0.63      0.51      0.57       357
           M       0.38      0.50      0.43       212

   micro avg       0.51      0.51      0.51       569
   macro avg       0.51      0.51      0.50       569
weighted avg       0.54      0.51      0.52       569

[[182 175]
 [105 107]]
```

Test DBSCAN

In [16]:
```
wdbc=wdbc.drop(["labelsvm"],axis=1)
#display(wdbc)
scaler = MinMaxScaler()
wdbcscal=scaler.fit_transform(wdbc)
#display(wdbcscal)
modeldbscan=DBSCAN(eps=0.6, min_samples = 100).fit(wdbcscal)
print(modeldbscan)
outliers_dbscan=pd.DataFrame(wdbcscal)
clusters = modeldbscan.fit_predict(wdbcscal)


from collections import Counter
print (Counter(modeldbscan.labels_))
```

```
DBSCAN(algorithm='auto', eps=0.6, leaf_size=30, metric='euclidean',
    metric_params=None, min_samples=100, n_jobs=None, p=None)
Counter({0: 460, -1: 109})
```

Conlcusion

Hay diferentes tipos de métodos para separar muestras anómalas basaados en agrupamientos y denssidades como DBSCAN. Pero también otros como Isolation Forest y OneClassSVM derivado de las técnicas de árboles y SVM.

Es todavía un desafío saber que algoritmo es el más óptimo para detectar los distintos tipos de anomalías.