In [1]:

```
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectKBest
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV, RandomizedSe
archCV
from sklearn.ensemble import AdaBoostClassifier,RandomForestCl
assifier
from sklearn.feature_selection import f_regression, chi2
from sklearn.feature_selection import mutual_info_regression
from scipy.stats import randint as sp_randint
from sklearn import preprocessing
from sklearn.metrics import log_loss,accuracy_score,precision_
score,recall_score
from sklearn.feature_extraction import FeatureHasher
from sklearn.svm import SVC
```

In [2]:

```
columns_name=['class','age','menopause','tumor-size','inv-node
s','node-caps','deg-malig','breast','breast-quad','irradiat']
```

In [3]:

```
data_df=pd.read_csv('breast-cancer.data',names=columns_name,he
ader=None)
```

In [4]:

```
data_df.shape
```

Out[4]:

```
(286, 10)
```

In [5]:

```
data1=data_df['node-caps'] != '?'
print(data1.sum())
```

```
278
```

In [6]:

```
#data_df=pd.to_numeric(data_df['node-caps'])
data_df = data_df[data_df['node-caps'] != '?']
data_df = data_df[data_df['breast-quad'] != '?']
```

In [7]:

```
data_df.reset_index(drop=True,inplace=True)
```

In [8]:

```
data_df.shape
```

Out[8]:

```
(277, 10)
```

In [9]:

```
data_df.describe()
```

Out[9]:

	deg-malig
count	277.000000
mean	2.057762
std	0.729989
min	1.000000
25%	2.000000
50%	2.000000
75%	3.000000
max	3.000000

In [10]:

```
data_df.dtypes
```

Out[10]:

```
class          object
age            object
menopause      object
tumor-size     object
inv-nodes      object
node-caps      object
deg-malig       int64
breast         object
breast-quad    object
irradiat       object
dtype: object
```

In [11]:

```
labels=np.array(data_df['class'])
X=data_df.ix[:,'age':]
```

```
/anaconda3/envs/fm1/lib/python3.7/site-packages/ip
ykernel_launcher.py:2: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/indexi
ng.html#ix-indexer-is-deprecated
```

In [12]:

```
labels=data_df['class']
```

In [13]:

```
RSEED=50
train, test, train_labels, test_labels = train_test_split(X,
                                                          labels,
                                                          stratify = labels,
                                                          test_size = 0.3,
                                                          random_state = RSEED)
```

In [ ]:

```
```

In [14]:

```
X_train_hash = pd.DataFrame.copy(train)
X_test_hash = pd.DataFrame.copy(test)
```

In [15]:

```
X_train_hash.head()
```

Out[15]:

	age	menopause	tumor-size	inv-nodes	node-caps	deg-malig	breast	breast quad
221	50-59	premeno	30-34	0-2	no	3	right	left_up
153	30-39	lt40	15-19	0-2	no	3	right	left_up
84	40-49	premeno	20-24	0-2	no	1	left	right_low
240	50-59	ge40	30-34	9-11	yes	3	left	right_low
183	40-49	premeno	10-14	0-2	no	2	right	left_up

In [16]:

```
h = FeatureHasher(n_features=100,input_type="string")
```

In [17]:

```
for i in range(X_train_hash.shape[1]):
    X_train_hash.iloc[:,i]=X_train_hash.iloc[:,i].astype('str'
)
for i in range(X_test_hash.shape[1]):
    X_test_hash.iloc[:,i]=X_test_hash.iloc[:,i].astype('str')
```

In [18]:

```
X_train_hash = h.transform(X_train_hash.values)
X_test_hash = h.transform(X_test_hash.values)
```

In [19]:

```
#print(X_train_hash.head())
```

In [20]:

```
r = RandomForestClassifier(n_estimators=100000,max_depth=20)
```

In [21]:

```
r.fit(X_train_hash,train_labels)
```

Out[21]:

```
RandomForestClassifier(bootstrap=True, class_weigh
t=None, criterion='gini',
                       max_depth=20, max_features=
'auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0,
min_impurity_split=None,
                       min_samples_leaf=1, min_sam
ples_split=2,
                       min_weight_fraction_leaf=0.
0, n_estimators=100000,
                       n_jobs=None, oob_score=Fals
e, random_state=None,
                       verbose=0, warm_start=False
)
```

In [ ]:

```

```

In [24]:

```
y_pred = r.predict_proba(X_test_hash)
```

In [25]:

```
print(log_loss(test_labels,y_pred))
```

```
0.5034557445251281
```

In [ ]:

```

```

In [26]:

```
y_pred1 = r.predict(X_test_hash)
print(accuracy_score(test_labels,y_pred1))
#print(recall_score(test_labels,y_pred1))
print(precision_score(test_labels,y_pred1,pos_label="no-recurr
ence-events"))
```

```
0.75
0.7794117647058824
```

In [27]:

```
print(accuracy_score(test_labels,y_pred1))
#print(recall_score(test_labels,y_pred1))
print(precision_score(test_labels,y_pred1,pos_label="no-recurr
ence-events"))
print(precision_score(test_labels,y_pred1,pos_label="recurrenc
e-events"))
```

```
0.75
0.7794117647058824
0.625
```

In [28]:

```
svc=SVC(probability=True, kernel='sigmoid')
```

In [29]:

```
adabc =AdaBoostClassifier(n_estimators=1000, base_estimator=sv
c,learning_rate=1)
```

In [30]:

```
adabc.fit(X_train_hash,train_labels)
```

```
/anaconda3/envs/fm1/lib/python3.7/site-packages/sk
learn/svm/base.py:193: FutureWarning: The default
value of gamma will change from 'auto' to 'scale'
in version 0.22 to account better for unscaled fea
tures. Set gamma explicitly to 'auto' or 'scale' t
o avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[30]:

```
AdaBoostClassifier(algorithm='SAMME.R',
                   base_estimator=SVC(C=1.0, cache
_size=200, class_weight=None,
                                      coef0=0.0, d
ecision_function_shape='ovr',
                                      degree=3, ga
mma='auto_deprecated',
                                      kernel='sigm
oid', max_iter=-1,
                                      probability=
True, random_state=None,
                                      shrinking=Tr
ue, tol=0.001,
                                      verbose=Fals
e),
                   learning_rate=1, n_estimators=1
000, random_state=None)
```

In [31]:

```
y_pred_adab = adabc.predict(X_test_hash)
```

In [32]:

```
print(accuracy_score(test_labels,y_pred_adab))
#print(recall_score(test_labels,y_pred1))
print(precision_score(test_labels,y_pred_adab,pos_label="no-re
currence-events"))
```

```
0.7023809523809523
0.7023809523809523
```

In [ ]: