Comencem a familiaritzar-nos amb Pipelines, grid search i text mining !!! Comencem amb uns quants exercicis bàsics

Nivell 1

- Exercici 1:

Agafa el conjunt de dades que vulguis i realitza un pipeline i un gridsearch aplicant l'algorisme de Random Forest.

In [23]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform as sp_rand
import matplotlib.pyplot as plt
```

In [24]:
```
df = pd.read_csv('pima-indians-diabetes.csv', header = None)
```

In [25]:
```
df
```

Out[25]:

	0	1	2	3	4	5	6	7	8
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [26]:
```
# asignación de nombres a las columnas
df.columns = ['Pregnancies', 'Glucose' , 'Blood_Preassure', 'Skin_Thickness',
              'Insuline', 'BMI', 'Diabetes_Pedigree_Function', 'Age', 'Class']
```

In [27]:
```
# al importar el DF, la columna Diabetes_pedigree_function los valores se han
# importado como decimales. El DF Original eran centenas. Se corrige.
df['Diabetes_Pedigree_Function'] = df['Diabetes_Pedigree_Function'].apply(lambda x: x*1000)
```

In [28]:
```
df.head()
```

Out[28]:

	Pregnancies	Glucose	Blood_Preassure	Skin_Thickness	Insuline	BMI	Diabetes_Pedigree_Function	Age	Class
0	6	148	72	35	0	33.6	627.0	50	1
1	1	85	66	29	0	26.6	351.0	31	0
2	8	183	64	0	0	23.3	672.0	32	1
3	1	89	66	23	94	28.1	167.0	21	0
4	0	137	40	35	168	43.1	2288.0	33	1

Pipeline

In [29]:
```
feature_cols = ['Pregnancies', 'Glucose' , 'Blood_Preassure', 'Skin_Thickness',
                'Insuline', 'BMI', 'Diabetes_Pedigree_Function', 'Age']
```

In [30]:
```
x = df[feature_cols]
y = df['Class']
```

In [31]:
```
# estandarización de vaores con la función MinMaxScaler
df = pd.DataFrame(MinMaxScaler(df.values), columns=df.columns, index=df.index)
```

In [32]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size= 0.25, random_state = 42 )
```

In [33]:
```
# Creación del modelo modificando PCA y la profundidad de Random Forest
Random_Forest_pipeline = Pipeline([('my_PCA', PCA(n_components = 3)),
                                   ('logistic_classifier', RandomForestRegressor(max_depth=10))])
```

In [34]:
```
Random_Forest_pipeline.fit(X_train, y_train)

# predict target values on the training data
Random_Forest_pipeline.predict(X_train)
```

Out[34]:
```
array([7.31578561e-01, 1.71656701e-01, 1.62244898e-02, 2.40147059e-02,
       9.98433856e-02, 0.00000000e+00, 6.83586111e-01, 8.39043504e-01,
       4.51867945e-02, 0.00000000e+00, 4.26223317e-01, 1.13026316e-01,
       2.99764348e-01, 3.08695652e-02, 0.00000000e+00, 1.47893205e-01,
       9.20000000e-01, 8.55190476e-01, 2.15481275e-01, 1.44690114e-01,
       7.98622393e-02, 3.08902156e-01, 6.39730133e-01, 6.71104604e-02,
       2.82355311e-01, 7.11299359e-01, 1.74934699e-01, 2.61428571e-01,
       8.80000000e-01, 4.68695652e-02, 1.59777778e-01, 8.88721897e-02,
       2.62272727e-02, 1.48388546e-01, 5.51745304e-01, 6.16033622e-01,
       2.30545678e-01, 1.00507127e-02, 0.00000000e+00, 0.00000000e+00,
       3.23109244e-02, 6.01791711e-01, 1.01339172e-01, 0.00000000e+00,
       3.00000000e-02, 2.11093419e-01, 6.00290346e-01, 7.48416394e-01,
       2.93777335e-01, 2.08525103e-02, 6.03674819e-01, 2.83909956e-01,
       2.81100679e-02, 2.64740637e-01, 2.47477719e-01, 2.97777778e-02,
       2.84535223e-01, 8.68718487e-02, 6.80882353e-01, 8.00000000e-02,
       7.81000000e-01, 2.58706816e-01, 2.05282217e-01, 2.03684839e-01,
       4.47731081e-02, 7.58112358e-01, 1.63640432e-02, 9.50712260e-02,
       2.51363880e-01, 8.21052535e-01, 1.21560928e-02, 2.70086010e-02,
       1.49840026e-01, 1.46639708e-01, 2.31092437e-03, 2.85714286e-03,
       8.23309084e-03, 2.44187254e-01, 5.60606061e-04, 6.40952381e-02,
       6.80309941e-01, 8.71428571e-01, 0.00000000e+00, 3.25283011e-01,
       1.54021378e-01, 7.75800420e-01, 3.53627618e-02, 2.18338799e-01,
       3.38738059e-02, 9.60000000e-01, 9.80000000e-01, 1.00000000e+00,
       2.55780509e-01, 2.86101879e-01, 9.40000000e-01, 8.56666667e-01,
       8.35665166e-02, 8.17692308e-02, 5.29545455e-02, 1.02272727e-02,
       6.71328148e-01, 3.35362319e-02, 9.10604066e-01, 6.24313368e-02,
       1.16753954e-01, 9.90000000e-01, 6.00431603e-01, 1.26948124e-01,
       2.17498943e-01, 7.11492947e-01, 9.60000000e-01, 4.88859478e-02,
       1.43640128e-01, 8.00581812e-03, 1.80416667e-01, 2.67863019e-01,
       2.64517625e-02, 7.16250928e-01, 5.76133508e-02, 7.90000000e-01,
       1.00000000e+00, 0.00000000e+00, 7.75787174e-02, 1.44519352e-01,
       8.29704656e-01, 1.28011597e-01, 1.40000000e-03, 5.19821098e-02,
       5.23979272e-01, 6.79219414e-01, 7.90000000e-01, 6.47142278e-02,
       6.92362858e-01, 1.63305281e-01, 1.28571429e-02, 9.22500000e-02,
       1.97325664e-01, 9.50000000e-02, 1.25000000e-03, 1.14285714e-02,
       0.00000000e+00, 7.90882353e-01, 6.82210217e-01, 1.00507127e-02,
       1.00000000e+00, 7.69473364e-01, 4.52658371e-03, 3.03333333e-02,
       6.91250000e-01, 3.99117928e-02, 4.39771979e-02, 1.00000000e+00,
       8.60986749e-01, 8.00000000e-01, 0.00000000e+00, 0.00000000e+00,
       8.47619048e-02, 2.74310304e-01, 1.72810349e-01, 8.52968556e-01,
       2.37813897e-01, 1.00000000e+00, 2.08374777e-01, 5.09772727e-02,
       3.40112773e-01, 4.23129286e-02, 1.67443732e-01, 2.08596042e-01,
       2.74208171e-01, 9.90000000e-01, 8.90000000e-01, 2.37858128e-01,
       2.25507127e-02, 4.35940028e-02, 1.22415671e-01, 8.23309084e-03,
       1.47846846e-01, 4.72413147e-01, 6.00416667e-01, 2.22952381e-01,
       1.00000000e+00, 7.54809741e-01, 7.53024778e-01, 8.35501264e-02,
       1.68130838e-01, 2.91444387e-01, 8.82352941e-04, 3.05641040e-01,
       6.04275380e-01, 9.90000000e-01, 9.10000000e-01, 9.20405106e-01,
       1.77710693e-01, 1.25790666e-01, 8.65469081e-01, 2.26666667e-01,
       8.23309084e-03, 7.76218438e-01, 5.11391559e-01, 1.00000000e-03,
       7.50989770e-01, 9.63333333e-01, 2.48500000e-01, 9.90000000e-01,
       7.24295924e-02, 2.40000000e-01, 7.44528721e-02, 6.38341115e-01,
       6.36172208e-02, 6.05321334e-01, 0.00000000e+00, 7.39361578e-01,
       4.45434207e-02, 4.12915541e-01, 2.54461008e-01, 9.60000000e-01,
       8.11102322e-01, 1.40227052e-01, 5.46428672e-01, 7.27619048e-02,
       2.14894526e-01, 7.86579428e-01, 2.33739496e-01, 8.93944444e-01,
       2.10168254e-01, 1.67177504e-01, 9.90000000e-01, 7.01115808e-02,
       3.19836080e-02, 6.85268444e-01, 1.18678264e-01, 4.76750458e-02,
       6.66591934e-01, 2.09343258e-01, 9.80000000e-01, 8.55310924e-01,
       1.00677477e-01, 4.60952381e-02, 5.61679376e-02, 2.88737231e-01,
       6.77515697e-01, 9.70000000e-01, 1.14122420e-01, 5.78013059e-01,
       5.71797256e-02, 1.10000000e-02, 7.50290491e-01, 8.23309084e-03,
       8.44062285e-01, 1.63333333e-01, 1.00000000e+00, 6.00000000e-02,
       6.73580022e-02, 2.10174030e-01, 2.01207126e-01, 8.00818936e-01,
       5.27878709e-01, 7.71628080e-01, 5.75445701e-01, 5.65829576e-01,
       9.93643461e-02, 3.90000000e-01, 3.33333333e-04, 2.78534605e-01,
       9.20000000e-01, 1.00000000e+00, 4.01470588e-03, 1.33914266e-01,
       1.00000000e-03, 7.71883796e-01, 1.40391972e-01, 6.26428106e-01,
       1.15866840e-01, 1.33428571e-01, 0.00000000e+00, 1.64260269e-01,
       2.12036299e-01, 9.01666667e-01, 9.10000000e-01, 2.70027417e-01,
       8.16960784e-01, 7.01229852e-01, 7.48707161e-01, 5.64712328e-01,
       8.53265873e-01, 1.00507127e-02, 1.42857143e-03, 3.47619048e-02,
       1.00000000e-02, 1.36323232e-02, 6.85861761e-01, 9.02782681e-01,
       1.00000000e-02, 9.40000000e-01, 9.40000000e-01, 1.00000000e-03,
       2.84264998e-01, 8.53333333e-01, 0.00000000e+00, 7.67666667e-01,
       1.00000000e-02, 7.17692308e-02, 3.72331656e-01, 6.28459596e-02,
       3.13472666e-01, 4.03512502e-02, 6.29456751e-01, 9.97893748e-02,
       9.80000000e-01, 2.37394958e-02, 1.99509724e-01, 3.78157926e-01,
       7.86954248e-01, 6.05335406e-02, 5.13707180e-01, 1.00000000e+00,
       8.33184885e-01, 5.83245697e-02, 2.56593419e-01, 2.27272727e-02,
       2.59213545e-01, 1.15683503e-01, 1.98463923e-02, 2.21142857e-02,
       4.58738059e-02, 1.00000000e-02, 8.82352941e-04, 1.77877498e-01,
       6.63843698e-01, 1.60786874e-01, 1.75443277e-01, 6.08385606e-02,
       8.32380952e-01, 9.44029988e-02, 6.62197802e-01, 0.00000000e+00,
       2.40147059e-02, 6.86094035e-01, 1.40000000e-03, 9.73333333e-01,
       1.97321463e-01, 1.63640432e-02, 5.00364410e-02, 9.40000000e-01,
       1.14285714e-02, 1.02272727e-02, 7.20000000e-01, 0.00000000e+00,
       1.92057904e-01, 8.37394958e-02, 2.27272727e-04, 5.55586692e-01,
       1.21560928e-02, 2.03515785e-02, 2.71833543e-01, 5.19836080e-02,
       1.22497622e-01, 7.36774653e-01, 1.94884611e-01, 2.24320138e-01,
       3.19906753e-01, 2.00208618e-01, 1.59753192e-01, 8.00000000e-02,
       2.72727273e-03, 4.00000000e-02, 5.97268542e-01, 9.02327594e-01,
       0.00000000e+00, 1.82213289e-03, 2.34333333e-01, 8.90747253e-01,
       2.89231284e-02, 1.00000000e-02, 5.92123212e-01, 5.73109244e-02,
       6.59529316e-01, 1.55889356e-01, 7.39321564e-01, 5.93964115e-01,
       2.18323232e-02, 1.63640432e-02, 2.11302716e-01, 7.34641148e-02,
       2.98777524e-01, 8.69583333e-01, 8.12125432e-01, 5.55457097e-01,
       7.25927466e-01, 8.38738059e-02, 5.97643098e-04, 1.18221329e-02,
       7.57471428e-01, 5.26930639e-01, 1.00000000e+00, 1.00000000e+00,
       2.27272727e-04, 6.40021142e-01, 2.36223790e-01, 3.60875297e-01,
       1.98791985e-01, 8.52239619e-02, 0.00000000e+00, 2.78420956e-01,
       0.00000000e+00, 0.00000000e+00, 3.19728909e-01, 0.00000000e+00,
       2.16899769e-01, 5.03109244e-02, 9.90000000e-01, 2.30724150e-01,
       2.02761425e-01, 0.00000000e+00, 3.19509158e-01, 2.35542429e-01,
       1.20000000e-02, 1.32842186e-01, 1.00123644e-01, 2.62272727e-02,
       5.82711817e-01, 2.92480690e-01, 1.92907063e-01, 4.63640432e-02,
       2.79676883e-01, 2.72801976e-01, 6.15516699e-01, 5.55544026e-02,
       4.67667744e-02, 4.61335313e-02, 6.27633015e-01, 2.39599355e-01,
       6.74479036e-01, 8.33190476e-01, 1.11831877e-01, 1.26011905e-01,
       6.94540149e-01, 6.04129934e-01, 3.31428571e-01, 1.34980396e-01,
       8.01861729e-01, 1.83401506e-01, 4.38181818e-02, 5.97643098e-04,
       1.17975192e-01, 2.23150482e-01, 1.00000000e-02, 9.50000000e-01,
       9.60000000e-01, 3.97759104e-03, 6.71651812e-01, 2.18151552e-01,
       6.94116652e-02, 7.92923341e-01, 8.51428571e-01, 1.67981848e-01,
       8.45825397e-01, 1.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       7.31377688e-01, 7.53333333e-01, 3.33333333e-04, 0.00000000e+00,
       2.33024186e-01, 1.08695652e-02, 3.55357143e-02, 6.48490141e-01,
       7.79315162e-01, 3.70000000e-01, 9.70000000e-01, 0.00000000e+00,
       5.60606061e-04, 1.00000000e+00, 7.20376472e-01, 9.00000000e-01,
       3.63774613e-01, 3.33333333e-04, 1.57639842e-01, 1.00000000e+00,
       5.77127913e-01, 1.00000000e-02, 1.00000000e-02, 6.12443056e-01,
       8.35179889e-01, 4.29035948e-02, 3.83333333e-01, 8.16100566e-03,
       5.41501976e-03, 7.85349586e-01, 6.24750722e-01, 0.00000000e+00,
       6.92793511e-01, 2.47514333e-01, 1.71594263e-01, 1.16781328e-01,
       8.80000000e-01, 9.15007267e-02, 6.54540145e-01, 1.40354308e-01,
       3.00000000e-02, 8.49804971e-02, 3.89231284e-02, 7.59183007e-01,
       2.47899473e-01, 1.07384271e-01, 2.54150198e-02, 8.23309084e-03,
       6.32855117e-02, 1.00000000e+00, 1.41680925e-01, 1.02540018e-01,
       2.85262100e-01, 3.08695652e-02, 1.00000000e+00, 1.62250000e-01,
       7.85152027e-01, 2.02602789e-01, 9.90000000e-01, 5.53826541e-02,
       2.94394932e-01, 0.00000000e+00, 8.45487179e-01, 6.51828119e-02,
       9.93257324e-02, 5.66212542e-01, 3.00000000e-02, 1.33666489e-01,
       9.30000000e-01, 9.53333333e-01, 6.79933724e-01, 8.03469081e-01,
       8.03261945e-02, 7.74276316e-01, 2.63777778e-02, 0.00000000e+00,
       1.10677477e-01, 7.02407788e-01, 2.28545559e-01, 1.22458777e-01,
       1.00000000e+00, 6.09785096e-01, 1.64196032e-01, 1.71280380e-01,
       7.03400509e-01, 0.00000000e+00, 1.42029163e-01, 2.57374887e-01,
       1.71534759e-01, 3.11666667e-01, 5.92284374e-01, 5.20090853e-02,
       3.42021006e-02, 2.45920088e-01, 2.31100679e-02, 2.61604199e-01,
       8.29306078e-01, 8.50000000e-01, 1.00000000e+00, 7.33028573e-01,
       5.58719751e-01, 1.76470588e-03, 9.77777778e-03, 7.08840095e-01,
       2.31473352e-01, 8.01984917e-01, 0.00000000e+00, 1.50586447e-01,
       8.47235032e-01, 2.38502754e-01, 1.12058258e-01, 3.23954093e-01,
       4.36337680e-02, 7.61333333e-01, 5.76027878e-01, 1.94563867e-01])
```

In [35]:
```
# Predicción
predict_train = Random_Forest_pipeline.predict(X_train)
predict_test = Random_Forest_pipeline.predict(X_test)

# RMSE del train and test
print('RMSE on train data: ', mean_squared_error(y_train, predict_train)**(0.5))
print('RMSE on test data: ',  mean_squared_error(y_test, predict_test)**(0.5))
```

```
RMSE on train data:  0.19922203424831628
RMSE on test data:  0.44591551677314745
```

Gridsearch

In [36]:
```
# Listado de los parámetros modificables para el modelo
rf = RandomForestRegressor(random_state = 42)
from pprint import pprint

print('Parameters currently in use:\n')
pprint(rf.get_params())
```

```
Parameters currently in use:

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'criterion': 'squared_error',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': 42,
 'verbose': 0,
 'warm_start': False}
```

In [37]:
```
# Trees del random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Número de valores por cada split
max_features = ['auto', 'sqrt']
# Profundidad del Random Forest
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Número mínimo de muestras por split
min_samples_split = [2, 5, 10]
# Mínimo de muestras por nivel del Ranfom Forest (leaf)
min_samples_leaf = [1, 2, 4]
# Selección de muestra
bootstrap = [True, False]
# Creación del objeto Random Grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]}
```

In [38]:
```
# Use the random grid to search for best hyperparameters
# Objeto
rf = RandomForestRegressor()

# Random search con modificación de parámetros
rf_random = RandomizedSearchCV(estimator = rf,
                               param_distributions = random_grid,
                               n_iter = 100,
                               cv = 3,
                               verbose=2,
                               random_state=42,
                               n_jobs = -1)
# Fitting del modelo con los Train test sets
rf_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

Out[38]:
```
RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [10, 20, 30, 40, 50, 60,
                                                      70, 80, 90, 100, 110,
                                                      None],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [200, 400, 600, 800,
                                                         1000, 1200, 1400, 1600,
                                                         1800, 2000]},
                   random_state=42, verbose=2)
```

In [39]:
```
rf_random.best_params_
```

Out[39]:
```
{'n_estimators': 600,
 'min_samples_split': 10,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 110,
 'bootstrap': True}
```

In [40]:
```
# Best estimator
best_random = rf_random.best_estimator_

#Predicciones
predict_train = best_random.predict(X_train)
predict_test = best_random.predict(X_test)

# Root Mean Squared Error
print('RMSE on train data: ', mean_squared_error(y_train, predict_train)**(0.5))
print('RMSE on test data: ', mean_squared_error(y_test, predict_test)**(0.5))
```

```
RMSE on train data:  0.2564960758867486
RMSE on test data:  0.40792931104757624
```

- Exercici 2:

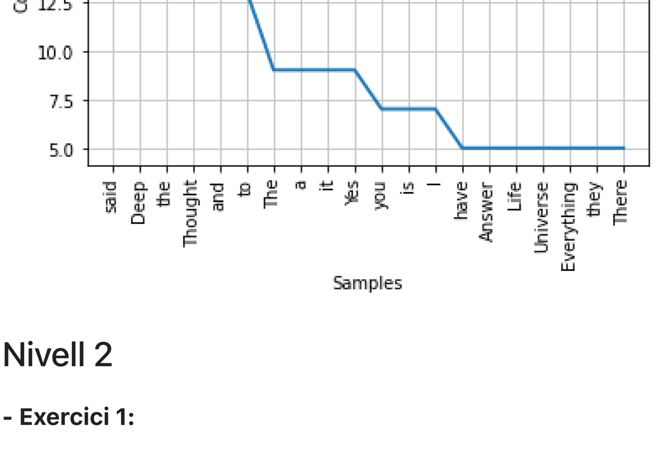
Agafa un text en anglès que vulguis, i calcula'n la freqüència de les paraules.

In [41]:
```
import nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
```

In [42]:
```
# Importar  txt
book_file = open("42.txt", "r")
book = book_file.read()
```

In [43]:
```
tokenized_text=sent_tokenize(book)
```

In [44]:
```
# Conteo de palabras repetidas
tokenized_book = word_tokenize(book)
# Ignorar puntuación
tokenized_book= [word for word in tokenized_book if word.isalnum()]
# repetición de palabras
fdist = FreqDist(tokenized_book)
# 20 palabras más repetidas
fdist.plot(20,cumulative=False)
plt.show()
```



Nivell 2

- Exercici 1:

Treu les stopwords i realitza stemming al teu conjunt de dades.

In [ ]:
```

```

Nivell 3

- Exercici 1:

Realitza sentiment analysis al teu conjunt de dades.