S12 T01: Pipelines, grid search i text mining

In [78]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import category_encoders as ce
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import  RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from sklearn.compose import make_column_selector
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.utils import check_array
from scipy import sparse
from sklearn import datasets
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.base import BaseEstimator
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import RandomizedSearchCV
import nltk
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk import sent_tokenize
from textblob import TextBlob
```

```
[nltk_data] Downloading package punkt to /home/rusi/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/rusi/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Exercici 1. Agafa el conjunt de dades que vulguis i realitza un pipeline i un gridsearch aplicant l'algorisme de Random Forest.

De l'Sprint07, carreguem les dades netes, sense nuls, amb l'històric de jugadors de la selecció espanyola de futbol absoluta masculina que han debutat (obtingudes a partir de la web bdfutbol.com). Recordem els noms de les columnes:

Sobrenom; Nom; Data Naixement; Lloc de Naixament; Província; País; Partits Jugats; Partits Titular; Partits Complets; Partits Suplent; Partits Substituït; Partits Convocats (sense jugar); Partits Guanyats; Partits Empetats; Partits Perduts; Minuts; Goles; Gols Penalt; Goles pròpia porta; Gols Encaixats; Targetes grogues; Targetes vermelles; Edat inicial; Edat final; Alçada; Pes

In [2]:
```python
jugadors = pd.read_csv('//home/rusi/Escritorio/rubenIT/DataSources/jugadores00.csv')#imp
```

In [3]:
```python
#Imprimim les dades filtrades per comprovar la importació
print(jugadors.describe())
print(jugadors.head(10))
print(jugadors.tail(10))
```

```
                PJ          PT          PC          PS          PX          PG  \
count   654.000000  654.000000  654.000000  654.000000  654.000000  654.000000
mean     14.155963   11.085627    8.006116    3.070336    3.056575    8.391437
std      22.460518   19.330256   14.271486    5.229901    7.115855   15.330149
min       1.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%       2.000000    1.000000    1.000000    0.000000    0.000000    1.000000
50%       5.000000    4.000000    3.000000    1.000000    1.000000    3.000000
75%      16.000000   12.000000    9.000000    3.000000    3.000000    9.000000
max     180.000000  161.000000  125.000000   42.000000   59.000000  131.000000

                PE          PP           Min           G          GP  \
count   654.000000  654.000000    654.000000  654.000000  654.000000
mean      3.333333    2.431193   1005.507645    1.960245    0.142202
std       4.831199    3.607972   1669.924268    5.165109    0.873092
min       0.000000    0.000000      1.000000    0.000000    0.000000
25%       0.000000    0.000000     90.000000    0.000000    0.000000
50%       1.000000    1.000000    360.000000    0.000000    0.000000
75%       4.000000    3.000000   1129.250000    1.000000    0.000000
max      33.000000   23.000000  13709.000000   59.000000   11.000000

               GPP          GE          TA          TR          EI          EF  \
count   654.000000  654.000000  654.000000  654.000000  654.000000  654.000000
mean      0.019878    0.905199    0.917431    0.032110   23.949541   26.831804
std       0.139687    6.868723    2.419149    0.184904    2.782392    3.488660
min       0.000000    0.000000    0.000000    0.000000   17.000000   17.000000
25%       0.000000    0.000000    0.000000    0.000000   22.000000   25.000000
50%       0.000000    0.000000    0.000000    0.000000   24.000000   27.000000
75%       0.000000    0.000000    1.000000    0.000000   26.000000   29.000000
max       1.000000  100.000000   24.000000    2.000000   34.000000   36.000000

            Altura        Peso
count   654.000000  654.000000
mean    177.594801   73.915902
std       6.021862    5.713472
min     160.000000   60.000000
25%     173.000000   70.000000
50%     178.000000   74.000000
75%     181.750000   77.000000
max     197.000000   95.000000
             Apodo                     Nombre       Fecha              Ciudad  \
0     Marcos Vales        Marcos Vales Illanes  05/04/1975           A Coruña
1            Acuña             Juan Acuña Naya  13/02/1923           A Coruña
2           Martín  José María Martín Rodríguez  25/04/1924           A Coruña
3          Casilla     Francisco Casilla Cortés  02/10/1986             Alcover
4      Juan Sánchez     Juan Ginés Sánchez Romero  15/05/1972              Aldaia
5        Cucurella       Marc Cucurella Saseta  22/07/1998              Alella
6           Piquer         Vicente Piquer Mora  24/02/1935  Algar de Palancia
7              Ito       Antonio Álvarez Pérez  21/01/1975        Almendralejo
8        Planas II          Javier Planas Abad  03/07/1949            Almudévar
9   Josep Martínez       Josep Martínez Riera  27/05/1998              Alzira
```

```
   Provincia    País  PJ  PT  PC  PS  ...  G  GP  GPP  GE  TA  TR  EI  EF  \
0   A Coruña  España   1   0   0   1  ...  0   0    0   0   0   0  23  23   
1   A Coruña  España   1   0   0   1  ...  0   0    0   1   0   0  18  18   
2   A Coruña  España   1   1   1   0  ...  0   0    0   0   0   0  28  28   
3  Tarragona  España   1   0   0   1  ...  0   0    0   1   0   0  28  28   
4   Valencia  España   1   0   0   1  ...  0   0    0   0   0   0  26  26   
5  Barcelona  España   1   1   0   0  ...  0   0    0   0   0   0  22  22   
6   Valencia  España   1   1   1   0  ...  0   0    0   0   0   0  26  26   
7    Badajoz  España   1   0   0   1  ...  0   0    0   0   0   0  23  23   
8     Huesca  España   1   1   1   0  ...  0   0    0   0   1   0  25  25   
9   Valencia  España   1   0   0   1  ...  0   0    0   0   0   0  23  23   

   Altura  Peso  
0   181.0  77.0  
1   179.0  88.0  
2   176.0  74.0  
3   192.0  83.0  
4   173.0  72.0  
5   172.0  68.0  
6   173.0  71.0  
7   175.0  70.0  
8   174.0  74.0  
9   191.0  78.0  

[10 rows x 25 columns]
                Apodo                       Nombre       Fecha           Ciudad  \
644         Fàbregas    Francesc Fàbregas Soler  04/05/1987    Arenys de Mar   
645  Fernando Torres  Fernando José Torres Sanz  20/03/1984      Fuenlabrada   
646      Xabi Alonso        Xabier Alonso Olano  25/11/1981           Tolosa   
647            Silva  David Josué Jiménez Silva  08/01/1986      Arguineguín   
648      Zubizarreta  Andoni Zubizarreta Urreta  23/10/1961  Vitoria-Gasteiz   
649          Iniesta       Andrés Iniesta Luján  11/05/1984    Fuentealbilla   
650         Busquets     Sergio Busquets Burgos  16/07/1988         Sabadell   
651             Xavi     Xavier Hernández Creus  25/01/1980         Terrassa   
652         Casillas    Iker Casillas Fernández  20/05/1981         Móstoles   
653     Sergio Ramos        Sergio Ramos García  30/03/1986            Camas   

        Provincia    País   PJ   PT   PC  PS  ...   G  GP  GPP   GE  TA  TR  \
644     Barcelona  España  110   68   22  42  ...  15   0    0    0  15   0   
645        Madrid  España  110   75   21  35  ...  38   5    0    0   4   0   
646      Gipuzkoa  España  114   86   48  28  ...  16   6    0    0  10   1   
647    Las Palmas  España  125   96   37  29  ...  35   2    0    0  10   0   
648   Araba/Álava  España  126  125  106   1  ...   0   0    1  100   2   1   
649      Albacete  España  131  105   47  26  ...  13   1    0    0   4   0   
650     Barcelona  España  133  119   89  14  ...   2   0    0    0  23   0   
651     Barcelona  España  133  108   64  25  ...  13   0    0    0   9   0   
652        Madrid  España  167  154  125  13  ...   0   0    0   93   2   0   
653       Sevilla  España  180  161  118  19  ...  23   8    0    0  24   0   

     EI  EF  Altura  Peso  
644  18  29   180.0  77.0  
645  19  30   186.0  78.0  
646  21  32   183.0  75.0  
647  20  32   170.0  67.0  
648  23  36   187.0  86.0  
649  22  34   171.0  68.0  
650  20  32   189.0  76.0  
651  20  34   170.0  68.0  
652  19  35   182.0  80.0  
653  18  34   184.0  83.0  

[10 rows x 25 columns]
```

Pipeline

1) Building a prototype Construïm el prototipus, millorant les dades del dataframe. Prescindirem dels features "Apodo", "Nombre", "Fecha" i "Ciudad", i convertirem en números "Provincia" i "País".

1.1) Encode the categorical variables.

In [4]:
```
jugadors00=jugadors
```

In [5]:
```
# create an object of the OneHotEncoder
OHE = ce.OneHotEncoder(cols=["Provincia","País"],use_cat_names=True)
# encode the categorical variables
jugadors00 = OHE.fit_transform(jugadors00)
```

In [6]:
```
print(jugadors00.head())
print(jugadors00.info())
```

```
          Apodo                          Nombre       Fecha    Ciudad  \
0  Marcos Vales         Marcos Vales Illanes  05/04/1975  A Coruña
1         Acuña              Juan Acuña Naya  13/02/1923  A Coruña
2        Martín  José María Martín Rodríguez  25/04/1924  A Coruña
3       Casilla     Francisco Casilla Cortés  02/10/1986   Alcover
4  Juan Sánchez    Juan Ginés Sánchez Romero  15/05/1972    Aldaia

   Provincia_A Coruña  Provincia_Tarragona  Provincia_Valencia  \
0                   1                    0                   0
1                   1                    0                   0
2                   1                    0                   0
3                   0                    1                   0
4                   0                    0                   1

   Provincia_Barcelona  Provincia_Badajoz  Provincia_Huesca  ...  G  GP  GPP  \
0                    0                  0                 0  ...  0   0    0
1                    0                  0                 0  ...  0   0    0
2                    0                  0                 0  ...  0   0    0
3                    0                  0                 0  ...  0   0    0
4                    0                  0                 0  ...  0   0    0

   GE  TA  TR  EI  EF  Altura  Peso
0   0   0   0  23  23   181.0  77.0
1   1   0   0  18  18   179.0  88.0
2   0   0   0  28  28   176.0  74.0
3   1   0   0  28  28   192.0  83.0
4   0   0   0  26  26   173.0  72.0

[5 rows x 90 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 654 entries, 0 to 653
Data columns (total 90 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Apodo                             654 non-null    object
 1   Nombre                            654 non-null    object
 2   Fecha                             654 non-null    object
 3   Ciudad                            654 non-null    object
 4   Provincia_A Coruña                654 non-null    int64
 5   Provincia_Tarragona               654 non-null    int64
 6   Provincia_Valencia                654 non-null    int64
 7   Provincia_Barcelona               654 non-null    int64
 8   Provincia_Badajoz                 654 non-null    int64
 9   Provincia_Huesca                  654 non-null    int64
 10  Provincia_Bizkaia                 654 non-null    int64
 11  Provincia_Málaga                  654 non-null    int64
 12  Provincia_La Rioja                654 non-null    int64
 13  Provincia_Gipuzkoa                654 non-null    int64
```

```
 14  Provincia_Extranjero               654 non-null    int64
 15  Provincia_Burgos                   654 non-null    int64
 16  Provincia_Cádiz                    654 non-null    int64
 17  Provincia_Murcia                   654 non-null    int64
 18  Provincia_Zamora                   654 non-null    int64
 19  Provincia_Ceuta                    654 non-null    int64
 20  Provincia_Cáceres                  654 non-null    int64
 21  Provincia_Alicante                 654 non-null    int64
 22  Provincia_León                     654 non-null    int64
 23  Provincia_Segovia                  654 non-null    int64
 24  Provincia_Asturias                 654 non-null    int64
 25  Provincia_Granada                  654 non-null    int64
 26  Provincia_Palencia                 654 non-null    int64
 27  Provincia_Huelva                   654 non-null    int64
 28  Provincia_Sevilla                  654 non-null    int64
 29  Provincia_Jaén                     654 non-null    int64
 30  Provincia_Santa Cruz de Tenerife   654 non-null    int64
 31  Provincia_Cantabria                654 non-null    int64
 32  Provincia_Las Palmas               654 non-null    int64
 33  Provincia_Lleida                   654 non-null    int64
 34  Provincia_Madrid                   654 non-null    int64
 35  Provincia_Toledo                   654 non-null    int64
 36  Provincia_Córdoba                  654 non-null    int64
 37  Provincia_Navarra                  654 non-null    int64
 38  Provincia_Lugo                     654 non-null    int64
 39  Provincia_Salamanca                654 non-null    int64
 40  Provincia_Pontevedra               654 non-null    int64
 41  Provincia_Valladolid               654 non-null    int64
 42  Provincia_Araba/Álava              654 non-null    int64
 43  Provincia_Zaragoza                 654 non-null    int64
 44  Provincia_Albacete                 654 non-null    int64
 45  Provincia_Almería                  654 non-null    int64
 46  Provincia_Castellón                654 non-null    int64
 47  Provincia_Melilla                  654 non-null    int64
 48  Provincia_Girona                   654 non-null    int64
 49  Provincia_Ciudad Real              654 non-null    int64
 50  Provincia_Teruel                   654 non-null    int64
 51  Provincia_Fernando Poo             654 non-null    int64
 52  Provincia_Soria                    654 non-null    int64
 53  Provincia_Islas Baleares           654 non-null    int64
 54  Provincia_Ourense                  654 non-null    int64
 55  Provincia_Ávila                    654 non-null    int64
 56  País_España                        654 non-null    int64
 57  País_Argentina                     654 non-null    int64
 58  País_Paraguay                      654 non-null    int64
 59  País_Suiza                         654 non-null    int64
 60  País_Italia                        654 non-null    int64
 61  País_Brasil                        654 non-null    int64
 62  País_Francia                       654 non-null    int64
 63  País_Dinamarca                     654 non-null    int64
 64  País_Guinea-Bisáu                  654 non-null    int64
 65  País_Hungría                       654 non-null    int64
 66  País_Guinea Ecuatorial             654 non-null    int64
 67  País_Marruecos                     654 non-null    int64
 68  País_Alemania                      654 non-null    int64
 69  País_Mauritania                    654 non-null    int64
 70  País_Uruguay                       654 non-null    int64
 71  PJ                                 654 non-null    int64
 72  PT                                 654 non-null    int64
 73  PC                                 654 non-null    int64
 74  PS                                 654 non-null    int64
 75  PX                                 654 non-null    int64
 76  PG                                 654 non-null    int64
 77  PE                                 654 non-null    int64
 78  PP                                 654 non-null    int64
 79  Min                                654 non-null    int64
```

```
 80  G                                 654 non-null    int64
 81  GP                                654 non-null    int64
 82  GPP                               654 non-null    int64
 83  GE                                654 non-null    int64
 84  TA                                654 non-null    int64
 85  TR                                654 non-null    int64
 86  EI                                654 non-null    int64
 87  EF                                654 non-null    int64
 88  Altura                            654 non-null    float64
 89  Peso                              654 non-null    float64
dtypes: float64(2), int64(84), object(4)
memory usage: 460.0+ KB
None
```

El nou dataframe té 90 columnes, cosa que és massa nombrós pel nostre anàlisi. En comptes de fer-ho amb dummies, donarem valors numèrics a "Provincia" i "País" directament.

In [7]:
```python
number=LabelEncoder()
jugadors=jugadors.drop(["Apodo","Nombre","Fecha","Ciudad"],axis=1)
jugadors["Provincia"]=number.fit_transform(jugadors["Provincia"].astype("str"))
jugadors["País"]=number.fit_transform(jugadors["País"].astype("str"))
```

In [8]:
```python
print(jugadors.iloc[0:50,:])
print(jugadors.info())
```

```
    Provincia  País  PJ  PT  PC  PS  PX  PG  PE  PP  ...  G  GP  GPP  GE  TA  \
0           0     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
1           0     4   1   0   0   1   0   1   0   0  ...  0   0    0   1   0
2           0     4   1   1   1   0   0   1   0   0  ...  0   0    0   0   0
3          44     4   1   0   0   1   0   0   0   1  ...  0   0    0   1   0
4          47     4   1   0   0   1   0   0   1   0  ...  0   0    0   0   0
5           7     4   1   1   0   0   1   1   0   0  ...  0   0    0   0   0
6          47     4   1   1   1   0   0   0   1   0  ...  0   0    0   0   0
7           6     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
8          23     4   1   1   1   0   0   1   0   0  ...  0   0    0   0   1
9          47     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
10          8     4   1   1   1   0   0   1   0   0  ...  0   0    0   0   0
11         34     4   1   0   0   1   0   0   0   1  ...  0   0    0   0   0
12         26     4   1   1   0   0   1   1   0   0  ...  0   0    0   0   0
13         26     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
14          8     4   1   1   1   0   0   0   0   1  ...  0   0    0   0   0
15         19     4   1   1   1   0   0   0   1   0  ...  0   0    0   0   0
16         19     4   1   0   0   1   0   0   0   1  ...  0   0    0   0   0
17         19     4   1   1   1   0   0   0   0   1  ...  0   0    0   0   0
18          6     4   1   1   1   0   0   0   1   0  ...  0   0    0   0   0
19          8     4   1   0   0   1   0   0   0   1  ...  0   0    0   2   0
20          8     4   1   1   1   0   0   0   1   0  ...  0   0    0   0   0
21          7     4   1   0   0   1   0   1   0   0  ...  1   0    0   0   0
22          7     4   1   1   1   0   0   1   0   0  ...  0   0    0   1   0
23          7     4   1   1   1   0   0   1   0   0  ...  0   0    0   0   0
24          7     4   1   1   1   0   0   1   0   0  ...  0   0    0   1   0
25          7     4   1   1   1   0   0   0   0   1  ...  0   0    0   0   0
26          7     4   1   0   0   1   0   0   1   0  ...  0   0    0   0   0
27         47     4   1   1   1   0   0   0   0   1  ...  0   0    0   0   0
28          8     4   1   1   1   0   0   0   0   1  ...  0   0    0   0   0
29          8     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
30          8     4   1   1   1   0   0   0   1   0  ...  0   0    0   0   0
31          8     4   1   1   1   0   0   0   0   1  ...  0   0    0   2   0
32         17     1   1   1   1   0   0   1   0   0  ...  0   0    0   0   0
33          9     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
34         15     4   1   1   1   0   0   0   0   1  ...  0   0    0   0   0
35         26     4   1   0   0   1   0   0   0   1  ...  0   0    0   0   0
36         17    12   1   1   1   0   0   1   0   0  ...  0   0    0   0   0
37         33     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
38         33     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
```

```
39          7     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
40         49     4   1   1   0   0   1   0   0   1  ...  0   0    0   0   0
41         47     4   1   1   0   0   1   0   1   0  ...  0   0    0   0   0
42         12     4   1   1   1   0   0   1   0   0  ...  0   0    0   0   0
43         34     4   1   1   0   0   1   0   0   1  ...  0   0    0   0   0
44         14     4   1   1   1   0   0   0   0   1  ...  0   0    0   4   0
45          8     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0
46          6     4   1   1   0   0   1   1   0   0  ...  0   0    0   0   1
47         19     4   1   1   0   0   1   1   0   0  ...  0   0    0   0   0
48         19     4   1   1   1   0   0   0   1   0  ...  0   0    0   0   0
49         19     4   1   0   0   1   0   1   0   0  ...  0   0    0   0   0

    TR  EI  EF  Altura  Peso
0    0  23  23   181.0  77.0
1    0  18  18   179.0  88.0
2    0  28  28   176.0  74.0
3    0  28  28   192.0  83.0
4    0  26  26   173.0  72.0
5    0  22  22   172.0  68.0
6    0  26  26   173.0  71.0
7    0  23  23   175.0  70.0
8    0  25  25   174.0  74.0
9    0  23  23   191.0  78.0
10   0  27  27   186.0  80.0
11   0  20  20   172.0  72.0
12   0  23  23   186.0  80.0
13   0  23  23   179.0  74.0
14   0  25  25   177.0  70.0
15   0  29  29   173.0  71.0
16   0  27  27   185.0  84.0
17   0  29  29   177.0  75.0
18   0  29  29   174.0  72.0
19   0  24  24   178.0  76.0
20   0  25  25   170.0  66.0
21   0  23  23   177.0  69.0
22   0  26  26   183.0  80.0
23   0  22  22   178.0  66.0
24   0  33  33   180.0  80.0
25   0  23  23   168.0  62.0
26   0  20  20   170.0  70.0
27   0  23  23   172.0  71.0
28   0  26  26   183.0  83.0
29   0  21  21   186.0  79.0
30   0  25  25   176.0  73.0
31   0  27  27   180.0  77.0
32   0  27  27   178.0  72.0
33   0  21  21   183.0  81.0
34   0  23  23   184.0  81.0
35   0  21  21   180.0  70.0
36   0  25  25   168.0  67.0
37   0  28  28   181.0  73.0
38   0  23  23   197.0  90.0
39   0  22  22   172.0  68.0
40   0  25  25   181.0  80.0
41   0  22  22   182.0  78.0
42   0  29  29   182.0  74.0
43   0  21  21   172.0  69.0
44   0  28  28   185.0  82.0
45   0  25  25   176.0  70.0
46   0  21  21   176.0  71.0
47   0  22  22   180.0  74.0
48   0  21  21   177.0  75.0
49   0  28  28   182.0  78.0

[50 rows x 21 columns]
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 654 entries, 0 to 653
Data columns (total 21 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Provincia  654 non-null    int64  
 1   País       654 non-null    int64  
 2   PJ         654 non-null    int64  
 3   PT         654 non-null    int64  
 4   PC         654 non-null    int64  
 5   PS         654 non-null    int64  
 6   PX         654 non-null    int64  
 7   PG         654 non-null    int64  
 8   PE         654 non-null    int64  
 9   PP         654 non-null    int64  
 10  Min        654 non-null    int64  
 11  G          654 non-null    int64  
 12  GP         654 non-null    int64  
 13  GPP        654 non-null    int64  
 14  GE         654 non-null    int64  
 15  TA         654 non-null    int64  
 16  TR         654 non-null    int64  
 17  EI         654 non-null    int64  
 18  EF         654 non-null    int64  
 19  Altura     654 non-null    float64
 20  Peso       654 non-null    float64
dtypes: float64(2), int64(19)
memory usage: 107.4 KB
None
```

1.2) Scale data

Podem escalar totes les columnes (entre 0 i 1), excepte "Peso", que el normalitzem (mitjana=0 i desviació=1). "Altura" és el target i no el tractem.

In [9]:
```python
#Estandarització i eliminació target "Altura"
jugadors01 = jugadors.drop(["Provincia","País","Peso","Altura"], axis=1)
jugadors01_norm = (jugadors01-jugadors01.min())/(jugadors01.max()-jugadors01.min())
#jugadors01_norm = jugadors.drop(["Altura"], axis=1)
```

In [10]:
```python
print(jugadors01_norm.head())
print(jugadors01_norm.info())
```

```
    PJ        PT     PC       PS   PX        PG        PE        PP       Min  \
0  0.0  0.000000  0.000  0.02381  0.0  0.007634  0.000000  0.000000  0.000000   
1  0.0  0.000000  0.000  0.02381  0.0  0.007634  0.000000  0.000000  0.001094   
2  0.0  0.006211  0.008  0.00000  0.0  0.007634  0.000000  0.000000  0.006493   
3  0.0  0.000000  0.000  0.02381  0.0  0.000000  0.000000  0.043478  0.000875   
4  0.0  0.000000  0.000  0.02381  0.0  0.000000  0.030303  0.000000  0.000802   

     G   GP  GPP    GE   TA   TR        EI        EF  
0  0.0  0.0  0.0  0.00  0.0  0.0  0.352941  0.315789  
1  0.0  0.0  0.0  0.01  0.0  0.0  0.058824  0.052632  
2  0.0  0.0  0.0  0.00  0.0  0.0  0.647059  0.578947  
3  0.0  0.0  0.0  0.01  0.0  0.0  0.647059  0.578947  
4  0.0  0.0  0.0  0.00  0.0  0.0  0.529412  0.473684  
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 654 entries, 0 to 653
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   PJ      654 non-null    float64
 1   PT      654 non-null    float64
 2   PC      654 non-null    float64
 3   PS      654 non-null    float64
```

```
 4   PX       654 non-null    float64
 5   PG       654 non-null    float64
 6   PE       654 non-null    float64
 7   PP       654 non-null    float64
 8   Min      654 non-null    float64
 9   G        654 non-null    float64
 10  GP       654 non-null    float64
 11  GPP      654 non-null    float64
 12  GE       654 non-null    float64
 13  TA       654 non-null    float64
 14  TR       654 non-null    float64
 15  EI       654 non-null    float64
 16  EF       654 non-null    float64
dtypes: float64(17)
memory usage: 87.0 KB
None
```

In [11]:

```
#Normalització
jugadors02=jugadors.loc[:,["Provincia","País","Peso"]]
ss = StandardScaler()
jugadors03 = ss.fit_transform(jugadors02.to_numpy())
jugadors03 = pd.DataFrame(jugadors03, columns=["Provincia","País","Peso"])
```

In [12]:

```
print(jugadors03.head())
print(jugadors03.info())
```

```
   Provincia      País      Peso
0  -1.524230 -0.070183  0.540207
1  -1.524230 -0.070183  2.466955
2  -1.524230 -0.070183  0.014730
3   1.476060 -0.070183  1.591161
4   1.680626 -0.070183 -0.335587
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 654 entries, 0 to 653
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Provincia  654 non-null    float64
 1   País       654 non-null    float64
 2   Peso       654 non-null    float64
dtypes: float64(3)
memory usage: 15.5 KB
None
```

In [13]:

```
jugadors04 = pd.concat((jugadors01_norm,jugadors03.loc[:,["Provincia","País","Peso"]]),
jugadors04= pd.concat((jugadors04,jugadors.loc[:,"Altura"]), 1)
```

```
/tmp/ipykernel_42366/669182810.py:1: FutureWarning: In a future version of pandas all ar
guments of concat except for the argument 'objs' will be keyword-only
  jugadors04 = pd.concat((jugadors01_norm,jugadors03.loc[:,["Provincia","País","Pes
o"]]), 1)
/tmp/ipykernel_42366/669182810.py:2: FutureWarning: In a future version of pandas all ar
guments of concat except for the argument 'objs' will be keyword-only
  jugadors04= pd.concat((jugadors04,jugadors.loc[:,"Altura"]), 1)
```

In [14]:

```
print(jugadors04.head())
print(jugadors04.info())
```

```
    PJ        PT     PC       PS   PX        PG        PE        PP       Min  \
0  0.0  0.000000  0.000  0.02381  0.0  0.007634  0.000000  0.000000  0.000000   
1  0.0  0.000000  0.000  0.02381  0.0  0.007634  0.000000  0.000000  0.001094   
2  0.0  0.006211  0.008  0.00000  0.0  0.007634  0.000000  0.000000  0.006493   
3  0.0  0.000000  0.000  0.02381  0.0  0.000000  0.000000  0.043478  0.000875   
4  0.0  0.000000  0.000  0.02381  0.0  0.000000  0.030303  0.000000  0.000802   

     G  ...  GPP    GE    TA    TR        EI        EF  Provincia      País  \
```

```
0  0.0  ...  0.0  0.00  0.0  0.0  0.352941  0.315789  -1.524230 -0.070183
1  0.0  ...  0.0  0.01  0.0  0.0  0.058824  0.052632  -1.524230 -0.070183
2  0.0  ...  0.0  0.00  0.0  0.0  0.647059  0.578947  -1.524230 -0.070183
3  0.0  ...  0.0  0.01  0.0  0.0  0.647059  0.578947   1.476060 -0.070183
4  0.0  ...  0.0  0.00  0.0  0.0  0.529412  0.473684   1.680626 -0.070183

       Peso  Altura
0  0.540207   181.0
1  2.466955   179.0
2  0.014730   176.0
3  1.591161   192.0
4 -0.335587   173.0

[5 rows x 21 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 654 entries, 0 to 653
Data columns (total 21 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   PJ         654 non-null    float64
 1   PT         654 non-null    float64
 2   PC         654 non-null    float64
 3   PS         654 non-null    float64
 4   PX         654 non-null    float64
 5   PG         654 non-null    float64
 6   PE         654 non-null    float64
 7   PP         654 non-null    float64
 8   Min        654 non-null    float64
 9   G          654 non-null    float64
 10  GP         654 non-null    float64
 11  GPP        654 non-null    float64
 12  GE         654 non-null    float64
 13  TA         654 non-null    float64
 14  TR         654 non-null    float64
 15  EI         654 non-null    float64
 16  EF         654 non-null    float64
 17  Provincia  654 non-null    float64
 18  País       654 non-null    float64
 19  Peso       654 non-null    float64
 20  Altura     654 non-null    float64
dtypes: float64(21)
memory usage: 107.4 KB
None
```

1.3) Model building

Fem servir el model predictiu de regressió Random Forest, i trobem el RMSE (Root Mean Squared Error).

In [15]:
```
X = jugadors04.drop(columns=["Altura"])
y = jugadors04["Altura"]

# randomly split the data
X, test_X, y, test_y = train_test_split(X,y,test_size=0.3,random_state=42)

# shape of train and test splits
X.shape, test_X.shape, y.shape, test_y.shape
```

Out[15]:
```
((457, 20), (197, 20), (457,), (197,))
```

In [16]:
```
# create an object of the RandomForestRegressor
model_RFR = RandomForestRegressor(max_depth=10)

# fit the model with the training data
model_RFR.fit(X, y)
```

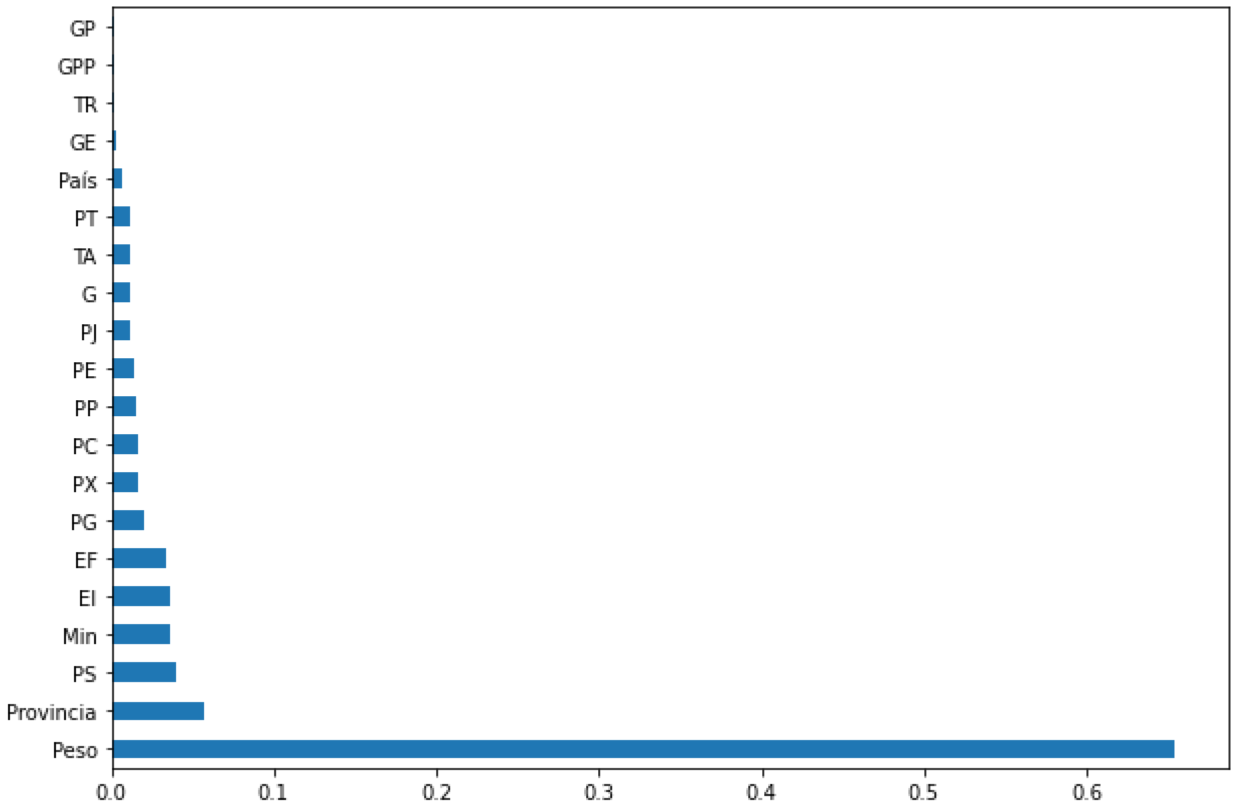
```python
# predict the target on train and test data
predict_train = model_RFR.predict(X)
predict_test = model_RFR.predict(test_X)

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

```
RMSE on train data:  1.6242631713439897
RMSE on test data:  3.6081435794826793
```

1.4) Feature Importance

Veiem quina és la importància de cada un dels atributs per predir el target "Altura".

In [17]:
```python
plt.figure(figsize=(10,7))
feat_importances = pd.Series(model_RFR.feature_importances_, index = X.columns)
feat_importances.nlargest(20).plot(kind='barh');
```



Amb 6 features arribaríem a prop del 80% del total d'importància. Fixem aquesta dada i comparem el RMSE per contrastar la millora amb aquesta reducció.

In [18]:
```python
X = jugadors04.loc[:,["Peso","Provincia","PS","EF","EI","Min"]]
y = jugadors04["Altura"]

# randomly split the data
X, test_X, y, test_y = train_test_split(X,y,test_size=0.3,random_state=42)

# shape of train and test splits
X.shape, test_X.shape, y.shape, test_y.shape
```

Out[18]:
```
((457, 6), (197, 6), (457,), (197,))
```

In [19]:
```python
# create an object of the RandomForestRegressor
model_RFR = RandomForestRegressor(max_depth=10)
```

```
# fit the model with the training data
model_RFR.fit(X, y)

# predict the target on train and test data
predict_train = model_RFR.predict(X)
predict_test = model_RFR.predict(test_X)

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

```
RMSE on train data:  1.660670329283194
RMSE on test data:  3.690543837014724
```

El model amb la reducció de ***features*** puja molt lleugerament el RMSE, així que aprovem el nou dataframe per aplicar-ho en el pipeline (1.624 vs 1.660 en el train, i 3.608 vs 3.690 en el test).

1.5) Identify features to build the Machine Learning pipeline

In [20]:
```
jugadors05=jugadors04.loc[:,["Altura","Peso","Provincia","PS","EF","EI","Min"]]
```

In [21]:
```
print(jugadors05.head())
print(jugadors05.info())
```

```
   Altura      Peso  Provincia       PS        EF        EI       Min
0   181.0  0.540207  -1.524230  0.02381  0.315789  0.352941  0.000000
1   179.0  2.466955  -1.524230  0.02381  0.052632  0.058824  0.001094
2   176.0  0.014730  -1.524230  0.00000  0.578947  0.647059  0.006493
3   192.0  1.591161   1.476060  0.02381  0.578947  0.647059  0.000875
4   173.0 -0.335587   1.680626  0.02381  0.473684  0.529412  0.000802
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 654 entries, 0 to 653
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Altura     654 non-null    float64
 1   Peso       654 non-null    float64
 2   Provincia  654 non-null    float64
 3   PS         654 non-null    float64
 4   EF         654 non-null    float64
 5   EI         654 non-null    float64
 6   Min        654 non-null    float64
dtypes: float64(7)
memory usage: 35.9 KB
None
```

2) Pipeline design

Hem identificat els següents passos de preprocessat per crear el nostre model de pipeline: 1)Drop columns. 2)Transform column (labelencoder). 3)Scale data. 4)Normalització.

In [22]:
```
jugadors = pd.read_csv('//home/rusi/Escritorio/rubenIT/DataSources/jugadores00.csv')#imp
```

In [23]:
```
# seperate the independent and target variables
X02 = jugadors.drop(columns=["Altura"])
y02 = jugadors["Altura"]
```

In [24]:
```
train_x, test_x, train_y, test_y = train_test_split(X02, y02,test_size=0.3,random_state=
```

In [25]:
```
numeric_preprocessor = Pipeline(
    steps=[
        ("imputation_mean", SimpleImputer(missing_values=np.nan, strategy="mean")),
```

```python
            ("scaler", StandardScaler()),
        ]
)

numeric_preprocessor02 = Pipeline(
    steps=[
        ("imputation_mean", SimpleImputer(missing_values=np.nan, strategy="mean")),
        ("normalizer", Normalizer()),
    ]
)

categorical_preprocessor = Pipeline(
    steps=[
        (
            "imputation_constant",
            SimpleImputer(fill_value="missing", strategy="constant"),
        ),
        ("ordinal_encoder", OrdinalEncoder(handle_unknown="use_encoded_value",unknown_va
        ("normalitza",StandardScaler()),
    ]
)

preprocessor = ColumnTransformer(
    [
        ("drop_columns", "drop", ["Apodo","Nombre","Fecha","Ciudad",
                                  "PX","PG", "PE","PP",
                                  "G", "GP", "GPP","GE", "TA", "TR",
                                  "Provincia","País"]),
        ("categorical", categorical_preprocessor, ["Provincia","País"]),
        ("numerical_standardize", numeric_preprocessor, ["Peso"]),
        ("numerical_normalizer", numeric_preprocessor02,["PS","Min","EI", "EF"]),
    ]
)

pipe = make_pipeline(preprocessor, RandomForestRegressor(max_depth=10,random_state=42))
pipe
```

Out[25]:

```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(transformers=[('drop_columns', 'drop',
                                                  ['Apodo', 'Nombre', 'Fecha',
                                                   'Ciudad', 'PX', 'PG', 'PE',
                                                   'PP', 'G', 'GP', 'GPP', 'GE',
                                                   'TA', 'TR', 'Provincia',
                                                   'País']),
                                                 ('categorical',
                                                  Pipeline(steps=[('imputation_constan
t',
                                                                   SimpleImputer(fill_va
lue='missing',
                                                                                 strateg
y='constant')),
                                                                  ('ordinal_encoder',
                                                                   OrdinalEncoder(handle
_...
                                                  ['Provincia', 'País']),
                                                 ('numerical_standardize',
                                                  Pipeline(steps=[('imputation_mean',
                                                                   SimpleImputer()),
                                                                  ('scaler',
                                                                   StandardScaler())]),
                                                  ['Peso']),
                                                 ('numerical_normalizer',
                                                  Pipeline(steps=[('imputation_mean',
                                                                   SimpleImputer()),
                                                                  ('normalizer',
                                                                   Normalizer())]),
```

```
                                   ['PS', 'Min', 'EI', 'EF'])])),
                ('randomforestregressor',
                 RandomForestRegressor(max_depth=10, random_state=42))])
```

In [26]:
```
# fit the pipeline with the training data
pipe.fit(train_x,train_y)
# predict target values on the training data
pipe.predict(test_x)
```

Out[26]:
```
array([175.70934259, 182.62275806, 184.61037535, 180.46492238,
       170.18388473, 181.04930736, 175.30013095, 170.73893078,
       170.73826255, 178.22134329, 172.7170088 , 172.96667349,
       176.73625134, 182.16222018, 176.70949405, 172.55355411,
       171.28748551, 175.18344977, 181.37762616, 171.78751732,
       183.39931668, 181.16247835, 181.79668817, 181.33066089,
       174.56070859, 175.37616553, 178.7142957 , 185.27348798,
       182.16330118, 181.04783697, 181.34951104, 179.71060396,
       173.97952544, 180.85123427, 178.48788979, 181.5993148 ,
       183.22544473, 174.74031947, 170.66501242, 178.46949287,
       169.65351367, 170.53061029, 172.018475  , 188.76316767,
       177.00195735, 180.57387038, 181.47389081, 178.38415521,
       174.96053548, 177.10657937, 176.16495238, 171.65366041,
       171.07559207, 171.02685326, 182.18522598, 169.45631047,
       182.65871301, 170.53725973, 175.76928863, 183.95496336,
       175.83548135, 177.96691955, 174.41996766, 171.1766132 ,
       169.06847222, 181.77740227, 169.62248369, 187.90526399,
       180.48785562, 181.20421831, 176.46749331, 175.22691993,
       174.07071871, 175.32938395, 177.43276428, 184.34861785,
       174.17520599, 178.51552404, 175.07894689, 183.4697094 ,
       177.06508308, 171.87961166, 182.1472239 , 180.16039857,
       171.16919851, 185.13050979, 169.47788166, 182.4696656 ,
       180.52105256, 176.87118235, 188.29472078, 169.95846078,
       178.33969122, 183.72580188, 176.5355347 , 184.51028431,
       187.93351282, 184.86294514, 170.35129575, 184.85678236,
       171.06296413, 180.16670976, 176.9005173 , 181.69344763,
       179.62538082, 181.94616793, 184.38561622, 186.23125857,
       180.04229917, 169.95613297, 173.65130767, 184.33304477,
       176.7964599 , 174.86317835, 181.01827842, 184.57695858,
       178.6910504 , 177.94267544, 179.50632988, 188.33526399,
       170.88980423, 177.79110949, 182.66744012, 179.811     ,
       175.14726626, 175.75420971, 180.65572222, 181.03099767,
       174.74295569, 179.61448914, 183.37850509, 183.10208857,
       178.4818889 , 175.01620144, 182.45703965, 187.29387191,
       175.26125397, 179.82677778, 180.80844372, 176.09262165,
       170.21541882, 182.08104202, 179.69972449, 174.11021197,
       180.07232937, 177.63708112, 171.62038814, 180.65467322,
       171.92103598, 182.74038885, 175.29847646, 173.05831802,
       172.09100553, 182.4357135 , 176.43702194, 176.17240629,
       179.41633176, 171.70437288, 176.08597554, 170.78820497,
       174.00777177, 179.37016667, 173.28406917, 180.66614059,
       176.45800619, 175.4774509 , 183.88111812, 170.38698255,
       171.23930135, 177.2563169 , 187.33748252, 170.60183886,
       178.3435    , 176.89428005, 185.81909363, 176.94138354,
       182.39317663, 181.18210509, 175.06751551, 186.44544239,
       171.2681509 , 182.44814201, 171.56614286, 169.83255565,
       176.48228878, 182.57381867, 184.76933874, 176.03640042,
       178.59716247, 170.2688297 , 169.0827544 , 182.05125032,
       176.56255556, 176.29161625, 183.97259376, 178.70810726,
       182.00939749])
```

In [27]:
```
# read the test data
test_data = jugadors = pd.read_csv('//home/rusi/Escritorio/rubenIT/DataSources/jugadores
predict_test = pipe.predict(test_x)
```

In [35]:
```
print("Mitja train_y: ",np.mean(train_y))
```

```
print("Mitja test_y: ",np.mean(test_y))
print("Mitja predict_train: ",np.mean(predict_train))
print("Mitja predict_test: ",np.mean(predict_test))
print("Mitja Altura jugadors: ", np.mean(jugadors.Altura))
```

```
Mitja train_y:  177.50765864332604
Mitja test_y:  177.7969543147208
Mitja predict_train:  177.5076879073611
Mitja predict_test:  177.81352830461853
Mitja Altura jugadors:  177.5948012232416
```

In [36]:
```
print('RMSE on train data: ', mean_squared_error(train_y, predict_train)**(0.5))
print('RMSE on test data: ', mean_squared_error(test_y, predict_test)**(0.5))
```

```
RMSE on train data:  2.7631782507651415
RMSE on test data:  3.654590587410196
```

Com podem veure, el valor "Altura" s'apropa molt a la mitjana dels valors d'entrenament i de tota la sèrie, 177.507 vs 177.598 vs 177.595. La predicció amb un 30% de les dades per fer el testeig, dona un valor de 177.853, molt a prop dels valors mitjans. Podem afirmar que el resultat obtingut és força bo en aquest sentit.

En quant a l'error RMSE, incrementa notablement en el train (2.763 vs 1.660), i és molt semblant en el test (3.654 vs 3.690). Té la seva lògica, ja que a mesura que hi ha més dades, tindrem més dispersió que fomentarà un increment en el RMSE.

Grid search

Grid Search Parameter Tuning. Grid search is an approach to parameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a grid.

In [40]:
```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'randomforestregressor__n_estimators': n_estimators,
               'randomforestregressor__max_features': max_features,
               'randomforestregressor__max_depth': max_depth,
               'randomforestregressor__min_samples_split': min_samples_split,
               'randomforestregressor__min_samples_leaf': min_samples_leaf,
               'randomforestregressor__bootstrap': bootstrap}
```

In [41]:
```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = pipe, param_distributions = random_grid,
                               n_iter = 30, cv = 3,random_state=42, n_jobs = -1)
```

In [42]:
```
rf_random.fit(train_x, train_y)
```

Out[42]:
```
RandomizedSearchCV(cv=3,
                   estimator=Pipeline(steps=[('columntransformer',
                                              ColumnTransformer(transformers=[('drop_col
umns',
                                                                               'drop',
                                                                               ['Apodo',
                                                                                'Nombr
e',
                                                                                'Fecha',
                                                                                'Ciuda
d',
                                                                                'PX',
                                                                                'PG',
                                                                                'PE',
                                                                                'PP',
                                                                                'G',
                                                                                'GP',
                                                                                'GPP',
                                                                                'GE',
                                                                                'TA',
                                                                                'TR',
                                                                                'Provinc
ia',
                                                                                'Paí
s']),
                                                                              ('categori
cal',
                                                                               Pipeline
(steps=[('imputation_constant',

        SimpleImputer(fill_value='missing',

                      strategy='constant')),

       ('ordin...
                   param_distributions={'randomforestregressor__bootstrap': [True,
                                                                              False],
                                        'randomforestregressor__max_depth': [10,
                                                                              20,
                                                                              30,
                                                                              40,
                                                                              50,
                                                                              60,
                                                                              70,
                                                                              80,
                                                                              90,
                                                                              100,
                                                                              110,
                                                                              None],
                                        'randomforestregressor__max_features': ['auto',
                                                                                 'sqrt'],
                                        'randomforestregressor__min_samples_leaf': [1,
                                                                                     2,
                                                                                     4],
                                        'randomforestregressor__min_samples_split': [2,
                                                                                      5,
                                                                                      1
0],
                                        'randomforestregressor__n_estimators': [200,
                                                                                 400,
                                                                                 600,
                                                                                 800,
                                                                                 1000,
                                                                                 1200,
```

```
                                                                                                1400,
                                                                                                1600,
                                                                                                1800,
                                                                                                2000]},
                                random_state=42)
```

In [43]:
```
rf_random.best_params_
```

Out[43]:
```
{'randomforestregressor__n_estimators': 1800,
 'randomforestregressor__min_samples_split': 10,
 'randomforestregressor__min_samples_leaf': 4,
 'randomforestregressor__max_features': 'auto',
 'randomforestregressor__max_depth': 90,
 'randomforestregressor__bootstrap': True}
```

In [44]:
```
# Best estimator
best_random = rf_random.best_estimator_

#Predictions
predict_train = best_random.predict(train_x)
predict_test = best_random.predict(test_x)

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

```
RMSE on train data:  2.7631782507651415
RMSE on test data:  3.6545905874l0196
```

L'error RMSE es manté en el mateix valor. Això podria ser degut a què les mostres són escasses, i que no té sentit aplicar una millora d'aquest tipus.

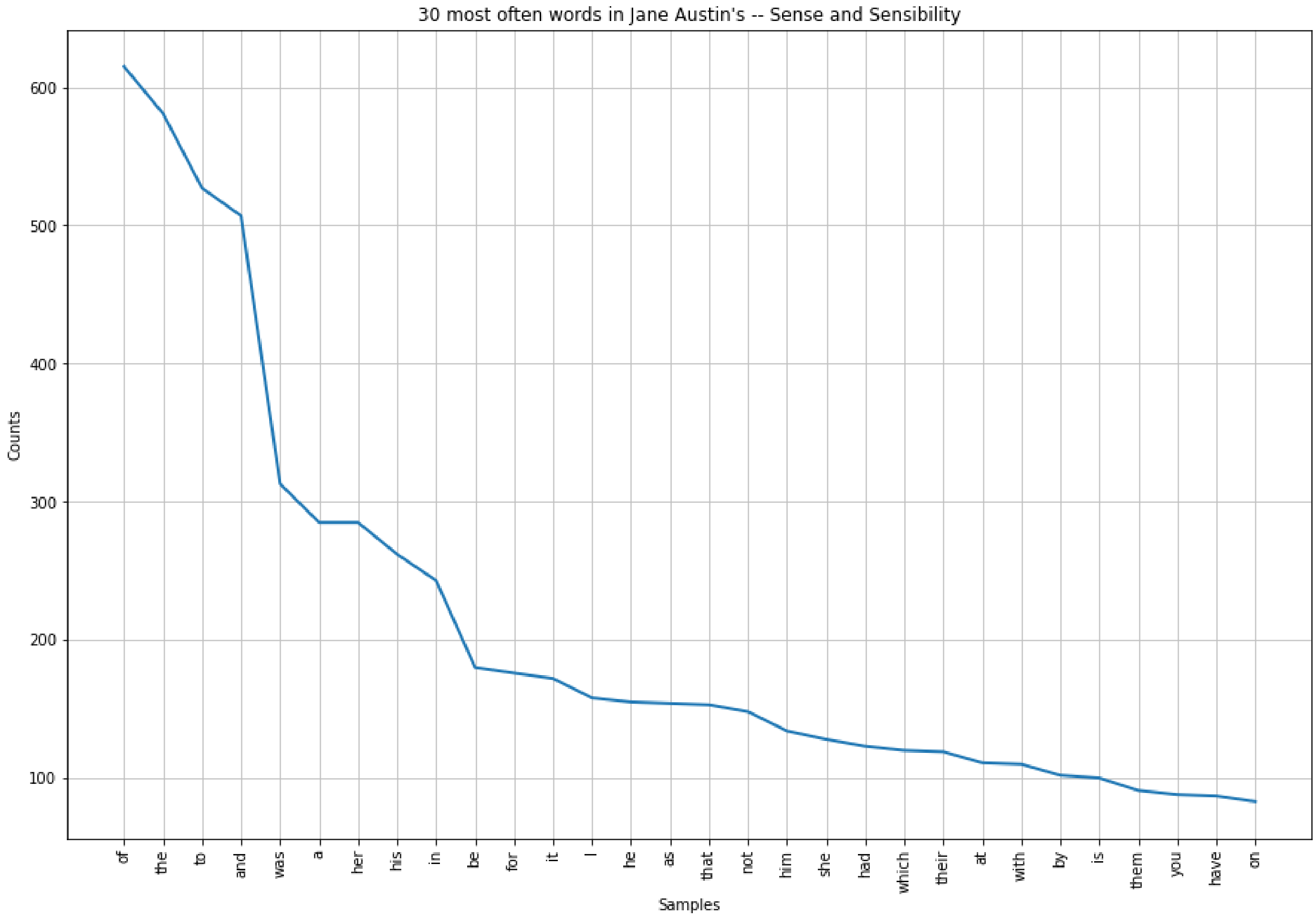
Exercici 2. Agafa un text en anglès que vulguis, i calcula'n la freqüència de les paraules

De la següent pàgina web, descarreguem un capítol del llibre de Jane Austin "Sense and Sensibility":

https://www.fulltextarchive.com/

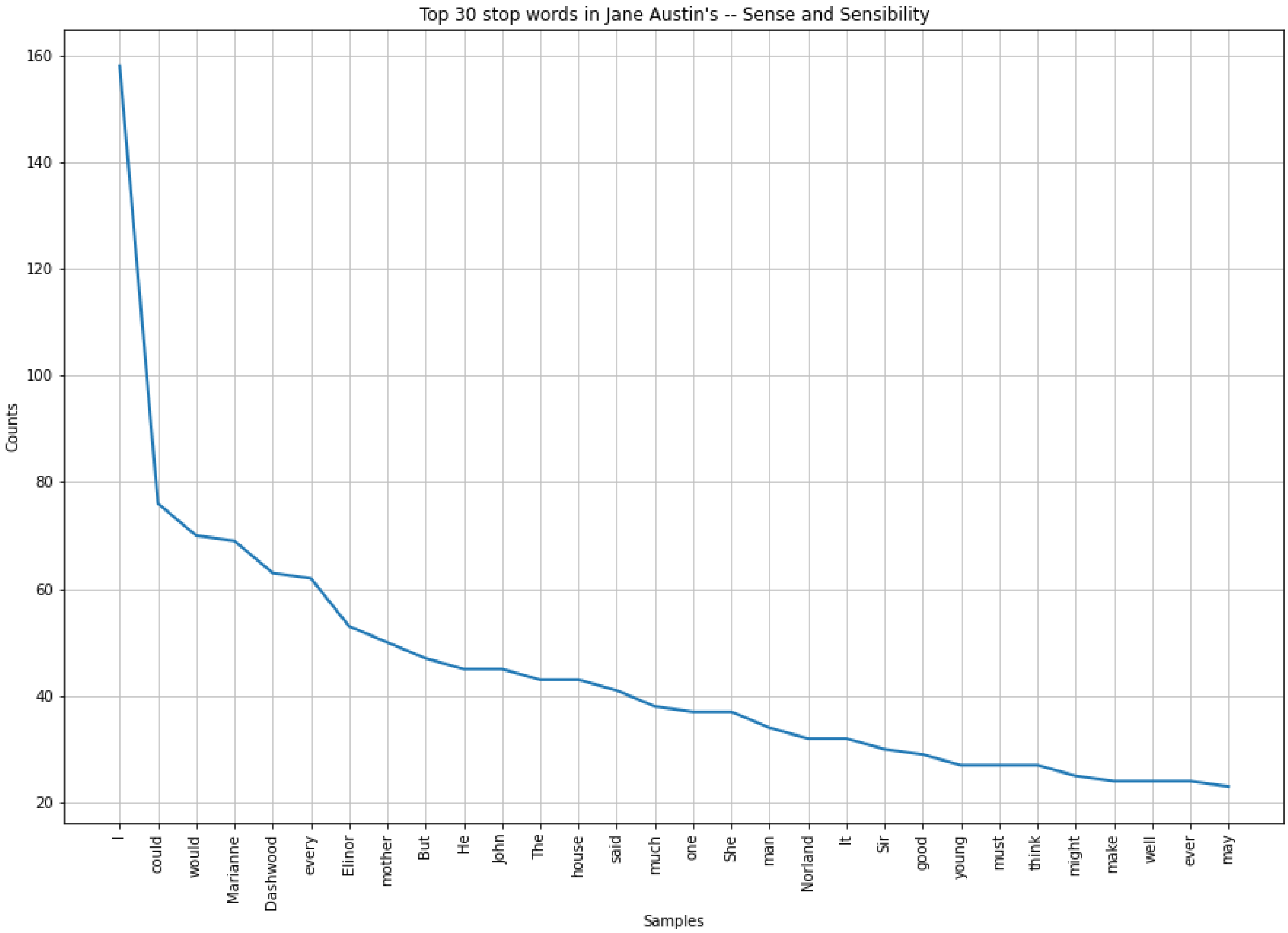
In [45]:
```
# Llegir llibre
book_file = open('//home/rusi/Escritorio/rubenIT/DataSources/sense_and_sensibility.txt',
book = book_file.read()
```

In [65]:
```
# Word freq
tokenized_book = word_tokenize(book)
# Remove punctuation
tokenized_book= [word for word in tokenized_book if word.isalnum()]
# Freq
fdist = FreqDist(tokenized_book)
# 30 most common words
fig, ax1 = plt.subplots(figsize = (15, 10))
fdist.plot(30,cumulative=False,
           title="30 most often words in Jane Austin's -- Sense and Sensibility")
plt.show()
```



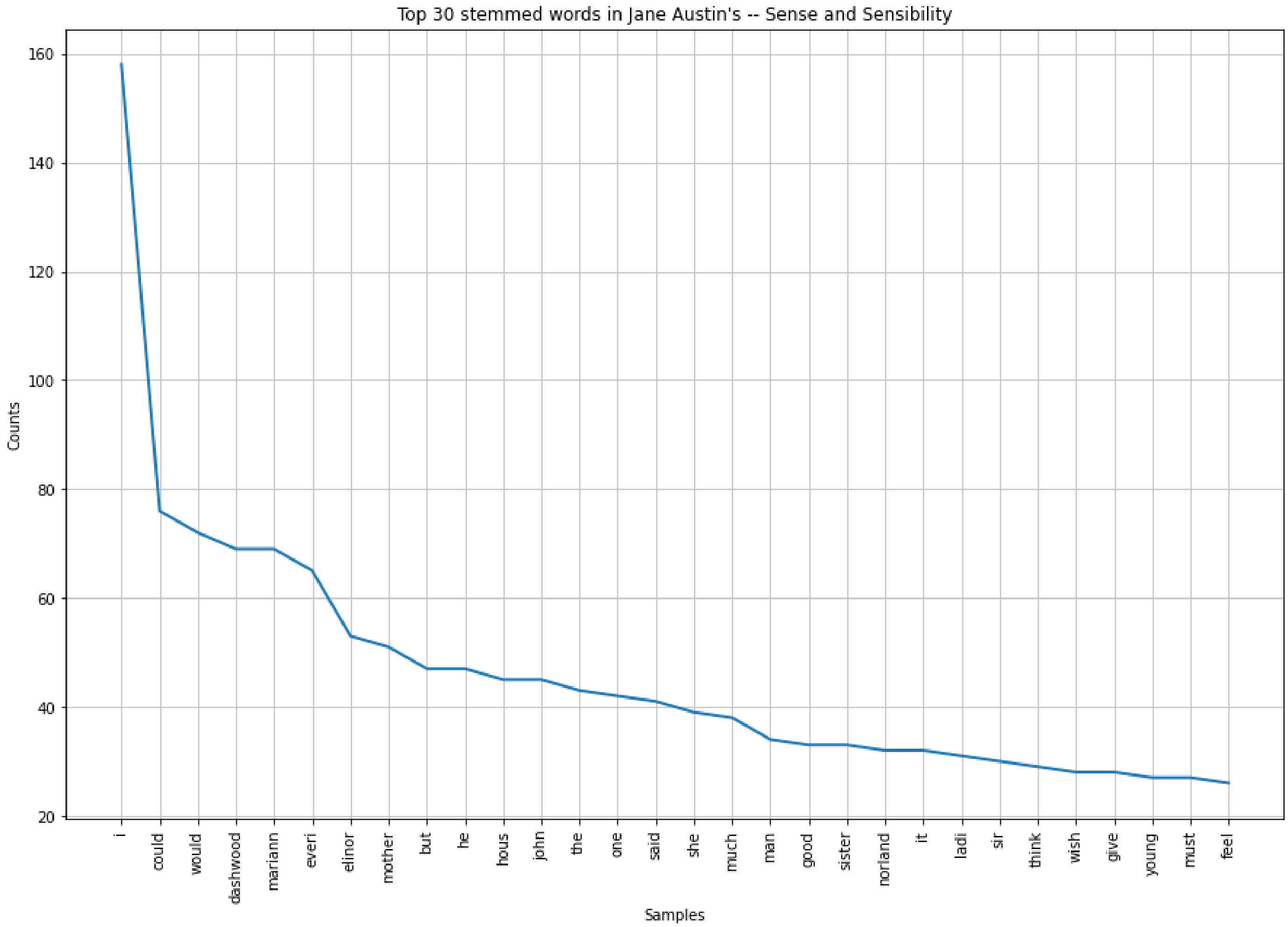
Exercici 3. Treu les stopwords i realitza stemming al teu conjunt de dades.

In [75]:
```
# Stop words list
stop_words=set(stopwords.words("english"))

# Filter
filtered_book=[]
for w in tokenized_book:
    if w not in stop_words:
        filtered_book.append(w)

# Word freq
fdist = FreqDist(filtered_book)

# 30 most common words
fig, ax1 = plt.subplots(figsize = (15, 10))
fdist.plot(30,cumulative=False,title="Top 30 stop words in Jane Austin's -- Sense and Se
plt.show()
```



In [74]:

```
# Stemming
ps = PorterStemmer()

stemmed_book=[]
for w in filtered_book:
    stemmed_book.append(ps.stem(w))

# Word freq
fdist = FreqDist(stemmed_book)

# 30 most common words
fig, ax1 = plt.subplots(figsize = (15, 10))
fdist.plot(30,cumulative=False,title="Top 30 stemmed words in Jane Austin's -- Sense and
plt.show()
```



Exercici 3. Realitza sentiment analysis al teu conjunt de dades

Sentiment Analysis

The sentiment property returns a namedtuple of the form Sentiment(polarity, subjectivity). The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

In [79]:
```
# Polarity of the text
book_sent = TextBlob(book)
book_sent.sentiment
```

Out[79]:
```
Sentiment(polarity=0.15604552058239224, subjectivity=0.535295224185392)
```

Com es tracta d'una novel·la, la quantitat de paraules emprades són molt variades, i no demostren una inclinació cap a una determinada sensació o sentiment. Per aquest motiu els paràmteres de neutralitat i subjectivitat es troben en un punt intermig.

In [ ]: