Tarea 4 Machine Learning Javiera San Martín

July 15, 2023

Section 6: Machine Learning

0.1 Housekeeping and Data

[1]:
```
pip install eli5
```

```
Requirement already satisfied: eli5 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (0.13.0)
Requirement already satisfied: numpy>=1.9.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from eli5) (1.20.1)
Requirement already satisfied: jinja2>=3.0.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from eli5) (3.1.2)
Requirement already satisfied: attrs>17.1.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from eli5) (20.3.0)
Requirement already satisfied: scipy in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from eli5) (1.6.2)
Requirement already satisfied: six in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from eli5) (1.15.0)
Requirement already satisfied: scikit-learn>=0.20 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from eli5) (1.2.2)
Requirement already satisfied: graphviz in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from eli5) (0.20.1)
Requirement already satisfied: tabulate>=0.7.7 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from eli5) (0.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
jinja2>=3.0.0->eli5) (2.1.3)
Requirement already satisfied: joblib>=1.1.1 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from scikit-
learn>=0.20->eli5) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from scikit-
learn>=0.20->eli5) (2.1.0)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
pip install pywaffle
```

```
Requirement already satisfied: pywaffle in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (1.1.0)
```

```
Requirement already satisfied: fontawesomefree in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from pywaffle)
(6.4.0)
Requirement already satisfied: matplotlib in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from pywaffle)
(3.3.4)
Requirement already satisfied: python-dateutil>=2.1 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib->pywaffle) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib->pywaffle) (1.3.1)
Requirement already satisfied: numpy>=1.15 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib->pywaffle) (1.20.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib->pywaffle) (2.4.7)
Requirement already satisfied: cycler>=0.10 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib->pywaffle) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib->pywaffle) (8.2.0)
Requirement already satisfied: six in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
cycler>=0.10->matplotlib->pywaffle) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

[3]:
```
pip install yellowbrick
```

```
Requirement already satisfied: yellowbrick in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (1.5)
Requirement already satisfied: numpy>=1.16.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from yellowbrick)
(1.20.1)
Requirement already satisfied: cycler>=0.10.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from yellowbrick)
(0.10.0)
Requirement already satisfied: scipy>=1.0.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from yellowbrick)
(1.6.2)
Requirement already satisfied: scikit-learn>=1.0.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from yellowbrick)
(1.2.2)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from yellowbrick)
(3.3.4)
```

```
Requirement already satisfied: six in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
cycler>=0.10.0->yellowbrick) (1.15.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.1)
Requirement already satisfied: pillow>=6.2.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (8.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.4.7)
Requirement already satisfied: joblib>=1.1.1 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from scikit-
learn>=1.0.0->yellowbrick) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/macbookair/opt/anaconda3/lib/python3.8/site-packages (from scikit-
learn>=1.0.0->yellowbrick) (2.1.0)
Note: you may need to restart the kernel to use updated packages.
```

[57]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import eli5
from matplotlib import pyplot as plt
from numpy import mean
from numpy import std
from numpy import absolute

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import RepeatedKFold
from sklearn.linear_model import Lasso
from sklearn.linear_model import LassoCV
from sklearn.metrics import mean_squared_error
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.ensemble import StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.svm import LinearSVC
from pywaffle import Waffle
from sklearn.impute import SimpleImputer

import yellowbrick
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import davies_bouldin_score, silhouette_score,␣
 ↪calinski_harabasz_score
from yellowbrick.cluster import KElbowVisualizer, SilhouetteVisualizer
from yellowbrick.style import set_palette
from yellowbrick.contrib.wrapper import wrap

%matplotlib inline
```

0.2 Lasso Regression (Regularization)

Pregunta 1 Utilizando el set de datos *junaeb2.csv* realice una regresion para predecir la variable *imce* con regularizacion via Lasso con cross-validation. Muestre que sus resultados son robustos a la seleccion de hiperparametros y compute una metrica de calidad de ajuste del modelo.

[58]:
```
df_Junaeb=pd.read_csv('../data/junaeb2.csv')
```

[59]:
```
# Check for missing data
print("Valores de Null en el Dataframe:")
print(df_Junaeb.isnull().sum())
```

```
Valores de Null en el Dataframe:
sexo            0
edad            0
imce            0
vive_padre      0
vive_madre      0
sk1             0
sk2             0
sk3             0
sk4             0
sk5             0
sk6             0
sk7             0
sk8             0
sk9             0
sk10            0
sk11            0
sk12            0
sk13            0
```

```
act_fisica    1435
area             0
educm          551
educp            0
madre_work       0
dtype: int64
```

[60]:
```
df_Junaeb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41854 entries, 0 to 41853
Data columns (total 23 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   sexo        41854 non-null  int64
 1   edad        41854 non-null  int64
 2   imce        41854 non-null  float64
 3   vive_padre  41854 non-null  int64
 4   vive_madre  41854 non-null  int64
 5   sk1         41854 non-null  int64
 6   sk2         41854 non-null  int64
 7   sk3         41854 non-null  int64
 8   sk4         41854 non-null  int64
 9   sk5         41854 non-null  int64
 10  sk6         41854 non-null  int64
 11  sk7         41854 non-null  int64
 12  sk8         41854 non-null  int64
 13  sk9         41854 non-null  int64
 14  sk10        41854 non-null  int64
 15  sk11        41854 non-null  int64
 16  sk12        41854 non-null  int64
 17  sk13        41854 non-null  int64
 18  act_fisica  40419 non-null  float64
 19  area        41854 non-null  int64
 20  educm       41303 non-null  float64
 21  educp       41854 non-null  int64
 22  madre_work  41854 non-null  int64
dtypes: float64(3), int64(20)
memory usage: 7.3 MB
```

[61]:
```
df_Junaeb.dropna(subset=['act_fisica', 'educm'], inplace=True)
```

[62]:
```
target = df_Junaeb.imce
features = df_Junaeb.drop('imce', axis=1)
features.describe()
```

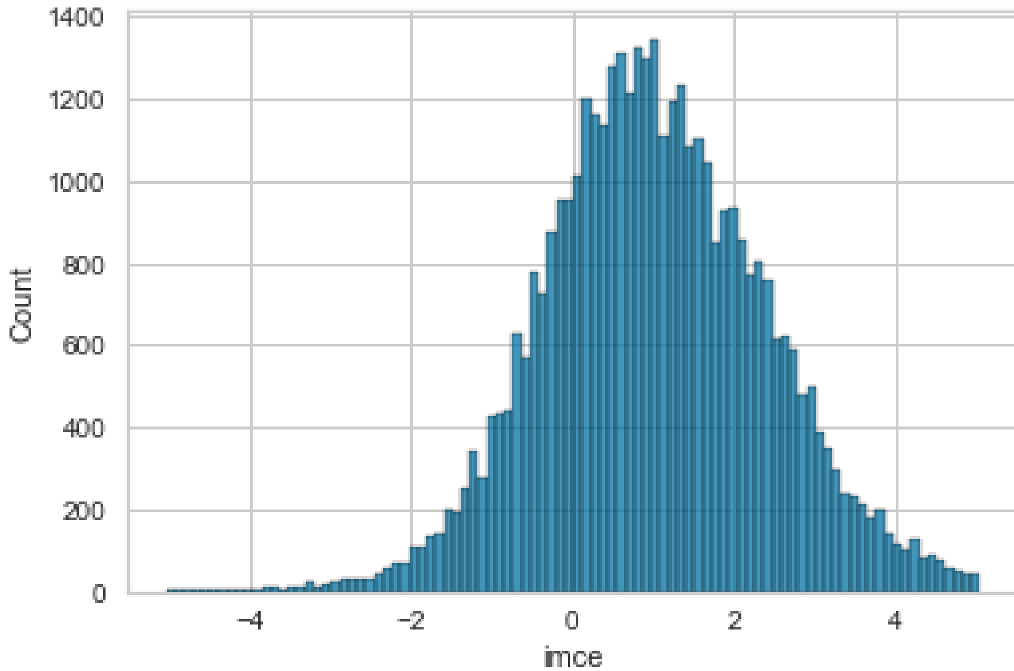
[62]:
```
               sexo          edad    vive_padre    vive_madre           sk1  \
count  39898.000000  39898.000000  39898.000000  39898.000000  39898.000000
```

```
mean        0.552409     83.022006      0.719284      0.975713      1.111885
std         0.497252      3.938669      0.450246      0.164488      0.385352
min         0.000000     62.000000      0.000000      0.000000      1.000000
25%         0.000000     81.000000      0.000000      1.000000      1.000000
50%         1.000000     82.000000      1.000000      1.000000      1.000000
75%         1.000000     84.000000      1.000000      1.000000      1.000000
max         1.000000    107.000000      2.000000      2.000000      5.000000

                sk2           sk3           sk4           sk5           sk6  \
count  39898.000000  39898.000000  39898.000000  39898.000000  39898.000000
mean       1.391874      1.263271      1.256730      1.271793      1.491002
std        0.653606      0.583646      0.578441      0.567844      0.739283
min        1.000000      1.000000      1.000000      1.000000      1.000000
25%        1.000000      1.000000      1.000000      1.000000      1.000000
50%        1.000000      1.000000      1.000000      1.000000      1.000000
75%        2.000000      1.000000      1.000000      1.000000      2.000000
max        5.000000      5.000000      5.000000      5.000000      5.000000

       …           sk9          sk10          sk11          sk12  \
count  …  39898.000000  39898.000000  39898.000000  39898.000000
mean   …      1.337862      1.877087      1.390596      1.502958
std    …      0.669773      0.950319      0.674868      0.800377
min    …      1.000000      1.000000      1.000000      1.000000
25%    …      1.000000      1.000000      1.000000      1.000000
50%    …      1.000000      2.000000      1.000000      1.000000
75%    …      2.000000      3.000000      2.000000      2.000000
max    …      5.000000      5.000000      5.000000      5.000000

               sk13    act_fisica          area         educm         educp  \
count  39898.000000  39898.000000  39898.000000  39898.000000  39898.000000
mean       1.708030      2.552409      0.911976     13.015916     12.947942
std        0.995917      1.069471      0.283334      3.365582      3.452305
min        1.000000      1.000000      0.000000      0.000000      0.000000
25%        1.000000      2.000000      1.000000     11.000000     11.000000
50%        1.000000      2.000000      1.000000     13.000000     13.000000
75%        2.000000      3.000000      1.000000     15.000000     14.000000
max        5.000000      5.000000      1.000000     22.000000     22.000000

         madre_work
count  39898.000000
mean       0.098150
std        0.941687
min       -1.000000
25%       -1.000000
50%        0.000000
75%        1.000000
max        1.000000
```

```
[8 rows x 22 columns]
```

[63]:
```
sns.histplot(data=target)
```

[63]:
```
<AxesSubplot:xlabel='imce', ylabel='Count'>
```



[64]:
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target,
    test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[65]:
```
# Train the Lasso regression model
lasso=Lasso(alpha=0.002,
            max_iter=10000,
            tol=0.0001)
lasso.fit(X_train_scaled, y_train)
```

[65]:
```
Lasso(alpha=0.002, max_iter=10000)
```

[67]:
```python
# Predict on the test set
y_pred = lasso.predict(X_test_scaled)

# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("Root Mean Squared Error:", rmse)
```

```
Root Mean Squared Error: 1.3695356353554322
```

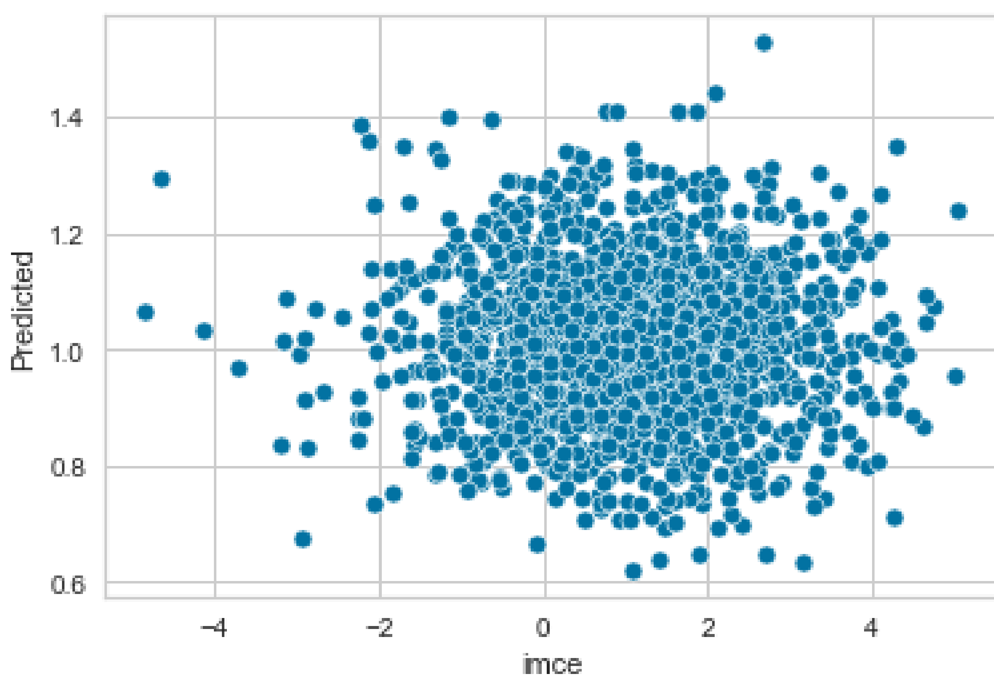
Al dar un error cuadratico medio es muy lato por lo que los datos no se ajustan de manera adecuada al modelo.

[68]:
```python
# define model evaluation method
cv = RepeatedKFold(n_splits=100, n_repeats=3, random_state=1)
# cross validation scores
scores = cross_val_score(lasso, X_train_scaled, y_train, cv=cv, n_jobs=-1,␣
 ↪scoring='neg_root_mean_squared_error')
scores=absolute(scores)
print('Mean RMSE: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Mean RMSE: 1.372 (0.061)
```

[69]:
```python
y_pred=pd.DataFrame(y_pred)
y_pred.rename(columns={0 :'Predicted'}, inplace=True )
test = pd.concat([y_test, y_pred], axis=1, join='inner')
sns.scatterplot(data=test, x='imce', y='Predicted')
```

[69]: <AxesSubplot:xlabel='imce', ylabel='Predicted'>



Se observa que los datos estan distribuidos especialmente entre los rangos de 0 a 2 para imce y de 0,8 a 1,2 para los valores predichos. En la tabla de abajo se observa que los datos más relevantes son "sexo", "sk8", "sk7" y "madre_work"

[70]:
```
eli5.show_weights(lasso, top=-1, feature_names = X_train.columns.tolist())
```

[70]:
```
<IPython.core.display.HTML object>
```

[71]:
```
# define model
model = LassoCV(n_alphas=100, cv=cv, n_jobs=-1, max_iter=10000)
# fit model
model.fit(X_train_scaled, y_train)
# summarize chosen configuration
print('alpha: %f' % model.alpha_)
```

```
alpha: 0.002316
```

El valor de alpha es de 0,002

0.3 Clasification

0.3.1 Random Forest

Pregunta 2 Utilizando el set de datos *charls2.csv* realice una clasificacion de la variable *retired* usando Random Forest sobre las demas variables del dataset con cross-validation. Muestre que sus resultados son sensibles a la seleccion de hiperparametros y compute una metrica de calidad de ajuste del modelo.

[18]:
```
df_charls2=pd.read_csv('../data/charls2.csv')
```

[19]:
```
df_charls2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9456 entries, 0 to 9455
Data columns (total 15 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       9456 non-null   int64
 1   cesd      8802 non-null   float64
 2   child     9456 non-null   int64
 3   drinkly   9456 non-null   object
 4   female    9456 non-null   int64
 5   hrsusu    9456 non-null   float64
 6   hsize     9456 non-null   int64
 7   intmonth  9456 non-null   int64
 8   married   9456 non-null   int64
 9   retage    9456 non-null   int64
 10  retin     9456 non-null   int64
```

```
 11   retired    9456 non-null   int64
 12   schadj     9456 non-null   int64
 13   urban      9456 non-null   int64
 14   wealth     8590 non-null   float64
dtypes: float64(3), int64(11), object(1)
memory usage: 1.1+ MB
```

[20]:
```
df_charls2.drop(df_charls2[df_charls2['drinkly'].str.contains('.r')].index,␣
 ↪inplace=True)
df_charls2.drop(df_charls2[df_charls2['drinkly'].str.contains('.m')].index,␣
 ↪inplace=True)
df_charls2.drop(df_charls2[df_charls2['drinkly'].str.contains('.d')].index,␣
 ↪inplace=True)
df_charls2['drinkly'] = df_charls2['drinkly'].astype(int)
df_charls2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9418 entries, 0 to 9455
Data columns (total 15 columns):
 #    Column     Non-Null Count  Dtype
---   ------     --------------  -----
 0    age        9418 non-null   int64
 1    cesd       8802 non-null   float64
 2    child      9418 non-null   int64
 3    drinkly    9418 non-null   int64
 4    female     9418 non-null   int64
 5    hrsusu     9418 non-null   float64
 6    hsize      9418 non-null   int64
 7    intmonth   9418 non-null   int64
 8    married    9418 non-null   int64
 9    retage     9418 non-null   int64
 10   retin      9418 non-null   int64
 11   retired    9418 non-null   int64
 12   schadj     9418 non-null   int64
 13   urban      9418 non-null   int64
 14   wealth     8585 non-null   float64
dtypes: float64(3), int64(12)
memory usage: 1.1 MB
```

[21]:
```
df_charls2.dropna(subset=['cesd', 'wealth'], inplace=True)
```

[22]:
```
target = df_charls2.retired
features = df_charls2.drop('retired', axis=1)
features.describe()
```

[22]:
```
               age         cesd        child      drinkly       female  \
count  8080.000000  8080.000000  8080.000000  8080.000000  8080.000000
mean     58.164356     9.112376     2.780074     0.325990     0.535272
```

```
std        9.374956     6.481237     1.397316     0.468773     0.498785
min       21.000000     0.000000     0.000000     0.000000     0.000000
25%       51.000000     4.000000     2.000000     0.000000     0.000000
50%       57.000000     8.000000     3.000000     0.000000     1.000000
75%       64.000000    13.000000     4.000000     1.000000     1.000000
max       95.000000    30.000000    10.000000     1.000000     1.000000

            hrsusu        hsize     intmonth      married       retage  \
count  8080.000000  8080.000000  8080.000000  8080.000000  8080.000000
mean      2.566736     3.764233     7.506931     0.879084     1.480817
std       1.788298     1.823838     1.001893     0.326050     4.206412
min       0.000000     1.000000     1.000000     0.000000     0.000000
25%       0.000000     2.000000     7.000000     1.000000     0.000000
50%       3.496508     4.000000     7.000000     1.000000     0.000000
75%       4.025352     5.000000     8.000000     1.000000     0.000000
max       5.123964    16.000000    12.000000     1.000000    37.000000

             retin       schadj        urban          wealth
count  8080.000000  8080.000000  8080.000000     8080.000000
mean      0.163861     4.039851     0.212005     1479.488490
std       0.370173     3.545666     0.408754    43479.865654
min       0.000000     0.000000     0.000000 -1000000.000000
25%       0.000000     0.000000     0.000000        0.000000
50%       0.000000     4.000000     0.000000      400.000000
75%       0.000000     8.000000     0.000000     2200.000000
max       1.000000    16.000000     1.000000   900100.000000
```

[23]:

```python
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target,␣
 ↪test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[24]:

```python
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Training the classifier
rf.fit(X_train, y_train)

# Making predictions on the test set
y_pred = rf.predict(X_test)

# Evaluating the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9047029702970297
```

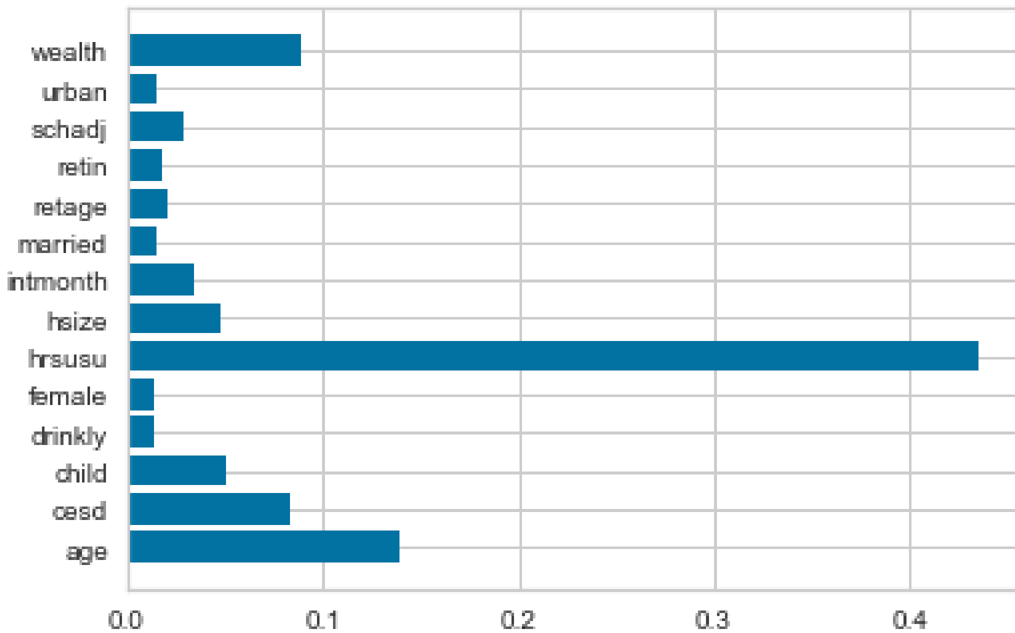
La presición del modelo es del 90,47%

[25]:
```
confusion_matrix(y_test, y_pred)
```

[25]:
```
array([[1220,   93],
       [  61,  242]])
```

[26]:
```
plt.barh(features.columns, rf.feature_importances_)
```

[26]:
```
<BarContainer object of 14 artists>
```



Ajustamos los hiperparametros.

[27]:
```
from sklearn.model_selection import GridSearchCV

# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [4, 8, 12],
    'max_features': [6, 8],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300]
}
```

```python
# Create a based model
rf = RandomForestClassifier()

# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                          cv = 10, n_jobs = -1, verbose = 2)
```

[28]:
```python
# Fit the grid search to the data
grid_search.fit(X_train_scaled, y_train)
#grid_search.best_params_
#best_grid = grid_search.best_estimator_
```

```
Fitting 10 folds for each of 162 candidates, totalling 1620 fits
```

[28]:
```
GridSearchCV(cv=10, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'bootstrap': [True], 'max_depth': [4, 8, 12],
                         'max_features': [6, 8], 'min_samples_leaf': [3, 4, 5],
                         'min_samples_split': [8, 10, 12],
                         'n_estimators': [100, 200, 300]},
             verbose=2)
```

[29]:
```python
results_df = pd.DataFrame(grid_search.cv_results_)
results_df = results_df.sort_values(by=["rank_test_score"])
results_df = results_df.set_index(
    results_df["params"].apply(lambda x: "_".join(str(val) for val in x.
 ↪values()))
).rename_axis("kernel")
results_df[["params", "rank_test_score", "mean_test_score", "std_test_score"]]
```

[29]:
```
                                                              params  \
kernel
True_12_6_4_10_300  {'bootstrap': True, 'max_depth': 12, 'max_feat…
True_12_6_3_8_300   {'bootstrap': True, 'max_depth': 12, 'max_feat…
True_8_6_5_8_100    {'bootstrap': True, 'max_depth': 8, 'max_featu…
True_12_6_4_12_100  {'bootstrap': True, 'max_depth': 12, 'max_feat…
True_12_6_4_10_100  {'bootstrap': True, 'max_depth': 12, 'max_feat…
…                                                                 …
True_4_8_4_12_100   {'bootstrap': True, 'max_depth': 4, 'max_featu…
True_4_8_3_10_100   {'bootstrap': True, 'max_depth': 4, 'max_featu…
True_4_6_4_10_200   {'bootstrap': True, 'max_depth': 4, 'max_featu…
True_8_8_5_10_100   {'bootstrap': True, 'max_depth': 8, 'max_featu…
True_4_8_4_10_100   {'bootstrap': True, 'max_depth': 4, 'max_featu…

                    rank_test_score  mean_test_score  std_test_score
kernel
True_12_6_4_10_300                1         0.907023        0.010344
True_12_6_3_8_300                 2         0.906867        0.011270
True_8_6_5_8_100                  3         0.906401        0.010367
```

```
True_12_6_4_12_100                  4        0.906247        0.012611
True_12_6_4_10_100                  5        0.905939        0.012815
…                               …              …             …
True_4_8_4_12_100                 158        0.901918        0.008162
True_4_8_3_10_100                 159        0.901763        0.008625
True_4_6_4_10_200                 160        0.901762        0.007764
True_8_8_5_10_100                 161        0.901761        0.010740
True_4_8_4_10_100                 162        0.901453        0.008487

[162 rows x 4 columns]
```

0.3.2 Boosting

Pregunta 3 Repita el analisis de la Pregunta 2 usando Stacking, con tres modelos (Random Forest, Gradient Boosting y SVM). Muestre que sus resultados son robustos a la seleccion de hiperparametros y compute una metrica de calidad de ajuste del modelo.

[30]:
```python
gb = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=42).fit(X_train, y_train)

# Making predictions on the test set
y_pred = gb.predict(X_test)

# Evaluating the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9065594059405940594
```

La presición del modelo es del 90,65%

[31]:
```python
## Considerando Grid Search
gb = GradientBoostingClassifier()
parameters = {'learning_rate': [0.02,0.03,0.04],
                  'subsample'    : [0.9, 0.5],
                  'n_estimators' : [100,200,300],
                  'max_depth'    : [4,6,8]}

grid_GBR = GridSearchCV(estimator=gb, param_grid = parameters, cv = 8, n_jobs=-1)
grid_GBR.fit(X_train_scaled, y_train)
```

[31]:
```
GridSearchCV(cv=8, estimator=GradientBoostingClassifier(), n_jobs=-1,
             param_grid={'learning_rate': [0.02, 0.03, 0.04],
                         'max_depth': [4, 6, 8],
                         'n_estimators': [100, 200, 300],
                         'subsample': [0.9, 0.5]})
```

[32]:
```python
results_df = pd.DataFrame(grid_GBR.cv_results_)
results_df = results_df.sort_values(by=["rank_test_score"])
results_df = results_df.set_index(
    results_df["params"].apply(lambda x: "_".join(str(val) for val in x.
 ↪values()))
).rename_axis("kernel")
results_df[["params", "rank_test_score", "mean_test_score", "std_test_score"]]
```

[32]:
```
                                                                      params  \
kernel
0.03_6_100_0.5  {'learning_rate': 0.03, 'max_depth': 6, 'n_est…
0.02_6_100_0.9  {'learning_rate': 0.02, 'max_depth': 6, 'n_est…
0.02_4_100_0.9  {'learning_rate': 0.02, 'max_depth': 4, 'n_est…
0.04_4_200_0.9  {'learning_rate': 0.04, 'max_depth': 4, 'n_est…
0.04_4_300_0.5  {'learning_rate': 0.04, 'max_depth': 4, 'n_est…
0.02_6_200_0.9  {'learning_rate': 0.02, 'max_depth': 6, 'n_est…
0.02_6_100_0.5  {'learning_rate': 0.02, 'max_depth': 6, 'n_est…
0.03_4_100_0.5  {'learning_rate': 0.03, 'max_depth': 4, 'n_est…
0.02_4_100_0.5  {'learning_rate': 0.02, 'max_depth': 4, 'n_est…
0.02_4_300_0.5  {'learning_rate': 0.02, 'max_depth': 4, 'n_est…
0.02_6_200_0.5  {'learning_rate': 0.02, 'max_depth': 6, 'n_est…
0.03_6_100_0.9  {'learning_rate': 0.03, 'max_depth': 6, 'n_est…
0.04_6_100_0.5  {'learning_rate': 0.04, 'max_depth': 6, 'n_est…
0.03_4_100_0.9  {'learning_rate': 0.03, 'max_depth': 4, 'n_est…
0.03_4_200_0.9  {'learning_rate': 0.03, 'max_depth': 4, 'n_est…
0.02_4_200_0.5  {'learning_rate': 0.02, 'max_depth': 4, 'n_est…
0.02_6_300_0.5  {'learning_rate': 0.02, 'max_depth': 6, 'n_est…
0.04_4_100_0.5  {'learning_rate': 0.04, 'max_depth': 4, 'n_est…
0.03_4_200_0.5  {'learning_rate': 0.03, 'max_depth': 4, 'n_est…
0.04_6_100_0.9  {'learning_rate': 0.04, 'max_depth': 6, 'n_est…
0.03_6_200_0.5  {'learning_rate': 0.03, 'max_depth': 6, 'n_est…
0.04_6_200_0.5  {'learning_rate': 0.04, 'max_depth': 6, 'n_est…
0.02_8_200_0.5  {'learning_rate': 0.02, 'max_depth': 8, 'n_est…
0.03_4_300_0.9  {'learning_rate': 0.03, 'max_depth': 4, 'n_est…
0.03_4_300_0.5  {'learning_rate': 0.03, 'max_depth': 4, 'n_est…
0.02_6_300_0.9  {'learning_rate': 0.02, 'max_depth': 6, 'n_est…
0.02_4_300_0.9  {'learning_rate': 0.02, 'max_depth': 4, 'n_est…
0.04_8_100_0.9  {'learning_rate': 0.04, 'max_depth': 8, 'n_est…
0.03_8_100_0.9  {'learning_rate': 0.03, 'max_depth': 8, 'n_est…
0.02_8_100_0.9  {'learning_rate': 0.02, 'max_depth': 8, 'n_est…
0.02_8_100_0.5  {'learning_rate': 0.02, 'max_depth': 8, 'n_est…
0.04_4_100_0.9  {'learning_rate': 0.04, 'max_depth': 4, 'n_est…
0.04_4_200_0.5  {'learning_rate': 0.04, 'max_depth': 4, 'n_est…
0.02_4_200_0.9  {'learning_rate': 0.02, 'max_depth': 4, 'n_est…
0.04_6_300_0.5  {'learning_rate': 0.04, 'max_depth': 6, 'n_est…
0.03_6_200_0.9  {'learning_rate': 0.03, 'max_depth': 6, 'n_est…
0.04_4_300_0.9  {'learning_rate': 0.04, 'max_depth': 4, 'n_est…
```

0.02_8_200_0.9	{'learning_rate': 0.02, 'max_depth': 8, 'n_est…
0.02_8_300_0.9	{'learning_rate': 0.02, 'max_depth': 8, 'n_est…
0.03_8_200_0.9	{'learning_rate': 0.03, 'max_depth': 8, 'n_est…
0.03_8_100_0.5	{'learning_rate': 0.03, 'max_depth': 8, 'n_est…
0.03_6_300_0.9	{'learning_rate': 0.03, 'max_depth': 6, 'n_est…
0.04_8_200_0.5	{'learning_rate': 0.04, 'max_depth': 8, 'n_est…
0.02_8_300_0.5	{'learning_rate': 0.02, 'max_depth': 8, 'n_est…
0.04_8_300_0.5	{'learning_rate': 0.04, 'max_depth': 8, 'n_est…
0.03_6_300_0.5	{'learning_rate': 0.03, 'max_depth': 6, 'n_est…
0.04_8_100_0.5	{'learning_rate': 0.04, 'max_depth': 8, 'n_est…
0.03_8_300_0.5	{'learning_rate': 0.03, 'max_depth': 8, 'n_est…
0.04_6_300_0.9	{'learning_rate': 0.04, 'max_depth': 6, 'n_est…
0.04_8_300_0.9	{'learning_rate': 0.04, 'max_depth': 8, 'n_est…
0.03_8_200_0.5	{'learning_rate': 0.03, 'max_depth': 8, 'n_est…
0.04_6_200_0.9	{'learning_rate': 0.04, 'max_depth': 6, 'n_est…
0.04_8_200_0.9	{'learning_rate': 0.04, 'max_depth': 8, 'n_est…
0.03_8_300_0.9	{'learning_rate': 0.03, 'max_depth': 8, 'n_est…

	rank_test_score	mean_test_score	std_test_score
kernel			
0.03_6_100_0.5	1	0.903775	0.006333
0.02_6_100_0.9	2	0.903465	0.009095
0.02_4_100_0.9	3	0.903156	0.007748
0.04_4_200_0.9	4	0.903001	0.007372
0.04_4_300_0.5	4	0.903001	0.010627
0.02_6_200_0.9	6	0.902847	0.007217
0.02_6_100_0.5	7	0.902847	0.007803
0.03_4_100_0.5	8	0.902692	0.006155
0.02_4_100_0.5	9	0.902692	0.007628
0.02_4_300_0.5	9	0.902692	0.007653
0.02_6_200_0.5	9	0.902692	0.006155
0.03_6_100_0.9	9	0.902692	0.008640
0.04_6_100_0.5	13	0.902537	0.008790
0.03_4_100_0.9	14	0.902382	0.007148
0.03_4_200_0.9	14	0.902382	0.007121
0.02_4_200_0.5	14	0.902382	0.007255
0.02_6_300_0.5	17	0.902228	0.010149
0.04_4_100_0.5	18	0.902073	0.006734
0.03_4_200_0.5	19	0.901918	0.008227
0.04_6_100_0.9	20	0.901918	0.006884
0.03_6_200_0.5	21	0.901764	0.008043
0.04_6_200_0.5	21	0.901764	0.008347
0.02_8_200_0.5	23	0.901764	0.008528
0.03_4_300_0.9	24	0.901609	0.007852
0.03_4_300_0.5	24	0.901609	0.008707
0.02_6_300_0.9	24	0.901609	0.007083
0.02_4_300_0.9	24	0.901609	0.007243

0.04_8_100_0.9	28	0.901454	0.009156
0.03_8_100_0.9	28	0.901454	0.006309
0.02_8_100_0.9	30	0.901300	0.006828
0.02_8_100_0.5	31	0.900990	0.005569
0.04_4_100_0.9	32	0.900835	0.006677
0.04_4_200_0.5	33	0.900681	0.007598
0.02_4_200_0.9	34	0.900526	0.006605
0.04_6_300_0.5	35	0.900217	0.008859
0.03_6_200_0.9	36	0.900062	0.008039
0.04_4_300_0.9	37	0.899907	0.008562
0.02_8_200_0.9	38	0.899598	0.007838
0.02_8_300_0.9	39	0.899443	0.009400
0.03_8_200_0.9	40	0.899288	0.008185
0.03_8_100_0.5	41	0.898515	0.008348
0.03_6_300_0.9	42	0.898205	0.007103
0.04_8_200_0.5	42	0.898205	0.008133
0.02_8_300_0.5	44	0.898051	0.008255
0.04_8_300_0.5	45	0.897587	0.010714
0.03_6_300_0.5	46	0.897432	0.008913
0.04_8_100_0.5	47	0.897277	0.005672
0.03_8_300_0.5	48	0.896349	0.006394
0.04_6_300_0.9	48	0.896349	0.007967
0.04_8_300_0.9	50	0.896194	0.008335
0.03_8_200_0.5	51	0.896040	0.006250
0.04_6_200_0.9	52	0.895575	0.010753
0.04_8_200_0.9	53	0.895266	0.008662
0.03_8_300_0.9	54	0.894647	0.007935

0.3.3 Support Vector Machine

[33]:
```
from sklearn import svm

svm_m = svm.SVC()
svm_m.fit(X_train, y_train)

y_pred = svm_m.predict(X_test)

# Evaluating the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8118811881188119
```

La presición del modelo es del 81,18%

[34]:
```
# defining parameter range
param_grid = {'C': [0.01, 0.1, 1, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001, 0.00001],
```

```python
               'kernel': ['rbf']}

grid_SVM = GridSearchCV(svm_m, param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid_SVM.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
[CV 1/5] END …C=0.01, gamma=1, kernel=rbf;, score=0.817 total time=   2.2s
[CV 2/5] END …C=0.01, gamma=1, kernel=rbf;, score=0.817 total time=   4.1s
[CV 3/5] END …C=0.01, gamma=1, kernel=rbf;, score=0.817 total time=   2.0s
[CV 4/5] END …C=0.01, gamma=1, kernel=rbf;, score=0.817 total time=   1.9s
[CV 5/5] END …C=0.01, gamma=1, kernel=rbf;, score=0.817 total time=   2.0s
[CV 1/5] END …C=0.01, gamma=0.1, kernel=rbf;, score=0.817 total time=   1.8s
[CV 2/5] END …C=0.01, gamma=0.1, kernel=rbf;, score=0.817 total time=   1.8s
[CV 3/5] END …C=0.01, gamma=0.1, kernel=rbf;, score=0.817 total time=   1.8s
[CV 4/5] END …C=0.01, gamma=0.1, kernel=rbf;, score=0.817 total time=   1.8s
[CV 5/5] END …C=0.01, gamma=0.1, kernel=rbf;, score=0.817 total time=   2.0s
[CV 1/5] END …C=0.01, gamma=0.01, kernel=rbf;, score=0.817 total time=   1.4s
[CV 2/5] END …C=0.01, gamma=0.01, kernel=rbf;, score=0.817 total time=   1.4s
[CV 3/5] END …C=0.01, gamma=0.01, kernel=rbf;, score=0.817 total time=   1.4s
[CV 4/5] END …C=0.01, gamma=0.01, kernel=rbf;, score=0.817 total time=   1.4s
[CV 5/5] END …C=0.01, gamma=0.01, kernel=rbf;, score=0.817 total time=   1.4s
[CV 1/5] END …C=0.01, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.2s
[CV 2/5] END …C=0.01, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.2s
[CV 3/5] END …C=0.01, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 4/5] END …C=0.01, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.2s
[CV 5/5] END …C=0.01, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 1/5] END ..C=0.01, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 2/5] END ..C=0.01, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 3/5] END ..C=0.01, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.1s
[CV 4/5] END ..C=0.01, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.1s
[CV 5/5] END ..C=0.01, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.1s
[CV 1/5] END …C=0.01, gamma=1e-05, kernel=rbf;, score=0.817 total time=   1.0s
[CV 2/5] END …C=0.01, gamma=1e-05, kernel=rbf;, score=0.817 total time=   1.0s
[CV 3/5] END …C=0.01, gamma=1e-05, kernel=rbf;, score=0.817 total time=   0.9s
[CV 4/5] END …C=0.01, gamma=1e-05, kernel=rbf;, score=0.817 total time=   0.9s
[CV 5/5] END …C=0.01, gamma=1e-05, kernel=rbf;, score=0.817 total time=   0.9s
[CV 1/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.817 total time=   2.0s
[CV 2/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.817 total time=   2.0s
[CV 3/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.817 total time=   2.1s
[CV 4/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.817 total time=   2.2s
[CV 5/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.817 total time=   2.0s
[CV 1/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.817 total time=   2.0s
[CV 2/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.817 total time=   2.0s
[CV 3/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.817 total time=   1.9s
[CV 4/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.817 total time=   2.0s
[CV 5/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.817 total time=   2.0s
```

```
[CV 1/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.819 total time=   1.6s
[CV 2/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.819 total time=   1.9s
[CV 3/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.817 total time=   1.6s
[CV 4/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.817 total time=   1.6s
[CV 5/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.820 total time=   1.6s
[CV 1/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 2/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.4s
[CV 3/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 4/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 5/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 1/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.2s
[CV 2/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.2s
[CV 3/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 4/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.4s
[CV 5/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.2s
[CV 1/5] END …C=0.1, gamma=1e-05, kernel=rbf;, score=0.817 total time=   1.1s
[CV 2/5] END …C=0.1, gamma=1e-05, kernel=rbf;, score=0.817 total time=   1.1s
[CV 3/5] END …C=0.1, gamma=1e-05, kernel=rbf;, score=0.817 total time=   1.1s
[CV 4/5] END …C=0.1, gamma=1e-05, kernel=rbf;, score=0.817 total time=   1.1s
[CV 5/5] END …C=0.1, gamma=1e-05, kernel=rbf;, score=0.817 total time=   1.1s
[CV 1/5] END …C=1, gamma=1, kernel=rbf;, score=0.817 total time=   2.2s
[CV 2/5] END …C=1, gamma=1, kernel=rbf;, score=0.817 total time=   2.2s
[CV 3/5] END …C=1, gamma=1, kernel=rbf;, score=0.817 total time=   2.2s
[CV 4/5] END …C=1, gamma=1, kernel=rbf;, score=0.817 total time=   2.4s
[CV 5/5] END …C=1, gamma=1, kernel=rbf;, score=0.817 total time=   2.2s
[CV 1/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.822 total time=   2.1s
[CV 2/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.829 total time=   2.1s
[CV 3/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.819 total time=   2.1s
[CV 4/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.823 total time=   2.1s
[CV 5/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.824 total time=   3.4s
[CV 1/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.854 total time=   3.3s
[CV 2/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.848 total time=   2.1s
[CV 3/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.851 total time=   2.4s
[CV 4/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.855 total time=   2.7s
[CV 5/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.854 total time=   2.8s
[CV 1/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.832 total time=   1.5s
[CV 2/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.848 total time=   1.5s
[CV 3/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.835 total time=   1.5s
[CV 4/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.852 total time=   1.4s
[CV 5/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.841 total time=   1.4s
[CV 1/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.817 total time=   1.3s
[CV 2/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.815 total time=   1.3s
[CV 3/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.816 total time=   1.3s
[CV 4/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.812 total time=   1.3s
[CV 5/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.810 total time=   1.3s
[CV 1/5] END …C=1, gamma=1e-05, kernel=rbf;, score=0.814 total time=   1.3s
[CV 2/5] END …C=1, gamma=1e-05, kernel=rbf;, score=0.814 total time=   1.3s
[CV 3/5] END …C=1, gamma=1e-05, kernel=rbf;, score=0.814 total time=   1.3s
```

```
[CV 4/5] END …C=1, gamma=1e-05, kernel=rbf;, score=0.815 total time=   1.3s
[CV 5/5] END …C=1, gamma=1e-05, kernel=rbf;, score=0.814 total time=   1.3s
[CV 1/5] END …C=100, gamma=1, kernel=rbf;, score=0.817 total time=   2.5s
[CV 2/5] END …C=100, gamma=1, kernel=rbf;, score=0.817 total time=   2.8s
[CV 3/5] END …C=100, gamma=1, kernel=rbf;, score=0.817 total time=   2.5s
[CV 4/5] END …C=100, gamma=1, kernel=rbf;, score=0.817 total time=   2.4s
[CV 5/5] END …C=100, gamma=1, kernel=rbf;, score=0.817 total time=   2.4s
[CV 1/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.825 total time=   2.5s
[CV 2/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.837 total time=   2.4s
[CV 3/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.828 total time=   2.4s
[CV 4/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.827 total time=   2.3s
[CV 5/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.830 total time=   2.3s
[CV 1/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.844 total time=   2.4s
[CV 2/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.848 total time=   2.5s
[CV 3/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.849 total time=   2.7s
[CV 4/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.841 total time=   2.1s
[CV 5/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.835 total time=   2.4s
[CV 1/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.862 total time=   1.7s
[CV 2/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.859 total time=   1.6s
[CV 3/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.865 total time=   1.6s
[CV 4/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.849 total time=   1.6s
[CV 5/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.856 total time=   1.6s
[CV 1/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.869 total time=   1.4s
[CV 2/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.863 total time=   1.6s
[CV 3/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.872 total time=   1.4s
[CV 4/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.862 total time=   1.9s
[CV 5/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.868 total time=   1.9s
[CV 1/5] END …C=100, gamma=1e-05, kernel=rbf;, score=0.858 total time=   1.9s
[CV 2/5] END …C=100, gamma=1e-05, kernel=rbf;, score=0.869 total time=   1.7s
[CV 3/5] END …C=100, gamma=1e-05, kernel=rbf;, score=0.862 total time=   1.4s
[CV 4/5] END …C=100, gamma=1e-05, kernel=rbf;, score=0.852 total time=   1.4s
[CV 5/5] END …C=100, gamma=1e-05, kernel=rbf;, score=0.858 total time=   1.4s
[CV 1/5] END …C=1000, gamma=1, kernel=rbf;, score=0.817 total time=   2.4s
[CV 2/5] END …C=1000, gamma=1, kernel=rbf;, score=0.817 total time=   2.6s
[CV 3/5] END …C=1000, gamma=1, kernel=rbf;, score=0.817 total time=   2.4s
[CV 4/5] END …C=1000, gamma=1, kernel=rbf;, score=0.817 total time=   2.4s
[CV 5/5] END …C=1000, gamma=1, kernel=rbf;, score=0.817 total time=   2.4s
[CV 1/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.825 total time=   2.3s
[CV 2/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.837 total time=   2.3s
[CV 3/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.828 total time=   2.5s
[CV 4/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.827 total time=   2.3s
[CV 5/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.830 total time=   2.3s
[CV 1/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.843 total time=   2.5s
[CV 2/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.848 total time=   2.5s
[CV 3/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.852 total time=   2.6s
[CV 4/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.841 total time=   2.4s
[CV 5/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.827 total time=   2.9s
[CV 1/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.855 total time=   2.9s
```

```
[CV 2/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.855 total time=   3.3s
[CV 3/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.869 total time=   3.2s
[CV 4/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.843 total time=   3.1s
[CV 5/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.851 total time=   3.1s
[CV 1/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.867 total time=   2.2s
[CV 2/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.865 total time=   1.9s
[CV 3/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.875 total time=   2.0s
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.855 total time=   2.0s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.868 total time=   2.1s
[CV 1/5] END …C=1000, gamma=1e-05, kernel=rbf;, score=0.880 total time=   2.3s
[CV 2/5] END …C=1000, gamma=1e-05, kernel=rbf;, score=0.873 total time=   2.5s
[CV 3/5] END …C=1000, gamma=1e-05, kernel=rbf;, score=0.877 total time=   1.9s
[CV 4/5] END …C=1000, gamma=1e-05, kernel=rbf;, score=0.855 total time=   2.5s
[CV 5/5] END …C=1000, gamma=1e-05, kernel=rbf;, score=0.878 total time=   2.4s
```

[34]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.01, 0.1, 1, 100, 1000],
                         'gamma': [1, 0.1, 0.01, 0.001, 0.0001, 1e-05],
                         'kernel': ['rbf']},
             verbose=3)
```

[35]:
```python
results_df = pd.DataFrame(grid_SVM.cv_results_)
results_df = results_df.sort_values(by=["rank_test_score"])
results_df = results_df.set_index(
    results_df["params"].apply(lambda x: "_".join(str(val) for val in x.
 ↪values()))
).rename_axis("kernel")
results_df[["params", "rank_test_score", "mean_test_score", "std_test_score"]]
```

[35]:
```
                                                       params  \
kernel
1000_1e-05_rbf    {'C': 1000, 'gamma': 1e-05, 'kernel': 'rbf'}
100_0.0001_rbf    {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
1000_0.0001_rbf  {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}
100_1e-05_rbf      {'C': 100, 'gamma': 1e-05, 'kernel': 'rbf'}
100_0.001_rbf      {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
1000_0.001_rbf    {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
1_0.01_rbf            {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
100_0.01_rbf        {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
1000_0.01_rbf      {'C': 1000, 'gamma': 0.01, 'kernel': 'rbf'}
1_0.001_rbf          {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
1000_0.1_rbf        {'C': 1000, 'gamma': 0.1, 'kernel': 'rbf'}
100_0.1_rbf          {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
1_0.1_rbf              {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
0.1_0.01_rbf        {'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}
0.1_0.1_rbf          {'C': 0.1, 'gamma': 0.1, 'kernel': 'rbf'}
0.01_0.1_rbf        {'C': 0.01, 'gamma': 0.1, 'kernel': 'rbf'}
```

```
1000_1_rbf            {'C': 1000, 'gamma': 1, 'kernel': 'rbf'}
0.01_0.01_rbf      {'C': 0.01, 'gamma': 0.01, 'kernel': 'rbf'}
0.01_0.001_rbf    {'C': 0.01, 'gamma': 0.001, 'kernel': 'rbf'}
0.01_0.0001_rbf  {'C': 0.01, 'gamma': 0.0001, 'kernel': 'rbf'}
100_1_rbf              {'C': 100, 'gamma': 1, 'kernel': 'rbf'}
0.01_1e-05_rbf    {'C': 0.01, 'gamma': 1e-05, 'kernel': 'rbf'}
1_1_rbf                  {'C': 1, 'gamma': 1, 'kernel': 'rbf'}
0.1_1e-05_rbf      {'C': 0.1, 'gamma': 1e-05, 'kernel': 'rbf'}
0.1_0.0001_rbf    {'C': 0.1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.1_0.001_rbf      {'C': 0.1, 'gamma': 0.001, 'kernel': 'rbf'}
0.1_1_rbf              {'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}
0.01_1_rbf            {'C': 0.01, 'gamma': 1, 'kernel': 'rbf'}
1_1e-05_rbf          {'C': 1, 'gamma': 1e-05, 'kernel': 'rbf'}
1_0.0001_rbf        {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}

                 rank_test_score  mean_test_score  std_test_score
kernel
1000_1e-05_rbf                 1         0.872835        0.009029
100_0.0001_rbf                 2         0.866801        0.003569
1000_0.0001_rbf                3         0.865873        0.006712
100_1e-05_rbf                  4         0.859993        0.005650
100_0.001_rbf                  5         0.858137        0.005290
1000_0.001_rbf                 6         0.854269        0.008291
1_0.01_rbf                     7         0.852568        0.002599
100_0.01_rbf                   8         0.843439        0.005185
1000_0.01_rbf                  9         0.842201        0.008660
1_0.001_rbf                   10         0.841584        0.007262
1000_0.1_rbf                  11         0.829363        0.004019
100_0.1_rbf                   11         0.829363        0.004019
1_0.1_rbf                     13         0.823484        0.003289
0.1_0.01_rbf                  14         0.818379        0.001103
0.1_0.1_rbf                   15         0.816832        0.000253
0.01_0.1_rbf                  15         0.816832        0.000253
1000_1_rbf                    15         0.816832        0.000253
0.01_0.01_rbf                 15         0.816832        0.000253
0.01_0.001_rbf                15         0.816832        0.000253
0.01_0.0001_rbf               15         0.816832        0.000253
100_1_rbf                     15         0.816832        0.000253
0.01_1e-05_rbf                15         0.816832        0.000253
1_1_rbf                       15         0.816832        0.000253
0.1_1e-05_rbf                 15         0.816832        0.000253
0.1_0.0001_rbf                15         0.816832        0.000253
0.1_0.001_rbf                 15         0.816832        0.000253
0.1_1_rbf                     15         0.816832        0.000253
0.01_1_rbf                    15         0.816832        0.000253
1_1e-05_rbf                   29         0.814356        0.000493
1_0.0001_rbf                  30         0.814046        0.002420
```

0.3.4 Stacking

[36]:
```
estimators = [
    ('rf', RandomForestClassifier(n_estimators=100, max_depth=16,␣
↪max_features=8, min_samples_split=8, random_state=42)),
    ('boost', GradientBoostingClassifier(n_estimators=300, learning_rate=0.04,␣
↪max_depth=8, random_state=42)),
    ('svc', svm.SVC(C=1, gamma=0.0001))]

stack = StackingClassifier(estimators=estimators,␣
↪final_estimator=LogisticRegression())

stack.fit(X_train, y_train).score(X_test, y_test)
```

[36]:
```
0.9028465346534653
```

Se observa que entre los 3 indicadores el mejor es Gradient Boosting, pero los 3 indicadores muestran se puede obtener buenos resultados del modelo, lo que nos muestra que los tres modelos proporcionan un resultado robusto, logrando validar los resultados para proximas predicciones o estimaciones.

[37]:
```
cv = cross_val_score(stack, X_train, y_train, cv=10, scoring='accuracy',␣
↪n_jobs=-1)
print('Accuracy: %.6f' % round(np.mean(cv),6))
```

```
Accuracy: 0.899596
```

0.4 Clustering

0.4.1 K-means

Pregunta 4 Utilizando la base de datos *enia.csv* realice un analisis de cluster usando k-means incluyendo todas las variables excepto *tamano* y *ID*. Muestre que sus resultados son sensibles a la seleccion de hiperparametros y compute una metrica de calidad de ajuste del modelo.

[38]:
```
df_enia=pd.read_csv('../data/enia.csv')
df_enia.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39106 entries, 0 to 39105
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   ID          39106 non-null  int64
 1   year        39106 non-null  int64
 2   tamano      39106 non-null  int64
 3   sales       39106 non-null  float64
 4   age         39106 non-null  int64
 5   foreign     39106 non-null  int64
 6   export      39104 non-null  float64
```

```
 7   workers     39106 non-null  float64
 8   fomento     39106 non-null  int64
 9   iyd         39106 non-null  int64
 10  impuestos   39106 non-null  float64
 11  utilidades  39106 non-null  float64
dtypes: float64(5), int64(7)
memory usage: 3.6 MB
```

[39]:
```
df_enia.isnull().sum()
```

[39]:
```
ID            0
year          0
tamano        0
sales         0
age           0
foreign       0
export        2
workers       0
fomento       0
iyd           0
impuestos     0
utilidades    0
dtype: int64
```

[40]:
```
df_enia.dropna(subset=['export'], inplace=True)
```

[41]:
```
df_enia1 = df_enia.drop('tamano', axis=1)
df_enia1 = df_enia.drop('ID', axis=1)
```

[42]:
```
df_enia1.describe()
```

[42]:
```
               year        tamano         sales           age       foreign  \
count  39104.000000  39104.000000  39104.000000  39104.000000  39104.000000
mean    2011.787183      2.248773      3.574172     15.305084      0.081859
std        3.781237      1.153089      1.692742     12.488330      0.274153
min     2007.000000      1.000000      0.000000      0.000000      0.000000
25%     2007.000000      1.000000      2.337643      7.000000      0.000000
50%     2013.000000      2.000000      3.553321     14.000000      0.000000
75%     2015.000000      3.000000      4.539098     20.000000      0.000000
max     2017.000000      4.000000     10.309005    190.000000      1.000000

             export       workers       fomento           iyd     impuestos  \
count  39104.000000  39104.000000  39104.000000  39104.000000  39104.000000
mean       0.111191      1.757726      0.076105      0.224887      0.203856
std        0.314372      1.186507      0.265169      0.417514     15.869466
min        0.000000      0.000000      0.000000      0.000000   -180.992528
25%        0.000000      0.778151      0.000000      0.000000      0.000000
50%        0.000000      1.785330      0.000000      0.000000      0.000007
```

```
75%         0.000000      2.661813      0.000000      0.000000      0.000167
max         1.000000      5.845915      1.000000      1.000000   2981.494528

         utilidades
count  3.910400e+04
mean   1.875255e+00
std    2.306899e+02
min   -2.443698e+02
25%    9.050000e-07
50%    8.080000e-05
75%    1.283704e-03
max    4.544069e+04
```

[43]:

```
target = df_enia1.year
features = df_enia1.drop('year', axis=1)
features.describe()
```

[43]:

```
               year         sales           age       foreign        export  \
count  39104.000000  39104.000000  39104.000000  39104.000000  39104.000000
mean    2011.787183      3.574172     15.305084      0.081859      0.111191
std        3.781237      1.692742     12.488330      0.274153      0.314372
min     2007.000000      0.000000      0.000000      0.000000      0.000000
25%     2007.000000      2.337643      7.000000      0.000000      0.000000
50%     2013.000000      3.553321     14.000000      0.000000      0.000000
75%     2015.000000      4.539098     20.000000      0.000000      0.000000
max     2017.000000     10.309005    190.000000      1.000000      1.000000

            workers       fomento           iyd     impuestos    utilidades
count  39104.000000  39104.000000  39104.000000  39104.000000  3.910400e+04
mean       1.757726      0.076105      0.224887      0.203856  1.875255e+00
std        1.186507      0.265169      0.417514     15.869466  2.306899e+02
min        0.000000      0.000000      0.000000   -180.992528 -2.443698e+02
25%        0.778151      0.000000      0.000000      0.000000  9.050000e-07
50%        1.785330      0.000000      0.000000      0.000007  8.080000e-05
75%        2.661813      0.000000      0.000000      0.000167  1.283704e-03
max        5.845915      1.000000      1.000000   2981.494528  4.544069e+04
```

[44]:

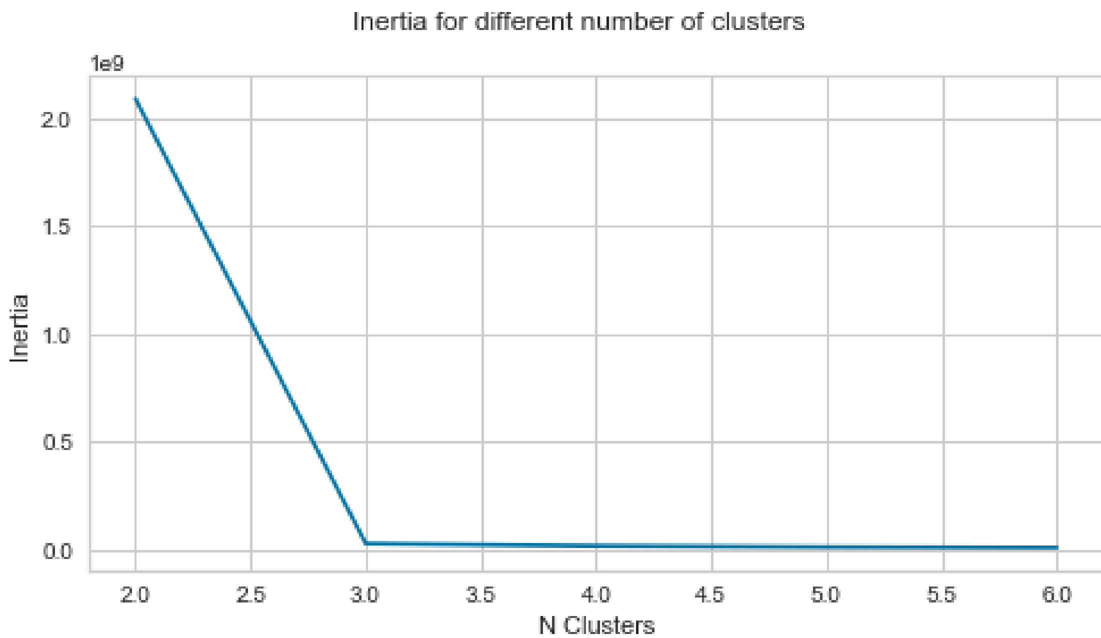
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target,␣
 ↪test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[45]:
```
total_clusters = 5
param_search = pd.DataFrame({
    "N Clusters": np.arange(2,total_clusters+2,dtype=int),
    "Inertia": [0]*total_clusters
})
for n_clusters in range(total_clusters):
    param_search.loc[n_clusters, "Inertia"] = KMeans(n_clusters=n_clusters+1,␣
 ↪random_state=1).fit(X_train).inertia_

fig, ax = plt.subplots(1,1, figsize=(8,4))
sns.lineplot(data=param_search, x="N Clusters", y="Inertia", ax=ax)
fig.suptitle("Inertia for different number of clusters")
fig.show()

n_clusters = 4
cluster_labels = pd.Series(KMeans(n_clusters=n_clusters, random_state=1).
 ↪fit(X_train).labels_)
cluster_labels.value_counts()
```

```
/Users/macbookair/opt/anaconda3/lib/python3.8/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
/Users/macbookair/opt/anaconda3/lib/python3.8/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
/Users/macbookair/opt/anaconda3/lib/python3.8/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
/Users/macbookair/opt/anaconda3/lib/python3.8/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
/Users/macbookair/opt/anaconda3/lib/python3.8/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
<ipython-input-45-d03d3961fb67>:12: UserWarning: Matplotlib is currently using
module://ipykernel.pylab.backend_inline, which is a non-GUI backend, so cannot
show the figure.
  fig.show()
```

```
/Users/macbookair/opt/anaconda3/lib/python3.8/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

[45]:
```
0    31276
3        5
1        1
2        1
dtype: int64
```



[ ]:
```
La grafica nos muestra que se necsitan 3 clusters para poder obtener obtener la␣
↪mayor cantidad de información.
```

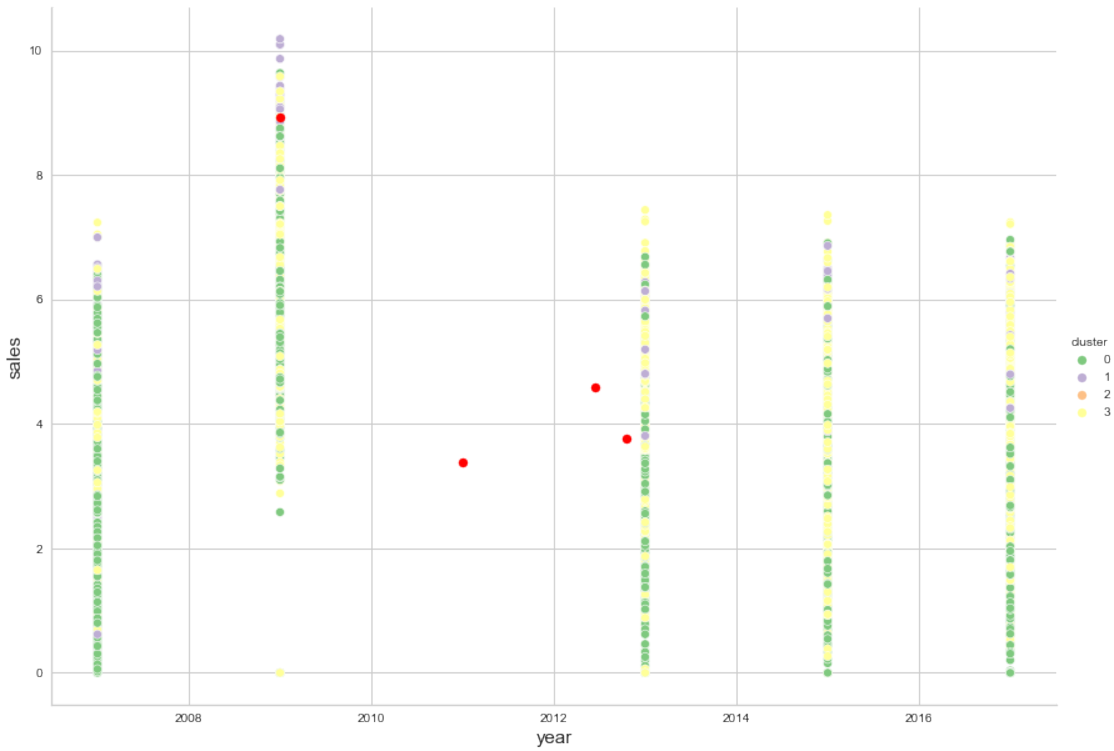
[46]:
```python
kmeans = KMeans(n_clusters=4, random_state=32, max_iter=500,n_init=5)
y_kmeans = kmeans.fit_predict(X_train)

kmeans = KMeans(n_clusters=4,init= "random", random_state = 1).fit(X_train)
centroids = kmeans.cluster_centers_
plt.figure(figsize=(15,8))
Data_kmean = X_train.copy()
Data_kmean['cluster'] = kmeans.labels_
sns.relplot(data = Data_kmean ,x='year' , y  ='sales', hue='cluster',␣
↪palette='Accent' ,kind='scatter', height=8.27, aspect = 11.7/8.27)
plt.scatter(centroids[:, 0], centroids[:,1], c='red', s=50)
```

```
plt.xlabel("year",fontsize=15)
plt.ylabel("sales",fontsize=15)
```

```
/Users/macbookair/opt/anaconda3/lib/python3.8/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

[46]:
```
Text(8.23543002136753, 0.5, 'sales')
```

```
<Figure size 1080x576 with 0 Axes>
```



0.4.2 DBSCAN

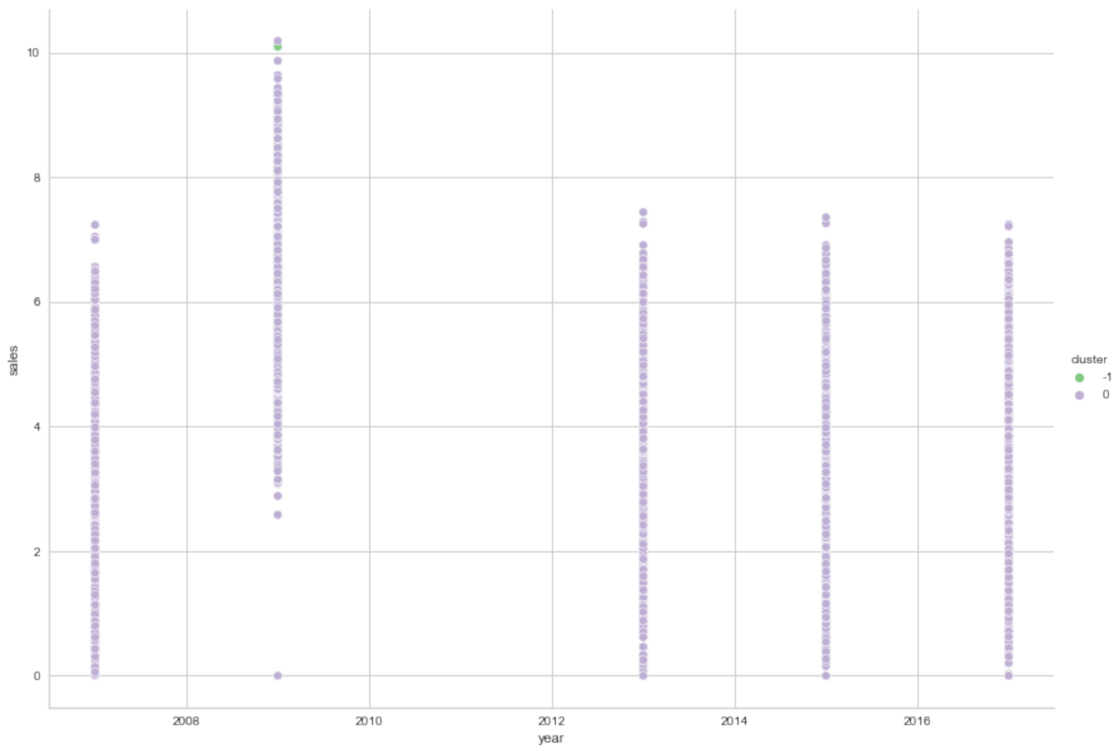
Pregunta 5 Repita el analisis de la Pregunta 4 usando DBSCAN. Muestre que sus resultados son sensibles a la seleccion de hiperparametros y compute una metrica de calidad de ajuste del modelo.

[47]:
```
from sklearn.neighbors import NearestNeighbors
neighbors = NearestNeighbors(n_neighbors=5)
neighbors_fit = neighbors.fit(X_train)
distances, indices = neighbors_fit.kneighbors(X_train)
```

```
[48]: dbscan = DBSCAN(eps=2100, min_samples=100)
# Fit the DBSCAN model to the data
dbscan.fit(X_train)
# Extract the cluster labels
labels = dbscan.labels_
# Print the number of clusters
n_clusters = len(np.unique(labels))
print(f"Number of clusters: {n_clusters}")

data_dbscan = X_train.copy()
data_dbscan['cluster'] = dbscan.labels_
sns.relplot(data = data_dbscan ,x='year' , y  ='sales', hue='cluster',␣
 ↪palette='Accent' ,kind='scatter', height=8.27, aspect = 11.7/8.27)
```

```
Number of clusters: 2
```

```
[48]: <seaborn.axisgrid.FacetGrid at 0x7f94261720d0>
```



Tarea 3

Instrucciones

Los resultados de los ejericicios propuestos se deben entregar como un notebook por correo electronico a *juancaros@udec.cl* el dia 30/6 hasta las 21:00. Es importante considerar que el código debe poder ejecutarse en cualquier computadora con la data original del repositorio. Recordar la convencion para el nombre de archivo ademas de incluir en su documento titulos y encabezados por seccion. Utilizar la base de datos segun indicado en las preguntas.