ml-Copy1

July 15, 2023

Section 6: Machine Learning

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.

[2]:
```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

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
```

Preguntas:

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.

R: El modelo produce un RMSE promedio de 1.372 con el alpha optimo (0.002316). La prediccion no es buena (carga casi todo el peso a la constante), y algunas variables contribuyen a la clasificacion, como genero y preguntas de comportamiento (sk en la data).

[5]:
```
df=pd.read_csv('../data/junaeb2.csv')
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)
target = df.imce
features = df.drop('imce', axis=1)
features.describe()
```

[5]:
```
               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]
```

[20]:

```
# 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)

# Train the Lasso regression model
lasso=Lasso(alpha=0.002316,
            max_iter=10000,
```

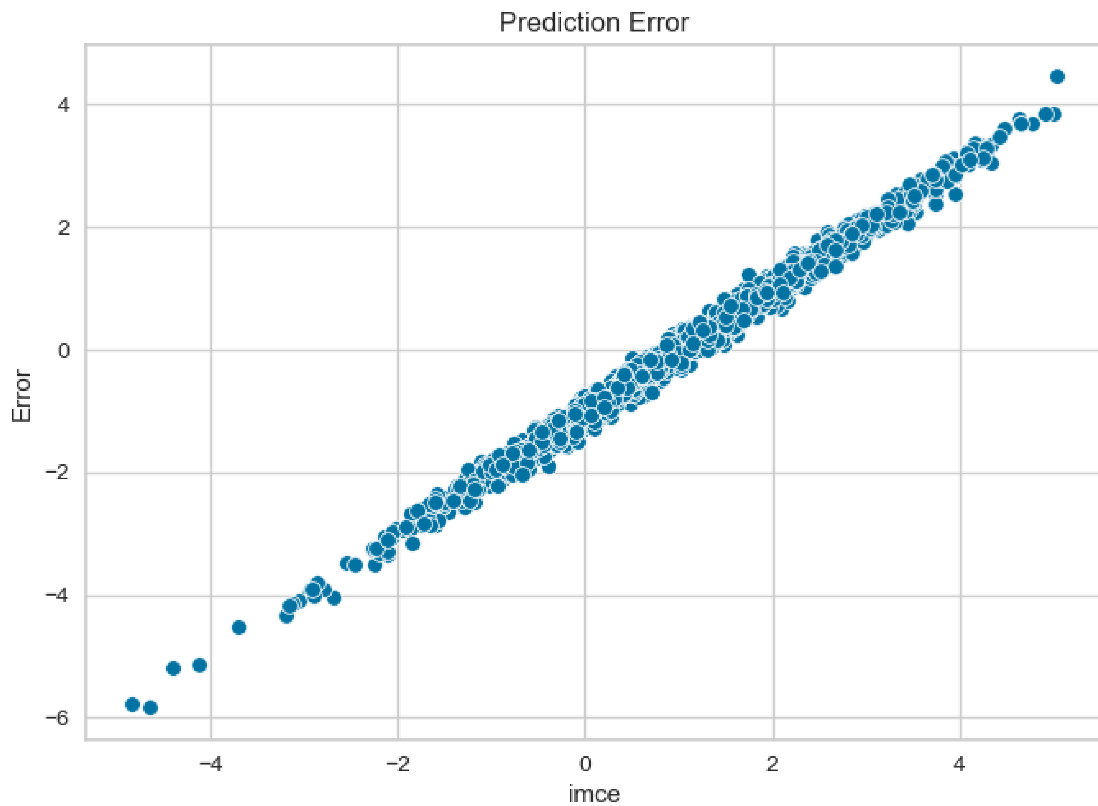
```
            tol=0.0001)  # You can adjust the alpha value to control the␣
 ↪regularization strength
lasso.fit(X_train_scaled, y_train)
y_pred = lasso.predict(X_test_scaled)

# 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)))
```

```
c:\Users\juanc\anaconda3\lib\site-
packages\joblib\externals\loky\process_executor.py:702: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.
  warnings.warn(
```

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

[26]:
```
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')
test['Error'] = test['imce']-test['Predicted']
sns.scatterplot(data=test, x='imce', y='Error').set_title('Prediction Error')
```

[26]:
```
Text(0.5, 1.0, 'Prediction Error')
```



[15]:
```
eli5.show_weights(lasso, top=-1, feature_names = X_train.columns.tolist())
```

[15]: <IPython.core.display.HTML object>

[16]:
```
# 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
```

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.

R: El mejor modelo produce una precision de .90, con el siguiente set de hiperparametros: max_depth = 8, max_features = 8, min_samples_leaf = 4, min_samples_split = 10, n_estimators = 300. Los principales predictores son las horas trabajadas (hrsusu) y la edad al momento de la encuesta.

[29]:
```
df=pd.read_csv('../data/charls2.csv')
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)
target = df.retired
features = df.drop(['retired','retage','retin'], axis=1)
features.describe()
```

[29]:
```
               age         cesd        child       female       hrsusu  \
count  8080.000000  8080.000000  8080.000000  8080.000000  8080.000000
mean     58.164356     9.112376     2.780074     0.535272     2.566736
std       9.374956     6.481237     1.397316     0.498785     1.788298
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     1.000000     3.496508
75%      64.000000    13.000000     4.000000     1.000000     4.025352
max      95.000000    30.000000    10.000000     1.000000     5.123964

             hsize     intmonth      married       schadj        urban  \
count  8080.000000  8080.000000  8080.000000  8080.000000  8080.000000
mean      3.764233     7.506931     0.879084     4.039851     0.212005
std       1.823838     1.001893     0.326050     3.545666     0.408754
min       1.000000     1.000000     0.000000     0.000000     0.000000
25%       2.000000     7.000000     1.000000     0.000000     0.000000
50%       4.000000     7.000000     1.000000     4.000000     0.000000
75%       5.000000     8.000000     1.000000     8.000000     0.000000
max      16.000000    12.000000     1.000000    16.000000     1.000000

               wealth
count     8080.000000
mean      1479.488490
std      43479.865654
min   -1000000.000000
25%          0.000000
50%        400.000000
75%       2200.000000
max     900100.000000
```

[35]:
```
# 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)
```

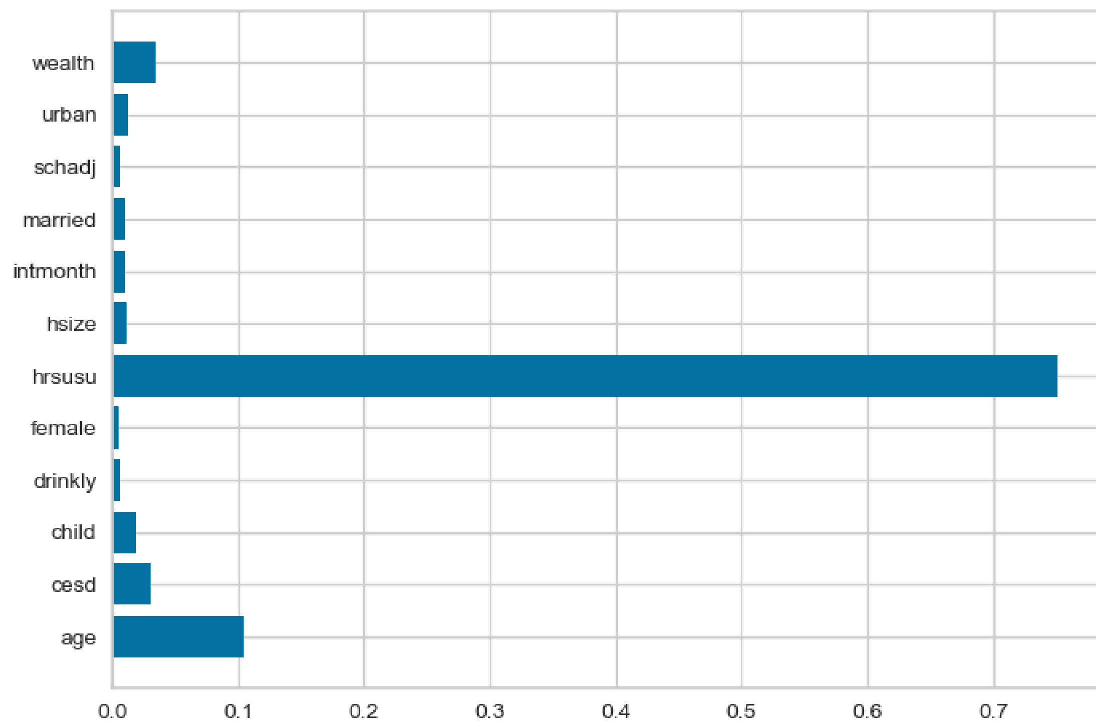
```
rf = RandomForestClassifier(max_depth = 8, max_features = 8, min_samples_leaf =␣
 ↪4, min_samples_split = 10, n_estimators = 300, random_state=42)

# 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)

confusion_matrix(y_test, y_pred)
```

```
Accuracy: 0.9034653465346535
```

[35]:
```
array([[1208,  105],
       [  51,  252]], dtype=int64)
```

[36]:
```
plt.barh(features.columns, rf.feature_importances_)
```

[36]:
```
<BarContainer object of 12 artists>
```



[33]:
```python
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, 16],
    'max_features': [6, 8],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300]
}

# 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)

grid_search.fit(X_train_scaled, y_train)
```

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

[33]:
```
GridSearchCV(cv=10, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'bootstrap': [True], 'max_depth': [4, 8, 16],
                         'max_features': [6, 8], 'min_samples_leaf': [3, 4, 5],
                         'min_samples_split': [8, 10, 12],
                         'n_estimators': [100, 200, 300]},
             verbose=2)
```

[34]:
```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"]]
```

[34]:
```
                                                              params  \
kernel
True_8_8_4_10_300   {'bootstrap': True, 'max_depth': 8, 'max_featu…
True_8_6_5_12_200   {'bootstrap': True, 'max_depth': 8, 'max_featu…
True_8_6_4_12_100   {'bootstrap': True, 'max_depth': 8, 'max_featu…
True_16_8_3_12_300  {'bootstrap': True, 'max_depth': 16, 'max_feat…
True_8_6_5_12_300   {'bootstrap': True, 'max_depth': 8, 'max_featu…
…                                                                 …
True_4_8_3_12_200   {'bootstrap': True, 'max_depth': 4, 'max_featu…
True_4_8_4_12_300   {'bootstrap': True, 'max_depth': 4, 'max_featu…
```

```
True_16_6_4_8_200   {'bootstrap': True, 'max_depth': 16, 'max_feat…
True_16_8_3_8_100   {'bootstrap': True, 'max_depth': 16, 'max_feat…
True_4_8_3_12_300   {'bootstrap': True, 'max_depth': 4, 'max_featu…

                    rank_test_score  mean_test_score  std_test_score
kernel
True_8_8_4_10_300                 1         0.900524        0.012516
True_8_6_5_12_200                 2         0.900524        0.011082
True_8_6_4_12_100                 3         0.900369        0.011574
True_16_8_3_12_300                4         0.900369        0.012681
True_8_6_5_12_300                 5         0.900215        0.012155
…                               …              …             …
True_4_8_3_12_200               158         0.895573        0.010144
True_4_8_4_12_300               159         0.895573        0.009643
True_16_6_4_8_200               160         0.895418        0.012398
True_16_8_3_8_100               161         0.895264        0.015384
True_4_8_3_12_300               162         0.894954        0.010061

[162 rows x 4 columns]
```

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.

R: El modelo via Stacking produce una precision menor al modelo RF con los hiperparametros optimos, pero la diferencia es marginal. En base a los resultados se sugiere usar RF para acelerar el proceso.

[38]:
```python
from sklearn import svm

estimators = [
    ('rf', RandomForestClassifier(n_estimators=300, max_depth=8,␣
 ↪max_features=8, min_samples_split=10, 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)
```

[38]:
```
0.8985148514851485
```

[39]:
```python
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: 0.899286
```

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.

R: Al explorar sobre el espacio de features estandarizados, se observa que K-means identifica un numero alto de clusters sin perdida de informacion, aunque ello conlleva a generar clusters con observaciones, potencialmente identificando outliers de la data. El numero optimo de clusters ignorando aquellos que son observaciones unicas es siete.

[3]:
```
df=pd.read_csv('../data/enia.csv')
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)
df.export = df.export.astype(int)
df['utilidades']=np.log(df['utilidades']-df['utilidades'].min()+0.1)
features = df.drop(['tamano','ID','year'], axis=1)
features.describe()
```

[3]:
```
              sales           age       foreign        export       workers  \
count  39104.000000  39104.000000  39104.000000  39104.000000  39104.000000
mean       3.574172     15.305084      0.081859      0.111191      1.757726
std        1.692742     12.488330      0.274153      0.314372      1.186507
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        2.337643      7.000000      0.000000      0.000000      0.778151
50%        3.553321     14.000000      0.000000      0.000000      1.785330
75%        4.539098     20.000000      0.000000      0.000000      2.661813
max       10.309005    190.000000      1.000000      1.000000      5.845915

            fomento           iyd     impuestos    utilidades
count  39104.000000  39104.000000  39104.000000  39104.000000
mean       0.076105      0.224887      0.203856      5.500845
std        0.265169      0.417514     15.869466      0.060706
min        0.000000      0.000000   -180.992528     -2.302585
25%        0.000000      0.000000      0.000000      5.499092
50%        0.000000      0.000000      0.000007      5.499092
75%        0.000000      0.000000      0.000167      5.499097
max        1.000000      1.000000   2981.494528     10.729529
```

[4]:
```
# Split the dataset into training and testing sets
X_train, X_test = train_test_split(features, 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)
```

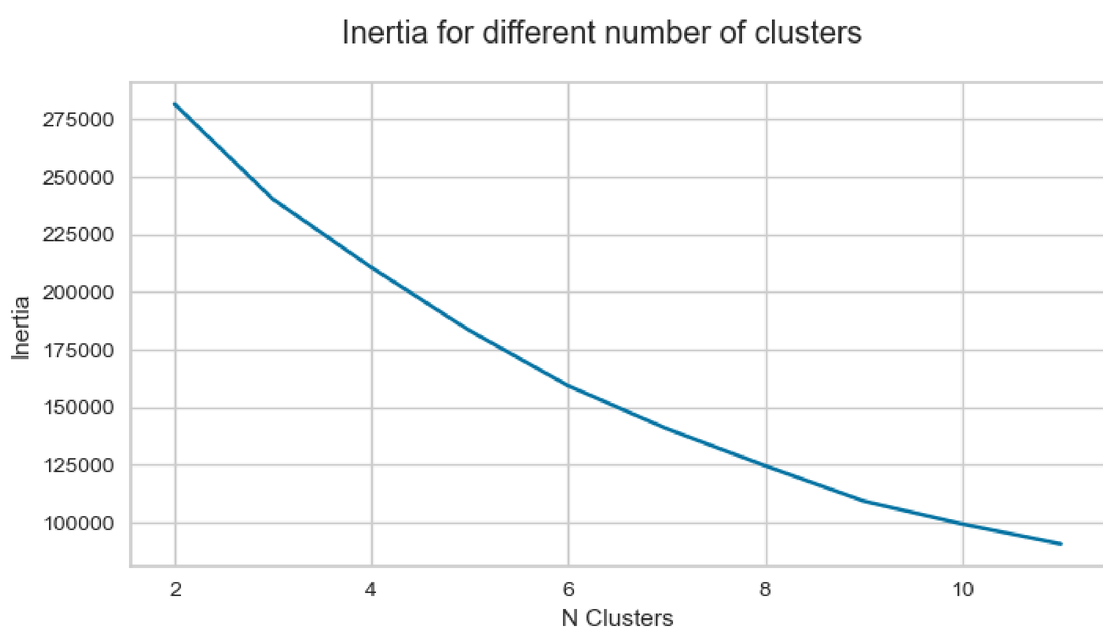
[5]:
```
total_clusters = 10
param_search = pd.DataFrame({
    "N Clusters": np.arange(2,total_clusters+2,dtype=int),
    "Inertia": [0]*total_clusters
```

```
})
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_scaled).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 = 9
cluster_labels = pd.Series(KMeans(n_clusters=n_clusters, random_state=1).
 ↪fit(X_train_scaled).labels_)
cluster_labels.value_counts()
```

```
c:\Users\juanc\anaconda3\lib\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(
c:\Users\juanc\anaconda3\lib\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(
c:\Users\juanc\anaconda3\lib\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(
c:\Users\juanc\anaconda3\lib\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(
c:\Users\juanc\anaconda3\lib\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(
c:\Users\juanc\anaconda3\lib\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(
c:\Users\juanc\anaconda3\lib\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(
c:\Users\juanc\anaconda3\lib\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(
```

```
c:\Users\juanc\anaconda3\lib\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(
c:\Users\juanc\anaconda3\lib\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(
C:\Users\juanc\AppData\Local\Temp\ipykernel_12916\607399981.py:12: UserWarning:
Matplotlib is currently using module://matplotlib_inline.backend_inline, which
is a non-GUI backend, so cannot show the figure.
  fig.show()
c:\Users\juanc\anaconda3\lib\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(
```

[5]:
```
0    8263
4    7940
7    4432
6    3937
2    2379
1    2257
5    2073
3       1
8       1
dtype: int64
```



[8]:
```
kmeans = KMeans(n_clusters=7,init= "random", random_state = 1).
    ↪fit(X_train_scaled)
y_kmeans = kmeans.fit_predict(X_train_scaled)
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='workers' , 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("workers",fontsize=15)
plt.ylabel("sales",fontsize=15)
```

```
c:\Users\juanc\anaconda3\lib\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(
c:\Users\juanc\anaconda3\lib\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(
```

[8]:
```
Text(35.23204204917169, 0.5, 'sales')
```

```
<Figure size 1500x800 with 0 Axes>
```

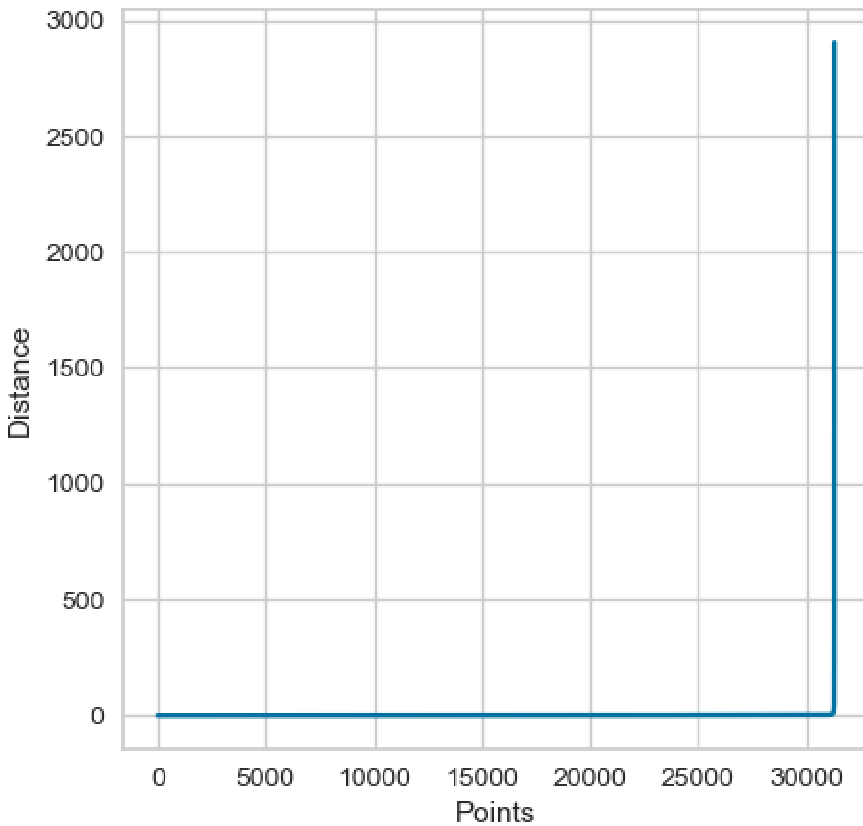
[9]:
```python
Data_kmean['cluster'].value_counts()
```

[9]:
```
5    8244
1    7970
0    4430
3    3937
2    3040
6    2114
4    1548
Name: cluster, dtype: int64
```

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.

R: Al repetir el mismo analisis anterior con DBSCAN, encontramos tres clusters en vez de siete (parametros optimizados en funcion de la distancia de los vecinos mas proximos).

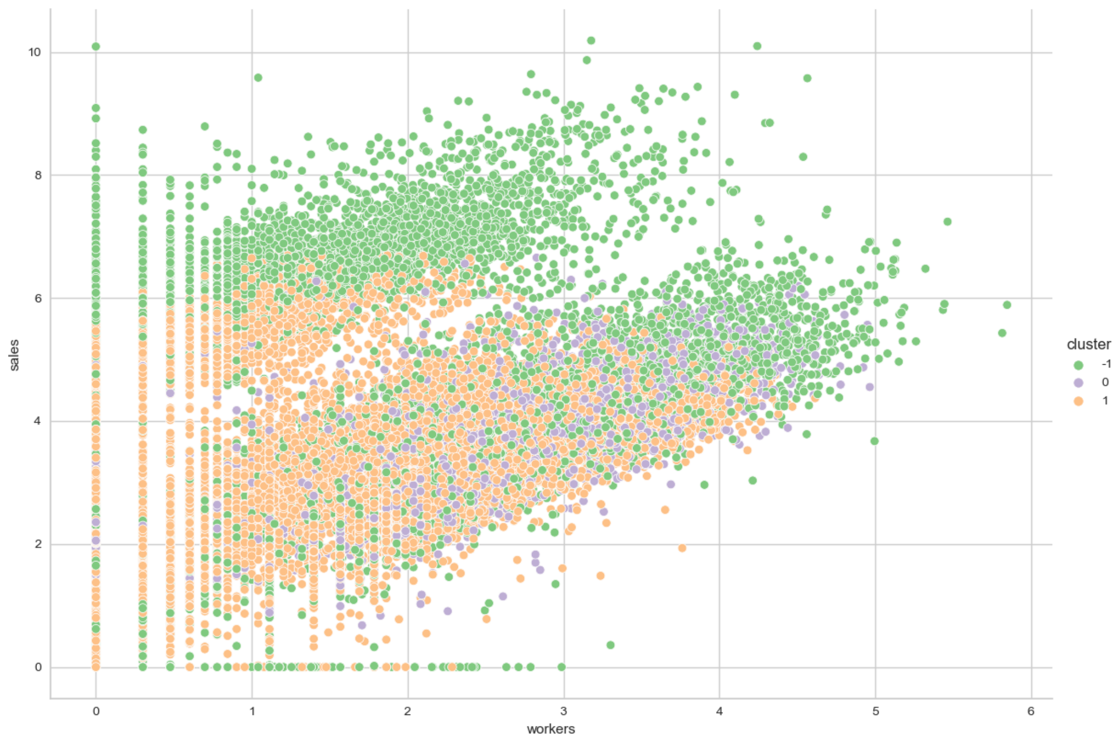
[23]:
```python
from sklearn.neighbors import NearestNeighbors
nearest_neighbors = NearestNeighbors(n_neighbors=11)
neighbors = nearest_neighbors.fit(X_train)
distances, indices = neighbors.kneighbors(X_train)
distances = np.sort(distances[:,10], axis=0)
fig = plt.figure(figsize=(5, 5))
plt.plot(distances)
plt.xlabel("Points")
plt.ylabel("Distance")
plt.savefig("Distance_curve.png", dpi=300)
```



[26]:
```
dbscan = DBSCAN(eps=2, min_samples=1500)
# 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='workers' , y  ='sales', hue='cluster',␣
 ↪palette='Accent' ,kind='scatter', height=8.27, aspect = 11.7/8.27)
```

```
Number of clusters: 3
```

[26]:
```
<seaborn.axisgrid.FacetGrid at 0x2becfa363d0>
```




```
[27]: data_dbscan['cluster'].value_counts()
```

```
[27]:  1     17777
      -1      9276
       0      4230
      Name: cluster, dtype: int64
```