Project Student-performance

Edgar Ariza Garcia
Juan Pablo Galvis
Jhonatan David Serrano

Objetivos Principales

✓ Identificar patrones en el rendimiento estudiantil a través de visualizaciones de datos.

✓ Analizar la correlación entre diferentes factores (interacción en clase, recursos visitados, asistencia, etc.).

✓ Determinar qué características tienen mayor impacto en la clasificación de los estudiantes en Low, Middle o High-Level.

✓ Visualizar distribuciones y relaciones entre variables usando histogramas, mapas de calor y gráficos de dispersión.

Columnas Relevantes

Class: Rendimiento del estudiante en 3 categorias.

raisedhands: Número de veces que el estudiante levantó la mano en clase.

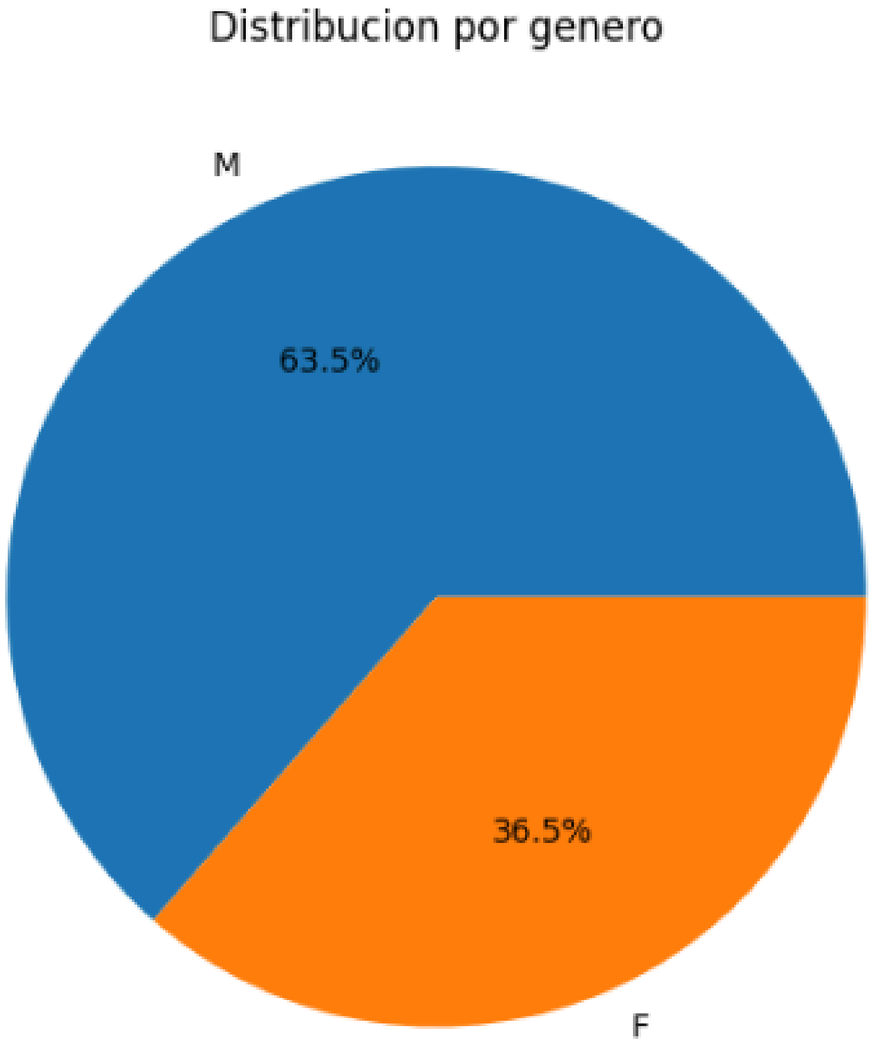
VisITedResources: Número de recursos educativos visitados.

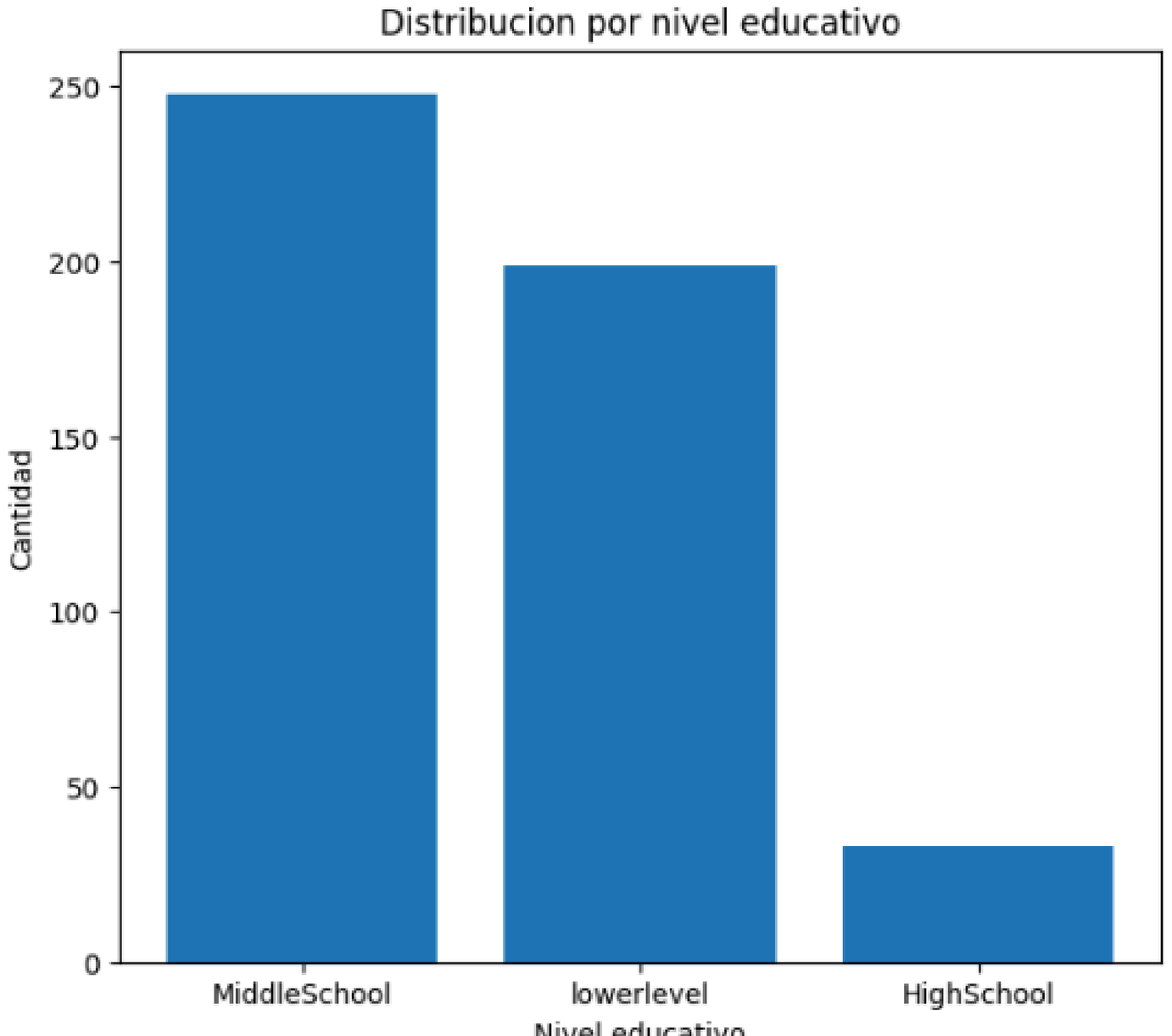
StudentAbsenceDays: Días de ausencia del estudiante (Menos de 7 vs. Más de 7).

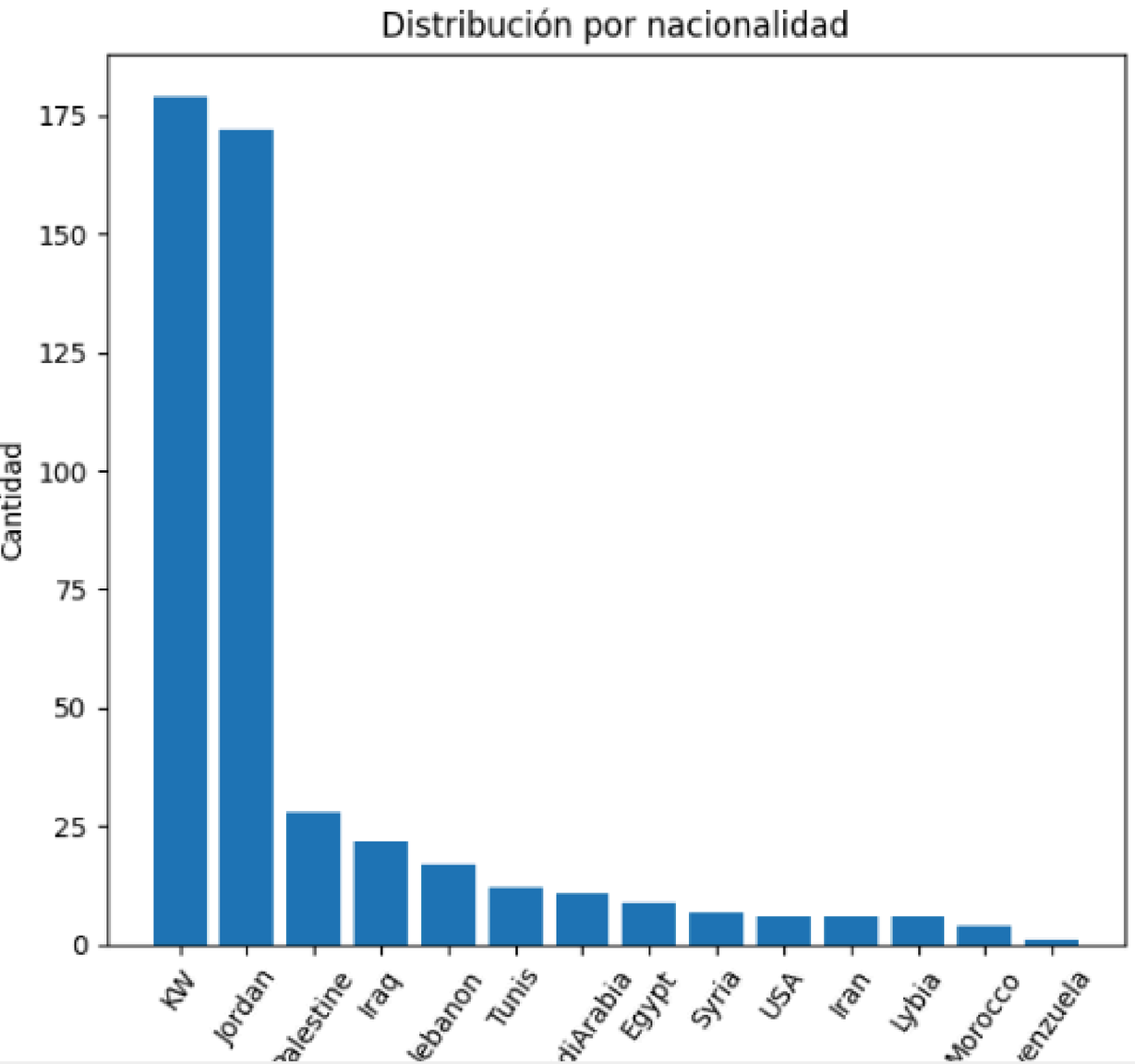
ParentschoolSatisfaction: Nivel de satisfacción de los padres con la escuela.

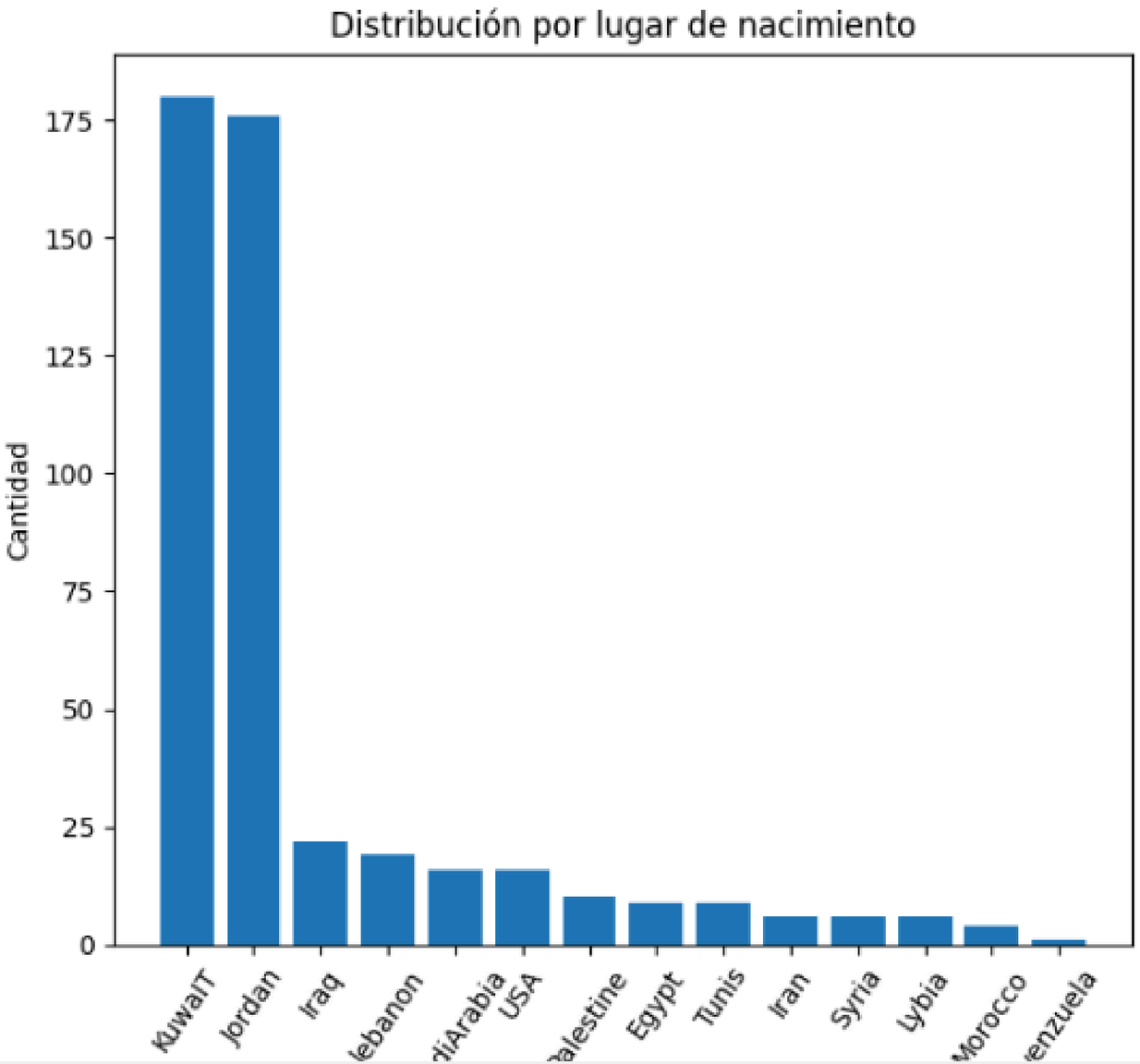
StageID: Nivel educativo (Lower Level, Middle School, High School).

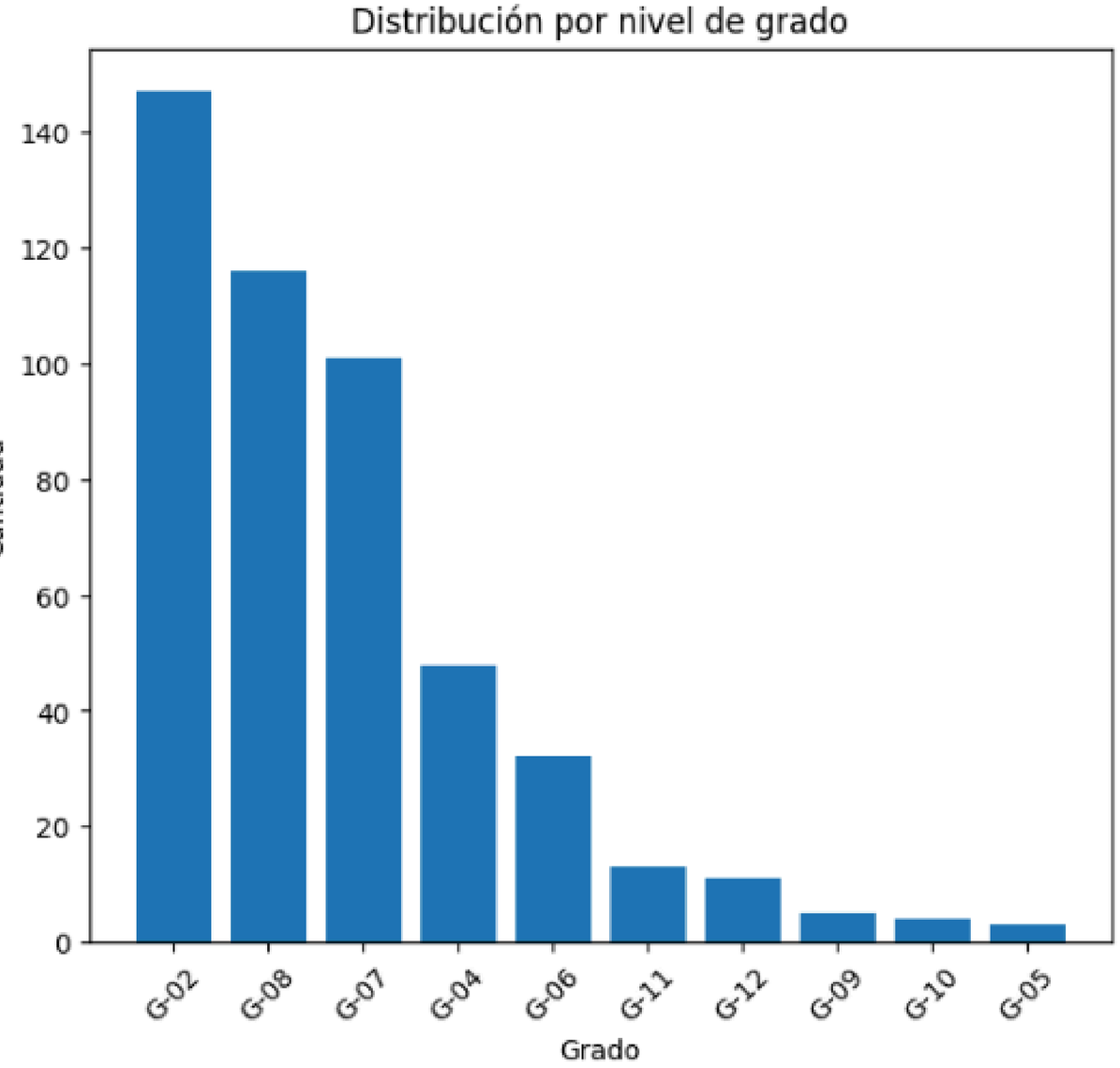
Graficas por columna

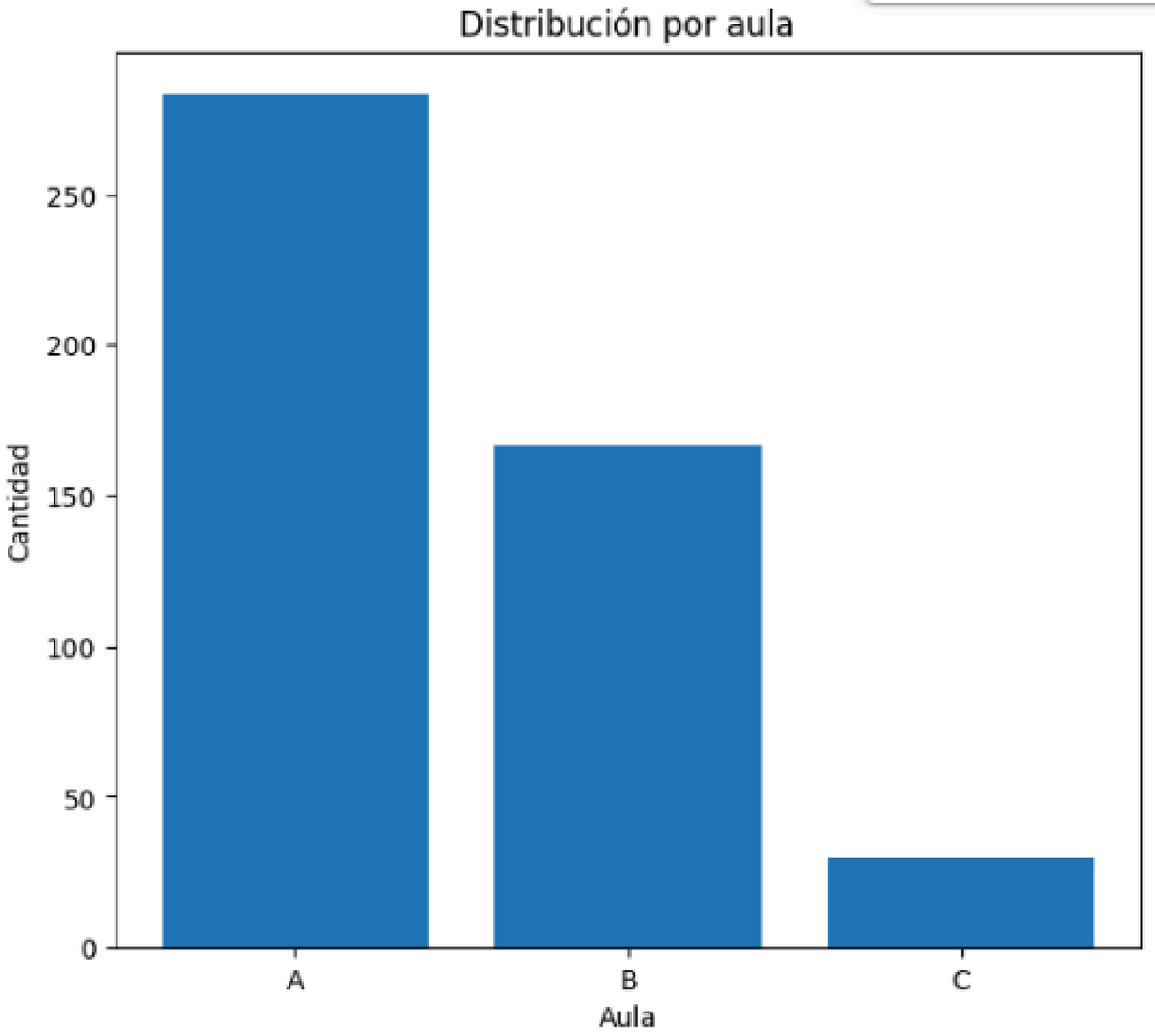


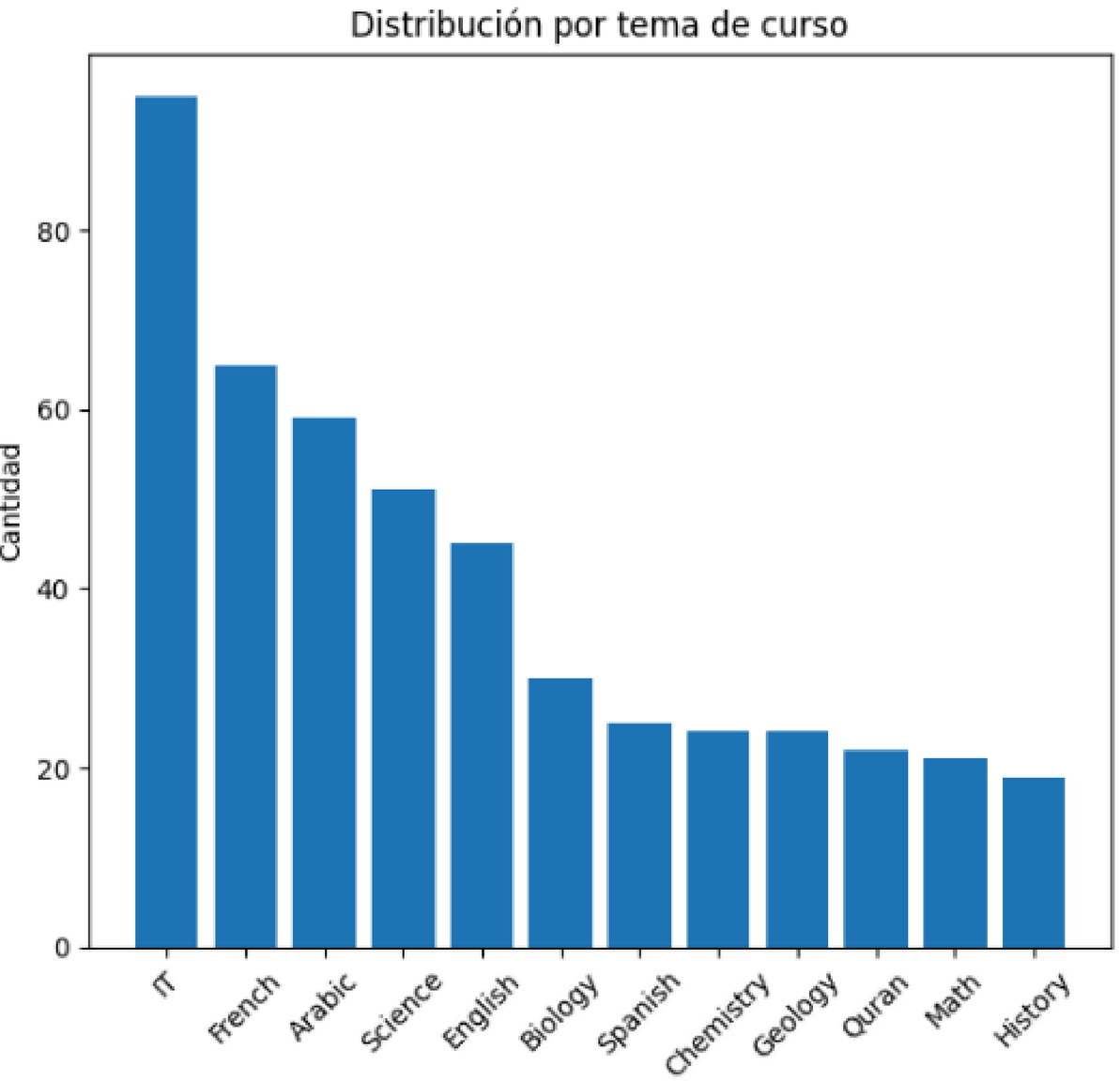


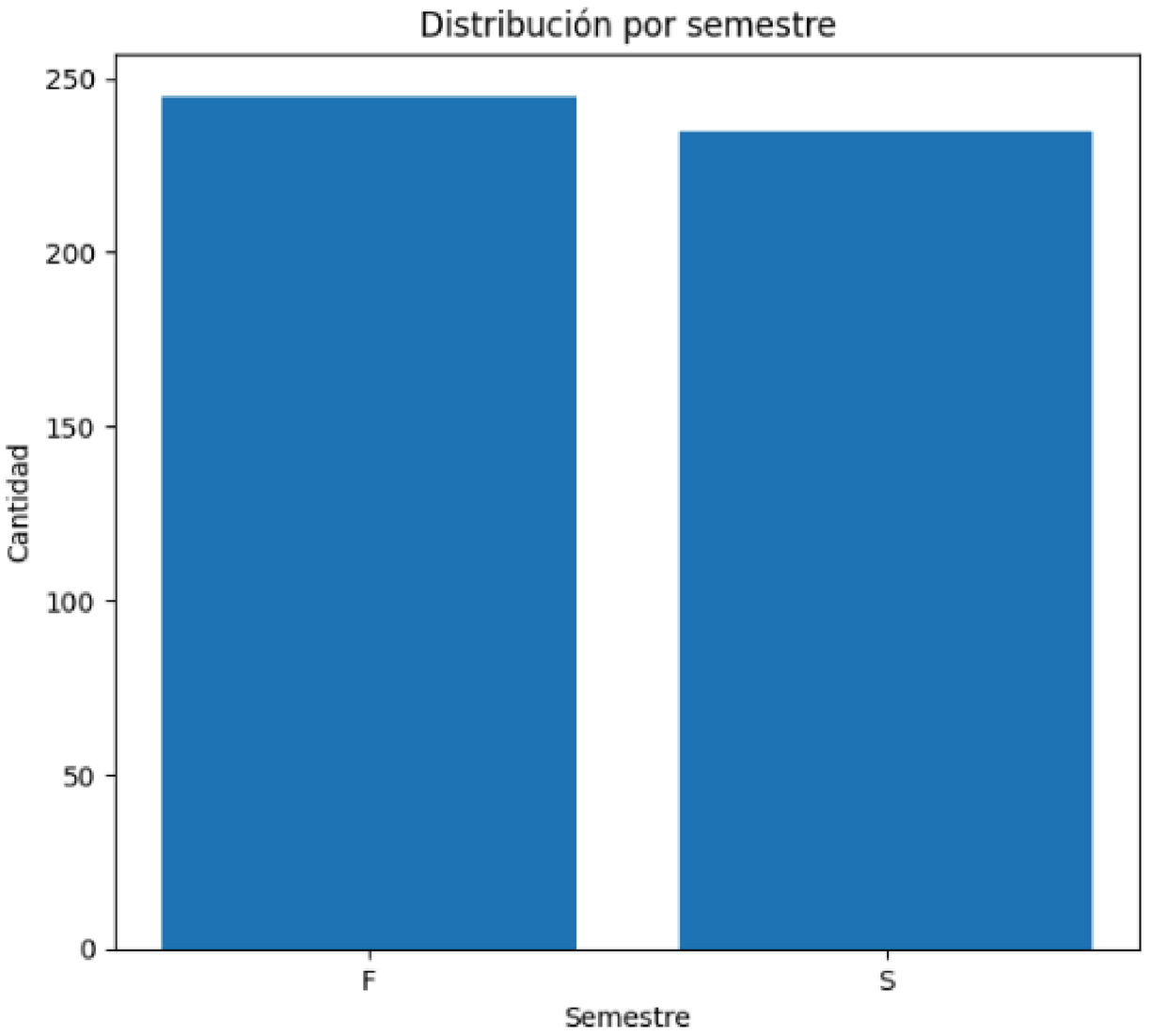

Distribución por nacionalidad
Cantidad
0
25
50
75
100
125
150
175
KW
Jordan
Palestine
Iraq
Lebanon
Tunis
SaudiArabia
Egypt
Syria
USA
Iran
Lybia
Morocco
Venezuela

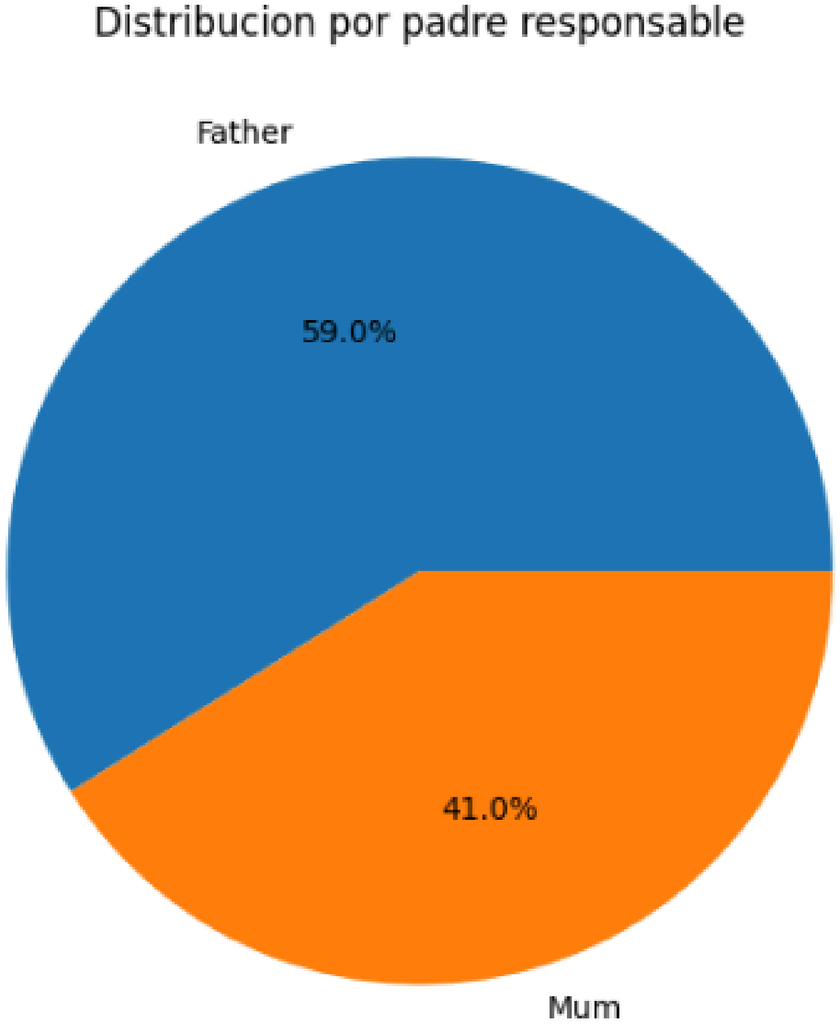

Distribución por lugar de nacimiento
Cantidad
0
25
50
75
100
125
150
175
KuwaIT
Jordan
Iraq
Lebanon
SaudiArabia
USA
Palestine
Egypt
Tunis
Iran
Syria
Lybia
Morocco
Venezuela

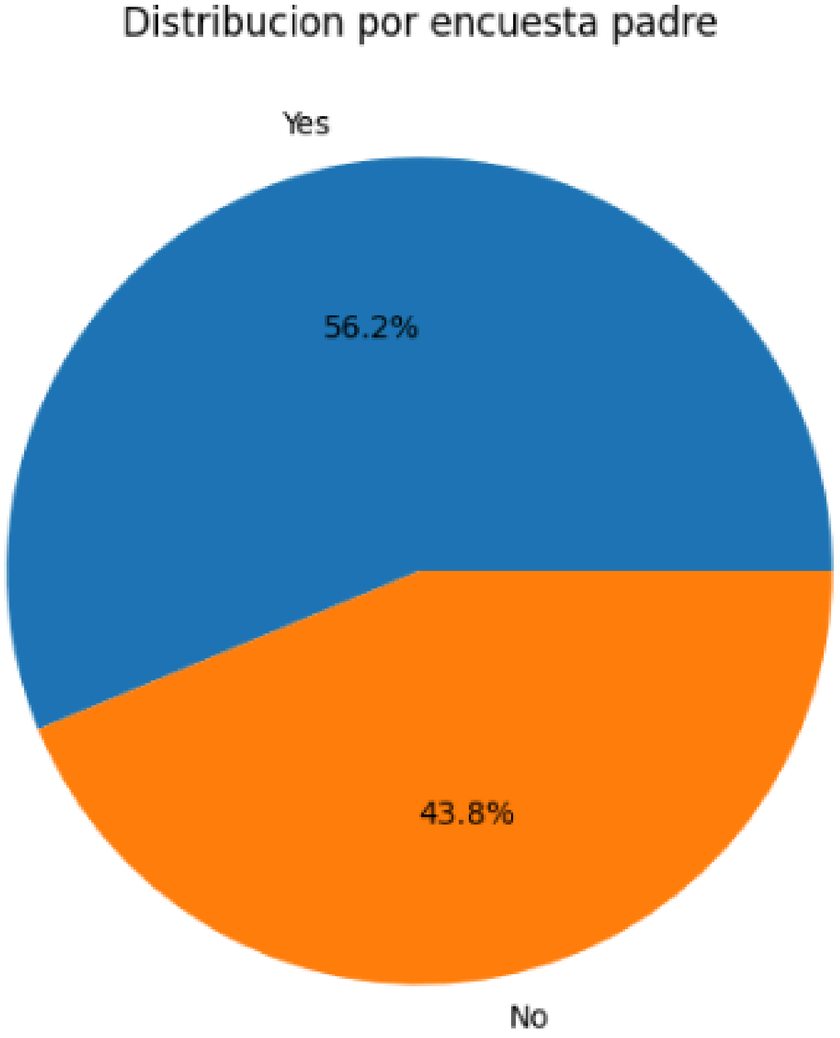

Distribución por nivel de grado
140
120
100
80
60
40
20
0
G-02
G-08
G-07
G-04
G-06
G-11
G-12
G-09
G-10
G-05
Grado

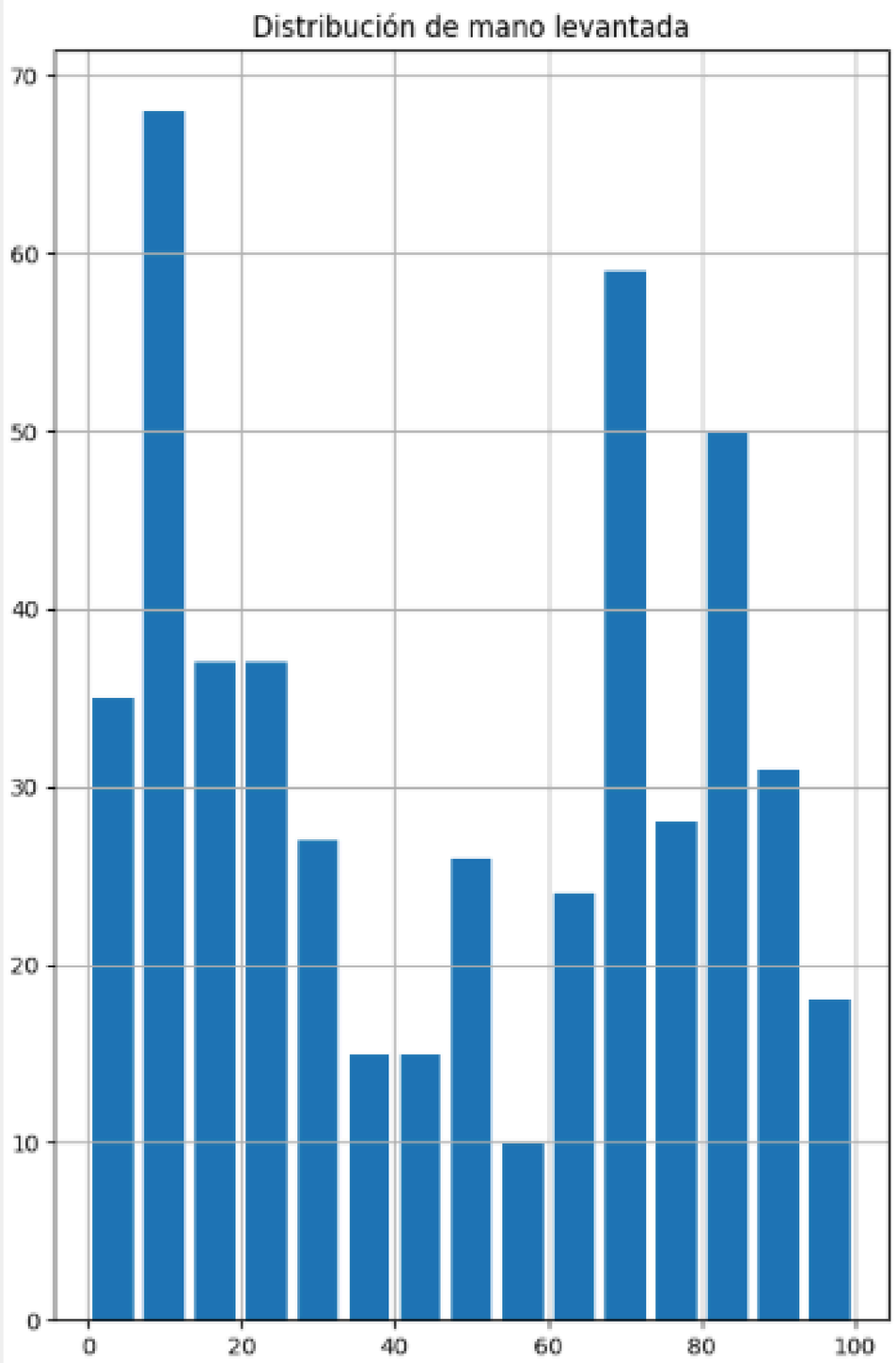

Distribución por aula
250
200
150
100
50
0
Cantidad
A
B
C
Aula

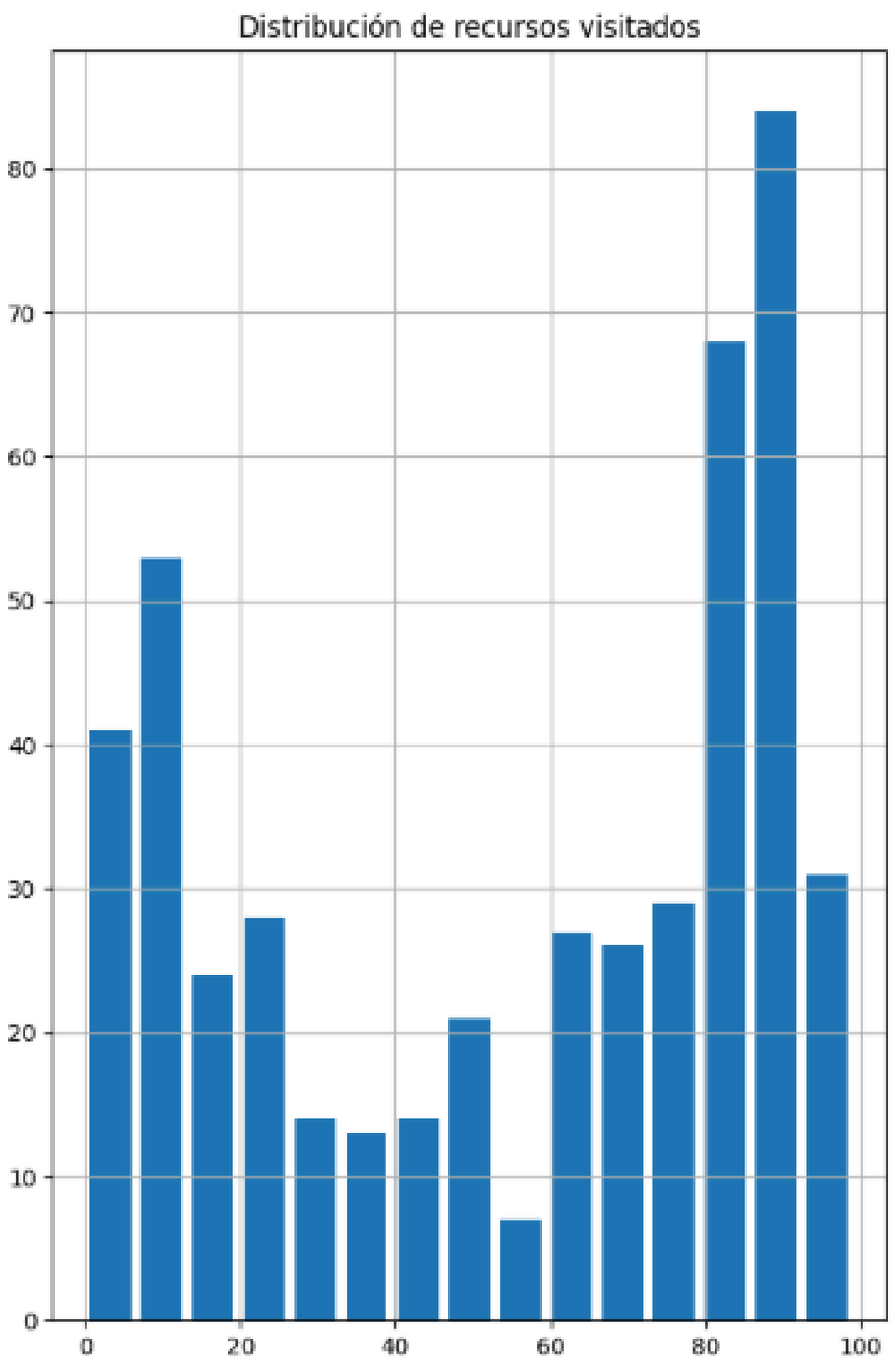

Distribución por tema de curso
Cantidad
0
20
40
60
80
IT
French
Arabic
Science
English
Biology
Spanish
Chemistry
Geology
Quran
Math
History

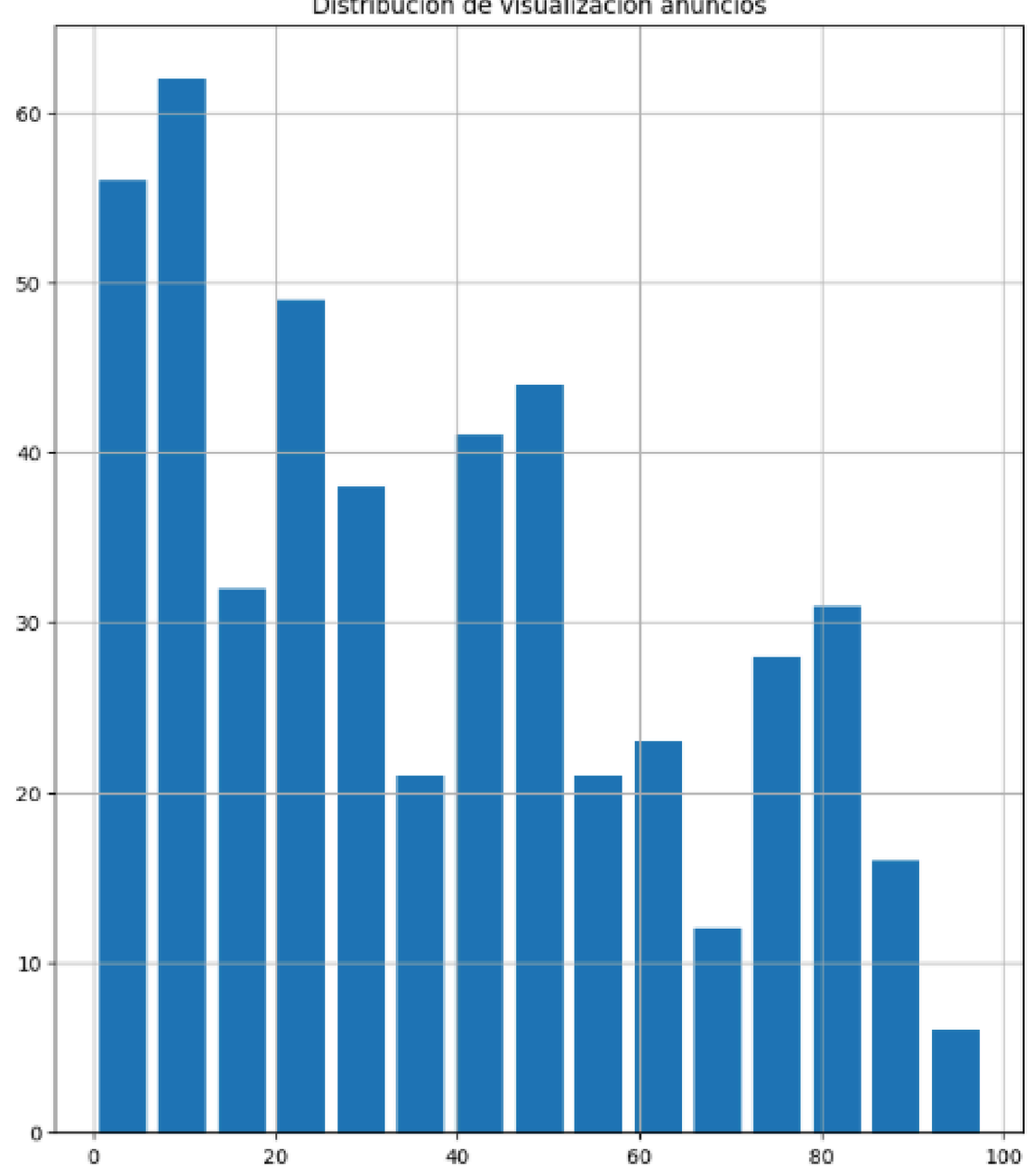

Distribución por semestre
Cantidad
0
50
100
150
200
250
F
S
Semestre

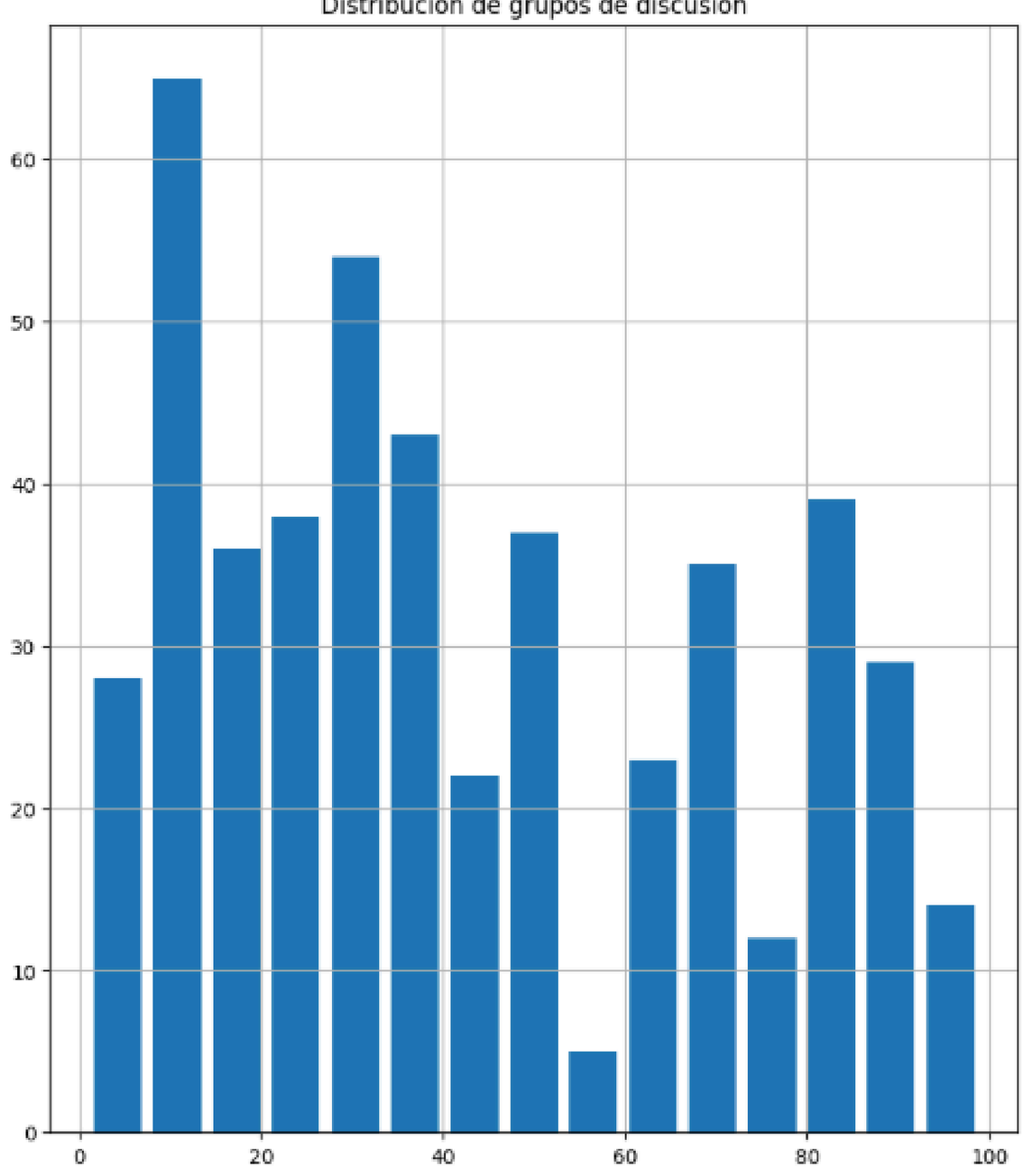

Distribucion por padre responsable
Father
59.0%
41.0%
Mum

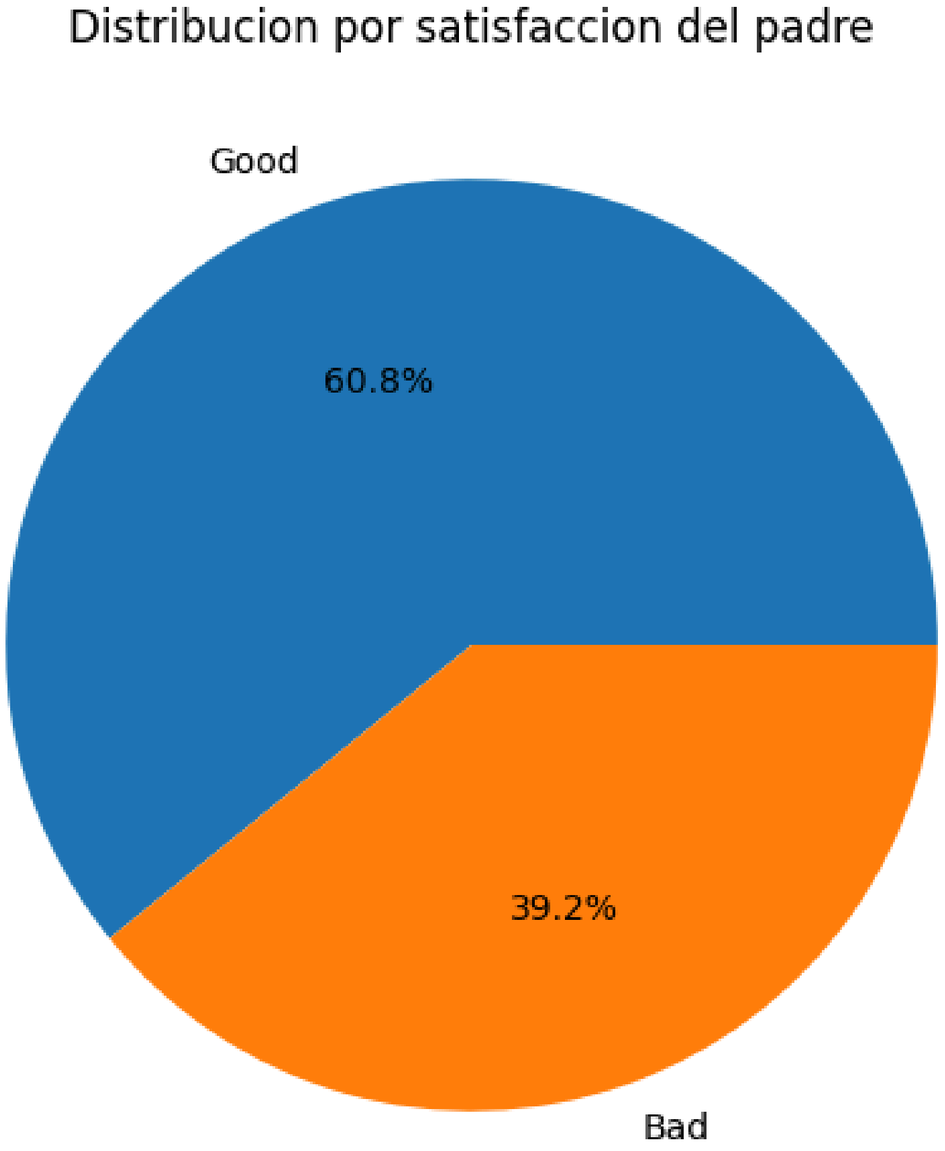

Distribucion por encuesta padre
Yes
56.2%
43.8%
No

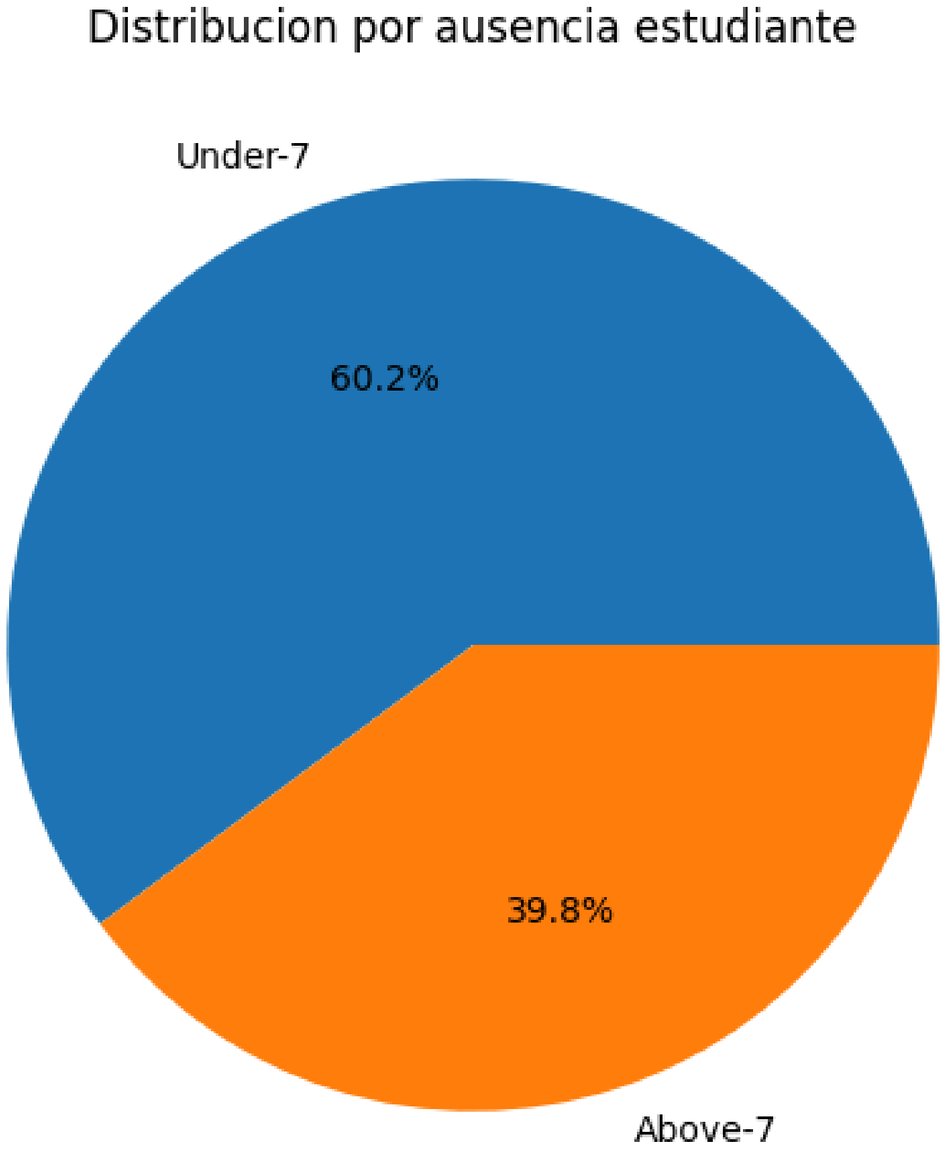

Distribución de mano levantada
0
10
20
30
40
50
60
70
0
20
40
60
80
100


Distribución de recursos visitados
0
10
20
30
40
50
60
70
80
0
20
40
60
80
100

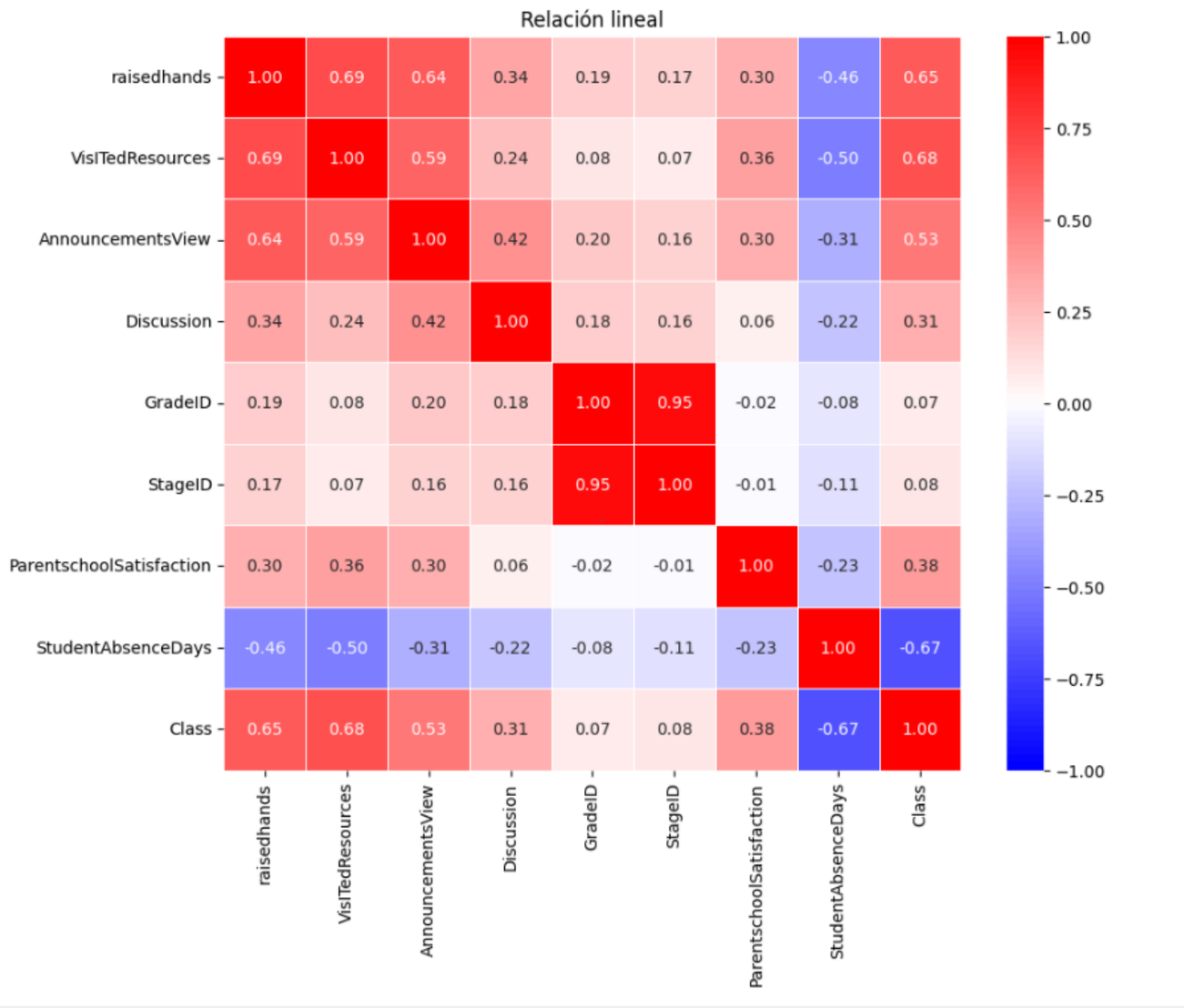

Distribución de visualización anuncios
0
10
20
30
40
50
60
0
20
40
60
80
100


Distribución de grupos de discusión
0
10
20
30
40
50
60
0
20
40
60
80
100

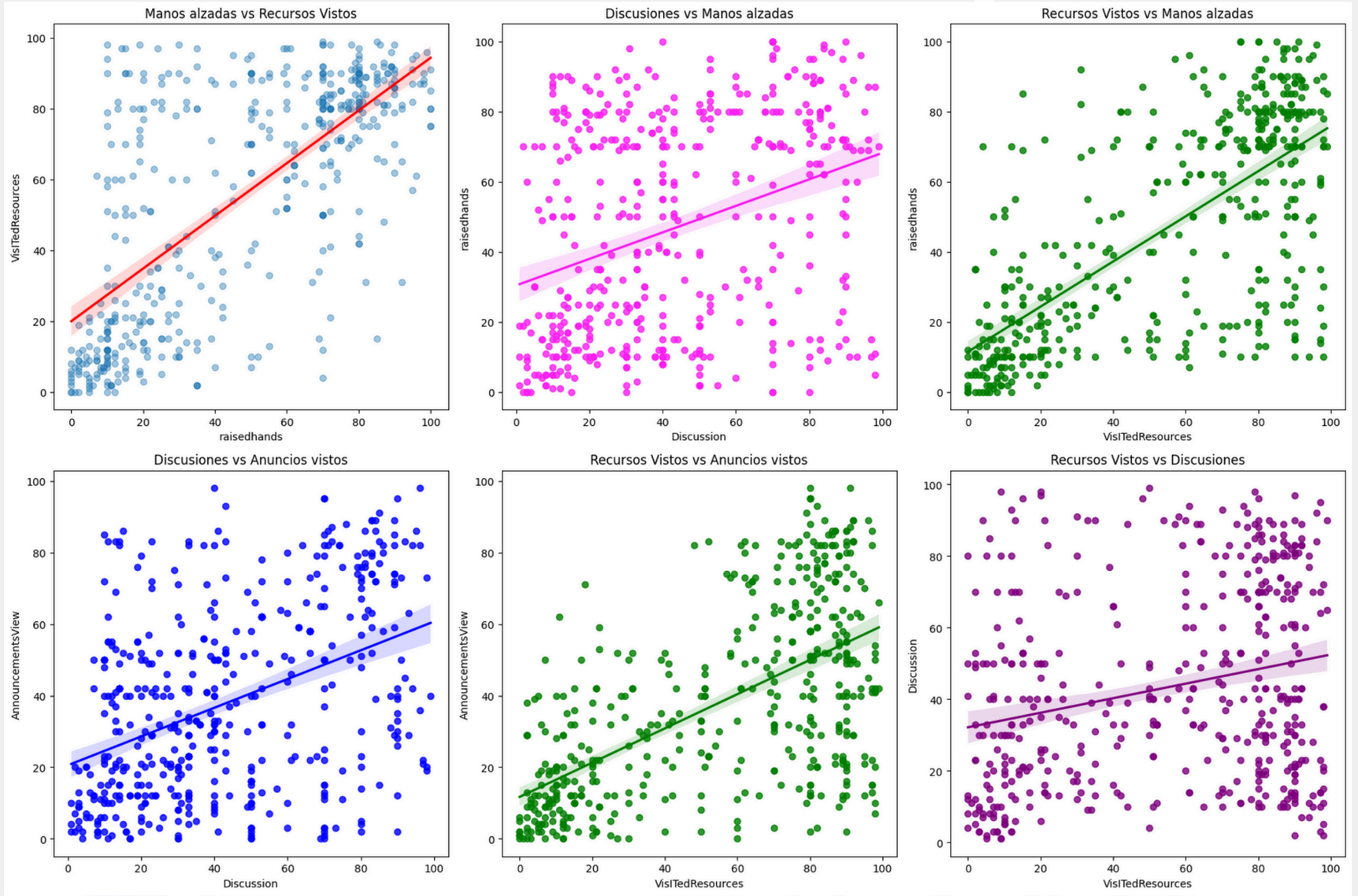

Distribucion por satisfaccion del padre
Good
60.8%
39.2%
Bad

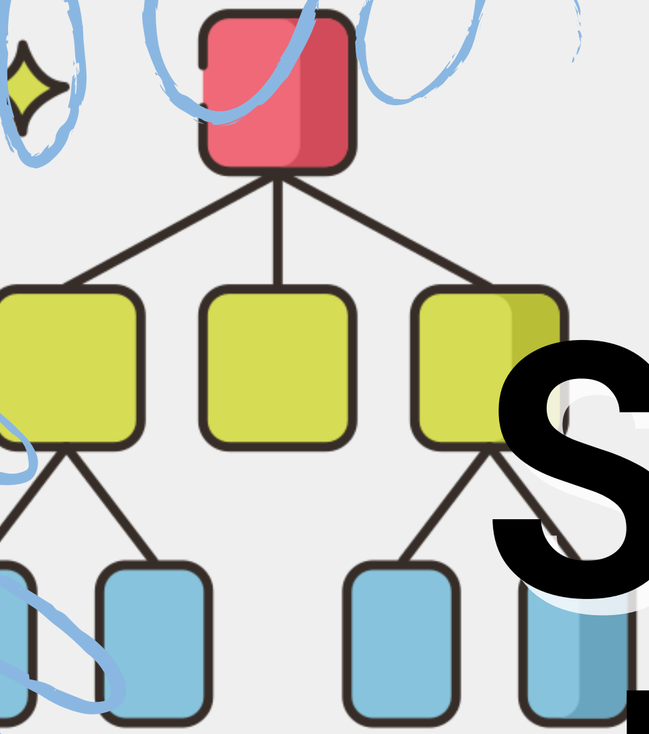

Distribucion por ausencia estudiante
Under-7
60.2%
39.8%
Above-7

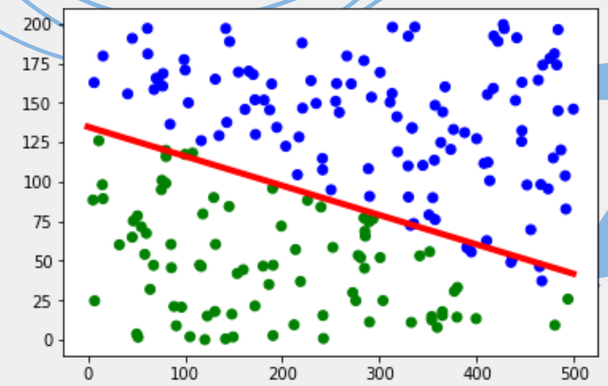
HeatMap


Relación lineal
raisedhands 1.00 0.69 0.64 0.34 0.19 0.17 0.30 -0.46 0.65
VisITedResources 0.69 1.00 0.59 0.24 0.08 0.07 0.36 -0.50 0.68
AnnouncementsView 0.64 0.59 1.00 0.42 0.20 0.16 0.30 -0.31 0.53
Discussion 0.34 0.24 0.42 1.00 0.18 0.16 0.06 -0.22 0.31
GradeID 0.19 0.08 0.20 0.18 1.00 0.95 -0.02 -0.08 0.07
StageID 0.17 0.07 0.16 0.16 0.95 1.00 -0.01 -0.11 0.08
ParentschoolSatisfaction 0.30 0.36 0.30 0.06 -0.02 -0.01 1.00 -0.23 0.38
StudentAbsenceDays -0.46 -0.50 -0.31 -0.22 -0.08 -0.11 -0.23 1.00 -0.67
Class 0.65 0.68 0.53 0.31 0.07 0.08 0.38 -0.67 1.00
raisedhands
VisITedResources
AnnouncementsView
Discussion
GradeID
StageID
ParentschoolSatisfaction
StudentAbsenceDays
Class
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

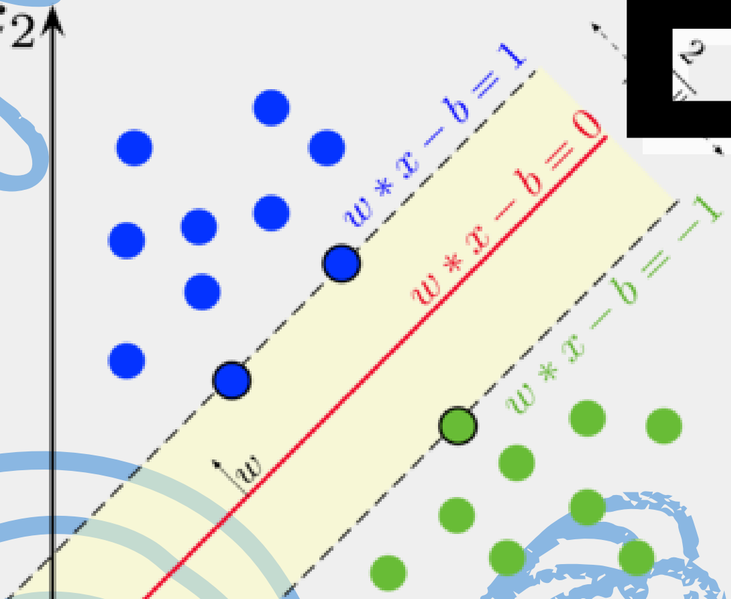
Gráficos de Dispersión

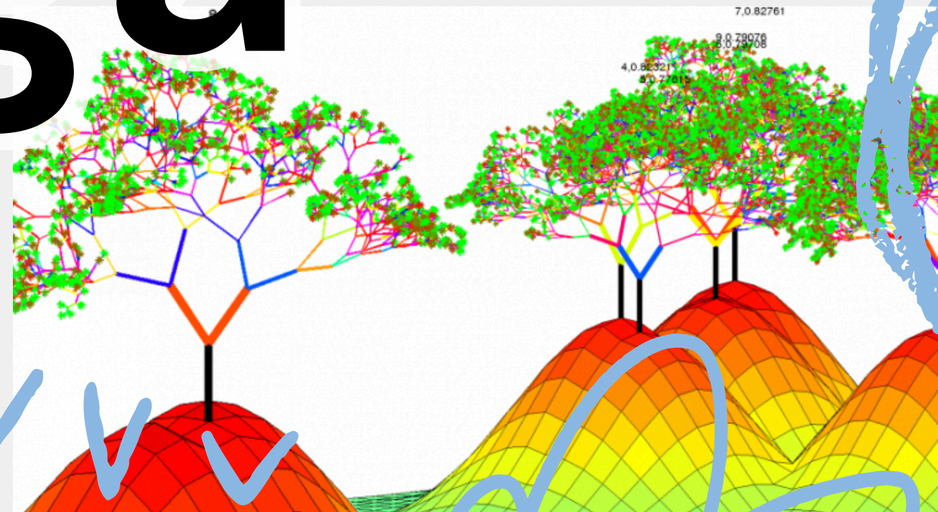






Segunda Entrega

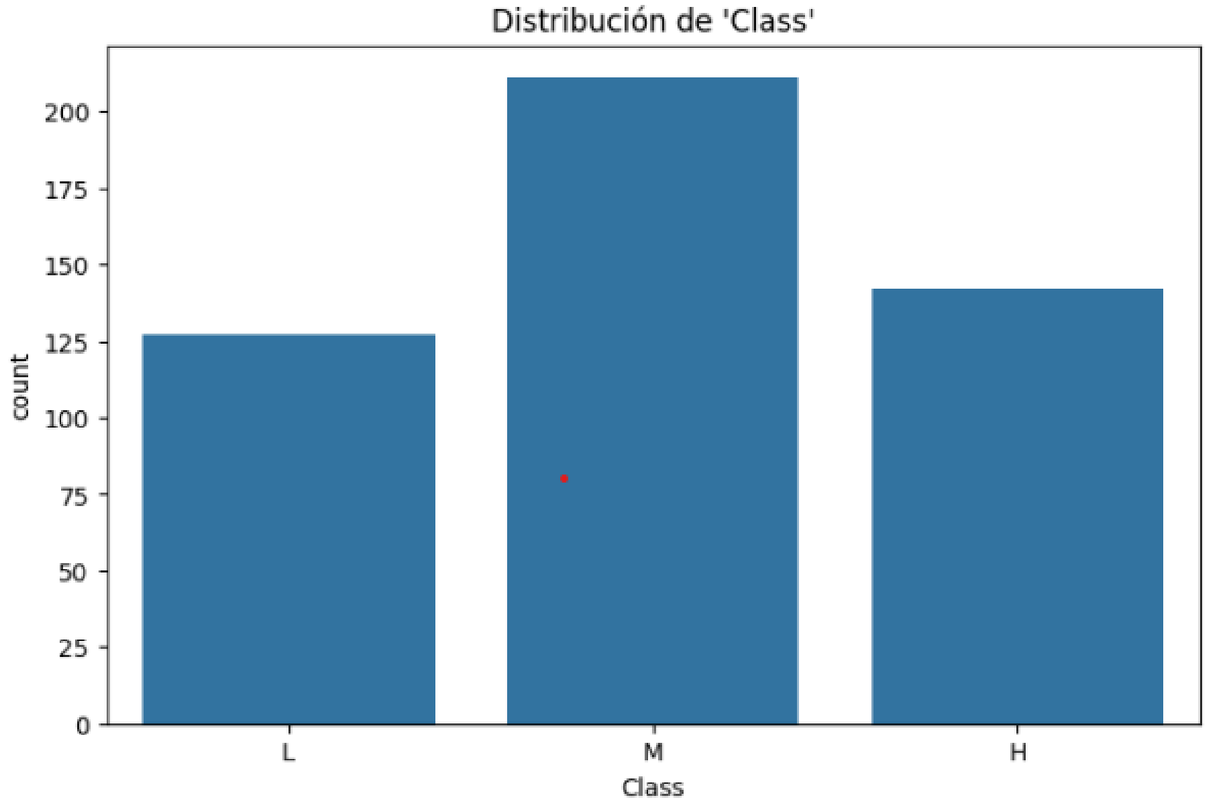




Label Encoding

```
ParentAnsweringSurvey_mapping = {'Yes': 0, 'No': 1}
df['ParentAnsweringSurvey'] = df['ParentAnsweringSurvey'].map(ParentAnsweringSurvey_mapping)

Relation_mapping = {'Father': 0, 'Mum': 1}
df['Relation'] = df['Relation'].map(Relation_mapping)

Semester_mapping = {'F': 0, 'S': 1}
df['Semester'] = df['Semester'].map(Semester_mapping)

Topic_mapping = {'IT': 0, 'Math': 1, 'Arabic': 2, 'Science':3, 'English':4, 'Quran':5, 'Spanish':6, 'French':7,
 'History':8, 'Biology':9, 'Chemistry':10, 'Geology':11}
df['Topic'] = df['Topic'].map(Topic_mapping)

SectionID_mapping = {'A': 0, 'B': 1, 'C': 2}
df['SectionID'] = df['SectionID'].map(SectionID_mapping)

PlaceofBirth_mapping = {'KuwaIT':0, 'lebanon':1, 'Egypt':2, 'SaudiArabia':3, 'USA':4, 'Jordan':5, 'venzuela':6,
                        'Iran':7, 'Tunis':8, 'Morocco':9, 'Syria':10, 'Iraq':11, 'Palestine':12, 'Lybia':13}
df['PlaceofBirth'] = df['PlaceofBirth'].map(PlaceofBirth_mapping)

NationalITy_mapping = {'KW':0, 'lebanon':1, 'Egypt':2, 'SaudiArabia':3, 'USA':4, 'Jordan':5, 'venzuela':6,
                       'Iran':7, 'Tunis':8, 'Morocco':9, 'Syria':10, 'Iraq':11, 'Palestine':12, 'Lybia':13}
df['NationalITy'] = df['NationalITy'].map(NationalITy_mapping)

gender_mapping = {'M': 0, 'F': 1}
df['gender'] = df['gender'].map(gender_mapping)

StageID_mapping = {'lowerlevel': 0, 'MiddleSchool': 1, 'HighSchool': 2}
df['StageID'] = df['StageID'].map(StageID_mapping)

GradeID_mapping = {
    'G-01': 1, 'G-02': 2, 'G-03': 3, 'G-04': 4, 'G-05': 5,
    'G-06': 6, 'G-07': 7, 'G-08': 8, 'G-09': 9, 'G-10': 10,
    'G-11': 11, 'G-12': 12
}
df['GradeID'] = df['GradeID'].map(GradeID_mapping)

ParentschoolSatisfaction_mapping = {'Good': 1, 'Bad': 0}
df['ParentschoolSatisfaction'] = df['ParentschoolSatisfaction'].map(ParentschoolSatisfaction_mapping)

StudentAbsenceDays_mapping = {'Under-7': 0, 'Above-7': 1}
df['StudentAbsenceDays'] = df['StudentAbsenceDays'].map(StudentAbsenceDays_mapping)
```

Distribución




```
Class
M    43.958333
H    29.583333
L    26.458333
```

```python
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='Class', order=['L', 'M', 'H'])
plt.title("Distribución de 'Class'")
plt.show()

print(df['Class'].value_counts(normalize=True) * 100)
```

Decision Tree

[12]:

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

X = df.values[:,:-1]
y = df.values[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21

est= DecisionTreeClassifier()
est.fit(X_train,y_train)
print(accuracy_score(est.predict(X_test), y_test))
```

```
0.7604166666666666
```

Learning Curve

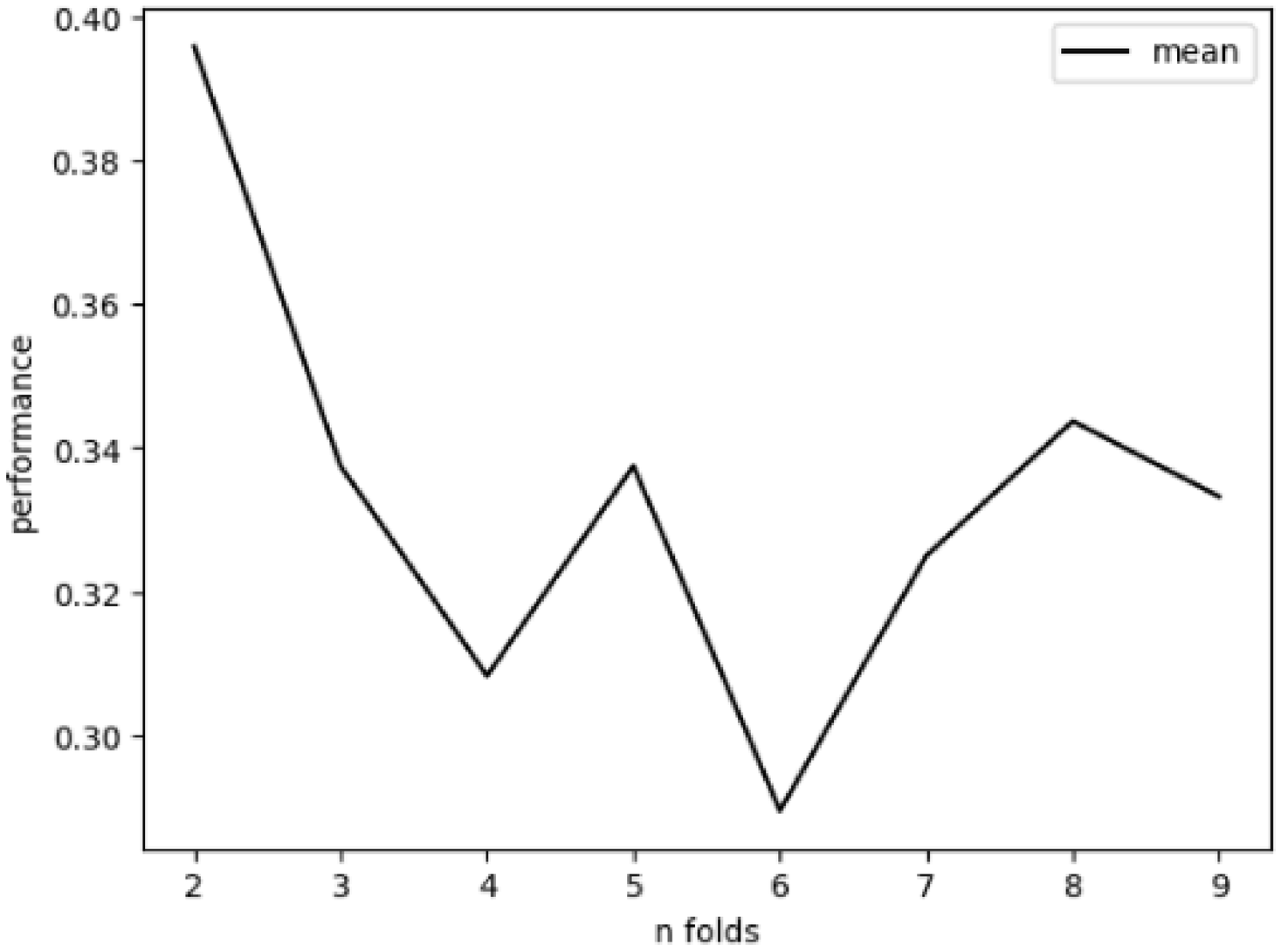
In [13]:

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

est_depth = DecisionTreeClassifier(max_depth=3)
est_depth.fit(X_train, y_train)
print("Acuraccy con max_depth=3: ",accuracy_score(est_depth.predict(X_test), y_test))

def curve(est):
    means = []
    nfolds_range = range(2,10)
    for nfolds in nfolds_range:
        #print (nfolds,)
        s = cross_val_score(est, X, y, cv=KFold(nfolds, shuffle=True), scoring=make_scorer(mean_squared_error))
        means.append(np.mean(s))

    means = np.r_[means]

    plt.plot(nfolds_range, means, label="mean", color="black")
    plt.xlabel("n folds")
    plt.ylabel("performance")
    plt.legend()
```

```
Acuraccy con max_depth=3:  0.7708333333333334
```

In [14]:

```
from sklearn.model_selection import cross_val_score
curve(est_depth)
```

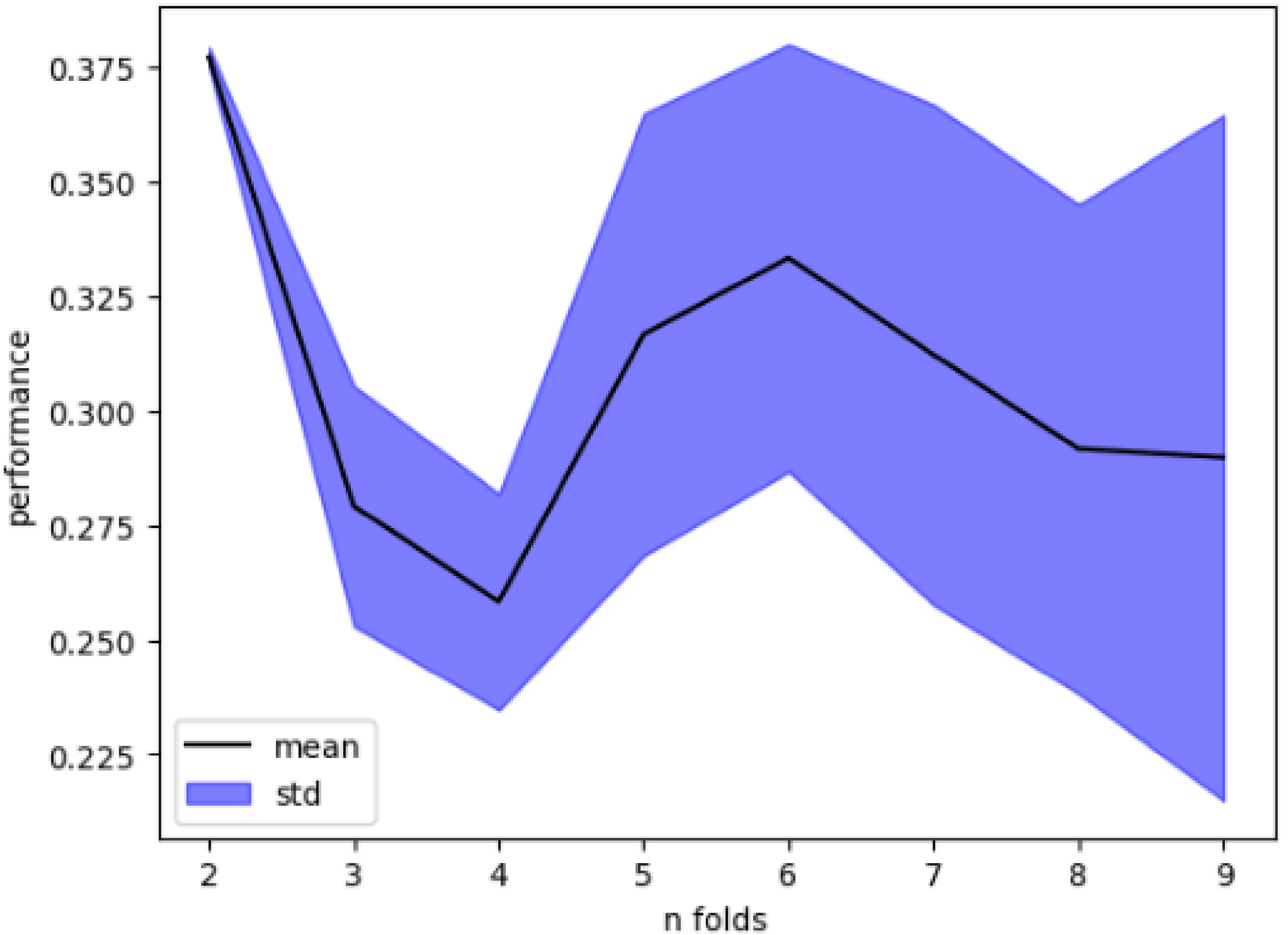


Buscamos el mejor n-fold

```
ef curve_std(est):
    means, stds = [], []
    nfolds_range = range(2,10)
    for nfolds in nfolds_range:
        #print (nfolds,)
        s = cross_val_score(est, X, y, cv=KFold(nfolds, shuffle=True), scoring=make_scorer(mean_squared_error))
        means.append(np.mean(s))
        stds.append(np.std(s))

    means = np.r_[means]
    stds  = np.r_[stds]

    plt.plot(nfolds_range, means, label="mean", color="black")
    plt.fill_between(nfolds_range, means-stds, means+stds, color="blue", alpha=.5, label="std")
    plt.xlabel("n folds")
    plt.ylabel("performance")
    plt.legend()
```

In [15]:

```
from sklearn.model_selection import cross_val_score
curve_std(est)
```



Random Forest

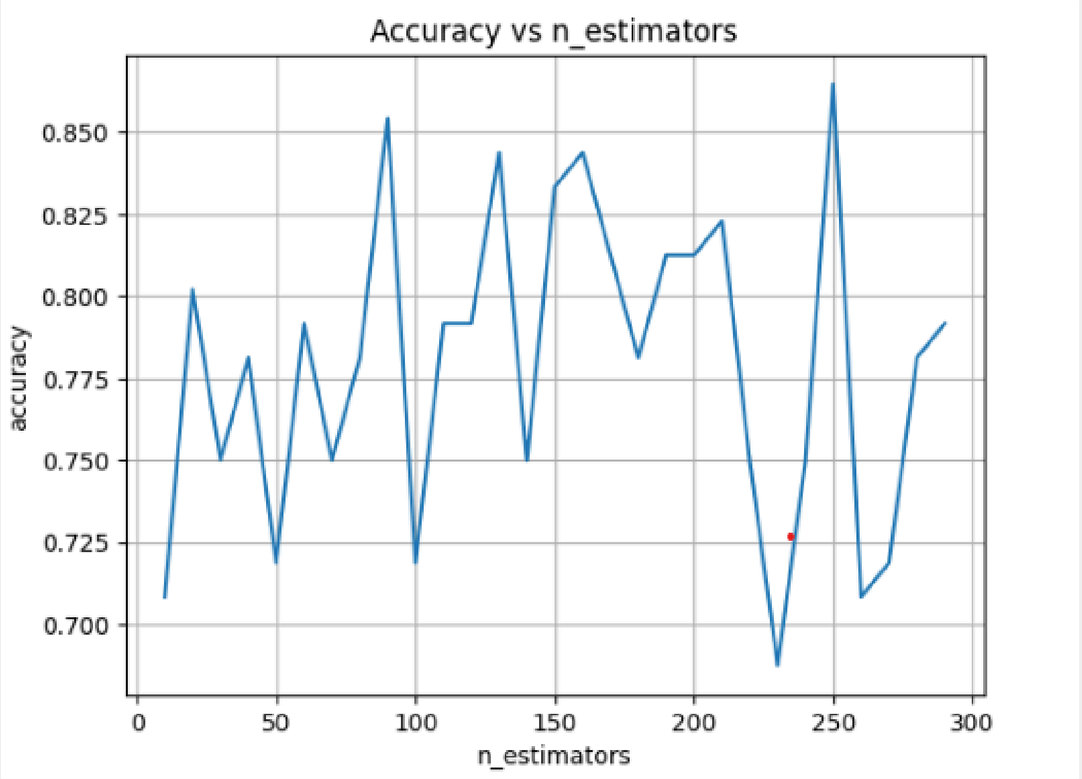
Estimador

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

X = df.values[:, :-1]
y = df.values[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

rf = RandomForestClassifier()
rf.fit(X_train, y_train)
print("El accuracy es: ",accuracy_score(rf.predict(X_test), y_test))
```

```
El accuracy es:  0.8020833333333334
```

Learning Curve

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

X = df.values[:, :-1]
y = df.values[:, -1]

n_vals = list(range(10, 300, 10))
accs = []

for n in n_vals:
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
    model = RandomForestClassifier(n_estimators=n)
    model.fit(X_train, y_train)
    acc = accuracy_score(y_test, model.predict(X_test))
    accs.append(acc)

plt.plot(n_vals, accs)
plt.xlabel("n_estimators")
plt.ylabel("accuracy")
plt.title("Accuracy vs n_estimators")
plt.grid()
plt.show()

best_index = accs.index(max(accs))
best_n = n_vals[best_index]

print("Mejor n_estimators:", best_n)
```



SVM

SVM

[14]

```python
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

est = SVC()
X = df.values[:,:-1]
y = df.values[:,-1]
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21)
est.fit(x_train, y_train)
print(accuracy_score(est.predict(x_test), y_test))
```

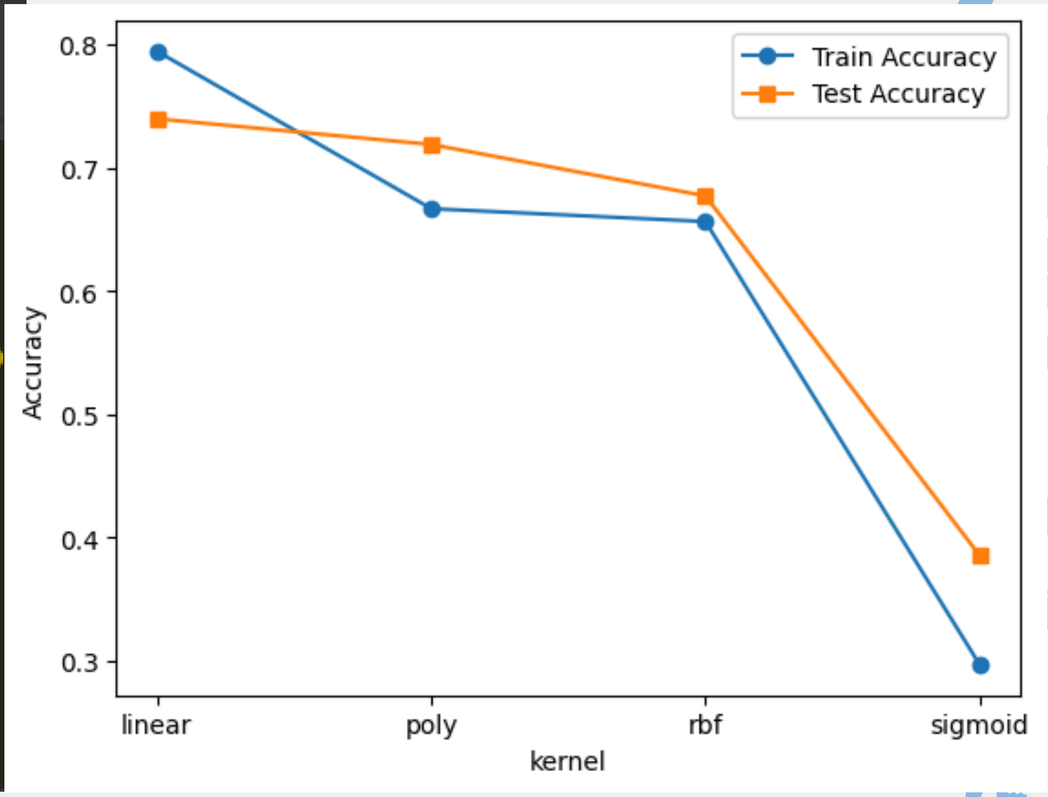
```
0.6770833333333334
```

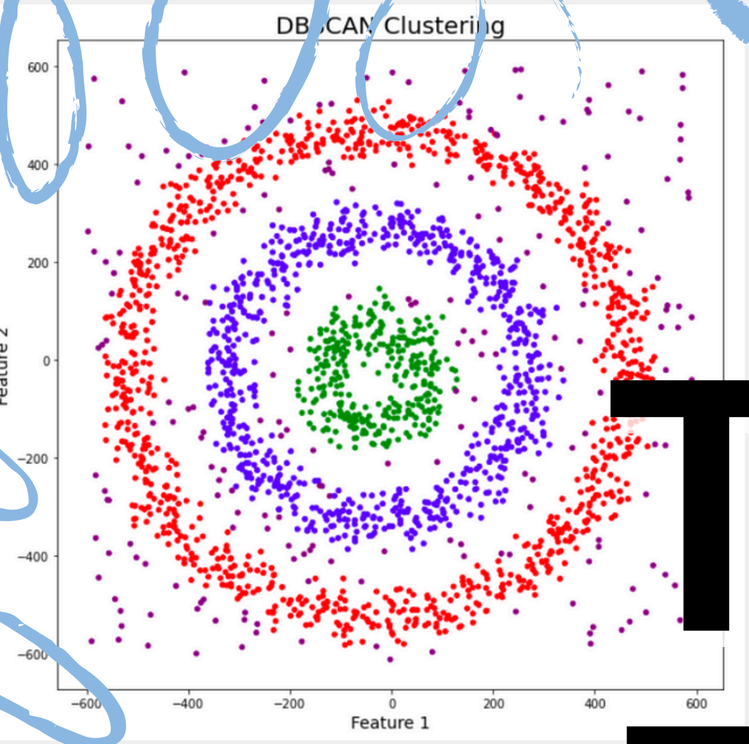
Learning Curve

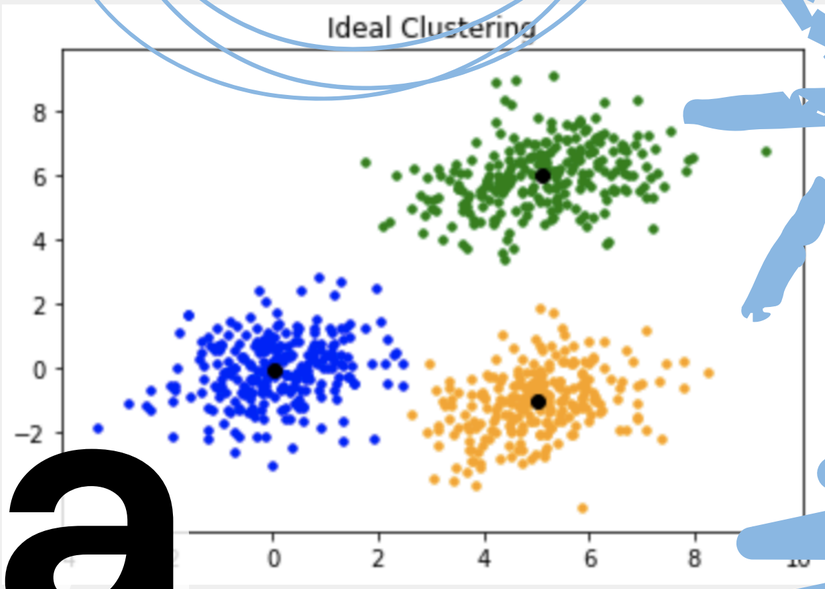
Learning curve

[33]

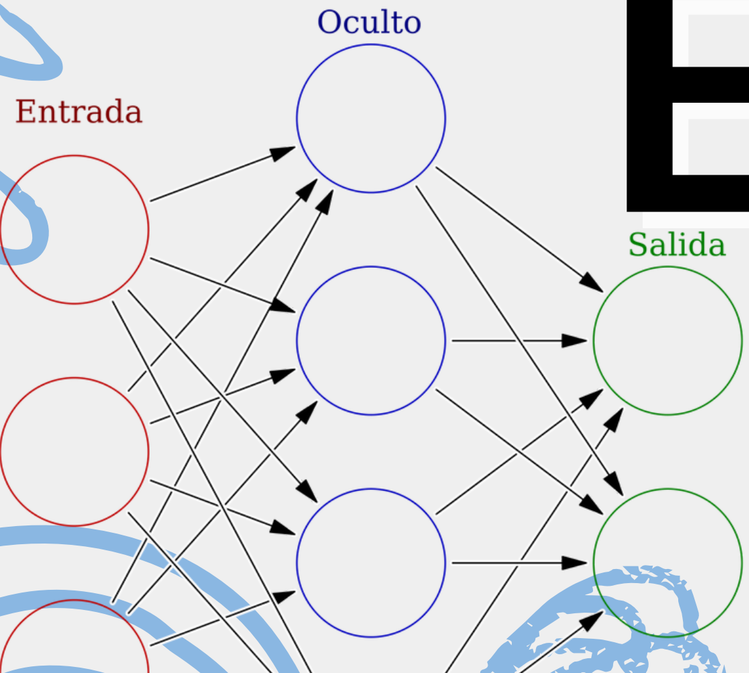
```python
kernel_functions = ['linear', 'poly', 'rbf', 'sigmoid']
train_scores = []
test_scores = []
for kernel in kernel_functions:
  model = SVC(kernel=kernel)
  x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=21)
  model.fit(x_train, y_train)
  train_accuracy = accuracy_score(y_train, model.predict(x_train))
  test_accuracy = accuracy_score(y_test, model.predict(x_test))
  train_scores.append(train_accuracy)
  test_scores.append(test_accuracy)
plt.plot(kernel_functions, train_scores, marker='o', label='Train Accuracy')
plt.plot(kernel_functions, test_scores, marker='s', label='Test Accuracy')
plt.xlabel("kernel")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

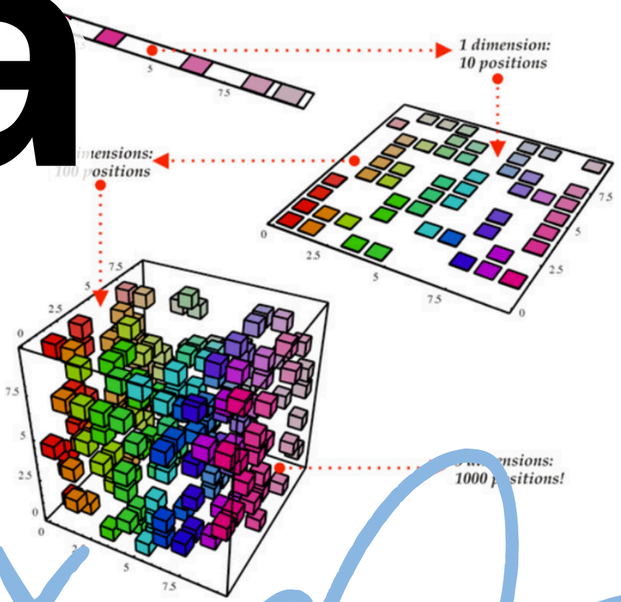






Tercera Entrega





Clasificación con red neuronal

Codigo general

```python
import tensorflow as tf
from tensorflow import keras
import numpy as np
from sklearn.metrics import accuracy_score

tf.random.set_seed(21)
np.random.seed(21)

unique_values = len(np.unique(y))
capas_ocultas = [3, 6, 9]

for n in capas_ocultas:
    print(f"\nEntrenando modelo con {n} capas ocultas...")

    model = keras.Sequential()
    model.add(keras.layers.Input(shape=(x_train.shape[1],)))

    for _ in range(n):
        model.add(keras.layers.Dense(128, activation='relu'))

    model.add(keras.layers.Dense(unique_values, activation='softmax'))

    model.compile(optimizer=tf.keras.optimizers.SGD(),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
```

```python
model.fit(x_train, y_train, epochs=10, verbose=1)

y_pred = model.predict(X_test, verbose=0)
y_pred = np.argmax(y_pred, axis=1)

acc = accuracy_score(y_test, y_pred)
print(f"Accuracy con {n} capas ocultas: {acc:.4f}")
```

3 capas oculta, 128 neuronas,relu

```
Entrenando modelo con 3 capas ocultas...
Epoch 1/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.3780 - loss: 5.0060
Epoch 2/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.4919 - loss: 0.9047
Epoch 3/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5118 - loss: 0.8517
Epoch 4/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5532 - loss: 0.8159
Epoch 5/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5968 - loss: 0.8014
Epoch 6/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6350 - loss: 0.7767
Epoch 7/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6364 - loss: 0.7641
Epoch 8/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6413 - loss: 0.7514
Epoch 9/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6561 - loss: 0.7408
Epoch 10/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6653 - loss: 0.7254
Accuracy con 3 capas ocultas: 0.2188
```

6 capas oculta, 128 neuronas,relu

```
Entrenando modelo con 6 capas ocultas...
Epoch 1/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.3226 - loss: 2.3991
Epoch 2/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.4842 - loss: 0.9185
Epoch 3/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.4928 - loss: 0.8835
Epoch 4/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5270 - loss: 0.8522
Epoch 5/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5436 - loss: 0.8348
Epoch 6/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5665 - loss: 0.8196
Epoch 7/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5978 - loss: 0.8080
Epoch 8/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5897 - loss: 0.7997
Epoch 9/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.6108 - loss: 0.7827
Epoch 10/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.5919 - loss: 0.7846
Accuracy con 6 capas ocultas: 0.2396
```

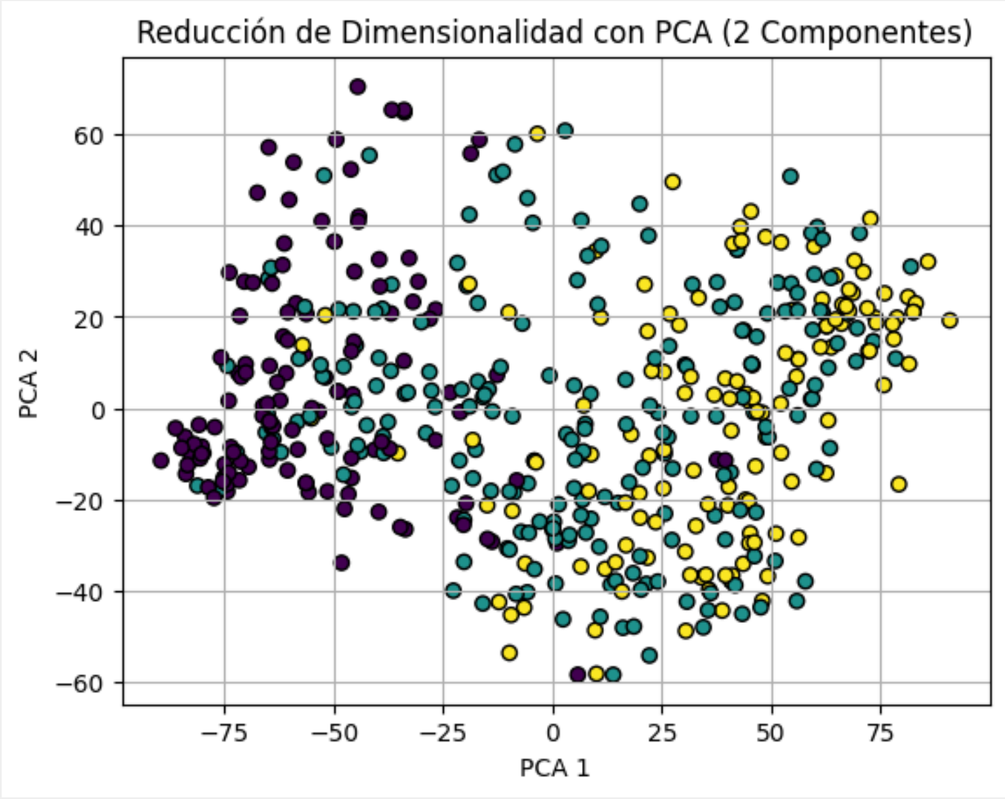
10 capas oculta, 128 neuronas,relu

```
Entrenando modelo con 10 capas ocultas...
Epoch 1/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.2808 - loss: 1.0772
Epoch 2/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.3913 - loss: 0.9659
Epoch 3/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.4331 - loss: 0.9339
Epoch 4/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5262 - loss: 0.9088
Epoch 5/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.5326 - loss: 0.8882
Epoch 6/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5655 - loss: 0.8675
Epoch 7/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.5599 - loss: 0.8559
Epoch 8/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5854 - loss: 0.8450
Epoch 9/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5971 - loss: 0.8312
Epoch 10/10
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6213 - loss: 0.8220
Accuracy con 10 capas ocultas: 0.2917
```

Reduccion dimensional y Unsupervised learning

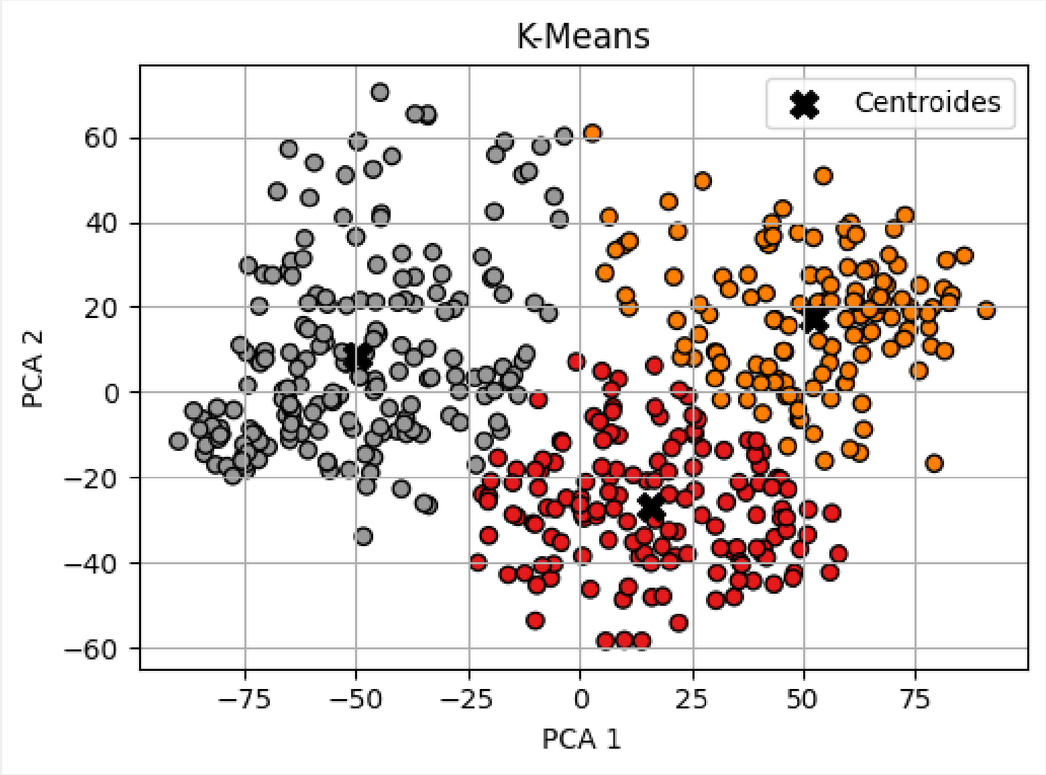
PCA

```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis', edgecolor='k')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.title('Reducción de Dimensionalidad con PCA (2 Componentes)')
plt.grid(True)
plt.show()
```



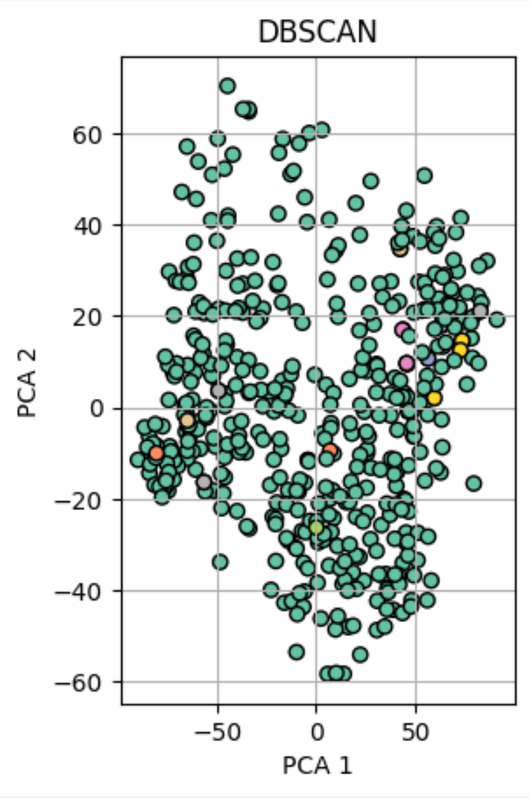
Kmeans

```python
from sklearn.cluster import KMeans, DBSCAN
import matplotlib.pyplot as plt

#K-Means
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(X_pca)

plt.figure(figsize=(10, 4))

# Gráfico de K-Means
plt.subplot(1, 2, 1)
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans_labels, cmap='Set1', edgecolor='k')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            c='black', marker='X', s=100, label='Centroides')
plt.title('K-Means')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()

```



DBScan

```python
from sklearn.cluster import DBSCAN

#DBSCAN
dbscan = DBSCAN(min_samples=2)
dbscan_labels = dbscan.fit_predict(X_pca)

# Gráfico de DBSCAN
plt.subplot(1, 2, 2)
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=dbscan_labels, cmap='Set2', edgecolor='k')
plt.title('DBSCAN')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.grid(True)

plt.show()
```



¡Muchas gracias!