

# 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.



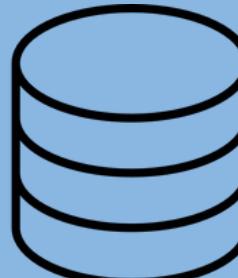
# Columnas Relevantes



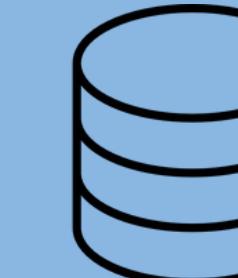
Class: Rendimiento del estudiante en 3 categorías.



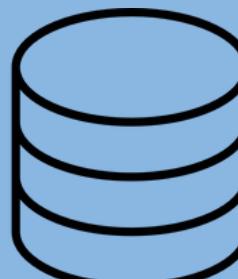
StudentAbsenceDays: Días de ausencia del estudiante (Menos de 7 vs. Más de 7).



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



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



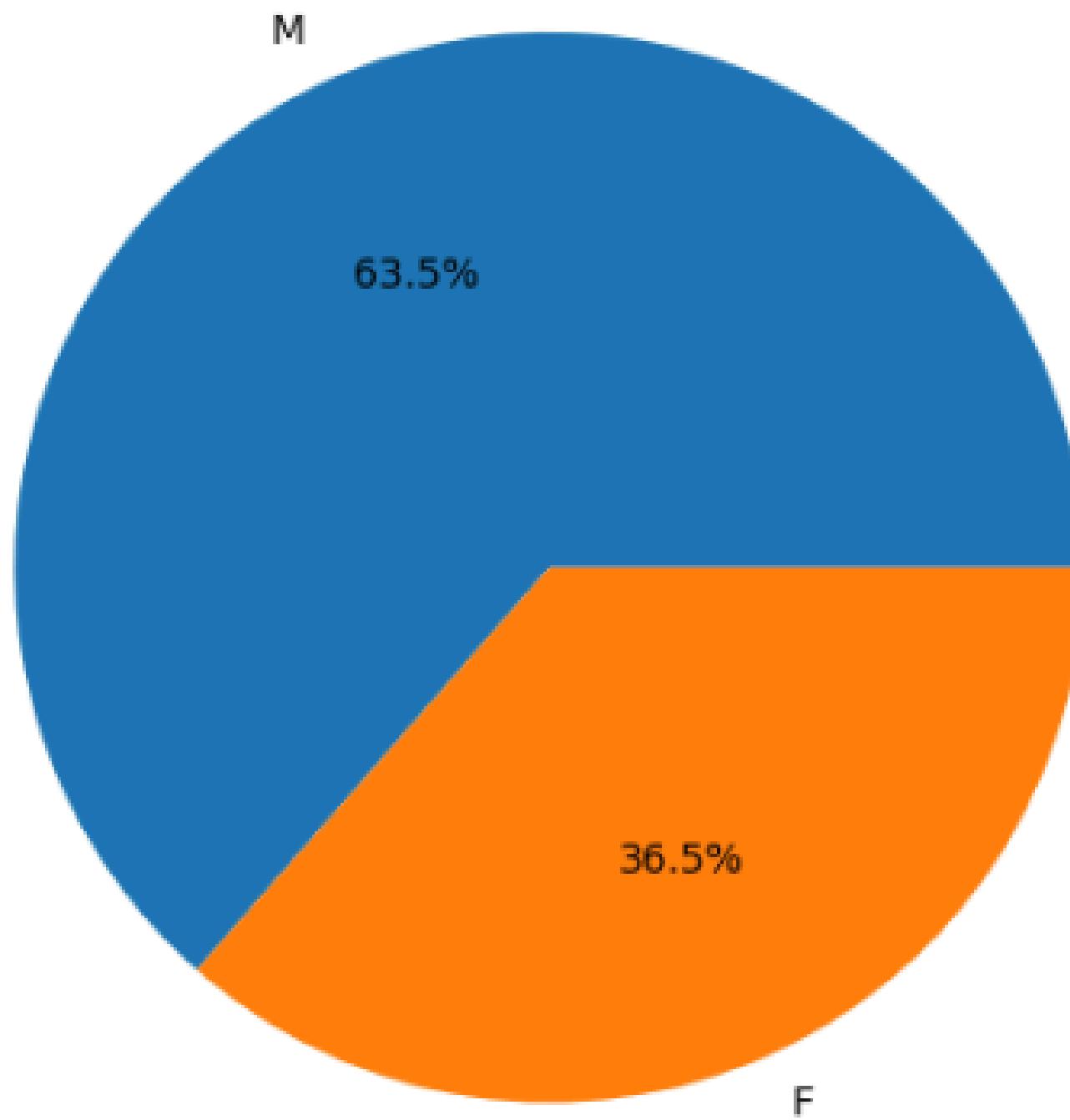
VisITedResources: Número de recursos educativos visitados.



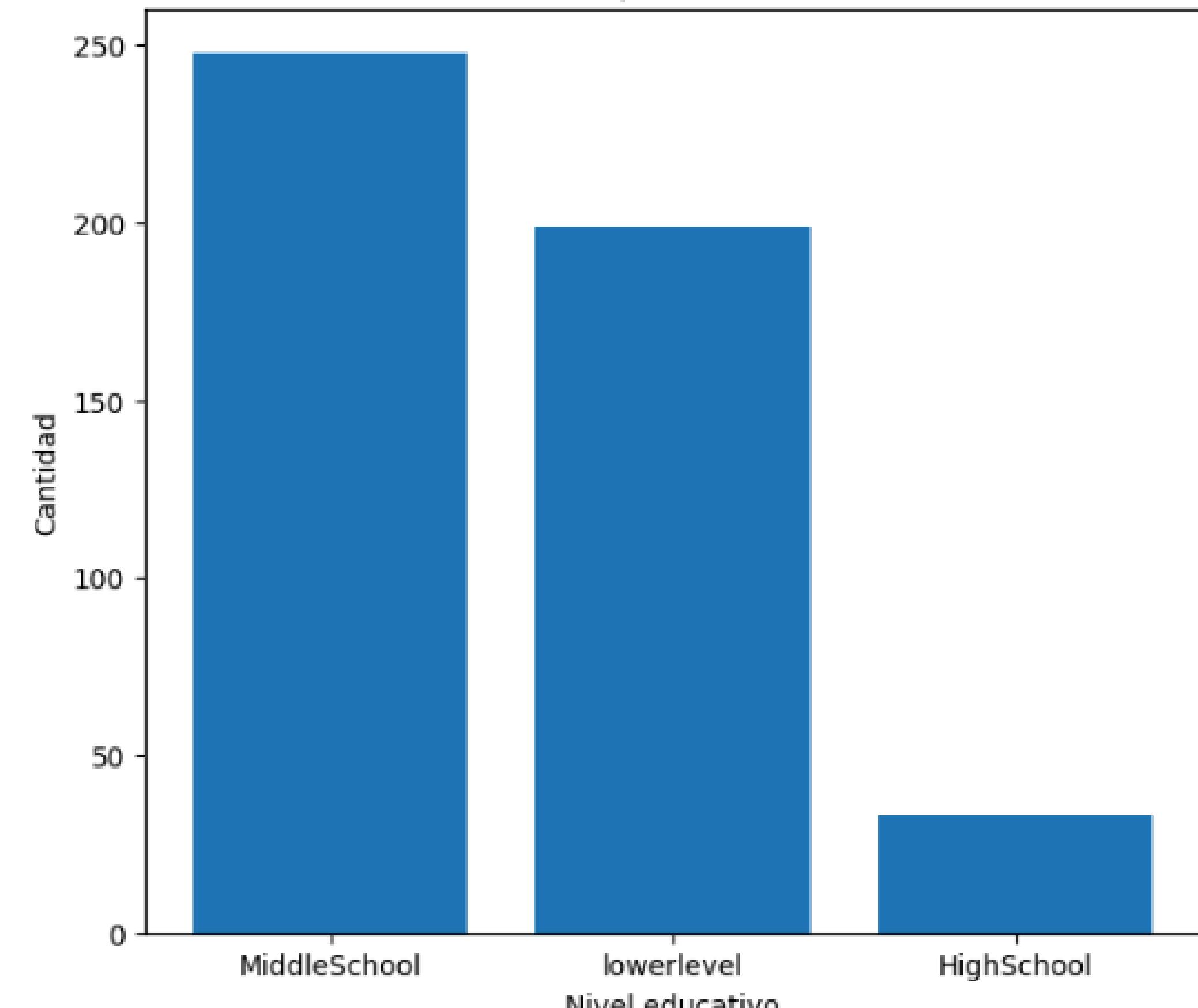
StageID: Nivel educativo (Lower Level, Middle School, High School).

# Graficas por columna

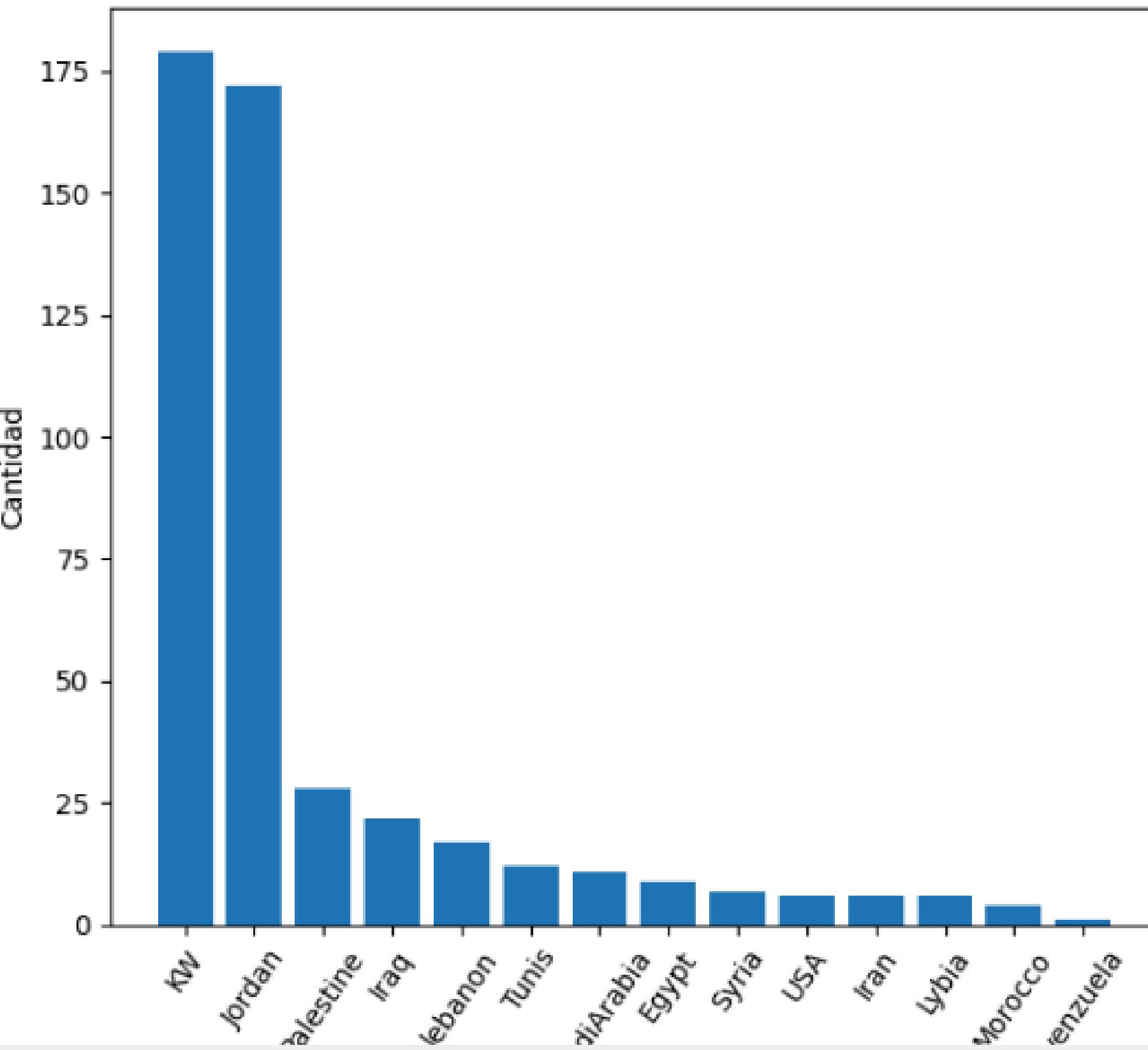
Distribucion por genero



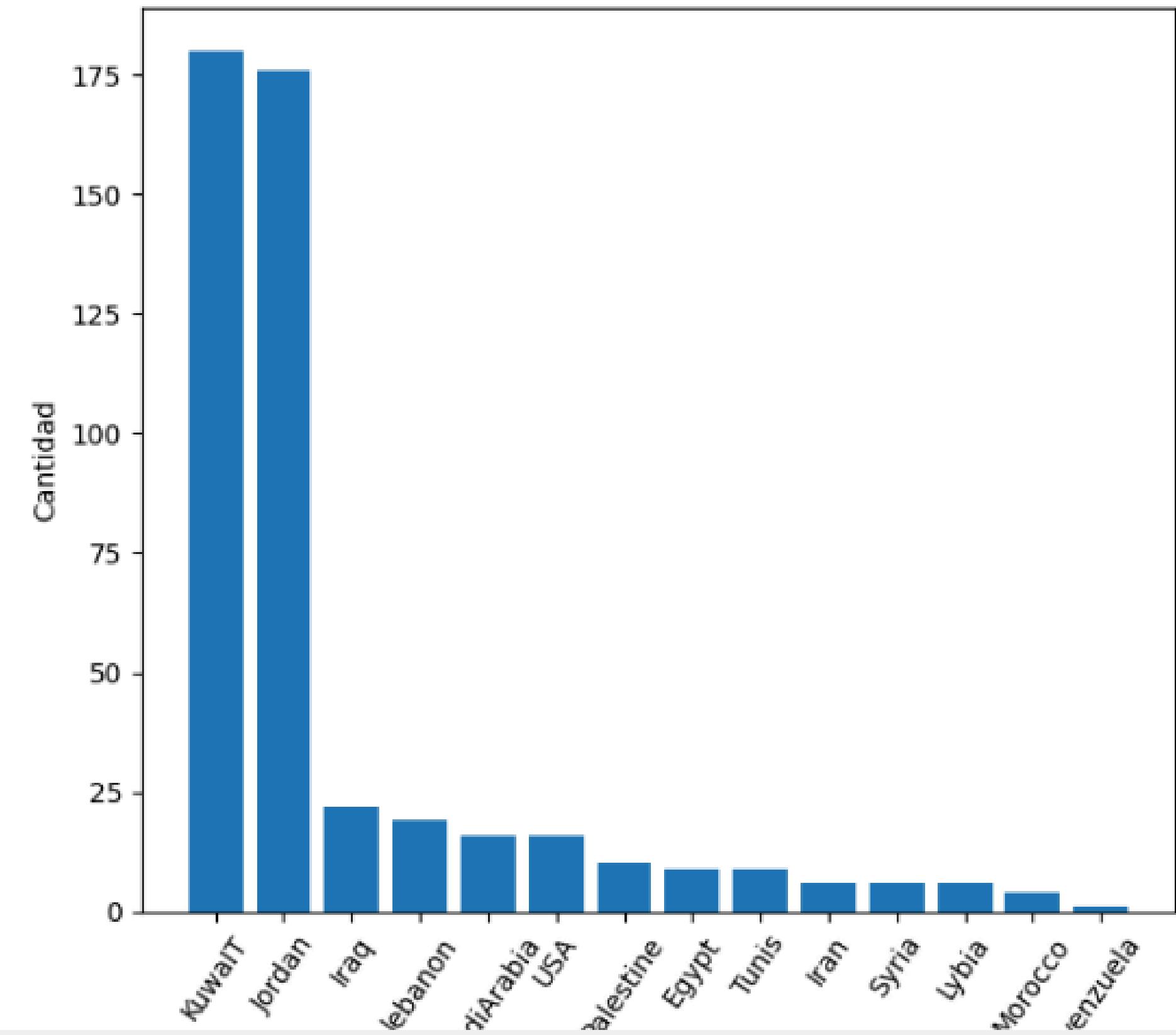
Distribucion por nivel educativo



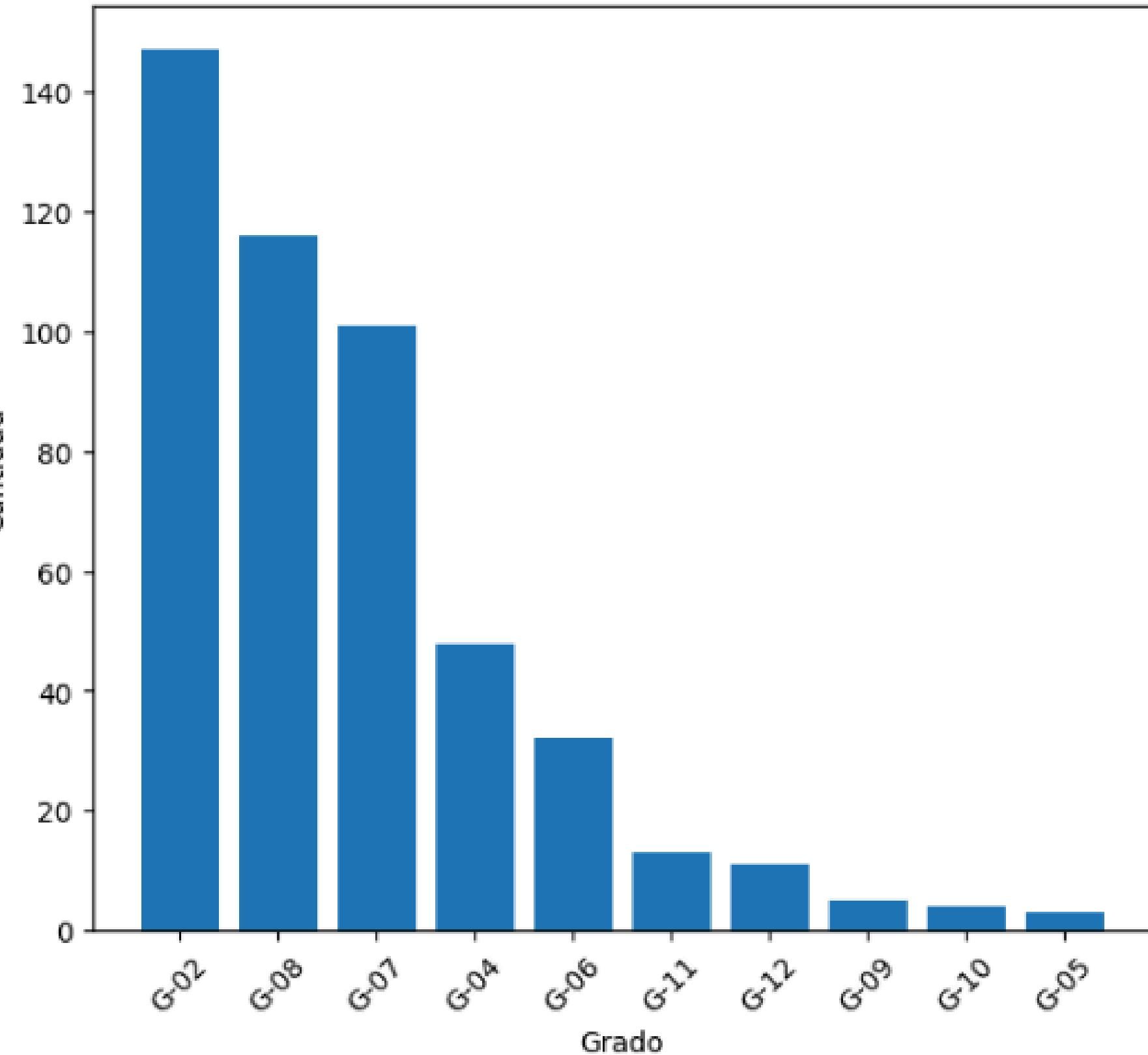
Distribución por nacionalidad



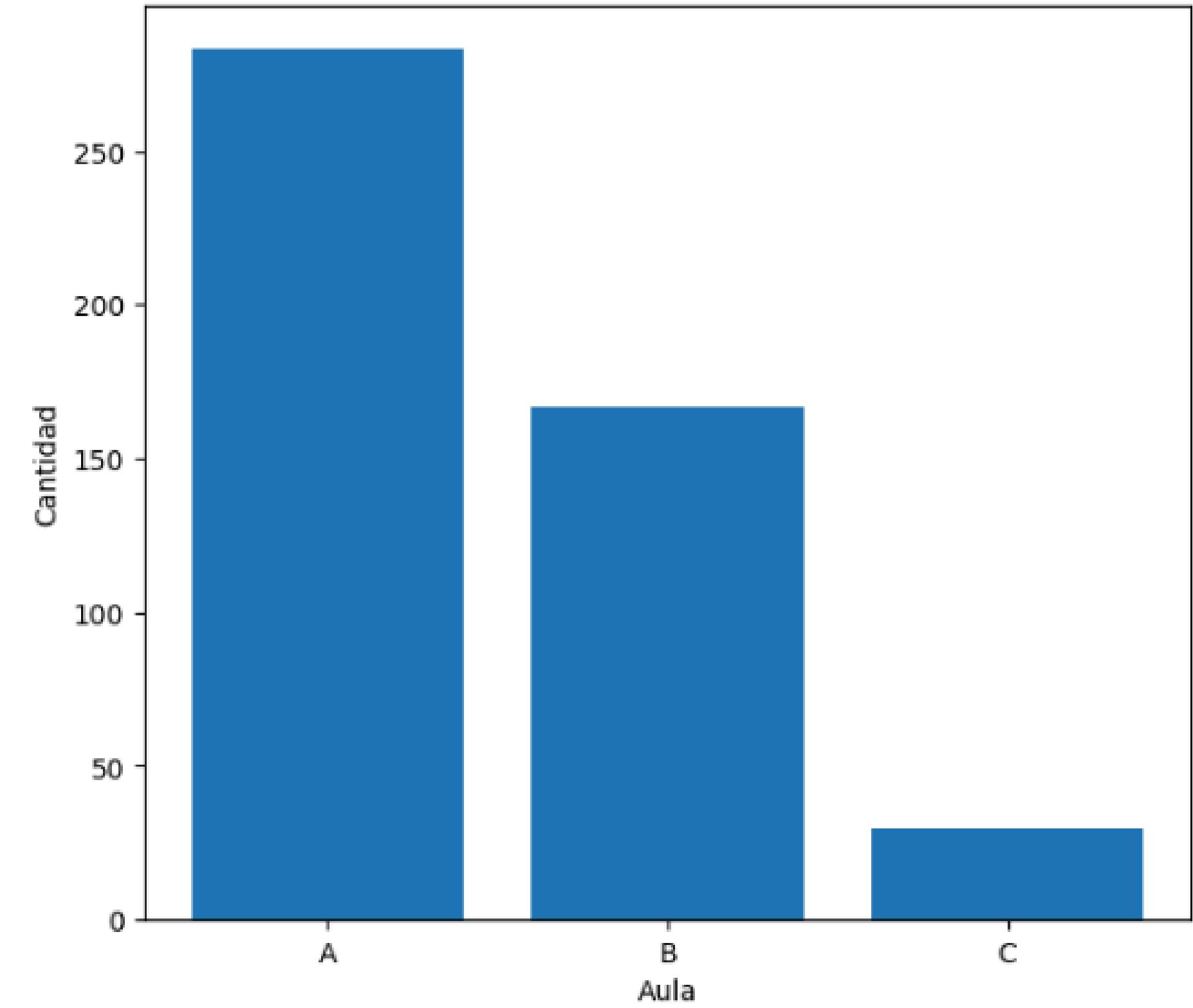
Distribución por lugar de nacimiento



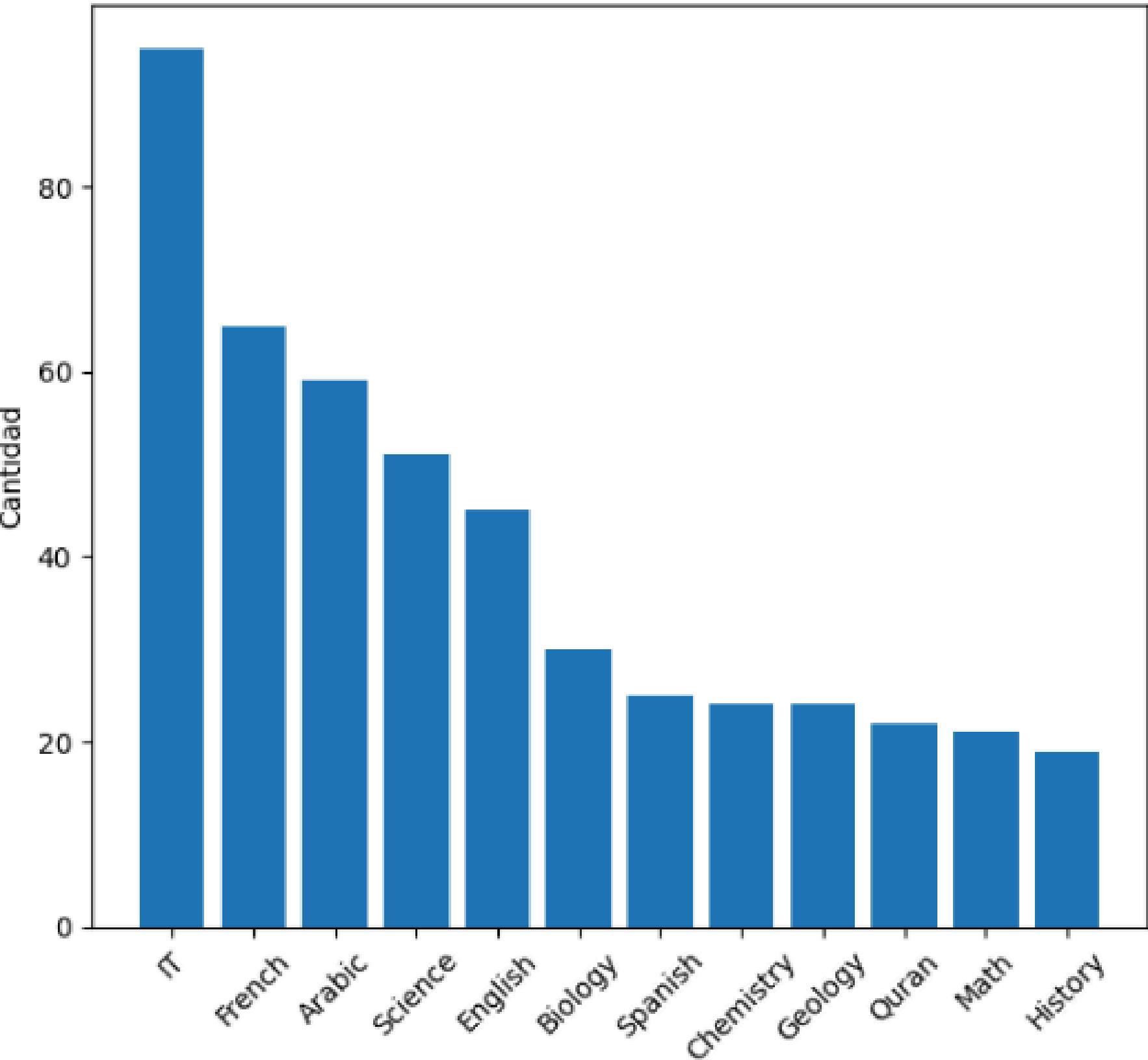
Distribución por nivel de grado



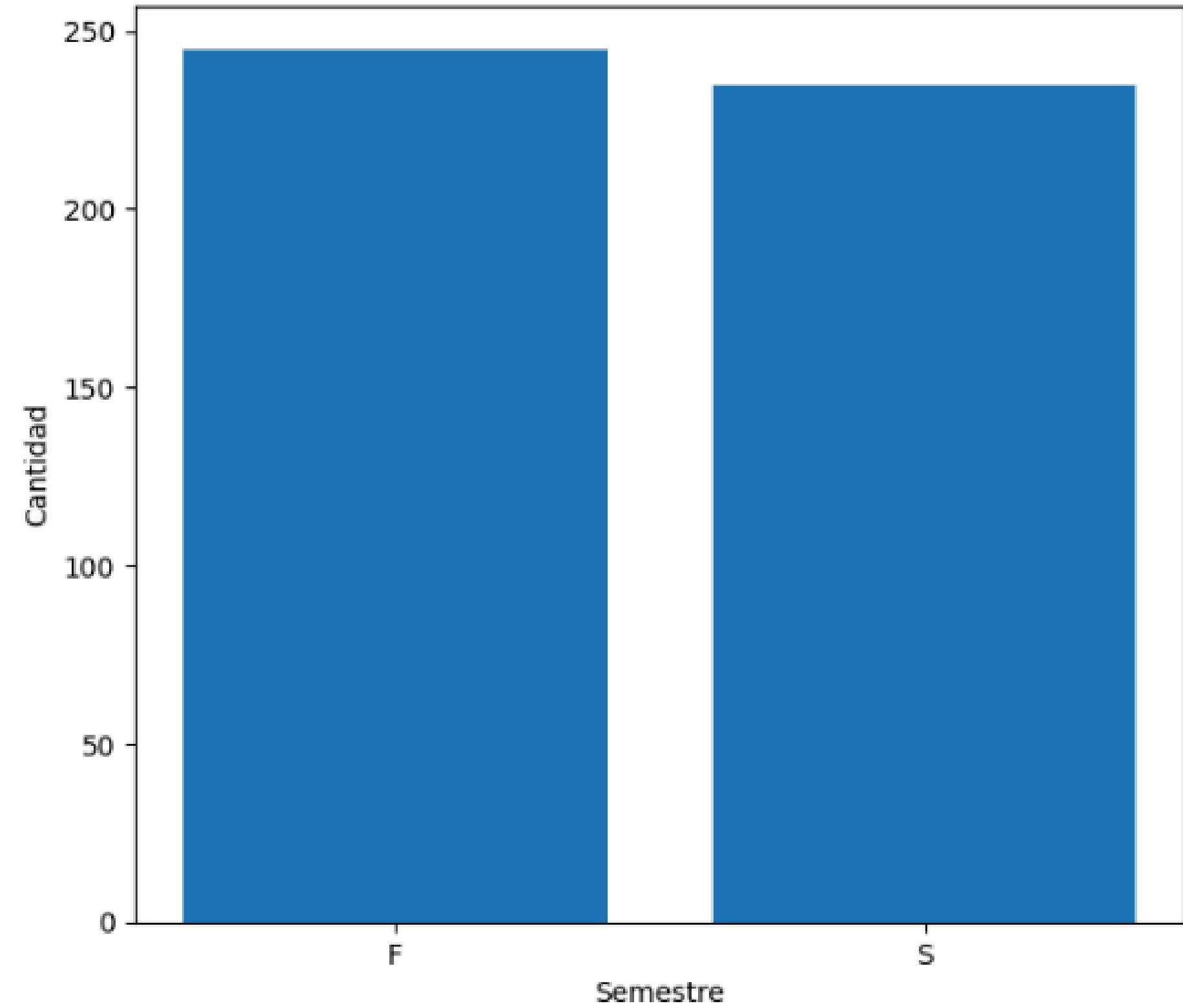
Distribución por aula



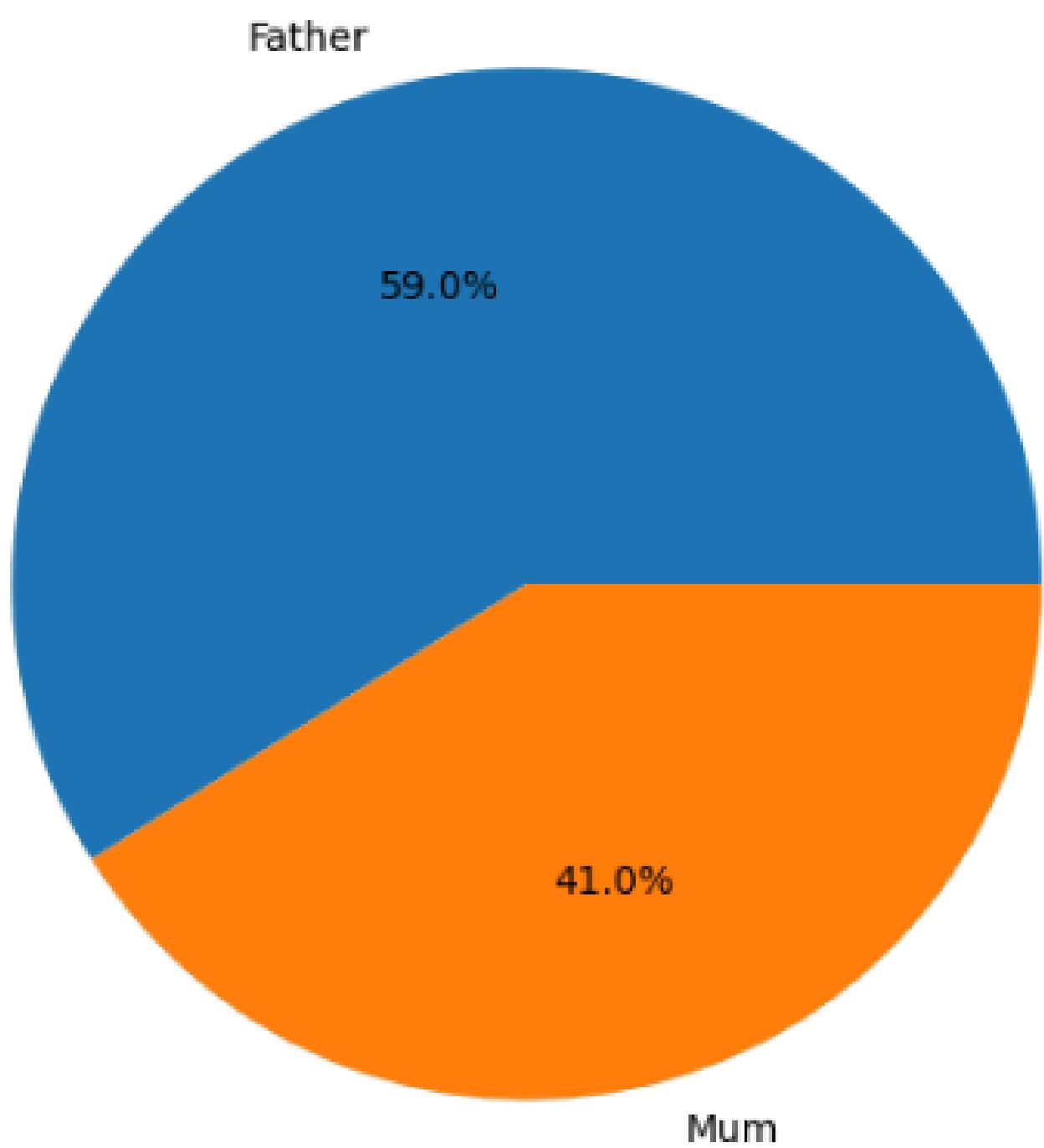
Distribución por tema de curso



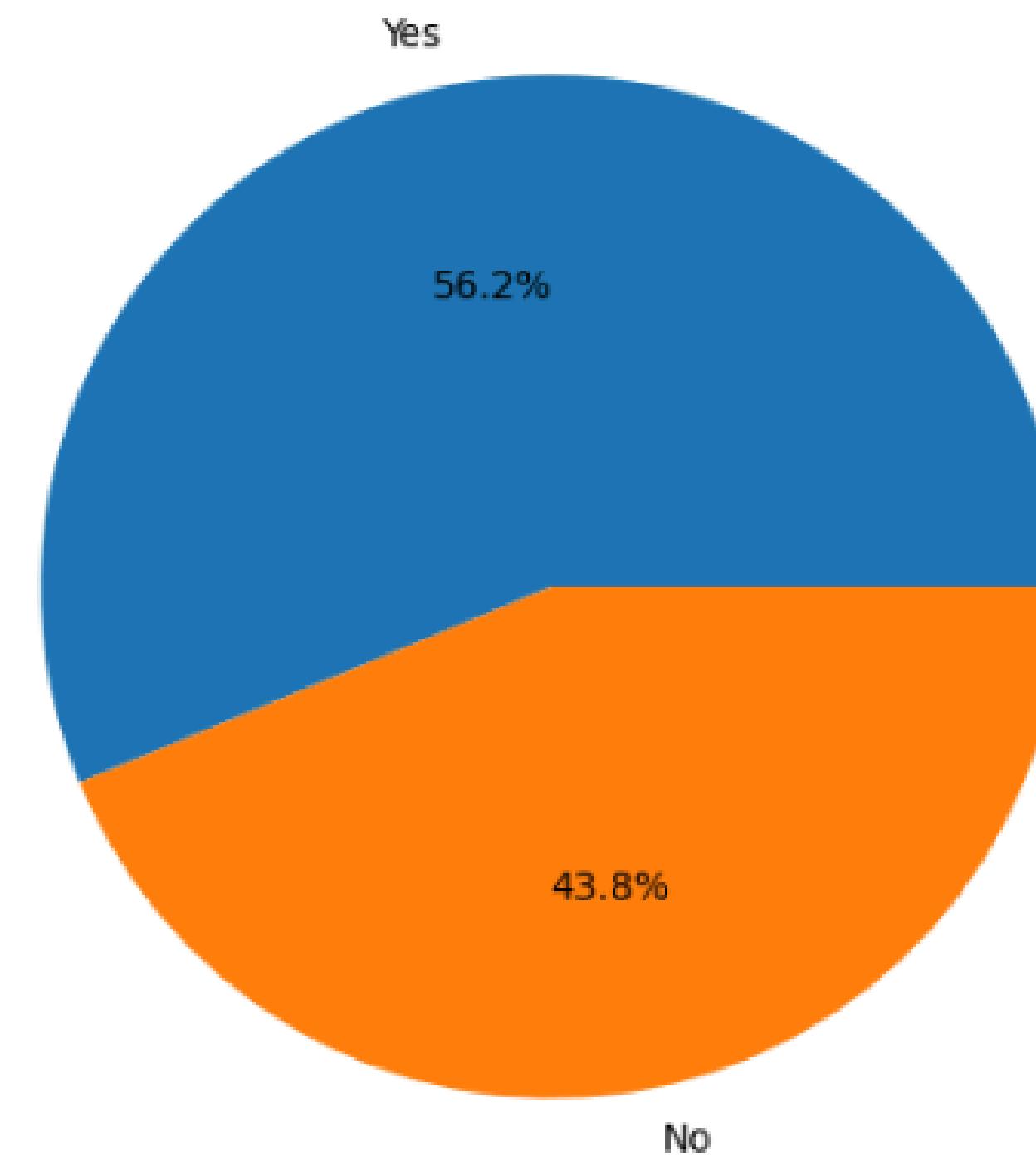
Distribución por semestre



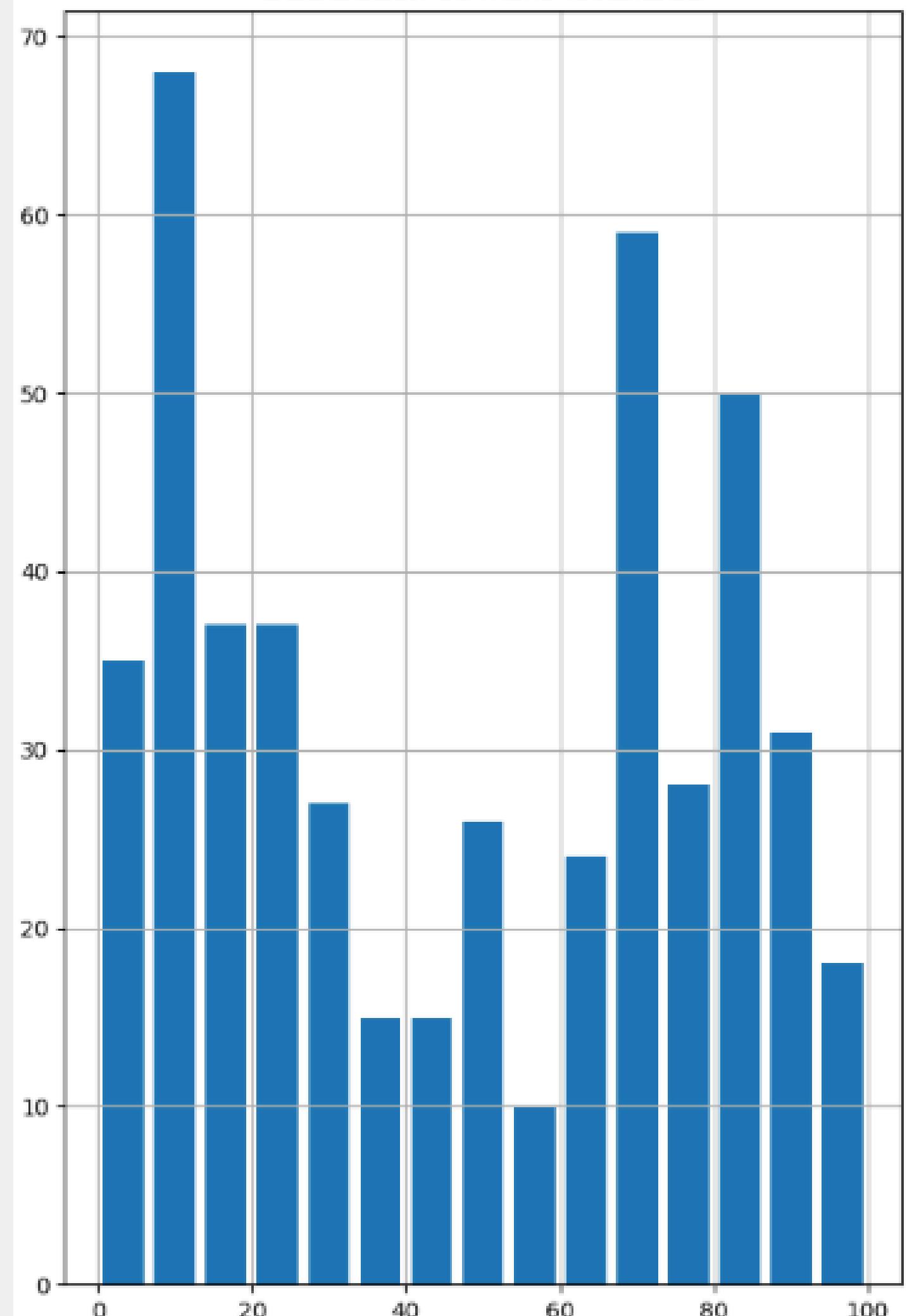
Distribucion por padre responsable



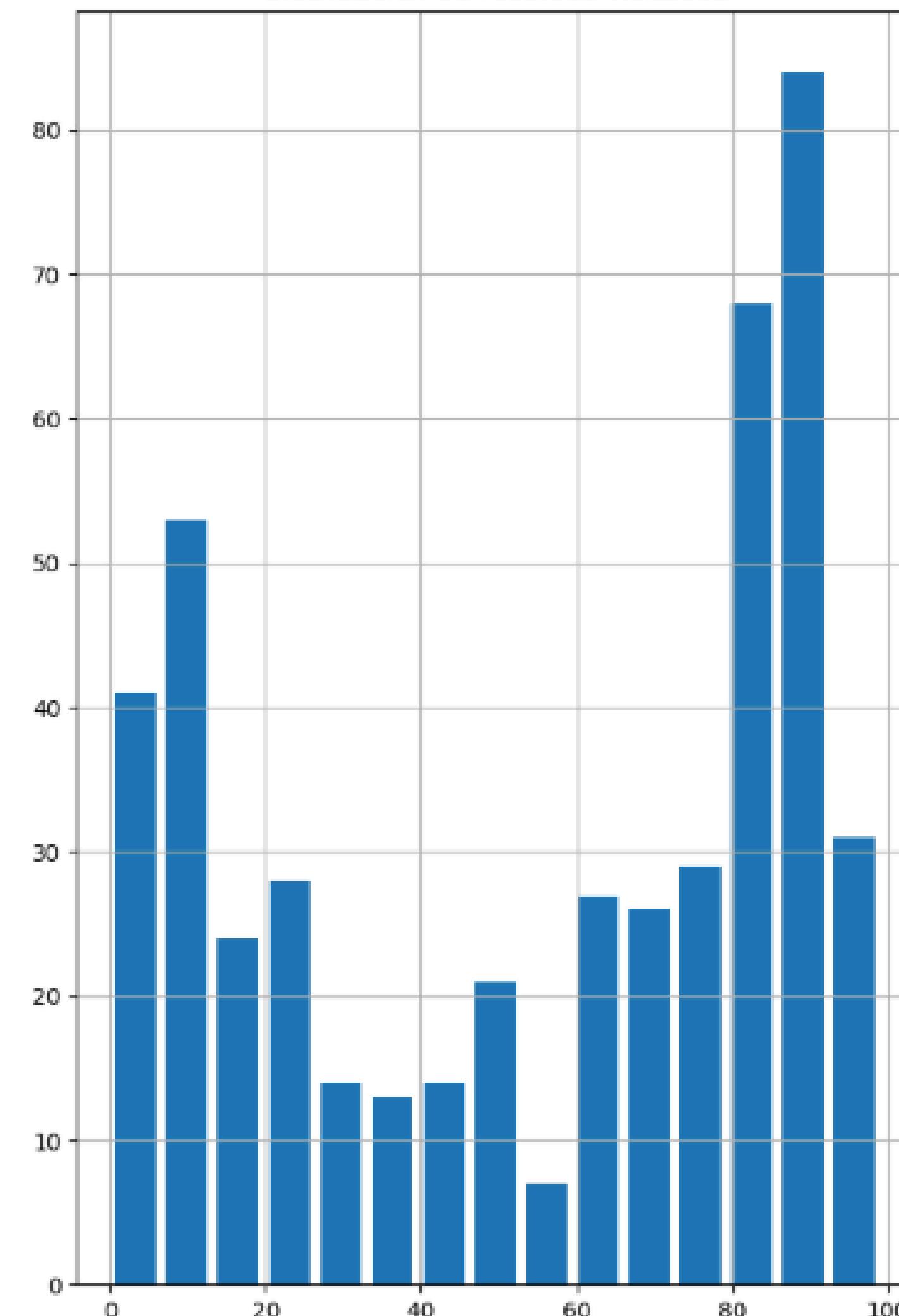
Distribucion por encuesta padre



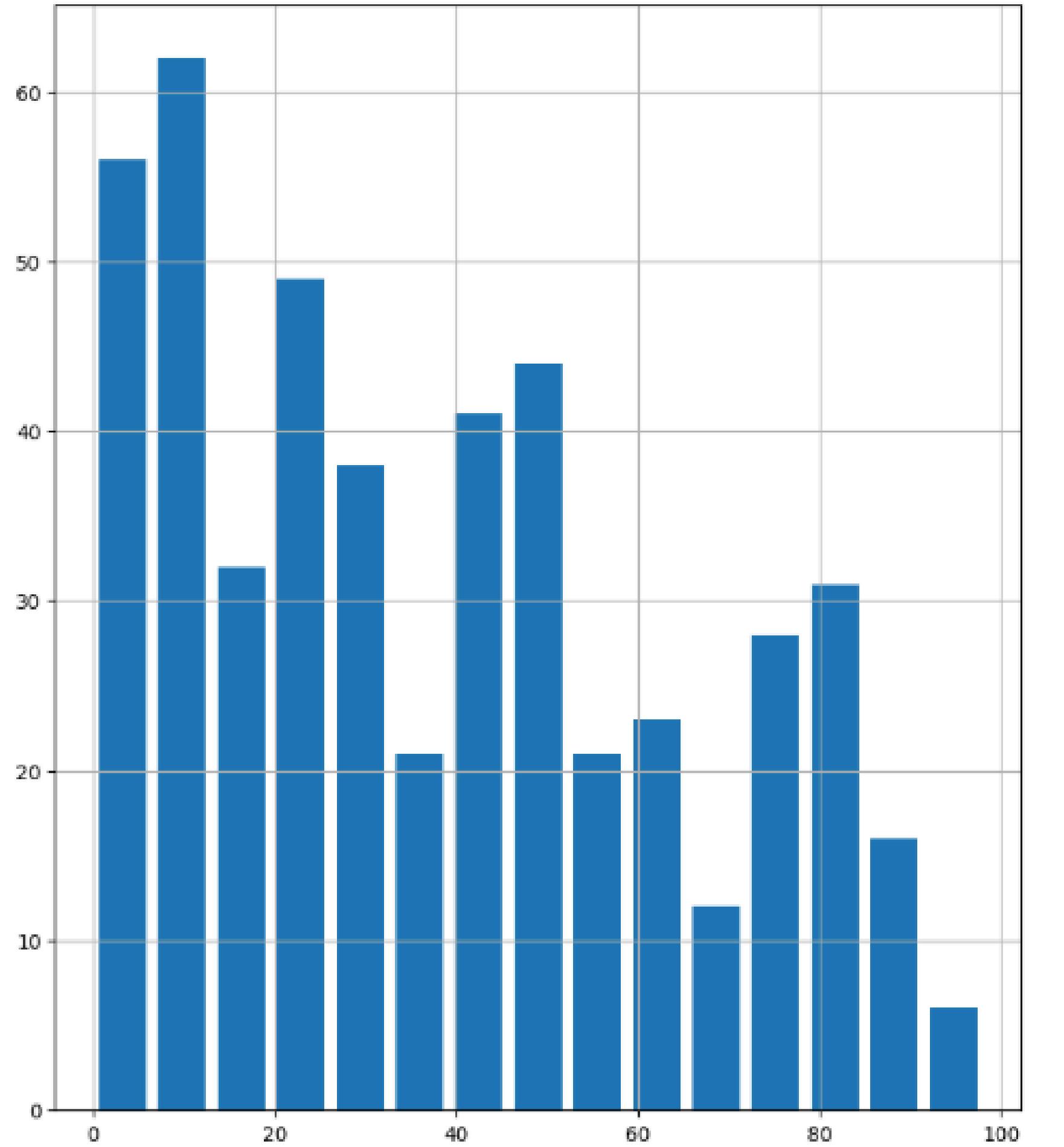
Distribución de mano levantada



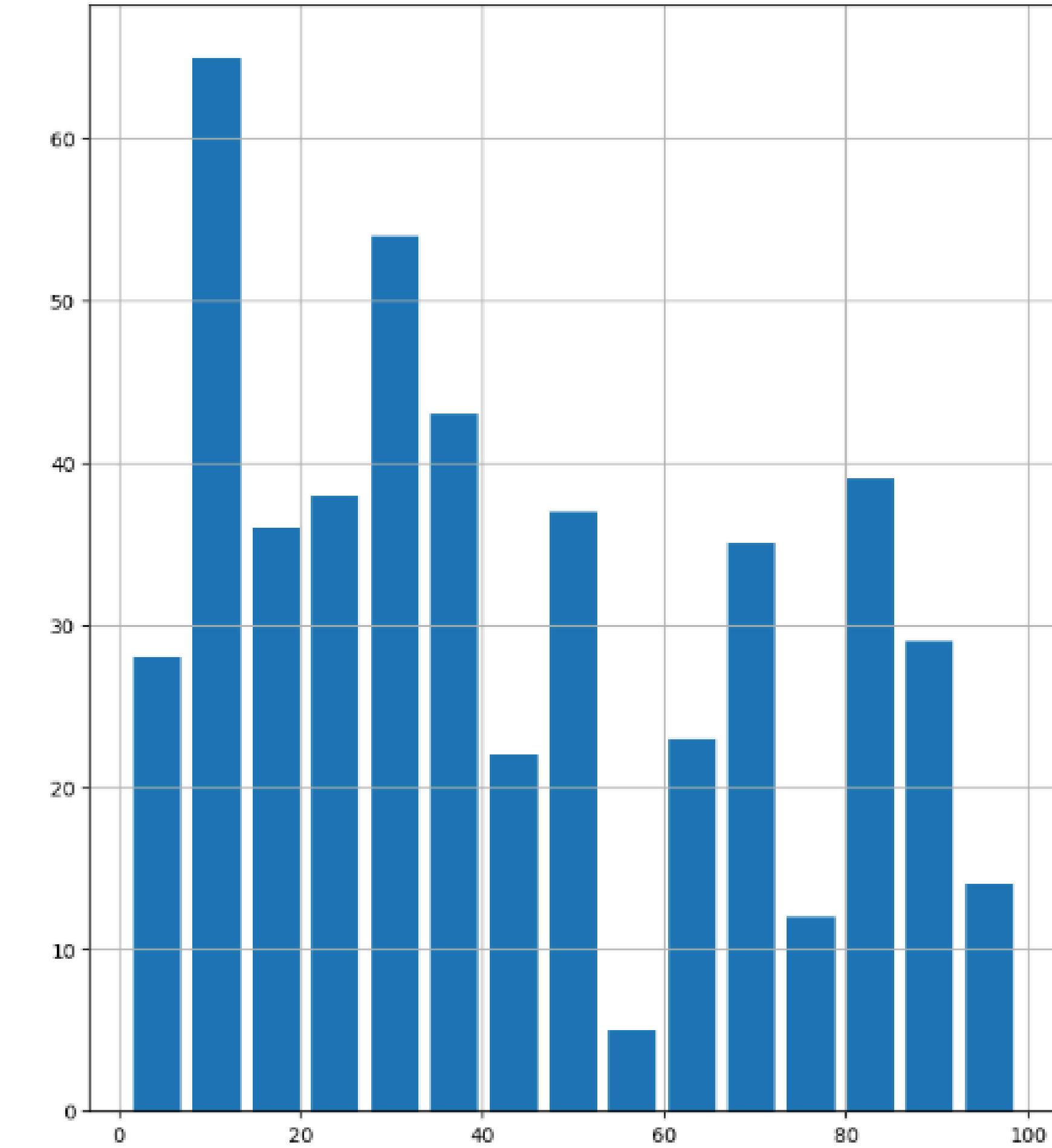
Distribución de recursos visitados



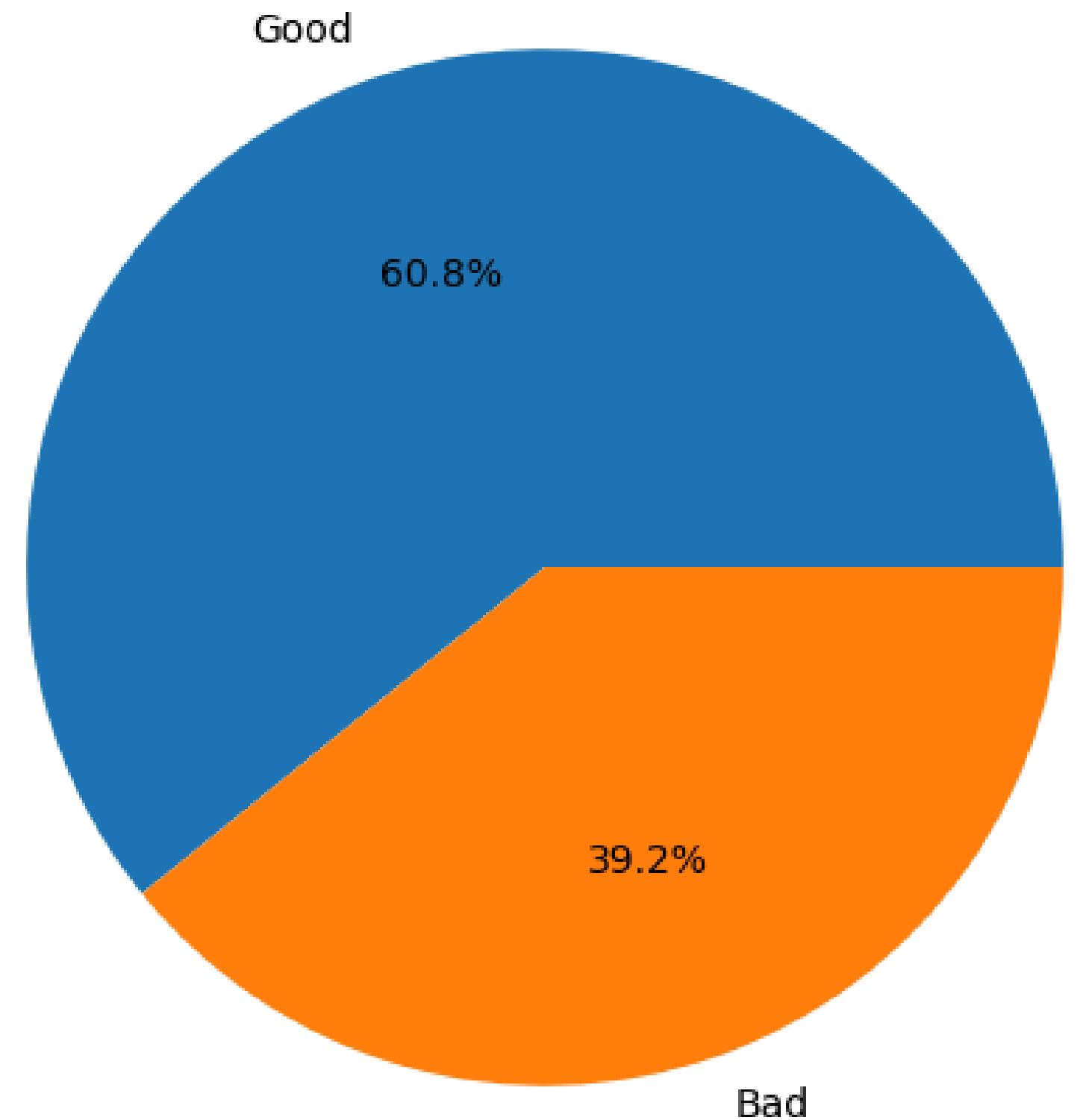
Distribucion de visualizacion anuncios



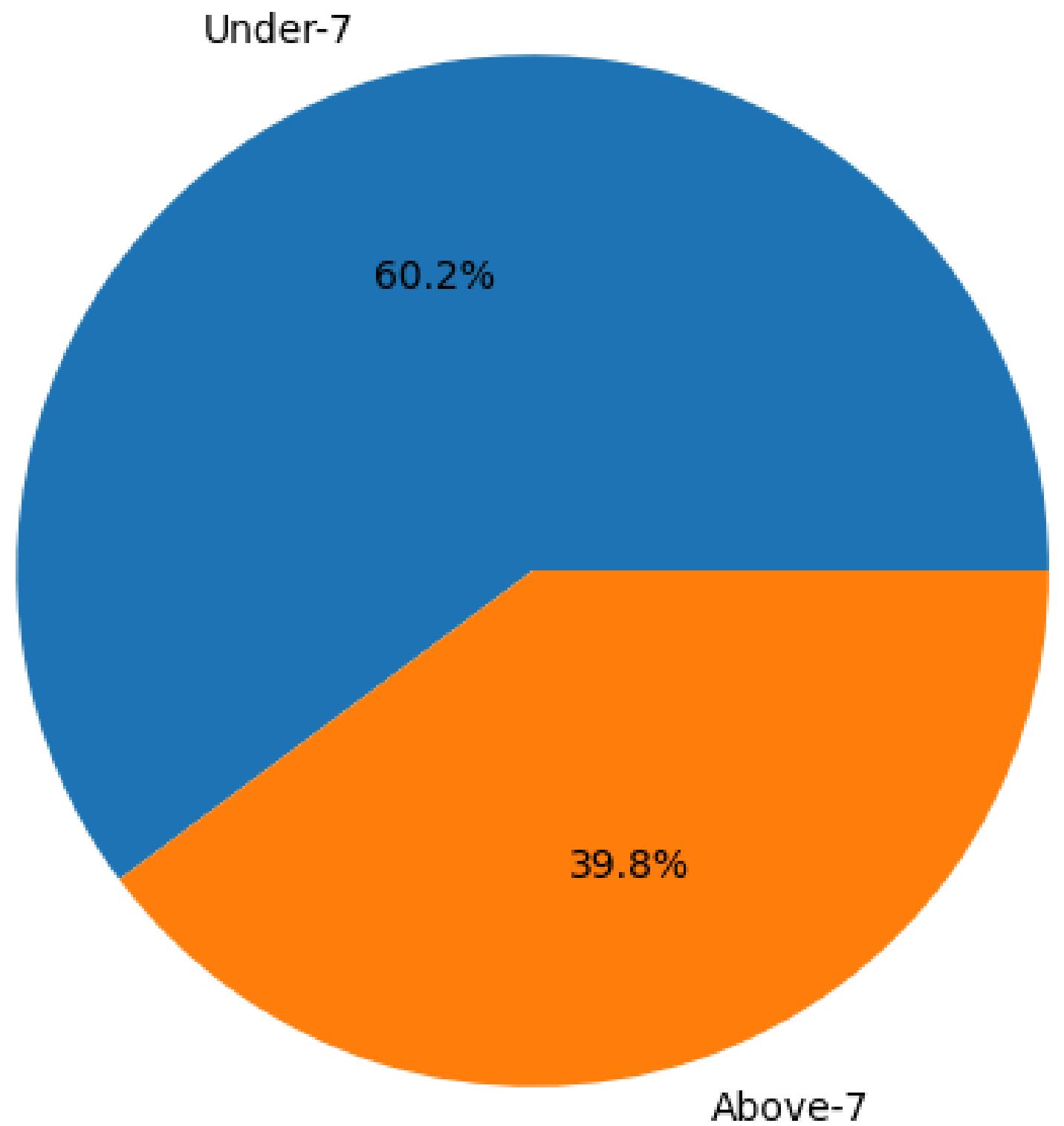
Distribucion de grupos de discusion



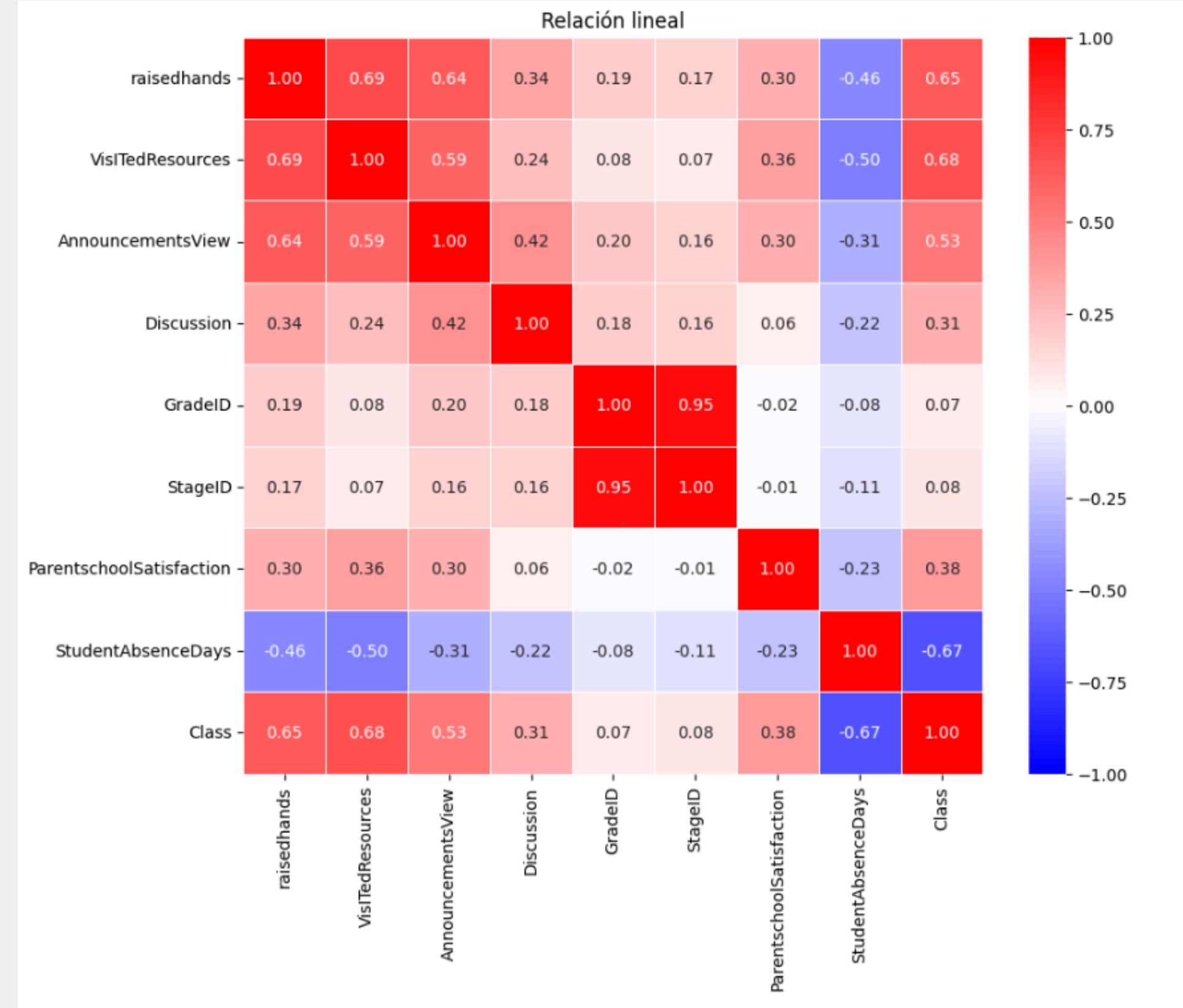
Distribucion por satisfaccion del padre



Distribucion por ausencia estudiante

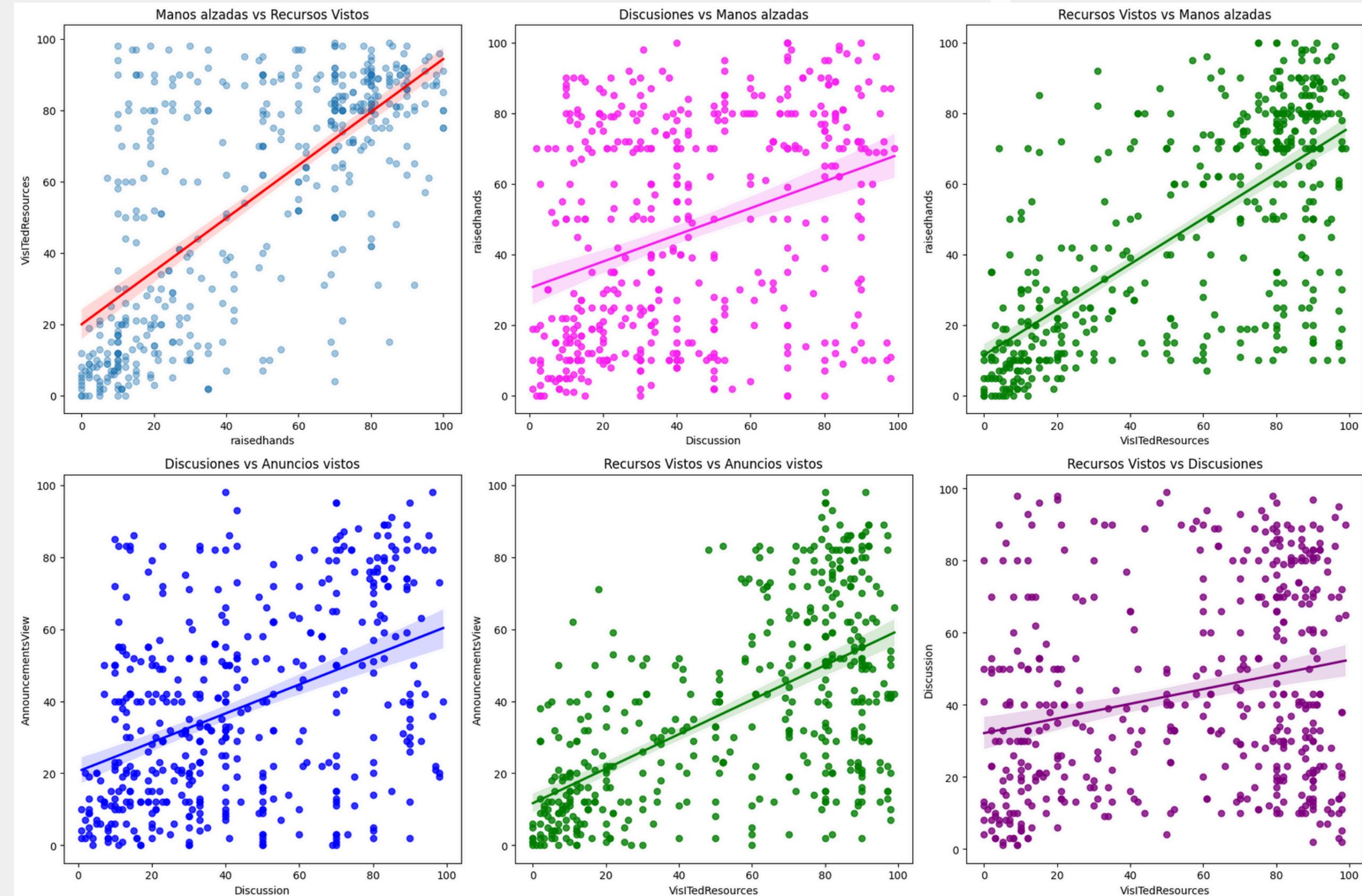


Relación lineal



deWnseH

# Gráficos de Dispersion



# Segunda Entrega



```
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, 'venezuela':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, 'venezuela':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)

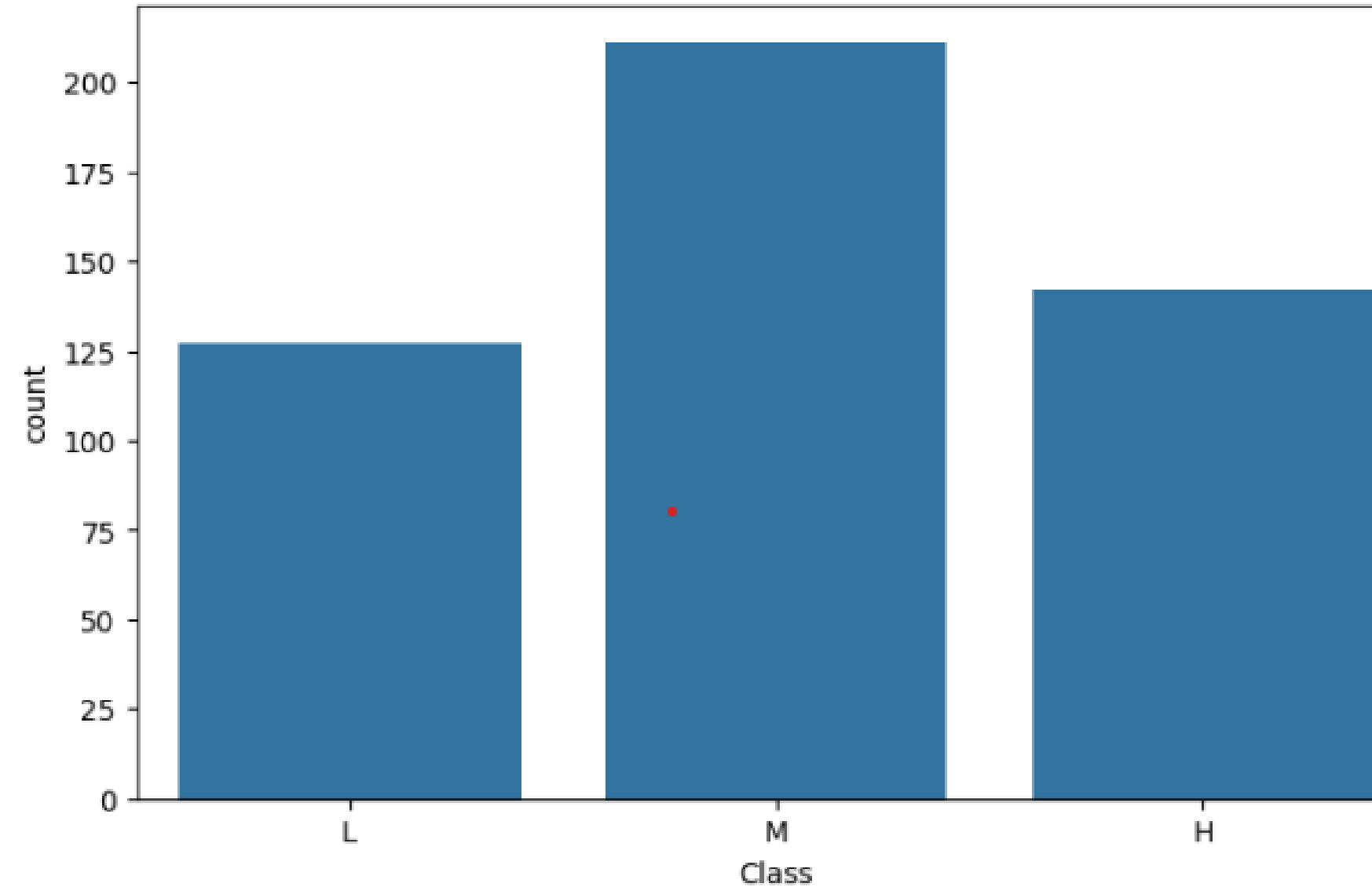
GradeID_mapping = {
    'G-01': 1, 'G-02': 2, 'G-03': 3, 'G-04': 4, 'G-05': 5,
    'G-06': 6, 'G-07': 7, 'G-08': 8, 'G-09': 9, 'G-10': 10,
    'G-11': 11, 'G-12': 12
}
df['GradeID'] = df['GradeID'].map(GradeID_mapping)

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

Distribución de 'Class'



Class  
M 43.958333  
H 29.583333  
L 26.458333

```
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("Accuracy 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))

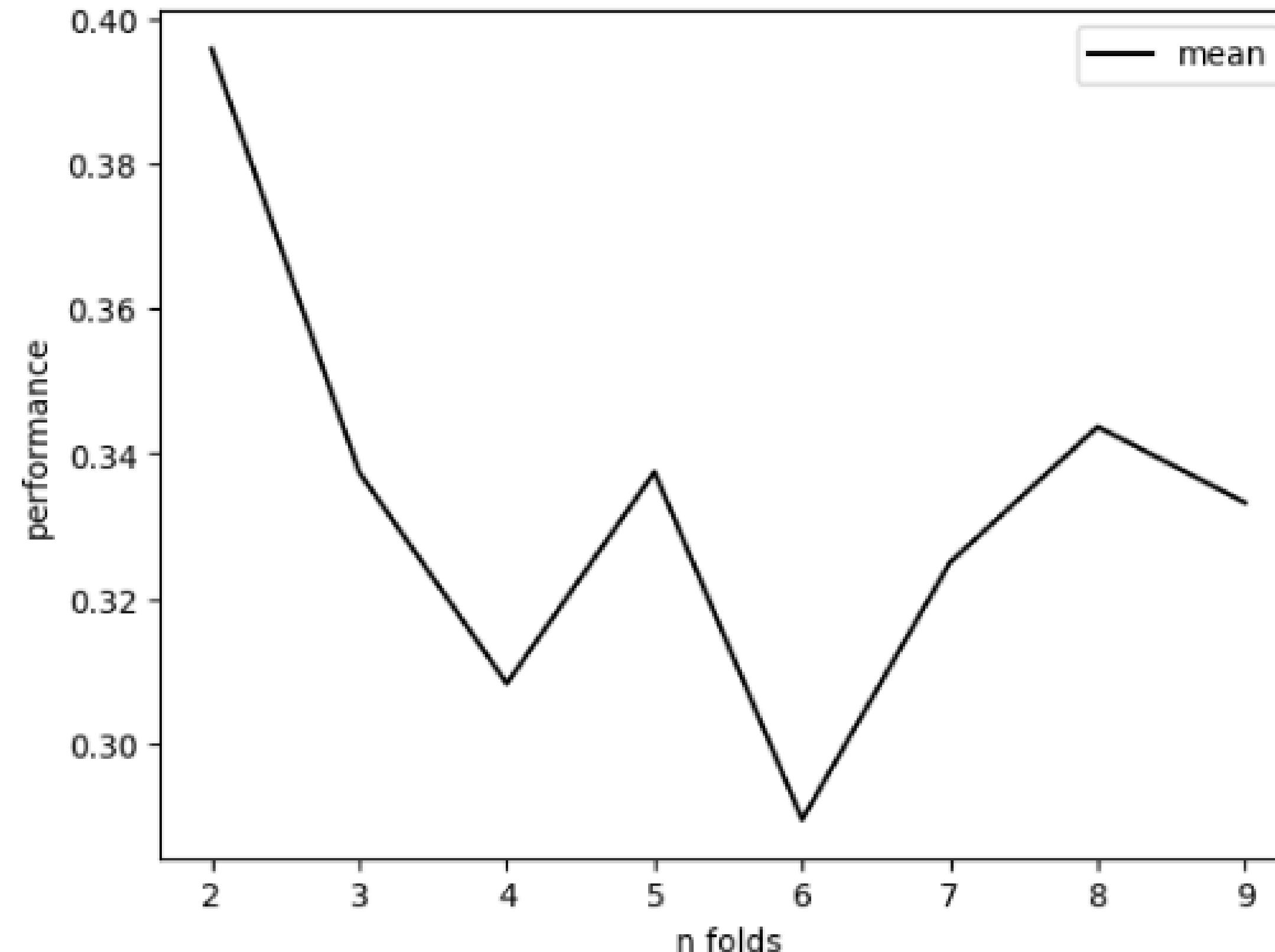
    means = np.r_[means]

    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

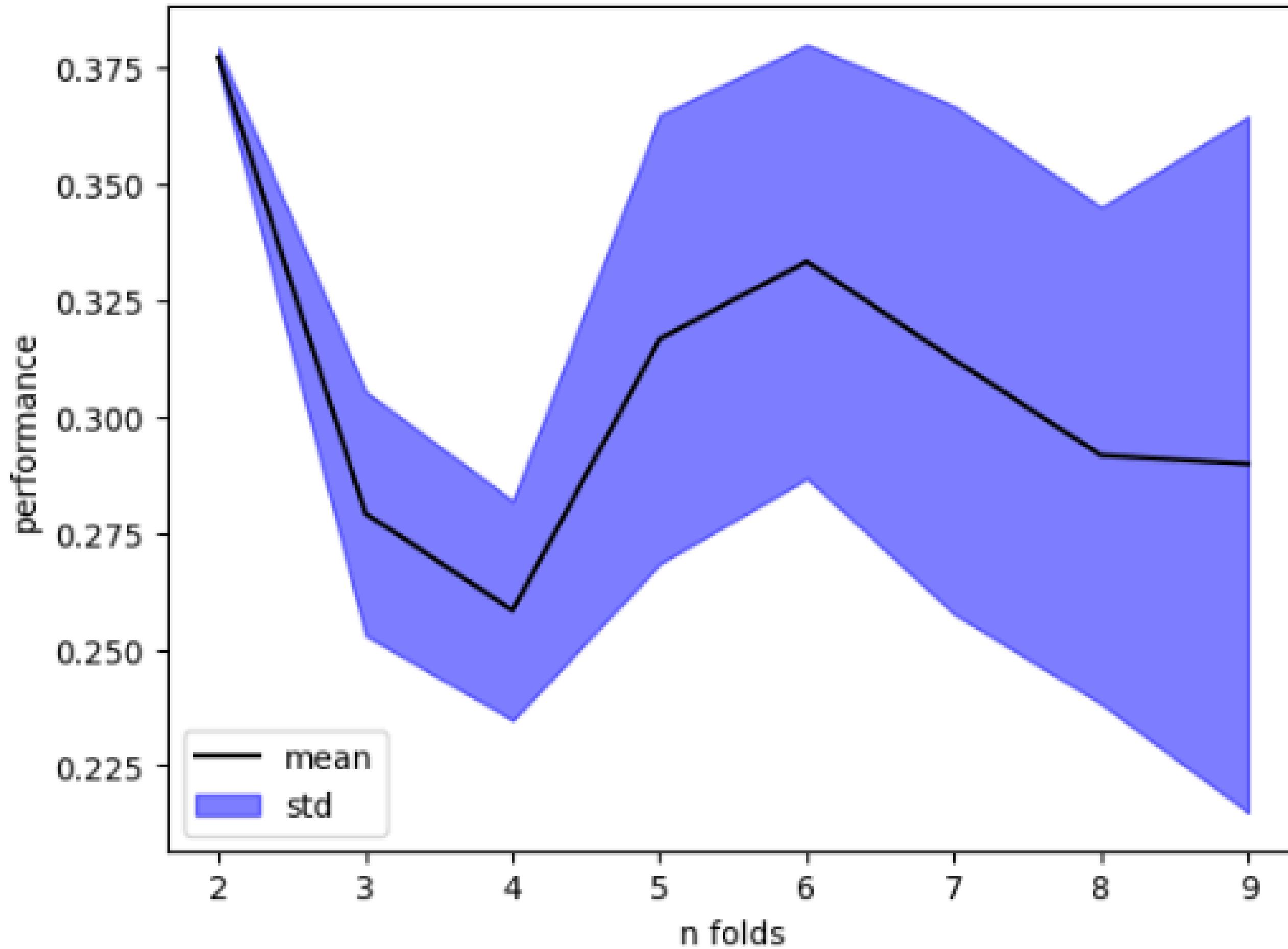
```
def curve_std(est):
    means, stds = [], []
    nfolds_range = range(2,10)
    for nfolds in nfolds_range:
        #print (nfolds,)
        s = cross_val_score(est, X, y, cv=KFold(nfolds, shuffle=True), scoring=make_scorer(mean_squared_error))
        means.append(np.mean(s))
        stds.append(np.std(s))

    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: 0.802083333333334

# Learning Curve

```
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]

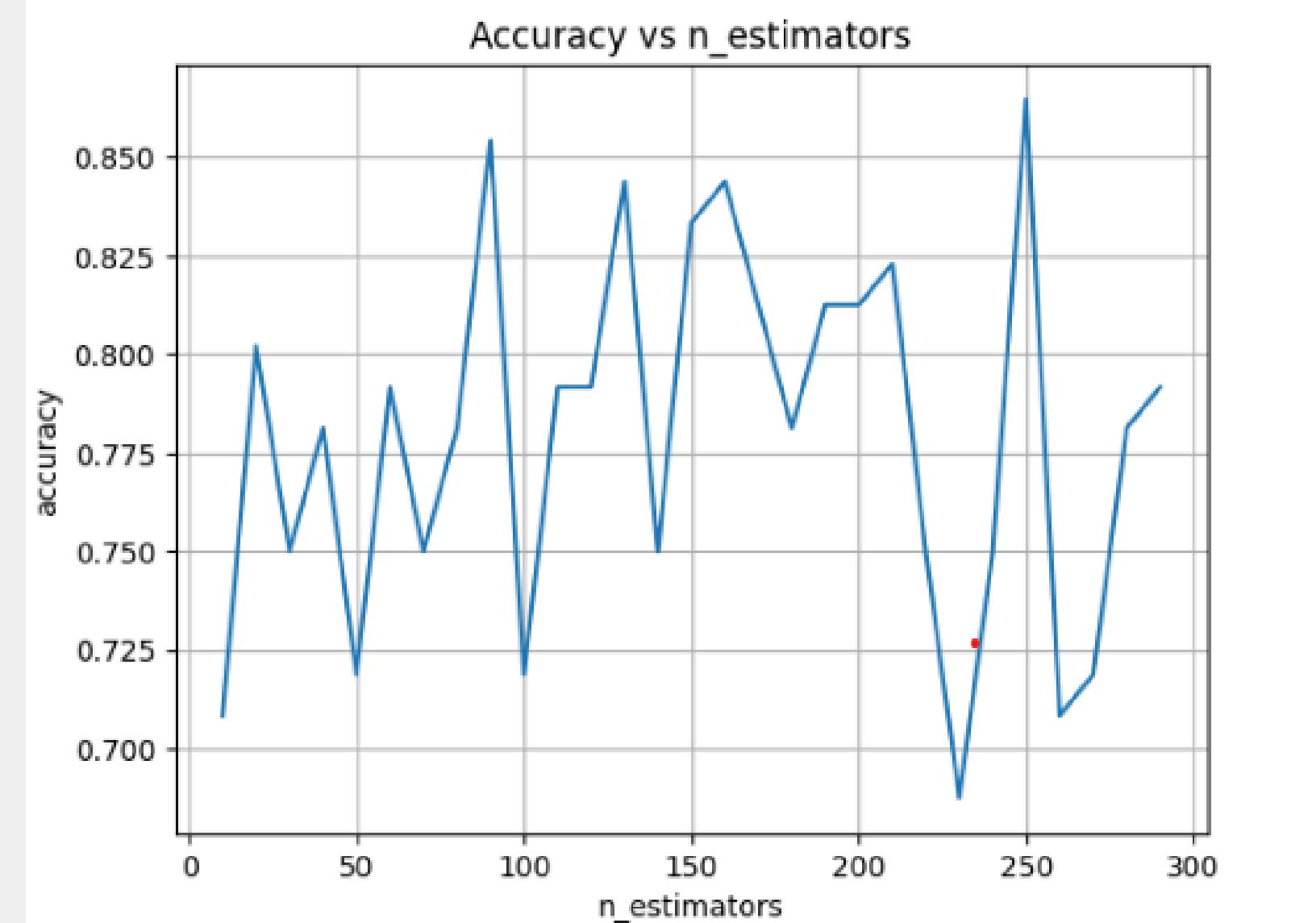
n_vals = list(range(10, 300, 10))
accs = []

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

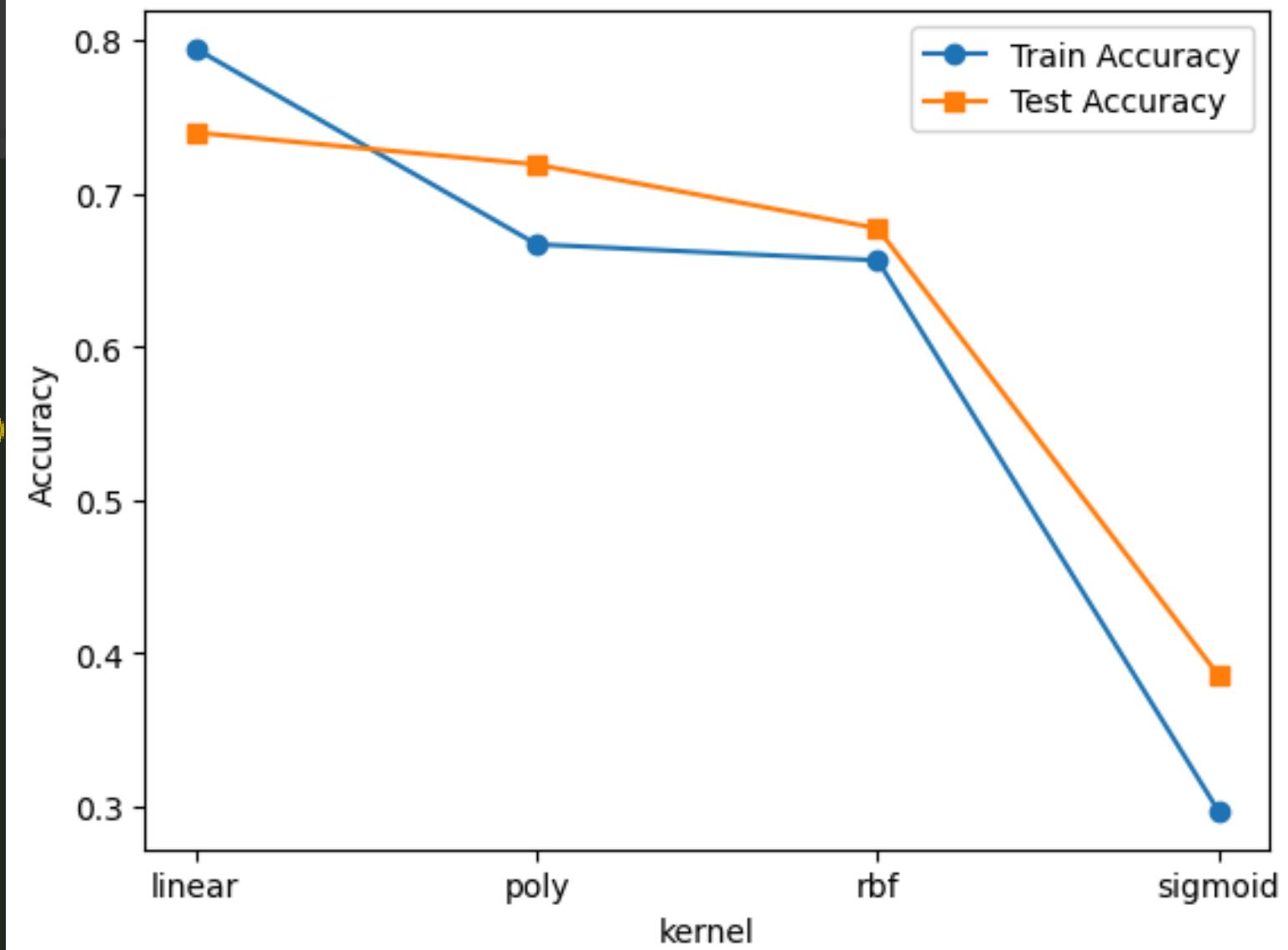
## SVM

```
[14] 1 from sklearn.svm import SVC  
2 from sklearn.model_selection import train_test_split  
3 from sklearn.metrics import accuracy_score  
4  
5 est = SVC()  
6 x = df.values[:, :-1]  
7 y = df.values[:, -1]  
8 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=21)  
9 est.fit(x_train, y_train)  
10 print(accuracy_score(est.predict(x_test), y_test))  
  
0.677833333333334
```

# Learning Curve

## Learning curve

```
[33]: 1 kernel_functions = ['linear', 'poly', 'rbf', 'sigmoid']
2 train_scores = []
3 test_scores = []
4 for kernel in kernel_functions:
5     model = SVC(kernel=kernel)
6     x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=21)
7     model.fit(x_train, y_train)
8     train_accuracy = accuracy_score(y_train, model.predict(x_train))
9     test_accuracy = accuracy_score(y_test, model.predict(x_test))
10    train_scores.append(train_accuracy)
11    test_scores.append(test_accuracy)
12 plt.plot(kernel_functions, train_scores, marker='o', label='Train Accuracy')
13 plt.plot(kernel_functions, test_scores, marker='s', label='Test Accuracy')
14 plt.xlabel("kernel")
15 plt.ylabel("Accuracy")
16 plt.legend()
17 plt.show()
```



**¡Muchas  
gracias!**