Reduciendo inconformidad en los grupos de la carrera de actuaría.

Descripción del problema:

Se ha realizado la encuesta para la elección de materias de los alumnos de la carrera de actuaría y capturado los datos en una hoja de cálculo. Todos los semestres se presentan quejas por la forma en que los grupos son formados así que se requiere hacer una agrupación de tal manera que la mayor cantidad de alumnos pueda estar en las materias de su preferencia. Por cuestiones de personal y económicas, se ha limitado la cantidad de grupos a 5 con 5 materias por grupo, y se busca que la cantidad de alumnos se distribuya de manera equitativa entre estos grupos.

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Import modules for clustering
from sklearn.cluster import KMeans
import scipy.stats as stats
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import StandardScaler

from warnings import filterwarnings
filterwarnings('ignore')
```

In [ ]:
```
# Path for google colab
# from google.colab import drive
# drive.mount('/content/drive')
# Path for local execution
#Import for google colab, installing modules if needed
# !pip install pandas
# !pip install sklearn
# !pip install matplotlib
# !pip install scipy
```

Carga y preparación inicial de los datos

In [ ]:
```
df = pd.read_excel("Grupos_Optativas.xlsx",sheet_name="Base completa de eleccion
df.head()
```

Out[ ]:

	Individuo	Orden de preferencia	Clave	Nombre de la Materia
0	1	5	2030.0	Análisis de datos
1	1	4	2031.0	Análisis Multivariado
2	1	3	2049.0	Evaluación de proyectos
3	1	2	2046.0	Auditoría Actuarial
4	1	1	2047.0	Contabilidad de Seguros

Breve exploración de los datos

Removemos la columna 'nombre de la materia' ya que esta presenta múltiples errores de captura y no aporta información relevante para el análisis, a su vez, se elimina, los registros nulos de la columna 'clave'

In [ ]:
```
df = df.drop(columns=["Nombre de la Materia"])
df = df.dropna(subset=['Clave'])
df["Clave"] = df["Clave"].astype(int)

#Modificar el valor de aquellos elementos que tengan la clave 2511 por 2037
df.loc[df['Clave'] == 2511, 'Clave'] = 2037

df=df.sort_values(by=["Individuo",'Orden de preferencia'])
df.reset_index(drop=True, inplace=True)
```

Formamos un nuevo dataset que tenga la información de los alumnos como vector y al alumno como su indice

In [ ]:
```
new_cols= []
new_cols.append("Individuo")
new_cols.extend(sorted([str(clave) for clave in df["Clave"].unique()]))
clustering_df = pd.DataFrame(columns=new_cols)
```

In [ ]:
```
keys = clustering_df.columns

for group in df.groupby("Individuo"):

  storage_dict = {columna: 0 for columna in keys}
  storage_dict["Individuo"] = group[0]

  for row in group[1].iterrows():
    storage_dict[str(row[1]["Clave"])] = row[1]["Orden de preferencia"]

  clustering_df = clustering_df.append(storage_dict, ignore_index=True)
```

Formamos un dataframe extra para hacer estadística descriptiva sobre los datos

In [ ]:
```
df['Frecuencia'] = df.groupby('Clave')['Clave'].transform('count')
df['Interes'] = df.groupby('Clave')['Orden de preferencia'].transform('sum')
#Adding the mean and the sum of the preference order for each subject
temp_df = df.groupby('Clave').agg({'Orden de preferencia':['mean','sum'], 'Frecu
temp_df.columns = ['_'.join(col) for col in temp_df.columns.values]
```

```
del df
```

In [ ]:
```
#Correlation analysis between the interest and the frequency with the pearson me
pearson_coef, p_value = stats.pearsonr(temp_df["Orden de preferencia_sum"], temp
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value o
```

```
The Pearson Correlation Coefficient is 0.9906466226833289  with a P-value of P =
5.053357925838739e-24
```

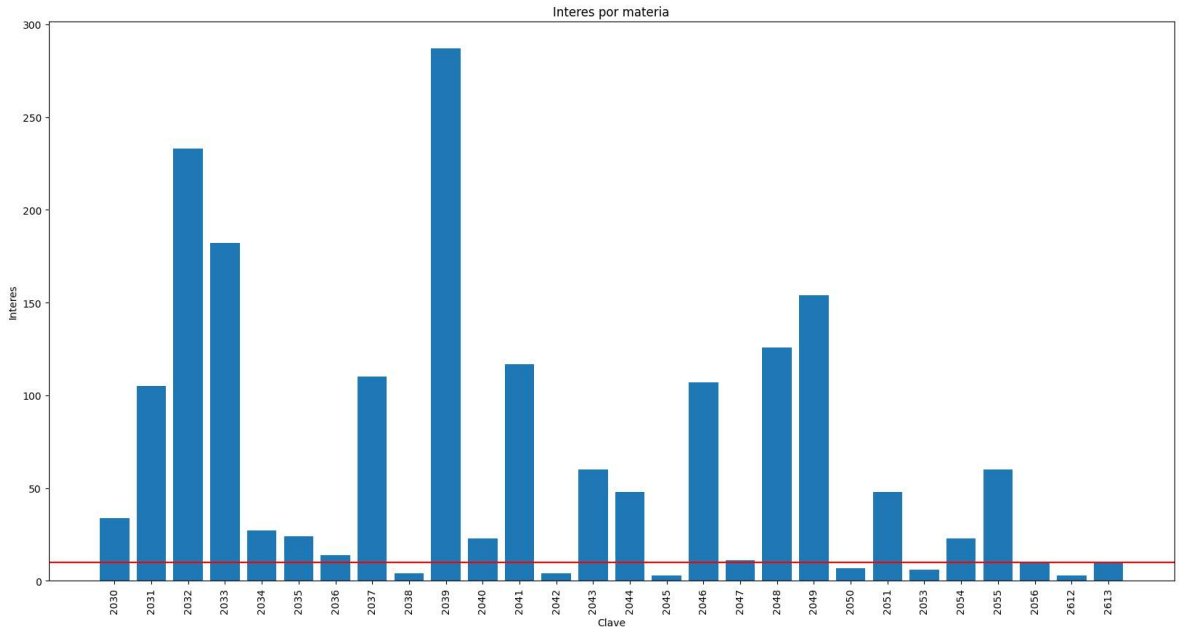
Al hacer las medidas para la suma del interes y la frecuencia se vió que tenían un comportamiento similar, con esto se observó que ambas variables están correlacionadas, por lo que se puede usar cualquiera de las dos y se eligió la suma del interes por materia.

In [ ]:
```
# Show the statistics of the frequency values
data = temp_df["Orden de preferencia_sum"]
summary = pd.Series({
    'count': len(data),
    'mean': np.mean(data),
    'std': np.std(data),
    'min': np.min(data),
    '25%': np.percentile(data, 25),
    '50%': np.percentile(data, 50),
    '75%': np.percentile(data, 75),
    'max': np.max(data),
})
print(summary)
condition = data >= np.percentile(data, 25)
count = np.count_nonzero(condition)
print(f'The number of courses with frequency greater than the 25% is: {count}')
#Get the list of the subjects with frequency greater than the 25%
```

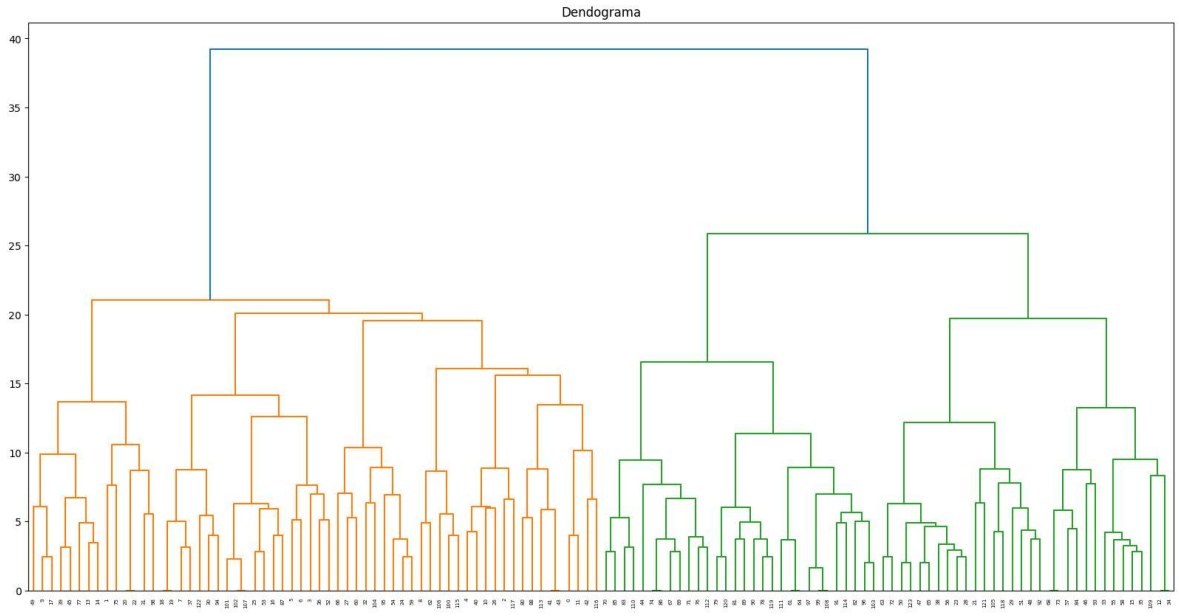
```
count     28.000000
mean      65.714286
std       73.848715
min        3.000000
25%       10.000000
50%       30.500000
75%      107.750000
max      287.000000
dtype: float64
The number of courses with frequency greater than the 25% is: 22
```

Generamos un histograma del interés en las materias y establecemos una línea de aquellos que están por debajo del percentel del 25 por ciento para ignorar estos posteriormente

In [ ]:
```
#Plot a histogram with the x axis the key and the y axis the interest
plt.figure(figsize=(20,10))
plt.bar(temp_df.index.astype(str), temp_df["Orden de preferencia_sum"])
plt.xticks(rotation=90)
plt.title("Interes por materia")
plt.xlabel("Clave")
plt.ylabel("Interes")
#Add a line at the y axis at value 10
plt.axhline(y=np.percentile(data, 25), color='r', linestyle='-')
plt.show()
```



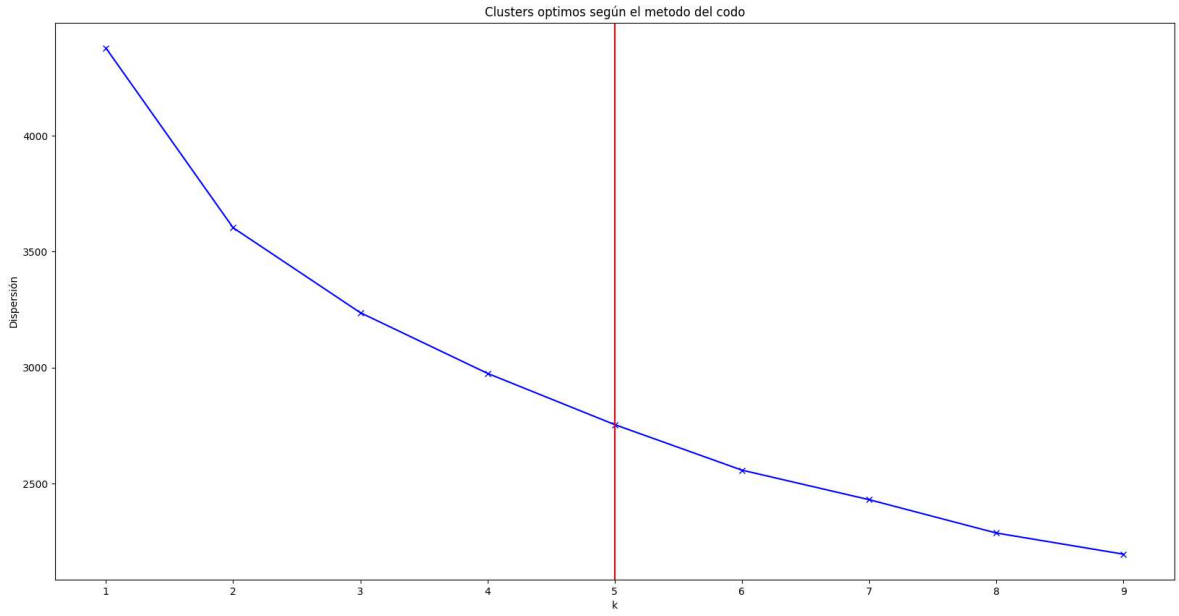
In [ ]:
```
#Create a dendogram with the clustering_df dataframe
plt.figure(figsize=(20,10))
plt.title("Dendograma")
dendrogram(linkage(clustering_df.iloc[:,1:], method='ward'))
plt.show()
```



In [ ]:
```
# Get the list of the subjects with frequency greater than the 25%
list_of_subjects = temp_df.index[condition].tolist()
list_of_subjects = [str(subject) for subject in list_of_subjects]
#Remove the element 2511 from the list_of_subjects
#Remove the columns that don't match the condition of interest
clustering_df = clustering_df[list_of_subjects]
```

In [ ]:
```
#Applying the elbow method to find the optimal number of clusters
to_cluster = clustering_df.iloc[:, 1:]
dispersiones = []
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(to_cluster)
```

```
    dispersiones.append(kmeanModel.inertia_)
plt.figure(figsize=(20,10))
plt.plot(K, dispersiones, 'bx-')
plt.xlabel('k')
plt.ylabel('Dispersión')
plt.title('Clusters optimos según el metodo del codo')
#Add a line at the x axis at value 5
plt.axvline(x=5, color='r', linestyle='-')
plt.show()
```



In [ ]:
```
clusters = 5
kmeans = KMeans(n_clusters=clusters, random_state=0)
clustering_df['Cluster'] = kmeans.fit_predict(to_cluster)
#Print the number of elements in each cluster
print(clustering_df['Cluster'].value_counts())
```

```
4    27
0    26
2    26
3    25
1    20
Name: Cluster, dtype: int64
```

In [ ]:
```
#Generate a dictionary with the subject key and their centroid for each cluster
clusters_dict = {}
for cluster in range(clusters):
  clusters_dict[cluster] = {}
  for key in clustering_df.columns[1:-1]:
    clusters_dict[cluster][key] = clustering_df[clustering_df["Cluster"] == clus
```

In [ ]:
```
#Get the 5 subjects with the highest interest for each cluster
top_5_interest = {}
for cluster in clusters_dict:
  top_5_interest[cluster] = sorted(clusters_dict[cluster].items(), key=lambda x:
```

In [ ]:
```
col_names = ['Materia', "Clave"]
keys_df = pd.read_excel("Grupos_Optativas.xlsx", sheet_name="materias", usecols=
```

In [ ]:
```
#Generate a dictionary with the cluster and the top 5 subjects with the highest
grupos = {}
```

```
for grupo in clustering_df["Cluster"].unique():
  grupos[grupo] = []
  for materia in top_5_interest[grupo]:
    grupos[grupo].append(keys_df[keys_df["Clave"] == int(materia[0])]["Materia"]
```

In [ ]:
```
#Print the groups ordered by the cluster
for grupo in sorted(grupos):
  print(f"Grupo {grupo}: {grupos[grupo]}")
```

```
Grupo 0: ['Derivados', 'Evaluación de Proyectos', 'Auditoria Actuarial', 'Análisi
s Multivariado', 'Modelos y simulación']
Grupo 1: ['Investigación de Operaciones II', 'Análisis de Regresión', 'Modelos y
simulación', 'Muestreo', 'Evaluación de Proyectos']
Grupo 2: ['Estadística Bayesiana', 'Muestreo', 'Modelos y simulación', 'Análisis
de Regresión', 'Series de Tiempo']
Grupo 3: ['Auditoria Actuarial', 'Modelos y simulación', 'Planeación financiera',
'Evaluación de Proyectos', 'Análisis de estados financieros']
Grupo 4: ['Análisis de Regresión', 'Modelos y simulación', 'Análisis Multivariad
o', 'Derivados', 'Series de Tiempo']
```

In [ ]: