UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

Universidad del Perú, Decana de América

Facultad de Ingeniería de Sistemas

EXAMEN FINAL

DOCENTE:

ROMÁN CONCHA , NORBERTO ULISES

GRUPO 10

INTEGRANTES:

ARRIETA PALACIOS, ANDRÉS LEONARDO

MORENO ALCA, ALEJANDRO

SALAS RAMIREZ, MARCELO MITCHELL

LA ROSA HUAPAYA, MIGUEL ESTEFANO

ROMERO RUIZ, JOSÉ DANIEL

Introducción

La enseñanza del uso del software Python para el uso de las REPITENCIAS, TUTORÍAS Y RENDIMIENTO EN LAS CALIFICACIONES viene siendo de vital importancia en el trabajo presentado, ya que, los datos trabajados tienen que presentarse bajo el esquema estadístico y Python logra ese objetivo dando como resultado un rendimiento óptimo al tema propuesto.

En esté tema vamos a poder visualizar la importación de librerías, la carga de data y la visualización de los datos de acuerdo a los puntos que queremos evaluar en prueba.

Fundamentación teórica

¿Qué es Python?

Python es un lenguaje de programación de alto nivel y de propósito general que se utiliza ampliamente en el desarrollo de aplicaciones y en la ciencia de datos.

En el contexto de Big Data, Python es una herramienta valiosa porque tiene una gran cantidad de librerías y herramientas que facilitan el manejo y el procesamiento de grandes conjuntos de datos. Algunas de las funcionalidades más importantes de Python en el ámbito del Big Data son:

- Manejo de datos: Python tiene una gran cantidad de librerías para leer, escribir y manipular datos en diferentes formatos, como CSV, JSON, Excel, etc.

- Análisis de datos: Python tiene librerías como NumPy y Pandas que proporcionan herramientas potentes para el análisis y la manipulación de datos. Estas librerías también tienen una gran integración con otras librerías de visualización de datos como Matplotlib, lo que facilita la exploración y el análisis de grandes conjuntos de datos.

- Integración con bases de datos: Python tiene módulos para conectarse a una amplia gama de bases de datos, lo que facilita el acceso y la manipulación de datos almacenados en una base de datos.

- Aprendizaje automático: Python tiene librerías como scikit-learn y TensorFlow que proporcionan una amplia gama de algoritmos de aprendizaje automático y herramientas para el entrenamiento y el uso de modelos de aprendizaje automático en grandes conjuntos de datos.

En resumen, Python es una herramienta valiosa en el contexto del Big Data debido a su gran cantidad de librerías y herramientas para el manejo, el análisis y el procesamiento de grandes conjuntos de datos.

Data de prueba repitencias 2017-1 al 2019-2

1. Importamos las librerías y hacemos la carga de la data de repitencias

In [26]:
```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt

archivo='DATA-PRUEBA_REPITENCIAS 2017-2020-UNMSM.xlsx'
datos=pd.read_excel('C:\\Users\\acer\\Pictures\\6to CICLO\\BIG DATA\\EXAMEN\\CASO-UNMSM\\'+archivo,sheet_name=None)
datoglobal=pd.concat(datos,ignore_index=True)
print(datosglobal)
```

```
        cod_facultad  cod_escuela cod_semestre cod_alumno cod_plan  \
0                1.0          1.0        20171   12010240   2015
1                1.0          1.0        20171   12010240   2015
2                1.0          1.0        20171   12010240   2015
3                1.0          1.0        20171   12010240   2015
4                1.0          1.0        20171   05010012   2015
...              ...          ...          ...        ...      ...
221674          20.0          2.0        20202   19200315   2018
221675          20.0          2.0        20202   19200315   2018
221676          20.0          2.0        20202   19200320   2018
221677          20.0          2.0        20202   19200324   2018
221678          20.0          2.0        20202   19200327   2018

       cod_asignatura                                asignatura  num_rep
0             M15011M      ATENCIÓN DE LA SALUD NIVEL I y II  (I)      1.0
1             M15026M   FARMACOLOGÍA BÁSICA APLICADA A LA MEDICINA      1.0
2             M15028M                              FISIOPATOLOGÍA      1.0
3             M15034M     ATENCIÓN DE LA SALUD NIVEL I y II  (II)      1.0
4             M15022M                         FISIOLOGÍA HUMANA II      4.0
...               ...                                          ...      ...
221674       202W0301                              ALGORÍTMICA I      1.0
221675       202W0307              ORGANIZACIÓN Y ADMINISTRACIÓN      1.0
221676       202W0301                              ALGORÍTMICA I      1.0
221677       202W0301                              ALGORÍTMICA I      1.0
221678       202W0301                              ALGORÍTMICA I      1.0

[221679 rows x 8 columns]
```

2. Cantidad de repitentes por facultad en el presente semestre

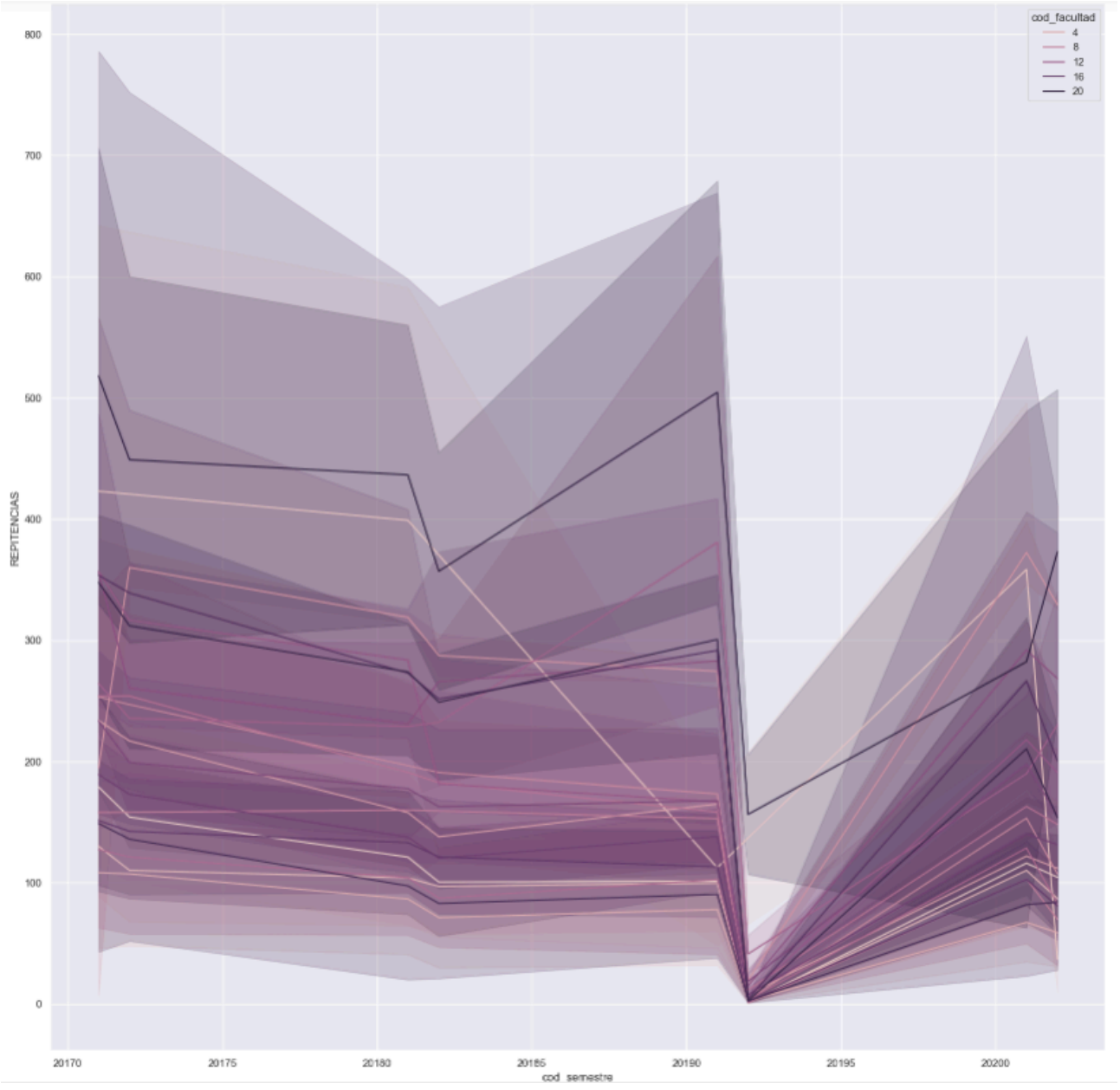
In [27]:
```
#Repitencias por facultad , semestre y escuela

repitencias=datoglobal.groupby(['cod_facultad','cod_escuela','cod_semestre'])['cod_alumno'].nunique().reset_index(name='REP
print(repitencias)
```

```
     cod_facultad  cod_escuela cod_semestre  REPITENCIAS
0             1.0          1.0        20171          211
1             1.0          1.0        20172          141
2             1.0          1.0        20181          156
3             1.0          1.0        20182          110
4             1.0          1.0        20191          118
..            ...          ...          ...          ...
498          20.0          2.0        20182          259
499          20.0          2.0        20191          330
500          20.0          2.0        20192          107
501          20.0          2.0        20201          295
502          20.0          2.0        20202          239

[503 rows x 4 columns]
```

3. Gráfica 1

Gráfica de línea donde el eje X se tiene los semestres (unidos de todas las hojas) y eje Y se tiene el total repitentes con una leyenda de facultades.

```
In [29]: sb.set(rc={'figure.figsize':(20,20)})
         grafica1=sb.lineplot(x='cod_semestre',y='REPITENCIAS',data=repitencias,hue='cod_facultad')
```

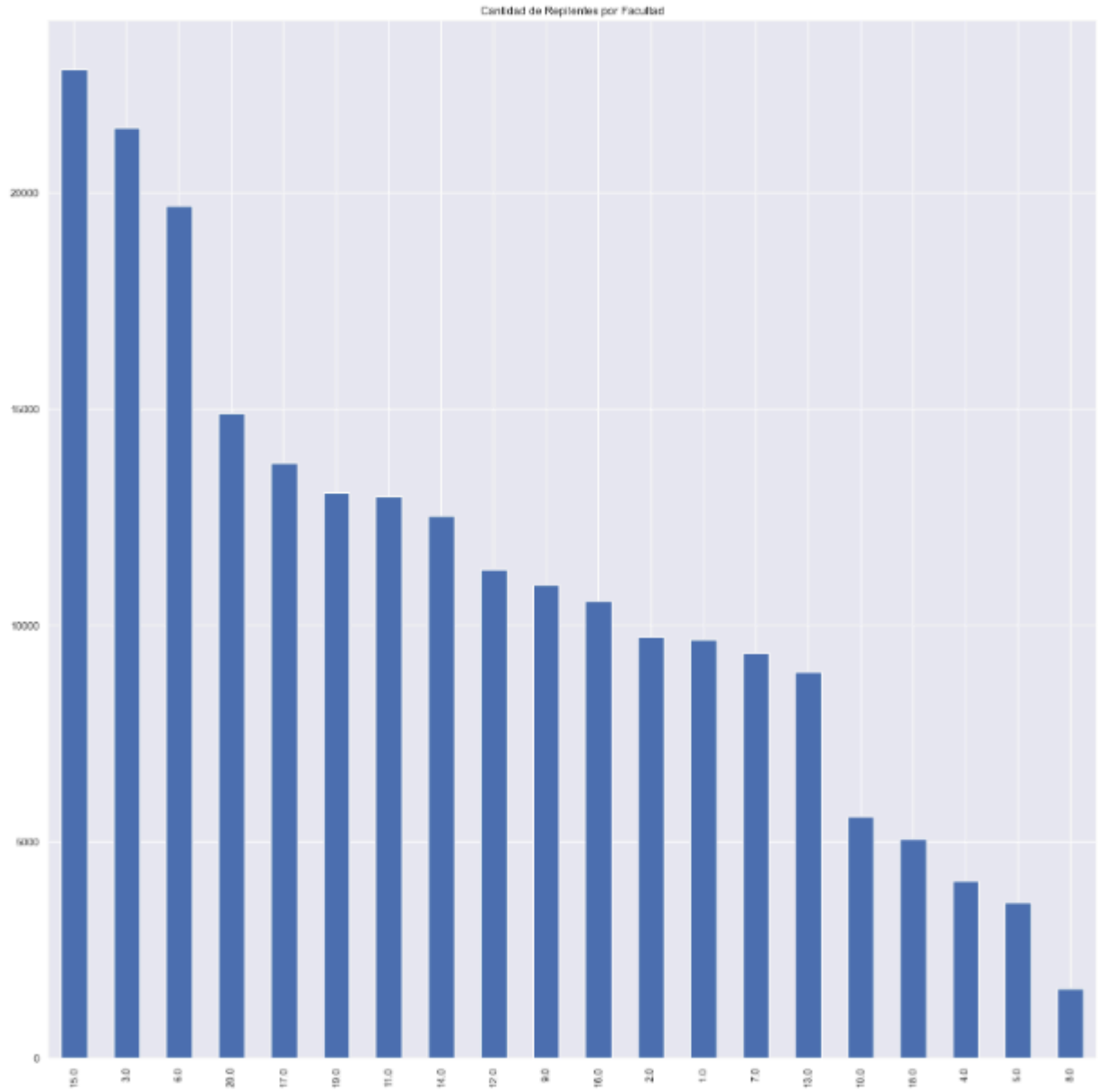


4. **Gráfica 2**

Facultad con cantidad de repitentes

```
In [31]: datoglobal['cod_facultad'].value_counts().plot(kind='bar',title='Cantidad de Repitentes por Facultad')
```

```
Out[31]: <AxesSubplot:title={'center':'Cantidad de Repitentes por Facultad'}>
```



La Facultad con la mayor cantidad de repitentes es la de código 15 y la Facultad con la menor cantidad de repitentes es la de 8.

5. **Gráfica 3**

Escuela por semestre cantidad de repitentes

In [75]:
```
g = sb.FacetGrid(repitencias, col="cod_escuela", height=4, col_wrap=5, ylim=(0, 500))
g.map(sb.pointplot, "cod_semestre", "REPITENCIAS", order=['20171','20172','20181','20182','20191','20192','20201','20202'])
```

Out[75]: <seaborn.axisgrid.FacetGrid at 0x2ad405b5f70>

6. DETERMINAR CANTIDAD DE REPITENCIAS (1 REP, 2DA REP, 3 REP, 4 ,REP, 5 REP.) POR ESCUELAS Y POR PLANES DE ESTUDIOS

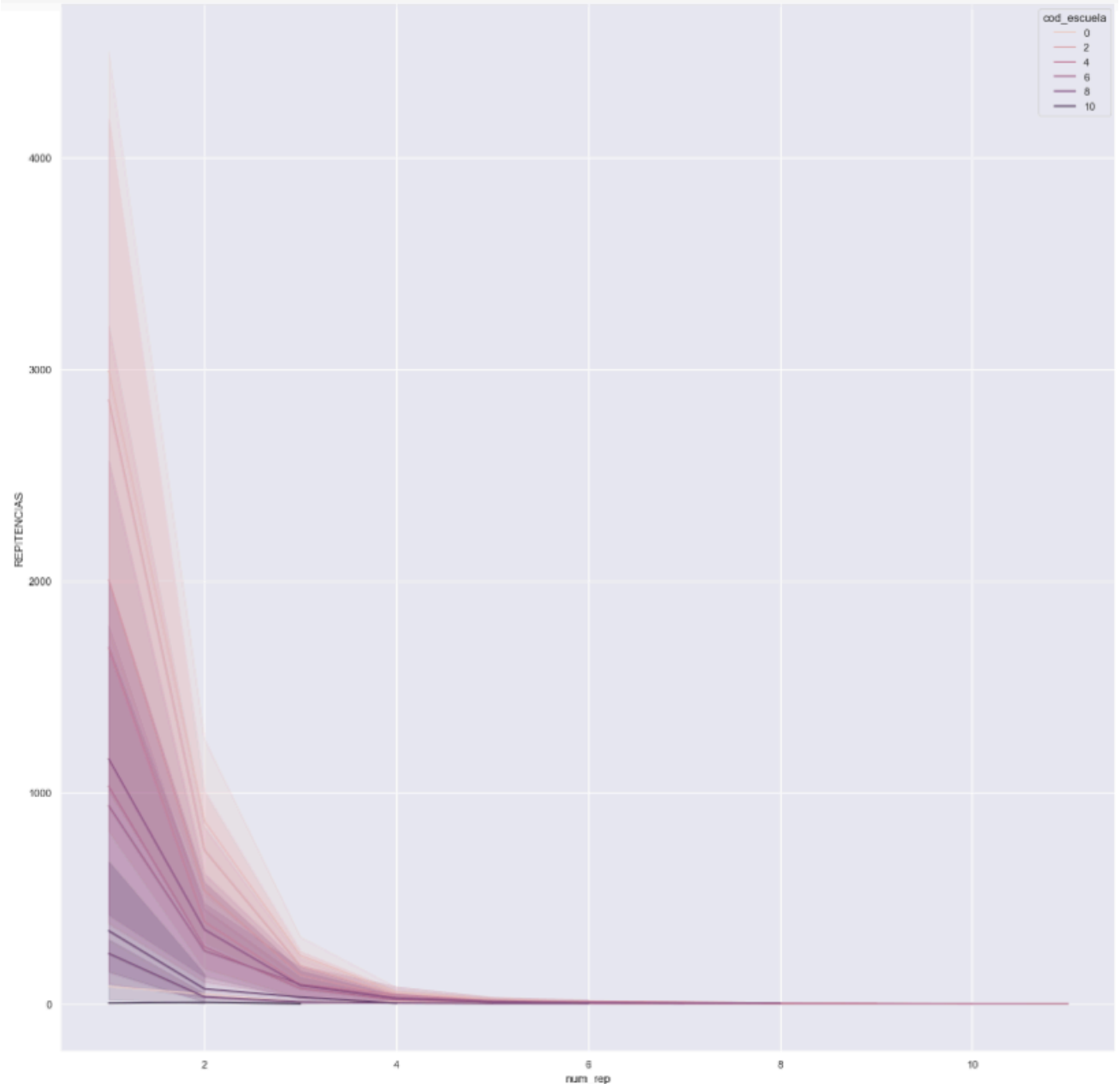
Dataframe de número de repitencias por escuela y planes de estudio

In [77]:
```
# nro de 1,2,3,4,5 de repitencias por escuela y planes de estudio

rep_esc_estudio=datoglobal.groupby(['cod_escuela','cod_plan','num_rep'])['num_rep'].count().reset_index(name='REPITENCIAS')
print(rep_esc_estudio)
```

```
     cod_escuela cod_plan  num_rep  REPITENCIAS
0            0.0   2009        1.0           82
1            0.0   2009        2.0           46
2            0.0   2009        3.0            9
3            0.0   2009        4.0           10
4            0.0   2009        5.0            4
..           ...      ...      ...          ...
448          9.0   2018        1.0           20
449          9.0   2018        2.0            2
450         11.0   2010        1.0            4
451         11.0   2010        2.0            7
452         11.0   2010        3.0            1

[453 rows x 4 columns]
```

7. **Gráfico 4:** CANTIDAD DE REPITENCIAS (1 REP, 2DA REP, 3 REP, 4 REP, 5 REP.) POR ESCUELAS

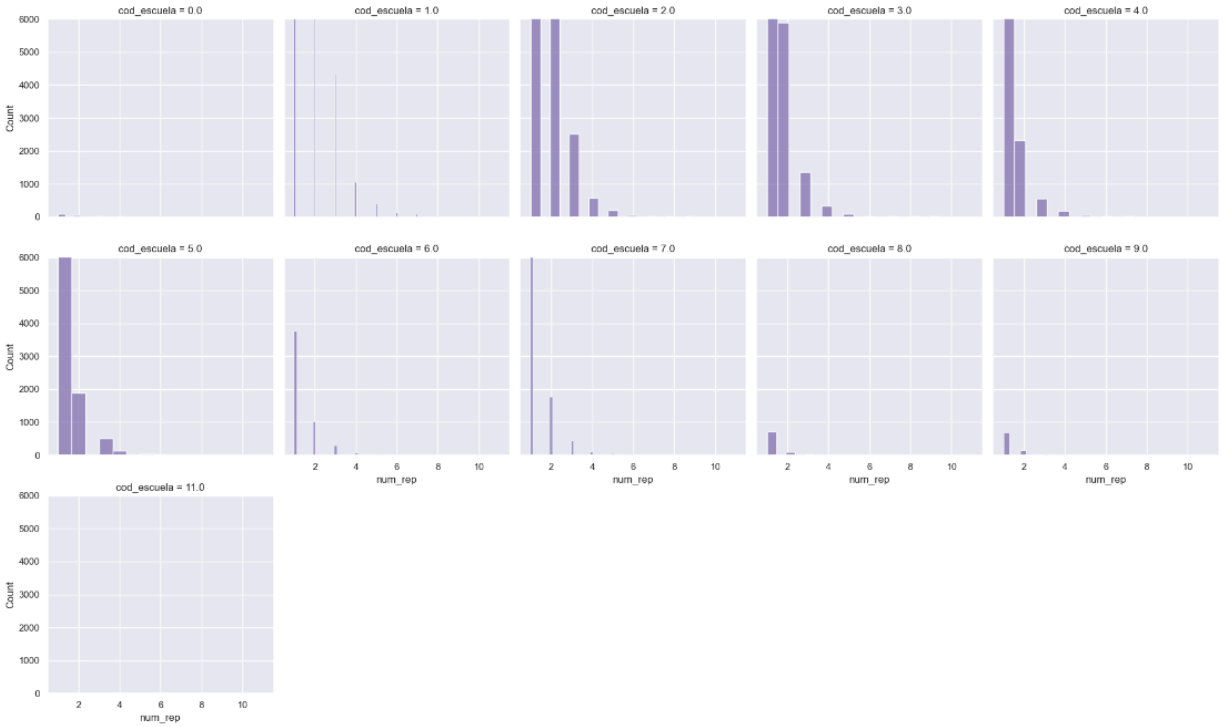
```
In [78]: grafica2=sb.lineplot(x='num_rep',y='REPITENCIAS',data=rep_esc_estudio,hue='cod_escuela')
```



8. **Gráfica 5:** Con FacetGrid

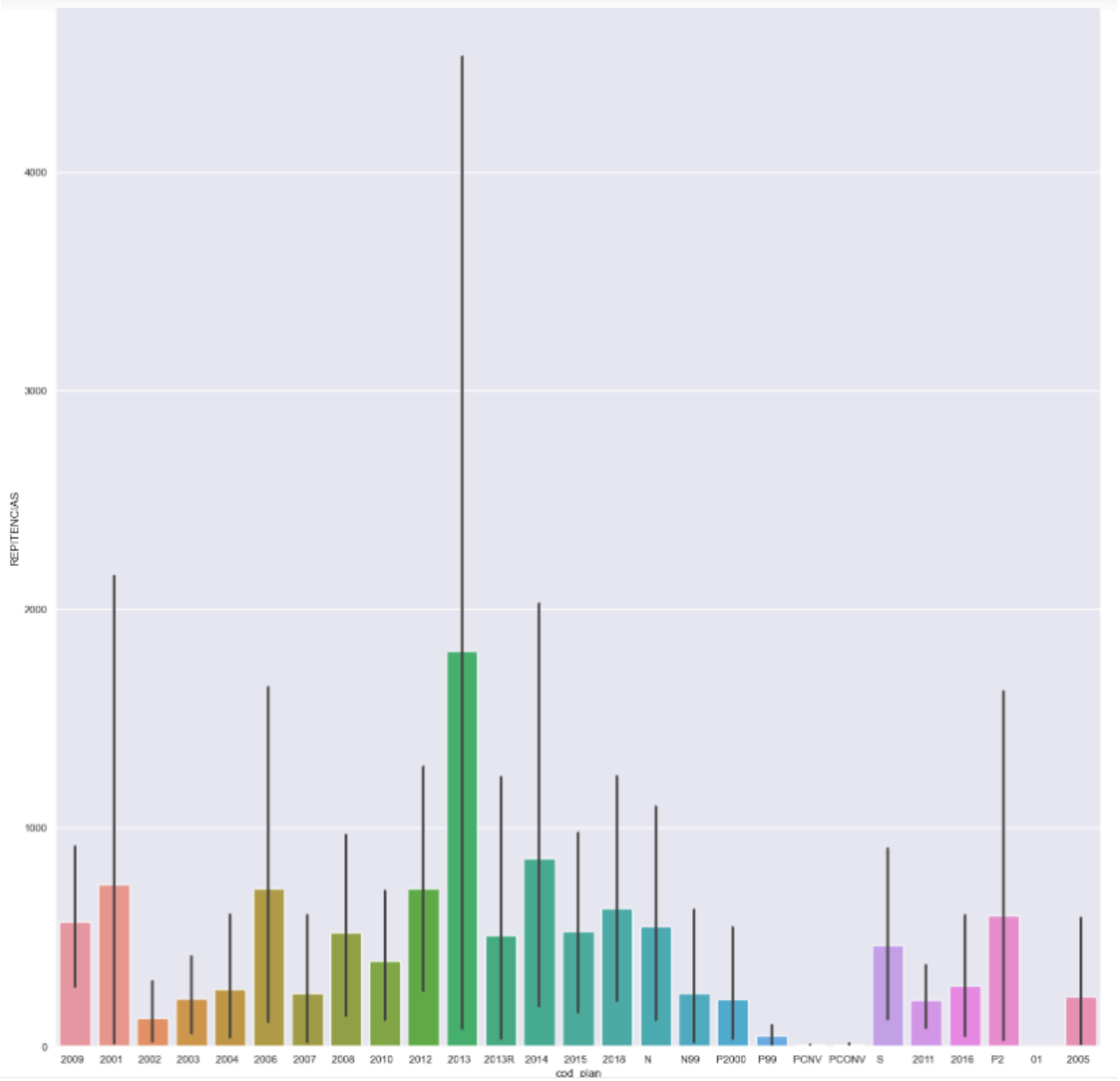
In [80]:
```
#FacetGrid

g = sb.FacetGrid(datoglobal, col="cod_escuela", height=4, col_wrap=5, ylim=(0, 6000))
g.map_dataframe(sb.histplot, x="num_rep",color='m')
```

Out[80]: <seaborn.axisgrid.FacetGrid at 0x2ad37c96ac0>



9. **Gráfica 6:** CANTIDAD DE REPITENCIAS (1 REP, 2DA REP, 3 REP, 4 REP, 5 REP.) por Plan

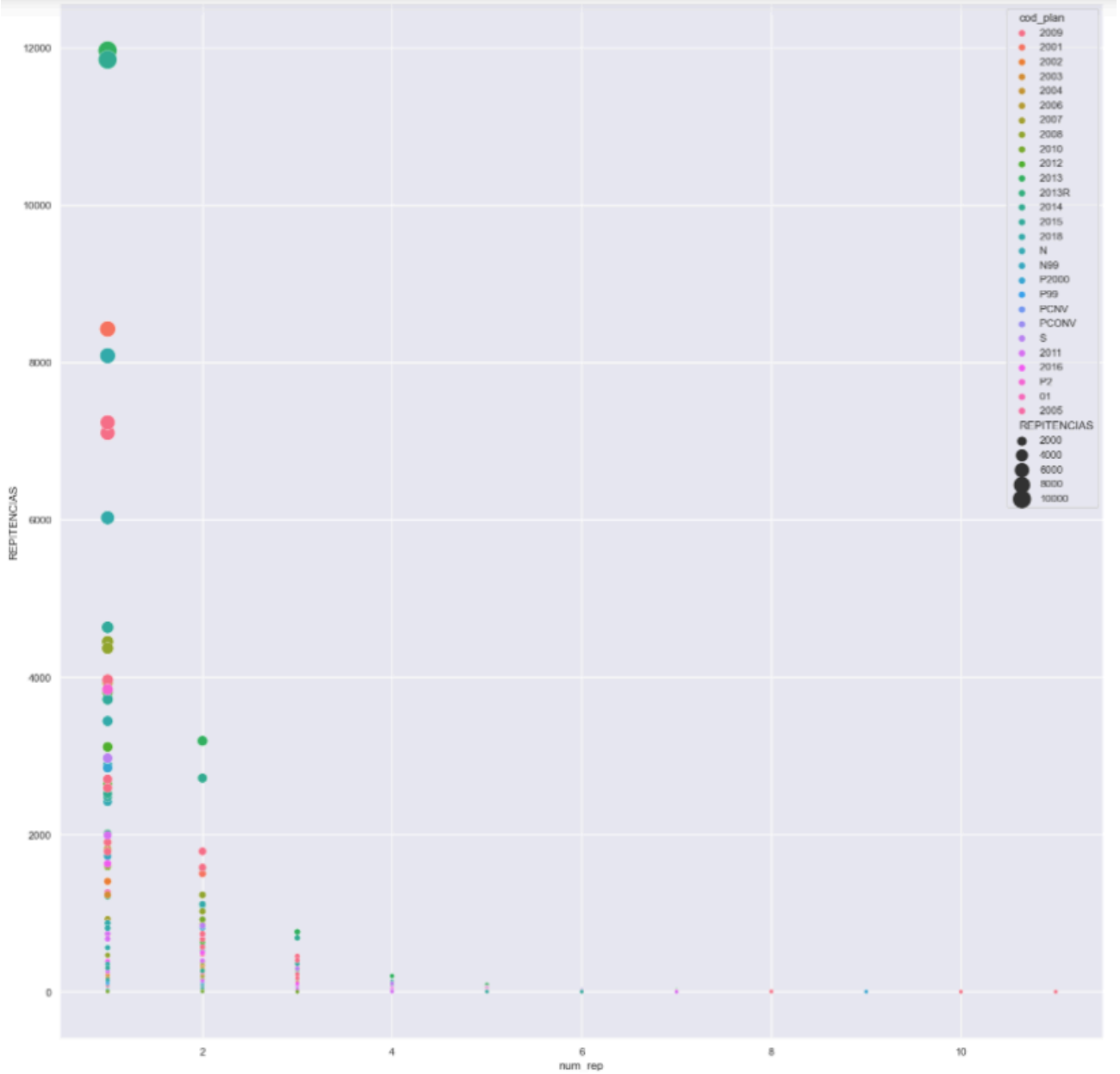
```
In [81]: graficaplan=sb.barplot(x='cod_plan',y='REPITENCIAS',data=rep_esc_estudio)
```



Se obtiene que el plan 2013 tiene mayor número de repitentes

10. Gráfica 7: CANTIDAD DE REPITENCIAS (1 REP, 2DA REP, 3 REP, 4 REP, 5 REP.) por plan

```
In [82]: aficaplan2=sb.scatterplot(x='num_rep',y='REPITENCIAS',data=rep_esc_estudio,hue='cod_plan',size='REPITENCIAS',sizes=(20,400))
```



11. DETERMINAR LA PREDICCIÓN USANDO MACHINE LEARNING COMO SERIA PARA 2020-1 Y 2020-2

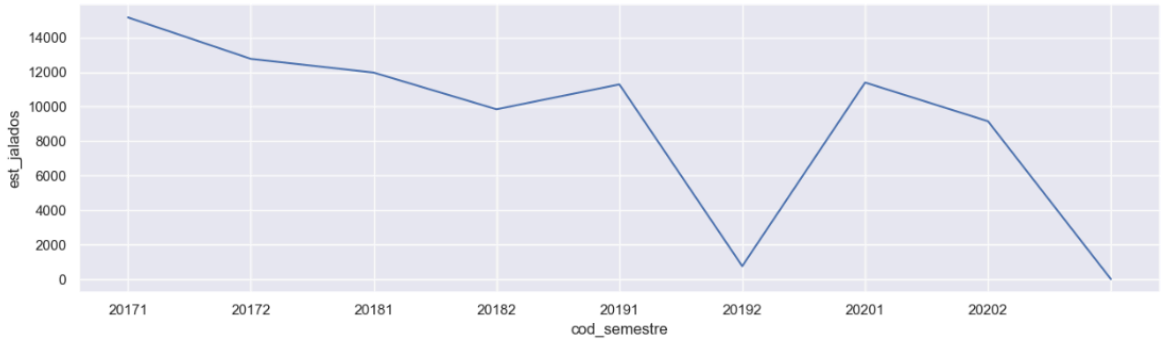
Algoritmo de Regresión

Cantidad de alumnos jalados en cada ciclo, data tomada para la predicción de la cantidad de alumnos que jalarán al menos un curso en el ciclo 2020-I y 2020-II.

In [85]:
```
# Estudiantes jalados por ciclo

est_jal_sem=datoglobal.groupby(['cod_semestre'])['cod_alumno'].nunique().reset_index(name='est_jalados')
print(est_jal_sem)
```

```
  cod_semestre  est_jalados
0        20171        15174
1        20172        12770
2        20181        11974
3        20182         9846
4        20191        11288
5        20192          750
6        20201        11397
7        20202         9153
8                         0
```

In [86]:
```
fig, ax=plt.subplots(figsize=(15,4))

est_jal_sem['cod_semestre'] = est_jal_sem['cod_semestre'].astype(str)

#print(est_jal_sem['cod_semestre'].dtypes)

graf_jal_sem=sb.lineplot(ax=ax,x='cod_semestre',y='est_jalados',ci=None,data=est_jal_sem)
```



12. Implementación del Algoritmo de Regresión Lineal

In [88]:
```
#Algoritmo de Regresión Lineal

from sklearn import linear_model
from sklearn.metrics import r2_score

est_jal_sem['cod_semestre'] = est_jal_sem['cod_semestre'].astype(str)
regr=linear_model.LinearRegression()

x=est_jal_sem['cod_semestre']
y=est_jal_sem['est_jalados']
```

Seleccionamos nuestra data o variables independientes(x) y nuestro target o variable dependiente(y)

Se entrena al algoritmo

In [89]:
```
X=x[:,np.newaxis]
print(X)
```

```
[['20171']
 ['20172']
 ['20181']
 ['20182']
 ['20191']
 ['20192']
 ['20201']
 ['20202']
 [' ']]
```

In [92]:
```
regr=linear_model.LinearRegression()
regr.fit(X,y)
print(regr.fit(X,y))
print(regr.coef_)
```

In [125]:
```
regr=linear_model.LinearRegression()
regr.fit(X,y2)
print(regr.fit(X,y2))
print(regr.coef_)
```

```
LinearRegression()
[-168.14071856]
```

Se halla el resultado

In [128]:
```
X = X.astype(float)
m=regr.coef_[0]
b=regr.intercept_
y_p=m*X+b
print('y={0}*x+{1}'.format(m,b))
print('El valor de R^2: ', r2_score(y2,y_p))
```

```
y=-168.1407185628742*x+3404466.6152694602
El valor de R^2:  0.22207023627135825
```

Gráfica

In [129]:
```
plt.scatter(x2,y2,color='blue')
plt.plot(x2,y_p,color='red')
plt.title('Regresion Lineal',fontsize=16)
plt.xlabel('Eje X',fontsize=13)
plt.ylabel('Eje Y',fontsize=13)
```

13. **Entrenado el algoritmo, creamos un dataframe con 2020-I y 2020-II y se lo pasamos como parámetro al algoritmo para obtener la predicción de jalados para estos nuevos ciclos.**

```
In [130]: X_new=[[20211],[20212]]
          predicciones=regr.predict(X_new)
          print(predicciones)

          [6174.55239521 6006.41167665]
```

In [133]:
```
new_row1 = {'cod_semestre':X_new[0][0],'est_jalados':predicciones[0]}
new_row2 = {'cod_semestre':X_new[1][0],'est_jalados':predicciones[1]}

est_jal_sem_predict=est_jal_sem2.append(new_row1, ignore_index=True)
est_jal_sem_predict=est_jal_sem_predict.append(new_row2, ignore_index=True)

est_jal_sem_predict['cod_semestre']=est_jal_sem_predict['cod_semestre'].astype(int)
est_jal_sem_predict['est_jalados']=est_jal_sem_predict['est_jalados'].astype(int)

print(est_jal_sem_predict)
```

```
   cod_semestre  est_jalados
0         20171        15174
1         20172        12770
2         20181        11974
3         20182         9846
4         20191        11288
5         20192          750
6         20201        11397
7         20202         9153
8         20211         6174
9         20212         6006
```

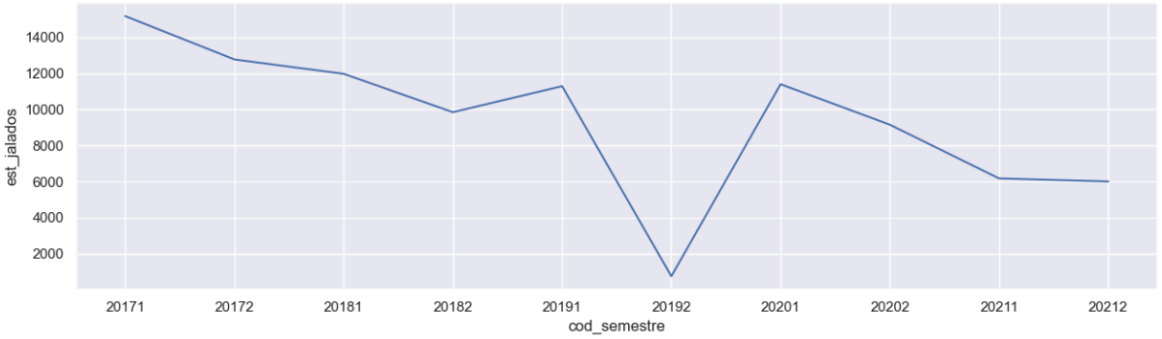
El algoritmo predice 6174 alumnos reprobados para el 2021-I y 6006 para el 2021-II.

In [134]:
```
y_pred=m*np.array([[20211],[20212]])+b
print(y_pred)
```

```
[[6174.55239521]
 [6006.41167665]]
```

Corroboramos que las predicciones son las mismas empleando la función predict() que haciéndolo mediante la fórmula de regresión lineal.

In [135]:
```
est_jal_sem_predict['cod_semestre'] = est_jal_sem_predict['cod_semestre'].astype(str)
fig, ax=plt.subplots(figsize=(15,4))

graf_est_jal_sem_predict=sb.lineplot(ax=ax,x='cod_semestre',y='est_jalados',ci=None,data=est_jal_sem_predict)
```



Data de prueba calificaciones 2017-1 al 2019-2

1. Importamos las librerías y hacemos la carga de la data de calificaciones

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import sys
```

[10] ✓ 0.4s

```
datos=pd.read_excel("C:\\Users\\User22\\Desktop\\CASO-UNMSM\\DATA-PRUEBA_CALIFICACIONES  2017-20120-UNMSM.xlsx")
```

[19] ✓ 48.1s

2. Usamos el método head para que nos devuelva un número específico de filas, cadena desde la parte superior. Este método devolverá las primeras 5 filas si no se especifica un número. Nota: También se devolverán los nombres de las columnas, además de las filas especificadas. Lo hacemos para comprobar la data cargada

```
[20]  datos.head()
      ✓ 0.1s
```

	cod_semestre	cod_facultad	cod_escuela	cod_plan	cod_asignatura	cod_alumno	val_calific_final	Calificaciones
0	20171.0	1.0	1.0	2004	MH0440	14010029	16.0	Bueno
1	20171.0	1.0	1.0	2004	MH0440	14010276	16.0	Bueno
2	20171.0	1.0	1.0	2004	MH0440	13010237	15.0	Bueno
3	20171.0	1.0	1.0	2004	MH0440	13010241	15.0	Bueno
4	20171.0	1.0	1.0	2004	MH0440	12010327	13.0	Aprobado

3. Usamos el método info muestra un resumen de un dataframe, incluyendo información sobre el tipo de los índices de filas y columnas, los valores no nulos y la memoria usada.

```
[21]  datos.info()
      ✓ 0.3s
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 8 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   cod_semestre       353316 non-null   float64
 1   cod_facultad       353316 non-null   float64
 2   cod_escuela        353316 non-null   float64
 3   cod_plan           353316 non-null   object
 4   cod_asignatura     353316 non-null   object
 5   cod_alumno         353316 non-null   object
 6   val_calific_final  353316 non-null   float64
 7   Calificaciones     1048575 non-null  object
dtypes: float64(4), object(4)
memory usage: 64.0+ MB
```

4. Usamos el método describe() para ver algunos detalles estadísticos básicos como percentil, media, estándar, etc. de un marco de datos o una serie de valores numéricos. Notamos que la media de calificación registrada en su totalidad es de 13.

```
[49]  datos.describe()
      ✓ 0.2s
```

	cod_semestre	cod_facultad	cod_escuela	val_calific_final
count	353316.000000	353316.000000	353316.000000	353316.000000
mean	20171.442162	11.221193	2.423191	13.122027
std	0.496644	5.678792	1.702875	3.958768
min	20171.000000	1.000000	1.000000	0.000000
25%	20171.000000	6.000000	1.000000	12.000000
50%	20171.000000	12.000000	2.000000	14.000000
75%	20172.000000	16.000000	3.000000	16.000000
max	20172.000000	20.000000	8.000000	20.000000

5. Usamos el método tail() para mostrarlas n filas inferiores (5 de forma predeterminada) de un marco de datos o serie. En este caso se obtienen valores nulos en las últimas filas.

```
datos.tail()
```

[51] ✓ 0.3s

	cod_semestre	cod_facultad	cod_escuela	cod_plan	cod_asignatura	cod_alumno	val_calific_final
1048570	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1048571	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1048572	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1048573	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1048574	NaN	NaN	NaN	NaN	NaN	NaN	NaN

6. Calculamos la cantidad de alumnos con calificaciones por facultad teniendo en cuenta que:
    - Notas de 0 a 10: Descripción - Desaprobado
    - Notas de 11 a 13: Descripción - Aprobado
    - Notas de 14 a 16: Descripción - Bueno
    - Notas de 17 a 18: Descripción - Muy bueno
    - Notas de 19 a 20: Descripción - Sobresaliente

```
calificaciones_data=datos.groupby(["Calificaciones","cod_facultad"])["Calificaciones"].count().reset_index(name="Conteo")
calificaciones_data
```

[22] ✓ 0.1s

	Calificaciones	cod_facultad	Conteo
0	Aprobado	1.0	5950
1	Aprobado	2.0	4880
2	Aprobado	3.0	1476
3	Aprobado	4.0	3272
4	Aprobado	5.0	1955
...	...	...	...
95	Sobresaliente	16.0	154
96	Sobresaliente	17.0	198
97	Sobresaliente	18.0	505
98	Sobresaliente	19.0	57
99	Sobresaliente	20.0	81

100 rows × 3 columns

7. Calculamos la cantidad de alumnos con calificaciones por escuela teniendo en cuenta que, como se calcula por escuela, irá repitiendo por cada facultad:

```
calificaciones_data1=datos.groupby(["Calificaciones","cod_escuela"])["Calificaciones"].count().reset_index(name="Conteo")
calificaciones_data1
```

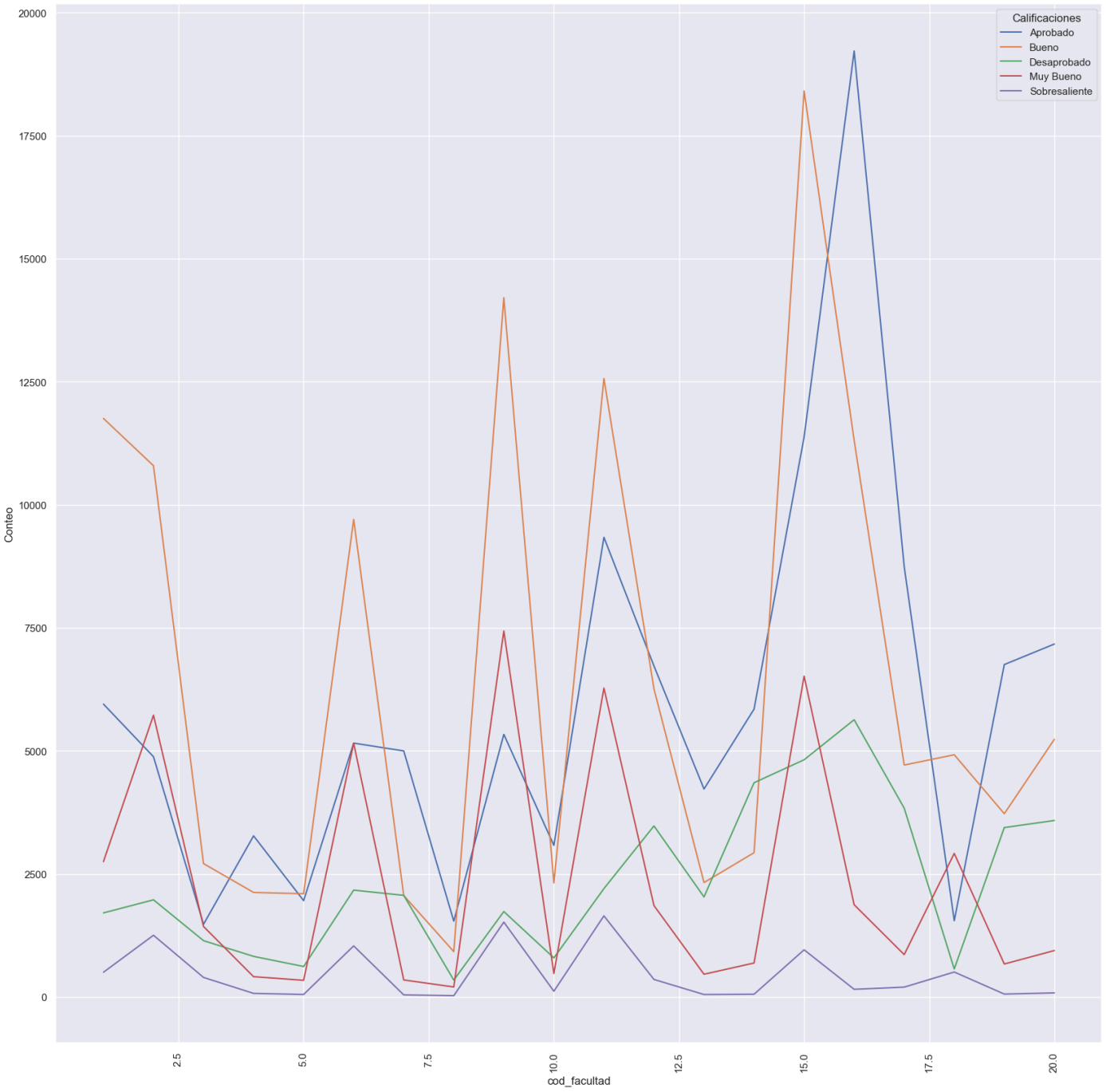
[50] ✓ 0.1s

	Calificaciones	cod_escuela	Conteo
0	Aprobado	1.0	41623
1	Aprobado	2.0	34614
2	Aprobado	3.0	18012
3	Aprobado	4.0	6311
4	Aprobado	5.0	5104
5	Aprobado	6.0	4304
6	Aprobado	7.0	7662
7	Aprobado	8.0	960
8	Bueno	1.0	48604
9	Bueno	2.0	38933
10	Bueno	3.0	20354
11	Bueno	4.0	7378
12	Bueno	5.0	6098
13	Bueno	6.0	2628
14	Bueno	7.0	5968
15	Bueno	8.0	1104
16	Desaprobado	1.0	18137
17	Desaprobado	2.0	12959

8. Elaboramos la gráfica donde se muestra la variación de calificaciones por código de facultades.

```
sb.set(rc={"figure.figsize":(20,20)})
grafica=sb.lineplot(x="cod_facultad",y="Conteo",data=calificaciones_data,hue="Calificaciones")
locs,labels=plt.xticks()
plt.setp(labels,rotation=90)
```

[23] ✓ 0.4s

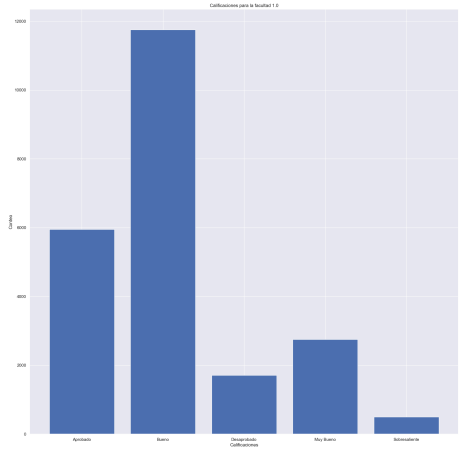


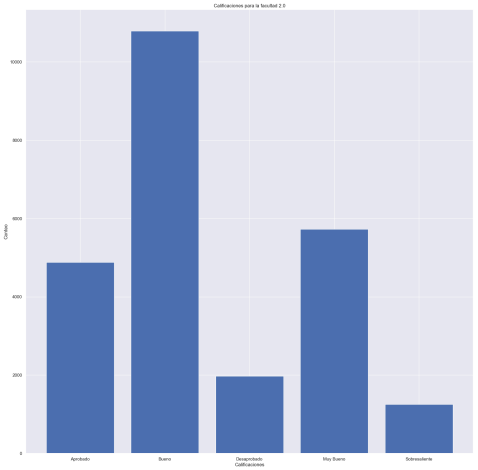
9. Elaboramos las gráficas de calificaciones para cada una de las facultades. Primero, obtenemos los valores únicos de código de facultad, luego iteramos sobre cada uno

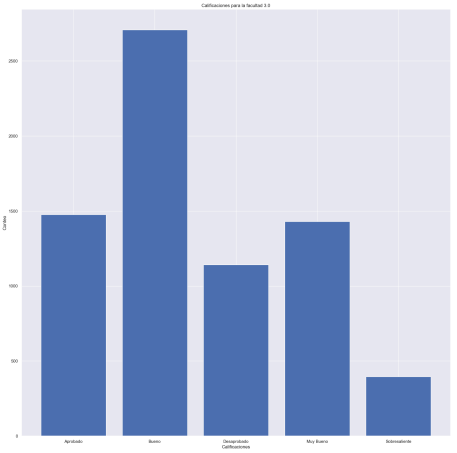
de estos códigos, luego filtramos el data frame para que solo nos de los datos de la facultad actual y luego creamos los gráficos de barras.

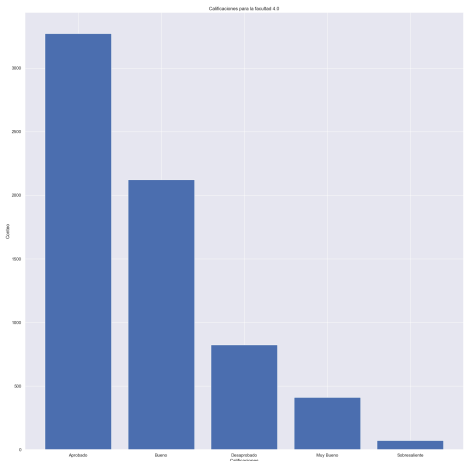
```
cod_facultad_values = calificaciones_data["cod_facultad"].unique()

for cod_facultad in cod_facultad_values:
    df_filtered = calificaciones_data[calificaciones_data["cod_facultad"] == cod_facultad]

    plt.bar(df_filtered["Calificaciones"], df_filtered["Conteo"])
    plt.title("Calificaciones para la facultad {}".format(cod_facultad))
    plt.xlabel("Calificaciones")
    plt.ylabel("Conteo")
    plt.show()
```

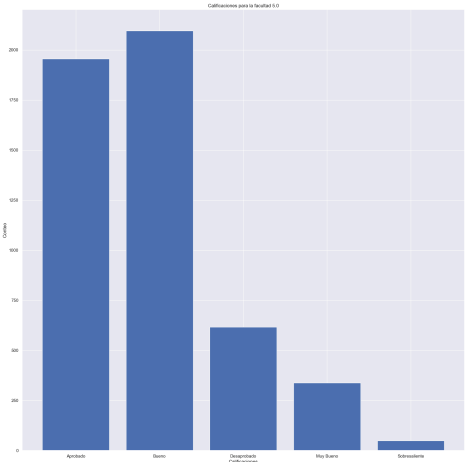
[48] ✓ 4.2s

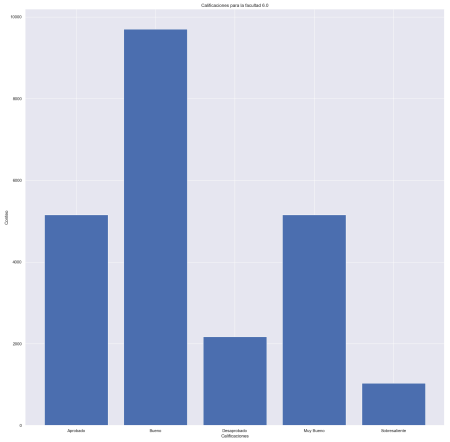


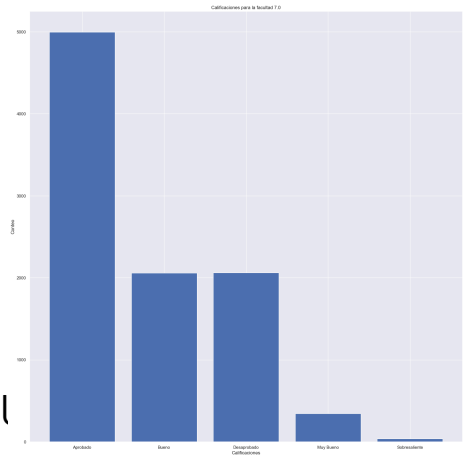


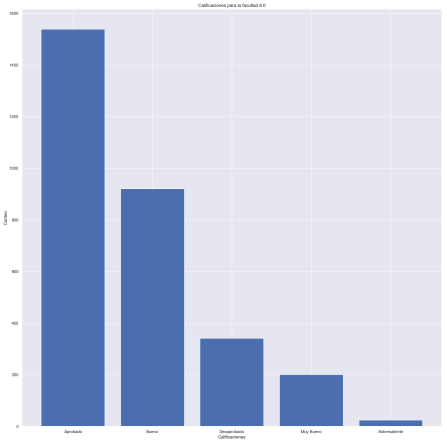


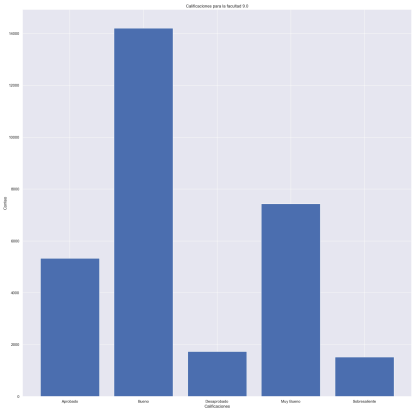












10. Añadimos una columna más al data frame para categorizar la sección de aprobados y desaprobados

```
datos2=datos
conditions=[
    (datos2["val_calific_final"]<10),
    (datos2["val_calific_final"]<=10)]
choices=["Aprobados","Desaprobados"]
datos2["Condicion"]=np.select(conditions,choices,default="No tiene calificacion")
datos2
```

[24] ✓ 0.1s

	cod_semestre	cod_facultad	cod_escuela	cod_plan	cod_asignatura	cod_alumno	val_calific_final	Calificaciones	Condicion
0	20171.0	1.0	1.0	2004	MH0440	14010029	16.0	Bueno	No tiene calificacion
1	20171.0	1.0	1.0	2004	MH0440	14010276	16.0	Bueno	No tiene calificacion
2	20171.0	1.0	1.0	2004	MH0440	13010237	15.0	Bueno	No tiene calificacion
3	20171.0	1.0	1.0	2004	MH0440	13010241	15.0	Bueno	No tiene calificacion
4	20171.0	1.0	1.0	2004	MH0440	12010327	13.0	Aprobado	No tiene calificacion
...	...	...	...	...	...	...	...	...	...
1048570	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Desaprobado	No tiene calificacion
1048571	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Desaprobado	No tiene calificacion
1048572	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Desaprobado	No tiene calificacion
1048573	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Desaprobado	No tiene calificacion
1048574	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Desaprobado	No tiene calificacion

1048575 rows × 9 columns

11. Determinamos la cantidad de alumnos aprobados o desaprobados por código de plan

```
alumnos_Condicion=datos.groupby(["cod_plan","Condicion"])["Condicion"].count().reset_index(name="Conteo")
alumnos_Condicion
```

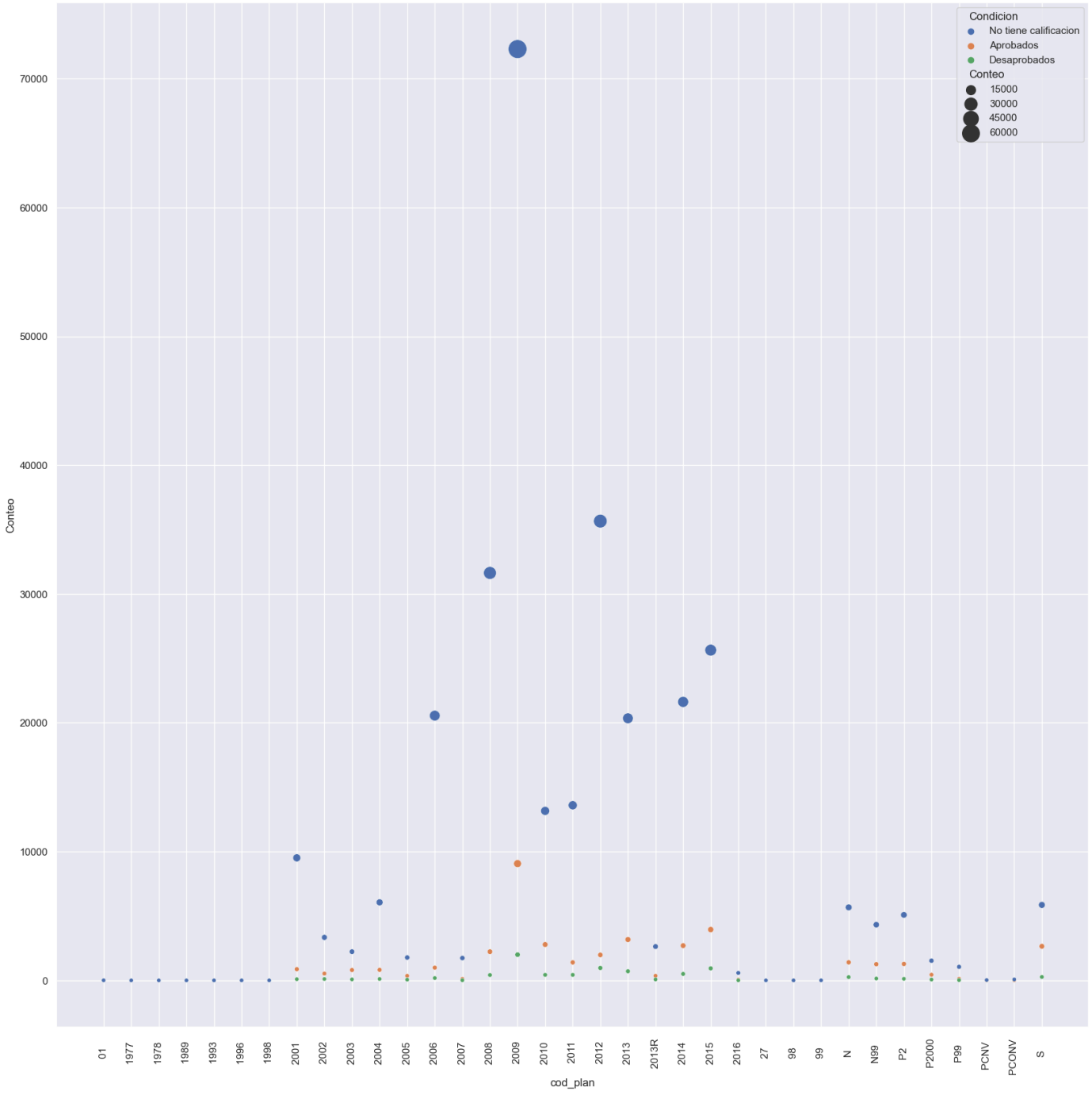
[25] ✓ 0.1s

	cod_plan	Condicion	Conteo
0	01	No tiene calificacion	8
1	1977	No tiene calificacion	1
2	1978	No tiene calificacion	1
3	1989	No tiene calificacion	1
4	1993	No tiene calificacion	2
...	...	...	...
78	PCONV	Aprobados	8
79	PCONV	No tiene calificacion	76
80	S	Aprobados	2646
81	S	Desaprobados	264
82	S	No tiene calificacion	5859

83 rows × 3 columns

12. Se realiza grafica correspondientes donde se observa que:
    Plan 2018 tiene la mayor cantidad de aprobados

```
graficaplan2=sb.scatterplot(x="cod_plan",y="Conteo",data=alumnos_Condicion,hue="Condicion",size="Conteo",sizes=(20,400))
locs,labels=plt.xticks()
plt.setp(labels,rotation=90)
```
[26]



13. Determinando la cantidad de alumnos desaprobados y aprobados por escuela

```
alumnos_Condicion2=datos.groupby(["cod_escuela","Condicion"])["Condicion"].count().reset_index(name="Conteo")
alumnos_Condicion2
```

[27]

	cod_escuela	Condicion	Conteo
0	1.0	Aprobados	15189
1	1.0	Desaprobados	2948
2	1.0	No tiene calificacion	113718
3	2.0	Aprobados	10896
4	2.0	Desaprobados	2063
5	2.0	No tiene calificacion	89906
6	3.0	Aprobados	5754
7	3.0	Desaprobados	1304
8	3.0	No tiene calificacion	46625
9	4.0	Aprobados	2220
10	4.0	Desaprobados	607
11	4.0	No tiene calificacion	16030
12	5.0	Aprobados	1778
13	5.0	Desaprobados	376
14	5.0	No tiene calificacion	14362
15	6.0	Aprobados	1359
16	6.0	Desaprobados	277

14. Determinando la cantidad de alumnos aprobados o desaprobados por facultad

```
alumnos_Condicion3=datos.groupby(["cod_facultad","Condicion"])["Condicion"].count().reset_index(name="Conteo")
alumnos_Condicion3
```

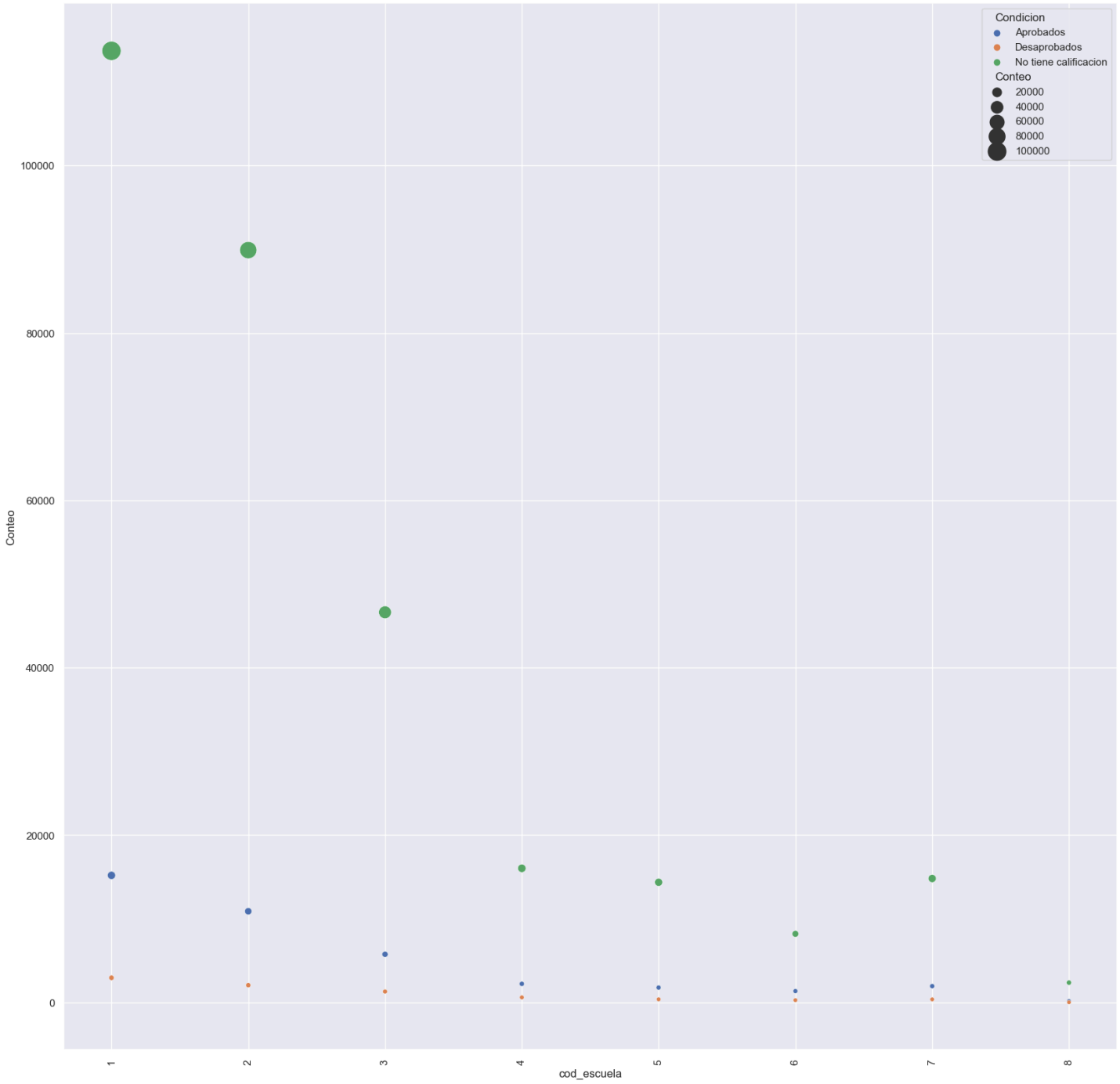
[28]

	cod_facultad	Condicion	Conteo
0	1.0	Aprobados	909
1	1.0	Desaprobados	797
2	1.0	No tiene calificacion	20954
3	2.0	Aprobados	1710
4	2.0	Desaprobados	262
5	2.0	No tiene calificacion	22651
6	3.0	Aprobados	960
7	3.0	Desaprobados	182
8	3.0	No tiene calificacion	6009
9	4.0	Aprobados	648
10	4.0	Desaprobados	175
11	4.0	No tiene calificacion	5875
12	5.0	Aprobados	482
13	5.0	Desaprobados	135
14	5.0	No tiene calificacion	4437
15	6.0	Aprobados	1979
16	6.0	Desaprobados	190

15. Gráfica de la cantidad de alumnos desaprobados y aprobados por escuela

```
graficaplan2=sb.scatterplot(x="cod_escuela",y="Conteo",data=alumnos_Condicion2,hue="Condicion",size="Conteo",sizes=(20,400))
locs,labels=plt.xticks()
plt.setp(labels,rotation=90)
```

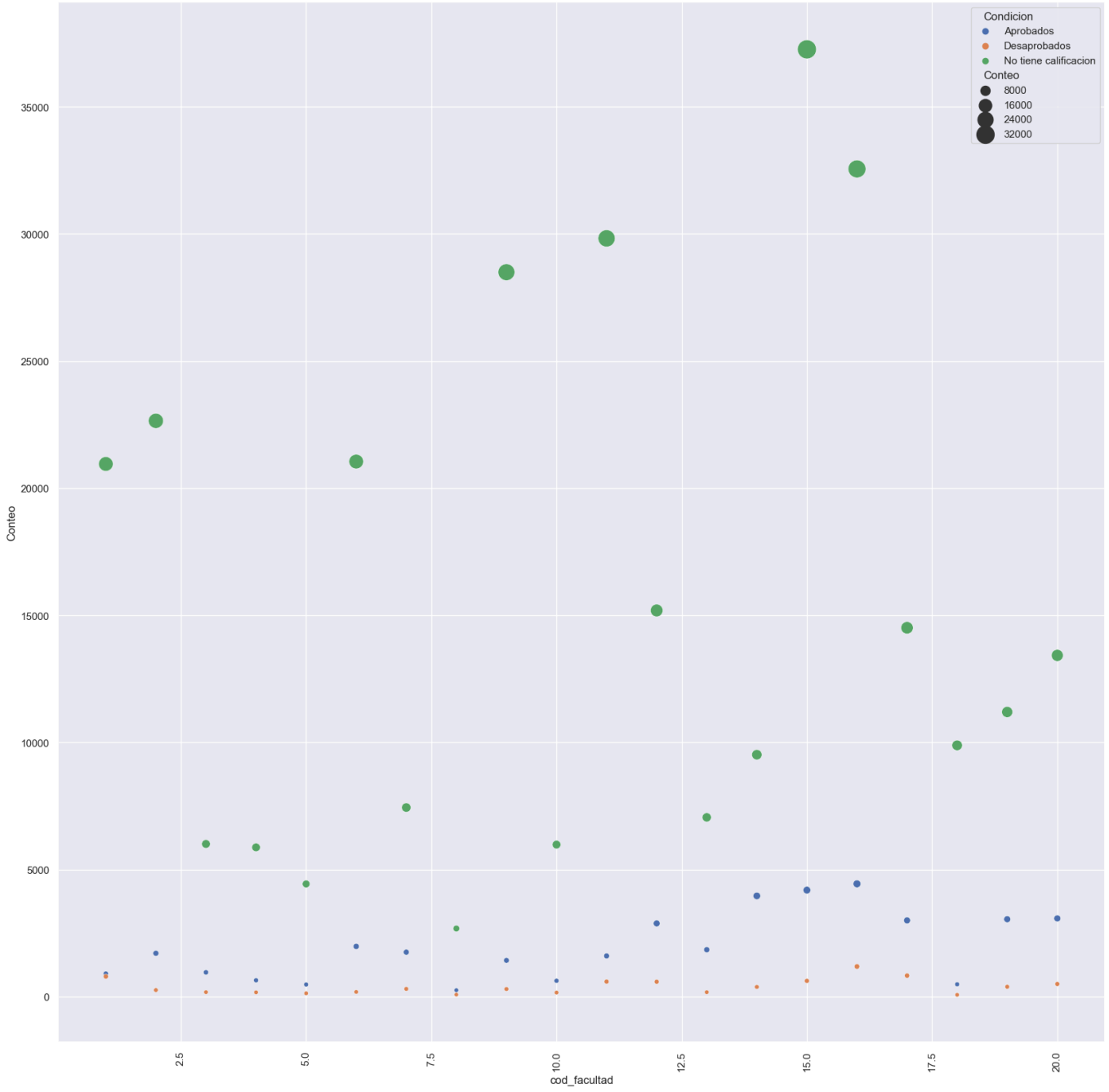
[20] ✓ 0.5s



16. Gráfica de la cantidad de alumnos desaprobados y aprobados por facultad

```
graficaplan2=sb.scatterplot(x="cod_facultad",y="Conteo",data=alumnos_Condicion3,hue="Condicion",size="Conteo",sizes=(20,400))
locs,labels=plt.xticks()
plt.setp(labels,rotation=90)
```

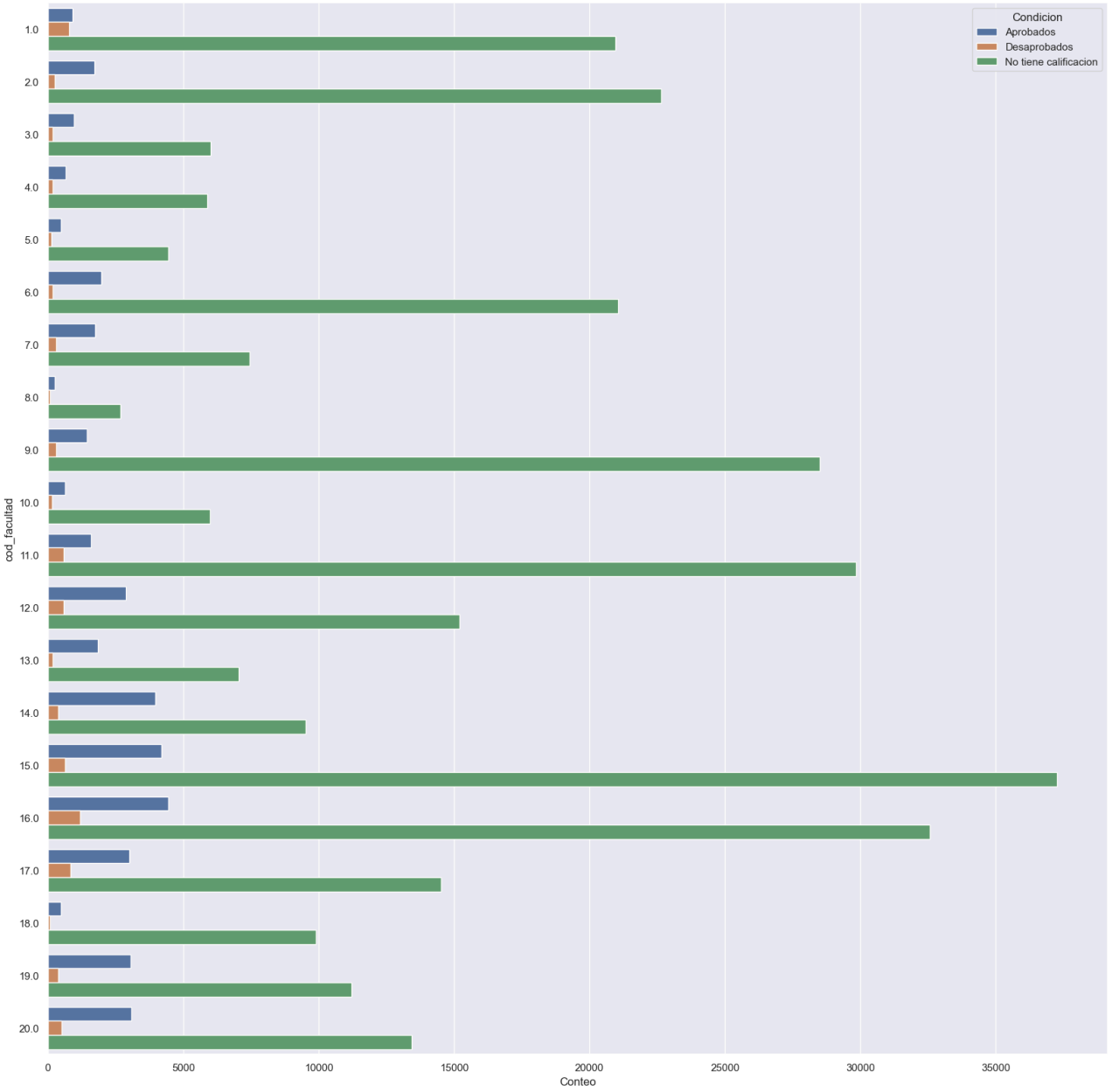
[21] ✓ 0.5s



17. Graficando alumnos aprobados y desaprobados por código de facultad

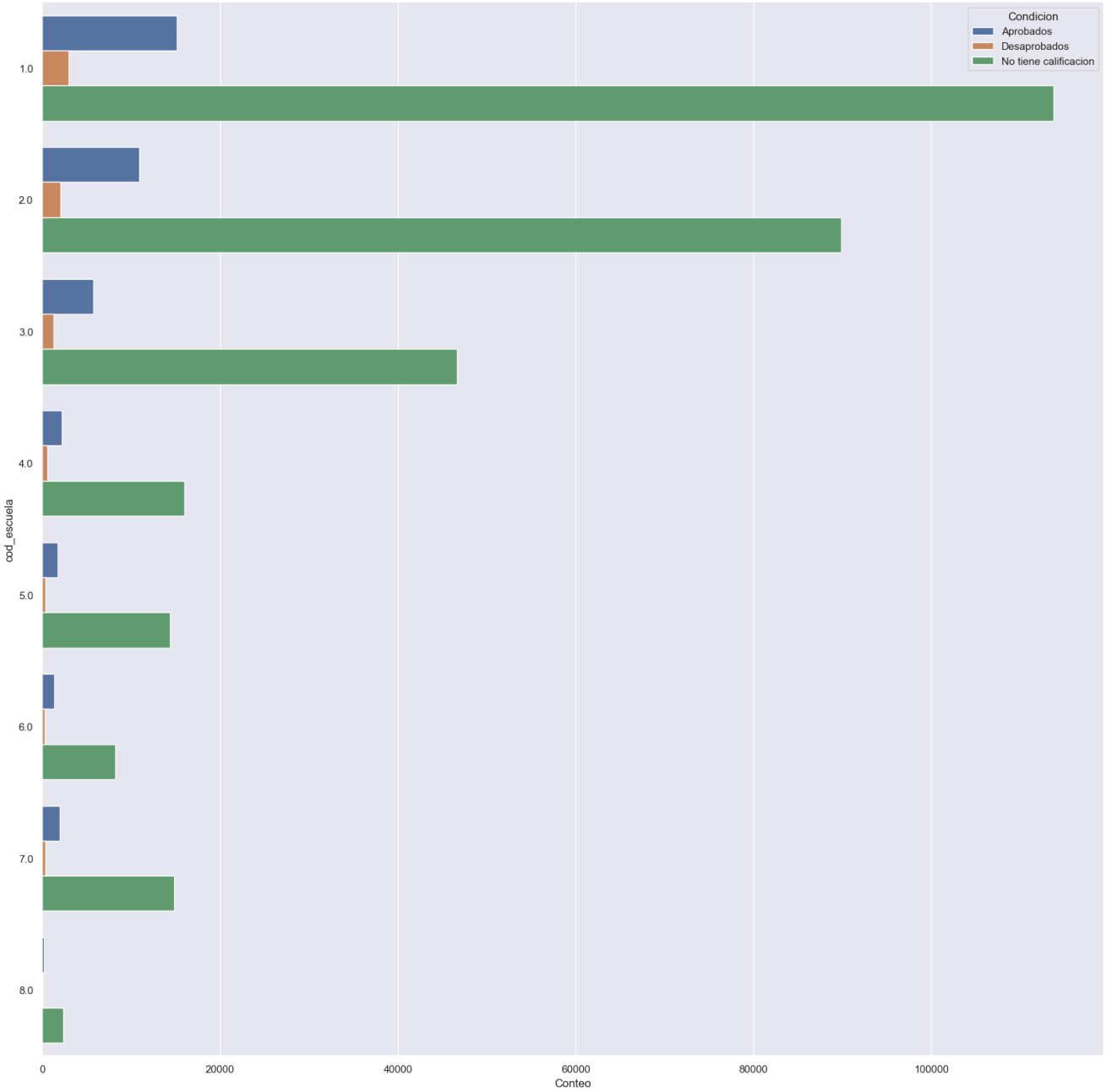
```
grafica4=sb.barplot(x="Conteo",y="cod_facultad",data=alumnos_Condicion3,hue="Condicion",orient="h")
```

[18] ✓ 0.5s



18. Graficando alumnos aprobados y desaprobados por código de escuela

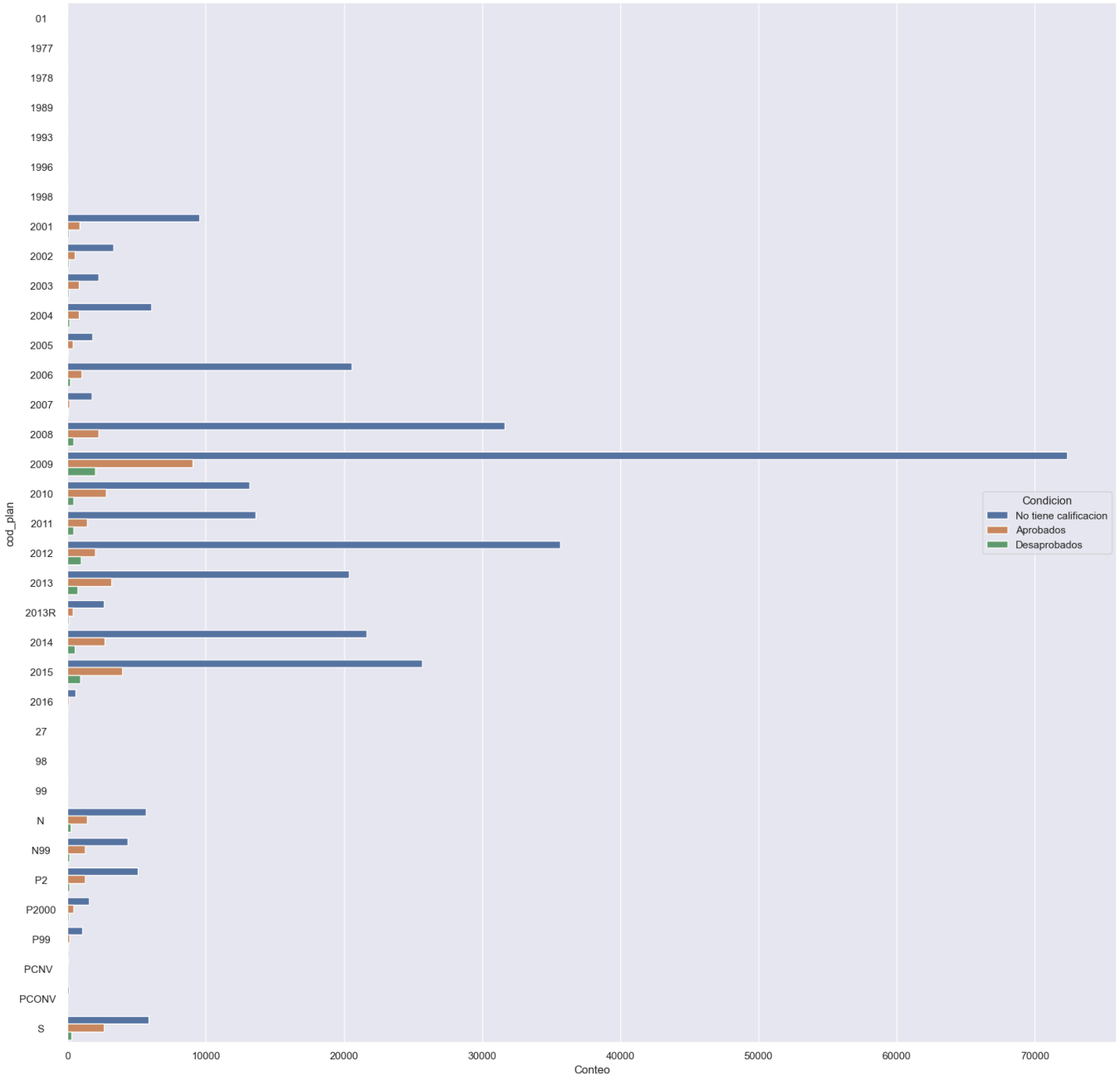
```
grafica4=sb.barplot(x="Conteo",y="cod_escuela",data=alumnos_Condicion2,hue="Condicion",orient="h")
```
[22] ✓ 0.4s



19. Graficando alumnos aprobados y desaprobados por código de plan

```
grafica4=sb.barplot(x="Conteo",y="cod_plan",data=alumnos_Condicion,hue="Condicion",orient="h")
```

[24] ✓ 0.8s



Data de prueba tutoría 2017-1 al 2019-2

(página 44-55)

1. Importamos las librerías y hacemos la carga de la data de repitencias

In [9]:
```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt

archivo='DATA-PRUEBA_TUTORIAS 2017-2020-UNMSM.xlsx'
datos=pd.read_excel('C:\\Users\\acer\\Pictures\\6to CICLO\\BIG DATA\\EXAMEN\\CASO-UNMSM\\'+archivo)
#datoglobal=pd.concat(datos,ignore_index=True)
datos.head()
```

Out[9]:

	cod_alumno	cod_semestre	cod_facultad	cod_escuela	num_res_autoriza	cod_tipo_autorizacion
0	08010216	20171	1	1	RD 0408-D-FM-2017	AM
1	13010409	20171	1	1	RD 0408-D-FM-2017	AM
2	13010044	20171	1	1	RD 0432-D-FM-2017	AM
3	12010212	20171	1	1	RD 0472-D-FM-2017	AM
4	12010242	20171	1	1	RD 0472-D-FM-2017	AM

2. Importamos las librerías y hacemos la carga de la data de repitencias

In [11]:
```
#DETERMINAR POR FACULTAD, CANTIDAD DE ALUMNOS CON TUTORIA (SEMESTRE 2017 1-2, 2018 1-2 y 2019 1-2)

tutorias=datos.groupby(['cod_facultad','cod_semestre'])['cod_alumno'].nunique().reset_index(name='TUTORIAS')
tutorias
```

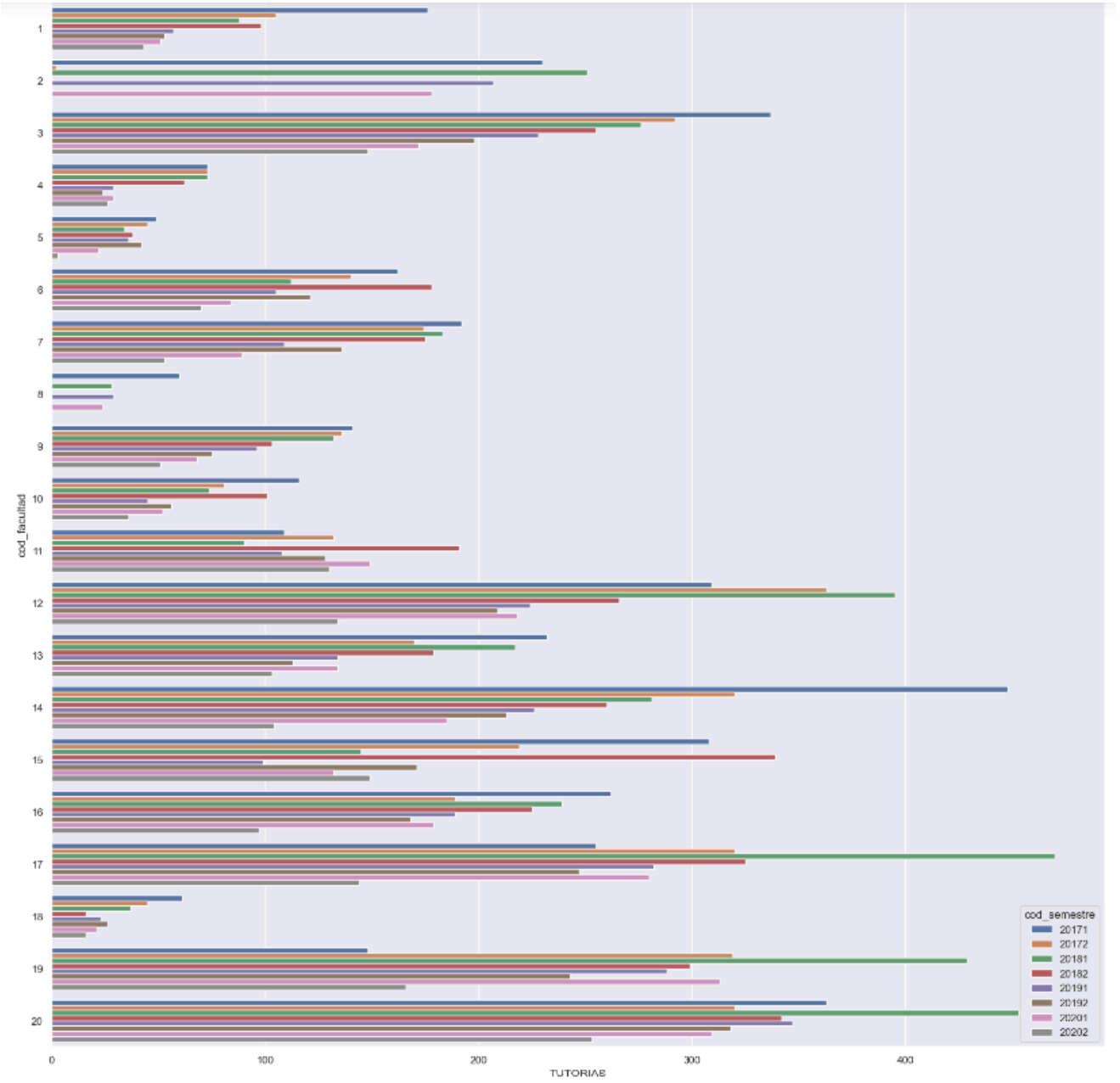
Out[11]:

	cod_facultad	cod_semestre	TUTORIAS
0	1	20171	176
1	1	20172	105
2	1	20181	88
3	1	20182	98
4	1	20191	57
...	...	...	...
148	20	20182	342
149	20	20191	347
150	20	20192	318
151	20	20201	309
152	20	20202	253

153 rows × 3 columns

3. fgfgf

In [14]:
```
sb.set(rc={'figure.figsize':(20,20)})
grafica=sb.barplot(x='TUTORIAS',y='cod_facultad',data=tutorias,hue='cod_semestre',orient='h')
```


cod_facultad
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
0
100
200
300
400
TUTORIAS
cod_semestre
20171
20172
20181
20182
20191
20192
20201
20202

4. DETERMINAR Por escuela,CANTIDAD DE ALUMNOS CON TUTORÍA (SEMESTRE 2017 1-2, 2018 1-2 y 2019 1-2)

In [16]:
```
#DETERMINAR Por escuela,CANTIDAD DE ALUMNOS CON TUTORIA (SEMESTRE 2017 1-2, 2018 1-2 y 2019 1-2)

tutorias2=datos.groupby(['cod_escuela','cod_semestre'])['cod_alumno'].nunique().reset_index(name='TUTORIAS')
tutorias2
```

Out[16]:

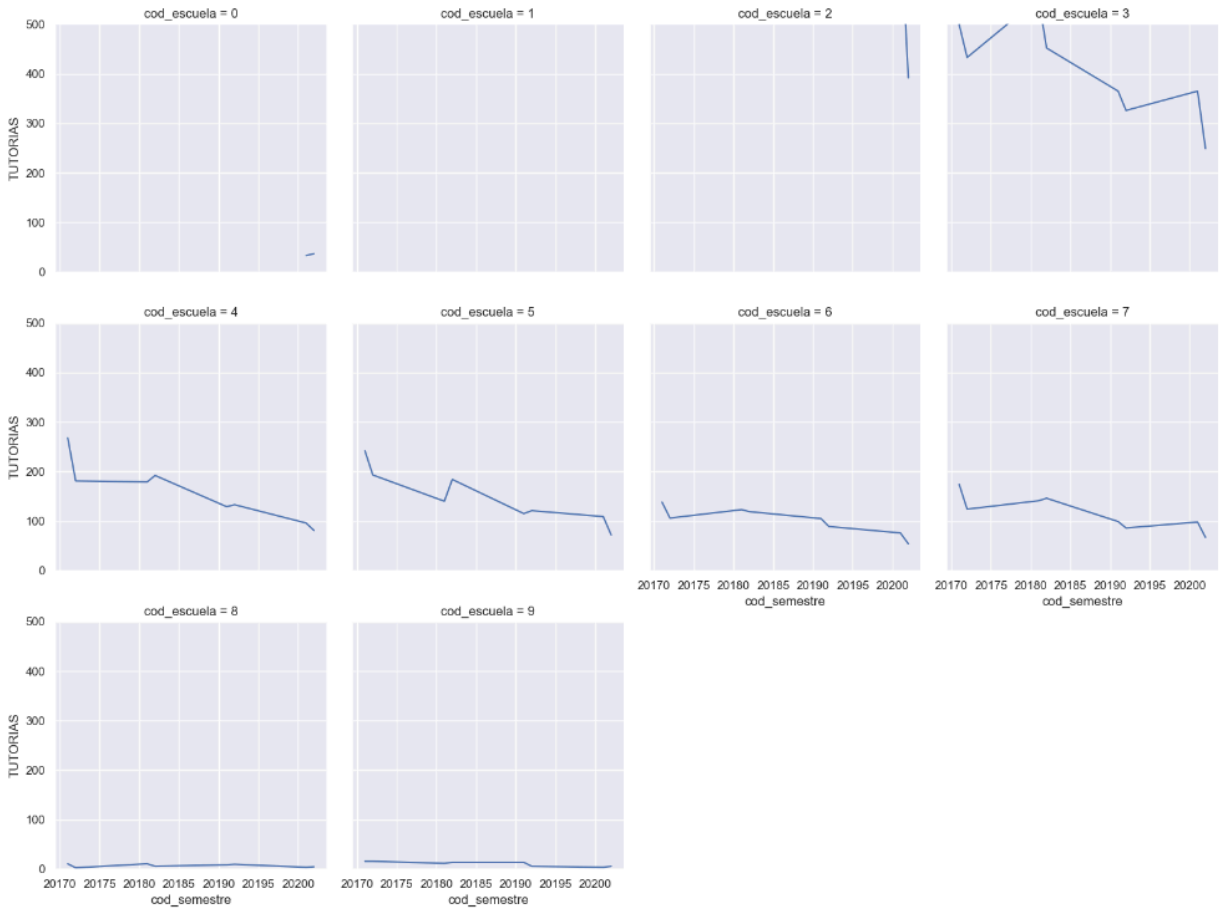
	cod_escuela	cod_semestre	TUTORIAS
0	0	20201	33
1	0	20202	37
2	1	20171	1661
3	1	20172	1518
4	1	20181	1739
...	...	...	...
69	9	20182	14
70	9	20191	14
71	9	20192	6
72	9	20201	4
73	9	20202	6

74 rows × 3 columns

5. Gráfica por escuela categorizado por semestre para mejor entendimiento

```
In [17]: #Grafica por escuela categorizado por semestre para mejor entendimiento

         g = sb.FacetGrid(tutorias2, col="cod_escuela", height=4, col_wrap=5, ylim=(0, 500))
         g.map(sb.lineplot, "cod_semestre", "TUTORIAS", ci=None)
```

```
Out[22]: <seaborn.axisgrid.FacetGrid at 0x11e2e715130>
```



Resultados

- Con respecto a la data de repetición. El semestre con más número de repitentes fue el 2017-I. La Facultad con la mayor cantidad de repitentes es la de código 15 y la Facultad con la menor cantidad de repitentes es la de 8. Nadie de la escuela de código 10 pasa de la 3ra repitencia. Se obtiene que el plan 2013 tiene mayor número de repitentes.

En el apartado de regresión lineal, el algoritmo predice 6174 alumnos reprobados para el 2021-I y 6006 para el 2021-II.Esto es corroborado por la función *predict().*

- Con respecto a la data de calificaciones podemos observar que la facultad 16 concentra la mayor cantidad de aprobados, y la facultad de código 8 concentra la mayor cantidad de estudiantes sobresalientes. Con respecto al código de plan, el plan 2009 concentra la mayor cantidad de estudiantes aprobados, pero a la par concentra la mayor cantidad de desaprobados.
- Con respecto a la data de tutorías podemos observar que se nos brinda la data de las tutorías realizadas en cada facultad con su respectivo código de escuela y el código que genera cada tutoría. Luego de esto procederemos a observar la información, respecto a código de facultad y a los códigos de semestres a los que pertenece a la vez que muestra la cantidad de tutorías que hay para cada facultad; luego se procede a realizar lo mismo, pero con cada escuela, por lo cual obtendremos más datos debido a que cada facultad cuenta con una cierta cantidad de escuelas. Por último, tenemos las gráficas visuales donde observamos respecto a las facultades la cantidad de estudiantes que han estado en tutorías con respecto a su código de semestre.

Conclusiones

- Por el lado de las repitencias, se concluye que con la ayuda de las librerías numpy y seaborn la visualización de los datos se da de una manera más óptima y entendible a simple vista. Se rescata que la mayor parte de los estudiantes jalados se encuentra en la base 17 (semestre 2017 I y II); este número fue bajando con el pasar del tiempo. Junto a ello se encuentran las proyecciones para los ciclos 2021 I y II los cuales continúan con la tendencia a la baja. En tanto a los planes de estudio , se podría decir que el 2013 (el que conserva el mayor número de estudiantes

jalados) debe considerar un futuro análisis y/o reestructuración para llegar a reducir tal número.

- Por el lado de las calificaciones, se concluye que gracias a las librerías matplotlib y seaborn se permite tener una visualizacion rapida y sencilla de las calificaciones de los estudiantes a través de los gráficos estadísticos que se llegan a formar, matplotlib permitió la generación más compleja de gráficas como la elaboración de histogramas de acuerdo al código de plan, al año, a las facultades y a los códigos de escuela. De esta manera se pueden llegar a interpretaciones importantes y rápidas con respecto a lo que se pueda tomar en las calificaciones.
- Por parte de las tutorías la facultad con código de escuela igual valor a 3, corresponde a la facultad de Química e Ingeniería Química, siendo esta la que cuenta con mayo cantidad de tutorías a lo largo de todos los semestres, además podemos observar que escuelas con valor igual a 0, 1, 8 y 9 cuentas con tan pocas cantidades de tutorías o nulas, que estas prácticamente no se ven reflejadas en las gráficas. Además notamos la ausencia de información como lo es el código de alumno, número de respuesta y código de tipo de autorización, esto debido a que resulta totalmente inutil para el modelo, debido a que no ayuda a observar el modelado de la cantidad de tutorías que se presentan en la Universidad Nacional Mayor de San Marcos.

Referencias