

# practica

January 4, 2021

## 1 Práctica 2

### Tipología y ciclo de vida de los datos

#### 1.0.1 Isabel Cabezas Martín

#### 1.0.2 Jorge Saura Fillat

En este repositorio: <https://github.com/Jorge-Saura/UOC-Tipologia-Practica2/> Encontramos los siguientes archivos:

**Dataset/EES\_2018.csv** - dataset descargado del [INE](#) con los resultados de la encuesta de estructura salarial.

**dr\_EES\_2018.xlsx** - fichero descargado también de la web del INe donde se explica el diseño del dataset con el que trabajamos.

**practica.ipynb** - Este fichero, la práctica en sí misma, con el análisis. Vemos el código, los comentarios e iremos comentando cada uno de los apartados / tareas a realizar expuestas en el enunciado de la práctica.

## 2 1. Descripción del dataset ¿Por qué es importante y que pregunta/problema pretende responder?

El dataset elegido se corresponde a la [Encuesta de estructura salarial](#), hecha por el [Insituto Nacional de Estadística](#).

En estas encuestas, encontramos la información (desde 1995) sobre la estructura y distribución de los salarios en España. Es una encuesta que se realiza por los diferentes países a nivel europeo, y que está regulada de forma común, de forma que investiga en todos los estados miembros los salarios en función de una gran variedad de variables, como el sexo, la ocupación, la rama de la actividad, la antigüedad (o experiencia del trabajador), o el tamaño de la empresa (y muchos más, en total tenemos 56 campos diferentes, tal y como veremos más adelante).

Es una encuesta que se realiza bianualmente, y además, se obtiene de los ficheros de la Seguridad Social y de la Agencia Tributaria, además de la utilización de un cuestionario específico. Podemos consultar muchos más detalles sobre cómo se realiza esta encuesta, en la [web del INE](#).

Existen numerosos datasets que comparan sueldos con otras variables en múltiples páginas de fuentes de datos libres. Podemos encontrar información parecida en Kaggle ([Employee Salary Dataset] (<https://www.kaggle.com/varungitboi/employee-salary-dataset>), [Kaggle](#), [SF Salaries](#) |

[Kaggle, Education, Languages and Salary](#) | [Kaggle, Where it Pays to Attend College](#)) sobre la comparativa de salarios en diferentes países del mundo, con respecto a la educación recibida por el trabajador, con respecto a los años de experiencia, e incluso según a qué universidad (estadounidense) asistieron.

Pero hemos decidido utilizar un dataset que refleje los datos nuestro país, por cercanía, por interés personal, por lo completo del mismo (tenemos 56 columnas, comparadas con las 10 o 12 que hemos encontrado en otros parecidos), por la cantidad de datos que tenemos en él (216726 muestras), porque es más fácil sacar conclusiones conociendo nuestro día a día y nuestra cultura, pero sobre todo, porque es un dataset del “mundo real”, más complejo sin duda, pero estamos seguros de que su estudio nos enriquecerá a nivel académico y personal.

Con este dataset, buscamos contestar **preguntas sobre la distribución de salarios en la población española**. Nos gustaría estudiar/averiguar si hay una correspondencia clara en cuanto a la experiencia/antigüedad y el salario (¿Son los trabajadores con más experiencia los que más cobran?), ¿cómo se distribuye el sueldo según el sexo? ¿tenemos una brecha de género con respecto al sexo en nuestro país?. Incluso haciendo zoom a nivel de actividad económica, ¿Cuales son las industrias mejores pagadas y las peores? ¿Y cuando al sexo de los trabajadores? ¿Cobran más los trabajadores con más estudios? ¿O es más importante el tipo de responsabilidad que el trabajador tiene en una empresa? ¿Hay diferencias de sueldo entre diferentes regiones del país?

Quedan fuera del ámbito de este estudio las características que son ajenas a la empresa y/o el trabajador como: días de baja, incapacidades, etc. No tendremos en cuenta en qué convenio o sector la seguridad social paga mejor en caso de baja (por ejemplo).

```
[1]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import numpy as np
import statsmodels.api as sm
import scipy.stats as stats

from scipy.stats import wilcoxon
from scipy.stats import chi2_contingency

import warnings
warnings.filterwarnings("ignore")

%matplotlib inline

plt.style.use('bmh')
```

Añadimos el siguiente cambio de etilo para poder alinear las tablas a la izquierda.

```
[2]: %%html
<style>
    table {margin-left: 0 !important;}
```

</style>

<IPython.core.display.HTML object>

Cargamos nuestro dataset extraido del INE.

```
[3]: dfSueldos = pd.read_csv('./Dataset/EES_2018.csv', delimiter="\t")
dfSueldos.head()
```

```
[3]:  IDENCCC  ORDENTRA  NUTS1  CNACE  ESTRATO2  CONTROL  MERCADO  REGULACION  \
0      25         1      1    H1         1         2         3         2
1      25         2      1    H1         1         2         3         2
2      25         3      1    H1         1         2         3         2
3      25         4      1    H1         1         2         3         2
4      25         5      1    H1         1         2         3         2
```

```
      SEXO  TIPOPAIS  ...  DSIESPA3  SIESPA4  DSIESPA4  RETRINOIN  RETRIIN  \
0      1         1  ...         0         6         0    6128.65     0.0
1      1         1  ...         0         6         0   16509.97     0.0
2      6         1  ...         0         6         0   18284.40     0.0
3      1         1  ...         0         6         0   16549.97     0.0
4      1         1  ...         0         6         0   16554.71     0.0
```

```
      GEXTRA  VESPNOIN  VESPIN  ANOS2  FACTOTAL
0    787.35         0.0     0.0     6     70.48
1   3127.50         0.0     0.0     6     70.48
2   3200.76         0.0     0.0     4     70.48
3   3127.77         0.0     0.0     6     70.48
4   3072.51         0.0     0.0     4     70.48
```

[5 rows x 56 columns]

```
[4]: #mostramos la amplitud del dataset
dfSueldos.shape
```

```
[4]: (216726, 56)
```

Vemos que para nuestro estudio tenemos un amplio dataset, formado por 216726 muestras y 56 características.

## 3 2. Integración y selección de los datos de interés a analizar.

Mostramos a continuación el detalle de todas las características incluidas en el dataset. Acompañamos la documentación con el archivo “dr\_EES\_2018.xlsx” descargado de la página del INE junto al dataset original y que contiene la explicación detallada de cada una de ellas y a que se corresponden los valores categóricos almacenados.

Variable	Descripción
IDENCCC	CÓDIGO DE IDENTIFICACIÓN DEL CENTRO DE COTIZACIÓN
ORDENTRA	NÚMERO DE ORDEN DEL TRABAJADOR
NUTS1	NUTS1
CNACE	CÓDIGO ACTIVIDAD ECONOMICA (CNAE 2009)
ESTRATO2	TAMAÑO DE LA UNIDAD
CONTROL	PROPIEDAD O CONTROL
MERCADO	MERCADO
REGULACION	NORMA DE REGULACIÓN DE LAS RELACIONES LABORALES
SEXO	SEXO
TIPOPAIS	NACIONALIDAD
CNO1	CODIGO DE OCUPACION (GRUPO PRINCIPAL CNO-11)
RESPONSA	RESPONSABILIDAD EN ORGANIZACIÓN Y/O SUPERVISION
ESTU	ESTUDIOS (CODIGO DE LA TITULACION)
ANOANTI	AÑOS DE ANTIGÜEDAD
MESANTI	MESES DE ANTIGÜEDAD
TIPOJOR	TIPO DE JORNADA
TIPOCON	DURACION DEL CONTRATO
FIJODISM	MESES DEL PERÍODO DE TRABAJO DEL TRABAJADOR FIJO DISCONTINUO
FIJODISD	DÍAS DEL PERÍODO DE TRABAJO DEL TRABAJADOR FIJO DISCONTINUO
VAL	DIAS DE VACACIONES ANUALES LABORABLES
VAN	DIAS DE VACACIONES ANUALES NATURALES
PUENTES	FIESTAS NO OFICIALES
JAP	JORNADA ANUAL PACTADA
JSP1	JORNADA SEMANAL PACTADA (HORAS)
JSP2	JORNADA SEMANAL PACTADA (MINUTOS)
HEXTRA	HORAS EXTRAORDINARIAS
DRELABM	DURACIÓN DE LA RELACIÓN LABORAL EN EL MES DE OCTUBRE
SIESPM1	SITUACIÓN ESPECIAL 1 EN EL MES DE OCTUBRE
DSIESPM1	DÍAS EN SITUACIÓN ESPECIAL 1 EN EL MES DE OCTUBRE
SIESPM2	SITUACIÓN ESPECIAL 2 EN EL MES DE OCTUBRE
DSIESPM2	DÍAS EN SITUACIÓN ESPECIAL 2 EN EL MES DE OCTUBRE
SALBASE	SALARIO BASE
EXTRAORM	PAGA EXTRAORDINARIA MENSUAL
PHEXTRA	PAGOS POR HORAS EXTRAORDINARIAS
COMSAL	COMPLEMENTOS SALARIALES
COMSALTT	COMPLEMENTOS SALARIALES POR TURNO DE TRABAJO
IRPFMES	RETENCIONES IRPF EN EL MES DE OCTUBRE
COTIZA	CONTRIBUCIONES A LA SEGURIDAD SOCIAL
BASE	BASE DE COTIZACIÓN A LA SEGURIDAD SOCIAL POR CONTINGENCIAS COMUNES DEL TRABAJADOR
DRELABAM	DURACIÓN DE LA RELACIÓN LABORAL EN EL AÑO (MESES)
DRELABAD	DURACIÓN DE LA RELACIÓN LABORAL EN EL AÑO (DÍAS)
SIESPA1	SITUACIÓN ESPECIAL 1 EN EL AÑO

Variable	Descripción
DSIESPA1	DÍAS EN SITUACIÓN ESPECIAL 1 EN AÑO
SIESPA2	SITUACIÓN ESPECIAL 2 EN EL AÑO
DSIESPA2	DÍAS EN SITUACIÓN ESPECIAL 2 EN AÑO
SIESPA3	SITUACIÓN ESPECIAL 3 EN EL AÑO
DSIESPA3	DÍAS EN SITUACIÓN ESPECIAL 3 EN AÑO
SIESPA4	SITUACIÓN ESPECIAL 4 EN EL AÑO
DSIESPA4	DÍAS EN SITUACIÓN ESPECIAL 4 EN AÑO
RETRINOIN	SALARIO BRUTO ANUAL NO DERIVADO DE IT
RETRIIN	SALARIO BRUTO ANUAL DERIVADO DE IT
GEXTRA	GRATIFICACIONES EXTRAORDINARIAS ABONADAS EN EL AÑO
VESPNOIN	VALORACIÓN EN ESPECIE NO DERIVADA DE IT
VESPIN	VALORACIÓN EN ESPECIE DERIVADA DE IT
ANOS2	EDAD
FACTOTAL	FACTOR DE ELEVACIÓN (12.2)

[5]: *#vemos información de las diferentes variables.*

```
dfSuealdos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 216726 entries, 0 to 216725
Data columns (total 56 columns):
#   Column          Non-Null Count  Dtype
---  -
0   IDENCCC          216726 non-null  int64
1   ORDENTRA         216726 non-null  int64
2   NUTS1            216726 non-null  int64
3   CNACE            216726 non-null  object
4   ESTRATO2         216726 non-null  int64
5   CONTROL          216726 non-null  int64
6   MERCADO          216726 non-null  int64
7   REGULACION       216726 non-null  int64
8   SEXO             216726 non-null  int64
9   TIPOPAIS         216726 non-null  int64
10  CNO1             216726 non-null  object
11  RESPONSA         216726 non-null  int64
12  ESTU             216726 non-null  int64
13  ANOANTI          216726 non-null  int64
14  MESANTI          216726 non-null  int64
15  TIPOJOR          216726 non-null  int64
16  TIPOCON          216726 non-null  int64
17  FIJODISM         216726 non-null  int64
18  FIJODISD         216726 non-null  int64
19  VAL              216726 non-null  int64
20  VAN              216726 non-null  int64
```

```

21 PUENTES      216726 non-null int64
22 JAP          216726 non-null int64
23 JSP1         216726 non-null int64
24 JSP2         216726 non-null int64
25 HEXTRA       216726 non-null int64
26 DRELABM      216726 non-null int64
27 SIESPM1      216726 non-null int64
28 DSIESPM1     216726 non-null int64
29 SIESPM2      216726 non-null int64
30 DSIESPM2     216726 non-null int64
31 SALBASE      216726 non-null float64
32 EXTRAORM     216726 non-null float64
33 PHEXTRA      216726 non-null float64
34 COMSAL       216726 non-null float64
35 COMSALTT     216726 non-null float64
36 IRPFMES      216726 non-null float64
37 COTIZA       216726 non-null float64
38 BASE         216726 non-null float64
39 DRELABAM     216726 non-null int64
40 DRELABAD     216726 non-null int64
41 SIESPA1      216726 non-null int64
42 DSIESPA1     216726 non-null int64
43 SIESPA2      216726 non-null int64
44 DSIESPA2     216726 non-null int64
45 SIESPA3      216726 non-null int64
46 DSIESPA3     216726 non-null int64
47 SIESPA4      216726 non-null int64
48 DSIESPA4     216726 non-null int64
49 RETRINOIN    216726 non-null float64
50 RETRIIN      216726 non-null float64
51 GEXTRA       216726 non-null float64
52 VESPNOIN     216726 non-null float64
53 VESPIN       216726 non-null float64
54 ANOS2        216726 non-null int64
55 FACTOTAL     216726 non-null float64
dtypes: float64(14), int64(40), object(2)
memory usage: 92.6+ MB

```

[6]: *# Echamos un vistazo a los valores únicos existentes en cada variable*

```
dfSueldos.nunique(axis=0)
```

```

[6]: IDENCCC      24710
ORDENTRA      50
NUTS1         7
CNACE         27
ESTRATO2      5

```

CONTROL	2
MERCADO	4
REGULACION	5
SEXO	2
TIPOPAIS	2
CNO1	17
RESPONSA	2
ESTU	7
ANOANTI	53
MESANTI	13
TIPOJOR	2
TIPOCON	2
FIJODISM	11
FIJODISD	30
VAL	65
VAN	76
PUENTES	41
JAP	1824
JSP1	54
JSP2	60
HEXTRA	101
DRELABM	28
SIESPM1	2
DSIESPM1	32
SIESPM2	2
DSIESPM2	19
SALBASE	90367
EXTRAORM	6006
PHEXTRA	7292
COMSAL	97788
COMSALTT	18103
IRPFMES	78825
COTIZA	23862
BASE	124533
DRELABAM	12
DRELABAD	32
SIESPA1	2
DSIESPA1	366
SIESPA2	2
DSIESPA2	291
SIESPA3	2
DSIESPA3	346
SIESPA4	2
DSIESPA4	126
RETRINOIN	204717
RETRIIN	34764
GEXTRA	113146

```
VESPNOIN      28720
VESPIN        525
ANOS2          6
FACTOTAL      9580
dtype: int64
```

Vamos a fijarnos en algunas de las columnas, empezando por NUTS1 que se refiere a la región donde residen las personas del estudio.

Vemos los códigos de las regiones utilizadas

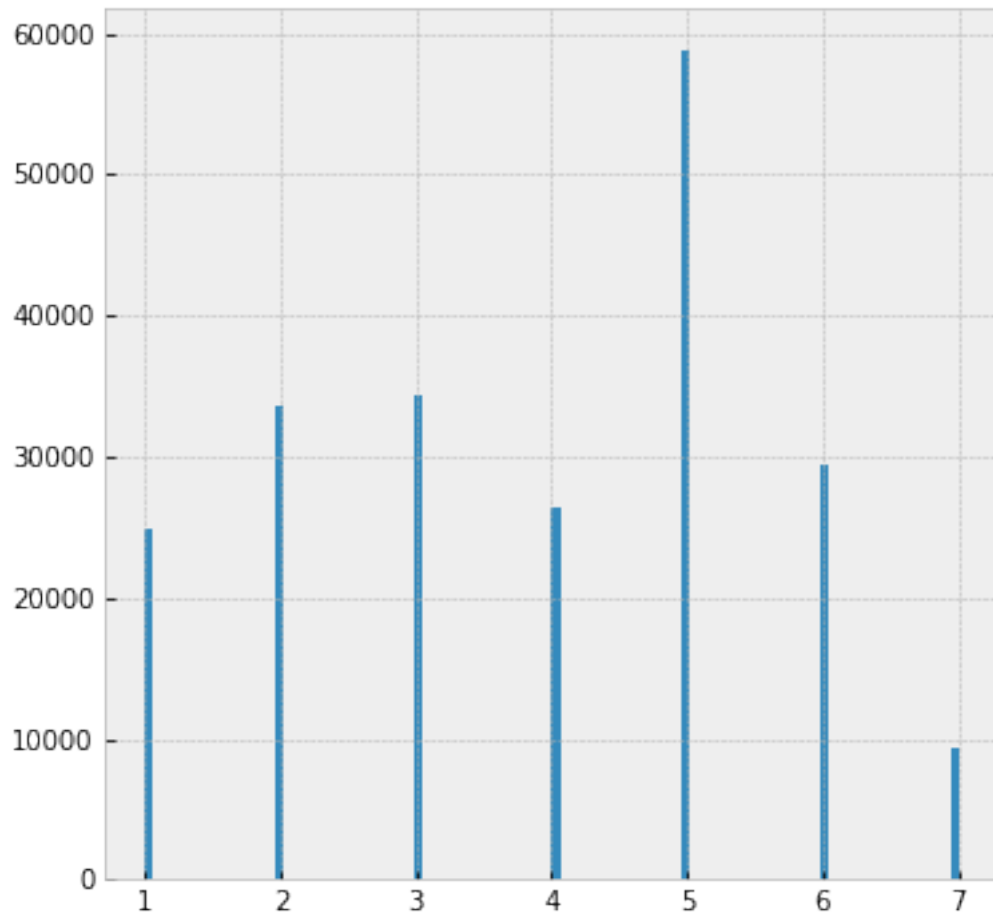
Código	Descripción
1	NOROESTE
2	NORESTE
3	COMUNIDAD DE MADRID
4	CENTRO
5	ESTE
6	SUR
7	CANARIAS

```
[7]: #Siete regiones
      columnName = 'NUTS1'

      dfSueldos[columnName].hist(figsize=(6,6), bins=100)
```

```
[7]: <matplotlib.axes._subplots.AxesSubplot at 0x2773f689188>
```





Vamos a centrarnos en la variable sexo.

Código	Descripción
1	HOMBRE
6	MUJER

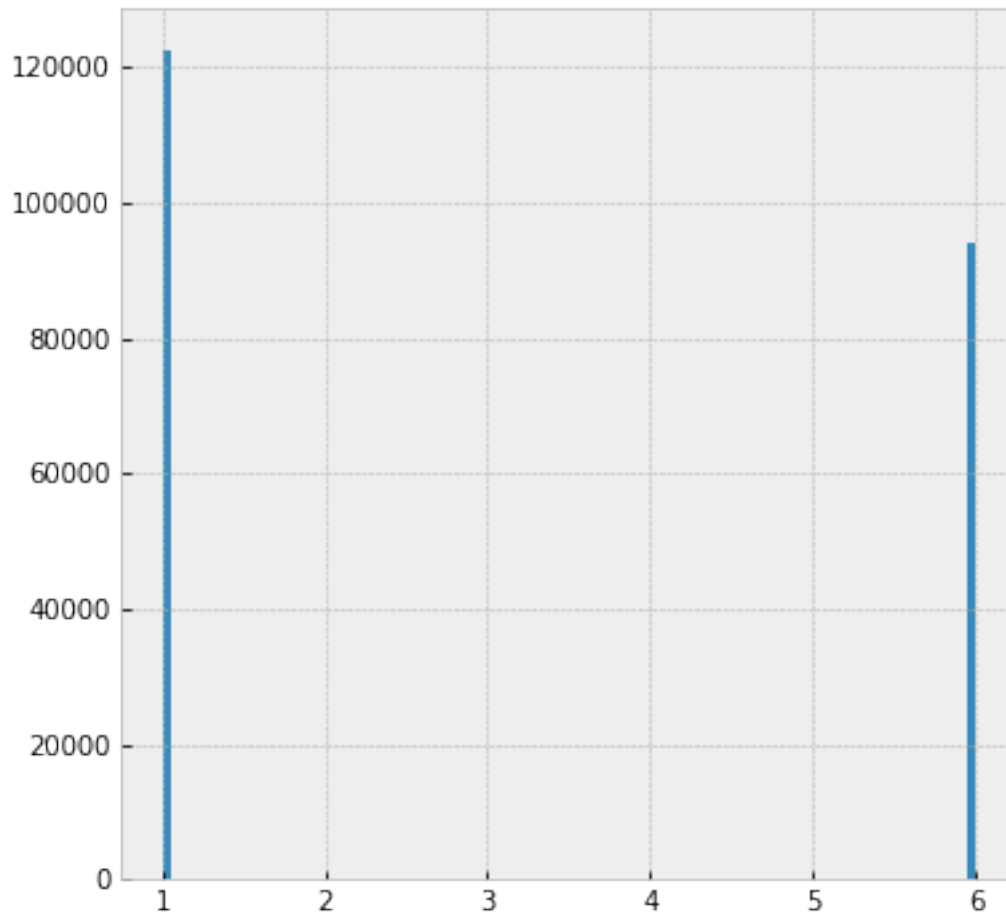
```
[8]: dfSueldos['SEXO'].value_counts()
```

```
[8]: 1    122558
      6     94168
      Name: SEXO, dtype: int64
```

```
[9]: columnName = 'SEXO'

      dfSueldos[columnName].hist(figsize=(6,6), bins=100)
```

```
[9]: <matplotlib.axes._subplots.AxesSubplot at 0x27740bd1ec8>
```



## 4 3. Limpieza de los datos.

### 4.0.1 3.1. ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

```
[10]: #comprobamos cuales son las que tienen menos datos.

df2 = dfSueldos[[column for column in dfSueldos if dfSueldos[column].count() /
↳ len(dfSueldos) <= 0.3]]
df2.columns
```

```
[10]: Index([], dtype='object')
```

Por lo que podemos ver los datos están completos y no encontramos variables con pocos datos registrados.

No hay columnas a las que les falten datos.

No es necesario imputar los datos que faltan.

No hay valores nulos en ninguno de los atributos.

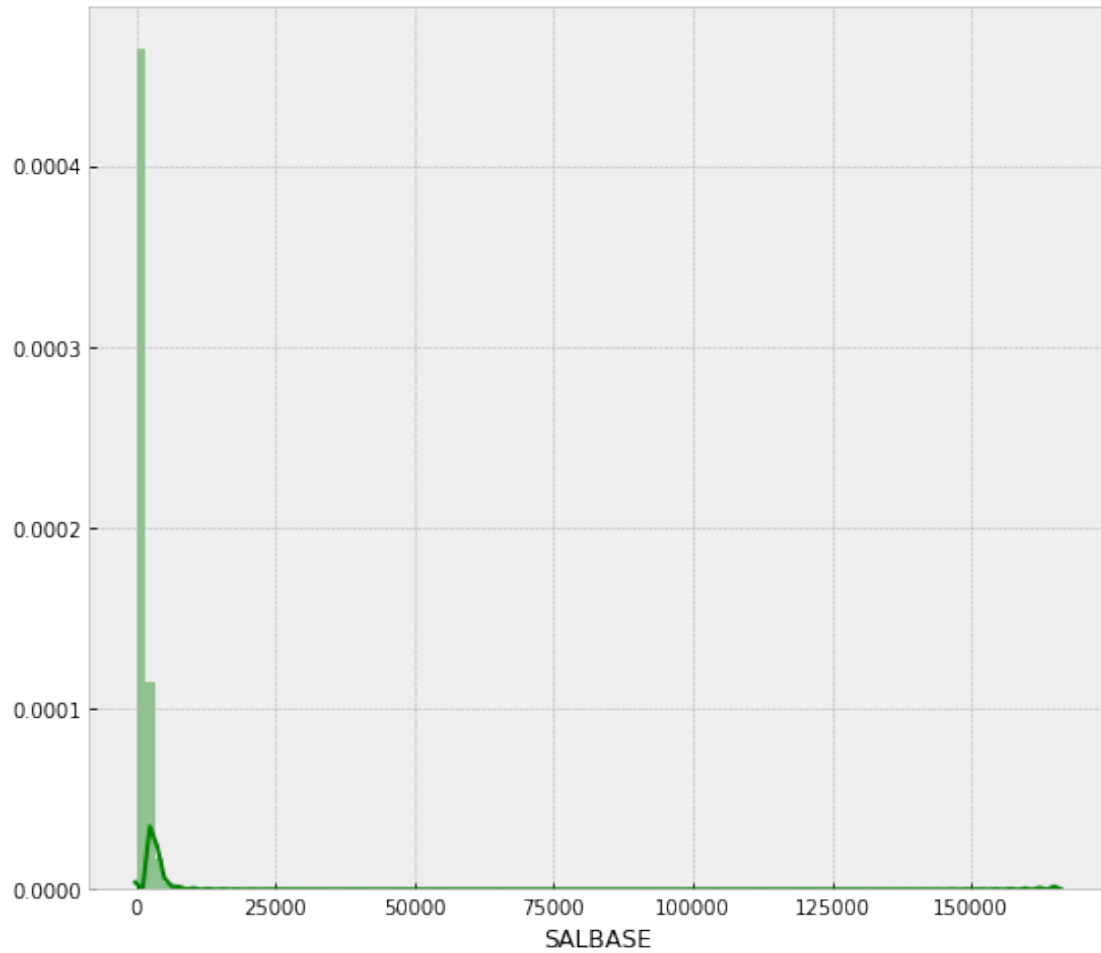
Nota: A veces se utilizan ceros para indicar la ausencia de ciertos valores, sin embargo en el análisis de los datos, los campos y sus dominios, hemos visto que las columnas que tienen ceros, pertenecen al dominio y tiene un significado: por ejemplo, en el campo ESTRATO2, el 0 significa “todos los estratos”.

```
[11]: #realizamos un estudio de nuestra variable objetivo

columnName = 'SALBASE'

print(dfSueldos[columnName].describe())
plt.figure(figsize=(9, 8))
sns.distplot(dfSueldos[columnName], color='g', bins=100, hist_kws={'alpha': 0.
↪4});
```

```
count    216726.000000
mean      1401.945641
std       1359.783859
min         0.000000
25%        900.000000
50%       1174.285000
75%       1602.175000
max      166084.020000
Name: SALBASE, dtype: float64
```



```
[12]: #vamos a ver la distribución de los datos de Salario base  
plt.figure(figsize = (12, 6))  
ax = sns.boxplot(y='SALBASE', data=dfSueldos)  
plt.setp(ax.artists, alpha=.5, linewidth=2, edgecolor="k")  
plt.xticks(rotation=45)
```

```
[12]: (array([0]), <a list of 1 Text xticklabel objects>)
```



```
[13]: # Vemos que hay muchos valores por encima del limite superior de la
      ↪ distribución de salarios
      # lo que provoca dificultades en la visualización.
      # Vamos a eliminar algunos de los registros para visualizar mejor la
      ↪ distribución de los datos
      # luego resolveremos si debemos considerar estos datos como outliers

      #en primer lugar calcularemos los percentiles de la distribución de los valores
      ↪ de SALBASE
      arr = dfSueldos[columnName]

      print("percentil 5 : ",
            np.percentile(arr, 5))

      print("percentil 50 : ",
            np.percentile(arr, 50))

      print("percentil 95 : ",
            np.percentile(arr, 95))
```

```
percentil 5 : 427.94
percentil 50 : 1174.2849999999999
percentil 95 : 3013.265
```

#### 4.0.2 3.2. Identificación y tratamiento de valores extremos.

```
[14]: #Eliminaremos los outliers (valores que se alejan tres veces la desviación
      ↪ estándar)

      # Para el salario:
```

```

mean= dfSueldos['SALBASE'].mean()
std= dfSueldos['SALBASE'].std()
limit = mean + 3*std

dfSueldos[(dfSueldos.SALBASE > limit)]
#Hay 1668 personas cuyo sueldo es un valor extremo

#Eliminamos estos Outliers:
dfSueldos = dfSueldos[(dfSueldos.SALBASE < limit)]

```

```

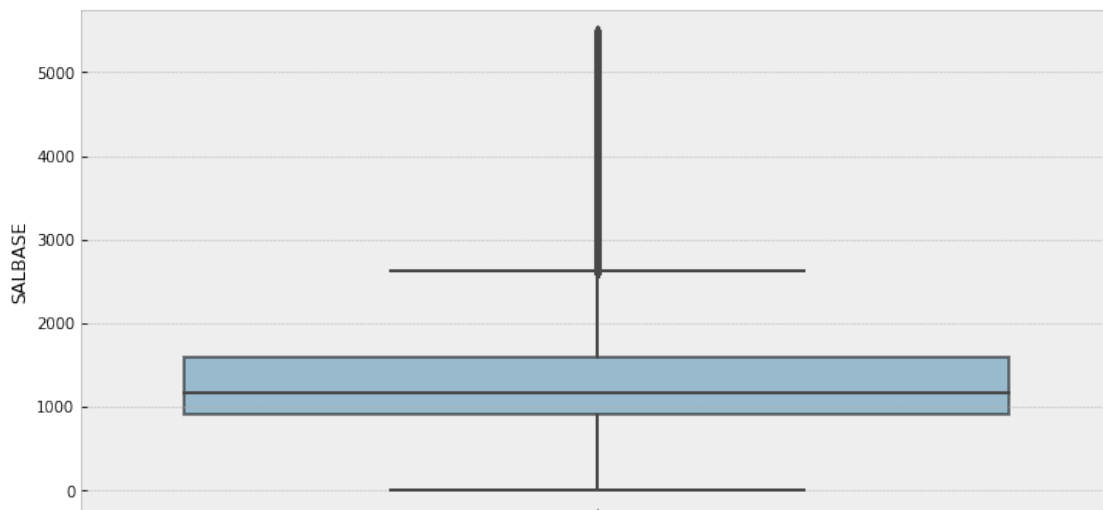
[15]: plt.figure(figsize = (12, 6))
      ax = sns.boxplot(y='SALBASE', data=dfSueldos)
      plt.setp(ax.artists, alpha=.5, linewidth=2, edgecolor="k")
      plt.xticks(rotation=45)

```

```

[15]: (array([0]), <a list of 1 Text xticklabel objects>)

```



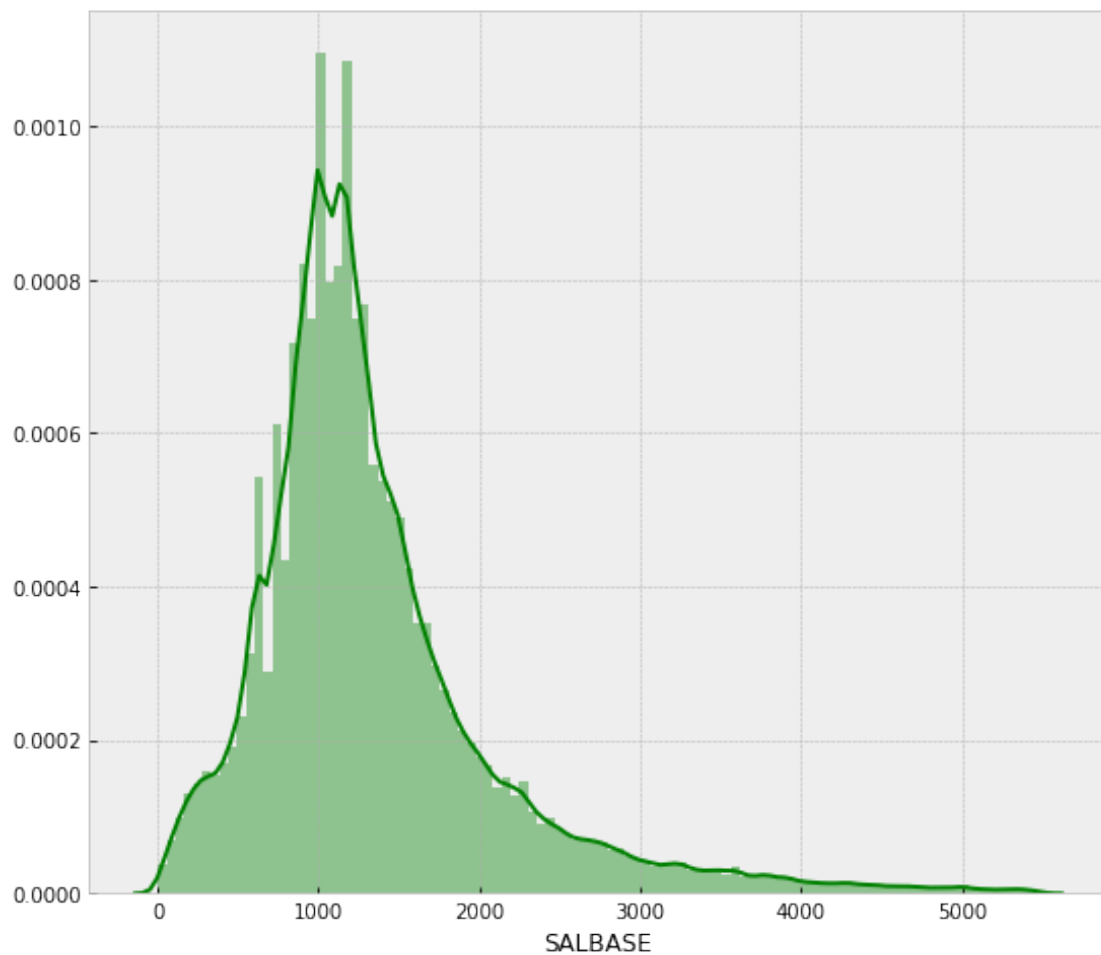
```

[16]: #Visualizamos mejor la distribución del Salario base

      columnName = 'SALBASE'

      plt.figure(figsize=(9, 8))
      sns.distplot(dfSueldos[columnName], color='g', bins=100, hist_kws={'alpha': 0.
      ↪4}, hist=True);

```



```
[17]: df_num = dfSueldos.select_dtypes(include = ['float64', 'int64'])
df_num.head()
```

```
[17]:
```

	IDENCCC	ORDENTRA	NUTS1	ESTRATO2	CONTROL	MERCADO	REGULACION	SEXO	\
0	25	1	1	1	2	3	2	1	
1	25	2	1	1	2	3	2	1	
2	25	3	1	1	2	3	2	6	
3	25	4	1	1	2	3	2	1	
4	25	5	1	1	2	3	2	1	

	TIPOPAIS	RESPONSA	...	DSIESPA3	SIESPA4	DSIESPA4	RETRINOIN	RETRIIN	\
0	1	0	...	0	6	0	6128.65	0.0	
1	1	0	...	0	6	0	16509.97	0.0	
2	1	0	...	0	6	0	18284.40	0.0	
3	1	0	...	0	6	0	16549.97	0.0	
4	1	0	...	0	6	0	16554.71	0.0	

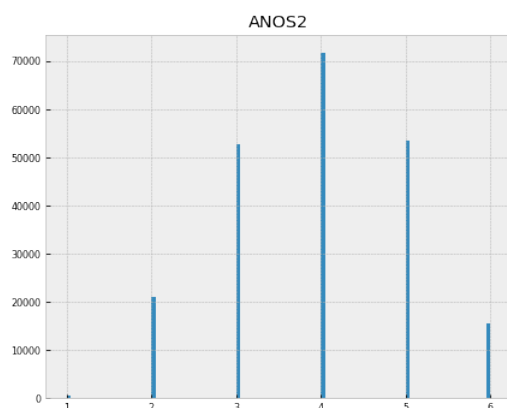
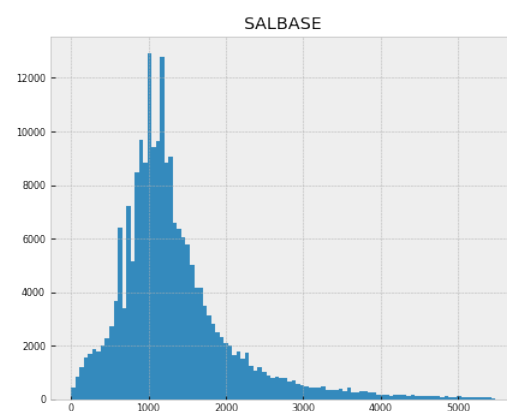
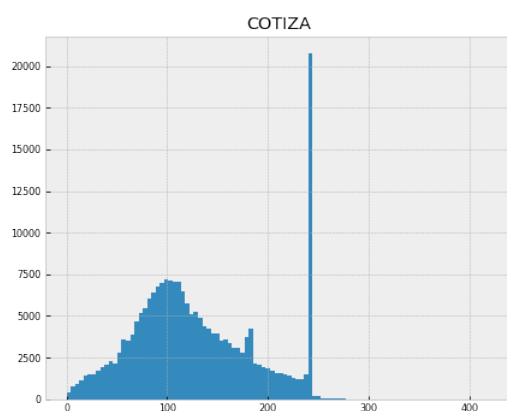
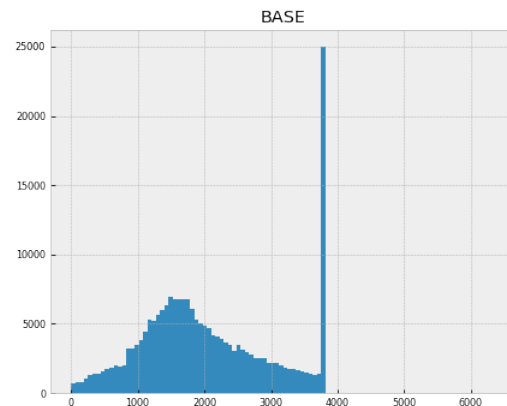
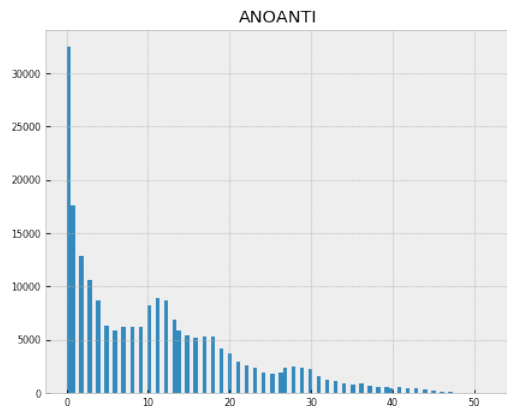
	GEXTRA	VESPNOIN	VESPIN	ANOS2	FACTOTAL
0	787.35	0.0	0.0	6	70.48
1	3127.50	0.0	0.0	6	70.48
2	3200.76	0.0	0.0	4	70.48
3	3127.77	0.0	0.0	6	70.48
4	3072.51	0.0	0.0	4	70.48

[5 rows x 54 columns]

```
[18]: numericColumns = ['ANOANTI', 'BASE', 'COTIZA', 'SALBASE', 'ANOS2']
```

```
dfSueudos.hist(figsize=(16,20), bins=100, xlabelsize=8,
→ylabelsize=8, column=numericColumns);
```





La base de cotización máxima para la seguridad social en España, en 2018, era de 3751€. Por eso, aunque quitemos los outliers de salario, todas las personas de la muestra que cobren más de esa cantidad, como no pagan más de base, aparecen en esa columna en el histograma.

De igual forma ocurre con la característica que indica las contribuciones a la seguridad social, que son un porcentaje fijo de esa base de cotización que hemos visto. Así que exactamente la misma columna aparece en esta gráfica, ya que es una variable totalmente dependiente.

```
[19]: dfSueldos_corr = dfSueldos.corr()['SALBASE']
      corr_features_list = dfSueldos_corr.sort_values(ascending=False)

      print("Mostramos la correlación de las variables:")
      print(corr_features_list[1:])
```

Mostramos la correlación de las variables:

COTIZA	0.682759
BASE	0.665110
RETRINOIN	0.510249
IRPFMES	0.491566
JSP1	0.399297
JAP	0.373218
ESTU	0.350257
RESPONSA	0.256395
MERCADO	0.217328
GEXTRA	0.216637
VAL	0.203136
ANOANTI	0.180501
RETRIIN	0.154888
DRELABAM	0.140991
VESPN0IN	0.134489
DSIESPM1	0.128118
ESTRATO2	0.092321
PUENTES	0.072012
DSIESPA1	0.069142
COMSAL	0.064092
ORDENTRA	0.063910
ANOS2	0.049008
SIESPA3	0.045717
EXTRAORM	0.042577
DSIESPA2	0.031286
DRELABM	0.022723
REGULACION	0.022318
MESANTI	0.022254
COMSALTT	0.019564
VESPIN	0.017308
CONTROL	0.013837
JSP2	0.008450
PHEXTRA	0.000273
DSIESPA4	-0.000563
IDENCCC	-0.001050
DSIESPM2	-0.001874
SIESPM2	-0.007484
HEXTRA	-0.023563
SIESPA4	-0.029745
TIPOPAIS	-0.033520

DSIESPA3	-0.049702
SIESPA2	-0.050428
NUTS1	-0.052568
FIJODISD	-0.061512
SIESPA1	-0.061513
FIJODISM	-0.085227
FACTOTAL	-0.089833
DRELABAD	-0.096994
SEXO	-0.138331
SIESPM1	-0.158128
VAN	-0.165431
TIPOCON	-0.191416
TIPOJOR	-0.387813

Name: SALBASE, dtype: float64

Hay variables que evidentemente tienen mucha correlación con el sueldo, básicamente porque son un cálculo de este. Aunque no sean 100% dependientes ya que la normativa española tiene excepciones, es obvio que la base de cotización, el IRPF, la cantidad aportada la seguridad social, el salario bruto anual, etc muestran mucha correlación.

En realidad no es que con estas variables podamos “predecir” el salario, es que estas variables se extraen del salario a través de cálculos (combinando con otros factores).

Por tanto, a la hora de elegir variables que nos ayuden a hacer una predicción, podemos apoyarnos en variables como “jornada semanal pactada” (es de sentido común que si trabajas más horas a la semana, cobrarás más que si tienes un trabajo de media jorana). También dejamos variables como responsabilidad y estudios, porque queremos estudiar cuánto influye el nivel de estudios o la responsabilidad de una persona en el salario final.

Pero debemos eliminar variables que son un cálculo directo del salario: base, cotización, sueldo bruto, IRPF, etc...

El siguiente apartado es donde realizamos la selección de los datos de interés.

#### 4.1 Integración y selección de los datos de interés a analizar.

Hay determinados campos que no son de interés para el estudio que se quiere realizar:

Quitamos: - IDENCCC (0): Este código de identificación del centro de cotización no nos permite identificar ninguna información relevante sobre el trabajador. - ORDENTRA (1): No es - RESPONSA (11): Porque con el campo código de ocupación tenemos información más precisa de la responsabilidad del trabajador en su empresa. - FIJODISM (17), FIJODISD (18): El hecho de que sean fijos discontinuos no aportan más información que el número de días (jornada anual pactada) que trabaja. - VAL (19), VAN (20), PUENTES (21): Las vacaciones pagadas son beneficios que aporta la empresa que no están relacionados con el salario.

- (Campos calculados) IRPFMES (36), COTIZA (37), BASE (38), RETRINOIN : Los impuestos que cada persona paga dependen de su salario, combinados con sus circunstancias personales. Toda la información necesaria para nuestro estudio lo obtendremos del salario, no de su IRPF/cotizaciones. RETRINOIN es un cálculo a partir del salario (o al revés).
- No consideraremos los días en situaciones especiales, ya que se podrían considerar outliers.

Situaciones especiales, o extraordinarias, no son la norma y son ruido en nuestra muestra. SIESPM1, DSIESPM1, SIESPM2, DSIESPM2 SIESPA1, DSIESPA1, SIESPA2, DSIESPA2 SIESPA3, DSIESPA3, SIESPA4, DSIESPA4. Tampoco son campos que el trabajador “elija”: si hay una incapacidad, una baja o algo así es un imprevisto contral el que no podemos hacer nada.

- RETRIIN: Queda fuera del ámbito del estudio los salarios brutos derivados de las incapacidades temporales.
- FACTOTAL: Es una variable estadística que no aporta nada en nuestro estudio.

```
[20]: dfSuealdos = dfSuealdos.drop(["IDENCCC", "ORDENTRA", "RESPONSA", "FIJODISM",
    ↪ "FIJODISD", "VAL", "VAN", "PUENTES", "IRPFMES", "COTIZA", "BASE",
    ## Integración y selección de los datos de interés a analizar.
    "SIESPM1", "DSIESPM1", "SIESPM2", "DSIESPM2", "SIESPA1", "DSIESPA1", "SIESPA2",
    ↪ "DSIESPA2", "SIESPA3", "DSIESPA3", "SIESPA4", "DSIESPA4", "RETRINOIN",
    ↪ "RETRIIN", "FACTOTAL"], axis=1)
```

## 4.2 COLUMNAS CALCULADAS

Cálculo de columnas

- Meses de antigüedad (tenemos dos campos para la antigüedad/experiencia-cotizado, años y meses, vamos a simplificarlo dejando sólo uno)
- Precio/hora (simplificamos el dataset, en lugar de considerar jornadas semanales pactadas y salario, unificamos el precio hora)
- Unificar Jornada semanal pactada (a horas)
- Salario: simplificamos salario, pagas extras, y decidiremos qué complementos consideramos “salario” y cuáles no. (por ejemplo: la cesta de navidad es un beneficio/complemento, pero ... ¿debería ser considerado salario para nuestro estudio?)

**Cálculo** Tenemos una columna “años de antigüedad” y otra “meses de antigüedad”. Vamos a pasarlo todo a meses, lo guardamos en la columna ‘MESANTI’ y borramos la columna años.

```
[21]: dfSuealdos['MESANTI'] = dfSuealdos['MESANTI'] + dfSuealdos['ANOANTI'] * 12
dfSuealdos = dfSuealdos.drop(["ANOANTI"], axis=1)
```

Cálculo Tenemos una columna “JORNADA SEMANAL PACTADA (HORAS)” y otra para los minutos. Vamos a pasarlo todo a horas, lo guardamos en la columna JSP1 y borramos la columna de minutos.

```
[22]: dfSuealdos['JSP1'] = dfSuealdos['JSP1'] + dfSuealdos['JSP2']/60
dfSuealdos = dfSuealdos.drop(["JSP2"], axis=1)
```

Estudio de las columnas asociadas a salario y complementos:

```
[23]: print("Salario base")
print(dfSuealdos['SALBASE'].describe(), "\n")
```

```

print("Pago extra")
print(dfSueldos['EXTRAORM'].describe(), "\n")

print("Pago por horas extras")
print(dfSueldos['PHEXTRA'].describe(), "\n" )

print("Complementos salariales")
print(dfSueldos['COMSAL'].describe(), "\n")

print("COMPLEMENTOS SALARIALES POR TURNO DE TRABAJO")
print(dfSueldos['COMSALTT'].describe())

```

Salario base

```

count    215058.000000
mean      1342.060956
std       763.257873
min        0.000000
25%       897.555000
50%      1169.090000
75%      1586.210000
max       5481.180000
Name: SALBASE, dtype: float64

```

Paga extra

```

count    215058.000000
mean       45.956117
std       353.679755
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max      54693.930000
Name: EXTRAORM, dtype: float64

```

Pago por horas extras

```

count    215058.000000
mean      10.617753
std       84.837671
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max      11556.810000
Name: PHEXTRA, dtype: float64

```

Complementos salariales

```

count    215058.000000
mean      629.140661

```

```
std      1082.113730
min       0.000000
25%      24.150000
50%     294.965000
75%     849.737500
max     83468.990000
Name: COMSAL, dtype: float64
```

#### COMPLEMENTOS SALARIALES POR TURNO DE TRABAJO

```
count    215058.000000
mean      35.853156
std      168.708559
min       0.000000
25%      0.000000
50%      0.000000
75%      0.000000
max      8513.000000
Name: COMSALTT, dtype: float64
```

#### Propuesta:

Sumamos el salario base + paga extra, para igualar las personas con 14 pagas o 12.

NO Añadimos los complementos salariales, ni valoraciones en especie, que forman parte del salario (coche, tickets restaurant, plan de pensiones, etc...), porque aunque se pueda pagar IRPF por eso, no es dinero “líquido” o “neto” que estamos recibiendo en la nómina. Ni tampoco los complementos salariales por turno de trabajo. (borramos las columnas)

NO añadimos el pago por horas extras, ya que el cálculo lo haremos con la “jornada semanal pactada” - para ver cuando cobran la hora. Si añadimos las horas extras, estamos “ruido” en el cálculo. Además, las horas extras no las cobra todo el mundo, no es un complemento en el sueldo estable, y lo mismo pasa con las gratificaciones extraordinarias (borramos la columna)

```
[24]: dfSueldos['SALARIO'] = dfSueldos['SALBASE']+dfSueldos['EXTRAORM']
```

Quitamos los outliers de “salario”. Estos datos desvirtúan el estudio.

```
[25]: # Quitar outliers de salario

# Para el salario:

mean= dfSueldos['SALARIO'].mean()
std= dfSueldos['SALARIO'].std()
limit = mean + 3*std

dfSueldos[(dfSueldos.SALARIO > limit)]
#Hay 4095 personas cuyo sueldo es un valor extremo

#Eliminamos estos Outliers:
```

```
dfSueldos = dfSueldos[(dfSueldos.SALARIO < limit)]
```

```
[26]: # En realidad para calcular el precio /hora que cobran los trabajadores, las
      ↪ columnas que nos interesan son el salario y la duración de la jornada
      ↪ laboral semanal pactada.
      # vamos a obviar el dato de cuántos meses/días trabajan al año.
      # Borramos las columnas DRELABAM y DRELABAD

dfSueldos = dfSueldos.drop(["DRELABAM", "DRELABAD"], axis=1)

# variables, bonos, y extras no "estables" también las desechamos para el
      ↪ estudio.
# aunque estas columnas sí dependan del trabajador, no es algo que tienen todas
      ↪ las empresas.
# igual que las valoraciones en especie.

dfSueldos = dfSueldos.drop(["GEXTRA", "VESPNOIN", "VESPIN"], axis=1)
```

### Cálculo del precio/hora:

horas trabajadas al mes = (jornada semanal pactada en horas \*4) salario / horas trabajadas al mes

```
[27]: dfSueldos['PRECIOHORA'] = dfSueldos['SALARIO'] / (dfSueldos['JSP1']*4)
```

Después de esta limpieza, volvamos a mirar las variables/correlación.

```
[28]: #Quito las columnas calculadas, o que dependen del salario y que no ayudarán a
      ↪ hacer la predicción.
dfSueldosReduced = dfSueldos.drop(["SALARIO", "SALBASE"], axis=1)
```

```
[29]: dfSueldos_corr = dfSueldosReduced.corr()['PRECIOHORA']
      #golden_features_list = dfSueldos_corr[abs(dfSueldosd_corr) > 0.5].
      ↪ sort_values(ascending=False)
golden_features_list = dfSueldos_corr.sort_values(ascending=False)
print("Mostramos la correlación de las variables:")
print(golden_features_list[1:])
```

Mostramos la correlación de las variables:

ESTU	0.297295
EXTRAORM	0.254397
MESANTI	0.180802
MERCADO	0.147426
ESTRATO2	0.085332
ANOS2	0.072171
DRELABM	0.019568
COMSAL	0.009294
COMSALTT	0.008489
REGULACION	0.007734
CONTROL	0.003143

```

PHEXTRA      -0.011482
HEXTRA       -0.034622
TIPOPAIS     -0.038410
TIPOJOR      -0.042955
JSP1         -0.046968
NUTS1        -0.051016
JAP          -0.055524
SEXO         -0.073423
TIPOCON      -0.080352
Name: PRECIOHORA, dtype: float64

```

Por una parte, teniendo en cuenta que tenemos variables que son categóricas, por ejemplo el nivel de estudios, no podemos esperar una correlación muy alta, ya que el nivel de estudio están enumerado del 1 al 7, siendo 1 menos que estudios básicos obligatorios, y 7 unos estudios de alto nivel: licenciados y doctorados. Pero aunque sea una variable categórica, estamos dándole un número al nivel de estudios. El hecho de que salga que la variable con más correlación del dataset es el nivel de estudios del trabajador nos parece significativo y coherente.

Después la experiencia del trabajador (en meses) y un poco menos la edad (aunque la experiencia y la edad suelen ir relacionadas). De las variables que aparecen con cierta correlación son: el tipo de contrato (los trabajadores indefinidos cobran un poco más), el sexo también parece influir (el general, las mujeres cobran menos), o la región de la empresa (tal y como está el código numérico: cuanto más al sur esté la región, menos sueldo se percibe).

Otras variables que “pertenecen a la empresa” y que influyen: los saldos son más altos en empresas más grandes, con mercados mayores (mejor pagados los trabajadores de empresas internacionales, que los de comercio local). Pero por ejemplo, no parece que afecte si el control de la empresa es público o privado.

## 5 4. Análisis de los datos.

### 5.1 4.1 Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

Del dataset nos interesa hacer diferentes análisis en función de diferentes subconjuntos de datos:

- Edad (ANOS2)
- Experiencia (ANOANTI)
- Sexo (SEXO)
- Industria/tipo de actividad (CNACE)
- Españoles o migrantes (TIPOPAIS)
- Estudios (ESTU)
- Código de ocupación.
- Región (NUTS1)

¿Son los trabajadores con más años y/o experiencia los que más cobran? ¿Cobran menos las mujeres? ¿Cobran menos las mujeres en el mismo tipo de trabajo que los hombres? ¿Cuál es la industria mejor pagada? ¿Y la peor? ¿Cobran más los trabajadores con más estudios? ¿Cobran más los trabajadores con mayor responsabilidad /código de ocupación?



## 5.2 4.2 Comprobación de la normalidad y homogeneidad de la varianza.

Veamos la distribución de la varianza en las siguientes variables numéricas: Aunque tengamos algunas variables cualitativas cuyo código es numérico, y por ejemplo, en el análisis hemos podido sacar algunas conclusiones, como no hablamos de variables continuas, no las incluyo en este apartado.

- Meses de antigüedad (-> MESANTI)
- Jornada semanal (en horas) (-> JSP1)
- Precio /hora (-> PRECIOHORA)
- Salario (-> SALARIO)

Como hemos visto en teoría, las pruebas para la comprobación de la normalidad más habituales son Shapiro-Wilk y Kolmogorov-Smirnov.

Existe también otra prueba llamada de “Anderson-Darling” que comprueba si sigue otra distribución particular (no sólo la normal), como exponencia, logística o Gumbel.

En ellas asumiremos como hipótesis nula que la población sigue una distribución normal. Después de aplicar estas pruebas, si el pvalor obtenido es inferior al nivel de significancia (normalmente = 0,05) entonces se rechaza la hipótesis nula (y por tanto se concluye que los datos no vienen de una distribución normal). En cambio, si el p-valor es superior al nivel de significancia, entonces no se puede rechazar la hipótesis nula y se asume que los datos siguen una distribución normal.

Para aplciar estas pruebas en python hemos recurrido a la siguiente documentación:

Test de Shapiro-wilk ref: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html>

Test de Anderson-Darling ref: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.anderson.html>

Test de Kolmogorov-Smirnov ref: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest.htm>

```
[30]: varcontinuas = dfSueldos[['MESANTI', 'JSP1', 'PRECIOHORA', 'SALARIO']]
      varcontinuas.head()

      for column in varcontinuas:

          # Test de Shapiro-Wilk
          shapiro_test = stats.shapiro(dfSueldos[column] )
          print(column,": " , shapiro_test)

          # Test de Anderson-Darling
          anderson_test = stats.anderson(dfSueldos[column], dist='norm')
          print(column,": " , anderson_test)

          # Test Kolmogorov-Smirnov
          kolmo_test = stats.kstest(dfSueldos[column], 'norm')
          print(column,": " , kolmo_test)

      print("\n")
```

```
MESANTI : (0.8888380527496338, 0.0)
```

```
MESANTI : AndersonResult(statistic=6550.9104059937235,
critical_values=array([0.576, 0.656, 0.787, 0.918, 1.092])),
```

```
significance_level=array([15. , 10. , 5. , 2.5, 1. ]))
MESANTI : KstestResult(statistic=0.9661467720005557, pvalue=0.0)
```

```
JSP1 : (0.5738488435745239, 0.0)
JSP1 : AndersonResult(statistic=37188.029671572905,
critical_values=array([0.576, 0.656, 0.787, 0.918, 1.092]),
significance_level=array([15. , 10. , 5. , 2.5, 1. ]))
JSP1 : KstestResult(statistic=0.9948579678026631, pvalue=0.0)
```

```
PRECIOHORA : (0.757660984992981, 0.0)
PRECIOHORA : AndersonResult(statistic=9971.082675367012,
critical_values=array([0.576, 0.656, 0.787, 0.918, 1.092]),
significance_level=array([15. , 10. , 5. , 2.5, 1. ]))
PRECIOHORA : KstestResult(statistic=0.9957514630494201, pvalue=0.0)
```

```
SALARIO : (0.9199879765510559, 0.0)
SALARIO : AndersonResult(statistic=4827.977608444839,
critical_values=array([0.576, 0.656, 0.787, 0.918, 1.092]),
significance_level=array([15. , 10. , 5. , 2.5, 1. ]))
SALARIO : KstestResult(statistic=0.9998672288309095, pvalue=0.0)
```

El test de **Shapiro-Wilk**, comprueba la hipótesis nula que los datos dados siguen una distribución normal. Para que la hipótesis no sea rechazada, p-value debe tener un valor mayor a 0.05. En el caso del test de **Anderson-Darling**: buscamos que los “critical\_values” sean mayores que los valores que da el array “significance\_level”, para que se cumpla la hipótesis nula. El test de **Kolmogorov-Smirnov**, cuando le damos sólo una distribución, compara este conjunto con otra distribución normal aleatoria. El valor de pvalue debe ser mayor a 0.05 para “no rechazarse” la hipótesis nula (es decir, que la muestra dada siga una distribución normal).

Con estos resultados se puede decir que ninguna de las 3 variables sigue una distribución normal, en todos los casos el p-value ha sido inferior a 0.05 y por tanto se han rechazado en los tres test la hipótesis nula.

Veamos gráficamente la distribución de cada una de las variables, a través de su histograma y de su curva de densidad.

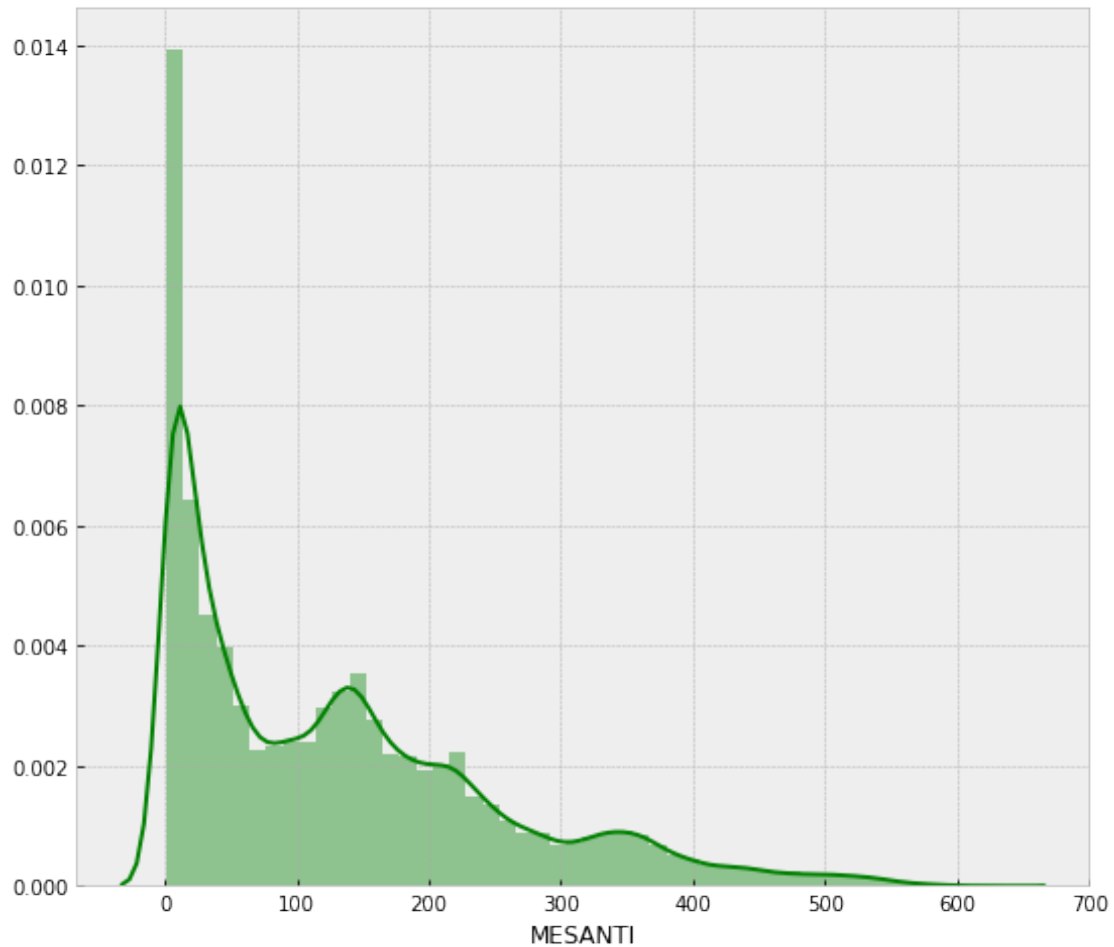
```
[31]: columnName = 'MESANTI'
print(dfSueldos[columnName].describe())
plt.figure(figsize=(9, 8))
sns.distplot(dfSueldos[columnName], color='g', bins=50, hist_kws={'alpha': 0.
→4});
```

```
count    210963.000000
mean      130.807715
std       121.455341
```

```

min          1.000000
25%          25.000000
50%         106.000000
75%         198.000000
max          632.000000
Name: MESANTI, dtype: float64

```



```

[32]: columnName = 'JSP1'
      print(dfSueldos[columnName].describe())
      plt.figure(figsize=(9, 8))
      sns.distplot(dfSueldos[columnName], color='g', bins=50, hist_kws={'alpha': 0.
      ↪4});

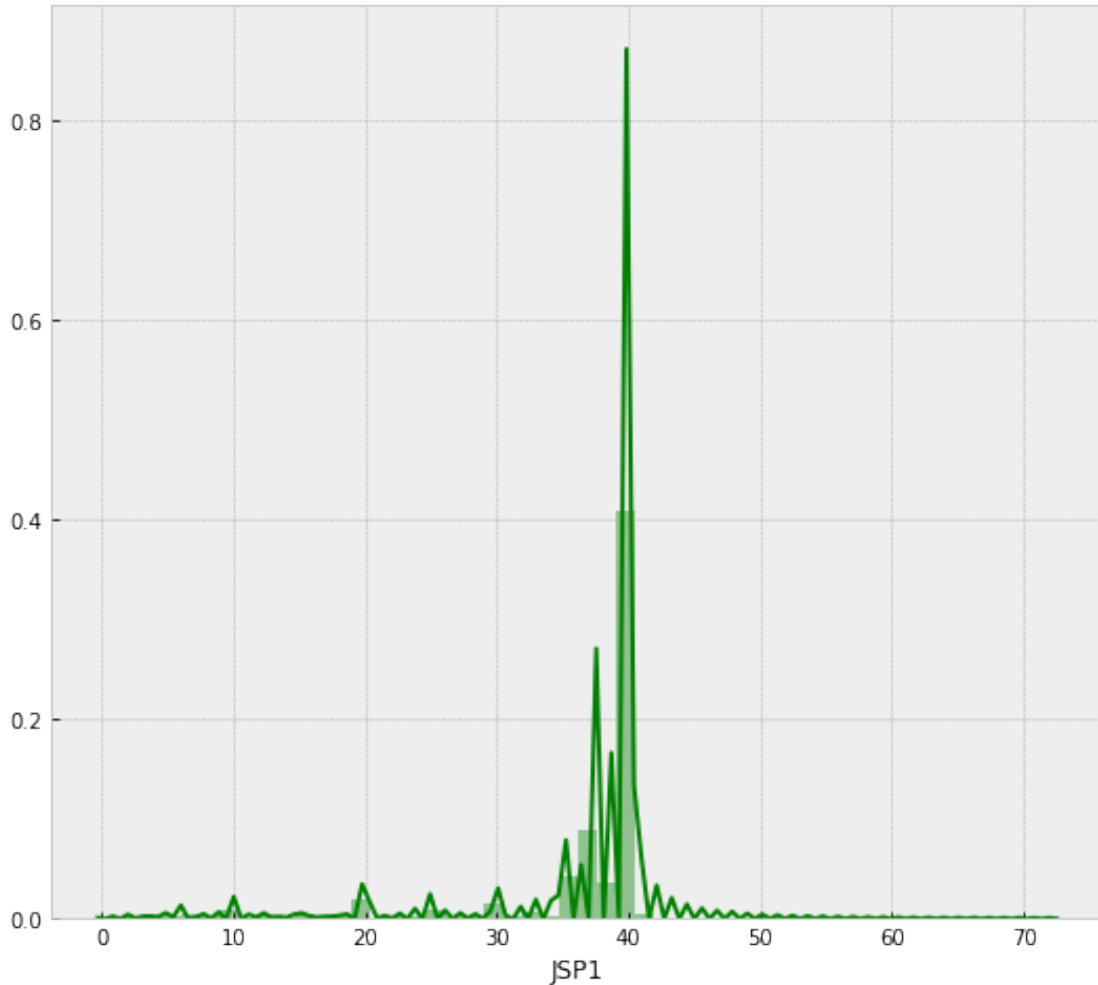
```

```

count      210963.000000
mean        35.930073
std         8.207350
min         0.250000
25%         37.500000

```

```
50%          40.000000
75%          40.000000
max           72.000000
Name: JSP1, dtype: float64
```

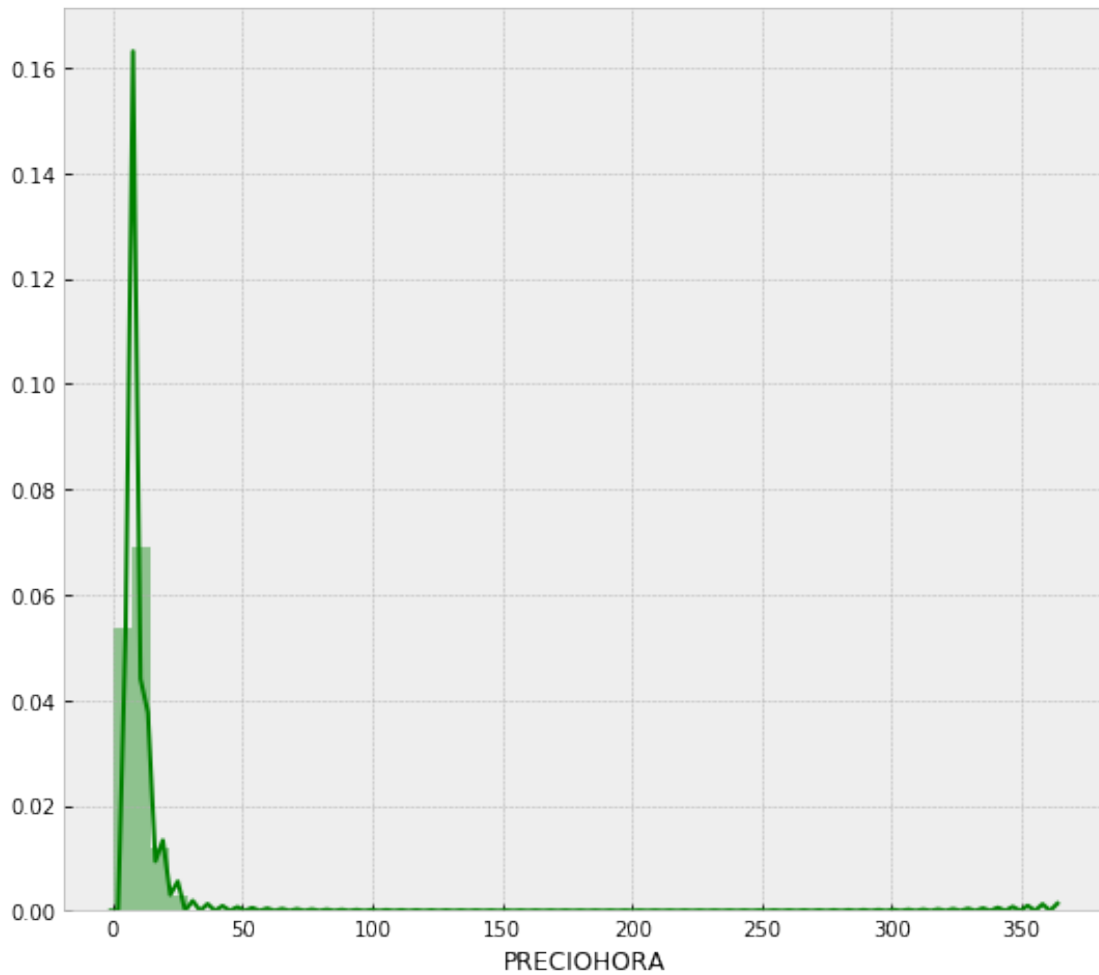


Legalmente el número máximo de horas (incluyendo las extra) que se pueden hacer a la semana son 80. Encontramos un máximo de 72 horas. Como era de esperar, la gran mayoría de los trabajadores tienen una jornada de 40 horas a la semana. Aunque encontramos casi un 10% en 35 horas, y también algunos (pero menos del 5%) en la media jornada (20 horas/semanales.)

Pero efectivamente, tal y como afirmaban los test anteriores, ninguna sigue una distribución normal.

```
[33]: columnName = 'PRECIOHORA'
      print(dfSueldos[columnName].describe())
      plt.figure(figsize=(9, 8))
      sns.distplot(dfSueldos[columnName], color='g', bins=50, hist_kws={'alpha': 0.
      ↪4});
```

```
count    210963.000000
mean       9.238699
std        4.697819
min         0.000000
25%        6.416563
50%        7.929038
75%       10.693812
max       362.962500
Name: PRECIOHORA, dtype: float64
```



Los outliers en este caso desvirtúan la gráfica muchísimo. Tenemos una varianza de 4.69, una media de 9.27 y un percentil 75% de 10 €/h. Sin embargo el máximo lo encontramos en 350€/h. En vista a esta gráfica, también vamos a quitar los outliers a esta columna, porque no es significativa los trabajadores que cobran tantísimo la hora.

```
[34]: # Quitar outliers de precio/hora.
```

```

mean= dfSueldos['PRECIOHORA'].mean()
std= dfSueldos['PRECIOHORA'].std()
limit = mean + 3*std

print(limit)

#Eliminamos estos Outliers:
dfSueldos = dfSueldos[(dfSueldos.PRECIOHORA < limit)]

#Hay 2971 personas cuyo precio hora está por encima de limite que hemos
→ establecido (23€/h)

```

23.33215531060908

```

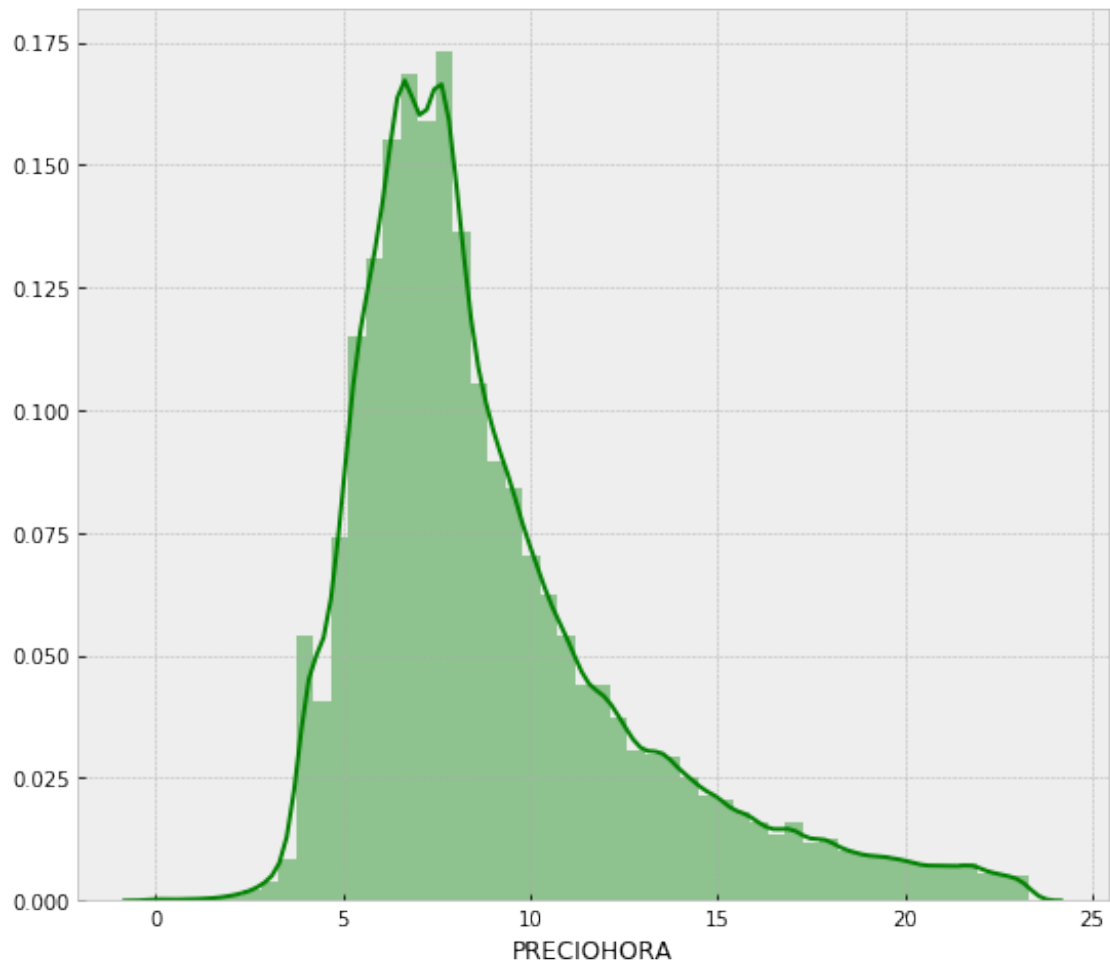
[35]: #y dibujamos la gráfica de nuevo:
columnName = 'PRECIOHORA'
print(dfSueldos[columnName].describe())
plt.figure(figsize=(9, 8))
sns.distplot(dfSueldos[columnName], color='g', bins=50, hist_kws={'alpha': 0.
→ 4});

```

```

count    207992.000000
mean      8.965949
std       3.822664
min       0.000000
25%       6.392250
50%       7.880760
75%      10.511375
max       23.329071
Name: PRECIOHORA, dtype: float64

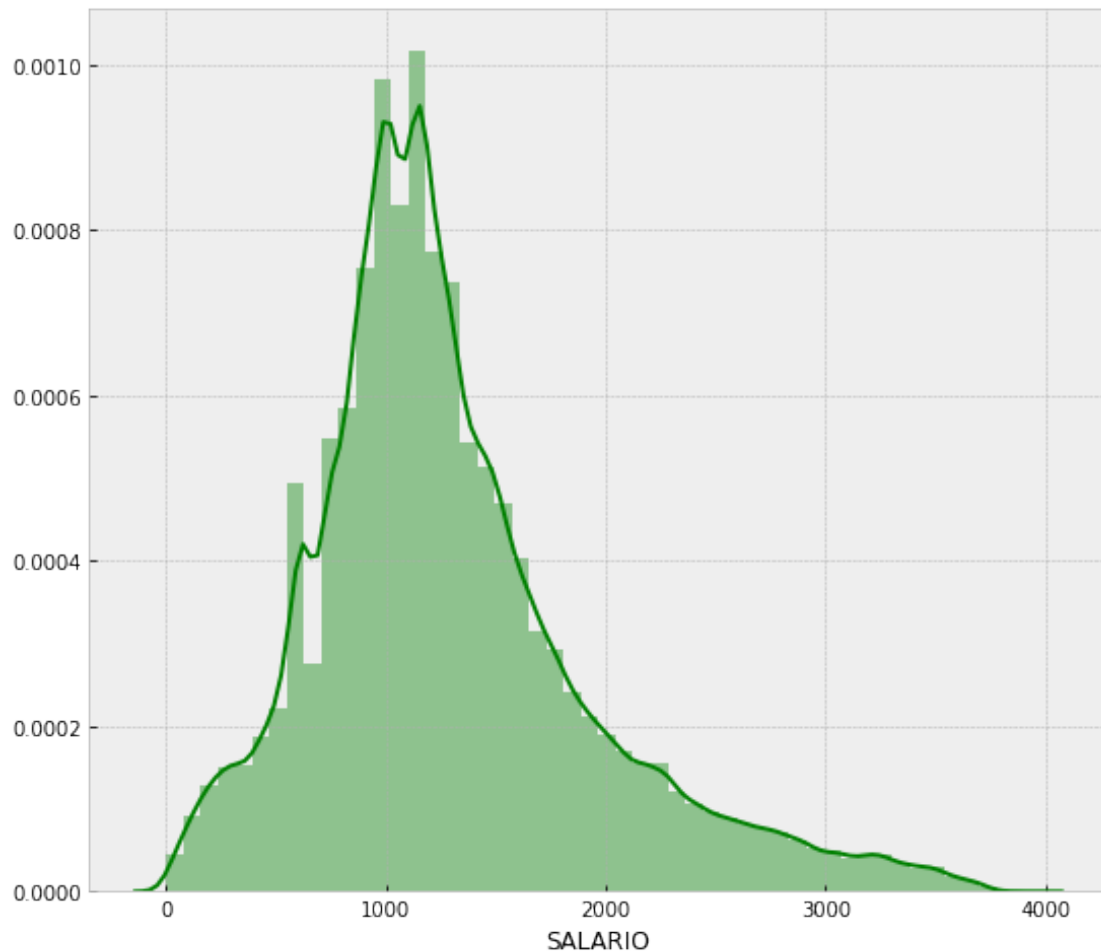
```



La columna sigue sin ser una distribución normal (estrictamente hablando) Pero la visualización es más clara. Tanto la media como la mediana de la gráfica, ronda los 9€/h.

```
[36]: columnName = 'SALARIO'
print(dfSueldos[columnName].describe())
plt.figure(figsize=(9, 8))
sns.distplot(dfSueldos[columnName], color='g', bins=50, hist_kws={'alpha': 0.
↪4});
```

```
count    207992.000000
mean      1295.553388
std        638.557696
min         0.000000
25%        897.810000
50%       1168.975000
75%       1576.150000
max       3934.200000
Name: SALARIO, dtype: float64
```



Al igual que el precio hora, se parece a una distribución normal, pero como hemos vistos en los test anteriores no es así.

### 5.2.1 gráficas Q-Q

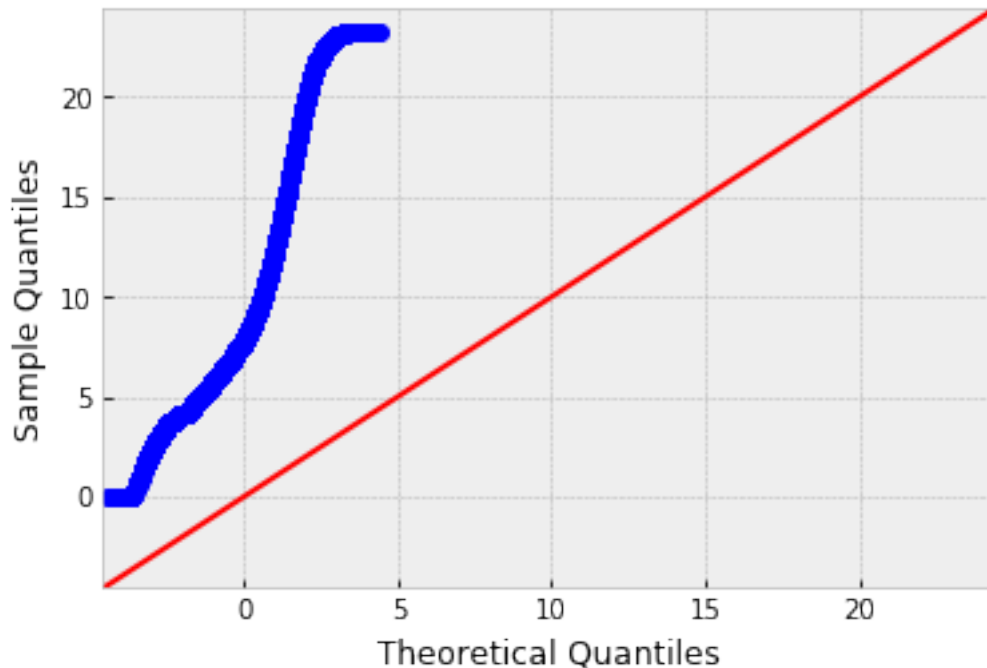
Aunque hayamos comprobado en los test anteriores que las distribuciones no son normales, parece interesante dibujar e interpretar las gráficas QQ de alguna de las anteriores variables.

ref: <https://www.geeksforgeeks.org/qqplot-quantile-quantile-plot-in-python/> (hacer las gráficas en python) ref: <https://boostedml.com/2019/03/linear-regression-plots-how-to-read-a-qq-plot.html> (interpretación de las gráficas QQ)

En concreto “Precio /hora” cuya gráfica de densidad es parecida a una distribución normal:

```
[37]: sm.qqplot(dfSueudos['PRECIOHORA'], line = '45');
```





La verticalidad de la gráfica en el centro, nos indica un crecimiento muy rápido (abrupto) de la gráfica de densidad. El principio, y el final, con los puntos horizontales, se pueden interpretar como colas ligeras: hay una gran mayoría que tiene el precio hora fijado en la mediana, casi un crecimiento exponencial, y en los extremos apenas hay población.

### 5.2.2 Homocedasticidad

Una vez probada la (no) normalidad de las distribuciones continuas del dataset, vamos a comprobar sus varianzas para diferentes grupos de datos. Para la comprobación si varios grupos tienen la misma distribución en sus variables tenemos varios tests: por ejemplo el test de Levene, si los datos siguen una distribución normal, o el de Fligner-Killeen cuando no. Como los datos, como hemos visto antes, no siguen la distribución normal, vamos a empezar haciendo el test de Fligner-Killeen para la comprobación de las varianzas de la variable salario, para hombres y mujeres.

```
[38]: # Para hacer el test de levene, cogemos los registros en los que el trabajador
      ↳ tiene sexo = hombre (1):
registros_hombres = dfSueldos[(dfSueldos.SEXO == 1)]
#y los de las mujeres
registros_mujeres = dfSueldos[(dfSueldos.SEXO == 6)]
```

### 5.2.3 Test de Fligner Killen

La **hipótesis nula** dice que no hay diferencias significativas entre los sueldos de mujeres y hombres.

```
[39]: from scipy.stats import fligner

print(fligner(registroshombres['SALARIO'], registrosmujeres['SALARIO']))
print(fligner(registroshombres['PRECIOHORA'], registrosmujeres['PRECIOHORA']))
```

```
FlignerResult(statistic=27.542164213474432, pvalue=1.537068213439582e-07)
```

```
FlignerResult(statistic=510.1102007780646, pvalue=6.001200488699651e-113)
```

Para el test de Fligner, se vuelve a confirmar que las varianzas del sueldo de hombres y mujeres NO son homogéneas (porque el pvalue es inferior a 0.05), luego se rechaza la hipótesis nula.

### 5.2.4 Test de Levene.

ref: <https://aaronshlegel.me/levenes-test-equality-variances-python.html> (El test de levene en Python)

```
[40]: print(stats.levene(registroshombres['SALARIO'], registrosmujeres['SALARIO'],
    ↪center='mean'))
print(stats.levene(registroshombres['PRECIOHORA'],
    ↪registrosmujeres['PRECIOHORA'], center='mean'))
```

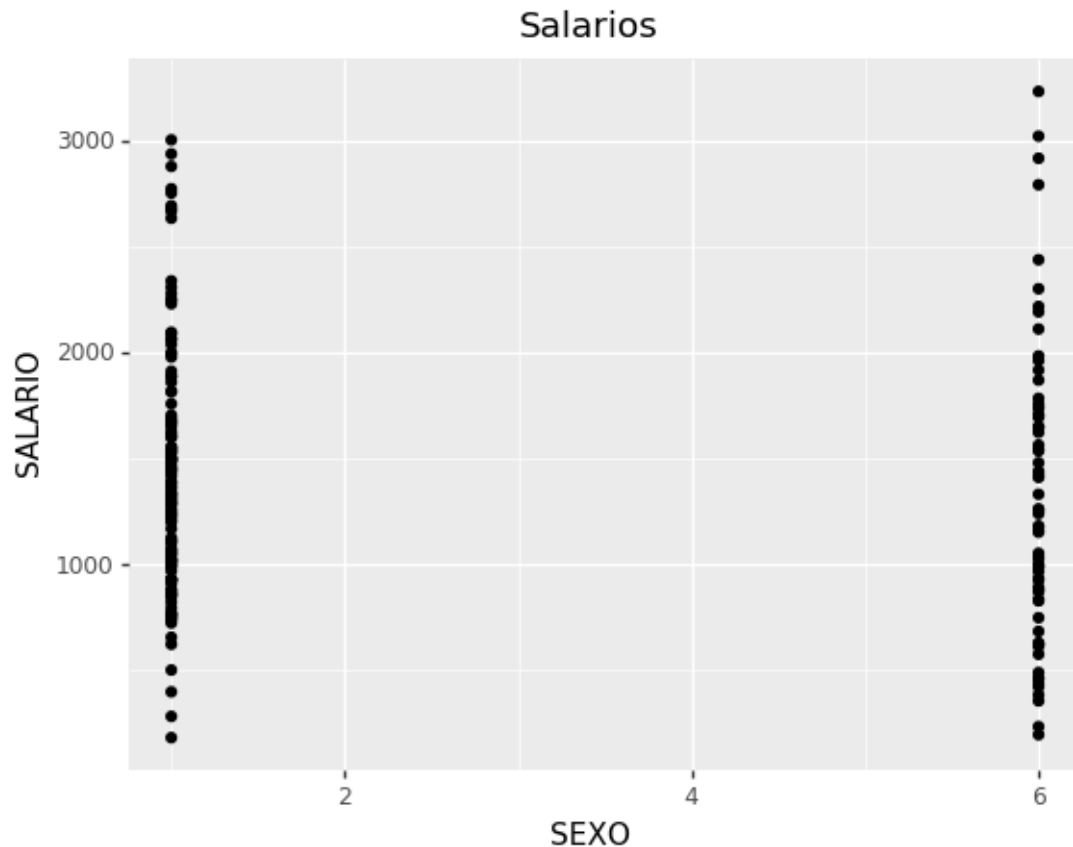
```
LeveneResult(statistic=119.980905510717, pvalue=6.500558119450657e-28)
```

```
LeveneResult(statistic=525.2518123637237, pvalue=4.248882521278517e-116)
```

En ambos casos el p-value es INFERIOR a 0.05, tanto para el precio/hora, como para el salario. Por tanto asumimos que NO hay homogeneidad de varianzas entre los grupos.

Veamos si podemos visualizar gráficamente los resultados de estos tests con ggplot2: ref: [https://ggplot2.tidyverse.org/reference/scale\\_manual.html](https://ggplot2.tidyverse.org/reference/scale_manual.html)

```
[41]: from plotnine.data import economics
from plotnine import ggplot, aes, geom_point, scale_x_discrete, labs
# Tenemos tantas muestras en este dataset que hemos tenido que extraer 200
    ↪registros aleatorios para dibujarlos y que parezcan puntos y no una línea
    ↪continua que no podemos interpretar.
ggplot(dfSueldos.sample(200)) + aes(x="SEXO", y="SALARIO") + geom_point() +
    ↪labs(title="Salarios")
```



[41]: <ggplot: (-9223371867403599696)>

Parece que para el dataset que estamos viendo, esta gráfica no es la más adecuada. Vamos a probar a dibujar un plotbox donde veamos media, varianza y sus distribuciones en hombres y mujeres. ref: <https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib>

```
[42]: salariomujeres = registrosmujeres['SALARIO']
salariohombres = registroshombres['SALARIO']

salarios = [salariomujeres, salariohombres]

fig = plt.figure(figsize =(7, 5))

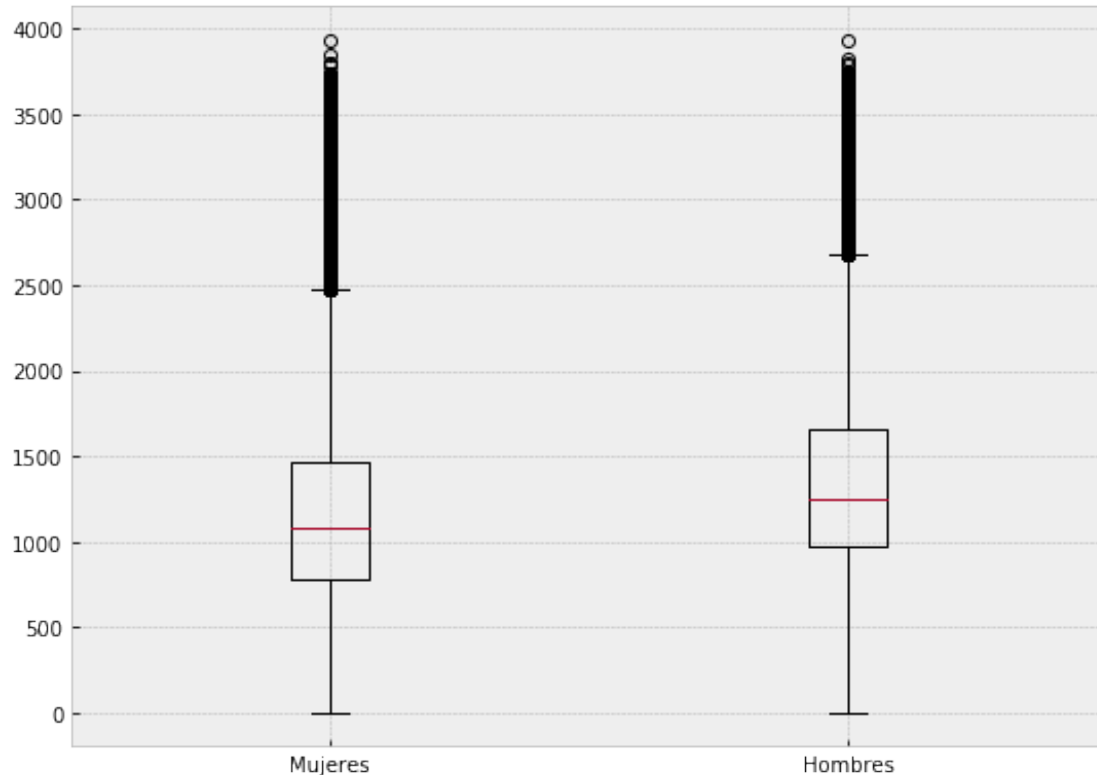
# Creating axes instance
ax = fig.add_axes([0, 0, 1, 1])

# Creating plot
bp = ax.boxplot(salarios)

# x-axis labels
```

```
ax.set_xticklabels(['Mujeres', 'Hombres'])

# show plot
plt.show()
```



### 5.3 4.3. Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc.

Podemos calcular si hay diferencias significativas entre la media de salario, de mujeres y de hombres. En este caso podemos aplicar el test paramétrico t-test de Student, que requiere que las muestras a comparar sigan una distribución normal.

Por un lado vimos que la variable “Salario”, no sigue exactamente una distribución normal, pero aplicando el teorema del límite central con una muestra suficientemente grande (mayor de 30, y en este caso lo es de sobra) se puede asumir que la variable sigue una distribución normal.

Cuando la normalidad y la homocedasticidad se cumplan (p-valores mayores al nivel de significancia), se podrán aplicar pruebas por contraste de hipótesis de tipo paramétrico, como la prueba t de Student. En los casos en los que no se cumplan (como el nuestro), se deberán aplicar pruebas no paramétricas como Wilcoxon (cuando se comparen datos dependientes) o Mann-Whitney (cuando los grupos de datos sean independientes).

```
[43]: # Tanto el test de Student como Wilcoxon necesita que los datasets tengan el
      ↪ mismo número de registros.
      # Así que vamos a coger 91417 registros de salarios de hombres al azar:
      salariomujeres = registrosmujeres['SALARIO']
      salariohombres = registroshombres['SALARIO'].sample(salariomujeres.shape[0])

      #Igual con el precio hora (ph)
      phmujeres = registrosmujeres['PRECIOHORA']
      phhombres = registroshombres['PRECIOHORA'].sample(salariomujeres.shape[0])
```

### 5.3.1 test de Student

```
[44]: # Vamos a comparar las medias de los dos grupos, con t-Test.
      print(stats.ttest_rel(salariohombres, salariomujeres))
      print(stats.ttest_rel(phhombres, phmujeres))
```

```
Ttest_relResult(statistic=65.2895131408458, pvalue=0.0)
```

```
Ttest_relResult(statistic=34.14143146221414, pvalue=7.181897021551857e-254)
```

El resultado de la prueba de Student, en la variable pvalue en ambos casos es muy inferior a 0.05, así que rechazamos la hipótesis nula: *las medias de los sueldos de hombres y mujeres no son iguales*.

Algo que teníamos bastante claro después de ver el boxplot dibujado anteriormente.

### 5.3.2 prueba de Wilcoxon

En python, la prueba de Wilcoxon se aplica mediante el método wilcoxon de la librería stats.

ref: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.wilcoxon.html>

La hipótesis nula dice que no hay diferencias entre la media de salarios de hombres y mujeres, así que vamos a hacer el test de wilcoxon indicando que las distribuciones en los grupos son iguales. Veamos si sale el mismo resultado que el test de Student.

```
[45]: print(wilcoxon(salariohombres, salariomujeres))
      print(wilcoxon(phhombres, phmujeres))
```

```
WilcoxonResult(statistic=1550414115.5, pvalue=0.0)
```

```
WilcoxonResult(statistic=1806913483.5, pvalue=2.071702935269014e-271)
```

De nuevo, al igual que la de Student, la variable pvalue sigue siendo muy inferior al limite que habíamos marcado.

Vamos a probar con esta hipótesis alternativa: “Los salarios de mujeres son menores que los de los hombres”

```
[46]: print(wilcoxon(salariohombres, salariomujeres, alternative='less', correction_
      ↪='false'))
      print(wilcoxon(phhombres, phmujeres, alternative='less', correction='false'))
```

```
WilcoxonResult(statistic=2624783774.5, pvalue=1.0)
```

```
WilcoxonResult(statistic=2368284406.5, pvalue=1.0)
```

Podemos aceptar la hipótesis alternativa.

### 5.3.3 Chi cuadrado

ref: <https://python-bloggers.com/2020/09/how-to-run-chi-square-test-in-python>

Queremos saber ahora la relación que existe entre la clase “Estudios” (ESTU) y el “sexo” (SEXO).

Al ser 2 variables categóricas vamos a aplicar por ejemplo el test Chi-Cuadrado. Aquí la hipótesis nula nos dice que las variables son independientes.

Vamos a comprobarlo:

```
[47]: #aislo el dataset estudios y sexo
estudiosporsexo = dfSuealdos[['SEXO', 'ESTU']]

#creo la tabla de contingencia
tablecontingency= pd.crosstab(estudiosporsexo['SEXO'], estudiosporsexo['ESTU'])

#con esta tabla de contingencia pasamos el test de chi-cuadrado:
c, p, dof, expected = (chi2_contingency(tablecontingency))

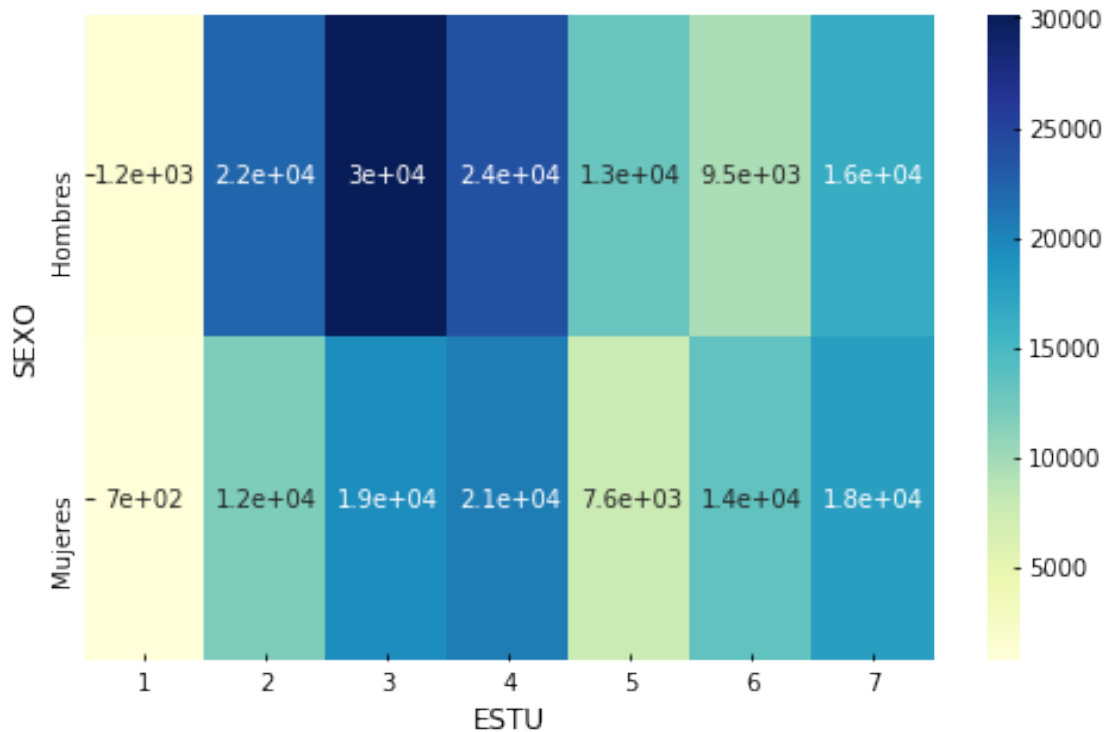
print (p)
```

0.0

Rechazamos la hipótesis nula: Los estudios no son independientes al sexo. Para verlo de una forma gráfica, vamos a hacer un mapa de calor y observar gráficamente las distribuciones:

```
[48]: plt.figure(figsize=(8,5))
y_axis_labels = [ "Hombres", "Mujeres"]
sns.heatmap(tablecontingency, yticklabels = y_axis_labels, annot=True,
            cmap="YlGnBu")
```

```
[48]: <matplotlib.axes._subplots.AxesSubplot at 0x277431257c8>
```



En esta tabla podemos ver que aproximadamente los mismos hombres y mujeres no terminaron la educación primaria.

Pero a partir de ahí, muchos **más hombres terminaron su educación primaria y educación secundaria**.

Sin embargo, hay una clara diferencia en cuanto a formaciones profesionales, y universitarios. Mientras que encontramos muchos **más hombres que optan por una FP**, o educación profesional, comparando, hay **más mujeres que terminan sus estudios universitarios**, tanto diplomados, como licenciados/postgrados.

Veamos otras condiciones diferenciadas por sexo, por ejemplo, el tipo de contrato.

### 5.3.4 Test exacto de Fisher

Para esto voy a aplicar el test de Fisher: ¿Tienen las mujeres y hombres el mismo tipo de contratos (definidos o indefinidos)?

```
[49]: #aislo el dataset contrato y sexo
contratosporsexo = dfSuealdos[['TIPOCON','SEXO']]

#creo la tabla de contingencia
contingencia= pd.crosstab(contratosporsexo['SEXO'], contratosporsexo['TIPOCON'])

print("Tabla de contingencia: ")
```

```
print(contingencia)

oddsratio, pvalue = stats.fisher_exact(contingencia)

print("Valor de pvalue: ", pvalue)
```

Tabla de contingencia:

TIPOCON	1	2
---------	---	---

SEXO		
------	--	--

1	92306	24269
---	-------	-------

6	72048	19369
---	-------	-------

Valor de pvalue: 0.040301110355900185

Pvalue ha salido menos de 5, que rechaza la hipótesis nula: sí que hay diferencia entre el tipo de contratos que hay en hombres y mujeres. Sin embargo, es el valor de pvalue más alto que hemos visto en el estudio. La diferencia no es abismal, y así podemos verlo en el siguiente heatmap:

```
[50]: plt.figure(figsize=(8,5))
y_axis_labels = [ "Hombres", "Mujeres"]
x_axis_labels = [ "Indefinido", "Dur. Determinada"]
sns.heatmap(contingencia, yticklabels = y_axis_labels, xticklabels =
↳x_axis_labels, annot=True, cmap="YlGnBu")
```

```
[50]: <matplotlib.axes._subplots.AxesSubplot at 0x2774316e0c8>
```





Repito el test de Fisher con otra variable: Quiero conocer la relación del tipo de jornadas partidas/completas, con respecto al sexo.

```
[51]: #aislo el dataset jornada y sexo
jornadaporsexo = dfSueldos[['TIPOJOR','SEXO']]

#creo la tabla de contingencia
contingencia= pd.crosstab(jornadaporsexo['SEXO'], jornadaporsexo['TIPOJOR'])

print("Tabla de contingencia: ")
print(contingencia)

oddsratio, pvalue = stats.fisher_exact(contingencia)

print("Valor de pvalue: ", pvalue)
```

Tabla de contingencia:

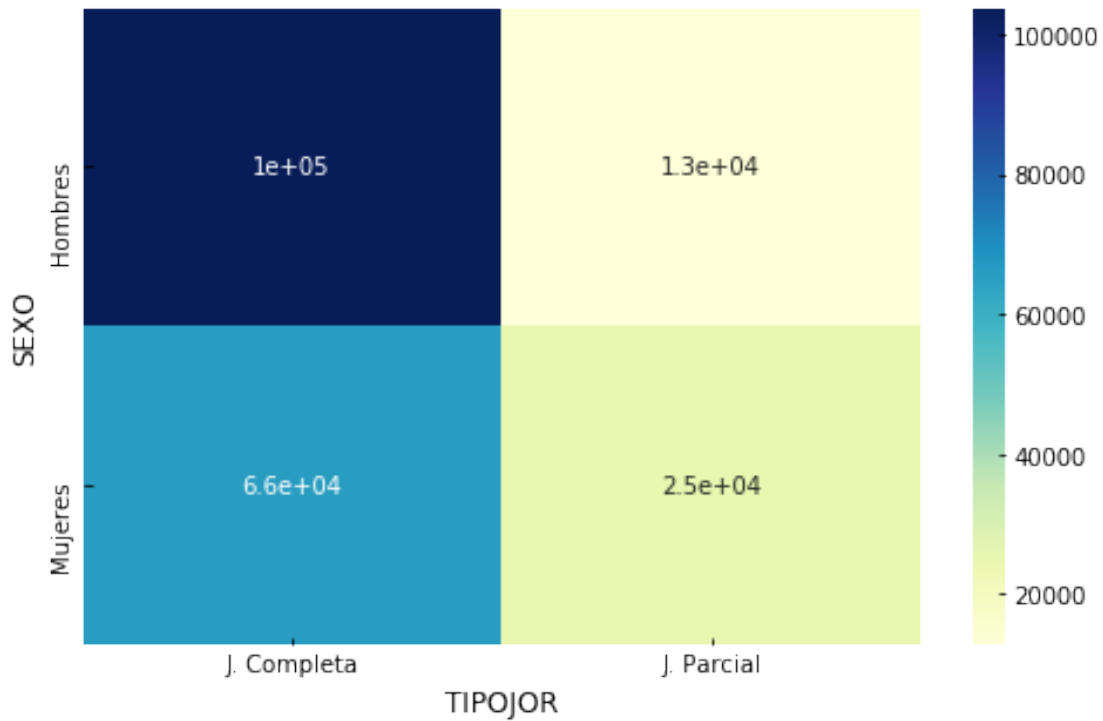
TIPOJOR	1	2
SEXO		
1	103949	12626
6	66017	25400

Valor de pvalue: 0.0

Por la tabla de contingencia ya se ve que hay bastante diferencia entre el número de mujeres y hombres y tipo de contratos. Veamos de nuevo el heatmap:

```
[52]: plt.figure(figsize=(8,5))
y_axis_labels = [ "Hombres", "Mujeres"]
x_axis_labels = [ "J. Completa", "J. Parcial"]
sns.heatmap(contingencia, yticklabels = y_axis_labels, xticklabels = _
↪x_axis_labels, annot=True, cmap="YlGnBu")
```

```
[52]: <matplotlib.axes._subplots.AxesSubplot at 0x2774b0f5408>
```



Aunque en el mercado existen menos contratos parciales que completos, sí que hay diferencia entre los que ocupan mujeres y hombres, como bien nos dice el Test de Fisher anterior: hay más hombres que mujeres que tienen jornadas completas y más mujeres que tienen jornadas parciales.

### 5.3.5 Regresión

**RANDOM FOREST REGRESSOR** Vamos ahora a generar un modelo que nos ayude a predecir el salario de una persona a partir de los datos introducidos.

Utilizaremos un Random Forest Regressor ya que la mayoría de nuestras variables son categóricas y este tipo de modelos funcionan mejor con este tipo de variables.

También aprovecharemos para obtener información sobre qué variables detecta nuestro modelo que influyen más sobre el salario.

En primer lugar realizaremos algunas transformaciones sobre nuestro dataset para que los datos se ajusten a lo que nuestro modelo necesita.

```
[53]: # Use numpy to convert to arrays
import numpy as np

##### REDUCIMOS TAMAÑO EN LAS PRUEBAS #####
features= dfSueldos.sample(frac=0.3, random_state = 1)

#features= dfSueldos
```

```

features_df = dfSueudos

#cambiamos las columnas CNACE y CNO1 a categorical
features['CNACE'] = features['CNACE'].astype('category')
features['CNO1'] = features['CNO1'].astype('category')

CNACE = features['CNACE'].astype('category')
CNO1 = features['CNO1'].astype('category')

features["CNACE"] = features["CNACE"].cat.codes
features["CNO1"] = features["CNO1"].cat.codes

```

Podemos realizar nuestro estudio sobre las variables **SALARIO** o **PRECIOHORA**. Vamos a utilizar **SALARIO** inicialmente.

```

[54]: columnToStudy = 'SALARIO'

# almacenamos los datos de la columna a estudiar para poder realizar_
↳ comparaciones.
labels = np.array(features[columnToStudy])

#Quitamos las columnas que que no queremos que influyan en el modelo ya que_
↳ están relacionadas entre ellas.
columns = ['SALARIO', 'PRECIOHORA', 'SALBASE']
features = features.drop(columns, axis = 1)

#Vamos a transformar nuestro dataset a array, así que guardamos los nombres de_
↳ las columnas para más adelante.
feature_list = list(features.columns)
features = np.array(features)

```

```

[55]: # Obtenemos conjuntos de entrenamiento y test

from sklearn.model_selection import train_test_split

train_features, test_features, train_labels, test_labels =
↳ train_test_split(features, labels, test_size = 0.25, random_state = 42)

```

```

[56]: # Aplicamos Random Forest Regresor

import time
from datetime import timedelta
from sklearn.ensemble import RandomForestRegressor

```

```

# Durante las pruebas hemos calculado los tiempos que costaba entrenar el
→ modelo para poder planificar mejor nuestras actuaciones
start = time.time()
print(f"Proceso iniciado a las: {timedelta(seconds=start)}")

# Instanciamos el modelo. Hemos probado con diferentes parámetros pero lo hemos
→ dejado con 1000.
# Esto hace que el entrenamiento del modelo tarde bastante más, pero creemos
→ que obtiene mejores resultados.
rf = RandomForestRegressor(n_estimators = 200, random_state = 0)
# Entrenamos el modelo con el conjunto de entrenamiento
rf.fit(train_features, train_labels);

# Tiempo final
end = time.time()

# Mostramos el tiempo transcurrido para entrenar el modelo.
print(f"Proceso finalizado a las: {timedelta(seconds=end)}")
print(f"Tiempo total: {timedelta(seconds=end - start)}")

```

Proceso iniciado a las: 18631 days, 20:55:19.647323  
Proceso finalizado a las: 18631 days, 20:56:21.037637  
Tiempo total: 0:01:01.390314

```

[57]: # Ahora vamos a comprobar las predicciones realizadas
predictions = rf.predict(test_features)

```

```

[58]: from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Error absoluto medio (MAE)
mae = mean_absolute_error(test_labels.ravel(), predictions)

# Error cuadrático medio (MSE)
mse = mean_squared_error(test_labels.ravel(), predictions)

# R-squared scores
r2 = r2_score(test_labels.ravel(), predictions)

# Print metrics
print('Error absoluto medio (MAE):', round(mae, 2))
print('Error cuadrático medio (MSE):', round(mse, 2))
print('R-squared scores:', round(r2, 2))

```

Error absoluto medio (MAE): 242.3  
Error cuadrático medio (MSE): 129751.19  
R-squared scores: 0.68

Vemos que nuestro modelo no consigue unos datos espectaculares ya que su coeficiente de correlación

es de un 68%.

Una vez generado nuestro modelo de regresión, vamos a comprobar como le influyen las diferentes variables utilizadas.

Vamos a comprobar la **Feature Importance** obtenida por nuestro modelo y vamos a representarla mediante un par de gráficas que nos ayude a entender mejor que variables están influyendo más.

```
[59]: # Obtenemos la Feature Importance
importances = list(rf.feature_importances_)

# Les añadimos los nombres de columnas para hacerlas más entendibles
feature_importances = [(feature, round(importance, 4)) for feature, importance
    ↪ in zip(feature_list, importances)]

# Ahora vamos a ordenar y mostrar los resultados
feature_importances = sorted(feature_importances, key = lambda x: x[1], reverse
    ↪ = True)

[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in
    ↪ feature_importances];
```

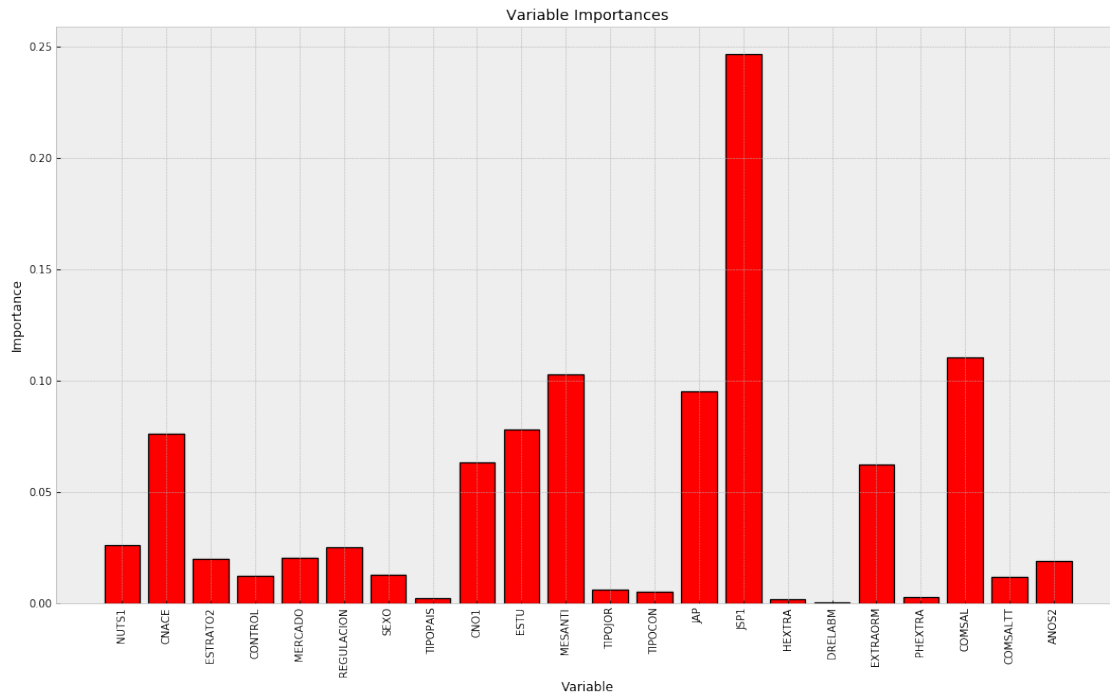
Variable: JSP1	Importance: 0.2467
Variable: COMSAL	Importance: 0.1105
Variable: MESANTI	Importance: 0.1026
Variable: JAP	Importance: 0.0953
Variable: ESTU	Importance: 0.078
Variable: CNACE	Importance: 0.0759
Variable: CNO1	Importance: 0.0631
Variable: EXTRAORM	Importance: 0.0621
Variable: NUTS1	Importance: 0.0261
Variable: REGULACION	Importance: 0.0253
Variable: MERCADO	Importance: 0.0205
Variable: ESTRATO2	Importance: 0.02
Variable: ANOS2	Importance: 0.0191
Variable: SEXO	Importance: 0.0127
Variable: CONTROL	Importance: 0.012
Variable: COMSALTT	Importance: 0.012
Variable: TIPOJOR	Importance: 0.0062
Variable: TIPOCON	Importance: 0.005
Variable: PHEXTRA	Importance: 0.0027
Variable: TIPOPAIS	Importance: 0.0024
Variable: HEXTRA	Importance: 0.0016
Variable: DRELABM	Importance: 0.0003

```
[60]: # Obtenemos las variables a mostrar en le gráfico
x_values = list(range(len(feature_importances)))

f, ax = plt.subplots(figsize=(18,10))
# Generamos un gráfico de barras
```

```
plt.bar(x_values, importances, orientation = 'vertical', color = 'r', edgecolor='k', linewidth = 1.2)

plt.xticks(x_values, feature_list, rotation='vertical')
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');
```



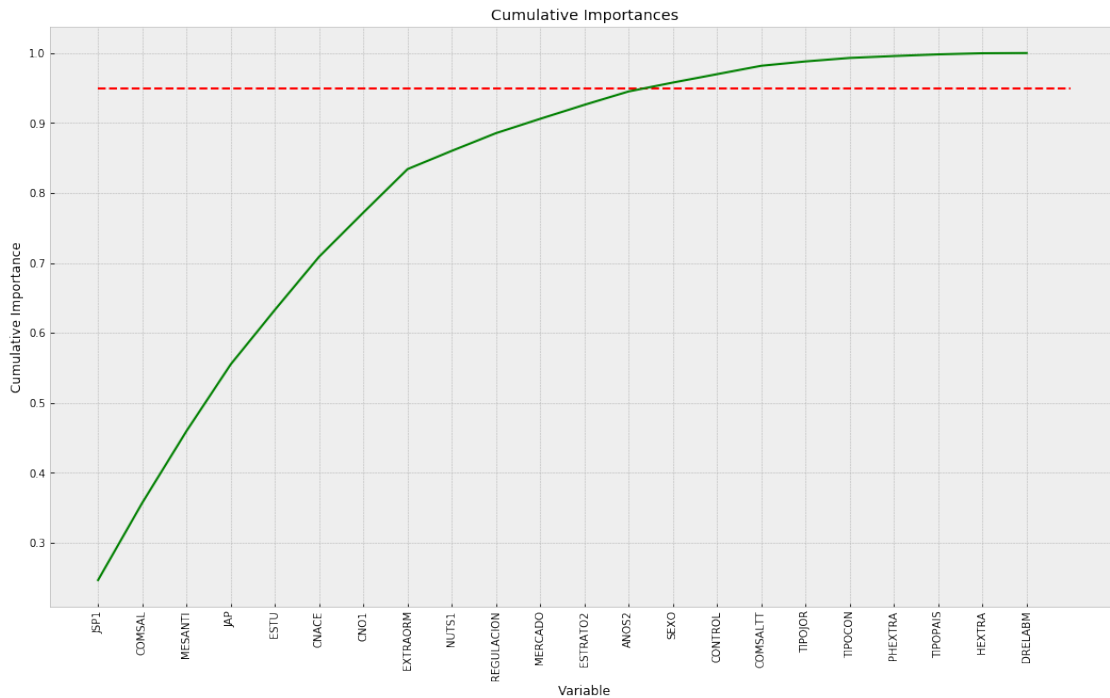
```
[61]: # Realizamos un nuevo gráfico con las variables ordenadas
sorted_importances = [importance[1] for importance in feature_importances]
sorted_features = [importance[0] for importance in feature_importances]

cumulative_importances = np.cumsum(sorted_importances)

f, ax = plt.subplots(figsize=(18,10))
plt.plot(x_values, cumulative_importances, 'g-')

plt.hlines(y = 0.95, xmin=0, xmax=len(sorted_importances), color = 'r',
linestyles = 'dashed')

plt.xticks(x_values, sorted_features, rotation = 'vertical')
plt.xlabel('Variable'); plt.ylabel('Cumulative Importance'); plt.
title('Cumulative Importances');
```



Podemos comprobar como **JSP1** (JORNADA SEMANAL PACTADA (HORAS)) es la variable que más influye, pero que el peso está bastante repartido.

Aunque para ayudarnos en el análisis, y para comprobar los resultados de las pruebas estadísticas, y los contrastes, regresión y correlación ya hemos utilizado muchas tablas y gráficas.

Vamos a centrarnos un poco más en la división de los datos entre hombres y mujeres, y comparación de diferentes variables entre estos dos grupos:

## 6 5. Representación de los resultados a partir de tablas y gráficas

Primero, vamos a realizar unas comparaciones de la distribución de salarios que predice nuestro modelo con la distribución de salario real.

```
[62]: # Obtenemos el dataset inicial y lo transformamos en arrays para poder pasarselo
      ↪ al modelo.

features_comprobacion = dfSueldos

#cambiamos las columnas CNACE y CNO1 a categorical
features_comprobacion['CNACE'] = features_comprobacion['CNACE'].
      ↪ astype('category')
features_comprobacion['CNO1'] = features_comprobacion['CNO1'].astype('category')

CNACE = features_comprobacion['CNACE'].astype('category')
```

```
CNO1 = features_comprobacion['CNO1'].astype('category')

features_comprobacion["CNACE"] = features_comprobacion["CNACE"].cat.codes
features_comprobacion["CNO1"] = features_comprobacion["CNO1"].cat.codes
```

```
[63]: #Quitamos las columnas que utilizamos en la predicción
columns = ['SALARIO', 'PRECIOHORA', 'SALBASE']
features_comprobacion = features_comprobacion.drop(columns, axis = 1)

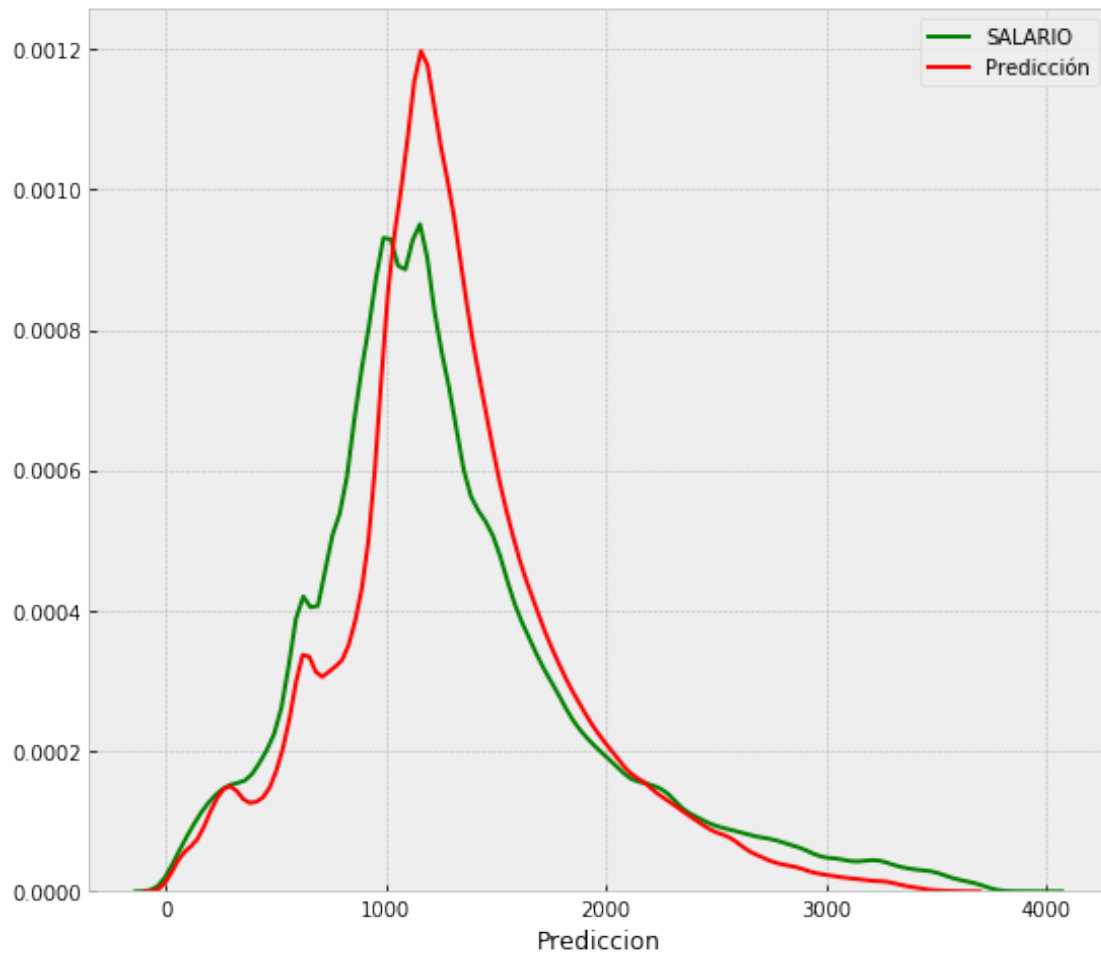
features_comprobacion = np.array(features_comprobacion)

prediccion = rf.predict(features_comprobacion)
```

```
[64]: #Añadimos la predicción al dataset original
dfWithPrediccion = dfSueldos
dfWithPrediccion['Prediccion'] = prediccion
```

```
[65]: fig = plt.figure(figsize=(9, 8))
sns.distplot(dfWithPrediccion[columnToStudy], color='g', bins=100,
↳hist_kws={'alpha': 0.4}, hist=False, label = columnToStudy);
sns.distplot(dfWithPrediccion['Prediccion'], color='r', bins=100,
↳hist_kws={'alpha': 0.4}, hist=False, label = 'Predicción');
```





Podemos ver que nuestra predicción está algo desajustada de la distribución real. Sobre todo en la parte central de la distribución.

## 7 Comparación de sueldos

Ahora vamos a realizar diferentes comparaciones de la variable **SALARIO** tratando de entender mejor si existen diferencias por sexo con respecto a esta variable. En nuestro modelo anterior para predecir el salario la variable **SEXO** no parece tener una importancia capital.

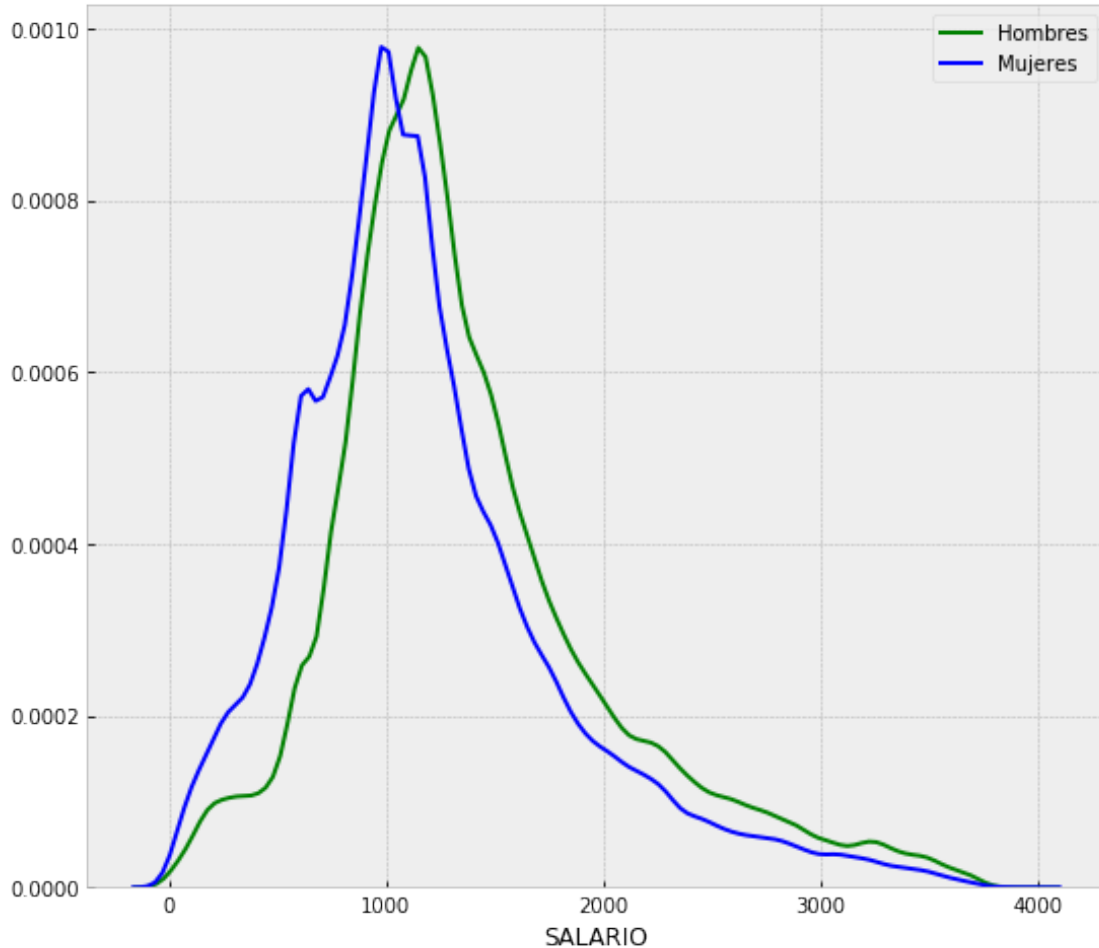
Vamos a ver que pasa si comparamos datos. Realizaremos diferentes comparativas por diferentes categorías de los datos a ver que conclusiones podemos obtener.

### 7.1 Comparación por sexo

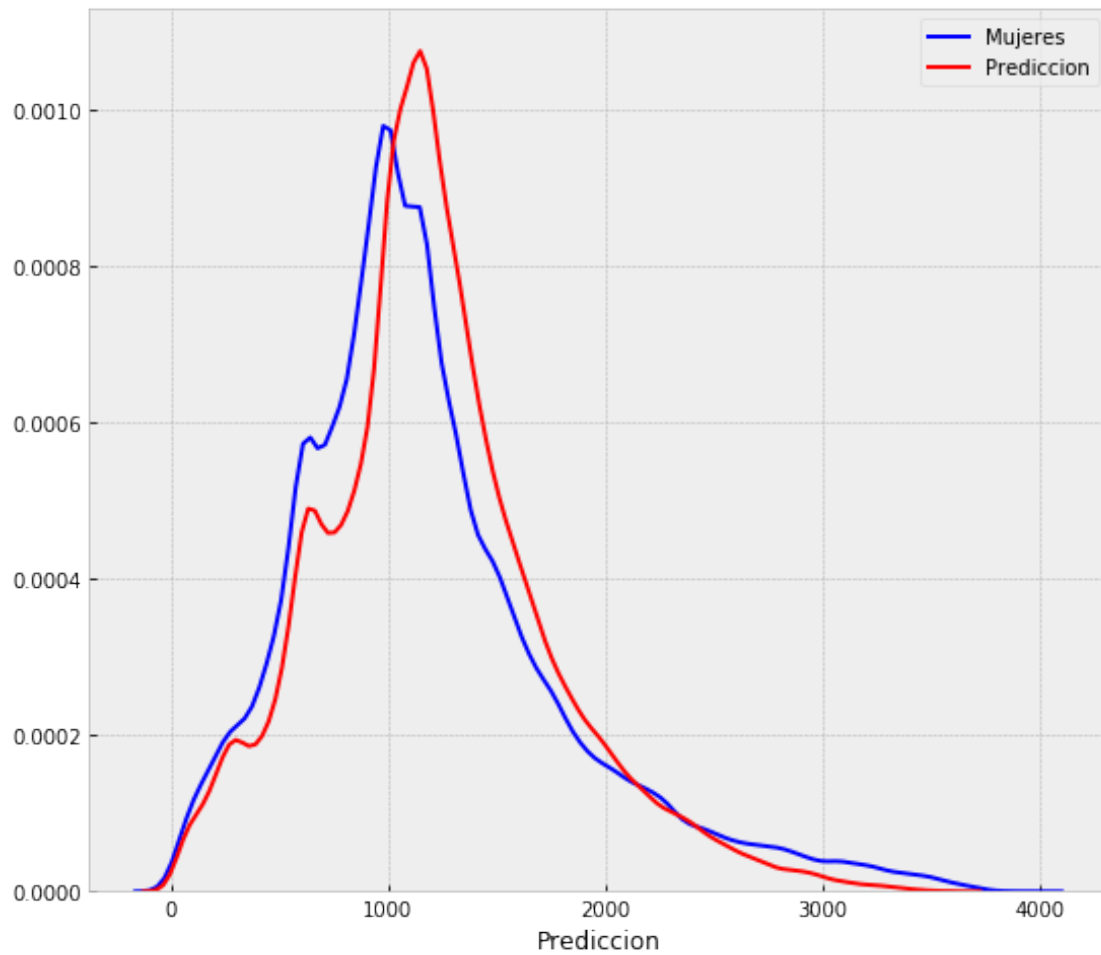
Primero realizaremos una comparación general de los dos sexos y luego una comparación contra los datos que predice nuestro modelo.

```
[66]: hombres = dfWithPrediccion[dfWithPrediccion['SEXO'] == 1]
      mujeres = dfWithPrediccion[dfWithPrediccion['SEXO'] == 6]

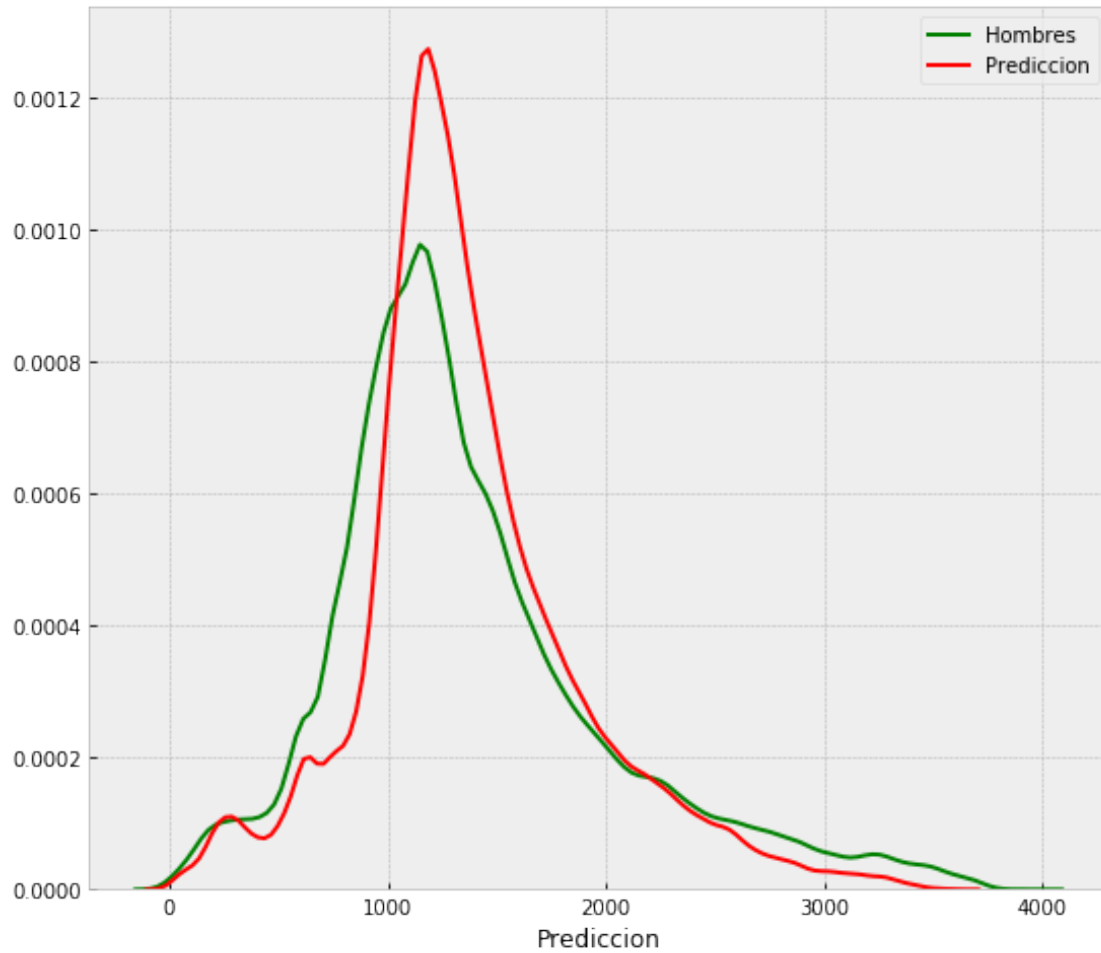
      fig = plt.figure(figsize=(9, 8))
      sns.distplot(hombres[columnToStudy], color='g', bins=100, hist_kws={'alpha': 0.
      ↪4}, hist=False, label = 'Hombres');
      sns.distplot(mujeres[columnToStudy], color='b', bins=100, hist_kws={'alpha': 0.
      ↪4}, hist=False, label = 'Mujeres');
```



```
[67]: fig = plt.figure(figsize=(9, 8))
      sns.distplot(mujeres[columnToStudy], color='b', bins=100, hist_kws={'alpha': 0.
      ↪4}, hist=False, label = 'Mujeres');
      sns.distplot(mujeres['Prediccion'], color='r', bins=100, hist_kws={'alpha': 0.
      ↪4}, hist=False, label = 'Prediccion');
```



```
[68]: fig = plt.figure(figsize=(9, 8))
sns.distplot(hombres[columnToStudy], color='g', bins=100, hist_kws={'alpha': 0.
↪4}, hist=False, label = 'Hombres');
sns.distplot(hombres['Prediccion'], color='r', bins=100, hist_kws={'alpha': 0.
↪4}, hist=False, label = 'Prediccion');
```



## 7.2 Comprobamos sueldo por edad y sexo

```
[69]: #Por Edad

edades = {1: 'MENOS 19 AÑOS',
2: 'DE 20 A 29',
3: 'DE 30 A 39',
4: 'DE 40 A 49',
5: 'DE 50 A 59',
6: 'MÁS DE 59'}

columnToSearch = 'ANOS2'

f, axes = plt.subplots(3, 2, figsize=(20, 10), sharex=True)

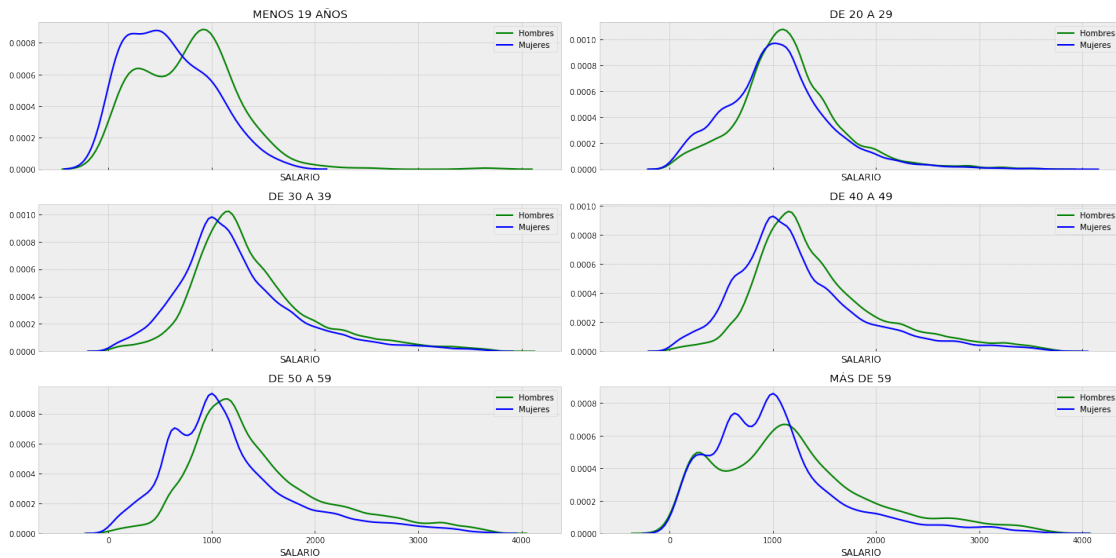
for i, ax in enumerate(axes.flatten()):
    df_category = dfWithPrediccion[dfWithPrediccion[columnToSearch] == i+1]
```

```

hombres = df_category[df_category['SEXO'] == 1]
mujeres = df_category[df_category['SEXO'] == 6]
sns.distplot(hombres[columnToStudy], color='g', bins=100, hist_kws={'alpha':
↪ 0.4}, hist=False, label = 'Hombres', ax = ax);
sns.distplot(mujeres[columnToStudy], color='b', bins=100, hist_kws={'alpha':
↪ 0.4}, hist=False, label = 'Mujeres', ax = ax);
ax.set_title(edades[i+1])

f.tight_layout()

```



```

[70]: f, axes = plt.subplots(1, 6, figsize=(20,8), sharex=False)

for i, ax in enumerate(axes.flatten()):
    df_category = dfWithPrediccion[dfWithPrediccion[columnToSearch] == i+1]
    if (len(df_category)>0):
        my_pal = {1: "green", 6: "blue"}

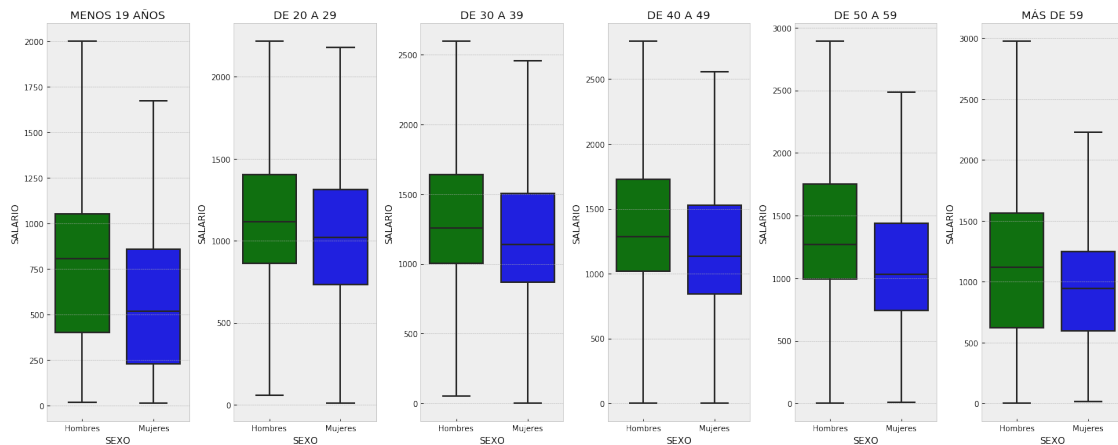
        a = sns.boxplot(x='SEXO', y="SALARIO", data=df_category, ax=ax,
↪ palette=my_pal, showfliers = False, width=0.75)

        ax.set(xticklabels=['Hombres', 'Mujeres'])
        ax.set_title(edades[i+1])

    else:
        ax.axis('off')

```

```
f.tight_layout();
```



Parece que los sueldos para el sexo femenino son inferiores en todas las categorías de edad. Las diferencias más apreciables se producen en las franjas de **Menos de 19** y **De 50 a 59**.

Llama especialmente la atención que en general, en todas las franjas de edad, la media de salarios de las mujeres es menor, y también los sueldos más altos son alcanzados por hombres.

En jóvenes de menos de 19 años, hay más mujeres trabajando, pero es la franja de edad en la que la diferencia de sueldo es más grande: las mujeres jóvenes trabajan más por menos dinero.

En edades avanzadas, a partir de 59, las mujeres siguen cobrando menos, sin embargo hay más mujeres que hombres trabajando. Como no ocurre en ninguna otra franja, podríamos pensar que se debe a que los hombres se prejubilán antes (necesitaríamos un estudio más en profundidad para averiguar las causas de esto -también sociológico-). Podríamos pensar que los hombres tienen más trabajos físicos que hacen que se tengan que jubilar antes, o que los puestos de los hombres y la riqueza acumulada a lo largo de su vida, les permite prejubilarse antes voluntariamente. (Obviamente son conjeturas y sabemos que es un tema demasiado complicado como para sacar la causa a partir de este estudio)

## 8 Comprobamos sueldo por puesto y sexo

Como los nombres de los valores de esta categoría son tan grandes los hemos reducido en las gráficas. Los mostramos aquí completos para poder consultarlos fácilmente.

Codigo	Puesto
0	DIRECTORES Y GERENTES
1	TÉCNICOS Y PROFESIONALES CIENTÍFICOS E INTELECTUALES DE LA SALUD Y LA ENSEÑANZA
2	OTROS TÉCNICOS Y PROFESIONALES CIENTÍFICOS E INTELECTUALES
3	TÉCNICOS; PROFESIONALES DE APOYO

Codigo	Puesto
4	EMPLEADOS DE OFICINA QUE NO ATIENDEN AL PÚBLICO
5	EMPLEADOS DE OFICINA QUE ATIENDEN AL PÚBLICO
6	TRABAJADORES DE LOS SERVICIOS DE RESTAURACION Y COMERCIO
7	TRABAJADORES DE LOS SERVICIOS DE SALUD Y EL CUIDADO DE PERSONAS
8	TRABAJADORES DE LOS SERVICIOS DE PROTECCION Y SEGURIDAD
9	TRABAJADORES CUALIFICADOS EN EL SECTOR AGRÍCOLA, GANADERO, FORESTAL Y PESQUERO
10	TRABAJADORES CUALIFICADOS DE LA CONSTRUCCION, EXCEPTO LOS OPERADORES DE MÁQUINAS
11	TRABAJADORES CUALIFICADOS DE LAS INDUSTRIAS MANUFACTURERAS, EXCEPTO OPERADORES DE INSTALACIONES Y MÁQUINAS
12	OPERADORES DE INSTALACIONES Y MAQUINARIA FIJAS, Y MONTADORES
13	CONDUCTORES Y OPERADORES DE MAQUINARIA MOVIL
14	TRABAJADORES NO CUALIFICADOS EN SERVICIOS
15	PEONES DE LA AGRICULTURA, PESCA, CONSTRUCCIÓN, INDUSTRIAS MANUFACTURERAS Y TRANSPORTES
16	OCUPACIONES MILITARES

[71]: *#Por cargo*

```
cargos = {0:'DIRECTORES Y GERENTES',
1:'TÉCNICOS Y PROFESIONALES CIENTÍFICOS...',
2:'OTROS TÉCNICOS Y PROFESIONALES ...',
3:'TÉCNICOS; PROFESIONALES DE APOYO',
4:'EMPLEADOS DE OFICINA QUE NO ATIENDEN AL PÚBLICO',
5:'EMPLEADOS DE OFICINA QUE ATIENDEN AL PÚBLICO',
6:'SERVICIOS DE RESTAURACION Y COMERCIO',
7:'SERVICIOS DE SALUD Y EL CUIDADO DE PERSONAS',
8:'SERVICIOS DE PROTECCION Y SEGURIDAD',
9:'SECTOR AGRÍCOLA, GANADERO, FORESTAL Y PESQUERO',
10:'CONSTRUCCION, EXCEPTO LOS OPERADORES DE MÁQUINAS',
11:'INDUSTRIAS MANUFACTURERAS, EXCEPTO...',
12:'OPERADORES DE INSTALACIONES ...',
13:'CONDUCTORES Y OPERADORES DE MAQUINARIA MOVIL',
14:'TRABAJADORES NO CUALIFICADOS EN SERVICIOS',
15:'PEONES DE LA AGRICULTURA, PESCA,...',
16:'OCUPACIONES MILITARES'}

columnToSearch = 'CNO1'

f, axes = plt.subplots(9, 2, figsize=(20, 40), sharex=False)

for i, ax in enumerate(axes.flatten()):
    df_category = dfWithPrediccion[dfWithPrediccion[columnToSearch] == i]
```

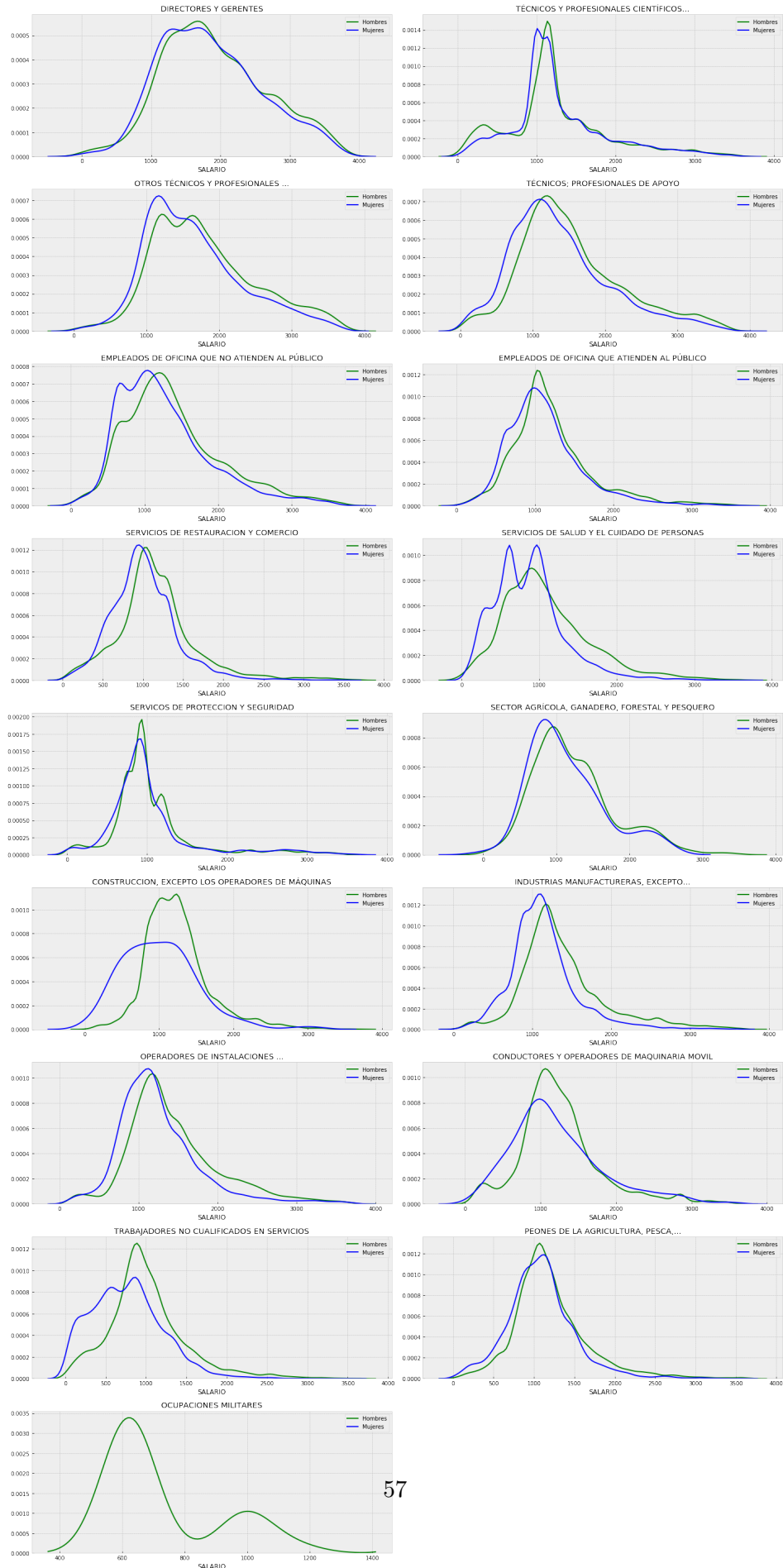
```

if (len(df_category)>0):
    hombres = df_category[df_category['SEXO'] == 1]
    mujeres = df_category[df_category['SEXO'] == 6]
    sns.distplot(hombres[columnToStudy], color='g', bins=100,
→hist_kws={'alpha': 0.4}, hist=False, label = 'Hombres', ax = ax);
    sns.distplot(mujeres[columnToStudy], color='b', bins=100,
→hist_kws={'alpha': 0.4}, hist=False, label = 'Mujeres', ax = ax);
    ax.set_title(cargos[i])
else:
    ax.axis('off')

f.tight_layout()

```





Llama la atención que respecto al cargo, en general, las mujeres tienden a cobrar menos. La media de los salarios de las líneas azules por lo general quedan más a la derecha. Algunos datos que llaman la atención son que hay más directoras y gerentes mujeres que cobran menos. Con el mismo cargo, y sueldos más grandes, la línea de los hombres sí que aparece más arriba que la de las mujeres.

También podemos observar que el puesto de cuidadores de personas está ocupado mayormente por mujeres, sobre todo con sueldos bajos: pero a partir de los aproximadamente 1300€ sí que hay más hombres.

En todas las categorías hay más hombres que cobran más, y hay más mujeres que cobran menos (en algunas cambia la diferencia -mayor, menor, se entrecruzan- pero en líneas generales esto está pasando).

No hay representación femenina en ocupaciones militares.

```
[72]: f, axes = plt.subplots(6, 3, figsize=(20,50), sharex=False)

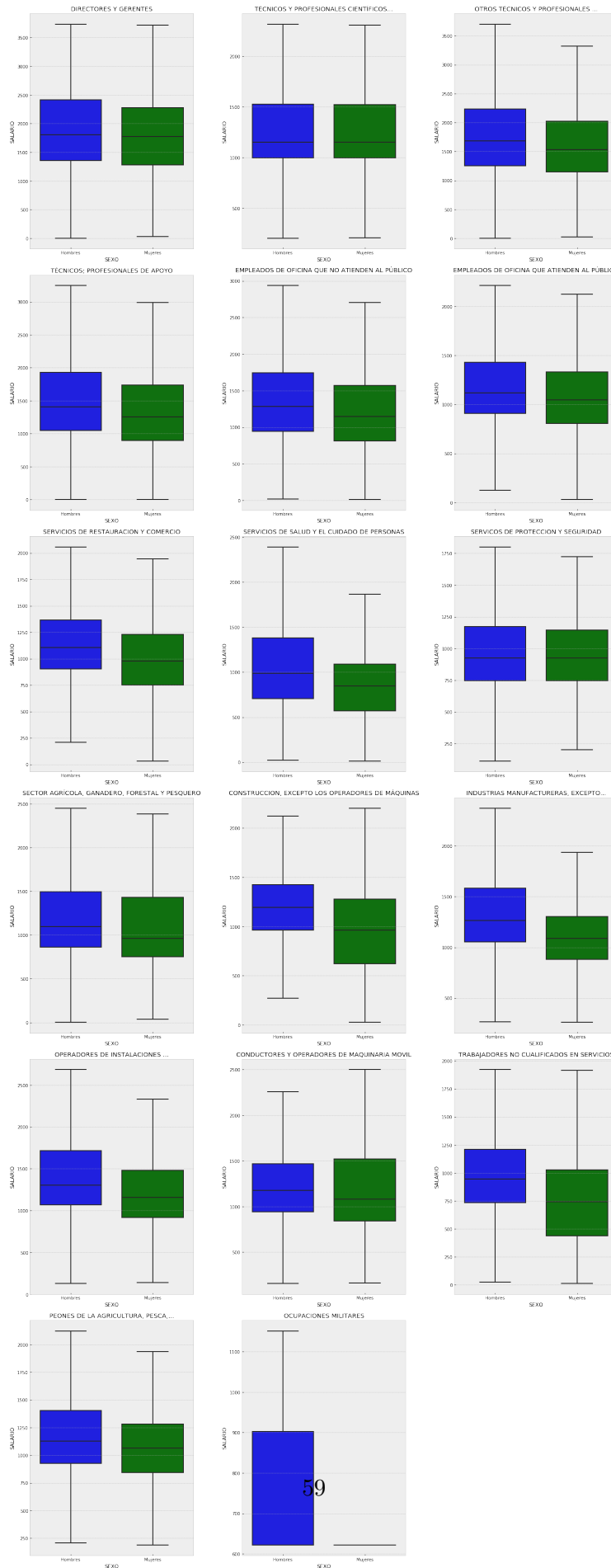
for i, ax in enumerate(axes.flatten()):
    df_category = dfWithPrediccion[dfWithPrediccion[columnToSearch] == i]
    if (len(df_category)>0):
        my_pal = {1: "blue", 6: "green"}

        a = sns.boxplot(x='SEXO', y="SALARIO", data=df_category, ax=ax,
                        palette=my_pal, showfliers = False, width=0.75)

        ax.set(xticklabels=['Hombres', 'Mujeres'])
        ax.set_title(cargos[i])

    else:
        ax.axis('off')

f.tight_layout();
```



Seguimos observando una tendencia en las gráficas de la distribución de salarios, donde las distribuciones de las mujeres aparecen escoradas hacia la izquierda, y con las gráficas de boxplot mostrando las medias y las concentraciones de sueldos de las mujeres por debajo de los hombres salvo en contadas excepciones, como los trabajos científicos e intelectuales, y los trabajos enfocados en protección y seguridad donde se encuentran todos los cuerpos de policía y bomberos y los sueldos están regulados por el gobierno.

Como curiosidad, en todo el dataset solo existía un registro de sexo femenino de profesión ocupaciones militares.

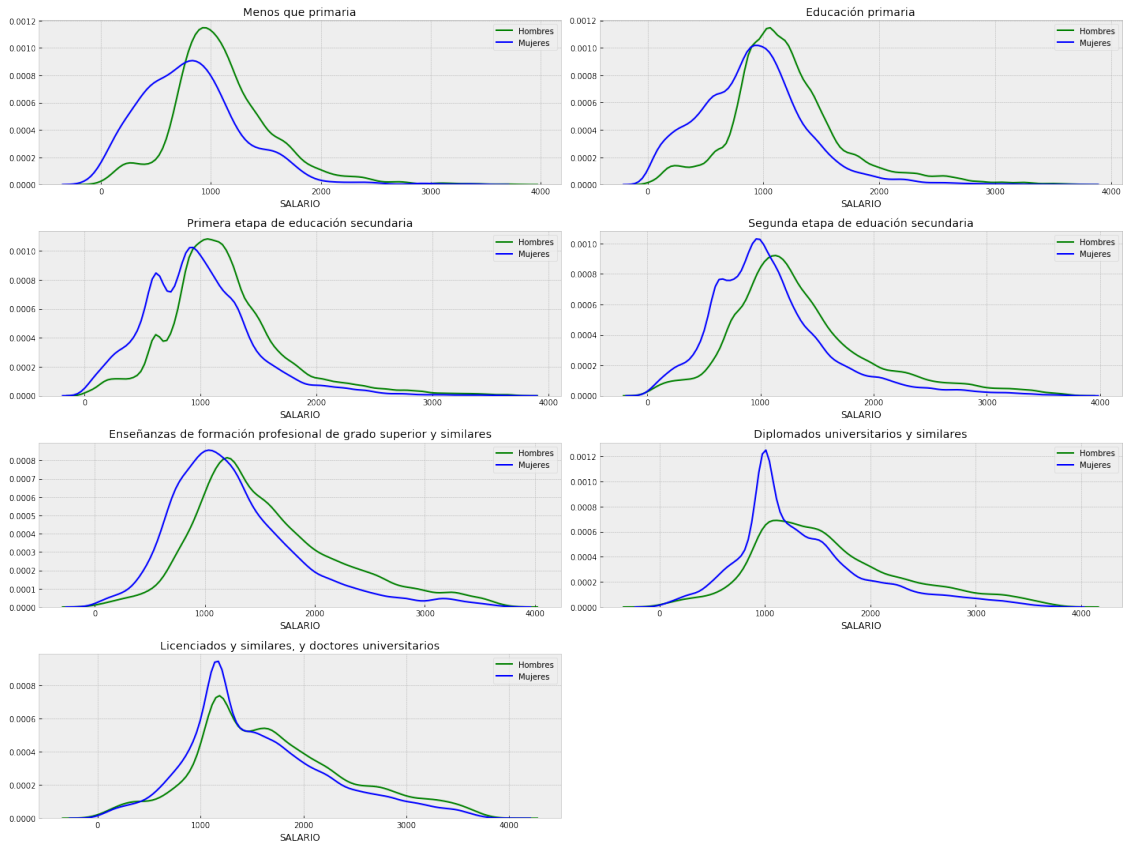
## 9 Comprobamos sueldo por nivel de estudios y sexo

```
[73]: estudios = {1:'Menos que primaria',
2:'Educación primaria',
3:'Primera etapa de educación secundaria',
4:'Segunda etapa de educación secundaria',
5:'Enseñanzas de formación profesional de grado superior y similares',
6:'Diplomados universitarios y similares',
7:'Licenciados y similares, y doctores universitarios'}

columnToSearch = 'ESTU'

f, axes = plt.subplots(4, 2, figsize=(20, 15), sharex=False)

for i, ax in enumerate(axes.flatten()):
    df_category = dfWithPrediccion[dfWithPrediccion[columnToSearch] == i+1]
    if (len(df_category)>0):
        hombres = df_category[df_category['SEXO'] == 1]
        mujeres = df_category[df_category['SEXO'] == 6]
        sns.distplot(hombres[columnToStudy], color='g', bins=100,
→hist_kws={'alpha': 0.4}, hist=False, label = 'Hombres', ax = ax);
        sns.distplot(mujeres[columnToStudy], color='b', bins=100,
→hist_kws={'alpha': 0.4}, hist=False, label = 'Mujeres', ax = ax);
        ax.set_title(estudios[i+1])
        f.tight_layout()
    else:
        ax.axis('off')
    #f.show()
```



```
[74]: f, axes = plt.subplots(2, 4, figsize=(20,15), sharex=False)

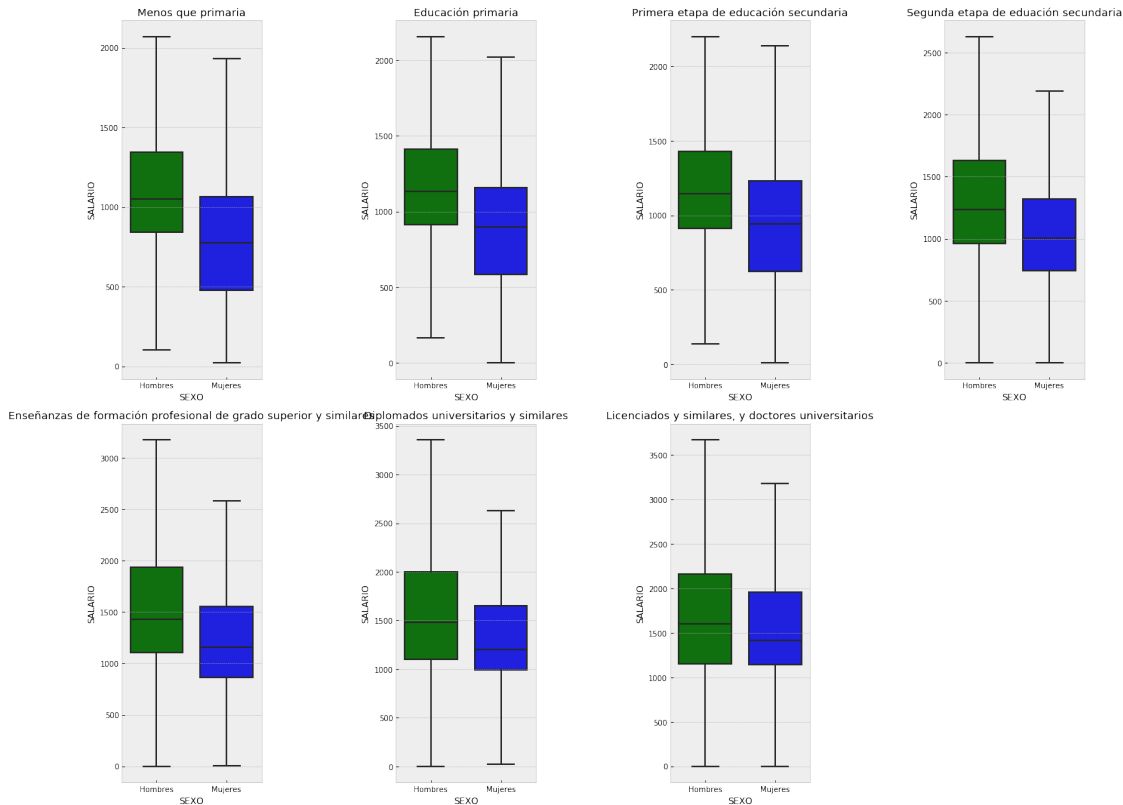
for i, ax in enumerate(axes.flatten()):
    df_category = dfWithPrediccion[dfWithPrediccion[columnToSearch] == i+1]
    if (len(df_category)>0):
        my_pal = {1: "green", 6: "blue"}

        a = sns.boxplot(x='SEXO', y="SALARIO", data=df_category, ax=ax,
                        palette=my_pal, showfliers = False, width=0.75)

        ax.set(xticklabels=['Hombres', 'Mujeres'])
        ax.set_title(estudios[i+1])

    else:
        ax.axis('off')

f.tight_layout();
```



Los resultados siguen en la línea de los anteriores. Cabe destacar como en la distribución de mujeres con **Diploma universitario o similares** aparece una gran concentración de registros alrededor de los 1000 €. Muy por encima del número de hombres: es decir, muchas diplomadas cobrando el sueldo mínimo interprofesional.

De nuevo se vuelve a repetir el patrón: para sueldos altos, con los mismos estudios, la línea verde de los hombres sigue estando por encima de la azul (hay más hombres que cobran más).

Además, como habíamos visto anteriormente, hay más mujeres con estudios universitarios que hombres. En cuestión de estudios de formación profesional, sí que hay más presencia masculina.

## 10 Comprobamos sueldos por industrias

```
[75]: industrias = {0:'Industrias extractivas: extracción de antracita, ...',
1:'Industria manufacturera: industria de la alimentación, ...',
2:'Industria manufacturera: industria de la madera ...',
3:'Industria manufacturera: artes gráficas y reproducción de soportes grabados',
4:'Industria manufacturera: petroleo, química, farmaceutica...',
5:'Industria manufacturera: fabricación de otros productos minerales no
↪metálicos',
6:'Industria manufacturera: metalurgia; fabricación de productos de hierro...',
```

```

7: 'Industria manufacturera: fabricación de productos informáticos, electrónicos.
   ↳...',
8: 'Industria manufacturera: fabricación de vehículos de motor...',
9: 'Suministro de energía eléctrica, gas, vapor y aire acondicionado',
10: 'Suministro de agua, actividades de saneamiento...',
11: 'Construcción: construcción de edificios, ingeniería civil...',
12: 'Venta y reparación de vehículos de motor y motocicletas, comercio al por_
   ↳mayor ...',
13: 'Comercio al por menor, excepto de vehículos de motor y motocicletas',
14: 'Transporte y almacenamiento: transporte terrestre y por tubería...',
15: 'Transporte y almacenamiento: almacenamiento y actividades anexas al_
   ↳transporte...',
16: 'Hostelería: servicios de alojamiento, servicios de comidas y bebidas',
17: 'Información y comunicaciones: edición, actividades cinematográficas...',
18: 'Actividades financieras y de seguros: servicios financieros...',
19: 'Actividades inmobiliarias',
20: 'Actividades profesionales, científicas y técnicas...',
21: 'Actividades administrativas y servicios auxiliares...',
22: 'Administración Pública y defensa; Seguridad Social obligatoria',
23: 'Educación',
24: 'Actividades sanitarias y de servicios sociales...',
25: 'Actividades artísticas, recreativas y de entrenamiento...',
26: 'Otros servicios: actividades asociativas...'}

columnToSearch = 'CNACE'

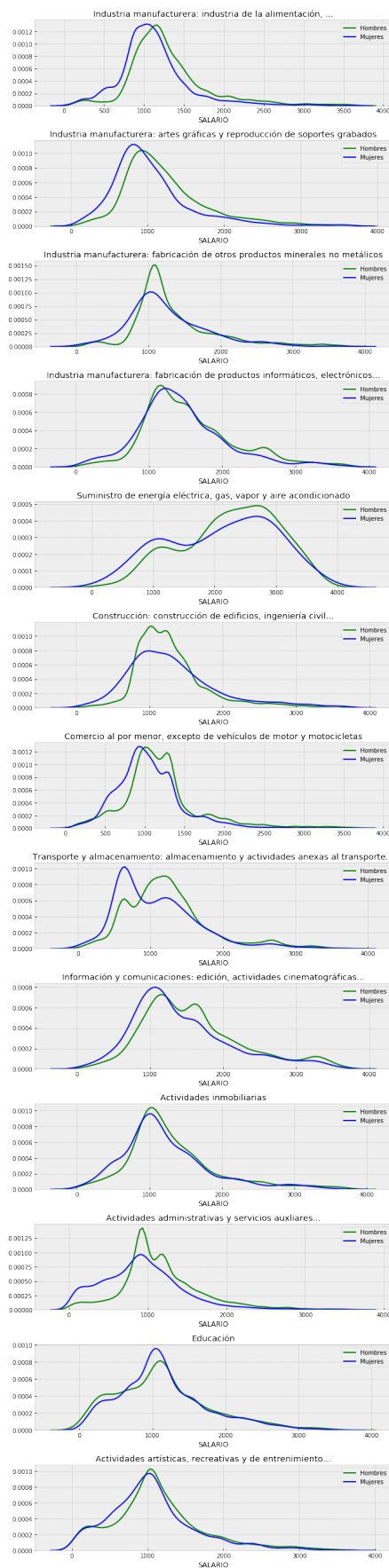
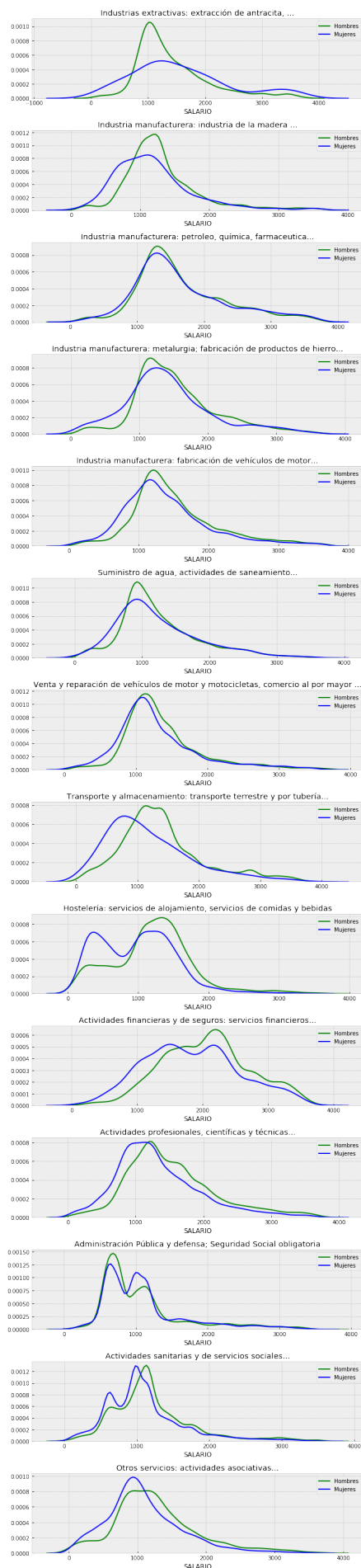
f, axes = plt.subplots(14, 2, figsize=(20, 40), sharex=False)

for i, ax in enumerate(axes.flatten()):
    df_category = dfWithPrediccion[dfWithPrediccion[columnToSearch] == i]
    if (len(df_category)>0):
        hombres = df_category[df_category['SEXO'] == 1]
        mujeres = df_category[df_category['SEXO'] == 6]
        sns.distplot(hombres[columnToStudy], color='g', bins=100,
↳hist_kws={'alpha': 0.4}, hist=False, label = 'Hombres', ax = ax);
        sns.distplot(mujeres[columnToStudy], color='b', bins=100,
↳hist_kws={'alpha': 0.4}, hist=False, label = 'Mujeres', ax = ax);
        ax.set_title(industrias[i])

    else:
        ax.axis('off')

f.tight_layout()

```





```

[76]: f, axes = plt.subplots(14, 2, figsize=(20,80), sharex=False)

for i, ax in enumerate(axes.flatten()):
    df_category = dfWithPrediccion[dfWithPrediccion[columnToSearch] == i]
    if (len(df_category)>0):
        my_pal = {1: "blue", 6: "green"}

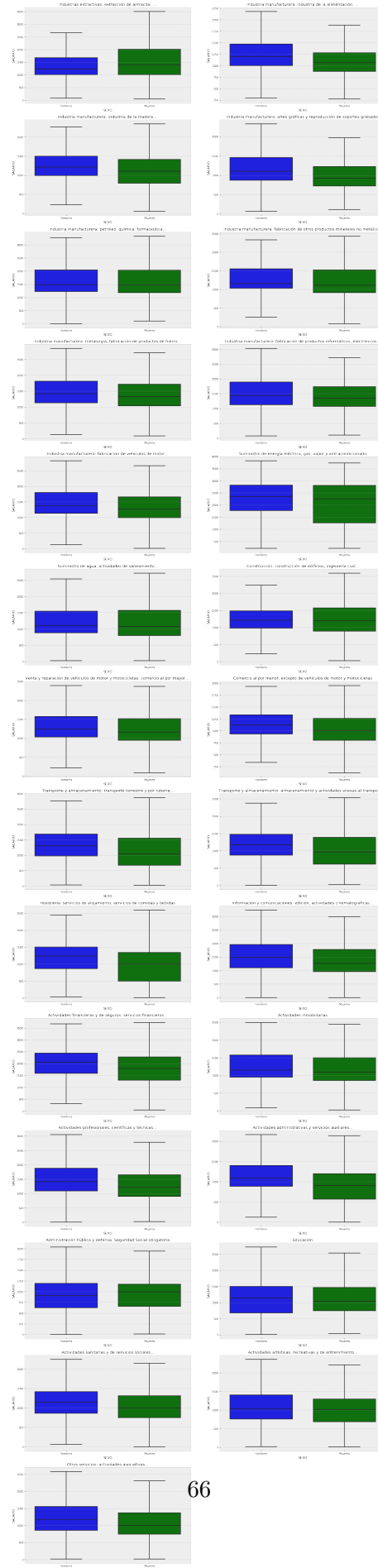
        a = sns.boxplot(x='SEXO', y="SALARIO", data=df_category, ax=ax,
                        palette=my_pal, showfliers = False, width=0.75)

        ax.set(xticklabels=['Hombres', 'Mujeres'])
        ax.set_title(industrias[i])

    else:
        ax.axis('off')

f.tight_layout();

```



Aunque en líneas generales, los resultados continúan siendo desfavorables a las mujeres, hay un par de puntos a destacar.

En las **Industrias extractivas: extracción de antracita, hulla y lignito, extracción de crudo de petróleo y gas natural, extracción de minerales metálicos, otras industrias extractivas, actividades de apoyo a las industrias extractivas** las mujeres muestran una media de sueldos mucho mayor. (Podríamos pensar que esto ocurre porque los puestos/cargos que ocupan las mujeres en estas industrias de más responsabilidad)

Y en las **Industria manufacturera: coquerías y refino de petróleo, industria química, fabricación de productos farmacéuticos, fabricación de productos de caucho y plásticos** parece que se ha alcanzado una cierta paridad en los sueldos, tanto en la media como en la distribución de los mismos.

## 11 Comprobamos hasta que punto cambia una predicción dependiendo del sexo

Vamos intentar comprobar hasta que punto nuestro modelo predictivo ha adquirido un sesgo relativo al sexo.

Para ello vamos a coger una muestra de los datos originales, vamos a cambiar el sexo de los registros de la muestra. Realizaremos una predicción de sueldo con nuestro modelo y compararemos los resultados.

```
[77]: # vamos a comparar los sueldos originales con los predichos

muestra_inicial = dfSueldos.sample(n=1000, random_state = 1)

test_sesgo_genero = muestra_inicial

#Cambiamos los valores de masculino a femenino y de femenino a masculino para
→realizar una nueva predicción
test_sesgo_genero['SEXO'] = test_sesgo_genero['SEXO'].map({1: 6, 6: 1})

#cambiamos las columnas CNACE y CNO1 a categorical
test_sesgo_genero['CNACE'] = test_sesgo_genero['CNACE'].astype('category')
test_sesgo_genero['CNO1'] = test_sesgo_genero['CNO1'].astype('category')

CNACE = test_sesgo_genero['CNACE'].astype('category')
CNO1 = test_sesgo_genero['CNO1'].astype('category')

test_sesgo_genero["CNACE"] = test_sesgo_genero["CNACE"].cat.codes
test_sesgo_genero["CNO1"] = test_sesgo_genero["CNO1"].cat.codes
```

```
[78]: # Labels are the values we want to predict
labels = np.array(test_sesgo_genero[columnToStudy])

#Quitamos las columnas que tienen un significado parecido al de nuestro estudio
columns = ['SALARIO', 'PRECIOHORA', 'SALBASE', 'Prediccion']
test_sesgo_genero = test_sesgo_genero.drop(columns, axis = 1)

# Convertimos a un formato que acepta nuestro modelo predictivo
test_sesgo_genero = np.array(test_sesgo_genero)
```

```
[79]: prediccion = rf.predict(test_sesgo_genero)
```

```
[80]: # Creamos nuevas columnas con los resultados
muestra_inicial['Prediccion2'] = prediccion
muestra_inicial['Diferencia'] = round(muestra_inicial['Prediccion2'] -
↳ muestra_inicial['Prediccion'], 2)
```

```
[81]: #nos quedamos solo con las columnas que queremos estudiar

df = muestra_inicial[['SALARIO', 'SEXO', 'Prediccion', 'Prediccion2',
↳ 'Diferencia']]

# volvemos a dejar la columna SEXO como estaba inicialmente
df['SEXO'] = df['SEXO'].map({1: 'Mujer', 6: 'Hombre'})
```

Vamos a mostrar los resultados. Si la diferencia es positiva significará que cambiar de sexo ha supuesto un cambio a mejor. Si la diferencia es negativa significará que el cambio de sexo ha supuesto un empeoramiento.

Mostraremos los datos par cada sexo.

Hay que tener en cuenta que todos estos datos se han calculado con nuestro modelo (que puede ser ampliamente mejorable).

```
[82]: print(df[df['SEXO']=='Mujer'].sort_values(by = 'Diferencia', ascending = True) )
```

	SALARIO	SEXO	Prediccion	Prediccion2	Diferencia
44954	2464.54	Mujer	2032.02530	1661.067400	-370.96
127509	1822.89	Mujer	1576.60795	1312.179250	-264.43
119340	1742.48	Mujer	1621.12560	1455.714375	-165.41
88505	3073.00	Mujer	2744.80530	2617.534050	-127.27
163918	1697.09	Mujer	1573.79630	1449.695100	-124.10
...	...	...	...	...	...
158848	1093.10	Mujer	1564.65375	2149.998400	585.34
8142	738.40	Mujer	983.17230	1696.492350	713.32
166041	735.90	Mujer	1021.04320	1737.052100	716.01
192719	1825.76	Mujer	1892.78525	2645.543300	752.76
895	1284.78	Mujer	1406.36370	2206.470050	800.11

[441 rows x 5 columns]

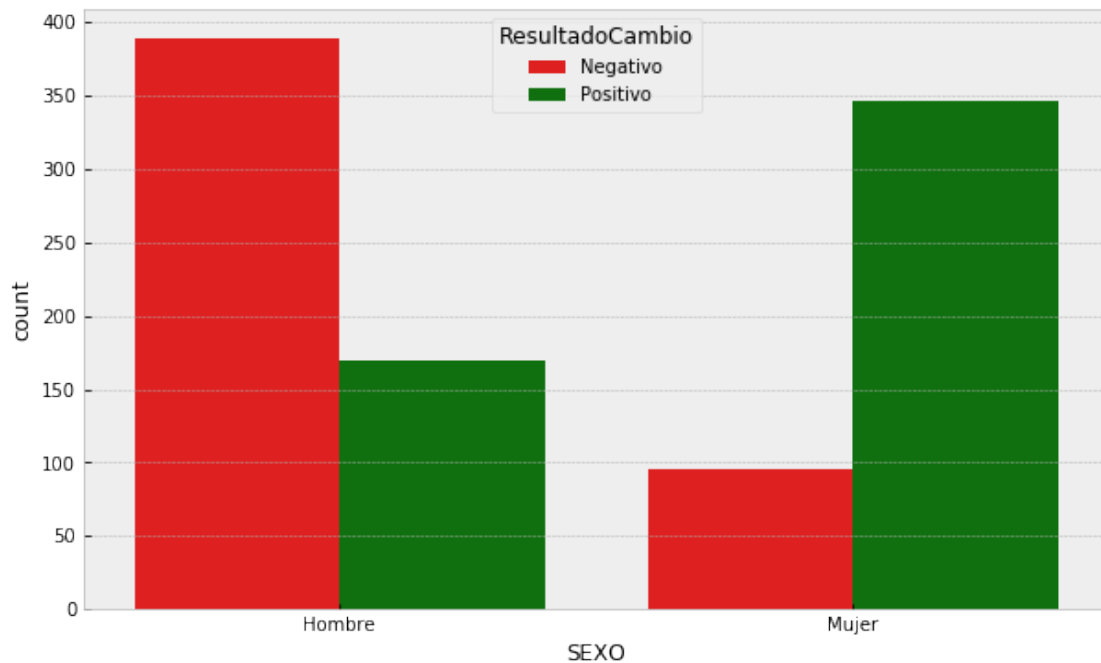
```
[83]: print(df[df['SEXO']=='Hombre'].sort_values(by = 'Diferencia', ascending = True))
```

	SALARIO	SEXO	Prediccion	Prediccion2	Diferencia
19400	2848.30	Hombre	2565.14540	1701.70000	-863.45
30567	2668.85	Hombre	2638.67510	1887.29265	-751.38
89670	2833.33	Hombre	2121.13385	1407.96955	-713.16
82362	2224.41	Hombre	2208.94015	1503.37225	-705.57
44895	2671.81	Hombre	2494.37415	1828.82770	-665.55
...	...	...	...	...	...
194828	1046.00	Hombre	1079.80040	1275.17040	195.37
12583	1230.28	Hombre	1251.42550	1500.54300	249.12
191940	1161.50	Hombre	1262.77480	1512.15190	249.38
158396	551.93	Hombre	710.68955	1177.18470	466.50
201713	1033.15	Hombre	1157.43925	1768.69830	611.26

[559 rows x 5 columns]

```
[84]: conditions = [  
    (df['Diferencia'] < 0),  
    (df['Diferencia'] >= 0)  
]  
  
values = ['Negativo', 'Positivo']  
  
df['ResultadoCambio'] = np.select(conditions, values)
```

```
[85]: plt.figure(figsize=(10, 6))  
sns.countplot(x='SEXO', hue='ResultadoCambio', data=df, palette=["red",  
    ↪ "green"]);
```



Podemos ver que al cambiar el sexo las predicciones cambian, las mujeres suelen tener predicciones positivas y los hombre suelen tener predicciones negativas.

Esto comportamiento nos advierte de que debemos tener un especial cuidado a la hora de entrenar nuestros modelos de IA para no transmitirles un sesgo de género.

### 11.1 Contribuciones en la práctica

Contribuciones	Firma
Investigación previa	Isabel Cabezas, Jorge Saura
Redacción de las respuestas	Isabel Cabezas, Jorge Saura
Desarrollo código	Isabel Cabezas, Jorge Saura