PROBLEMA DE CLASIFICACION

In [108]:

```
import pandas as pd
from plotnine import ggplot, aes, geom_line
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [2]:

```
paises = pd.read_csv("adult.data", header = None)
```

In [3]:

```
paises
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	
...	...	...	...	...	...	...	...	...	...	...	
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15

32561 rows × 15 columns

In [4]:

```
# 39 : variable edad.

# State-gov: Clase de trabajo(privado, trabaja para el gobierno local, trabaja para el
 gobierno regional etc...)

# ¿?

# Nivel_eduacion

# Codigo_nivel_educacion

# Estado civil:

# Ocuapcion

# Relacion_sentimental

# Raza

# Sexo

# Dinero_ganado

# Dinero_perdido

# Horas_semanales_trabajo

# Pais_origen

# Salario_futuro


lista_columnas = ["Edad", "Clase_trabajo", "¿?", "Nivel_eduacion", "Codigo_nivel_educac
ion",
                                                    "Estado_civil", "Ocupacion",
"Relacion_sentimental", "Raza", "Sexo",
                                                   "Dinero_ganado", "Dinero_perd
ido", "Horas_semanales_trabajo", "Pais_origen",
                                                   "Salario_futuro"]


paises.columns = lista_columnas

paises.columns
```

Out[4]:

```
Index(['Edad', 'Clase_trabajo', '¿?', 'Nivel_eduacion',
       'Codigo_nivel_educacion', 'Estado_civil', 'Ocupacion',
       'Relacion_sentimental', 'Raza', 'Sexo', 'Dinero_ganado',
       'Dinero_perdido', 'Horas_semanales_trabajo', 'Pais_origen',
       'Salario_futuro'],
      dtype='object')
```

In [5]:

```
paises
```

Out[5]:

	Edad	Clase_trabajo	¿?	Nivel_eduacion	Codigo_nivel_educacion	Estado_civil	C
0	39	State-gov	77516	Bachelors	13	Never-married	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	I
2	38	Private	215646	HS-grad	9	Divorced	
3	53	Private	234721	11th	7	Married-civ-spouse	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	
...	...	...	...	...	...	...	
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	
32558	58	Private	151910	HS-grad	9	Widowed	
32559	22	Private	201490	HS-grad	9	Never-married	
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	I

32561 rows × 15 columns

Analisis del dataframe en su conjunto

In [6]:

```
paises.describe()
```

Out[6]:

	Edad	¿?	Codigo_nivel_educacion	Dinero_ganado	Dinero_perdido
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000

In [7]:

```
# En relacion a las variables cuantitavas, podemos ver que en la variable edad, hay eda
des comprendidas entre los 17 y los 90 años,
# Codigo nivel de educacion tiene 16 niveles
# Horas semanales comprende desde 1 hora trabajada hasta las 99(algo extraño, ya que es
o supone trabajar mas de 16 horas al dia)
```

In [8]:

```
paises.dtypes
```

Out[8]:

```
Edad                     int64
Clase_trabajo           object
¿?                       int64
Nivel_eduacion          object
Codigo_nivel_educacion   int64
Estado_civil            object
Ocupacion               object
Relacion_sentimental    object
Raza                    object
Sexo                    object
Dinero_ganado            int64
Dinero_perdido           int64
Horas_semanales_trabajo  int64
Pais_origen             object
Salario_futuro          object
dtype: object
```

In [9]:

```
paises.shape

# tenemos 32561 filas y 15 columnas
```

Out[9]:

```
(32561, 15)
```

In [10]:

```
paises.info()

# No hay valores nulos en el df
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Edad                     32561 non-null  int64 
 1   Clase_trabajo            32561 non-null  object
 2   ¿?                       32561 non-null  int64 
 3   Nivel_eduacion           32561 non-null  object
 4   Codigo_nivel_educacion   32561 non-null  int64 
 5   Estado_civil             32561 non-null  object
 6   Ocupacion                32561 non-null  object
 7   Relacion_sentimental     32561 non-null  object
 8   Raza                     32561 non-null  object
 9   Sexo                     32561 non-null  object
 10  Dinero_ganado            32561 non-null  int64 
 11  Dinero_perdido           32561 non-null  int64 
 12  Horas_semanales_trabajo  32561 non-null  int64 
 13  Pais_origen              32561 non-null  object
 14  Salario_futuro           32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Nos encontramos con variables categoricas y varaibles cuantitativas

Variables categoricas: Clase_trabajo, Nivel_eduacion, Estado_civil, Ocupacion, Relacion_sentimental, Raza, Sexo, Pais_origen, Salario_futuro

Variables numericas: Edad, ¿?, Dinero_ganado, Dinero_perdido, Horas_semanales_trabajo

Analisis univariable de variables cuantitativas

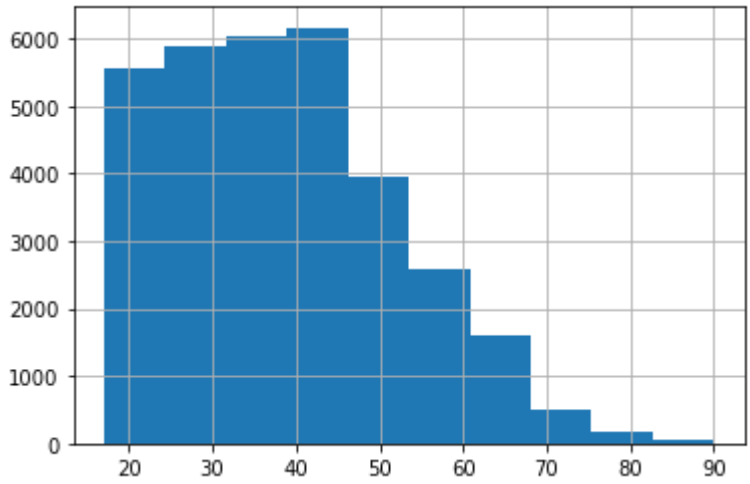
1 - EDAD

In [11]:

```
# Analisis de la variable Edad

paises.Edad.hist()

# Como puede verse en el histograma, las edades que se repiten mas en este dataset, son las comprendidas en la franja de
# 17 a 45años
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12912375370>
```

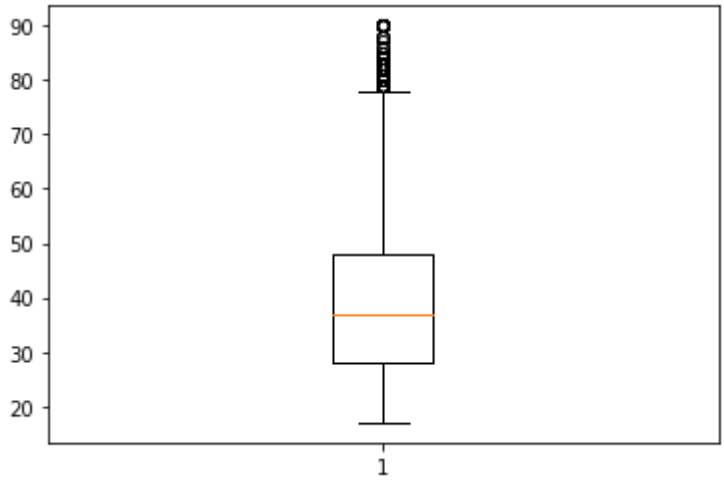


In [12]:

```
# Vamos a ver un diagrama de caja, para ver si existen valores atipicos

plt.boxplot(paises.Edad)

# Tenemos valores atipicos por encima de los 82 años
```

Out[12]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1291447cee0>,
  <matplotlib.lines.Line2D at 0x1291448b280>],
 'caps': [<matplotlib.lines.Line2D at 0x1291448b610>,
  <matplotlib.lines.Line2D at 0x1291448b970>],
 'boxes': [<matplotlib.lines.Line2D at 0x1291447cb80>],
 'medians': [<matplotlib.lines.Line2D at 0x1291448bcd0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1291448bfd0>],
 'means': []}
```



¿Que edades son las que tienen un salario mayor y cuales menor?

Podemos crear una nueva variable que contenga varias franjas de edad, para poder analizar como respondoden al salario

Para ello, vamos a crear las franjas de edad -25, 25-40, 41-65, +65

In [13]:

```
bins = [17,24, 40, 65, 90]

names = ["-25", "25-40", "41-65", "+65"]

paises["Rango_edad"] = pd.cut(paises["Edad"], bins, labels = names)

paises
```

Out[13]:

	Edad	Clase_trabajo	¿?	Nivel_eduacion	Codigo_nivel_educacion	Estado_civil	C
0	39	State-gov	77516	Bachelors	13	Never-married	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	I
2	38	Private	215646	HS-grad	9	Divorced	
3	53	Private	234721	11th	7	Married-civ-spouse	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	
...	...	...	...	...	...	...	
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	
32558	58	Private	151910	HS-grad	9	Widowed	
32559	22	Private	201490	HS-grad	9	Never-married	
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	I

32561 rows × 16 columns

In [ ]:

In [14]:

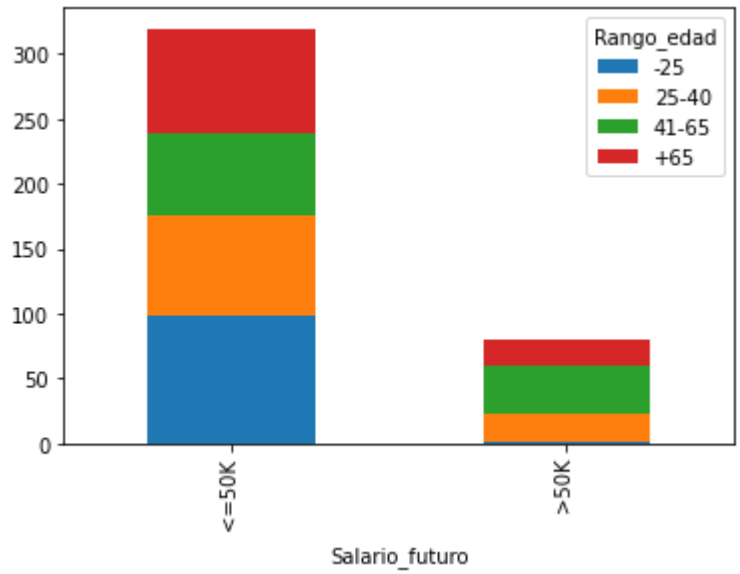
```
pd.crosstab(index=paises['Salario_futuro'], columns=paises['Rango_edad']
          ).apply(lambda r: r/r.sum() *100,
                                axis=0)
```

Out[14]:

Rango_edad	-25	25-40	41-65	+65
Salario_futuro				
<=50K	98.821256	77.657219	63.207163	79.879102
>50K	1.178744	22.342781	36.792837	20.120898

In [15]:

```
plot = pd.crosstab(index=paises['Salario_futuro'],
           columns=paises['Rango_edad']
                 ).apply(lambda r: r/r.sum() *100,
                         axis=0).plot(kind='bar', stacked=True)
```



En este grafico podemos ver que los que cobran menos de 50k son mayoria los menores de 25 años, seguidos de los mayores de 65 y los que cobran mas de 50k son los que tienen una edad comprendida entre los 41 y 65 años, seguidos de los que tienen entre 25 y 40 años.

Podemos decir que ser menor de 25 años es determinante a la hora de tener un salario mas bajo. Mientres que tener una edad media 40-65 años, lo es para tener una sueldo alto.

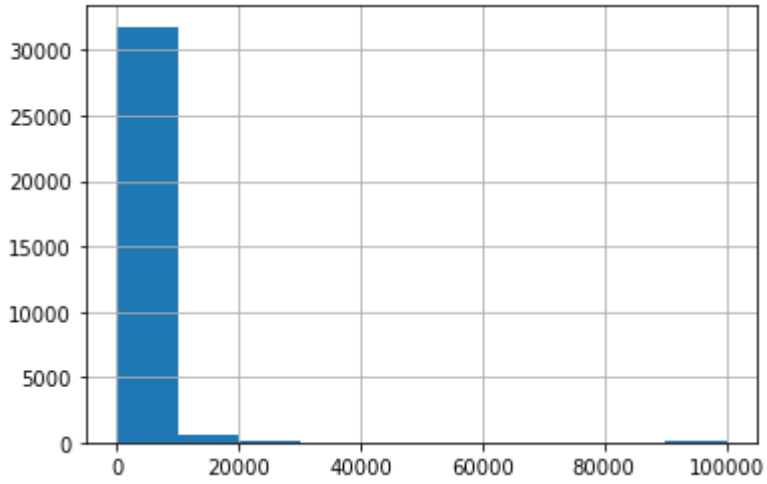
2 - Dinero_ganado

In [16]:

```
paises.Dinero_ganado.hist()

# Podemos ver que la gran mayoria de los analizados en este dataset tienen un capital e
ntre 0 y 10000
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1291557cfa0>
```

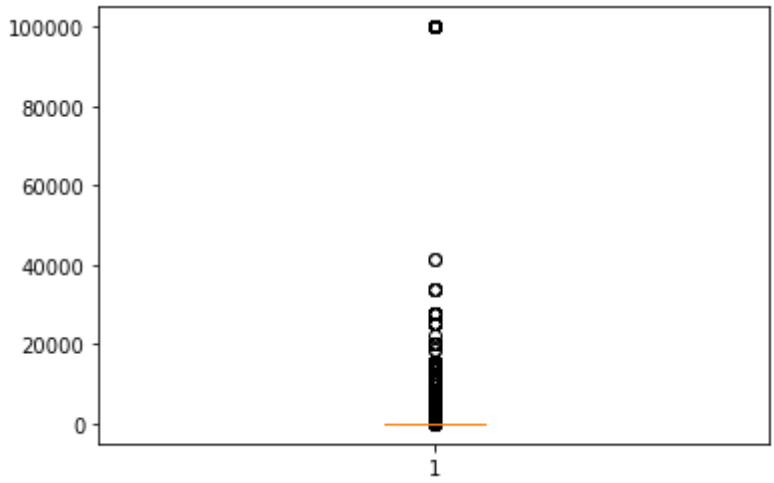


In [17]:

```
plt.boxplot(paises.Dinero_ganado)

# Nos encontramos con un diagrama de caja muy extraño, donde no hay percentiles. Esto p
uede deberse a que la mayoria de los datos
# que tenemos de dinero ganado es 0, que puede deberse a que se desconoce o que se ha g
anado nada
```

Out[17]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x12915637b50>,
  <matplotlib.lines.Line2D at 0x12915637eb0>],
 'caps': [<matplotlib.lines.Line2D at 0x12915644250>,
  <matplotlib.lines.Line2D at 0x129156445b0>],
 'boxes': [<matplotlib.lines.Line2D at 0x129156377f0>],
 'medians': [<matplotlib.lines.Line2D at 0x12915644910>],
 'fliers': [<matplotlib.lines.Line2D at 0x12915644c10>],
 'means': []}
```



In [18]:

```
print("El ", len(paises[paises["Dinero_ganado"] == 0]["Dinero_ganado"]) / len(paises) *
100, "de las personas analizadas presentan 0euros ganados")
```

```
El  91.67101747489328 de las personas analizadas presentan 0euros ganados
```

Estamos ante una variable que en la mayoria de casos es 0, puede que no sea muy util

In [19]:

```
paises["Dinero_ganado"].describe()
```

Out[19]:

```
count    32561.000000
mean      1077.648844
std       7385.292085
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max      99999.000000
Name: Dinero_ganado, dtype: float64
```

In [20]:

```
# Para ver una tabla de frecuencias de como se comporta esta variable, vamos a generar
 unos rangos de dinero ganado
# estos rangos seran de 0 - 1, 2-10000, 10001 - 50000, 50001 - 99999

bins = [0,0.9, 10000, 50000, 99999]

names = ["0", "1-10000", "10001 - 50000", "50001 - 99999"]

paises["Rango_dinero_ganado"] = pd.cut(paises["Dinero_ganado"], bins, labels = names)
```

In [ ]:

```

```

In [21]:

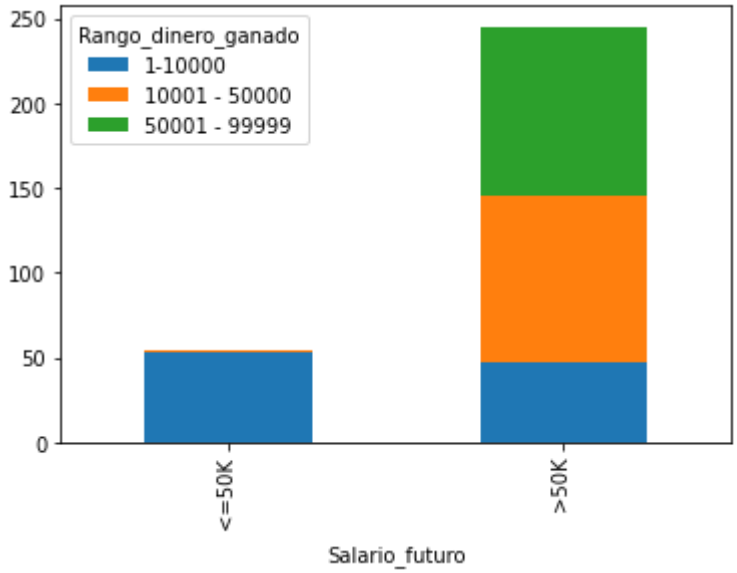
```
# ¿Son los que han ganado mas dinero, los que tendran un salario mas alto?

pd.crosstab(index=paises['Salario_futuro'], columns=paises['Rango_dinero_ganado']
           ).apply(lambda r: r/r.sum() *100,
                                  axis=0)
```

Out[21]:

Rango_dinero_ganado	1-10000	10001 - 50000	50001 - 99999
Salario_futuro			
<=50K	52.574665	2.291326	0.0
>50K	47.425335	97.708674	100.0

In [22]:

```
plot = pd.crosstab(index=paises['Salario_futuro'],
            columns=paises['Rango_dinero_ganado']
                  ).apply(lambda r: r/r.sum() *100,
                          axis=0).plot(kind='bar', stacked=True)
```



Como puede observarse en la tabla y en el grafico, los que han ganado menos dinero, tendran un empleo peor pagado y casi todos los que han ganado mas de 10000 euros, tendran un salario mayor de 50000.

Por lo tanto esta variable parece bastante significativa, el problema es que hay muchos valores no reflejados.

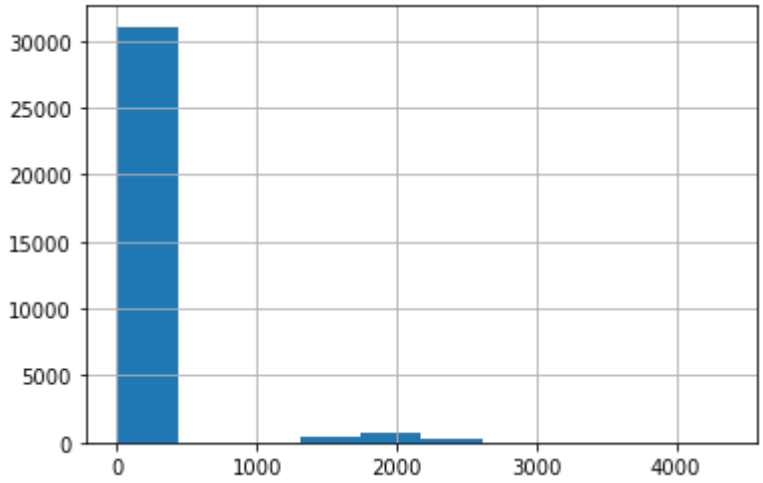
3 - Dinero_perdido

In [23]:

```
paises["Dinero_perdido"].hist()

# igual que antes, la moyoria parece que tienen perdidas cercanas a 0
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x12905acf250>
```



In [24]:

```
print("El ", len(paises[paises["Dinero_perdido"] == 0]["Dinero_perdido"]) / len(paises)
* 100, "de las personas analizadas presentan 0euros perdidos")
```

```
El  95.33490986149073 de las personas analizadas presentan 0euros perdidos
```

4 - Horas_semanales_trabajo

In [25]:

```
# vamos a analizar ahora la variable hora de trabajo semanales, que comprenden las hora
s de trabajo que realizan cada uno de
# las personas que se incluyen en este dataset.

# En primer lugar, vamos a ver como se distribuyen las horas semanales

paises["Horas_semanales_trabajo"].describe()

# Vemos que el minimo trabajado es 1 hora y el que mas horas ha realizado son 99
```

Out[25]:

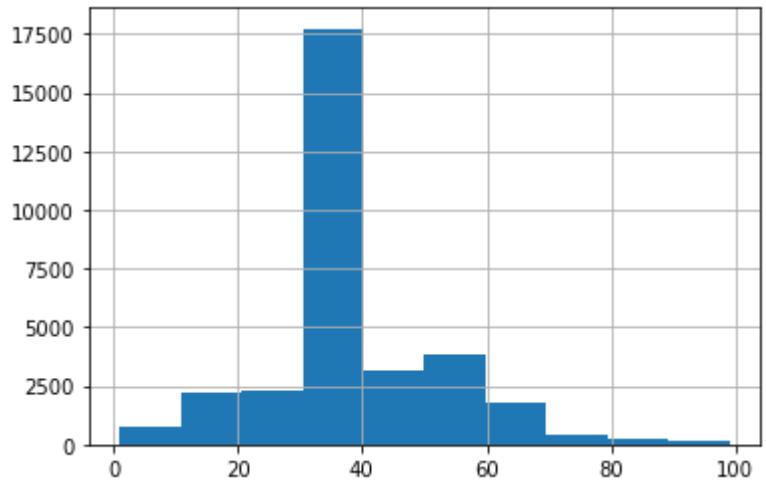
```
count    32561.000000
mean        40.437456
std         12.347429
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000
Name: Horas_semanales_trabajo, dtype: float64
```

In [26]:

```
paises["Horas_semanales_trabajo"].hist()

# La frecuencia de horas mas habitaul es la de 30-40 horas semanales
```

Out[26]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1291574f7f0>
```

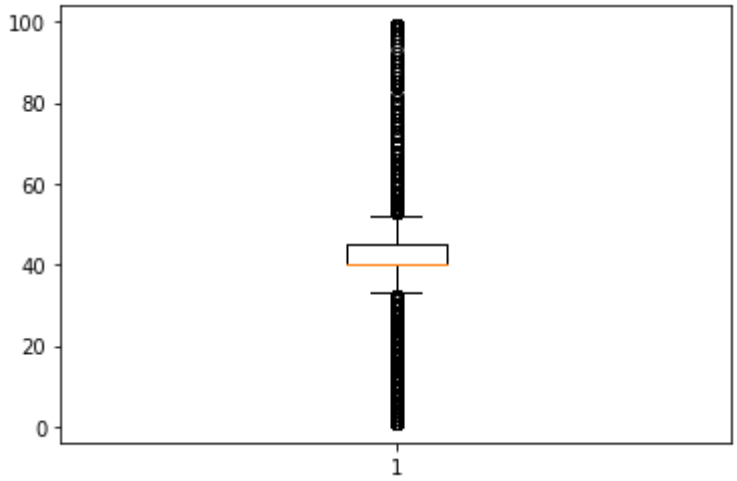


In [27]:

```
# vamos a ver un grafico de cajas

plt.boxplot(paises["Horas_semanales_trabajo"])

# Vemos como la gran mayoria de las horas trabajadas se concrentran entre las 30 - 50 y
tenemos muchos valores atipicos.
```

Out[27]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x129157efeb0>,
  <matplotlib.lines.Line2D at 0x129157fb250>],
 'caps': [<matplotlib.lines.Line2D at 0x129157fb5b0>,
  <matplotlib.lines.Line2D at 0x129157fb910>],
 'boxes': [<matplotlib.lines.Line2D at 0x129157efb50>],
 'medians': [<matplotlib.lines.Line2D at 0x129157fbc70>],
 'fliers': [<matplotlib.lines.Line2D at 0x129157fbf70>],
 'means': []}
```



In [28]:

```
# ¿Afectan las horas semanales al salario?, los que mas horas trabajan tienen un salari
o mayor ? Vamos a verlo

# Igual que antes, creamos una nueva variable que contenga los rangos de horas trabajad
as: 1 - 25, 26 - 40, 41 - 60, 60 - 99


bins = [1, 25, 40, 60, 99]

names = ["1 - 25", "26 - 40", "41 - 60", "60 - 99"]


paises["Rango_h/semanales"] = pd.cut(paises["Horas_semanales_trabajo"], bins, labels =
names)
```

In [29]:

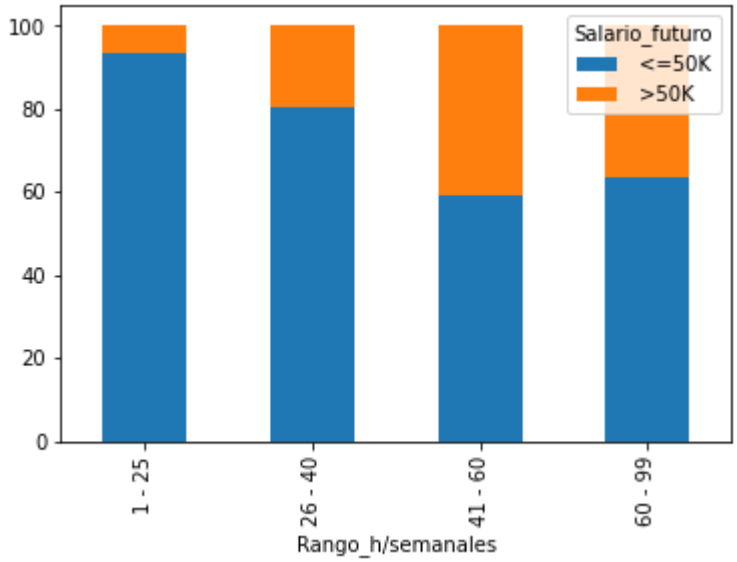
```
pd.crosstab(index=paises['Salario_futuro'], columns=paises["Rango_h/semanales"]
          ).apply(lambda r: r/r.sum() *100,
                                axis=0)
```

Out[29]:

Rango_h/semanales	1 - 25	26 - 40	41 - 60	60 - 99
Salario_futuro				
<=50K	93.448891	80.427588	59.249203	63.603604
>50K	6.551109	19.572412	40.750797	36.396396

In [30]:

```
plot = pd.crosstab(index=paises["Rango_h/semanales"],
           columns=paises["Salario_futuro"]
                 ).apply(lambda r: r/r.sum() *100,
                         axis=1).plot(kind='bar', stacked=True)
```



Como se puede observar en la grafica, las personas que tienen un salario mayor de 50k son aquellos que pertenecen a la categoria de mas horas trabajadas(41-60 y 60-99).

Por lo tanto las horas trabajadas si son relevantes a la hora del futuro salario, a mas horas trabajadas, mayor salario.

5 - Clase_trabajo

In [31]:

```
paises.Clase_trabajo.unique()
```

Out[31]:

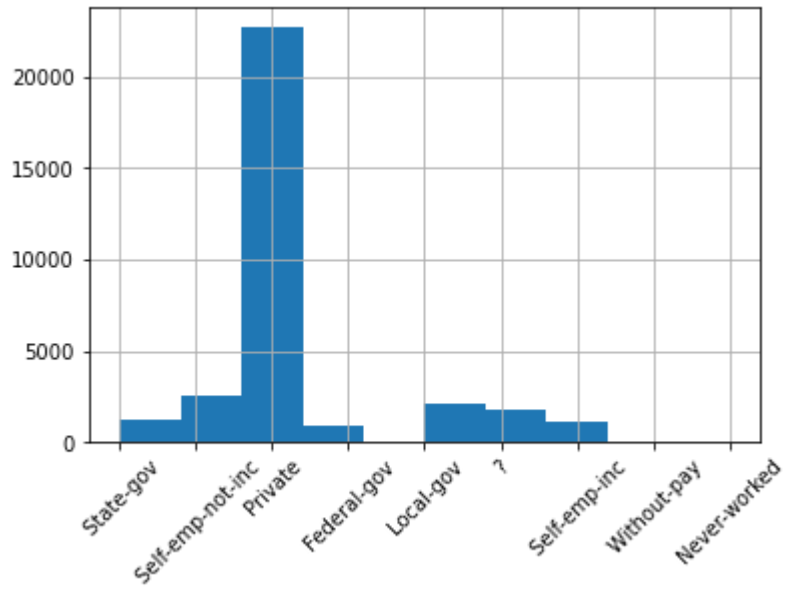
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

In [33]:

```
# Vamos a ver las frecuencias de las distintas clases de trabajo

paises.Clase_trabajo.hist()
plt.xticks(rotation=45)

# Podemos ver como el trabajo privado(en sector privado) es el que destaca entre las pe
rsonas de este dataset
```

Out[33]:

```
([0, 1, 2, 3, 4, 5, 6, 7, 8], <a list of 9 Text major ticklabel objects>)
```



In [34]:

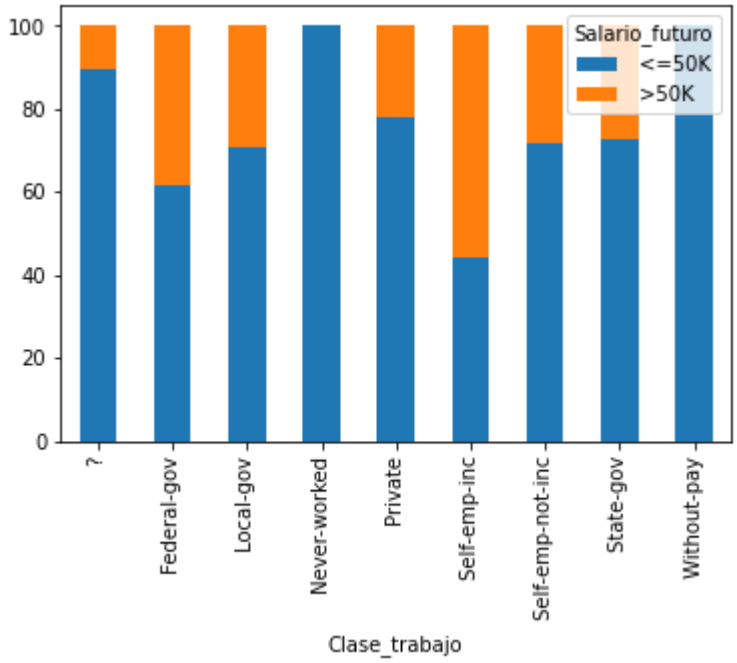
```
# Influye la clase de trabajo a la hora de cobrar mas de 50K??, que clase de trabajo es
mas propenso a sueldos mayores??

pd.crosstab(index=paises['Salario_futuro'], columns=paises["Clase_trabajo"]
          ).apply(lambda r: r/r.sum() *100,
                                axis=0)
```

Out[34]:

Clase_trabajo	?	Federal-gov	Local-gov	Never-worked	Private	Self-emp-inc	Self-emp-not-inc	Sta
Salario_futuro								
<=50K	89.59695	61.354167	70.520784	100.0	78.132711	44.265233	71.507281	72.8
>50K	10.40305	38.645833	29.479216	0.0	21.867289	55.734767	28.492719	27.1

In [43]:

```
plot = pd.crosstab(index=paises["Clase_trabajo"],
           columns=paises["Salario_futuro"]
                 ).apply(lambda r: r/r.sum() *100,
                         axis=1).plot(kind='bar', stacked=True)



# En este grafico podemos ver como los autonomos(self-emp-inc) y los trabajadores para
 el gobierno federal son los mas
# propensos a tener un salario mayor de 50k
```



6 - Nivel_eduacion

In [44]:

```
paises["Nivel_eduacion"].unique()
```

Out[44]:

```
array([' Bachelors', ' HS-grad', ' 11th', ' Masters', ' 9th',
       ' Some-college', ' Assoc-acdm', ' Assoc-voc', ' 7th-8th',
       ' Doctorate', ' Prof-school', ' 5th-6th', ' 10th', ' 1st-4th',
       ' Preschool', ' 12th'], dtype=object)
```

In [46]:

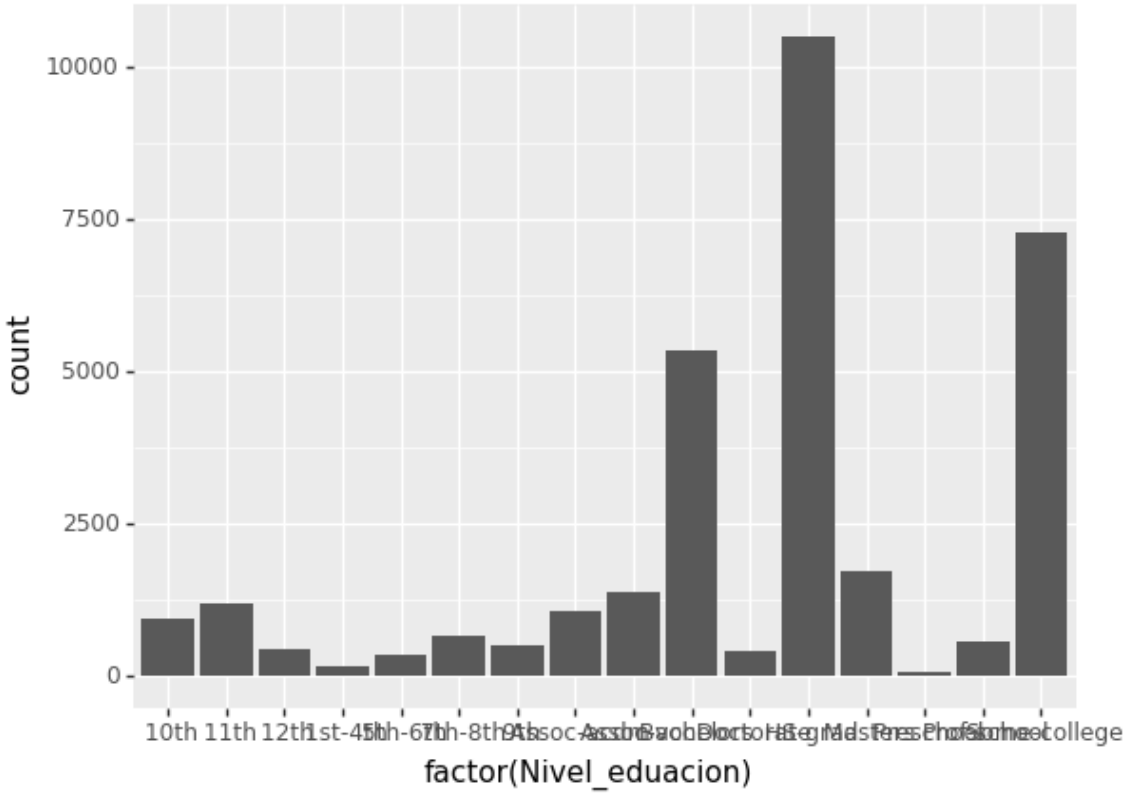
```
pd.value_counts(paises["Nivel_eduacion"])
```

Out[46]:

```
 HS-grad         10501
 Some-college     7291
 Bachelors        5355
 Masters          1723
 Assoc-voc        1382
 11th             1175
 Assoc-acdm       1067
 10th              933
 7th-8th           646
 Prof-school       576
 9th               514
 12th              433
 Doctorate         413
 5th-6th           333
 1st-4th           168
 Preschool          51
Name: Nivel_eduacion, dtype: int64
```

In [62]:

```
# Vamos a ver cuales son los niveles de educacion mas frecuentes
import plotnine as p9
(p9.ggplot(data=paises,
           mapping=p9.aes(x='factor(Nivel_eduacion)'))
    + p9.geom_bar()
)
```



Out[62]:

<ggplot: (79800455159)>

In [63]:

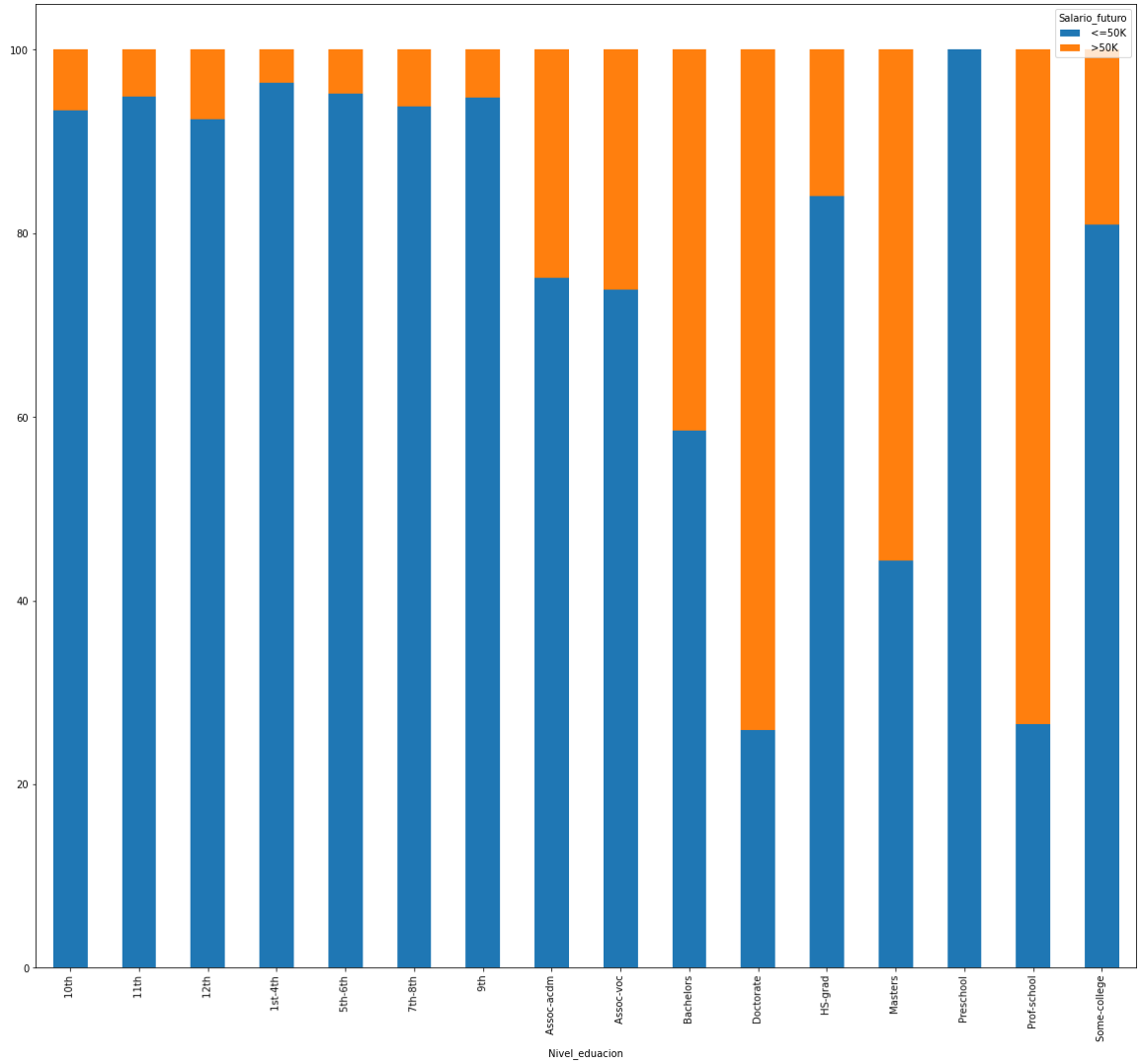
```
# ¿Es el nivel de educacion determinante a la hora de tener un salario mayor de 50k? qu
e estudios son necesarios??

pd.crosstab(index=paises['Salario_futuro'], columns=paises["Nivel_eduacion"]
           ).apply(lambda r: r/r.sum() *100,
                                axis=0)
```

Out[63]:

Nivel_eduacion	10th	11th	12th	1st-4th	5th-6th	7th-8th	9th	
Salario_futuro								
<=50K	93.35477	94.893617	92.378753	96.428571	95.195195	93.80805	94.747082	7
>50K	6.64523	5.106383	7.621247	3.571429	4.804805	6.19195	5.252918	2

In [64]:

```
plot = pd.crosstab(index=paises["Nivel_eduacion"],
            columns=paises["Salario_futuro"]
                  ).apply(lambda r: r/r.sum() *100,
                          axis=1).plot(kind='bar', stacked=True)

# Podemos ver en el grafico que las personas que son mas propensas a tener un salario m
ayor de 50k son aquellos que tienen
# un doctorado, masters o estudios en escuela profesional
```


Salario_futuro
<=50K
>50K
100
80
60
40
20
0
10th
11th
12th
1st-4th
5th-6th
7th-8th
9th
Assoc-acdm
Assoc-voc
Bachelors
Doctorate
HS-grad
Masters
Preschool
Prof-school
Some-college
Nivel_eduacion

7 - Raza

In [65]:

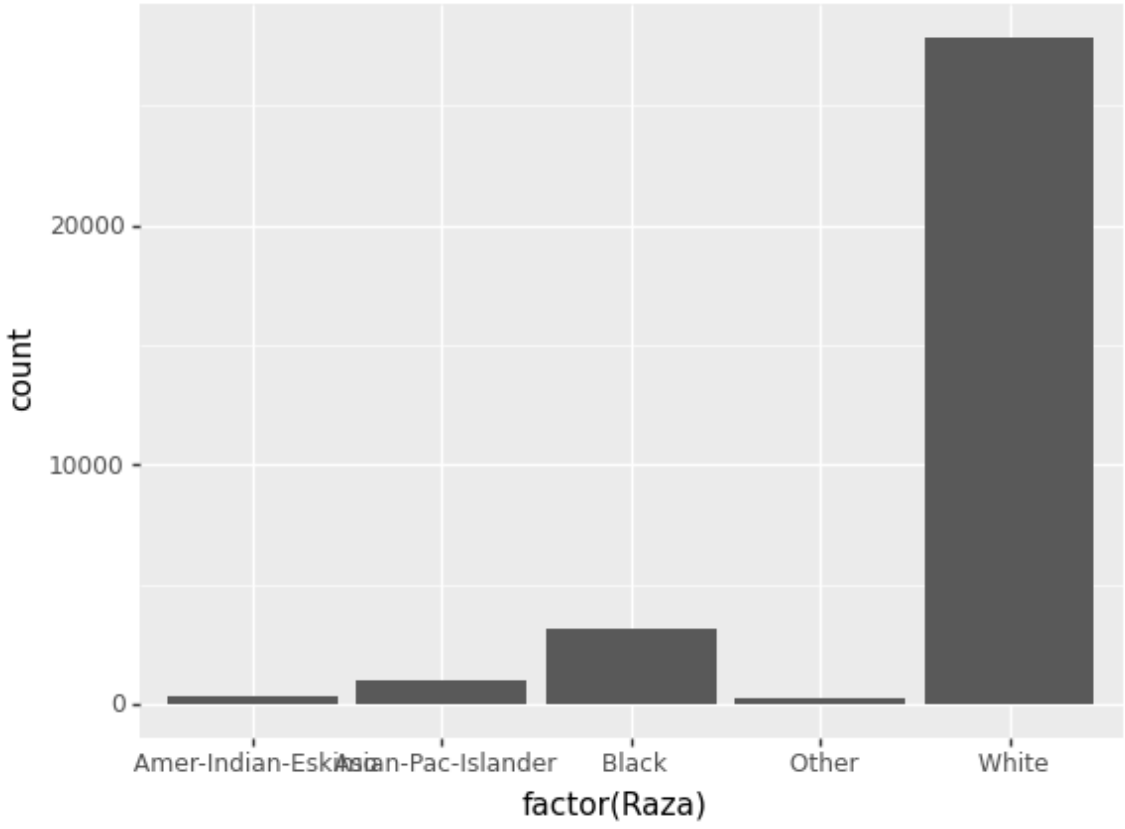
```
paises["Raza"].unique()
pd.value_counts(paises["Raza"])
```

Out[65]:

```
 White                 27816
 Black                  3124
 Asian-Pac-Islander     1039
 Amer-Indian-Eskimo      311
 Other                   271
Name: Raza, dtype: int64
```

In [66]:

```
(p9.ggplot(data=paises,
           mapping=p9.aes(x='factor(Raza)'))
    + p9.geom_bar()
)
```



Out[66]:

```
<ggplot: (79828547708)>
```

In [67]:

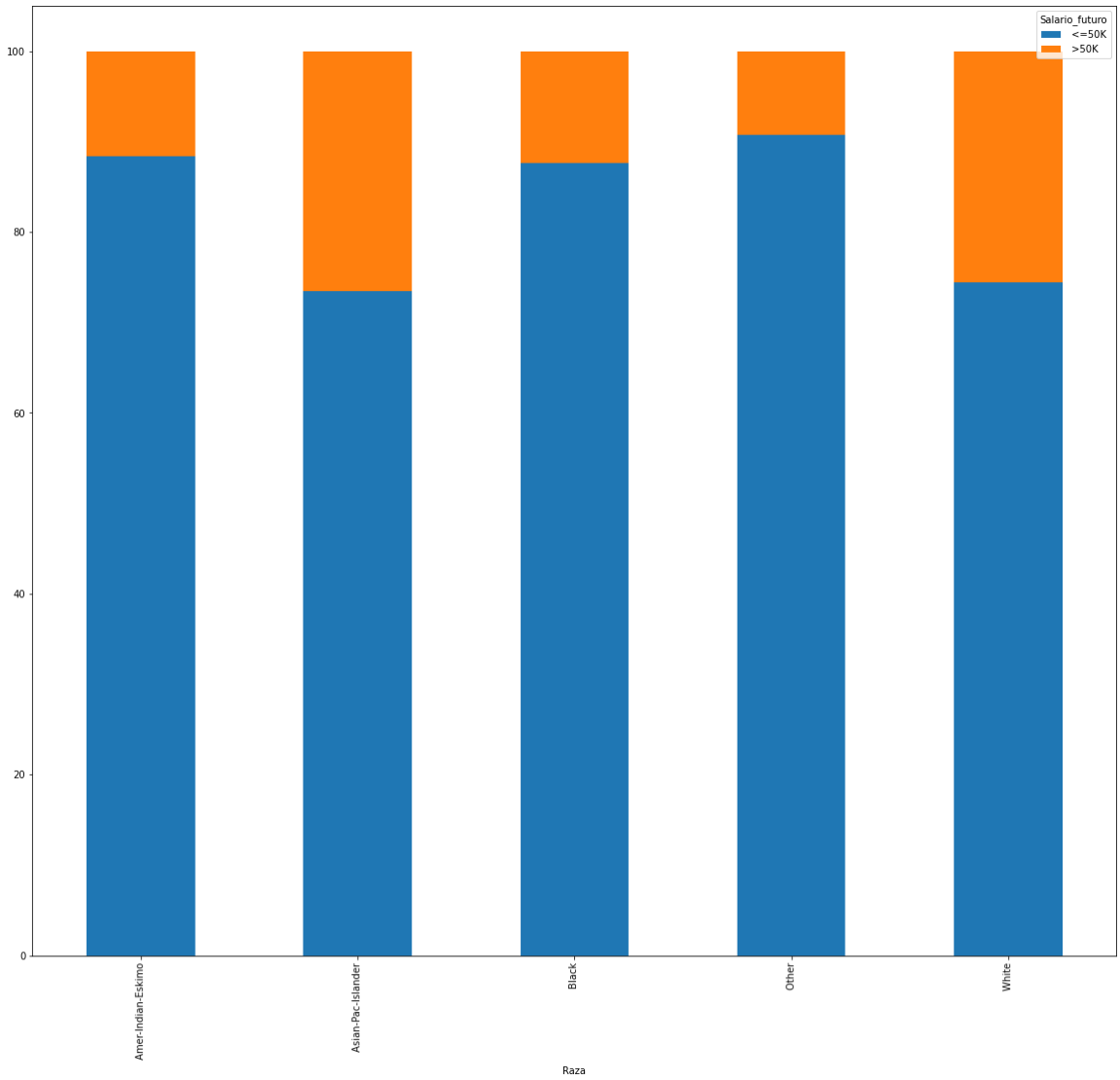
```
# ¿Es la raza determinante a la hora de tener un salario mayor a 50k?

pd.crosstab(index=paises['Salario_futuro'], columns=paises["Raza"]
          ).apply(lambda r: r/r.sum() *100,
                                axis=0)
```

Out[67]:

Raza Salario_futuro	Amer-Indian-Eskimo	Asian-Pac-Islander	Black	Other	White
<=50K	88.424437	73.435996	87.612036	90.774908	74.414006
>50K	11.575563	26.564004	12.387964	9.225092	25.585994

In [68]:

```
plot = pd.crosstab(index=paises["Raza"],
            columns=paises["Salario_futuro"]
                  ).apply(lambda r: r/r.sum() *100,
                          axis=1).plot(kind='bar', stacked=True)

# Las razas que alcanzan de forma mas habitual salarios mayores a 50k son la blanca y l
a asiatica
```



8 - Sexo

In [69]:

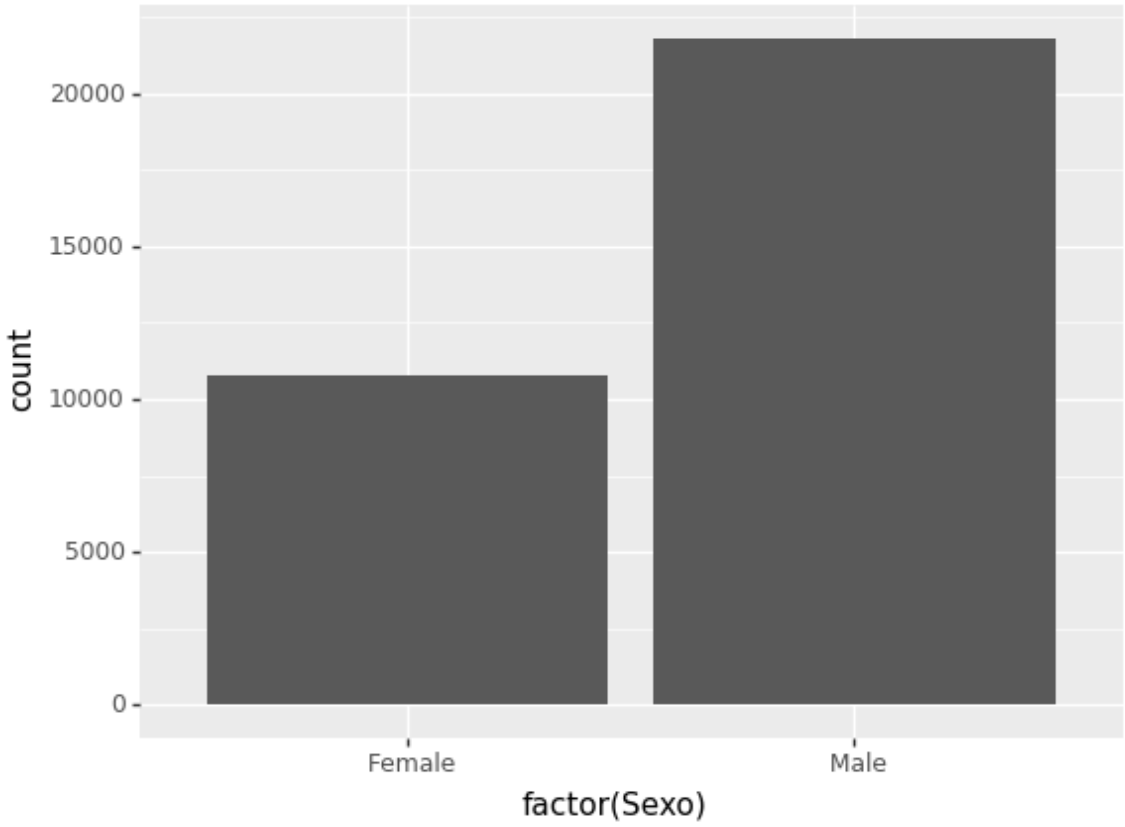
```
pd.value_counts(paises["Sexo"])
```

Out[69]:

```
 Male      21790
 Female    10771
Name: Sexo, dtype: int64
```

In [70]:

```
(p9.ggplot(data=paises,
           mapping=p9.aes(x='factor(Sexo)'))
    + p9.geom_bar()
)
```



Out[70]:

```
<ggplot: (79832036241)>
```

In [71]:

```
pd.crosstab(index=paises['Salario_futuro'], columns=paises["Sexo"]
          ).apply(lambda r: r/r.sum() *100,
                                axis=0)
```

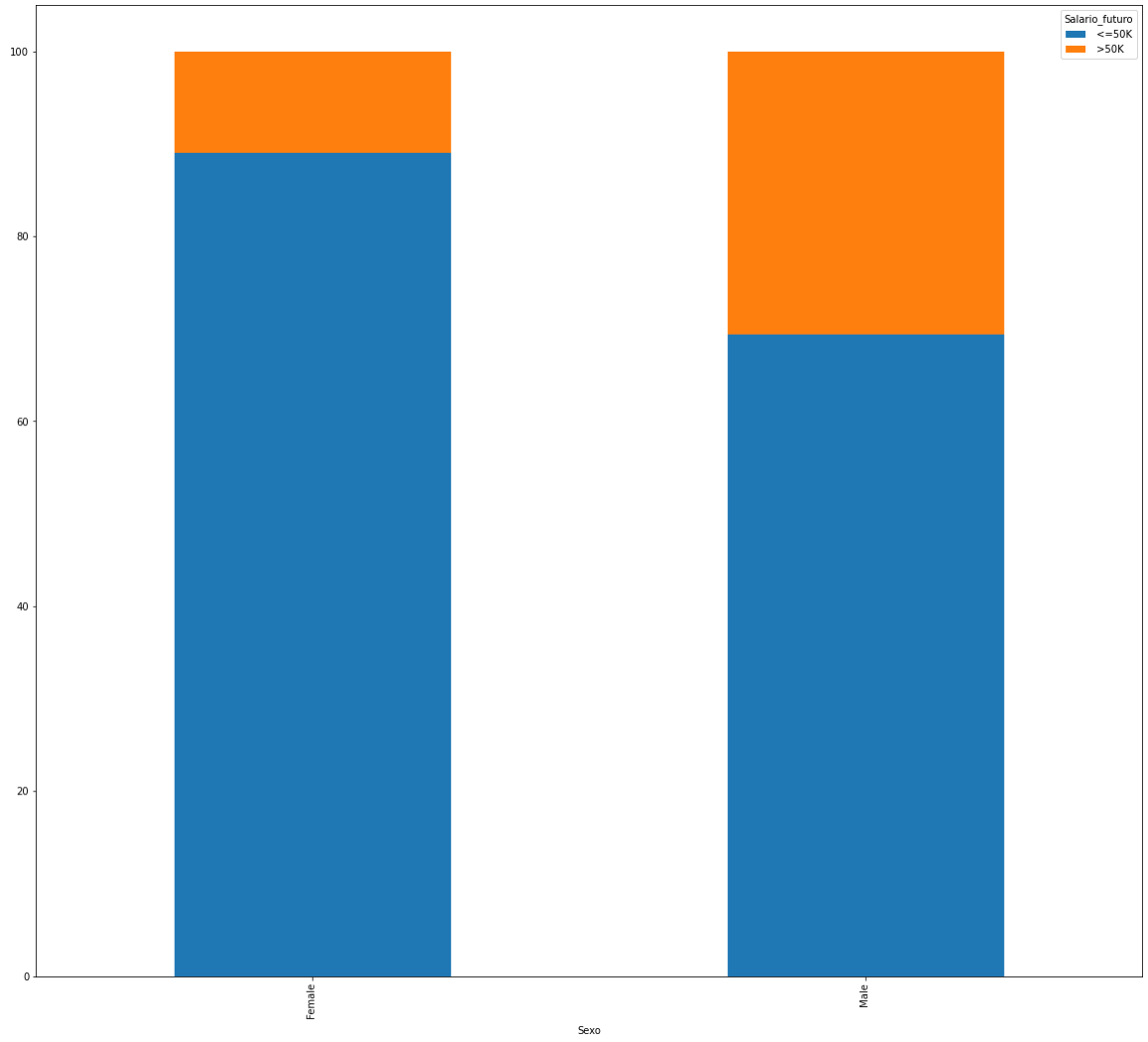
Out[71]:

Sexo	Female	Male
Salario_futuro		
<=50K	89.053941	69.426342
>50K	10.946059	30.573658

In [74]:

```
plt.figure(figsize=(8,4))
plot = pd.crosstab(index=paises["Sexo"],
            columns=paises["Salario_futuro"]
                  ).apply(lambda r: r/r.sum() *100,
                          axis=1).plot(kind='bar', stacked=True)

# Claramente los hombres son mas propensos a alcanzar un salario de mas de 50k
```

```
<Figure size 576x288 with 0 Axes>
```



Tendriamos que analizar todas las variables pero como estamos practicando no lo vamos a hacer y directamente vamos a pasar a otra parte

Estamos ante un dataset con una variable dependiente, Salario futuro, de tipo categorico por lo tanto si quieremos predecir si el salario será mayor o menor de 50k, tenemos que aplicar uno de los metodos o algoritos de clasifiacion del aprendizaje supervisado.

Entre estos algoritmos nos encontramos con:

1 - Regresion logistica:Es un algoritmo sensible a los autliers o valores atipicos, hay que estandarizar datos

2 - KNN: Se usa cuando el df es pequeño y tiene pocas variables

3 - Arboles de decision: Los datos no hay que estandarizarlos, NO son sensible a outliers, no hay que dummificar las variables categoricas.

4 - Random forest: No es necesaria una interpretacion de los datos, es uno de los mejores algoritmos por rendimiento y rapidez

5 - XGBOOST: es un algoritmo caja negra, es una de los mejores, no es necesario mucho procesamiento de datos

6 - Maquina de soporte vectorial: Necesita tiempo para ejecutarse, modelo caja negra, es util para dataset pequeños

In [77]:

```
paises.dtypes
```

Out[77]:

```
Edad                         int64
Clase_trabajo               object
¿?                           int64
Nivel_eduacion              object
Codigo_nivel_educacion       int64
Estado_civil                object
Ocupacion                   object
Relacion_sentimental        object
Raza                        object
Sexo                        object
Dinero_ganado                int64
Dinero_perdido               int64
Horas_semanales_trabajo      int64
Pais_origen                 object
Salario_futuro              object
Rango_edad                category
Rango_dinero_ganado       category
Rango_h/semanales         category
dtype: object
```

In [76]:

```
# Creamos los valores X e y

paises.columns
```

Out[76]:

```
Index(['Edad', 'Clase_trabajo', '¿?', 'Nivel_eduacion',
       'Codigo_nivel_educacion', 'Estado_civil', 'Ocupacion',
       'Relacion_sentimental', 'Raza', 'Sexo', 'Dinero_ganado',
       'Dinero_perdido', 'Horas_semanales_trabajo', 'Pais_origen',
       'Salario_futuro', 'Rango_edad', 'Rango_dinero_ganado',
       'Rango_h/semanales'],
      dtype='object')
```

In [98]:

```
paises = paises.astype({"Sexo": 'category', "Pais_origen": 'category', "Salario_futuro"
: "category"})



paises.dtypes
```

Out[98]:

```
Edad                       int64
Clase_trabajo           category
¿?                         int64
Nivel_eduacion          category
Codigo_nivel_educacion     int64
Estado_civil            category
Ocupacion               category
Relacion_sentimental    category
Raza                    category
Sexo                    category
Dinero_ganado              int64
Dinero_perdido             int64
Horas_semanales_trabajo    int64
Pais_origen             category
Salario_futuro          category
Rango_edad              category
Rango_dinero_ganado     category
Rango_h/semanales       category
dtype: object
```

Intento 1 - Random forest

En este primer intento SOLO vamos a utilizar las variables numericas de tipo INT, ya que el random forest, no permite variables que contengan string

In [118]:

```
X = paises[['Edad', '¿?',
       'Codigo_nivel_educacion','Dinero_ganado',
       'Dinero_perdido', 'Horas_semanales_trabajo']]

y = paises['Salario_futuro']
```

In [124]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42)
```

In [125]:

```
clf=RandomForestClassifier(n_estimators=100)
```

In [126]:

```
params = {"n_estimators" : [10,50,100,200],
         "max_depth" : [5,7,9],
         "max_features": [10, "sqrt"]}
```

In [145]:

```
clf_grid = GridSearchCV(clf, params, cv=3, n_jobs = -1)
```

In [146]:

```
clf_grid.fit(X_train,y_train)
```

Out[146]:

```
GridSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_depth': [5, 7, 9], 'max_features': [10, 'sqr
t'],
                         'n_estimators': [10, 50, 100, 200]})
```

In [149]:

```
print("Mejores parametros del modelo: {}".format(clf_grid.best_estimator_.score))

print("Acurrancy de los datos de train: {} \n - Acurracy de los datos de test: {}"\
      .format(clf_grid.score(X_train,y_train),
     round(clf_grid.score(X_test, y_test),2)))
```

```
Mejores parametros del modelo: <bound method ClassifierMixin.score of Rand
omForestClassifier(max_depth=9, max_features='sqrt', n_estimators=200)>
Acurrancy de los datos de train: 0.8432841932841932
 - Acurracy de los datos de test: 0.84
```

In [161]:

```
clf_grid.predict([[31, 13600, 9, 30000, 5000, 40]])
```

Out[161]:

```
array([' >50K'], dtype=object)
```

INTENTO 2 - REGRESION LOGISTICA

In [170]:

```
# para poder realizar regresion logistica, necesitamos dumificar las variables categoricas o mapearlas

# Ejemplo de dumificacion

pd.get_dummies(paises, columns = ["Sexo"])
```

Out[170]:

	Edad	Clase_trabajo	¿?	Nivel_eduacion	Codigo_nivel_educacion	Estado_civil	C
0	39	State-gov	77516	Bachelors	13	Never-married	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	
2	38	Private	215646	HS-grad	9	Divorced	
3	53	Private	234721	11th	7	Married-civ-spouse	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	
...	...	...	...	...	...	...	
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	
32558	58	Private	151910	HS-grad	9	Widowed	
32559	22	Private	201490	HS-grad	9	Never-married	
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	

32561 rows × 19 columns

In [176]:

```
# Ejemplo de mapeo de variables

paises.Clase_trabajo.unique()

codigo_trabajo = {' State-gov':1, ' Self-emp-not-inc':2, ' Private':3, ' Federal-gov':4,' Local-gov':5,  ' ?':6, ' Self-emp-inc':7, ' Without-pay':8, ' Never-worked':9}
```

In [179]:

```
# trabajo

paises["Codigo_trabajo"] = paises["Clase_trabajo"].map(codigo_trabajo)
```

In [227]:

```
paises
```

Out[227]:

	Edad	Clase_trabajo	¿?	Nivel_eduacion	Codigo_nivel_educacion	Estado_civil	C
0	39	State-gov	77516	Bachelors	13	Never-married	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	I
2	38	Private	215646	HS-grad	9	Divorced	
3	53	Private	234721	11th	7	Married-civ-spouse	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	
...	...	...	...	...	...	...	
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	
32558	58	Private	151910	HS-grad	9	Widowed	
32559	22	Private	201490	HS-grad	9	Never-married	
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	I

32561 rows × 24 columns

In [206]:

```
#Educacion

list(paises['Nivel_eduacion'].value_counts().index)

codigo_educacion = {' HS-grad':1,
 ' Some-college':2,
 ' Bachelors':3,
 ' Masters':4,
 ' Assoc-voc':5,
 ' 11th':6,
 ' Assoc-acdm':7,
 ' 10th':8,
 ' 7th-8th':9,
 ' Prof-school':10,
 ' 9th':11,
 ' 12th':12,
 ' Doctorate':13,
 ' 5th-6th':14,
 ' 1st-4th':15,
 ' Preschool':15}

paises["Codigo_educacion"] = paises['Nivel_eduacion'].map(codigo_educacion)
```

In [209]:

```
# raza

list(paises['Raza'].value_counts().index)

codigo_raza = {' White':1, ' Black':2, ' Asian-Pac-Islander':3, ' Amer-Indian-Eskimo':4
, ' Other':5}

paises["Codigo_raza"] = paises['Raza'].map(codigo_raza)
```

In [214]:

```
# sexo

list(paises['Sexo'].value_counts().index)

codigo_sexo = {' Male':0, ' Female':1,}

paises["Codigo_sexo"] = paises['Sexo'].map(codigo_sexo)
```

In [223]:

```
# pais

list(paises['Pais_origen'].value_counts().index)

codigo_pais = {' United-States':1,
 ' Mexico':2,
 ' ?':3,
 ' Philippines':4,
 ' Germany':5,
 ' Canada':6,
 ' Puerto-Rico':7,
 ' El-Salvador':8,
 ' India':9,
 ' Cuba':10,
 ' England':11,
 ' Jamaica':12,
 ' South':13,
 ' China':14,
 ' Italy':15,
 ' Dominican-Republic':16,
 ' Vietnam':17,
 ' Guatemala':18,
 ' Japan':19,
 ' Poland':20,
 ' Columbia':21,
 ' Taiwan':22,
 ' Haiti':23,
 ' Iran':24,
 ' Portugal':25,
 ' Nicaragua':26,
 ' Peru':27,
 ' Greece':28,
 ' France':29,
 ' Ecuador':30,
 ' Ireland':31,
 ' Hong':32,
 ' Cambodia':33,
 ' Trinadad&Tobago':34,
 ' Laos':35,
 ' Thailand':36,
 ' Yugoslavia':37,
 ' Outlying-US(Guam-USVI-etc)':38,
 ' Hungary':39,
 ' Honduras':40,
 ' Scotland':41,
 ' Holand-Netherlands':42}

paises["Codigo_pais"] = paises["Pais_origen"].map(codigo_pais)
```

In [226]:

```
# salario

paises["Salario_futuro"].unique()

codigo_salario = {' <=50K':0, ' >50K':1}

paises["Codigo_salario"] = paises["Salario_futuro"].map(codigo_salario)
```

Creamos la division en X e y

In [237]:

```
X = paises[['Edad', '¿?',
       'Dinero_ganado','Dinero_perdido', 'Horas_semanales_trabajo',
       'Codigo_trabajo', 'Codigo_educacion',
       'Codigo_raza', 'Codigo_sexo', 'Codigo_pais']]

y = paises["Codigo_salario"]
```

In [257]:

```
from sklearn import linear_model

from sklearn.metrics import mean_squared_error, r2_score, precision_score

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25, random_state=12
34, stratify = y)
```

In [239]:

```
modelo = LogisticRegression()
```

In [240]:

```
modelo.fit(X_train, y_train)
```

```
C:\Users\jaime\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
ression
```

Out[240]:

```
LogisticRegression()
```

In [254]:

```
y_test_pred = modelo.predict(X_test)

y_test_prob = modelo.predict_proba(X_test)

print("Ejemplo de prediccion: {}". format(y_test_prob[:1]))
```

```
Ejemplo de prediccion: [[0.80108911 0.19891089]]
```

In [258]:

```
print("Precision: ", round(precision_score(y_test, y_test_pred),2))
```

```
Precision:  0.73
```

In [ ]:

```

```

In [ ]:

```

```