Modelo predictivo para puesto de trabajo en Empresa X como desarrollador web en el área de backend

Yessenia Olvera Angeles

Kevin Israel López Mendoza

Importar librerias

In [ ]:
```
import pandas as pd
```

In [ ]:
```
work_prediction = pd.read_csv('works_oferts.csv')
work_prediction.head()
```

Out[ ]:

	name	edad	exp	node	sql	postgsql	aws	js	ningles	visap	hofice	estudios	probabilid
0	Narmo	25	5	10	5	6	0	9	16.60	1	1	50	60.23
1	Gustavo	21	3	8	9	8	2	8	33.00	0	1	25	29.28
2	Mauricio	26	5	10	5	6	7	9	49.00	1	0	15	43.65
3	Gabriel	30	7	10	8	8	9	10	66.40	1	0	20	81.46
4	x	34	5	10	10	10	10	10	83.06	0	0	15	60.31

ver datos numericos de requisitos

In [ ]:
```
numeric_requirements = ["exp","node","sql","postgsql","aws","js","ningles","visap",
work_prediction[numeric_requirements + ["probabilidad"]].describe()
```

Out[ ]:

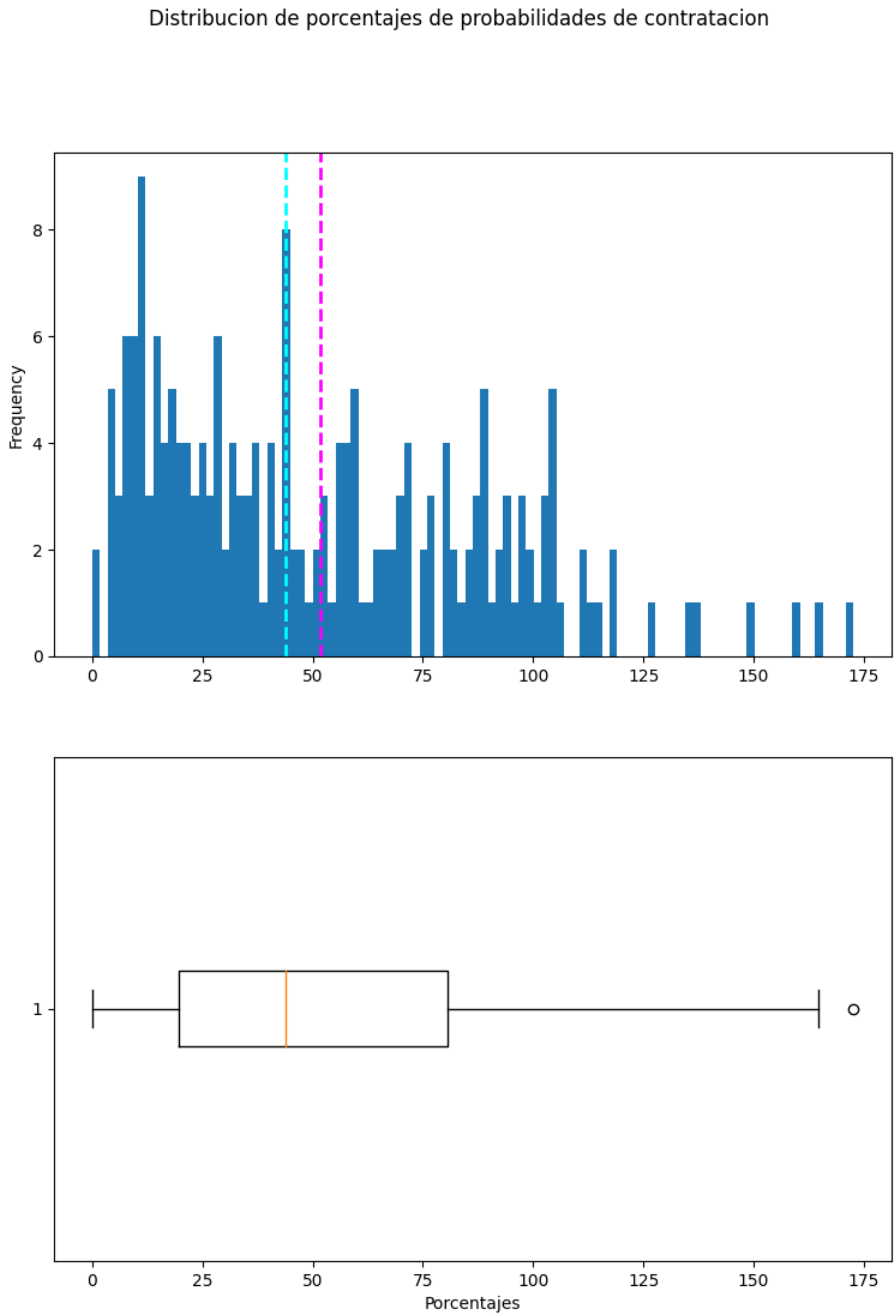
	exp	node	sql	postgsql	aws	js	ningles	visap
count	200.0000	200.000000	200.000000	200.000000	200.000000	200.00000	200.000000	200.00000
mean	5.4000	5.505000	5.485000	5.475000	5.435000	5.52000	43.744700	0.45000
std	2.8725	2.953573	2.924472	2.934695	2.959658	2.95691	28.943021	0.49874
min	0.0000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.00000
25%	3.0000	3.000000	3.000000	3.000000	3.000000	3.00000	16.600000	0.00000
50%	5.0000	5.500000	5.000000	6.000000	5.000000	6.00000	33.000000	0.00000
75%	8.0000	8.000000	8.000000	8.000000	8.000000	8.00000	66.400000	1.00000
max	10.0000	10.000000	10.000000	10.000000	10.000000	10.00000	99.000000	1.00000

trazo de historigrama de los porcentajes

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

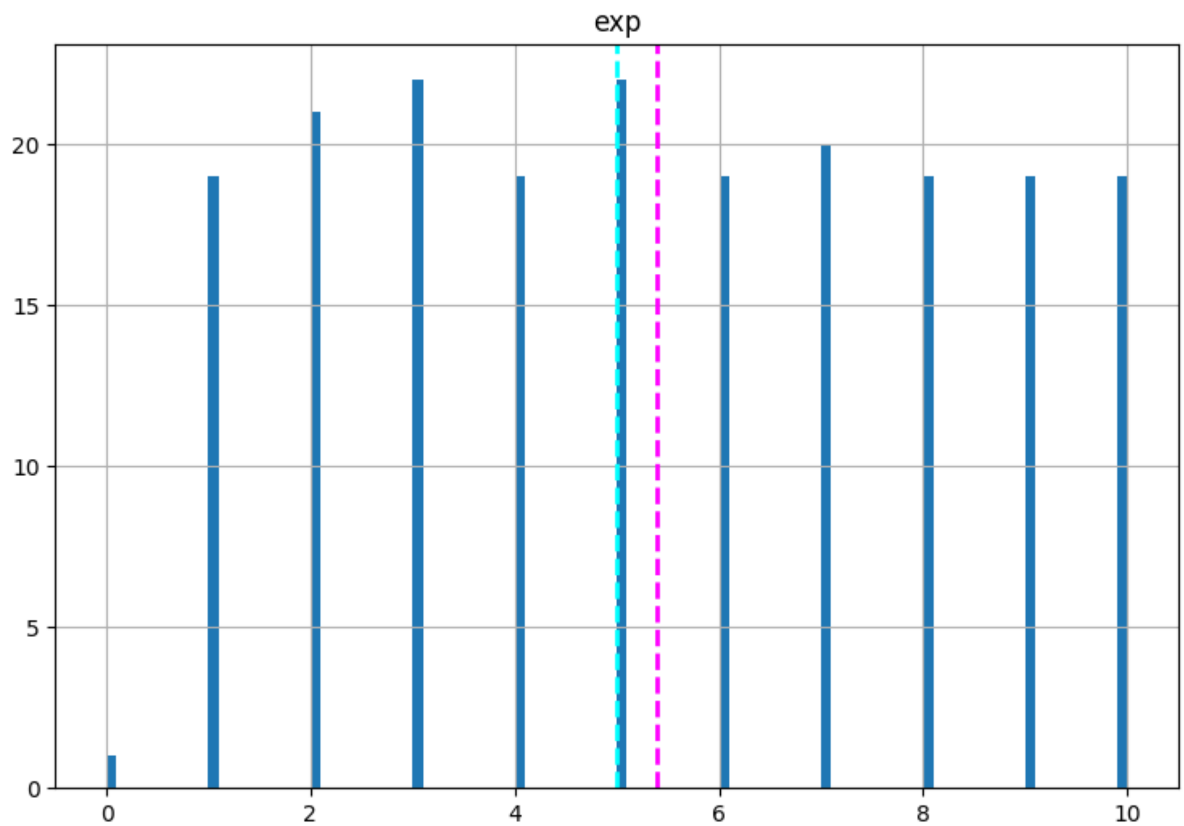
In [ ]:
```
%matplotlib inline
label = work_prediction["probabilidad"]
fig, ax = plt.subplots(2,1, figsize = (9,12))
ax[0].hist(label,bins=100)
ax[0].set_ylabel("Frequency")
ax[0].axvline(label.mean(), color='magenta', linestyle='dashed', linewidth=2)
ax[0].axvline(label.median(), color='cyan', linestyle='dashed', linewidth=2)

ax[1].boxplot(label, vert=False)
ax[1].set_xlabel('Porcentajes')

fig.suptitle('Distribucion de porcentajes de probabilidades de contratacion')

fig.show()
```

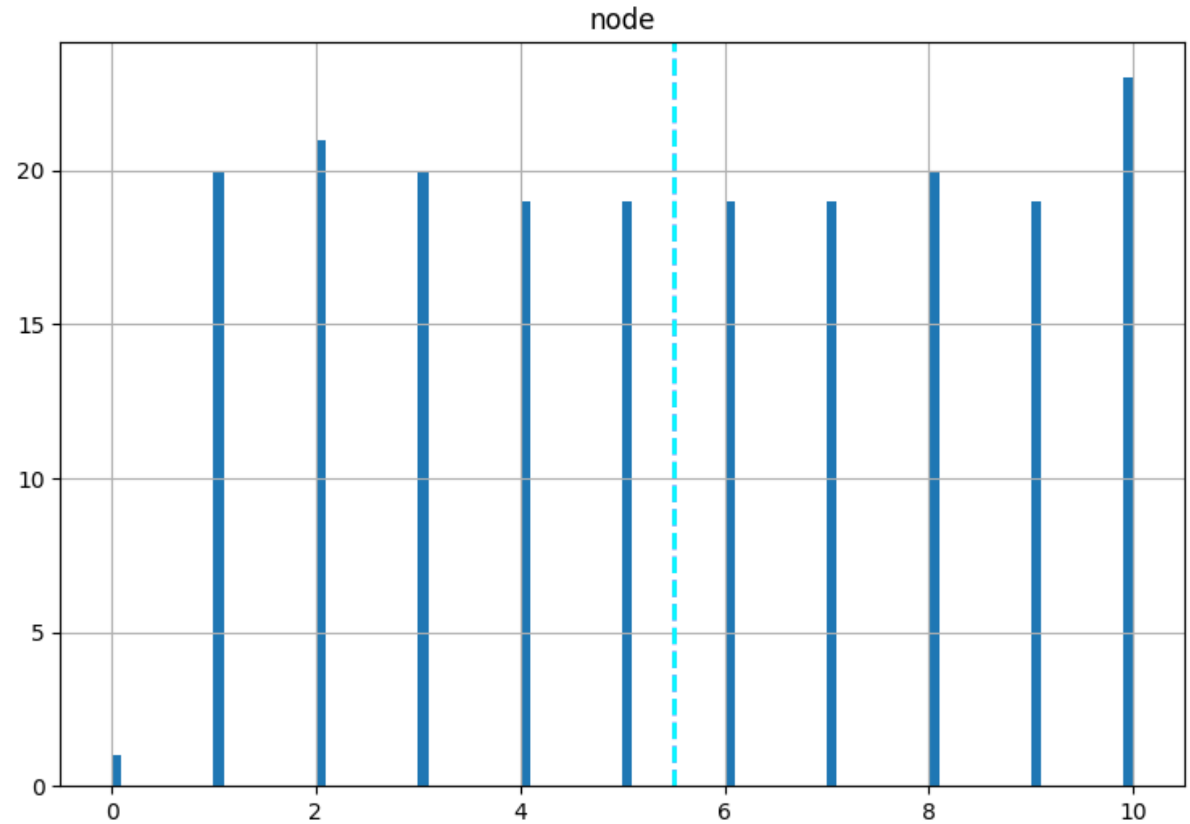
```
C:\Users\lenovo\AppData\Local\Temp\ipykernel_4124\584083157.py:14: UserWarning: Mat
plotlib is currently using module://matplotlib_inline.backend_inline, which is a no
n-GUI backend, so cannot show the figure.
  fig.show()
```

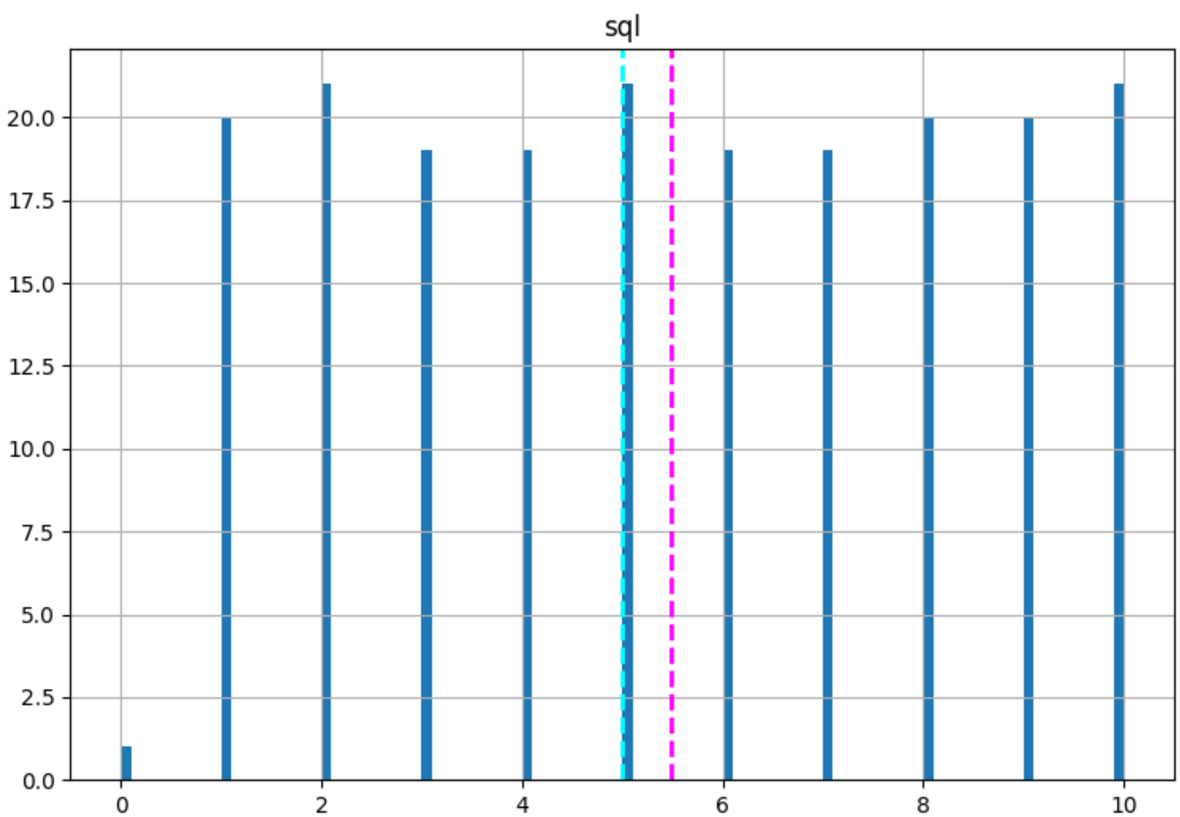

Distribucion de porcentajes de probabilidades de contratacion
Frequency
0
2
4
6
8
0
25
50
75
100
125
150
175
1
Porcentajes

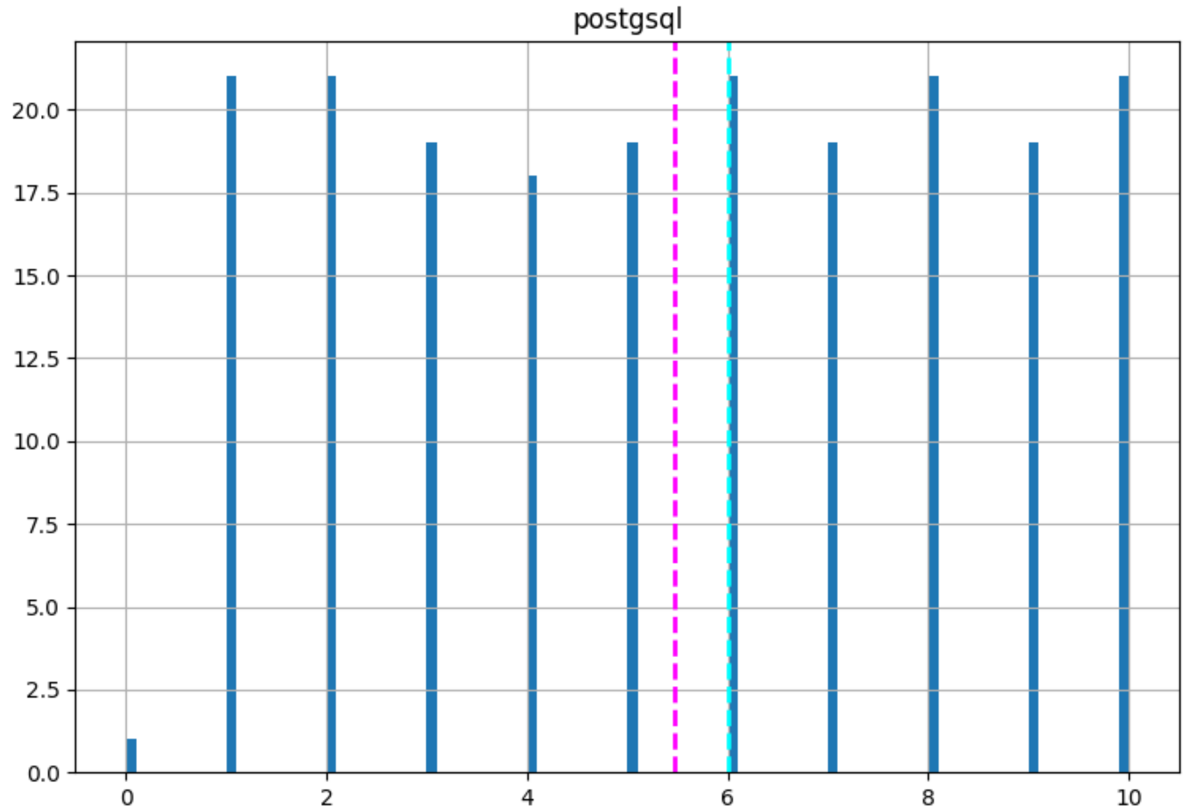
Trazo de historigrama por identidades numericas del csv

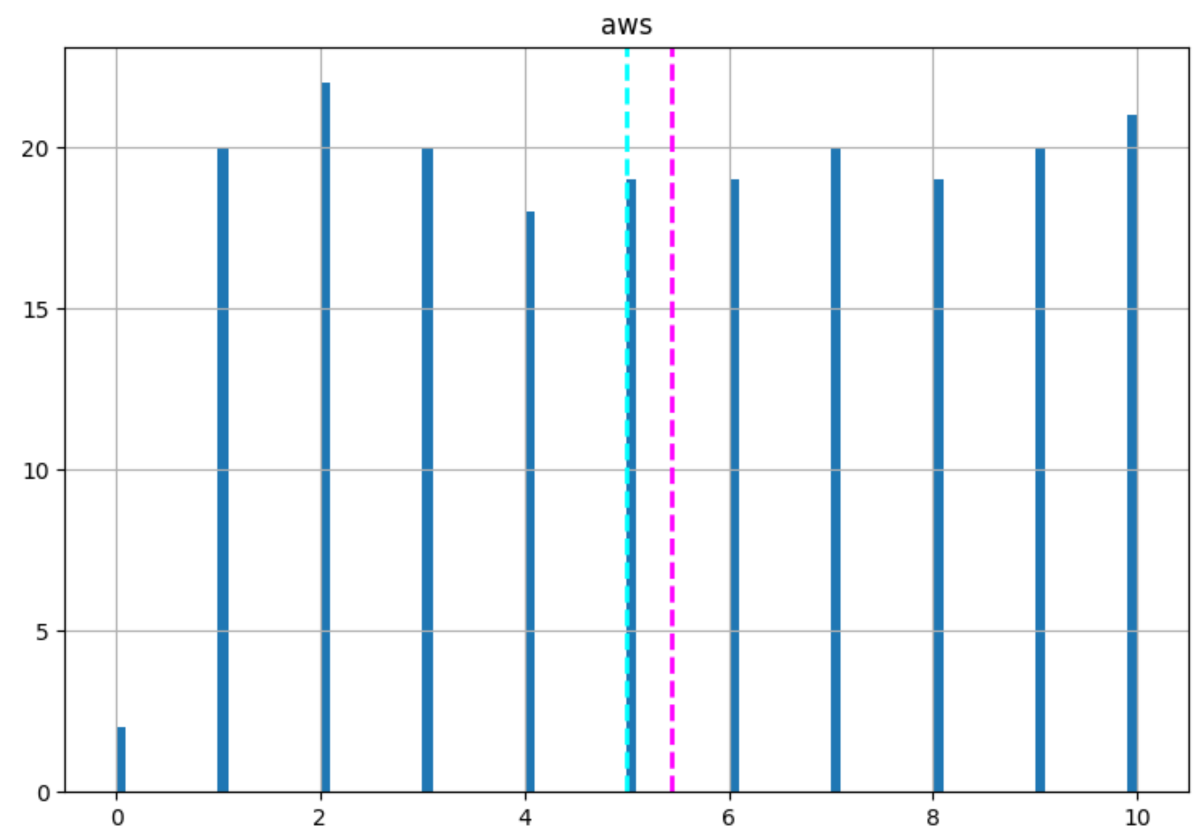
In [ ]:
```
for col in numeric_requirements:
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca()
    feature = work_prediction[col]
    feature.hist(bins=100, ax = ax)
    ax.axvline(feature.mean(), color='magenta', linestyle='dashed', linewidth=2)
    ax.axvline(feature.median(), color='cyan', linestyle='dashed', linewidth=2)
    ax.set_title(col)
plt.show()
```

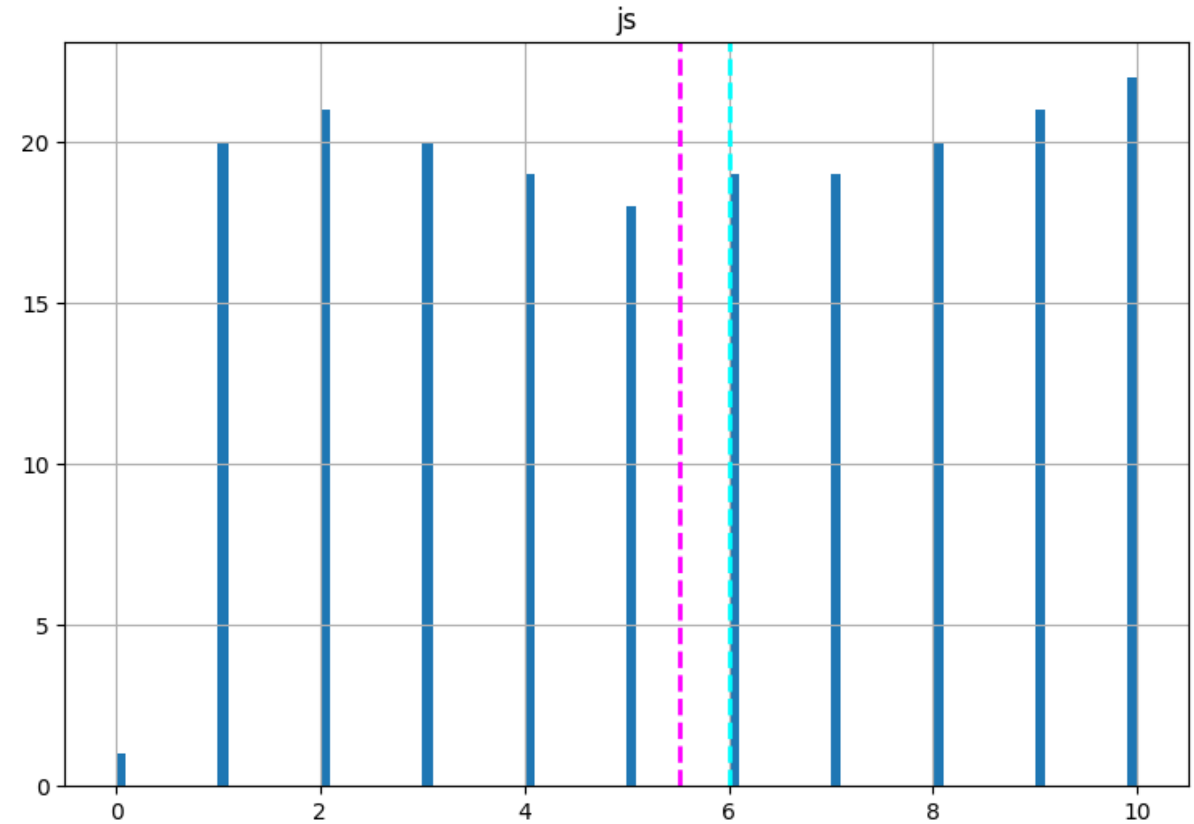


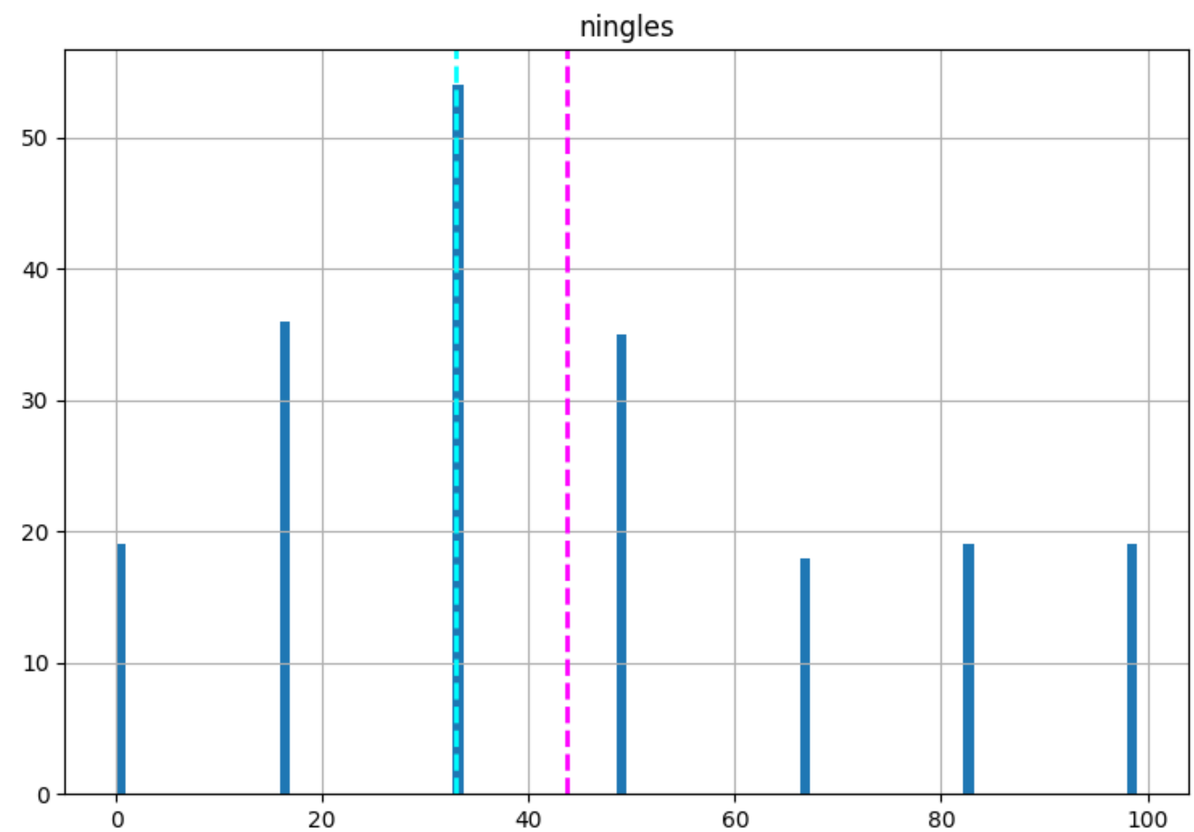

node
0
5
10
15
20
0
2
4
6
8
10

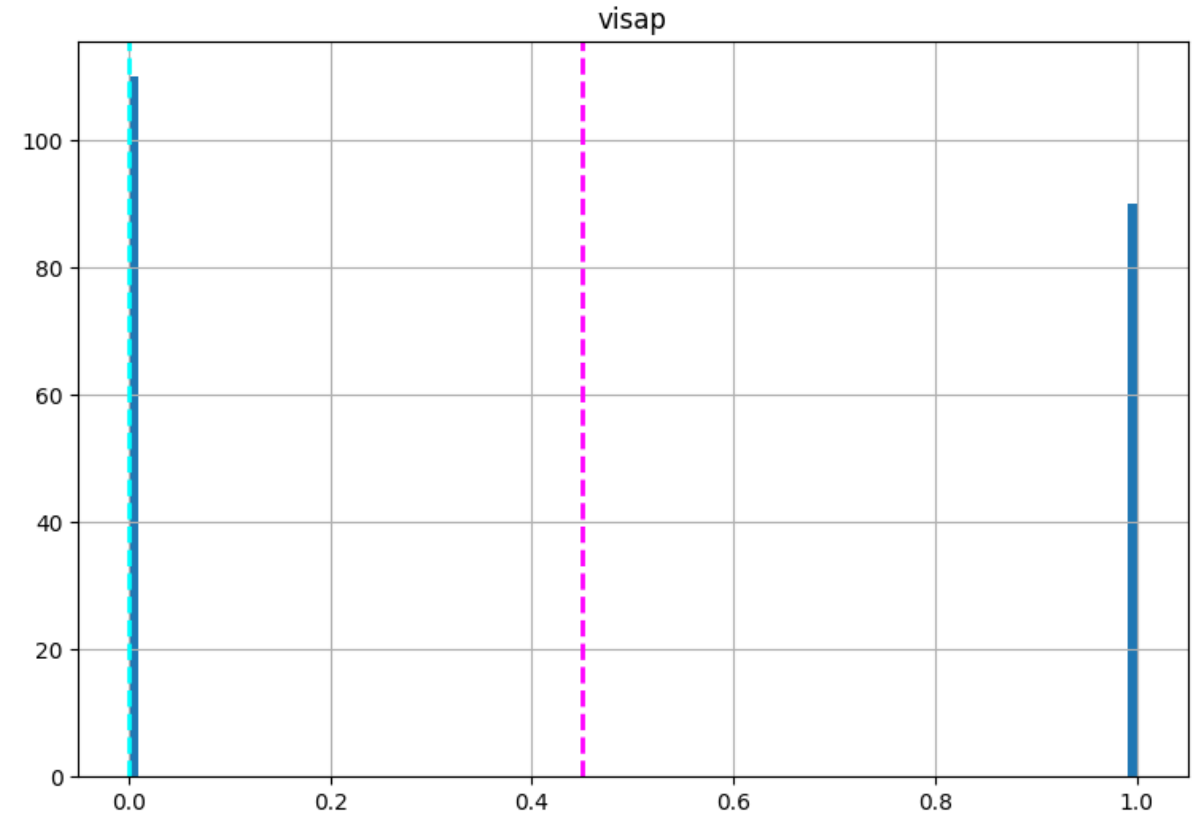

sql
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
2
4
6
8
10

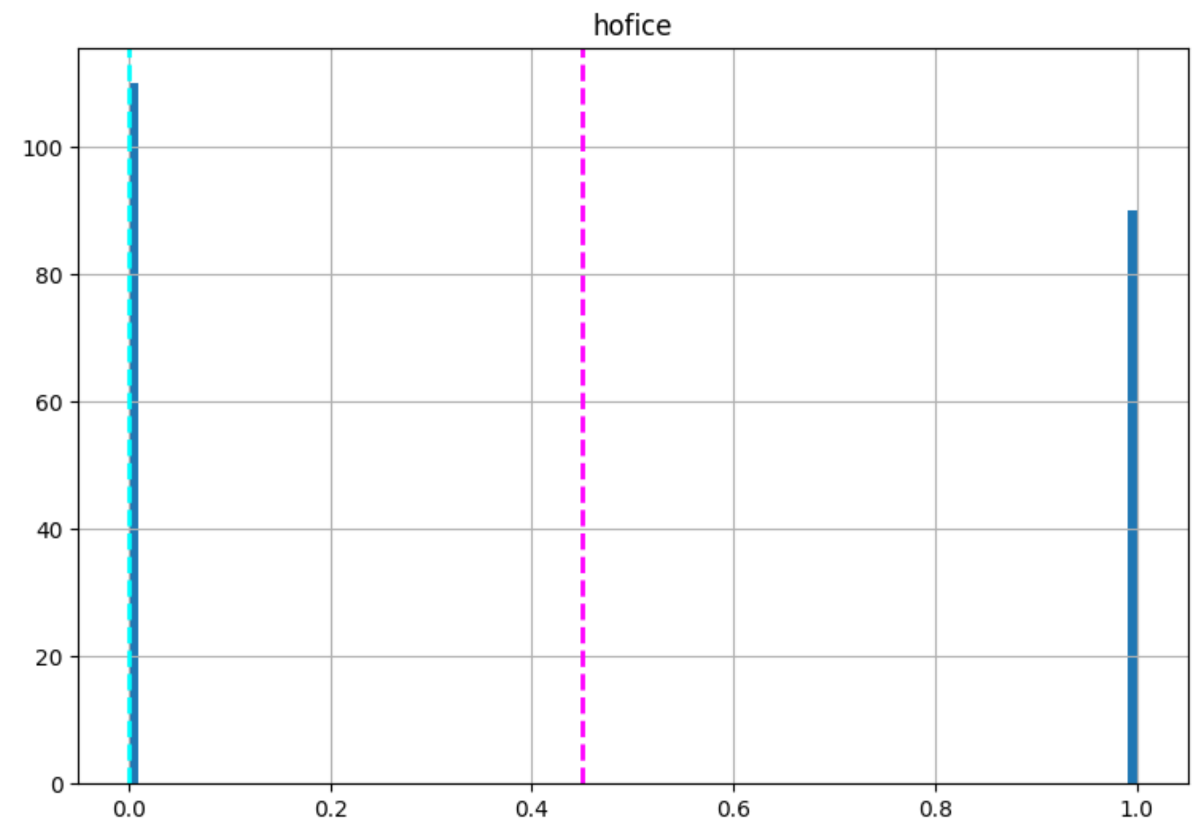

postgsql
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
2
4
6
8
10


aws
0
5
10
15
20
0
2
4
6
8
10


js
0
5
10
15
20
0
2
4
6
8
10


ningles
0
10
20
30
40
50
0
20
40
60
80
100


visap
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0


hofice
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
1.0

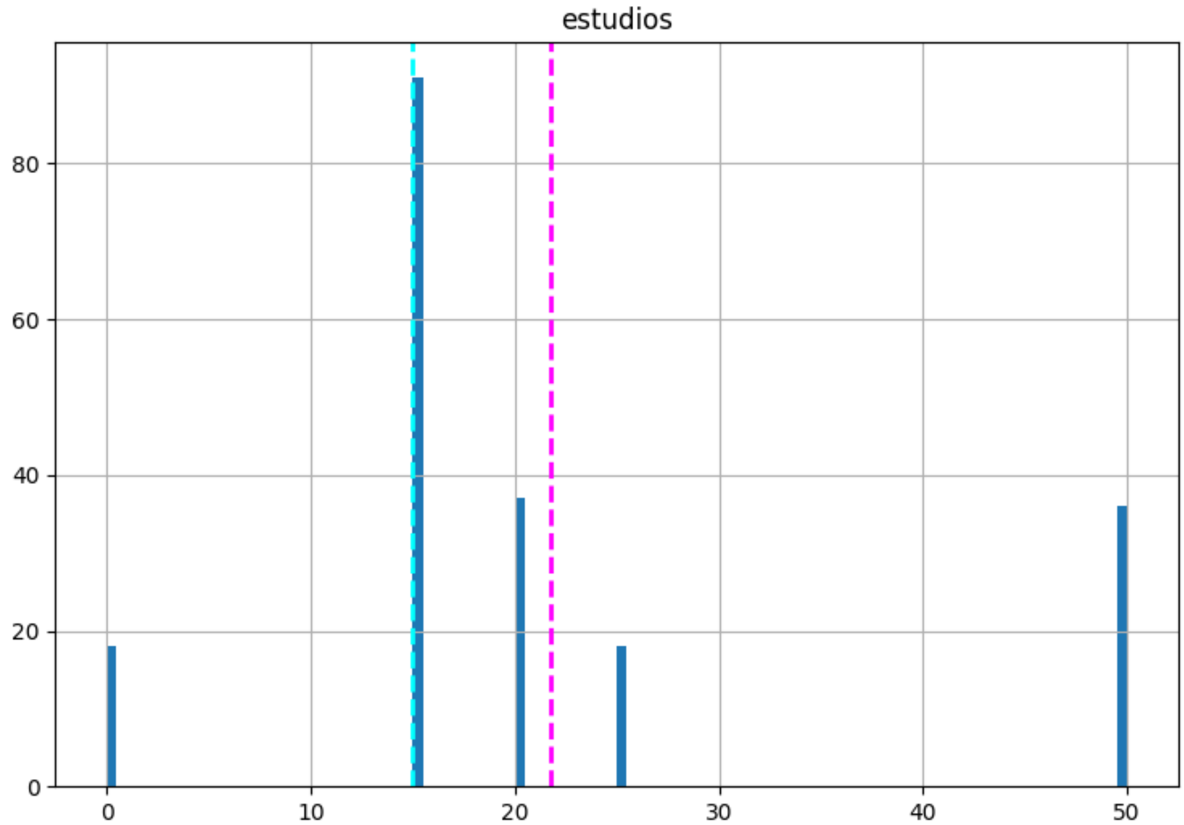
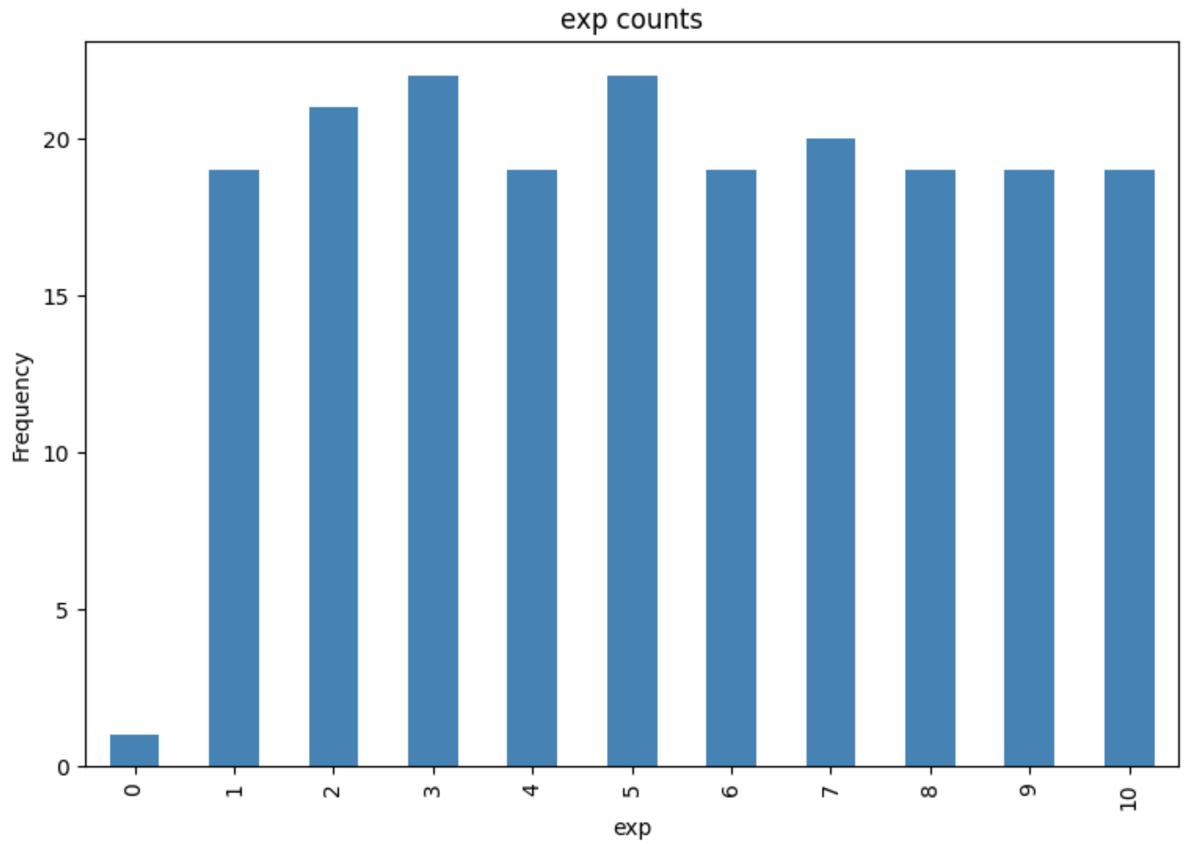
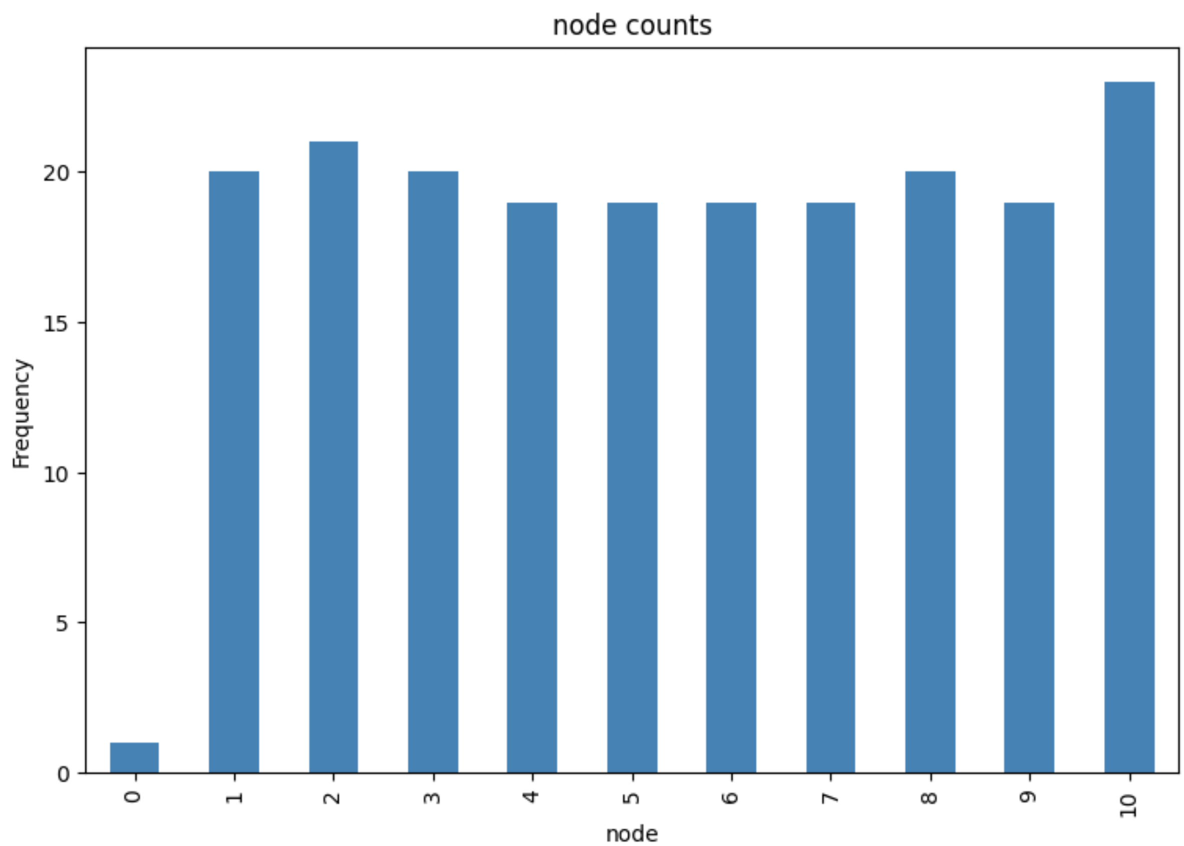
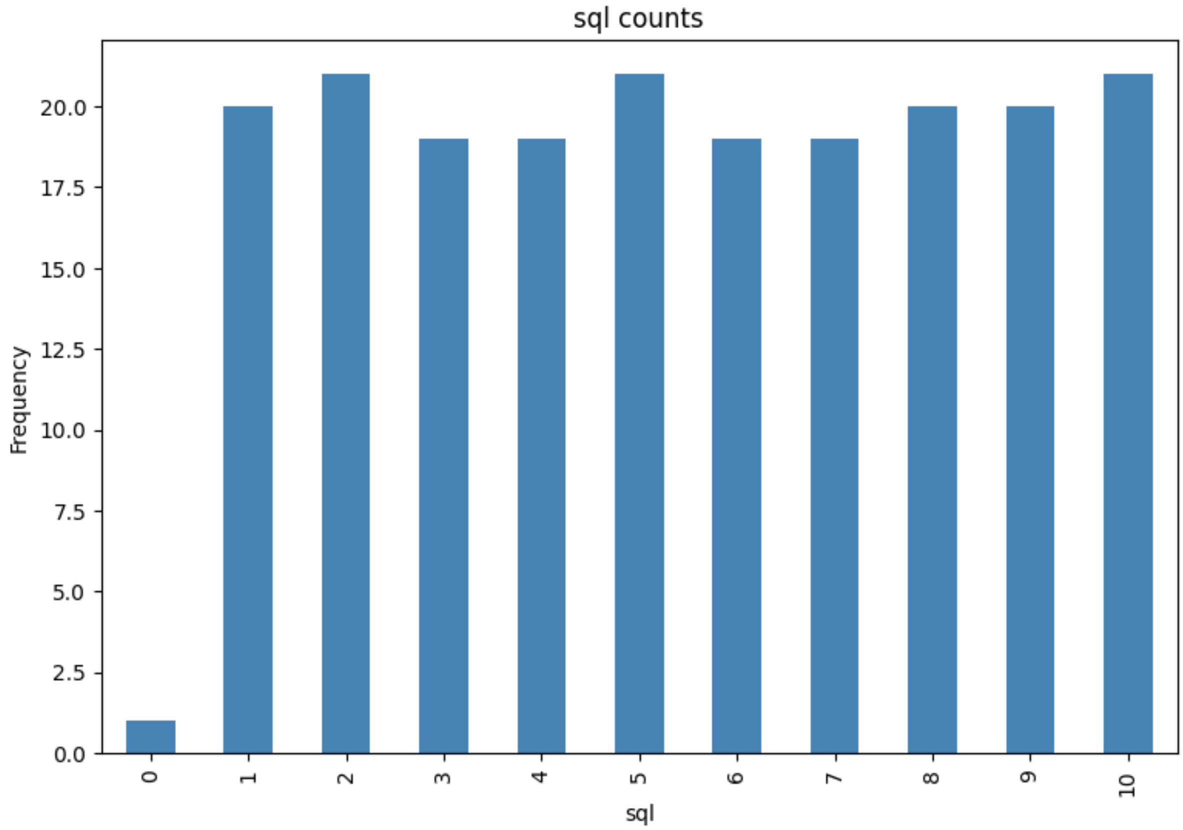


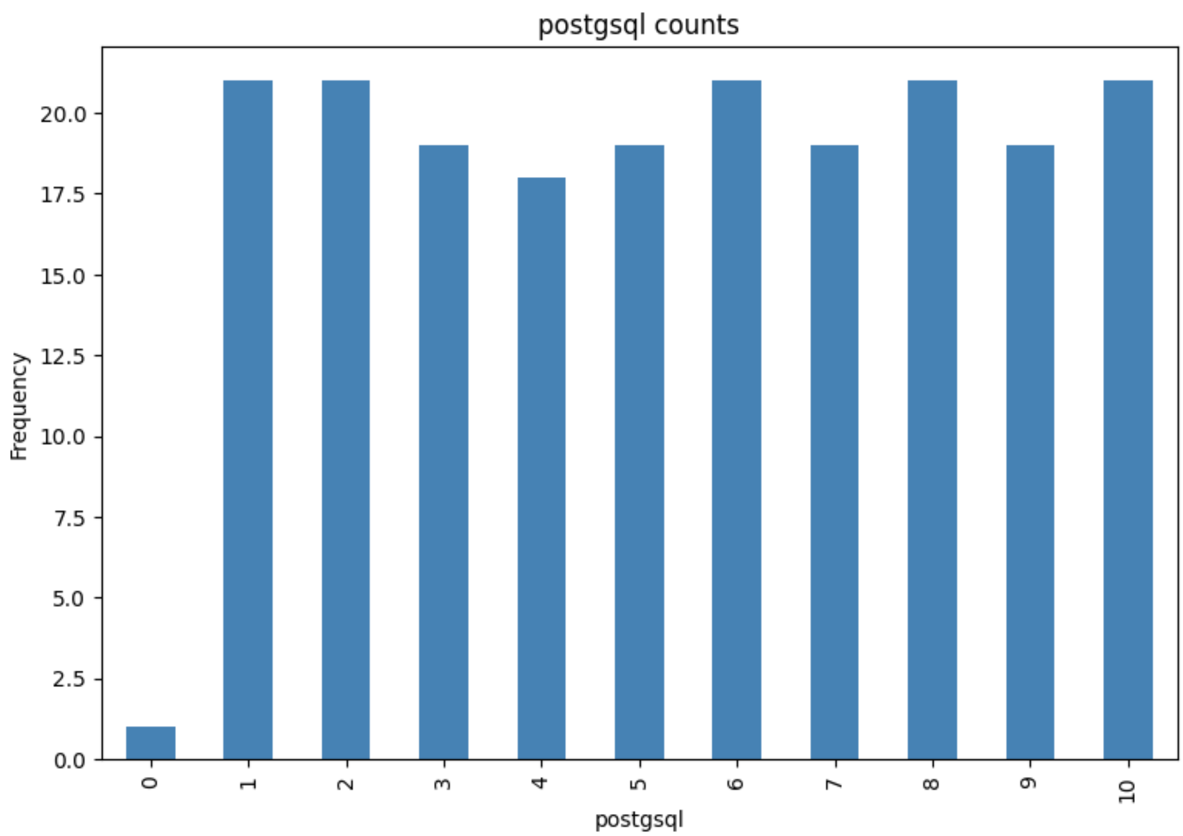
Grafico de barras para los datos categoricos

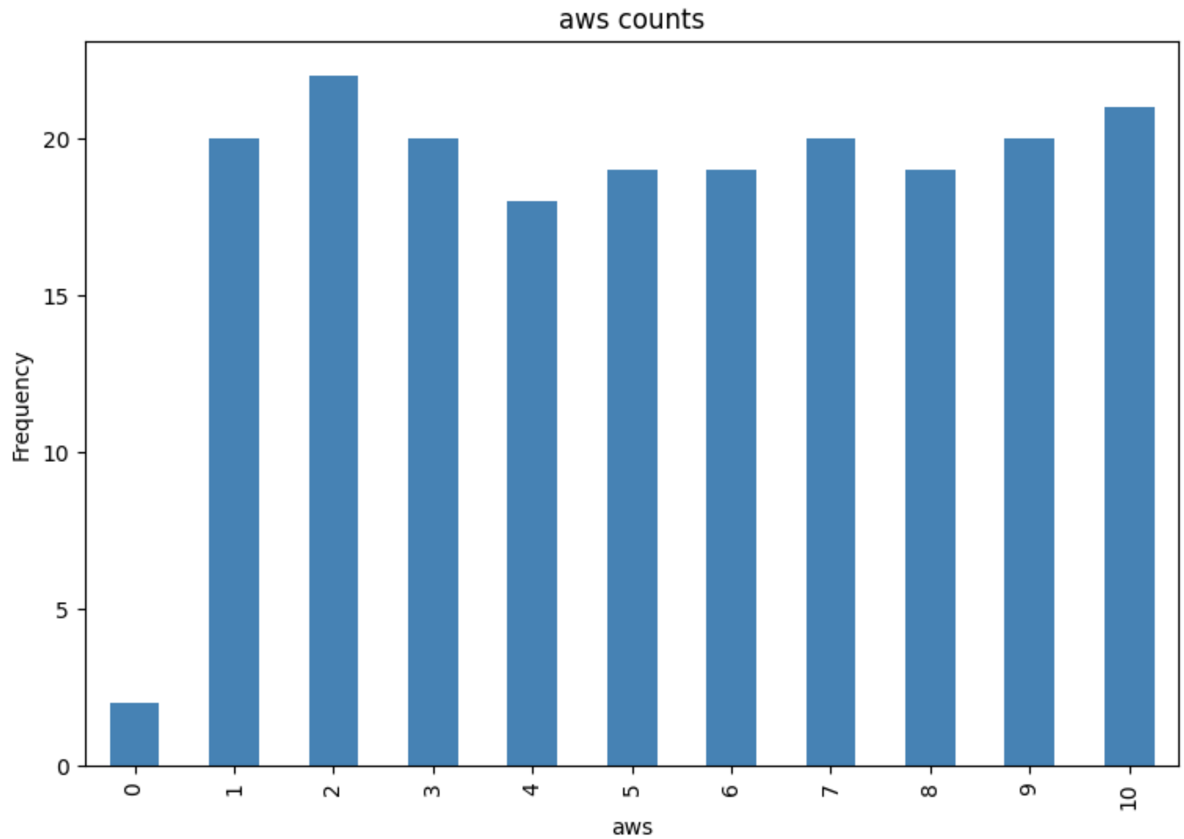
In [ ]:
```
numeric_requirements = ["exp","node","sql","postgsql","aws","js","ningles","visap",
for col in numeric_requirements:
    counts = work_prediction[col].value_counts().sort_index()
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca()
    counts.plot.bar(ax = ax, color='steelblue')
    ax.set_title(col + ' counts')
    ax.set_xlabel(col)
    ax.set_ylabel("Frequency")
plt.show()
```

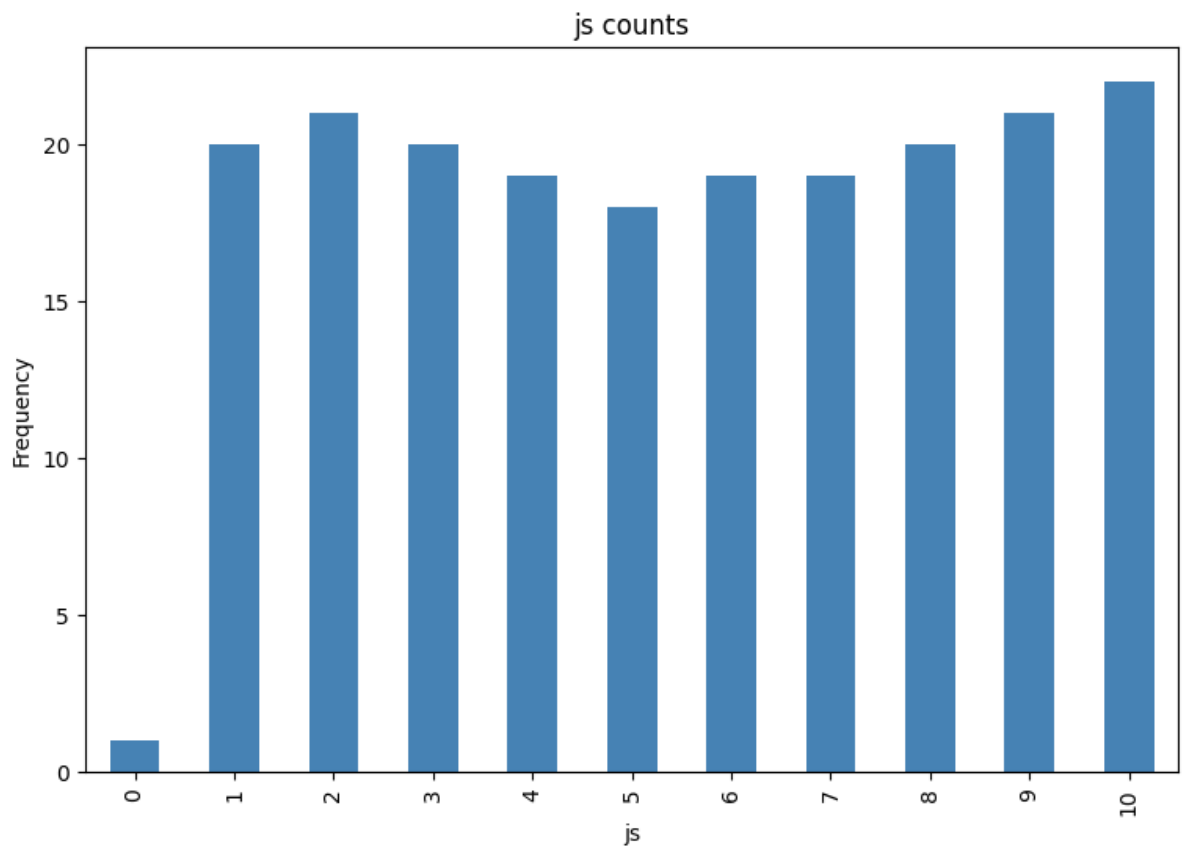

exp counts
Frequency
0
5
10
15
20
0
1
2
3
4
5
6
7
8
9
10
exp

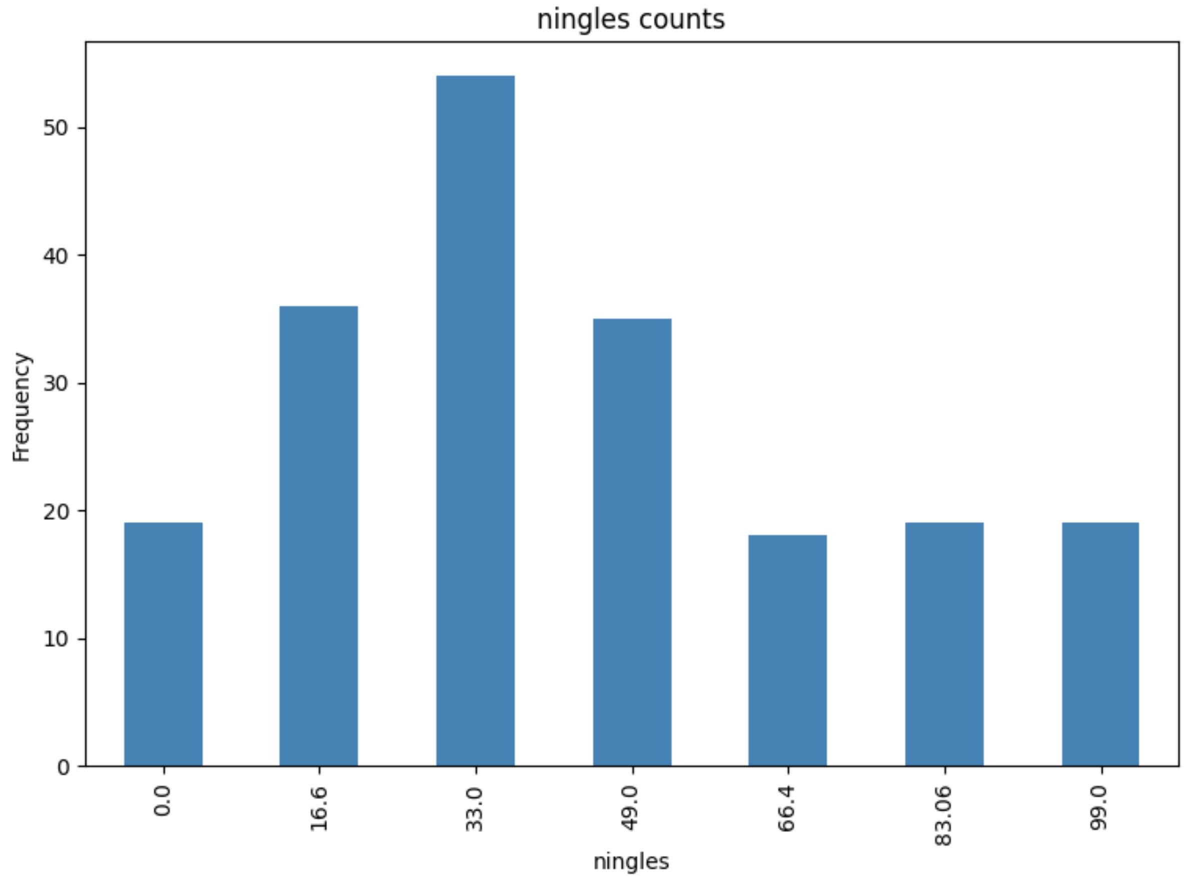

node counts
Frequency
0
5
10
15
20
0
1
2
3
4
5
6
7
8
9
10
node

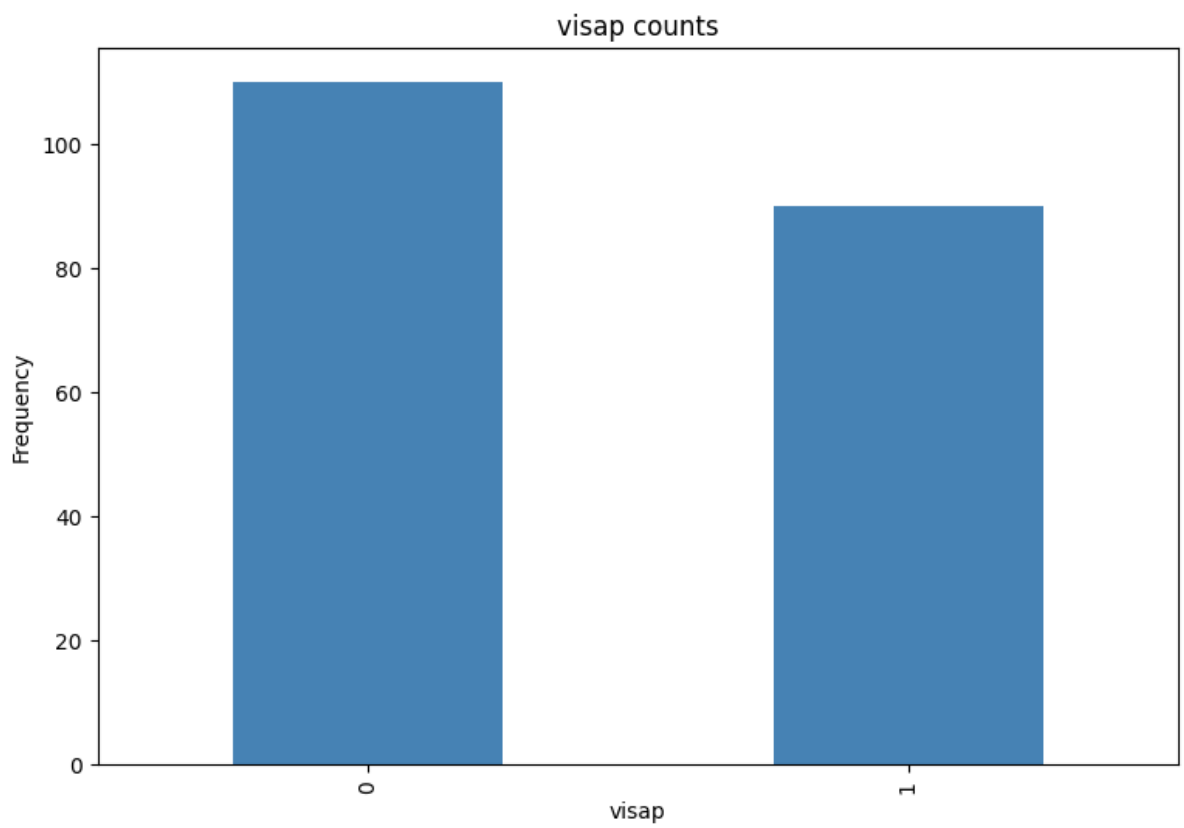

sql counts
Frequency
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
1
2
3
4
5
6
7
8
9
10
sql

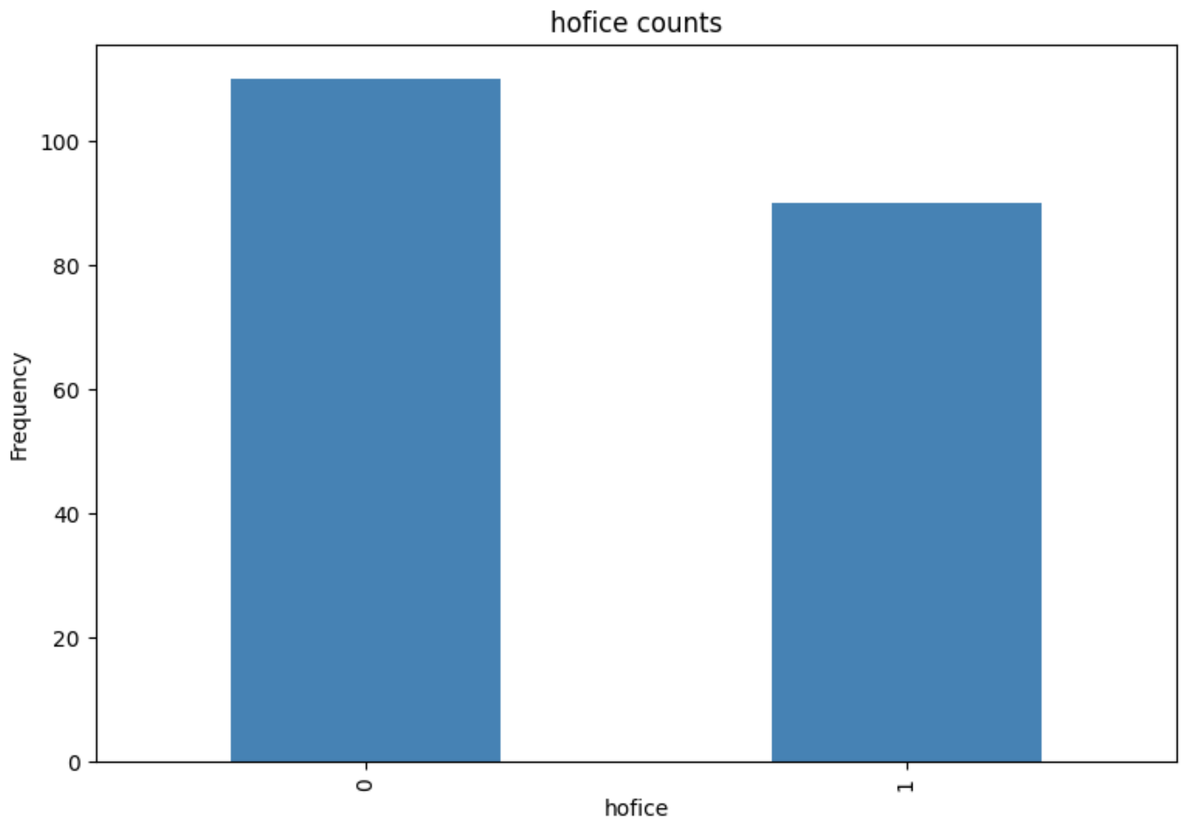

postgsql counts
Frequency
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
1
2
3
4
5
6
7
8
9
10
postgsql

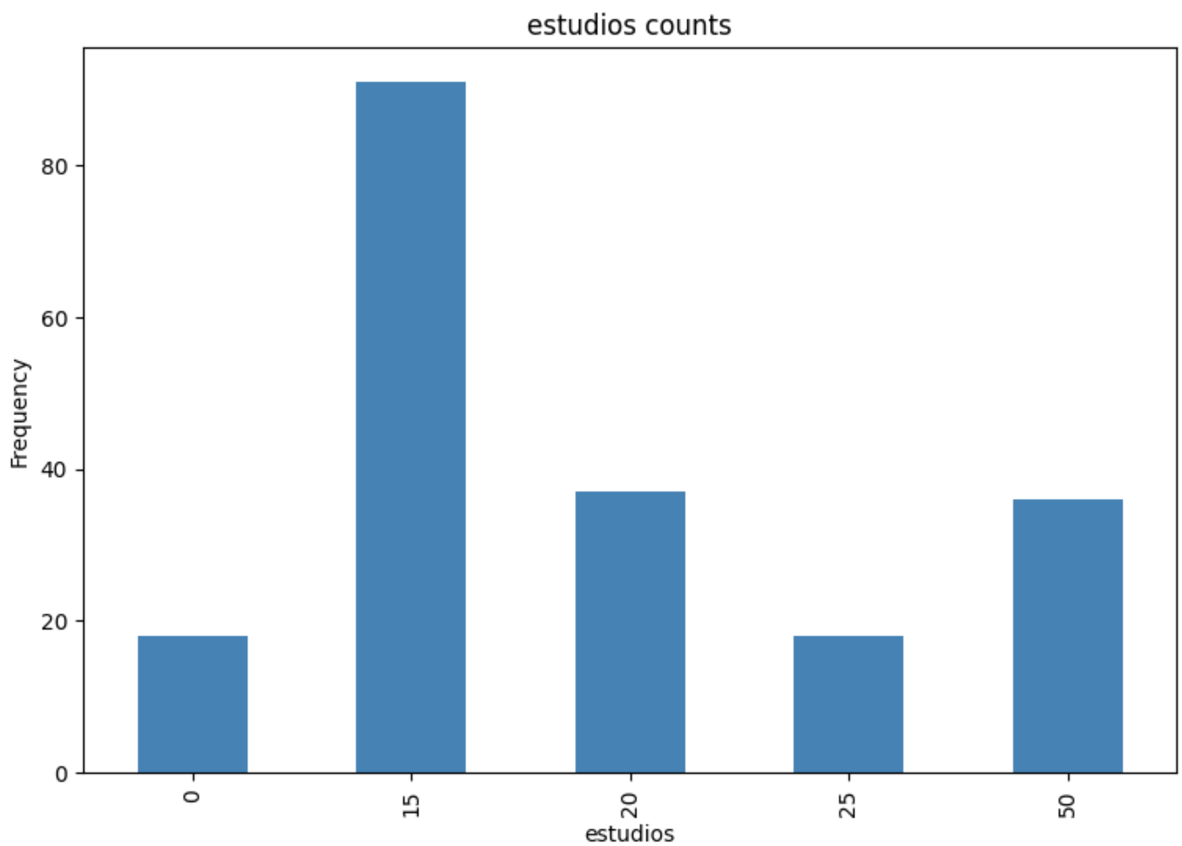

aws counts
Frequency
0
5
10
15
20
0
1
2
3
4
5
6
7
8
9
10
aws


js counts
Frequency
0
5
10
15
20
0
1
2
3
4
5
6
7
8
9
10
js


ningles counts
Frequency
0
10
20
30
40
50
0.0
16.6
33.0
49.0
66.4
83.06
99.0
ningles

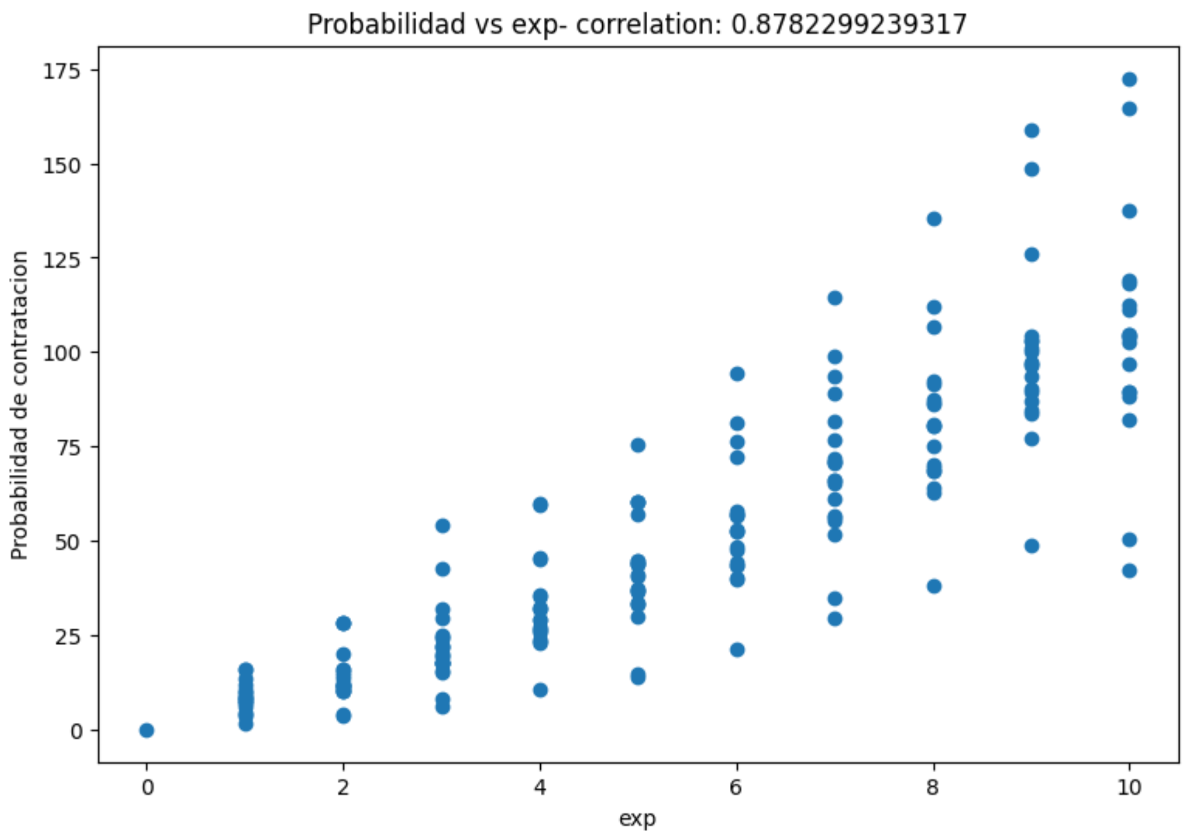

visap counts
Frequency
0
20
40
60
80
100
0
1
visap

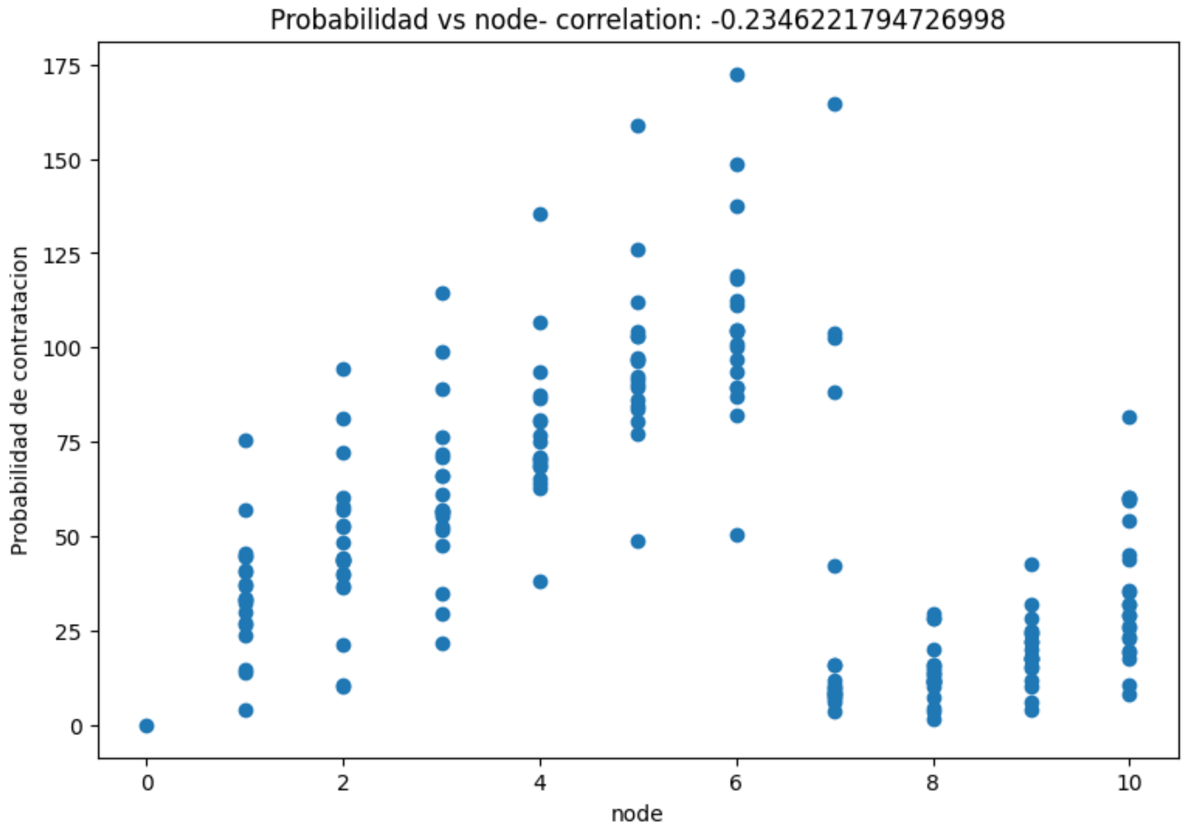


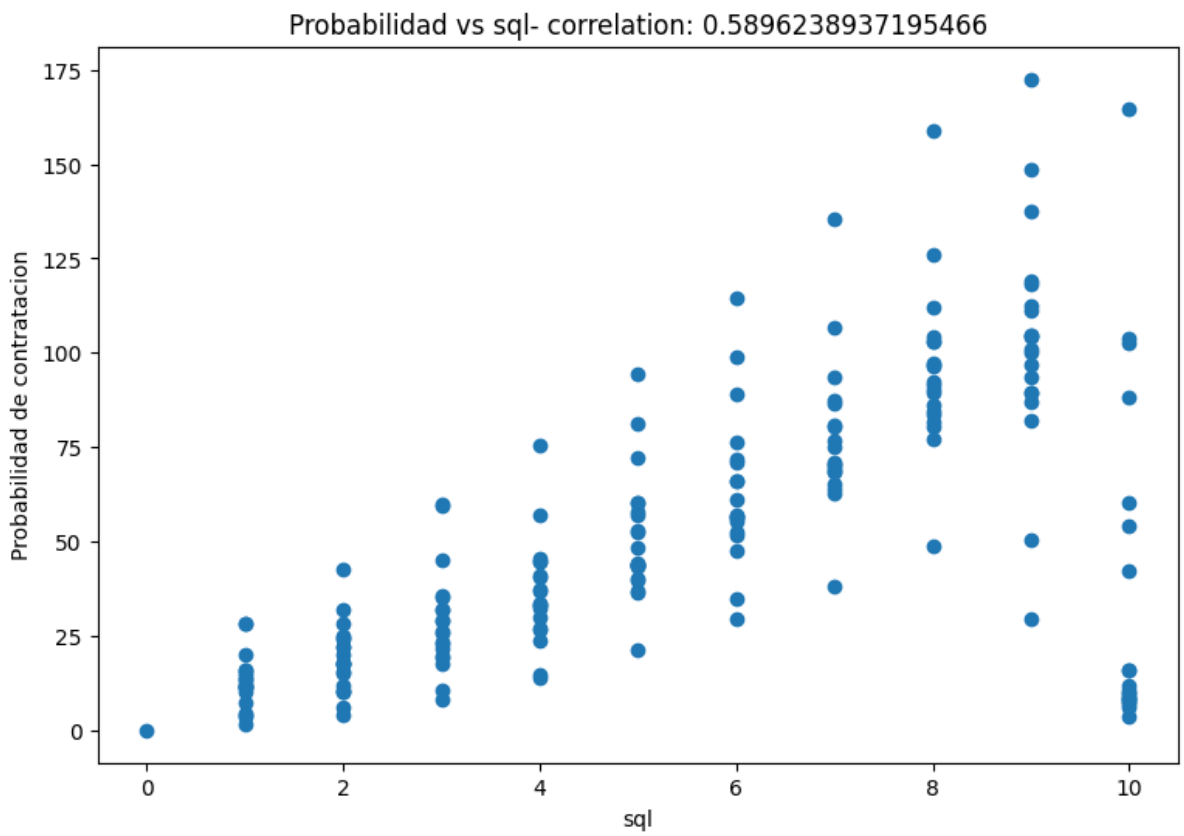


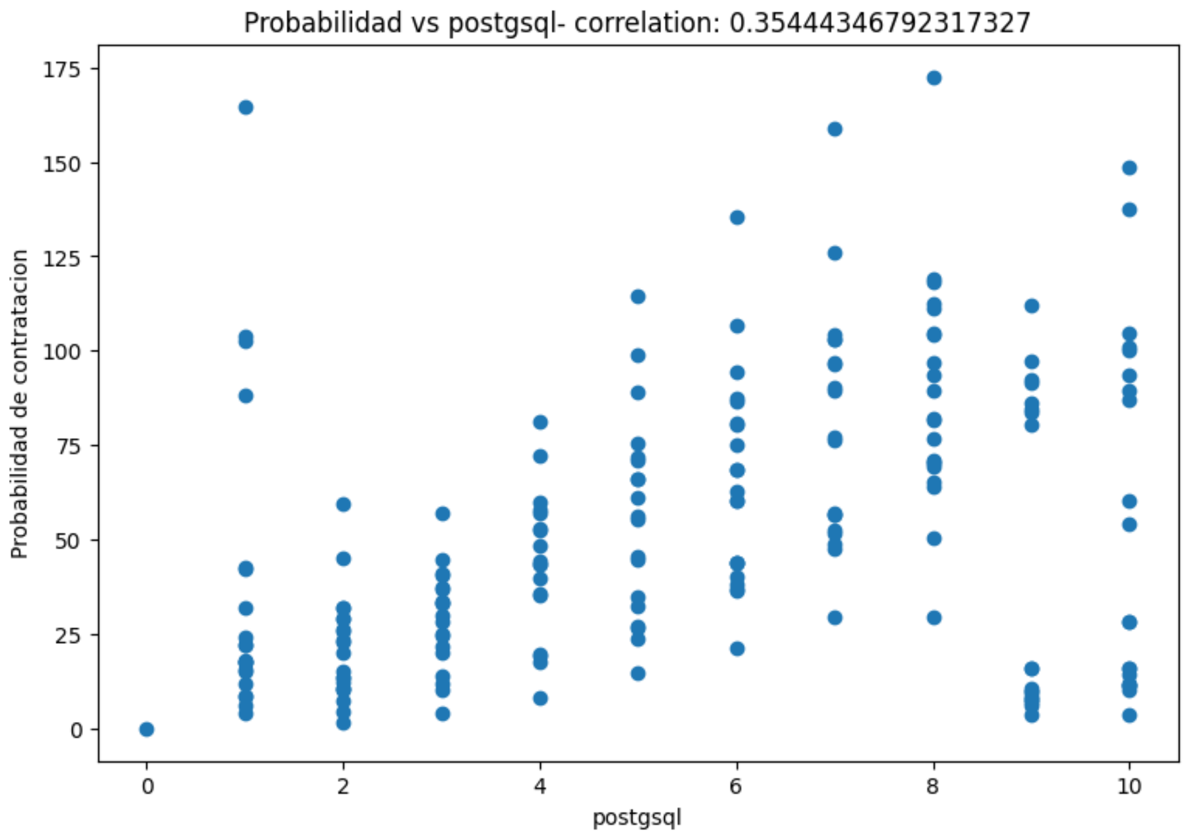
distribución de puntos

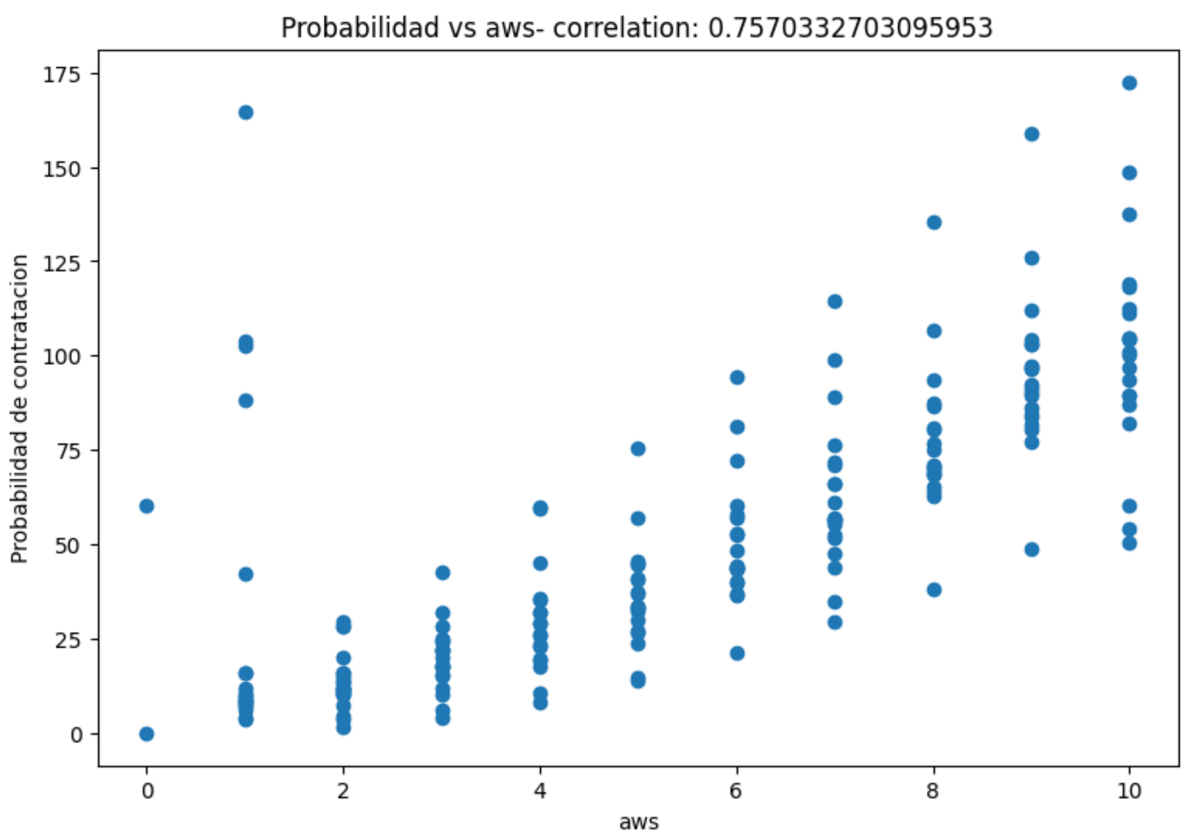
In [ ]:
```
for col in numeric_requirements:
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca()
    feature = work_prediction[col]
    label = work_prediction['probabilidad']
    correlation = feature.corr(label)
    plt.scatter(x=feature, y=label)
    plt.xlabel(col)
    plt.ylabel('Probabilidad de contratacion')
    ax.set_title('Probabilidad vs ' + col + '- correlation: ' + str(correlation))
plt.show()
```

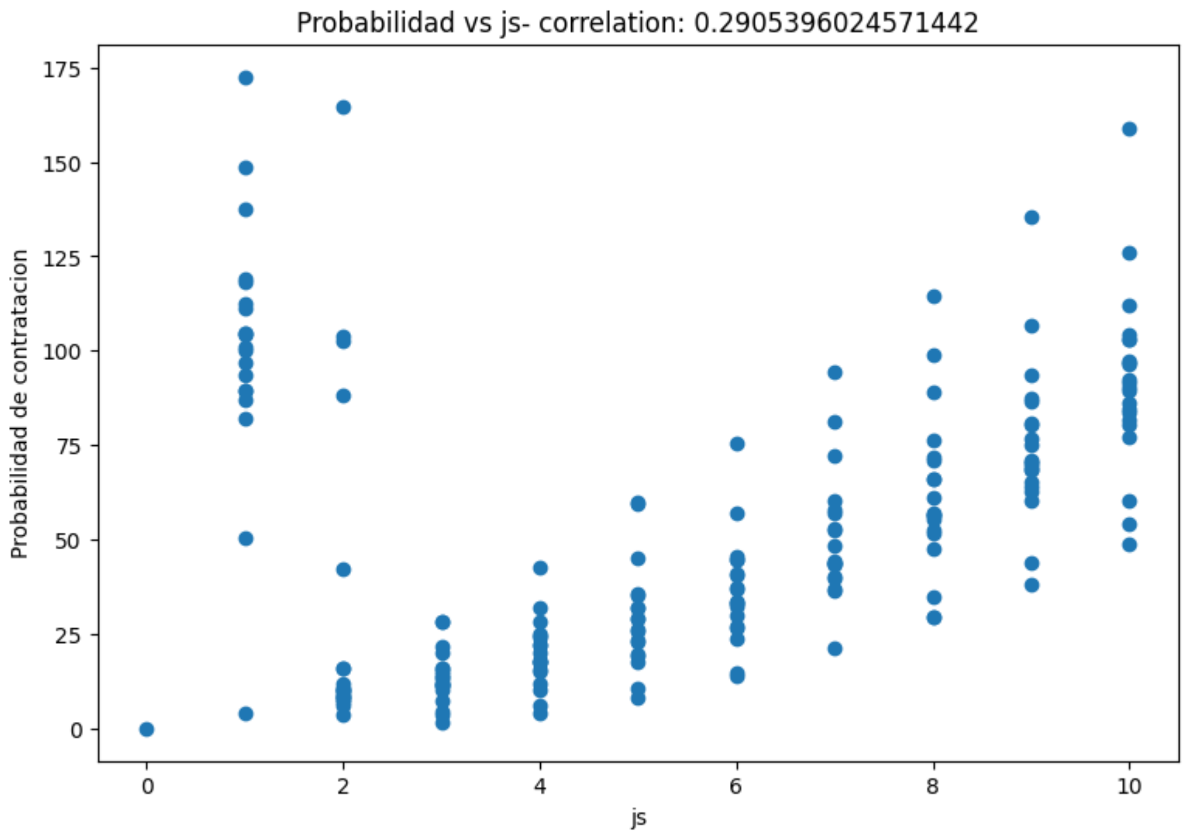


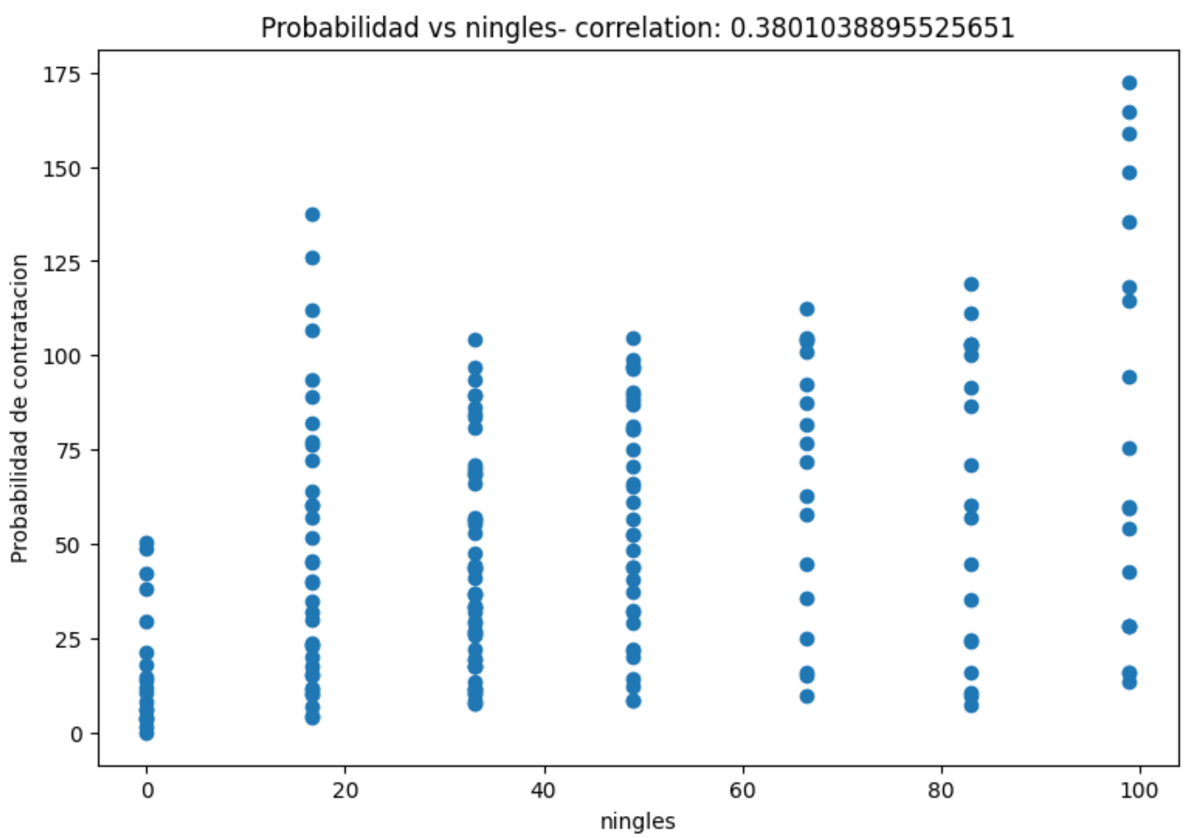

Probabilidad vs node- correlation: -0.2346221794726998
Probabilidad de contratacion
node

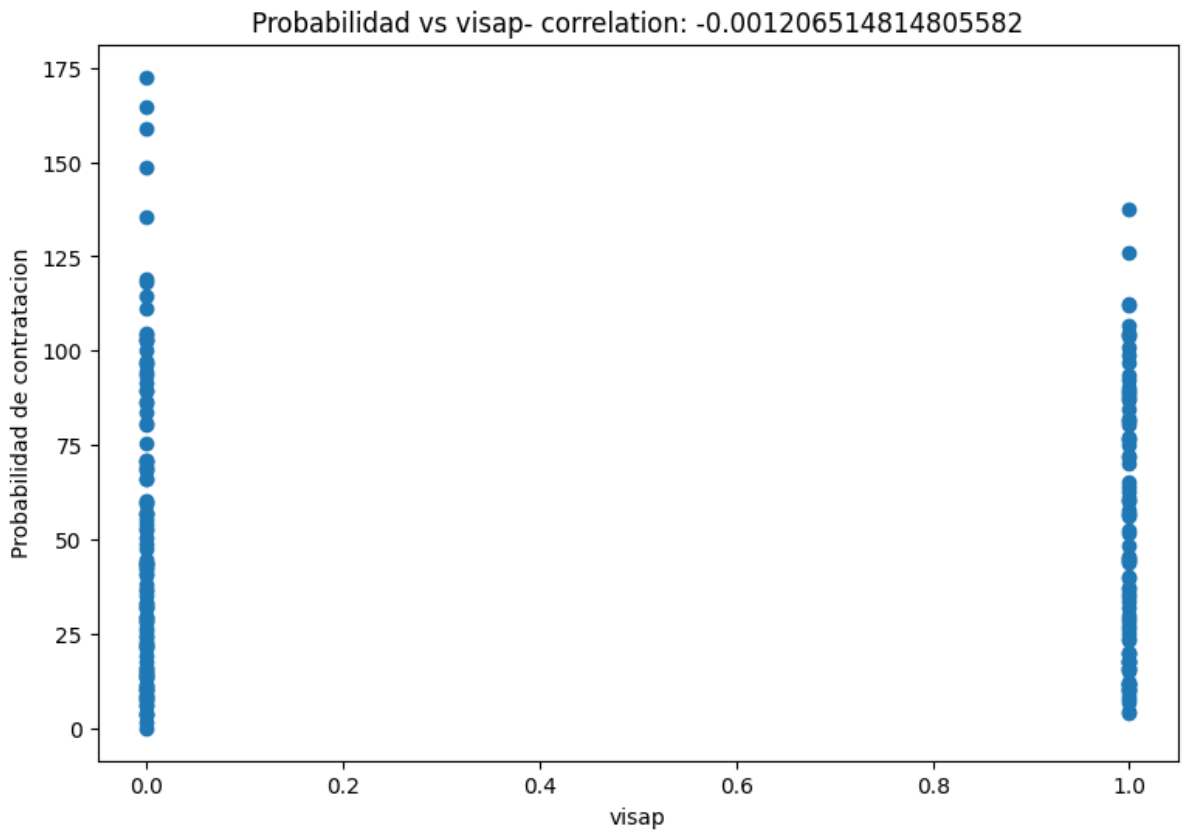

Probabilidad vs sql- correlation: 0.5896238937195466
Probabilidad de contratacion
sql

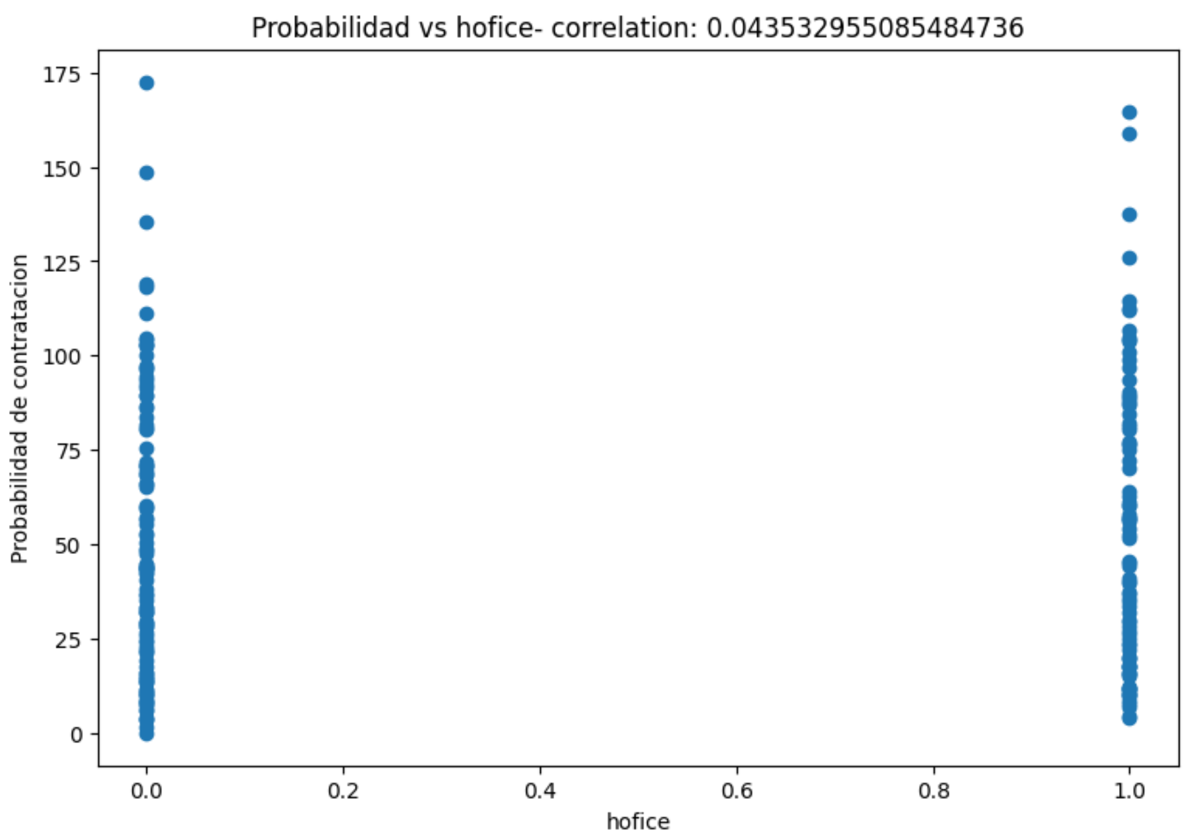

Probabilidad vs postgsql- correlation: 0.35444346792317327
Probabilidad de contratacion
postgsql
0
25
50
75
100
125
150
175
0
2
4
6
8
10

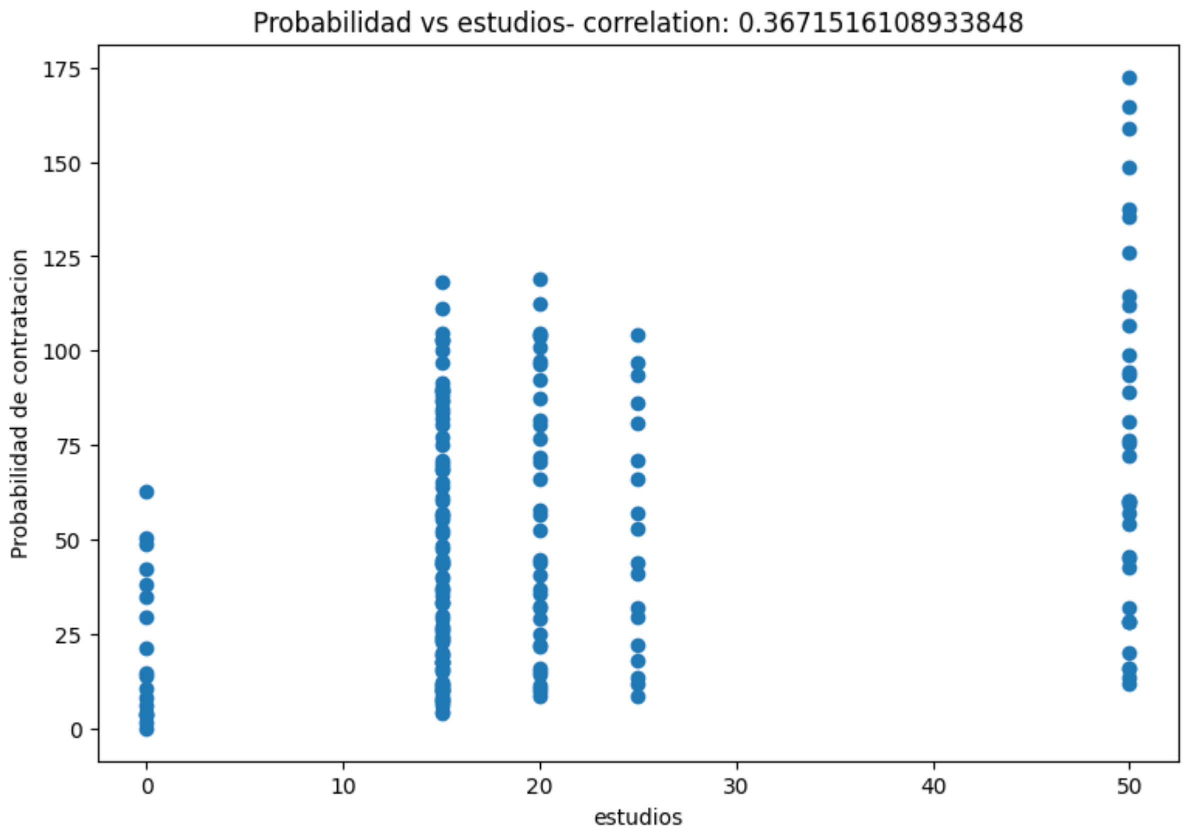

Probabilidad vs aws- correlation: 0.7570332703095953
Probabilidad de contratacion
aws
0
25
50
75
100
125
150
175
0
2
4
6
8
10


Probabilidad vs js- correlation: 0.2905396024571442
Probabilidad de contratacion
js


Probabilidad vs ningles- correlation: 0.3801038895525651
Probabilidad de contratacion
ningles


Probabilidad vs visap- correlation: -0.001206514814805582
Probabilidad de contratacion
visap
175
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0

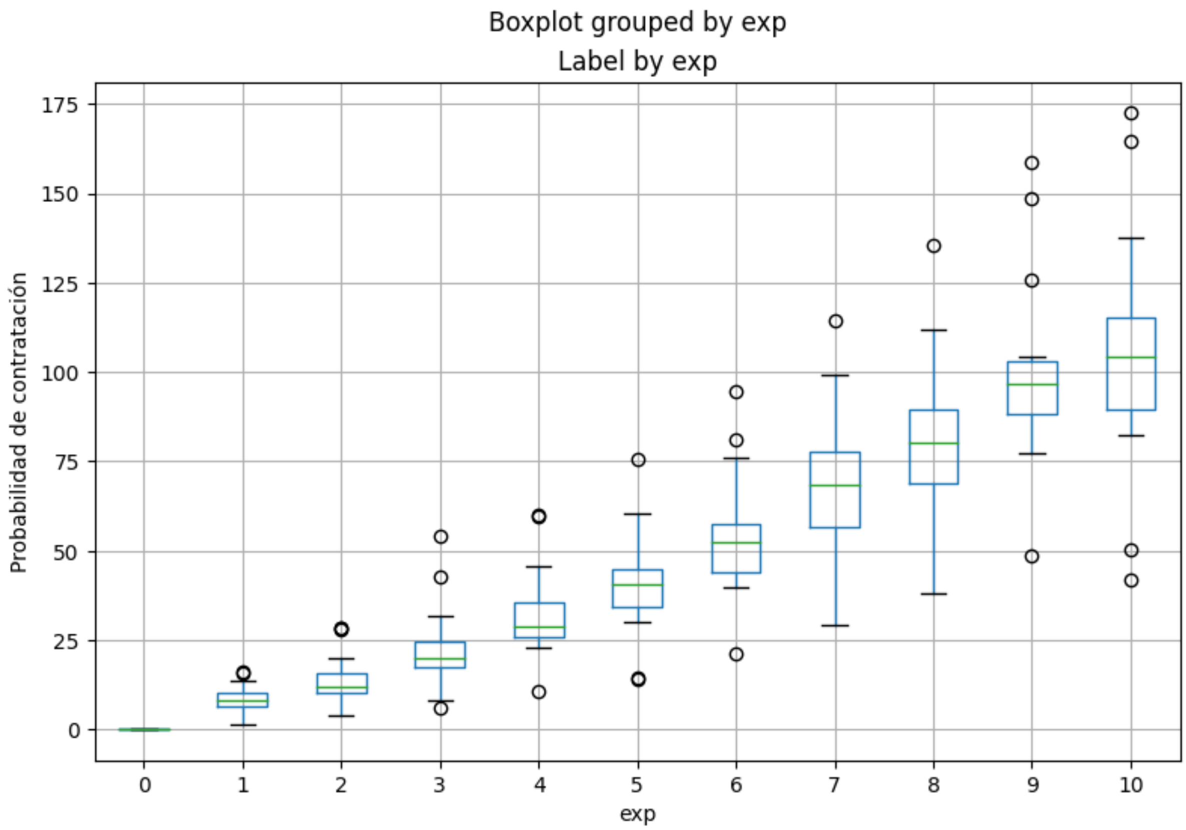

Probabilidad vs hofice- correlation: 0.043532955085484736
Probabilidad de contratacion
hofice
175
150
125
100
75
50
25
0
0.0
0.2
0.4
0.6
0.8
1.0

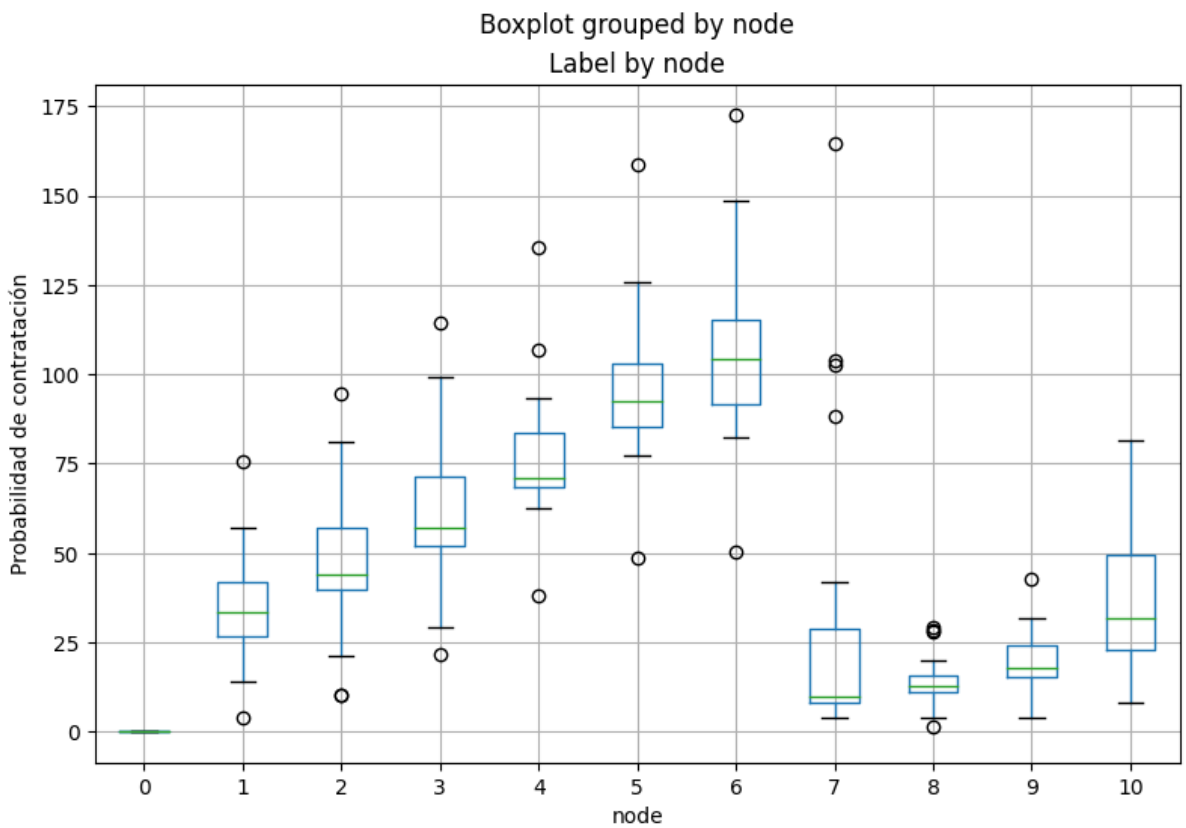


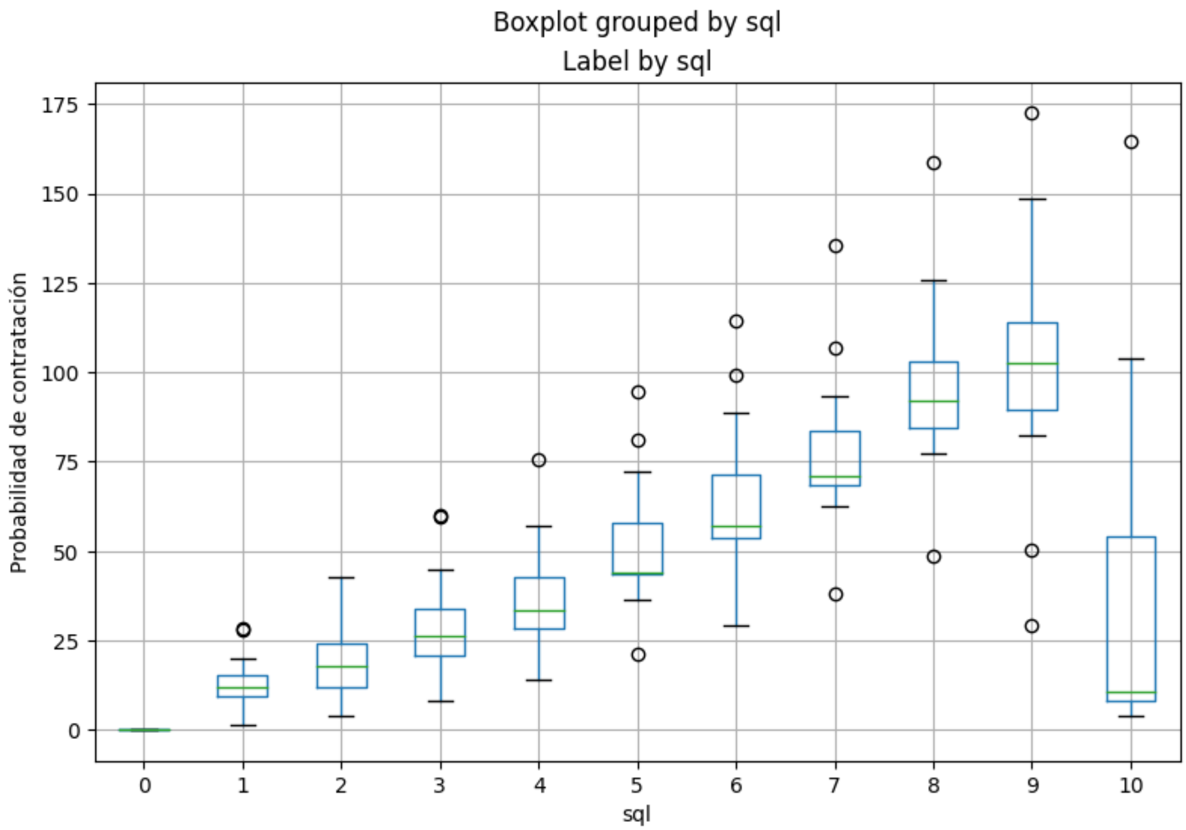
Trazo de diagrama de caja por cada uno de las categorias

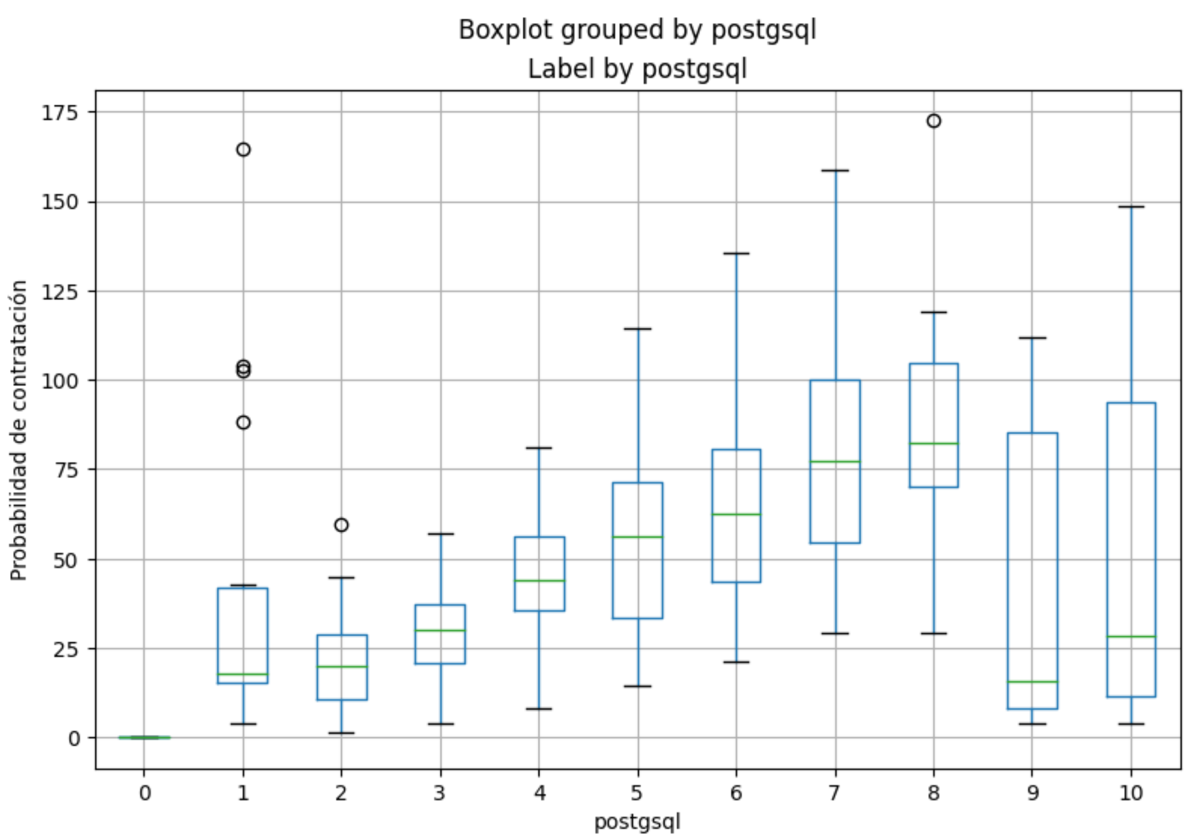
In [ ]:

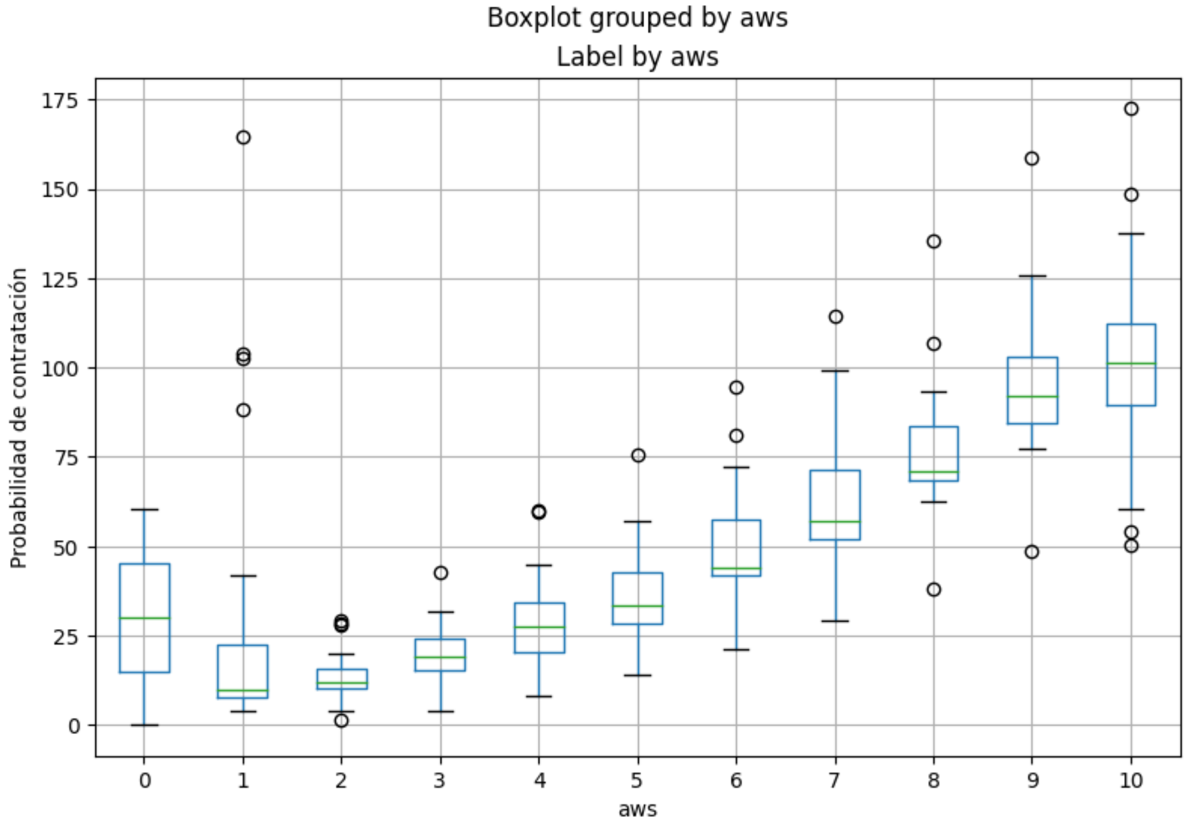
```
for col in numeric_requirements:
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca()
    work_prediction.boxplot(column = 'probabilidad', by = col, ax = ax)
    ax.set_title('Label by ' + col)
    ax.set_ylabel("Probabilidad de contratación")
plt.show()
```

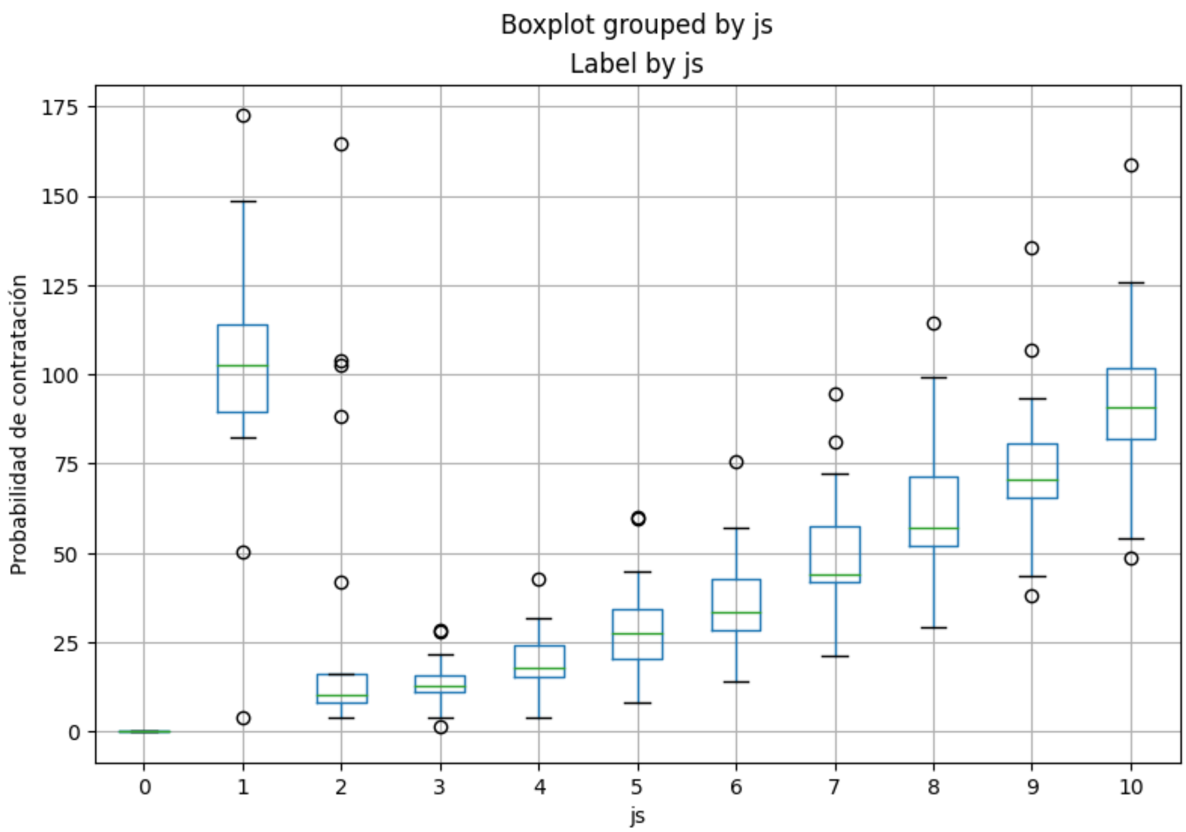

Boxplot grouped by exp
Label by exp
Probabilidad de contratación
0
25
50
75
100
125
150
175
0
1
2
3
4
5
6
7
8
9
10
exp

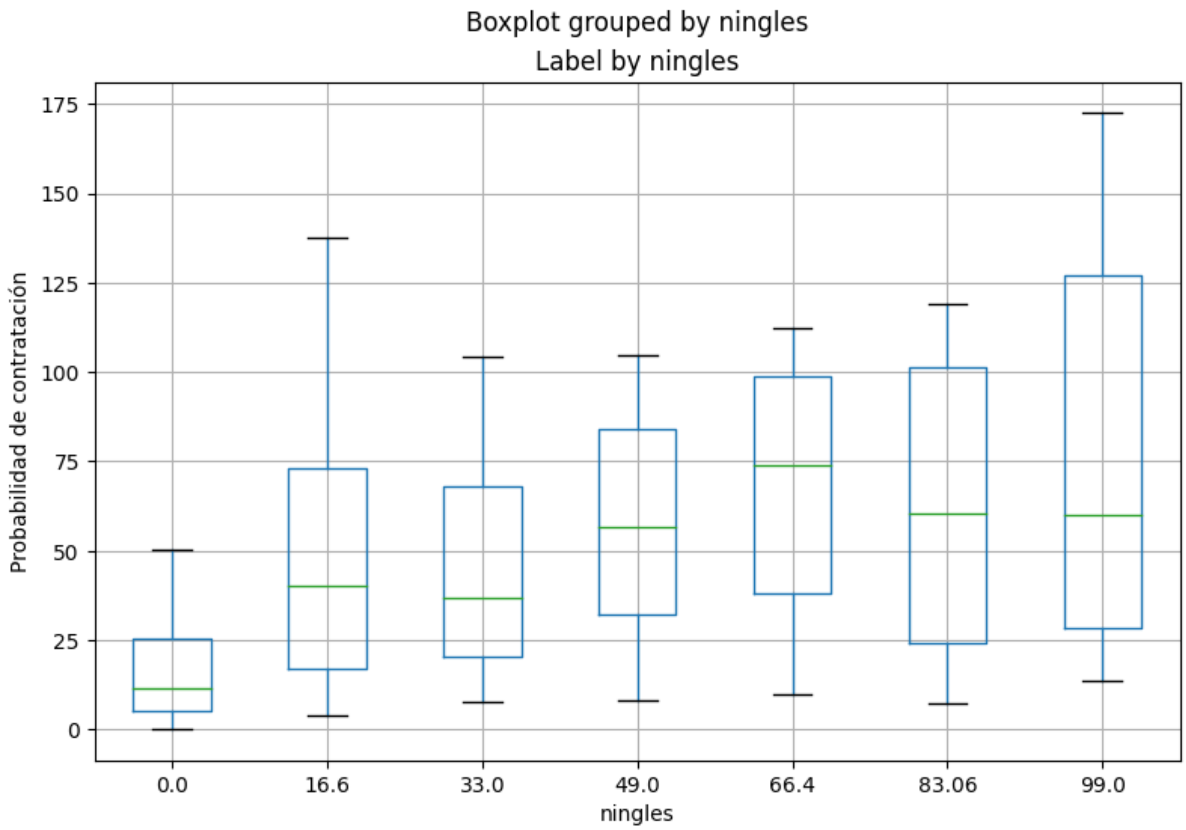

Boxplot grouped by node
Label by node
Probabilidad de contratación
0
25
50
75
100
125
150
175
0
1
2
3
4
5
6
7
8
9
10
node

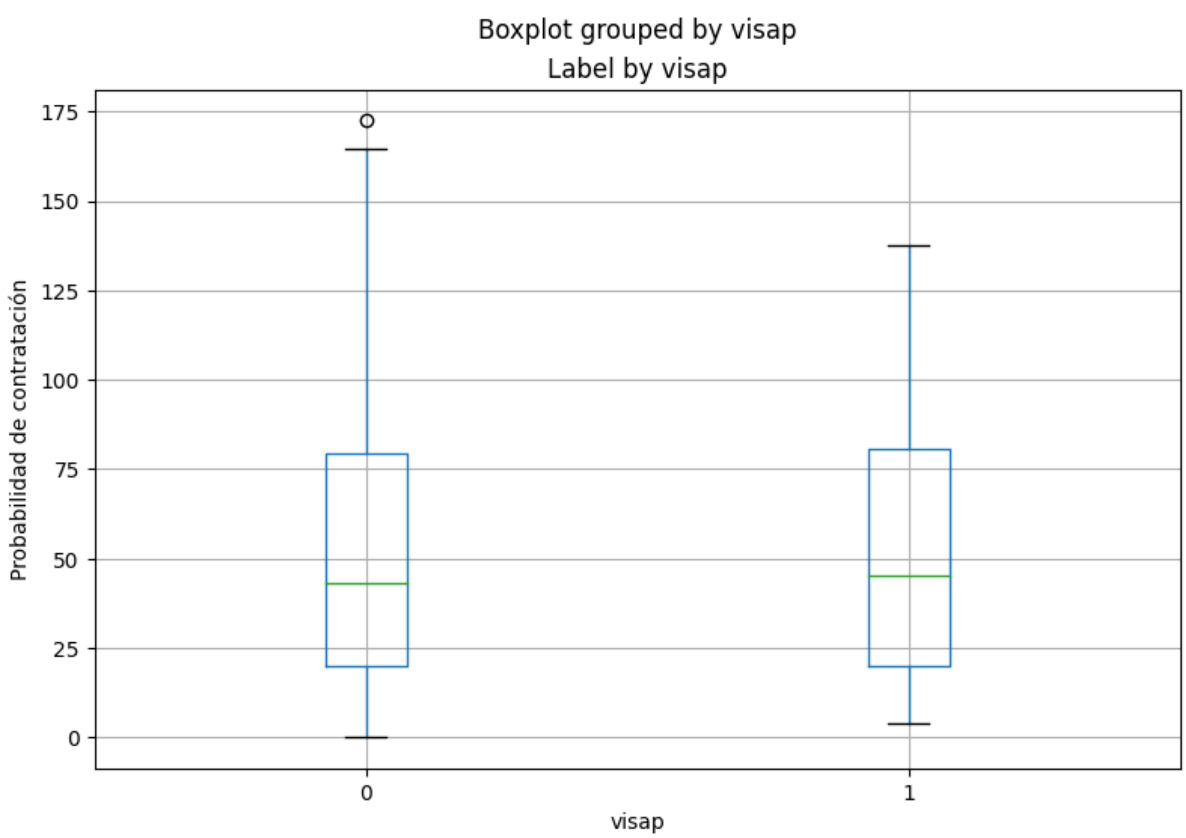

Boxplot grouped by sql
Label by sql
Probabilidad de contratación
0
25
50
75
100
125
150
175
0 1 2 3 4 5 6 7 8 9 10
sql

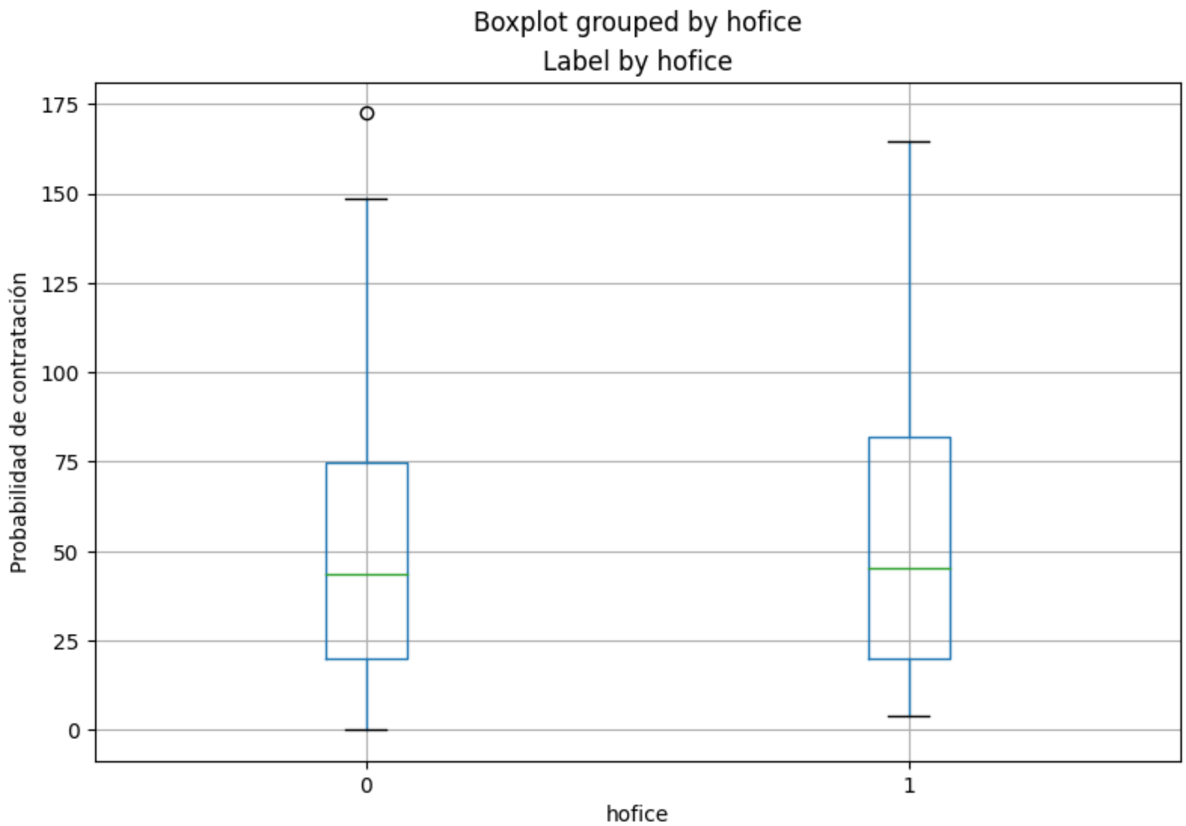

Boxplot grouped by postgsql
Label by postgsql
Probabilidad de contratación
0
25
50
75
100
125
150
175
0 1 2 3 4 5 6 7 8 9 10
postgsql

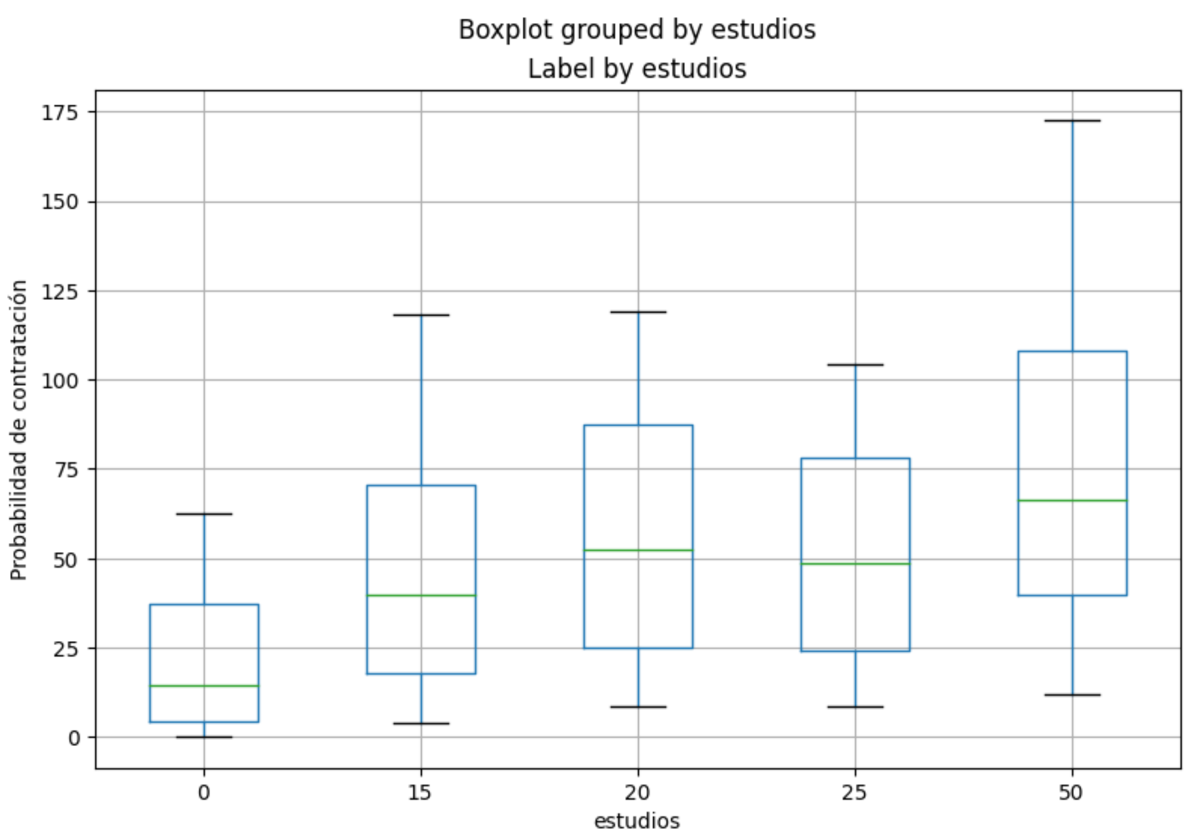

Boxplot grouped by aws
Label by aws
Probabilidad de contratación
175
150
125
100
75
50
25
0
0 1 2 3 4 5 6 7 8 9 10
aws


Boxplot grouped by js
Label by js
Probabilidad de contratación
175
150
125
100
75
50
25
0
0 1 2 3 4 5 6 7 8 9 10
js


Boxplot grouped by ningles
Label by ningles
Probabilidad de contratación
175
150
125
100
75
50
25
0
0.0
16.6
33.0
49.0
66.4
83.06
99.0
ningles


Boxplot grouped by visap
Label by visap
Probabilidad de contratación
175
150
125
100
75
50
25
0
0
1
visap





Separar caracteristicas y etiquetas

In [ ]:
```
x, y = work_prediction[["exp", "node", "sql", "postgsql", "aws", "js", "ningles",
                       "visap", "hofice", "estudios"]].values, work_prediction["pr
print("Features: ", x[:10],"\nLabels: ", y[:10], sep='\n')
```

```
Features:
[[ 5.   10.    5.    6.    0.    9.   16.6   1.    1.   50.  ]
 [ 3.    8.    9.    8.    2.    8.   33.    0.    1.   25.  ]
 [ 5.   10.    5.    6.    7.    9.   49.    1.    0.   15.  ]
 [ 7.   10.    8.    8.    9.   10.   66.4   1.    0.   20.  ]
 [ 5.   10.   10.   10.   10.   10.   83.06  0.    0.   15.  ]
 [ 3.   10.   10.   10.   10.   10.   99.    0.    1.   50.  ]
 [ 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 1.    1.    1.    1.    1.    1.   16.6   1.    1.   15.  ]
 [ 2.    2.    2.    2.    2.    2.   33.    1.    1.   15.  ]
 [ 2.    2.    2.    2.    2.    2.   33.    0.    0.   15.  ]]

Labels:
[60.235  29.28   43.65   81.466  60.3135 54.165   0.      3.822  10.42
 10.22  ]
```

Divir datos por porcentaje del 30% hasta un 70%

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_st

print ('Training Set: %d rows\nTest Set: %d rows' % (X_train.shape[0], X_test.shape
```

```
Training Set: 140 rows
Test Set: 60 rows
```

Ajuste de modelo

In [ ]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression().fit(X_train, y_train)
print (model)
```

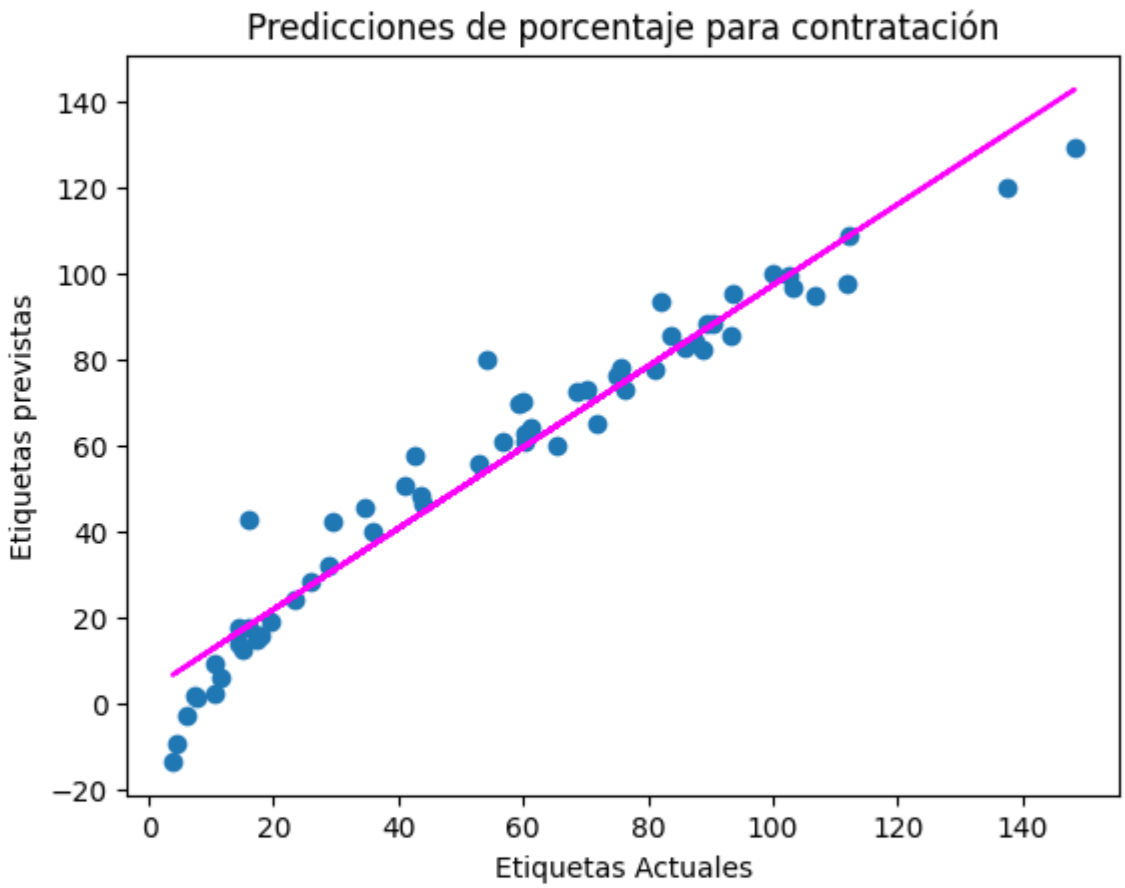
```
LinearRegression()
```

In [ ]:
```
import numpy as np

predictions = model.predict(X_test)
np.set_printoptions(suppress=True)
print('Etiqueta prevista: ', np.round(predictions)[:10])
print('Etiqueta Actual  : ' ,y_test[:10])
```

```
Etiqueta prevista:  [ -9.  70.   2.  14.  43.  78.  80. 109.  47.  48.]
Etiqueta Actual  :  [  4.522  59.42    7.61   14.31   16.005  81.03   54.165 112.38
44.05
  43.56 ]
```

Linea de regreción

In [ ]:
```
import matplotlib.pyplot as plt

%matplotlib inline

plt.scatter(y_test, predictions)
plt.xlabel('Etiquetas Actuales')
plt.ylabel('Etiquetas previstas')
plt.title('Predicciones de porcentaje para contratación')
#
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



Calculo de error predictivo por cada registro

In [ ]:
```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)

rmse = np.sqrt(mse)
print("RMSE:", rmse)

r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 75.62172675842041
RMSE: 8.696075365267967
R2: 0.9434267531488779
```

Modelo Experimental

Importar librerias nesesarias

In [ ]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Conjunto de datos de entrenamiento

In [ ]:
```
work_prediction = pd.read_csv('works_oferts.csv')
numeric_features = ["ningles",
                    "visap", "hofice", "estudios"]
categorical_features = ["exp", "node", "sql", "postgsql", "aws", "js"]
work_prediction[numeric_features + ['probabilidad']].describe()
print(work_prediction.head())
```

```
       name  edad  exp  node  sql  postgsql  aws  js  ningles  visap  hofice  \
0     Narmo    25    5    10    5         6    0   9    16.60      1       1
1   Gustavo    21    3     8    9         8    2   8    33.00      0       1
2  Mauricio    26    5    10    5         6    7   9    49.00      1       0
3   Gabriel    30    7    10    8         8    9  10    66.40      1       0
4         x    34    5    10   10        10   10  10    83.06      0       0

   estudios  probabilidad
0        50       60.2350
1        25       29.2800
2        15       43.6500
3        20       81.4660
4        15       60.3135
```

Caracteristicas separadas por matriz en "X" y "Y"

In [ ]:
```
x, y = work_prediction[["exp", "node", "sql", "postgsql",
                        "aws", "js", "ningles", "visap", "hofice", "estudios"]].val
```

Dividir los datos en 70-30

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(
    x, y, test_size=0.30, random_state=0)

print('Set de entrenamiento: %d Filas\nTest Set: %d Filas' %
      (X_train.shape[0], X_test.shape[0]))
```

```
Set de entrenamiento: 140 Filas
Test Set: 60 Filas
```

Modelo de lazo en para entramiento del modelo

In [ ]:
```
from sklearn.linear_model import Lasso

model = Lasso().fit(X_train, y_train)
print (model, "\n")
```

```
Lasso()
```

Prediccion con datos de prueva

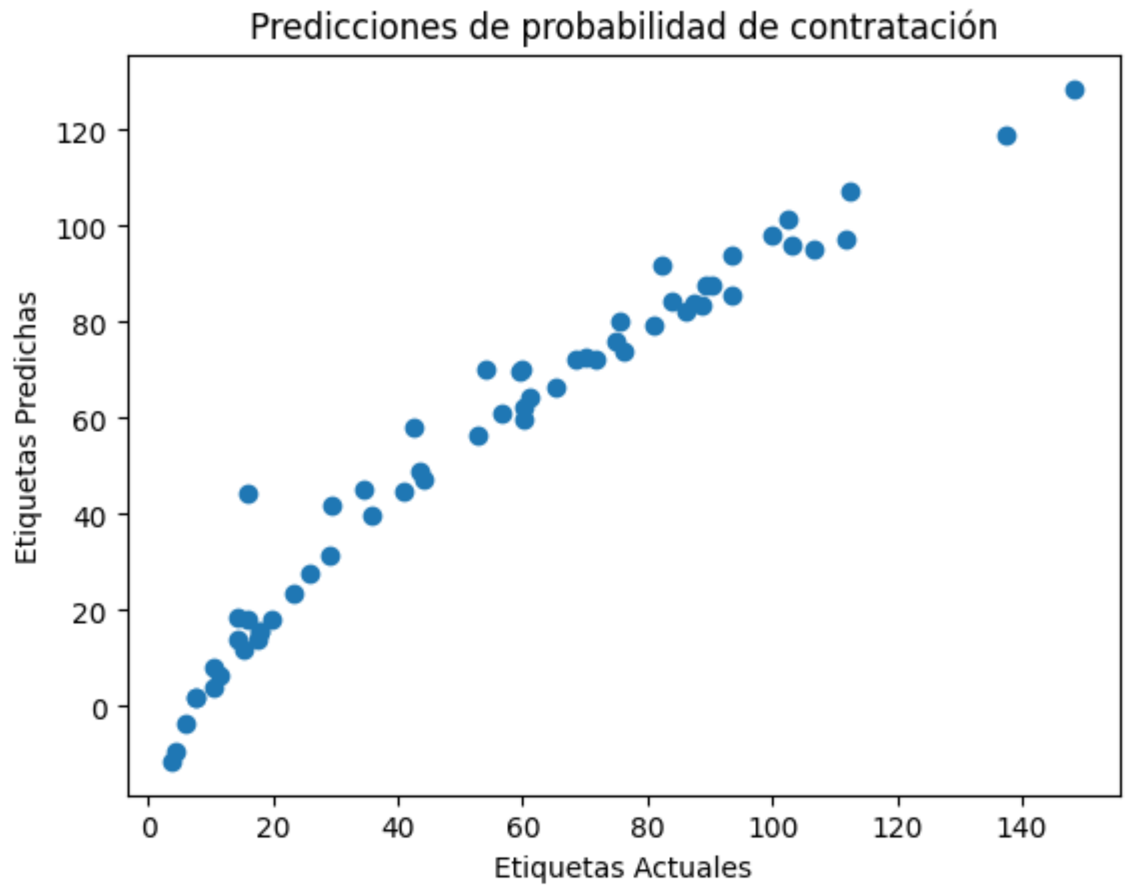
In [ ]:
```
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 67.47806586301988
RMSE: 8.214503385051337
R2: 0.9495190940389386
```

Grafico predicicio vs Grafico Real

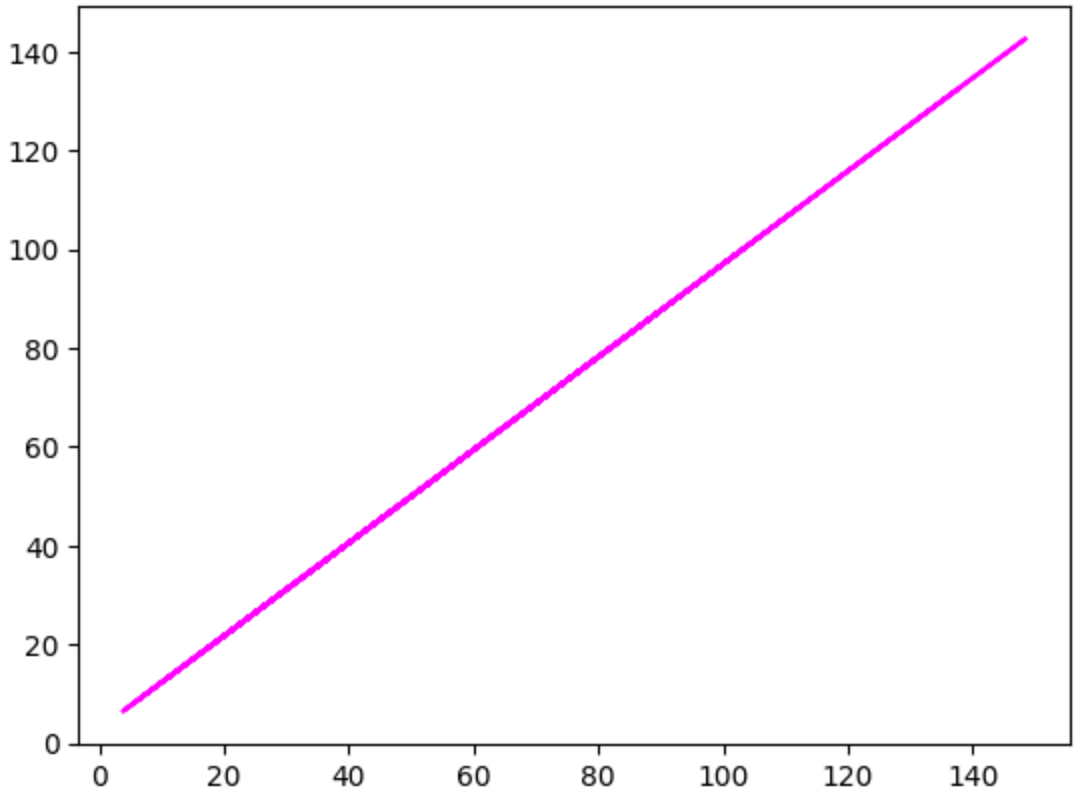
In [ ]:
```
plt.scatter(y_test, predictions)
plt.xlabel('Etiquetas Actuales')
plt.ylabel('Etiquetas Predichas')
plt.title('Predicciones de probabilidad de contratación')
```

Out[ ]:
```
Text(0.5, 1.0, 'Predicciones de probabilidad de contratación')
```



Linea de regreción

In [ ]:
```
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



Entrenar el moddelo

In [ ]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import export_text

model = DecisionTreeRegressor().fit(X_train, y_train)
print (model, "\n")
```

```
DecisionTreeRegressor()
```

Arbol de deciciones

In [ ]:
```
tree = export_text(model)
print(tree)
```

```
|--- feature_0 <= 6.50
|   |--- feature_0 <= 3.50
|   |   |--- feature_0 <= 2.50
|   |   |   |--- feature_9 <= 37.50
|   |   |   |   |--- feature_9 <= 7.50
|   |   |   |   |   |--- feature_3 <= 2.50
|   |   |   |   |   |   |--- feature_2 <= 0.50
|   |   |   |   |   |   |   |--- value: [0.00]
|   |   |   |   |   |   |--- feature_2 >  0.50
|   |   |   |   |   |   |   |--- value: [1.35]
|   |   |   |   |   |--- feature_3 >  2.50
|   |   |   |   |   |   |--- feature_1 <= 8.50
|   |   |   |   |   |   |   |--- value: [3.70]
|   |   |   |   |   |   |--- feature_1 >  8.50
|   |   |   |   |   |   |   |--- value: [4.00]
|   |   |   |   |--- feature_9 >  7.50
|   |   |   |   |   |--- feature_0 <= 1.50
|   |   |   |   |   |   |--- feature_6 <= 24.80
|   |   |   |   |   |   |   |--- feature_6 <= 8.30
|   |   |   |   |   |   |   |   |--- value: [6.03]
|   |   |   |   |   |   |   |--- feature_6 >  8.30
|   |   |   |   |   |   |   |   |--- value: [6.87]
|   |   |   |   |   |   |--- feature_6 >  24.80
|   |   |   |   |   |   |   |--- feature_3 <= 5.50
|   |   |   |   |   |   |   |   |--- feature_4 <= 1.50
|   |   |   |   |   |   |   |   |   |--- feature_9 <= 22.50
|   |   |   |   |   |   |   |   |   |   |--- value: [8.61]
|   |   |   |   |   |   |   |   |   |--- feature_9 >  22.50
|   |   |   |   |   |   |   |   |   |   |--- value: [8.66]
|   |   |   |   |   |   |   |   |--- feature_4 >  1.50
|   |   |   |   |   |   |   |   |   |--- value: [7.41]
|   |   |   |   |   |   |   |--- feature_3 >  5.50
|   |   |   |   |   |   |   |   |--- feature_6 <= 57.70
|   |   |   |   |   |   |   |   |   |--- value: [8.33]
|   |   |   |   |   |   |   |   |--- feature_6 >  57.70
|   |   |   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |   |   |--- value: [9.76]
|   |   |   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [10.54]
|   |   |   |   |   |   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [9.89]
|   |   |   |   |   |--- feature_0 >  1.50
|   |   |   |   |   |   |--- feature_6 <= 57.70
|   |   |   |   |   |   |   |--- feature_9 <= 22.50
|   |   |   |   |   |   |   |   |--- feature_6 <= 41.00
|   |   |   |   |   |   |   |   |   |--- feature_5 <= 2.50
|   |   |   |   |   |   |   |   |   |   |--- value: [10.22]
|   |   |   |   |   |   |   |   |   |--- feature_5 >  2.50
|   |   |   |   |   |   |   |   |   |   |--- feature_6 <= 24.80
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 3
|   |   |   |   |   |   |   |   |   |   |--- feature_6 >  24.80
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |--- feature_6 >  41.00
|   |   |   |   |   |   |   |   |   |--- value: [12.16]
|   |   |   |   |   |   |   |--- feature_9 >  22.50
|   |   |   |   |   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |   |   |   |   |--- value: [13.62]
|   |   |   |   |   |   |   |   |--- feature_8 >  0.50
```

```
|   |   |   |   |   |   |   |   |   |--- value: [11.65]
|   |   |   |   |   |   |--- feature_6 >  57.70
|   |   |   |   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |   |   |   |--- value: [15.83]
|   |   |   |   |   |   |   |--- feature_8 >  0.50
|   |   |   |   |   |   |   |   |--- value: [15.28]
|   |   |   |--- feature_9 >  37.50
|   |   |   |   |--- feature_0 <= 1.50
|   |   |   |   |   |--- feature_3 <= 5.50
|   |   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |   |--- value: [13.55]
|   |   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |   |--- value: [11.90]
|   |   |   |   |   |--- feature_3 >  5.50
|   |   |   |   |   |   |--- value: [15.90]
|   |   |   |   |--- feature_0 >  1.50
|   |   |   |   |   |--- feature_3 <= 2.50
|   |   |   |   |   |   |--- value: [20.09]
|   |   |   |   |   |--- feature_3 >  2.50
|   |   |   |   |   |   |--- feature_2 <= 1.50
|   |   |   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |   |   |--- value: [28.11]
|   |   |   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |   |   |--- value: [28.31]
|   |   |   |   |   |   |--- feature_2 >  1.50
|   |   |   |   |   |   |   |--- value: [28.41]
|   |   |--- feature_0 >  2.50
|   |   |   |--- feature_6 <= 8.30
|   |   |   |   |--- value: [7.95]
|   |   |   |--- feature_6 >  8.30
|   |   |   |   |--- feature_9 <= 22.50
|   |   |   |   |   |--- feature_6 <= 41.00
|   |   |   |   |   |   |--- feature_6 <= 24.80
|   |   |   |   |   |   |   |--- value: [15.52]
|   |   |   |   |   |   |--- feature_6 >  24.80
|   |   |   |   |   |   |   |--- feature_4 <= 3.50
|   |   |   |   |   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |   |   |   |   |--- value: [17.43]
|   |   |   |   |   |   |   |   |--- feature_8 >  0.50
|   |   |   |   |   |   |   |   |   |--- value: [17.73]
|   |   |   |   |   |   |   |--- feature_4 >  3.50
|   |   |   |   |   |   |   |   |--- value: [19.38]
|   |   |   |   |   |--- feature_6 >  41.00
|   |   |   |   |   |   |--- feature_6 <= 57.70
|   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |--- value: [20.19]
|   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |--- feature_2 <= 2.50
|   |   |   |   |   |   |   |   |   |--- value: [21.91]
|   |   |   |   |   |   |   |   |--- feature_2 >  2.50
|   |   |   |   |   |   |   |   |   |--- value: [21.77]
|   |   |   |   |   |   |--- feature_6 >  57.70
|   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |--- feature_3 <= 2.00
|   |   |   |   |   |   |   |   |   |--- value: [24.19]
|   |   |   |   |   |   |   |   |--- feature_3 >  2.00
|   |   |   |   |   |   |   |   |   |--- value: [24.49]
|   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |--- value: [24.86]
```

```
|   |   |   |   |--- feature_9 >  22.50
|   |   |   |   |   |--- feature_6 <= 24.80
|   |   |   |   |   |   |--- value: [31.79]
|   |   |   |   |   |--- feature_6 >  24.80
|   |   |   |   |   |   |--- feature_4 <= 2.50
|   |   |   |   |   |   |   |--- value: [29.28]
|   |   |   |   |   |   |--- feature_4 >  2.50
|   |   |   |   |   |   |   |--- value: [22.23]
|   |--- feature_0 >  3.50
|   |   |--- feature_9 <= 37.50
|   |   |   |--- feature_2 <= 4.50
|   |   |   |   |--- feature_6 <= 24.80
|   |   |   |   |   |--- feature_9 <= 7.50
|   |   |   |   |   |   |--- value: [14.00]
|   |   |   |   |   |--- feature_9 >  7.50
|   |   |   |   |   |   |--- feature_0 <= 4.50
|   |   |   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |   |   |--- value: [22.89]
|   |   |   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |   |   |--- value: [23.89]
|   |   |   |   |   |   |--- feature_0 >  4.50
|   |   |   |   |   |   |   |--- value: [29.86]
|   |   |   |   |--- feature_6 >  24.80
|   |   |   |   |   |--- feature_0 <= 4.50
|   |   |   |   |   |   |--- feature_6 <= 41.00
|   |   |   |   |   |   |   |--- feature_9 <= 20.00
|   |   |   |   |   |   |   |   |--- feature_4 <= 4.50
|   |   |   |   |   |   |   |   |   |--- value: [26.24]
|   |   |   |   |   |   |   |   |--- feature_4 >  4.50
|   |   |   |   |   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- value: [26.44]
|   |   |   |   |   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- value: [26.84]
|   |   |   |   |   |   |   |--- feature_9 >  20.00
|   |   |   |   |   |   |   |   |--- value: [32.04]
|   |   |   |   |   |   |--- feature_6 >  41.00
|   |   |   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |   |   |--- feature_6 <= 66.03
|   |   |   |   |   |   |   |   |   |--- feature_4 <= 4.50
|   |   |   |   |   |   |   |   |   |   |--- value: [31.82]
|   |   |   |   |   |   |   |   |   |--- feature_4 >  4.50
|   |   |   |   |   |   |   |   |   |   |--- value: [32.42]
|   |   |   |   |   |   |   |   |--- feature_6 >  66.03
|   |   |   |   |   |   |   |   |   |--- value: [35.25]
|   |   |   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |   |   |--- value: [28.92]
|   |   |   |   |   |--- feature_0 >  4.50
|   |   |   |   |   |   |--- feature_6 <= 57.70
|   |   |   |   |   |   |   |--- feature_6 <= 41.00
|   |   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- value: [33.05]
|   |   |   |   |   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- value: [33.55]
|   |   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |   |--- value: [36.92]
|   |   |   |   |   |   |   |--- feature_6 >  41.00
|   |   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |   |--- value: [37.15]
```

```
|   |   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |   |--- value: [40.52]
|   |   |   |   |   |   |--- feature_6 >  57.70
|   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |--- value: [44.81]
|   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |--- value: [44.69]
|   |   |   |--- feature_2 >  4.50
|   |   |   |   |--- feature_9 <= 7.50
|   |   |   |   |   |--- value: [21.30]
|   |   |   |   |--- feature_9 >  7.50
|   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |--- feature_6 <= 41.00
|   |   |   |   |   |   |   |--- feature_0 <= 5.50
|   |   |   |   |   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |   |   |   |   |--- value: [36.30]
|   |   |   |   |   |   |   |   |--- feature_8 >  0.50
|   |   |   |   |   |   |   |   |   |--- value: [36.80]
|   |   |   |   |   |   |   |--- feature_0 >  5.50
|   |   |   |   |   |   |   |   |--- feature_6 <= 24.80
|   |   |   |   |   |   |   |   |   |--- feature_3 <= 5.00
|   |   |   |   |   |   |   |   |   |   |--- value: [39.73]
|   |   |   |   |   |   |   |   |   |--- feature_3 >  5.00
|   |   |   |   |   |   |   |   |   |   |--- value: [40.33]
|   |   |   |   |   |   |   |   |--- feature_6 >  24.80
|   |   |   |   |   |   |   |   |   |--- feature_1 <= 2.50
|   |   |   |   |   |   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [43.56]
|   |   |   |   |   |   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [44.16]
|   |   |   |   |   |   |   |   |   |--- feature_1 >  2.50
|   |   |   |   |   |   |   |   |   |   |--- value: [47.46]
|   |   |   |   |   |   |--- feature_6 >  41.00
|   |   |   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |   |   |--- value: [57.08]
|   |   |   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |   |   |--- feature_1 <= 6.50
|   |   |   |   |   |   |   |   |   |--- feature_5 <= 7.50
|   |   |   |   |   |   |   |   |   |   |--- value: [48.18]
|   |   |   |   |   |   |   |   |   |--- feature_5 >  7.50
|   |   |   |   |   |   |   |   |   |   |--- value: [52.38]
|   |   |   |   |   |   |   |   |--- feature_1 >  6.50
|   |   |   |   |   |   |   |   |   |--- value: [43.65]
|   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |--- feature_0 <= 5.50
|   |   |   |   |   |   |   |--- value: [43.77]
|   |   |   |   |   |   |--- feature_0 >  5.50
|   |   |   |   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |   |   |   |--- feature_5 <= 7.50
|   |   |   |   |   |   |   |   |   |--- value: [52.53]
|   |   |   |   |   |   |   |   |--- feature_5 >  7.50
|   |   |   |   |   |   |   |   |   |--- value: [56.43]
|   |   |   |   |   |   |   |--- feature_8 >  0.50
|   |   |   |   |   |   |   |   |--- feature_1 <= 2.50
|   |   |   |   |   |   |   |   |   |--- value: [57.83]
|   |   |   |   |   |   |   |   |--- feature_1 >  2.50
|   |   |   |   |   |   |   |   |   |--- value: [57.06]
|   |   |--- feature_9 >  37.50
|   |   |   |--- feature_4 <= 5.50
```

```
|   |   |   |   |--- feature_0 <= 4.50
|   |   |   |   |   |--- feature_1 <= 5.50
|   |   |   |   |   |   |--- value: [45.59]
|   |   |   |   |   |--- feature_1 >  5.50
|   |   |   |   |   |   |--- value: [44.99]
|   |   |   |   |--- feature_0 >  4.50
|   |   |   |   |   |--- feature_3 <= 4.50
|   |   |   |   |   |   |--- value: [56.98]
|   |   |   |   |   |--- feature_3 >  4.50
|   |   |   |   |   |   |--- value: [60.23]
|   |   |   |--- feature_4 >  5.50
|   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |--- value: [94.53]
|   |   |   |   |--- feature_8 >  0.50
|   |   |   |   |   |--- value: [72.28]
|--- feature_0 >  6.50
|   |--- feature_6 <= 91.03
|   |   |--- feature_9 <= 7.50
|   |   |   |--- feature_7 <= 0.50
|   |   |   |   |--- feature_4 <= 8.50
|   |   |   |   |   |--- feature_5 <= 5.50
|   |   |   |   |   |   |--- value: [42.00]
|   |   |   |   |   |--- feature_5 >  5.50
|   |   |   |   |   |   |--- value: [38.00]
|   |   |   |   |--- feature_4 >  8.50
|   |   |   |   |   |--- feature_2 <= 8.50
|   |   |   |   |   |   |--- value: [48.60]
|   |   |   |   |   |--- feature_2 >  8.50
|   |   |   |   |   |   |--- value: [50.50]
|   |   |   |--- feature_7 >  0.50
|   |   |   |   |--- value: [62.70]
|   |   |--- feature_9 >  7.50
|   |   |   |--- feature_2 <= 7.50
|   |   |   |   |--- feature_9 <= 37.50
|   |   |   |   |   |--- feature_5 <= 8.50
|   |   |   |   |   |   |--- feature_6 <= 41.00
|   |   |   |   |   |   |   |--- feature_9 <= 20.00
|   |   |   |   |   |   |   |   |--- feature_3 <= 6.00
|   |   |   |   |   |   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |   |   |   |   |   |--- value: [55.37]
|   |   |   |   |   |   |   |   |   |--- feature_8 >  0.50
|   |   |   |   |   |   |   |   |   |   |--- value: [56.07]
|   |   |   |   |   |   |   |   |--- feature_3 >  6.00
|   |   |   |   |   |   |   |   |   |--- value: [51.60]
|   |   |   |   |   |   |   |--- feature_9 >  20.00
|   |   |   |   |   |   |   |   |--- value: [66.22]
|   |   |   |   |   |   |--- feature_6 >  41.00
|   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |--- value: [71.14]
|   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |--- value: [65.83]
|   |   |   |   |   |--- feature_5 >  8.50
|   |   |   |   |   |   |--- feature_3 <= 7.00
|   |   |   |   |   |   |   |--- feature_6 <= 41.00
|   |   |   |   |   |   |   |   |--- feature_9 <= 20.00
|   |   |   |   |   |   |   |   |   |--- value: [68.48]
|   |   |   |   |   |   |   |   |--- feature_9 >  20.00
|   |   |   |   |   |   |   |   |   |--- value: [80.88]
|   |   |   |   |   |   |   |--- feature_6 >  41.00
```

```
|   |   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |   |--- value: [86.50]
|   |   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |   |--- value: [80.44]
|   |   |   |   |   |   |--- feature_3 >  7.00
|   |   |   |   |   |   |   |--- feature_6 <= 57.70
|   |   |   |   |   |   |   |   |--- feature_8 <= 0.50
|   |   |   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |   |   |--- value: [69.28]
|   |   |   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |   |   |--- feature_9 <= 22.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [70.39]
|   |   |   |   |   |   |   |   |   |   |--- feature_9 >  22.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [70.77]
|   |   |   |   |   |   |   |   |--- feature_8 >  0.50
|   |   |   |   |   |   |   |   |   |--- value: [64.18]
|   |   |   |   |   |   |   |--- feature_6 >  57.70
|   |   |   |   |   |   |   |   |--- value: [76.57]
|   |   |   |   |--- feature_9 >  37.50
|   |   |   |   |   |--- value: [99.08]
|   |   |   |--- feature_2 >  7.50
|   |   |   |   |--- feature_9 <= 37.50
|   |   |   |   |   |--- feature_7 <= 0.50
|   |   |   |   |   |   |--- feature_3 <= 8.50
|   |   |   |   |   |   |   |--- feature_0 <= 9.50
|   |   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |   |--- value: [103.16]
|   |   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |   |--- feature_9 <= 22.50
|   |   |   |   |   |   |   |   |   |   |--- value: [96.34]
|   |   |   |   |   |   |   |   |   |--- feature_9 >  22.50
|   |   |   |   |   |   |   |   |   |   |--- value: [96.84]
|   |   |   |   |   |   |   |--- feature_0 >  9.50
|   |   |   |   |   |   |   |   |--- feature_9 <= 22.50
|   |   |   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |   |   |--- value: [111.13]
|   |   |   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |   |   |--- value: [118.88]
|   |   |   |   |   |   |   |   |--- feature_9 >  22.50
|   |   |   |   |   |   |   |   |   |--- value: [104.10]
|   |   |   |   |   |   |--- feature_3 >  8.50
|   |   |   |   |   |   |   |--- feature_9 <= 17.50
|   |   |   |   |   |   |   |   |--- feature_6 <= 58.03
|   |   |   |   |   |   |   |   |   |--- value: [89.60]
|   |   |   |   |   |   |   |   |--- feature_6 >  58.03
|   |   |   |   |   |   |   |   |   |--- value: [91.70]
|   |   |   |   |   |   |   |--- feature_9 >  17.50
|   |   |   |   |   |   |   |   |--- feature_0 <= 9.50
|   |   |   |   |   |   |   |   |   |--- value: [97.25]
|   |   |   |   |   |   |   |   |--- feature_0 >  9.50
|   |   |   |   |   |   |   |   |   |--- value: [104.55]
|   |   |   |   |   |--- feature_7 >  0.50
|   |   |   |   |   |   |--- feature_6 <= 57.70
|   |   |   |   |   |   |   |--- feature_2 <= 8.50
|   |   |   |   |   |   |   |   |--- feature_6 <= 24.80
|   |   |   |   |   |   |   |   |   |--- value: [77.15]
|   |   |   |   |   |   |   |   |--- feature_6 >  24.80
|   |   |   |   |   |   |   |   |   |--- feature_6 <= 41.00
|   |   |   |   |   |   |   |   |   |   |--- value: [84.69]
```

```
|   |   |   |   |   |   |   |   |   |--- feature_6 >  41.00
|   |   |   |   |   |   |   |   |   |   |--- value: [80.24]
|   |   |   |   |   |   |   |--- feature_2 >  8.50
|   |   |   |   |   |   |   |   |--- feature_0 <= 9.50
|   |   |   |   |   |   |   |   |   |--- value: [87.12]
|   |   |   |   |   |   |   |   |--- feature_0 >  9.50
|   |   |   |   |   |   |   |   |   |--- feature_2 <= 9.50
|   |   |   |   |   |   |   |   |   |   |--- feature_6 <= 41.00
|   |   |   |   |   |   |   |   |   |   |   |--- value: [89.60]
|   |   |   |   |   |   |   |   |   |   |--- feature_6 >  41.00
|   |   |   |   |   |   |   |   |   |   |   |--- value: [96.80]
|   |   |   |   |   |   |   |   |   |--- feature_2 >  9.50
|   |   |   |   |   |   |   |   |   |   |--- value: [88.30]
|   |   |   |   |   |   |--- feature_6 >  57.70
|   |   |   |   |   |   |   |--- feature_0 <= 8.50
|   |   |   |   |   |   |   |   |--- feature_1 <= 7.50
|   |   |   |   |   |   |   |   |   |--- value: [92.30]
|   |   |   |   |   |   |   |   |--- feature_1 >  7.50
|   |   |   |   |   |   |   |   |   |--- value: [81.47]
|   |   |   |   |   |   |   |--- feature_0 >  8.50
|   |   |   |   |   |   |   |   |--- feature_3 <= 9.00
|   |   |   |   |   |   |   |   |   |--- feature_2 <= 9.50
|   |   |   |   |   |   |   |   |   |   |--- feature_2 <= 8.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [104.29]
|   |   |   |   |   |   |   |   |   |   |--- feature_2 >  8.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [104.63]
|   |   |   |   |   |   |   |   |   |--- feature_2 >  9.50
|   |   |   |   |   |   |   |   |   |   |--- value: [103.88]
|   |   |   |   |   |   |   |   |--- feature_3 >  9.00
|   |   |   |   |   |   |   |   |   |--- value: [101.14]
|   |   |   |   |--- feature_9 >  37.50
|   |   |   |   |   |--- value: [125.97]
|   |--- feature_6 >  91.03
|   |   |--- feature_2 <= 7.50
|   |   |   |--- feature_8 <= 0.50
|   |   |   |   |--- value: [135.64]
|   |   |   |--- feature_8 >  0.50
|   |   |   |   |--- value: [114.48]
|   |   |--- feature_2 >  7.50
|   |   |   |--- feature_9 <= 32.50
|   |   |   |   |--- value: [118.30]
|   |   |   |--- feature_9 >  32.50
|   |   |   |   |--- feature_4 <= 9.50
|   |   |   |   |   |--- feature_1 <= 6.00
|   |   |   |   |   |   |--- value: [158.90]
|   |   |   |   |   |--- feature_1 >  6.00
|   |   |   |   |   |   |--- value: [164.55]
|   |   |   |   |--- feature_4 >  9.50
|   |   |   |   |   |--- value: [172.55]
```

Evaluar el modelo con datos de prueba

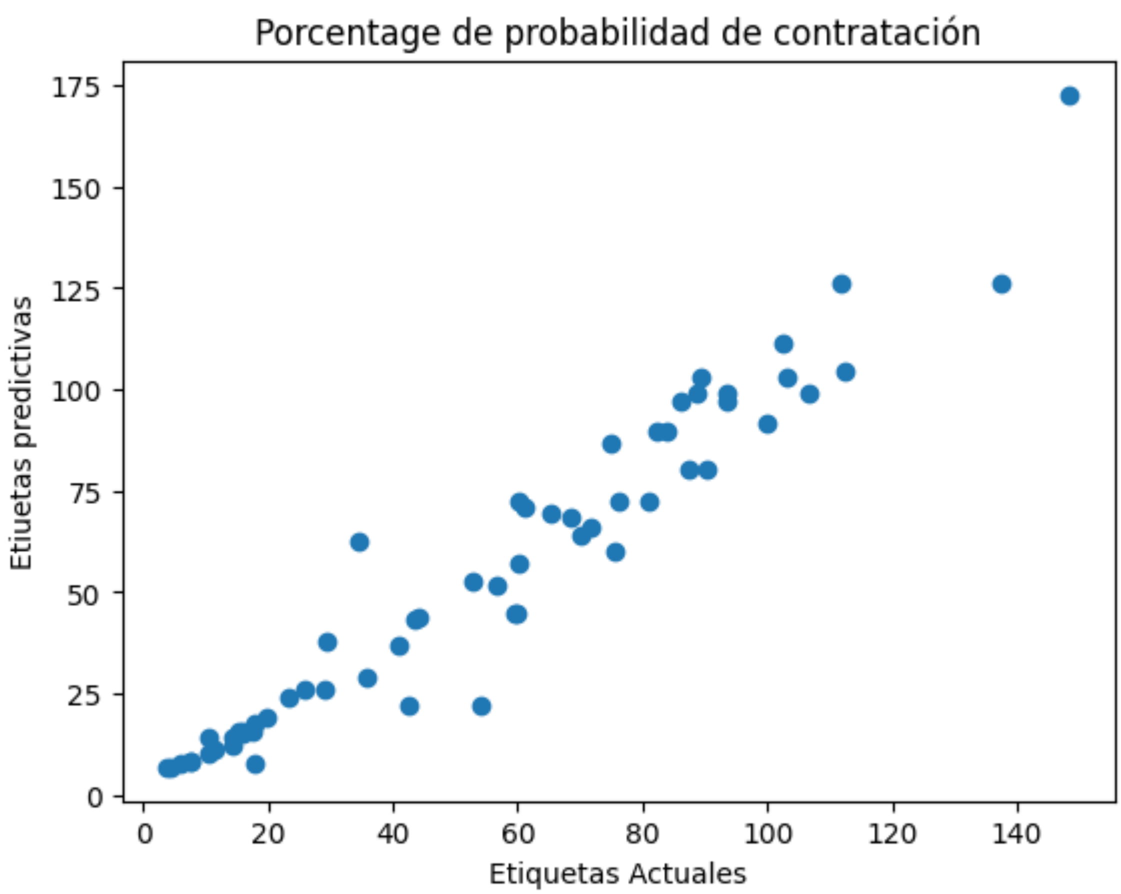
In [ ]:
```
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 93.93697166400003
RMSE: 9.692108731540316
R2: 0.9297249651099433
```

Grafico Prediccititicio vs Grafico Real

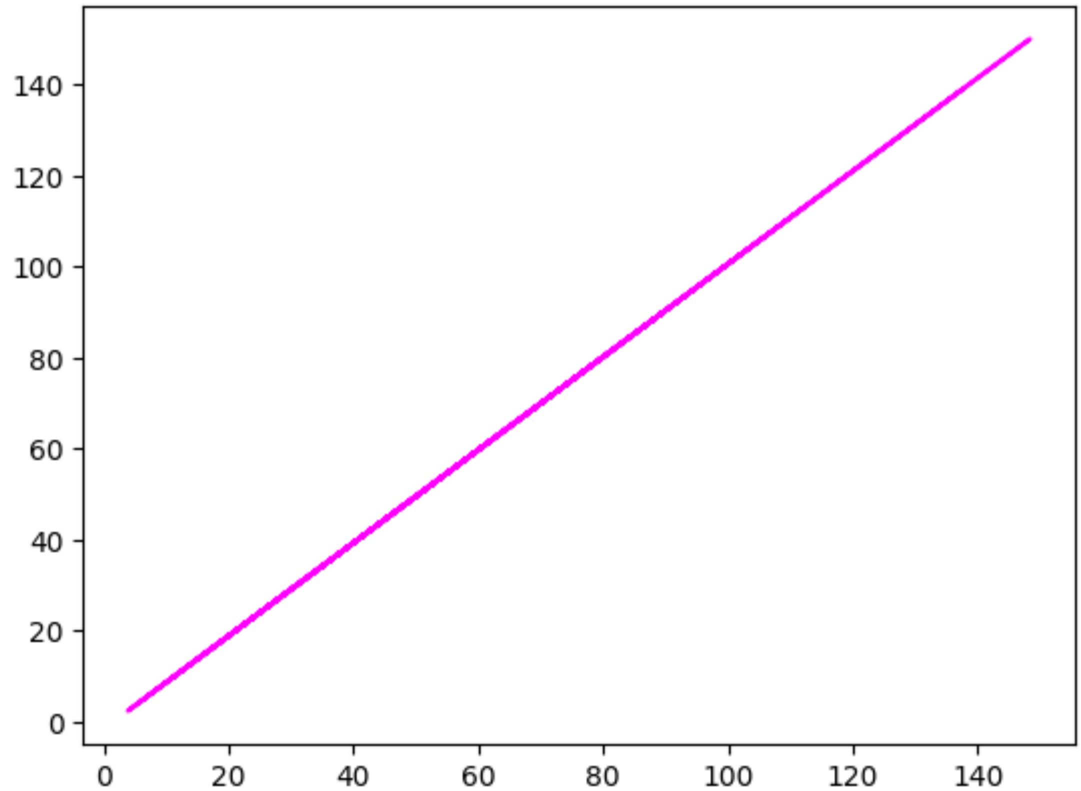
In [ ]:
```
plt.scatter(y_test, predictions)
plt.xlabel('Etiquetas Actuales')
plt.ylabel('Etiuetas predictivas')
plt.title('Porcentage de probabilidad de contratación')
```

Out[ ]:
```
Text(0.5, 1.0, 'Porcentage de probabilidad de contratación')
```



Linea de regreción

In [ ]:
```
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



Entrenar modelo con datos de prueba

In [ ]:
```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor().fit(X_train, y_train)
print (model, "\n")
```

```
RandomForestRegressor()
```

Evaluacion de modelo con datos de prueba

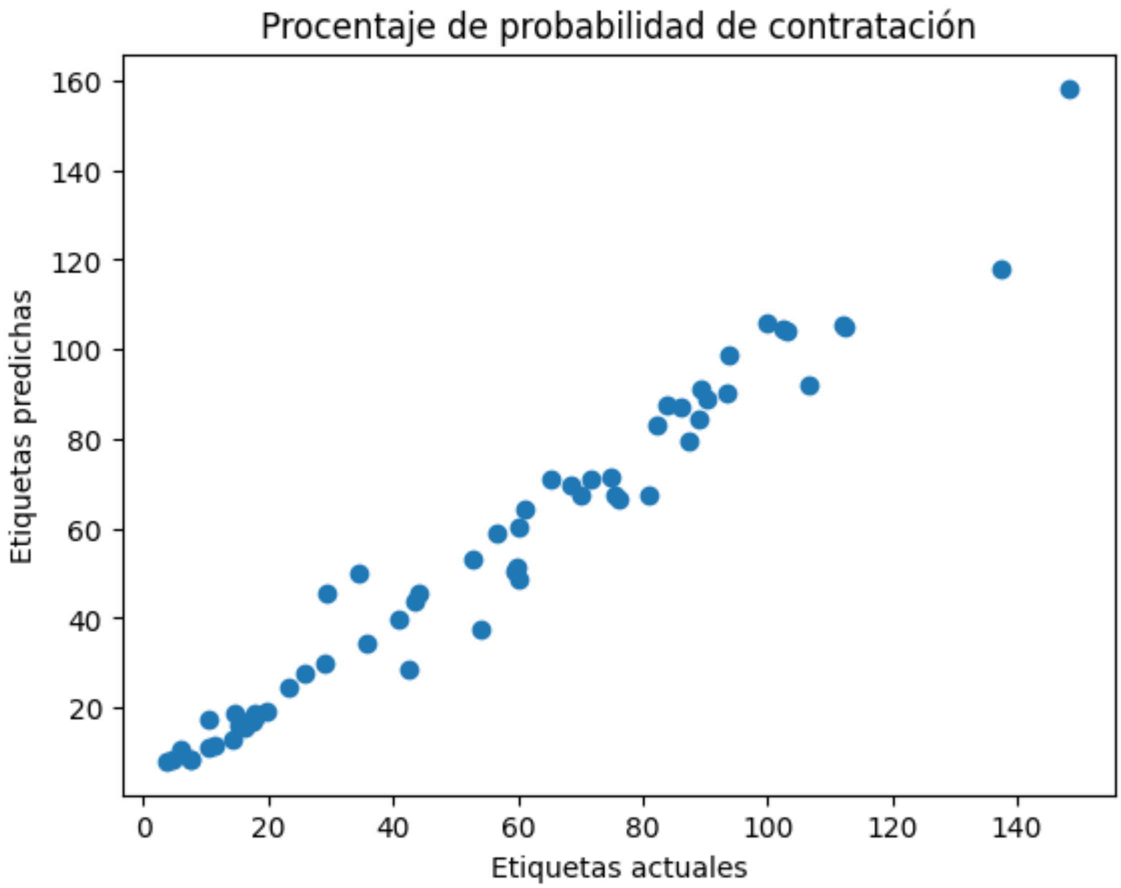
In [ ]:
```
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 46.26222103183748
RMSE: 6.801633703150845
R2: 0.9653908451644303
```

Gráfico predicho vs real

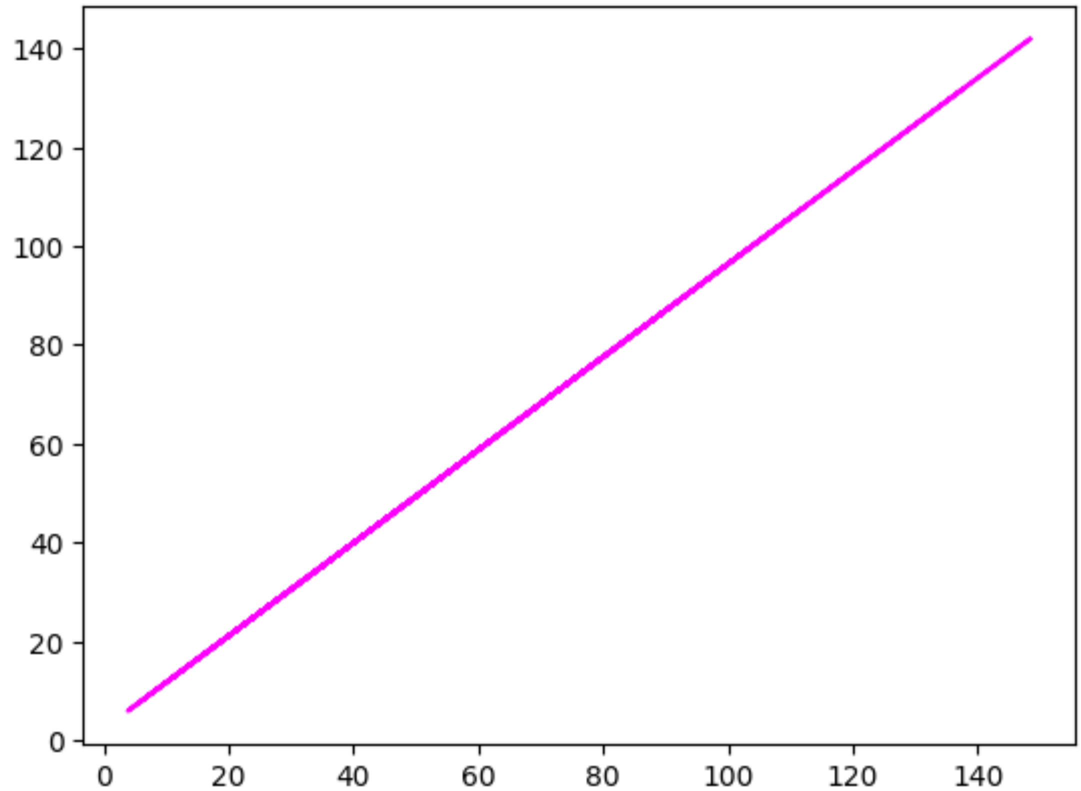
In [ ]:
```
plt.scatter(y_test, predictions)
plt.xlabel('Etiquetas actuales')
plt.ylabel('Etiquetas predichas')
plt.title('Procentaje de probabilidad de contratación')
```

Out[ ]:
```
Text(0.5, 1.0, 'Procentaje de probabilidad de contratación')
```



Linea de regreción

In [ ]:
```
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



Entrenamiento de modelo

In [ ]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

Modelo de lazo

In [ ]:
```
model = GradientBoostingRegressor().fit(X_train, y_train)
print (model, "\n")
```

```
GradientBoostingRegressor()
```

Evaluación de modelo con datos de prueba

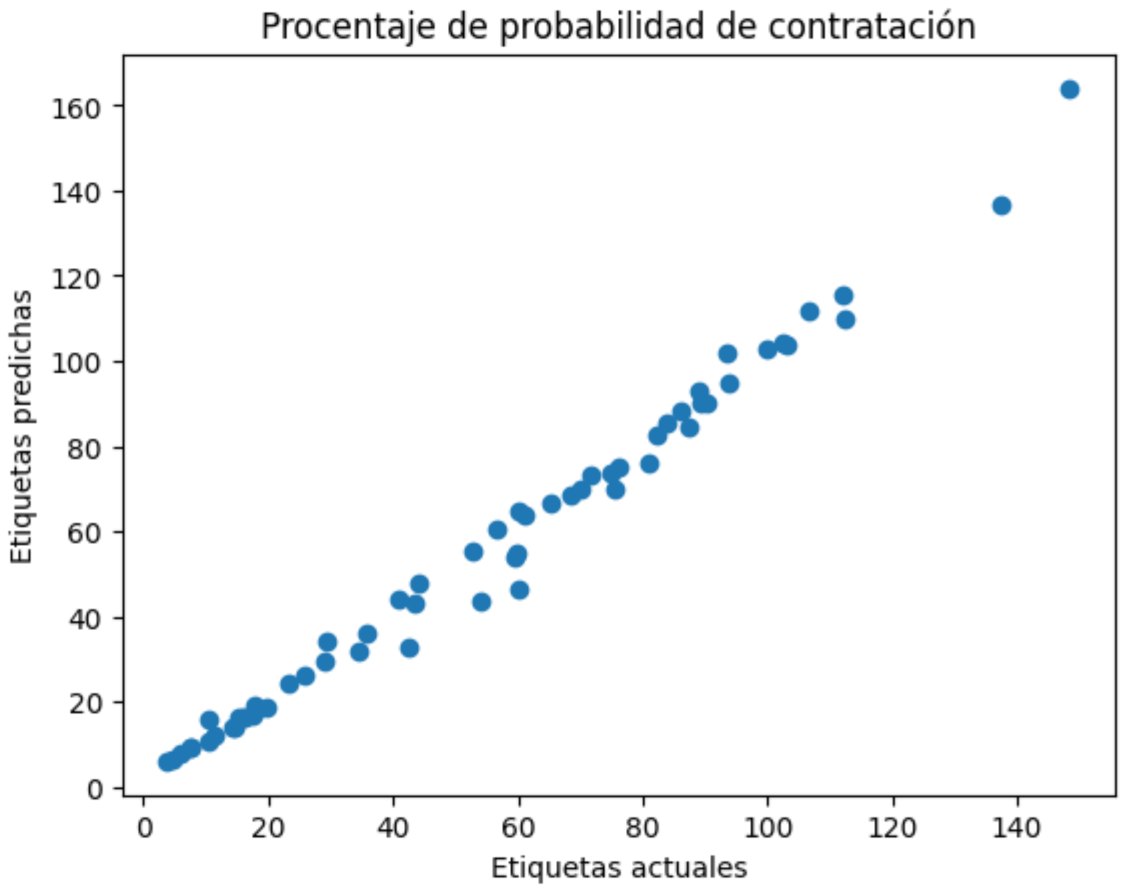
In [ ]:
```
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 17.81275422532956
RMSE: 4.220515871943803
R2: 0.9866741294455337
```

Grafico predicticio vs real

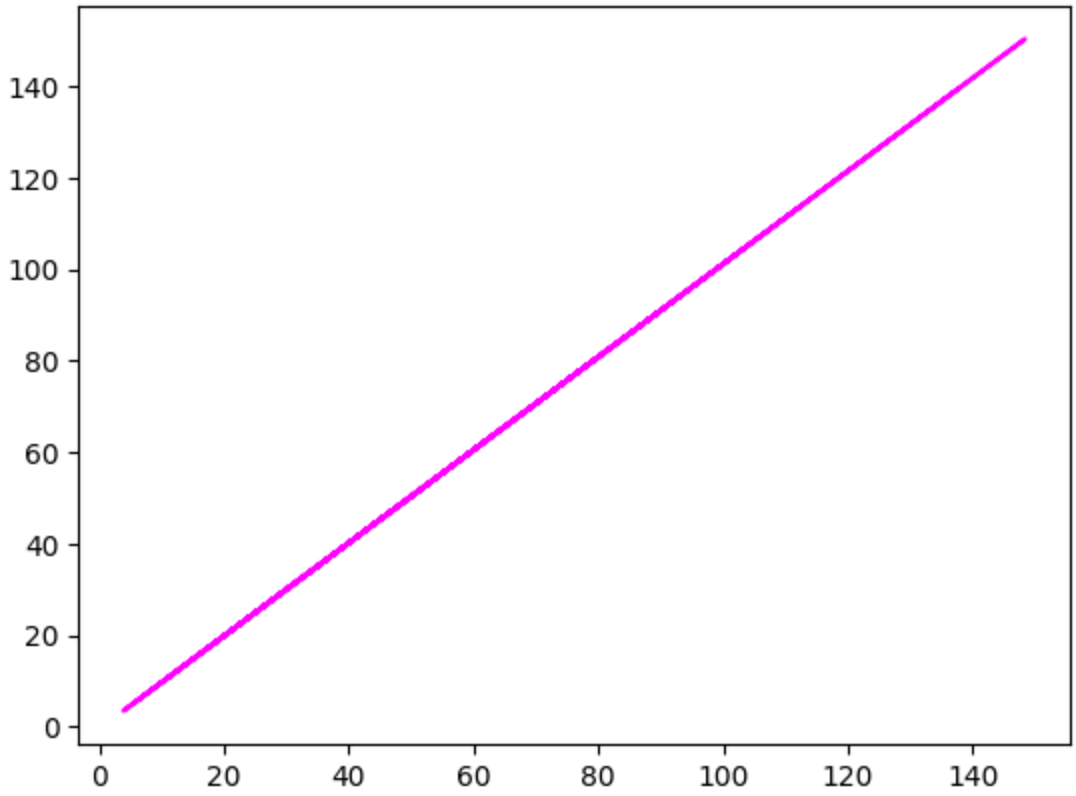
In [ ]:
```
plt.scatter(y_test, predictions)
plt.xlabel('Etiquetas actuales')
plt.ylabel('Etiquetas predichas')
plt.title('Procentaje de probabilidad de contratación')
```

Out[ ]:
```
Text(0.5, 1.0, 'Procentaje de probabilidad de contratación')
```



Regrecion

In [ ]:
```
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



Guardar modelo

In [ ]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Cargar el conjunto de datos de entrenamiento

In [ ]:
```
work_prediction = pd.read_csv('works_oferts.csv')
numeric_features = ["ningles",
                    "visap", "hofice", "estudios"]
categorical_features = ["exp", "node", "sql", "postgsql", "aws", "js"]
work_prediction[numeric_features + ['probabilidad']].describe()
print(work_prediction.head())
```

```
       name  edad  exp  node  sql  postgsql  aws  js  ningles  visap  hofice  \
0     Narmo    25    5    10    5         6    0   9    16.60      1       1
1   Gustavo    21    3     8    9         8    2   8    33.00      0       1
2  Mauricio    26    5    10    5         6    7   9    49.00      1       0
3   Gabriel    30    7    10    8         8    9  10    66.40      1       0
4         x    34    5    10   10        10   10  10    83.06      0       0

   estudios  probabilidad
0        50       60.2350
1        25       29.2800
2        15       43.6500
3        20       81.4660
4        15       60.3135
```

Separar por matrises de datos X & Y

In [ ]:
```
x, y = work_prediction[["exp", "node", "sql", "postgsql",
                        "aws", "js", "ningles", "visap", "hofice", "estudios"]].val
```

Dividir los datos 70%-30% en conjunto de entrenamiento y conjunto de prueba

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_st

print ('Training Set: %d rows\nTest Set: %d rows' % (X_train.shape[0], X_test.shape
```

```
Training Set: 140 rows
Test Set: 60 rows
```

Entrenar el modelo

In [ ]:
```
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
```

Encajar modelo de lazo

In [ ]:
```
model = GradientBoostingRegressor().fit(X_train, y_train)
print (model, "\n")
```

```
GradientBoostingRegressor()
```

Evaluiar entrenamiento con datos de prueba

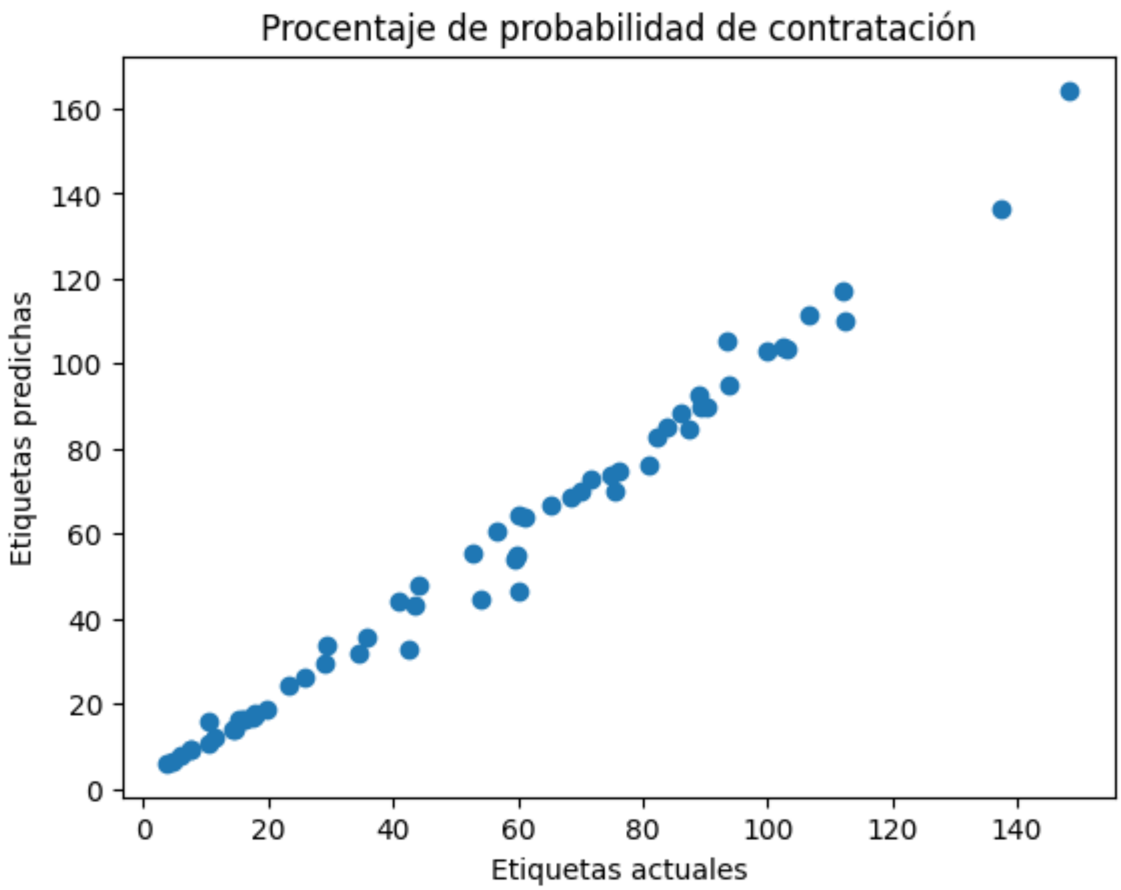
In [ ]:
```
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 19.112771873590887
RMSE: 4.371815626669415
R2: 0.9857015753598427
```

Grafico predicticio vs Grafico real

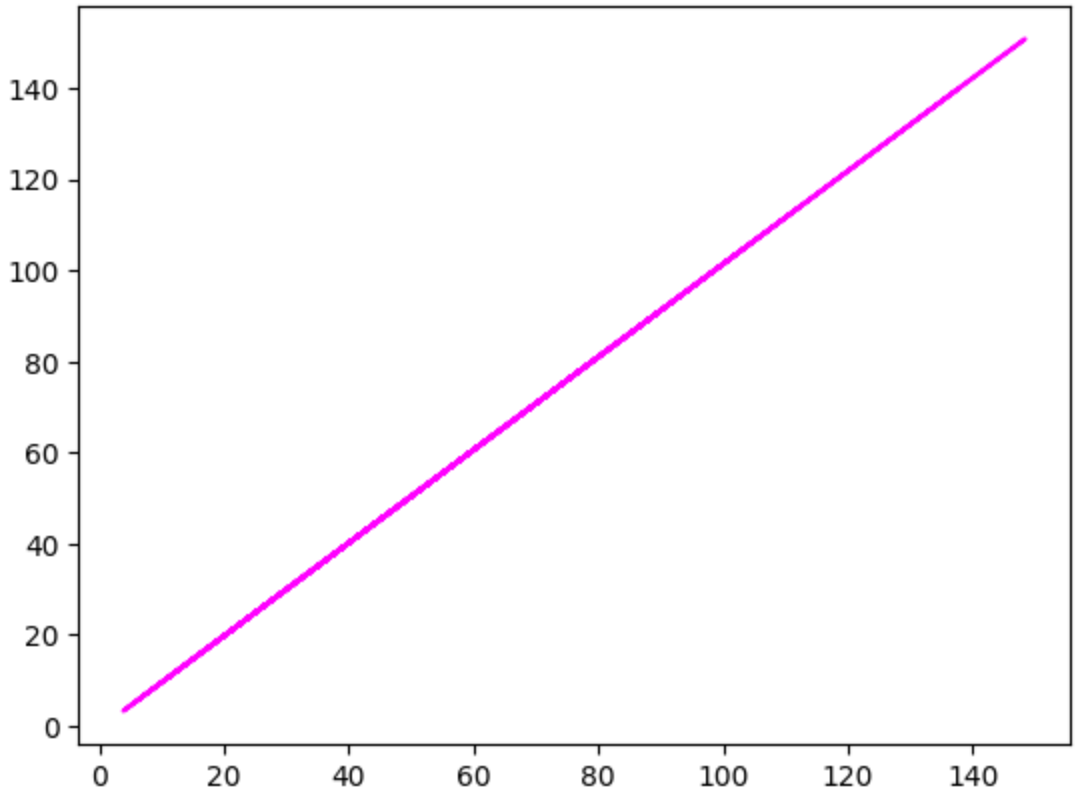
In [ ]:
```
plt.scatter(y_test, predictions)
plt.xlabel('Etiquetas actuales')
plt.ylabel('Etiquetas predichas')
plt.title('Procentaje de probabilidad de contratación')
```

Out[ ]:
```
Text(0.5, 1.0, 'Procentaje de probabilidad de contratación')
```



Linea de regreción

In [ ]:
```
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



Usar algoritmo como un aumento de gradiante

In [ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, r2_score
alg = GradientBoostingRegressor()
```

In [ ]:
```
params = {
 'learning_rate': [0.1, 0.5, 1.0],
 'n_estimators' : [50, 100, 150]
 }
```

Encontrar las mejores combinaciones de hiperparamtetros para optimizar el valor de R2

In [ ]:
```
score = make_scorer(r2_score)
gridsearch = GridSearchCV(alg, params, scoring=score, cv=3, return_train_score=True
gridsearch.fit(X_train, y_train)
print("Mejor combinación de valores:", gridsearch.best_params_, "\n")
```

```
Mejor combinación de valores: {'learning_rate': 0.1, 'n_estimators': 150}
```

Obtener el modelo predicticio

In [ ]:
```
model=gridsearch.best_estimator_
print(model, "\n")
```

```
GradientBoostingRegressor(n_estimators=150)
```

Evaluar el modelo con datos de prueba

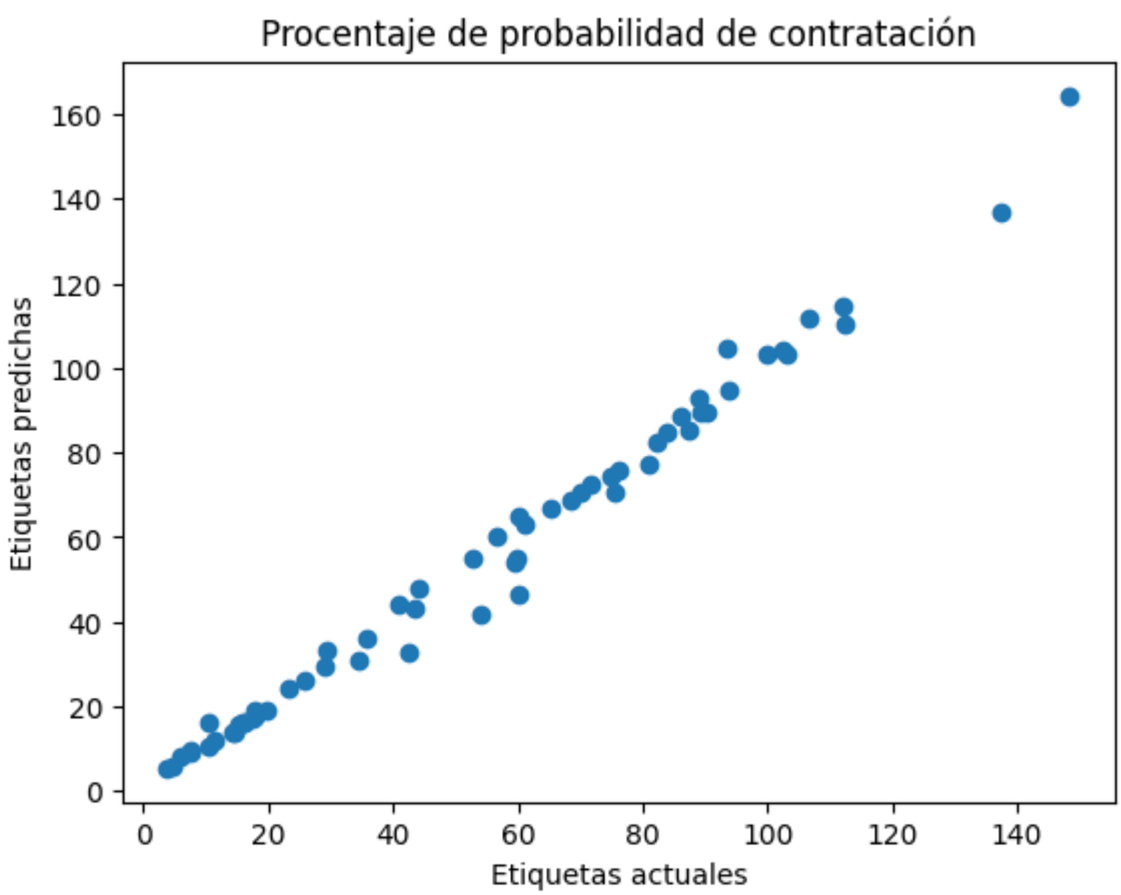
In [ ]:
```
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 19.044432887988897
RMSE: 4.363992769011985
R2: 0.9857527003270676
```

Grafico prediccticio vs grafico real

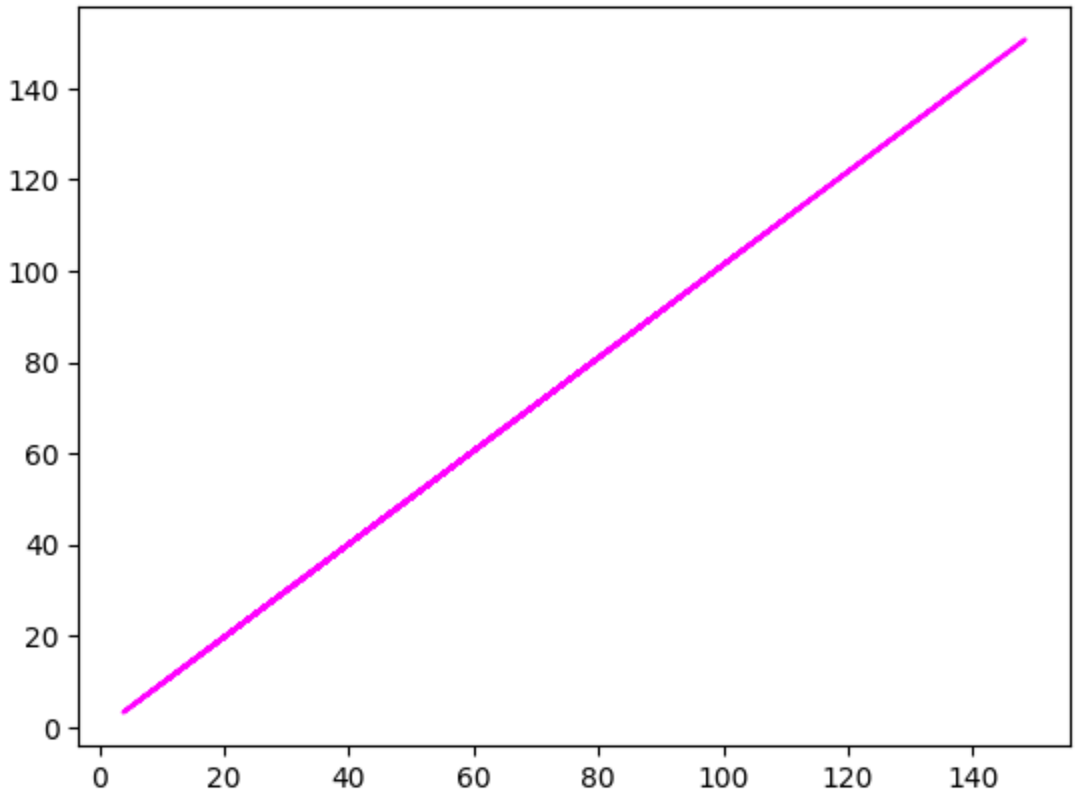
In [ ]:
```
plt.scatter(y_test, predictions)
plt.xlabel('Etiquetas actuales')
plt.ylabel('Etiquetas predichas')
plt.title('Procentaje de probabilidad de contratación')
```

Out[ ]:
```
Text(0.5, 1.0, 'Procentaje de probabilidad de contratación')
```



Linea de regreción

In [ ]:
```
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



In [ ]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, r2_score
```

Usar el algoritmo para aumento de gradiante

In [ ]:
```
alg = GradientBoostingRegressor()
```

Prueba de hiperparametros

In [ ]:
```
params = {
 'learning_rate': [0.1, 0.5, 1.0],
 'n_estimators' : [40, 90, 140]
 }
```

Encontrar la mejor combinación de hiperparametros para disminuir el valor de R2

In [ ]:
```
score = make_scorer(r2_score)
gridsearch = GridSearchCV(alg, params, scoring=score, cv=3, return_train_score=True
gridsearch.fit(X_train, y_train)
print("Best parameter combination:", gridsearch.best_params_, "\n")
```

```
Best parameter combination: {'learning_rate': 0.1, 'n_estimators': 140}
```

Conseguir el modelo

In [ ]:
```python
model=gridsearch.best_estimator_
print(model, "\n")
```

```
GradientBoostingRegressor(n_estimators=140)
```

Evaluar modelo con datos de prueba

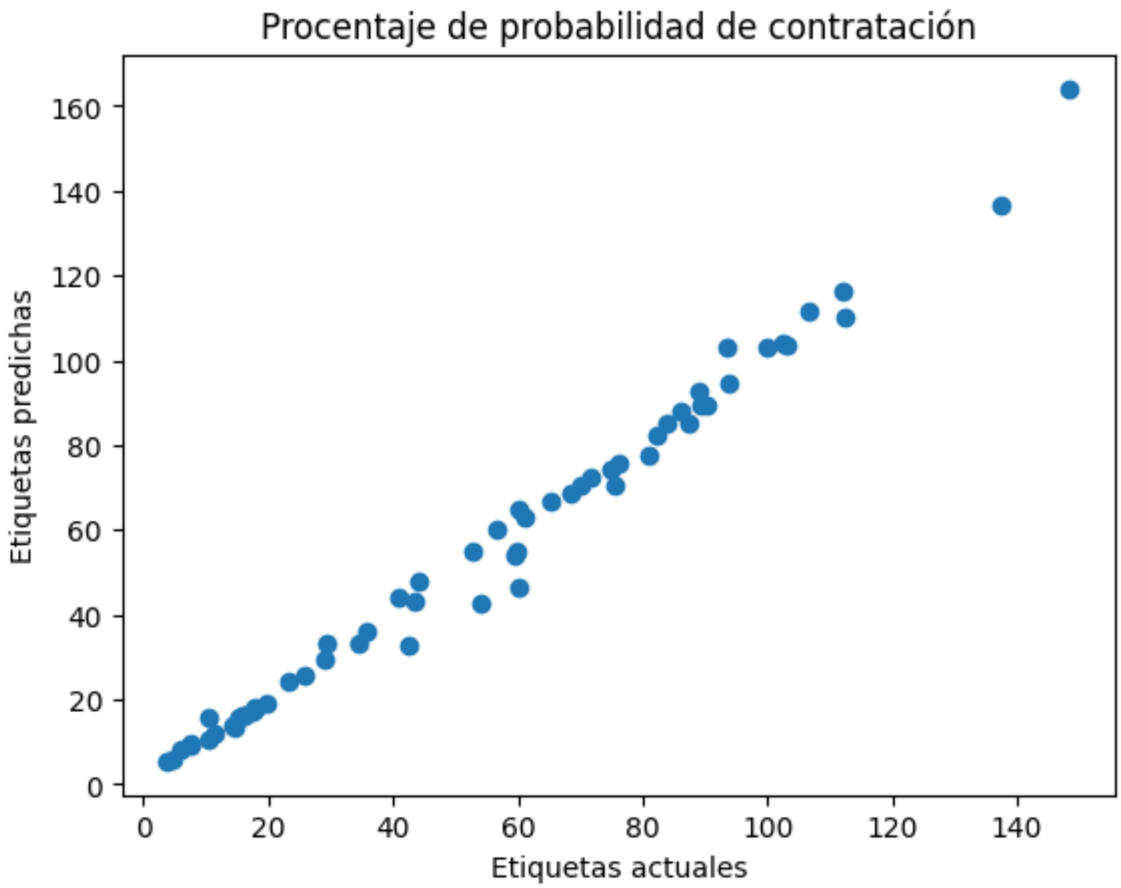
In [ ]:
```python
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 17.8869098733245
RMSE: 4.229291887931655
R2: 0.9866186529844787
```

Gráfico predicticio vs real

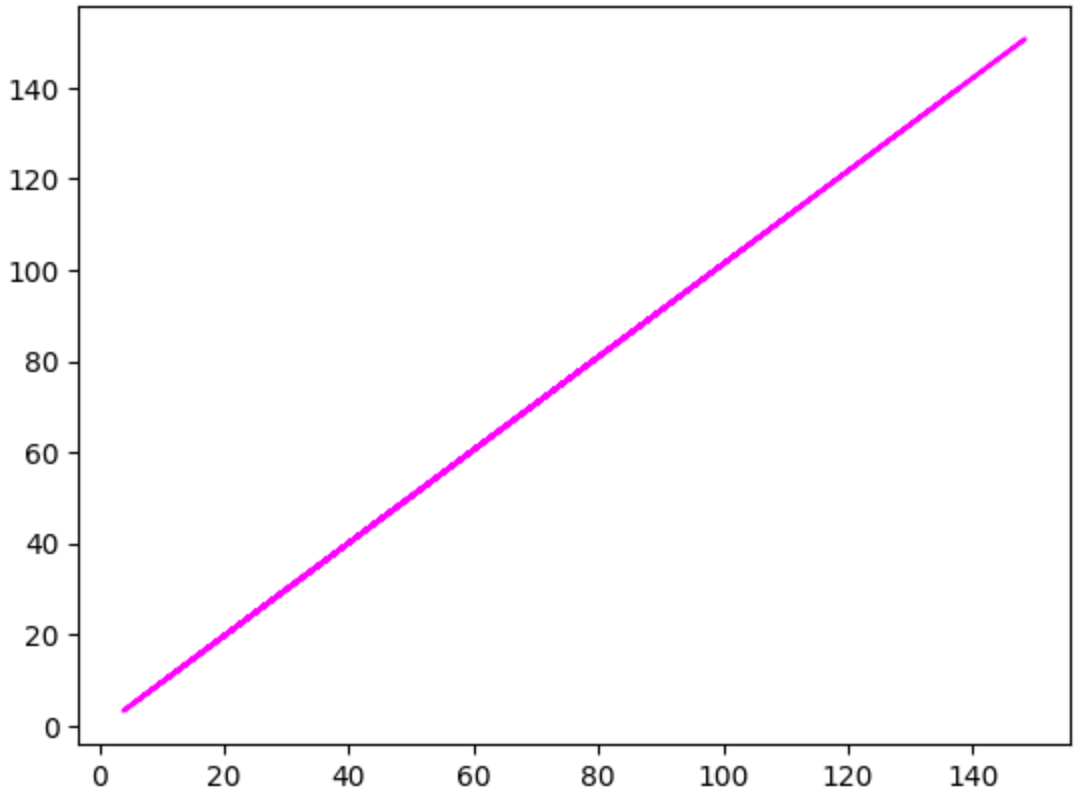
In [ ]:
```python
plt.scatter(y_test, predictions)
plt.xlabel('Etiquetas actuales')
plt.ylabel('Etiquetas predichas')
plt.title('Procentaje de probabilidad de contratación')
```

Out[ ]:
```
Text(0.5, 1.0, 'Procentaje de probabilidad de contratación')
```



Linea de regreción

In [ ]:
```
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



Entremaiento de modelo

In [ ]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LinearRegression
import numpy as np
```

Definir el preprosamiento de las columnas numericas

In [ ]:
```
numeric_features = [6,7,8,9]
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())])
```

Procesamiento de caracteristicas categoricas

In [ ]:
```
categorical_features = [0,1,2,3,4,5]
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])
```

Combinacions de procesamientos

In [ ]:
```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])
```

Canalizacion de procesamiento

In [ ]:
```
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('regressor', GradientBoostingRegressor())])
```

Ajuste de canalización

In [ ]:
```
model = pipeline.fit(X_train, (y_train))
print (model)
```

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('scaler',
                                                                   StandardScaler
())]),
                                                  [6, 7, 8, 9]),
                                                 ('cat',
                                                  Pipeline(steps=[('onehot',
                                                                   OneHotEncoder(ha
ndle_unknown='ignore'))]),
                                                  [0, 1, 2, 3, 4, 5])])),
                ('regressor', GradientBoostingRegressor())])
```

Retornar predicciones

In [ ]:
```
predictions = model.predict(X_test)
```

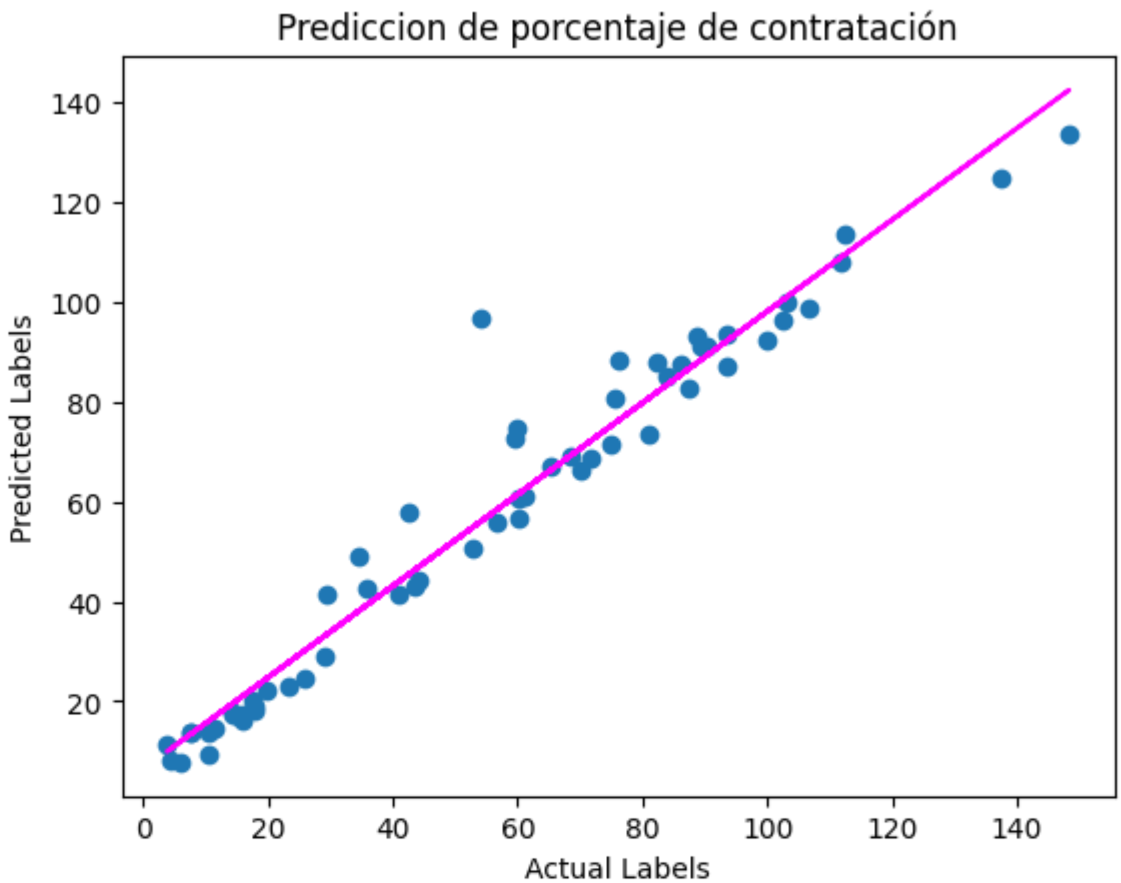
Mostrar metricas

In [ ]:
```
mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 67.21648781873382
RMSE: 8.198566205058896
R2: 0.9497147827636553
```

Grafico predicctício vs Grafico real

In [ ]:
```
plt.scatter(y_test, predictions)
plt.xlabel('Actual Labels')
plt.ylabel('Predicted Labels')
plt.title('Prediccion de porcentaje de contratación')
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



Estimador diferente para ser canalisado

In [ ]:
```
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('regressor', RandomForestRegressor())])
```

Ajuste de canalización

In [ ]:
```
model = pipeline.fit(X_train, (y_train))
print (model, "\n")
```

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('scaler',
                                                                   StandardScaler
()))]),
                                                  [6, 7, 8, 9]),
                                                 ('cat',
                                                  Pipeline(steps=[('onehot',
                                                                   OneHotEncoder(ha
ndle_unknown='ignore'))]),
                                                  [0, 1, 2, 3, 4, 5])])),
                ('regressor', RandomForestRegressor())])
```

Retornar predicciones

In [ ]:

```
predictions = model.predict(X_test)
```

Mostrar predicciones

In [ ]:

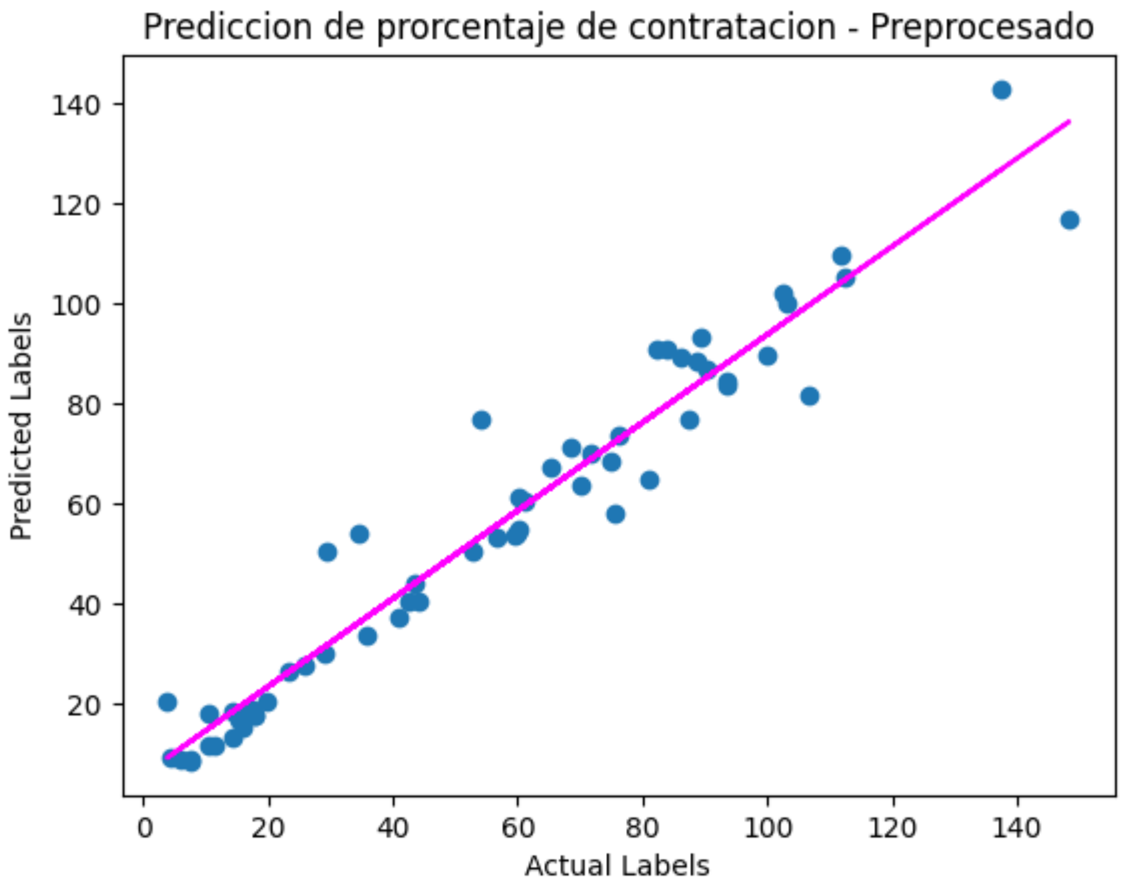
```
mse = mean_squared_error(y_test, predictions)
print("MSE:", mse)
rmse = np.sqrt(mse)
print("RMSE:", rmse)
r2 = r2_score(y_test, predictions)
print("R2:", r2)
```

```
MSE: 80.675597751584
RMSE: 8.981959571918813
R2: 0.9396459099506878
```

Grafico real vs predictivo

In [ ]:

```
plt.scatter(y_test, predictions)
plt.xlabel('Actual Labels')
plt.ylabel('Predicted Labels')
plt.title('Prediccion de proncentaje de contratacion - Preprocesado')
z = np.polyfit(y_test, predictions, 1)
p = np.poly1d(z)
plt.plot(y_test,p(y_test), color='magenta')
plt.show()
```



Obter modelo para usar en API

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from joblib import *
import numpy as np
import pandas as pd
```

Gurdar modelo

In [ ]:
```
work_prediction = pd.read_csv("works_oferts.csv")
x = work_prediction[["exp", "node", "sql", "postgsql",
                        "aws", "js", "ningles", "visap", "hofice", "estudios"]].val
y = work_prediction["probabilidad"].values
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.30, random_state=0)

model = LinearRegression().fit(x_train , y_train)
predictions = model.predict(x_test)

print(x_test[0])
print(predictions[0])


dump(model, 'model.joblib')
```

```
[ 1.   8.   1.   2.   2.   3.  16.6  1.   1.  15. ]
-9.255958893083147
```

Out[ ]:

```
['model.joblib']
```