In [45]:
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import scipy.stats as stats
import pylab
from scipy.stats import shapiro
import pingouin as pg
import numpy as np
from plotly import express as px
from plotly import graph_objects as go
from statsmodels.graphics.gofplots import qqplot
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [46]:
```
data = pd.read_excel("valecapclf.xlsx")
data
```

Out[46]:

	Vale	Cap	CLF
0	10.67	4717.2	8.97
1	10.31	4991.7	8.62
2	10.18	5219.2	8.72
3	10.48	5241.9	8.73
4	10.42	5288.7	9.15
...	...	...	...
1460	15.71	6061.0	15.58
1461	15.81	6283.0	15.42
1462	15.22	6689.0	15.29
1463	15.33	6201.0	14.75
1464	15.07	6249.0	14.33

1465 rows × 3 columns

In [47]:
```
data.shape
```

Out[47]:
```
(1465, 3)
```

In [48]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Vale    1465 non-null   float64
 1   Cap     1465 non-null   float64
 2   CLF     1465 non-null   float64
dtypes: float64(3)
memory usage: 34.5 KB
```

In [49]:
```
data.head()
```

Out[49]:

	Vale	Cap	CLF
0	10.67	4717.2	8.97
1	10.31	4991.7	8.62
2	10.18	5219.2	8.72
3	10.48	5241.9	8.73
4	10.42	5288.7	9.15

In [50]:
```
data.keys()
```

Out[50]:
```
Index(['Vale', 'Cap', 'CLF'], dtype='object')
```

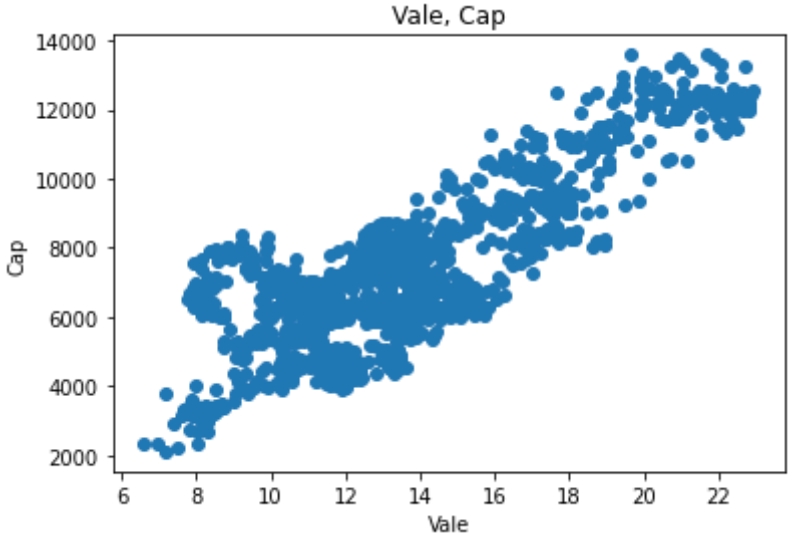
In [51]:
```
data.describe()
```

Out[51]:

	Vale	Cap	CLF
count	1465.000000	1465.000000	1465.000000
mean	13.406819	7277.196594	11.960710
std	3.359630	2165.751931	6.346423
min	6.580000	2115.650000	3.070000
25%	11.200000	5995.600000	7.280000
50%	13.000000	7040.500000	9.310000
75%	14.730000	8117.800000	16.610000
max	22.940000	13583.950000	33.070000

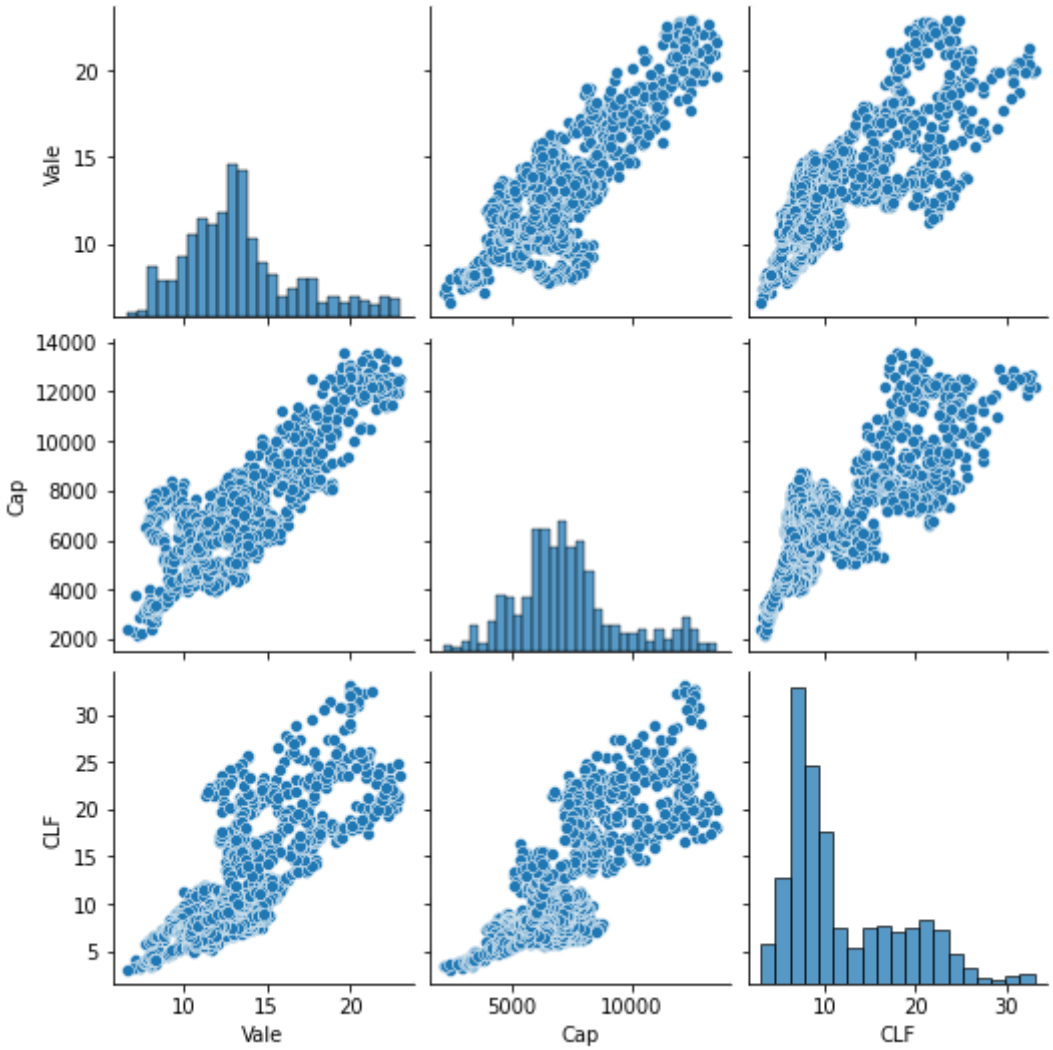
In [52]:
```
plt.scatter(data['Vale'],data ['Cap'])
plt.title ("Vale, Cap")
plt.xlabel ("Vale")
plt.ylabel ("Cap")
```

Out[52]:
```
Text(0, 0.5, 'Cap')
```



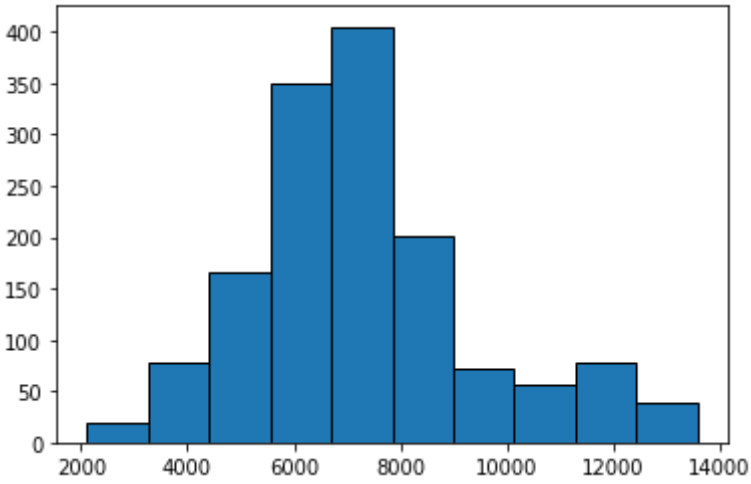
In [53]:
```
sns.pairplot(data)
```

Out[53]:
```
<seaborn.axisgrid.PairGrid at 0x12b40a170>
```



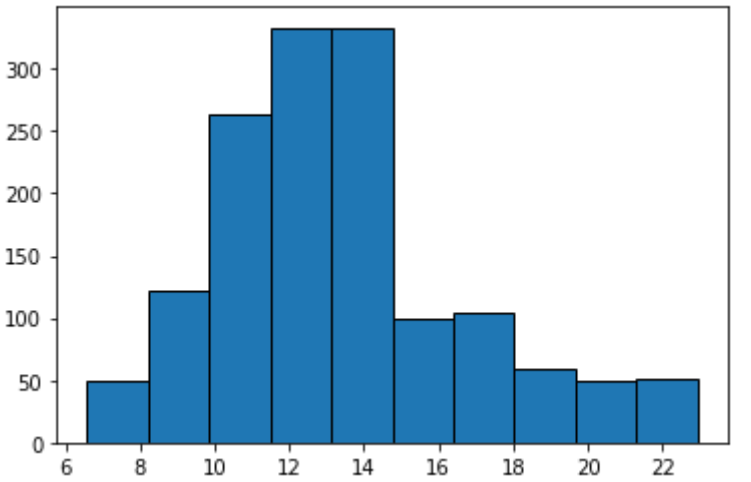
In [54]:
```
plt.hist(data['Cap'], edgecolor = 'black', linewidth=1)
```

Out[54]:
```
(array([ 20.,  78., 166., 349., 405., 202.,  71.,  57.,  78.,  39.]),
 array([ 2115.65,  3262.48,  4409.31,  5556.14,  6702.97,  7849.8 ,
         8996.63, 10143.46, 11290.29, 12437.12, 13583.95]),
 <BarContainer object of 10 artists>)
```



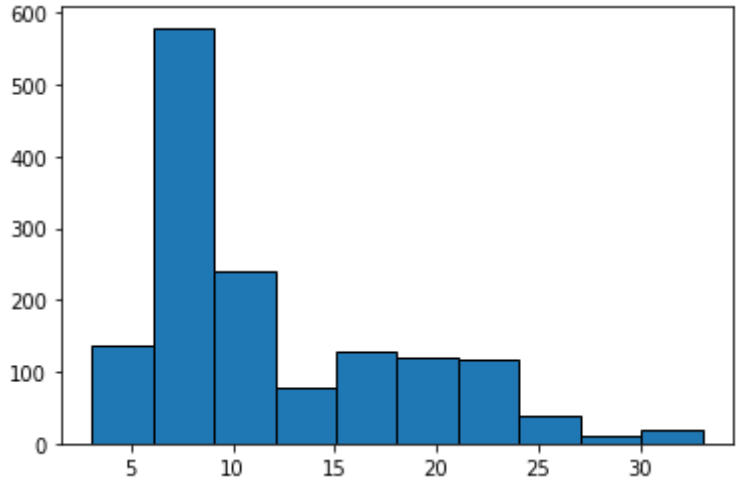
In [55]:
```
plt.hist(data['Vale'], edgecolor = 'black', linewidth=1)
```

Out[55]:
```
(array([ 50., 122., 264., 333., 332.,  99., 105.,  59.,  50.,  51.]),
 array([ 6.58 ,  8.216,  9.852, 11.488, 13.124, 14.76 , 16.396, 18.032,
        19.668, 21.304, 22.94 ]),
 <BarContainer object of 10 artists>)
```

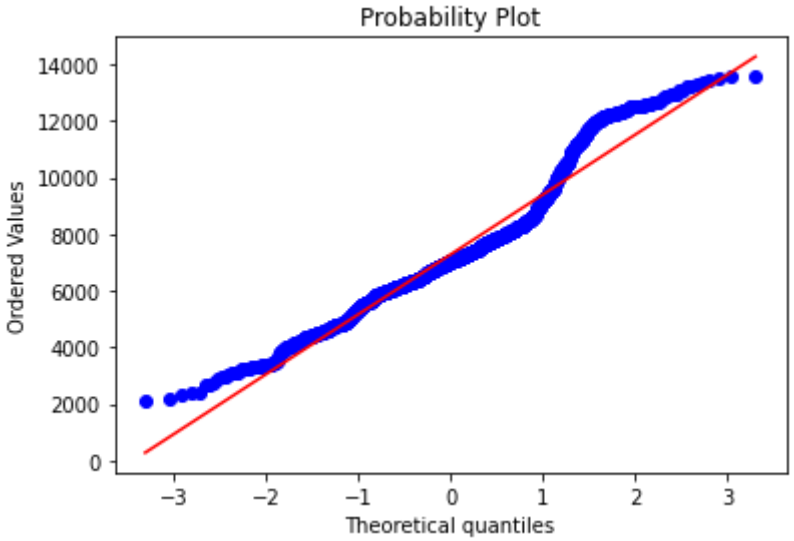


In [56]:
```
plt.hist(data['CLF'], edgecolor = 'black', linewidth=1)
```

Out[56]:
```
(array([136., 579., 241.,  79., 128., 119., 116.,  38.,  10.,  19.]),
 array([ 3.07,  6.07,  9.07, 12.07, 15.07, 18.07, 21.07, 24.07, 27.07,
        30.07, 33.07]),
 <BarContainer object of 10 artists>)
```



In [57]:
```
stats.probplot (data['Cap'], dist="norm", plot=pylab)
pylab.show()
```



In [58]:
```
estadistico, p_value = shapiro(data['Cap'])
print('Estadisticos=%.3f, p=%.3f' %  (estadistico, p_value))
```

```
Estadisticos=0.952, p=0.000
```

In [59]:
```
data_corr = data.corr(method='pearson')
data_corr
```

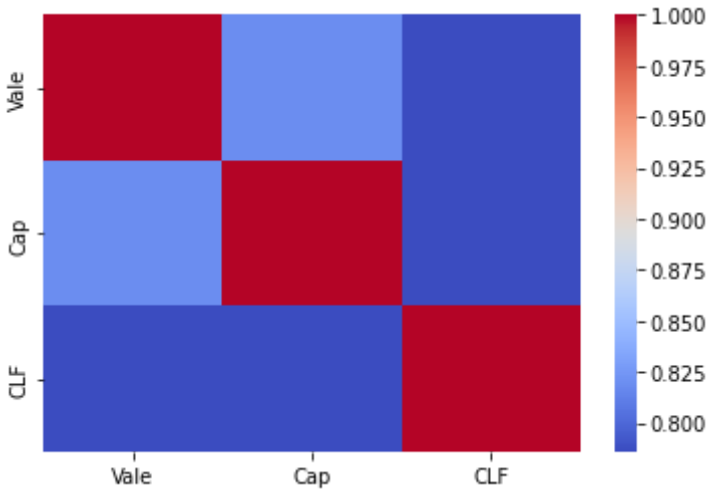
Out[59]:

	Vale	Cap	CLF
Vale	1.000000	0.819205	0.785961
Cap	0.819205	1.000000	0.786271
CLF	0.785961	0.786271	1.000000

In [60]:
```
#Mapa de calor
sns.heatmap(data_corr,
            xticklabels=data_corr.columns,
```

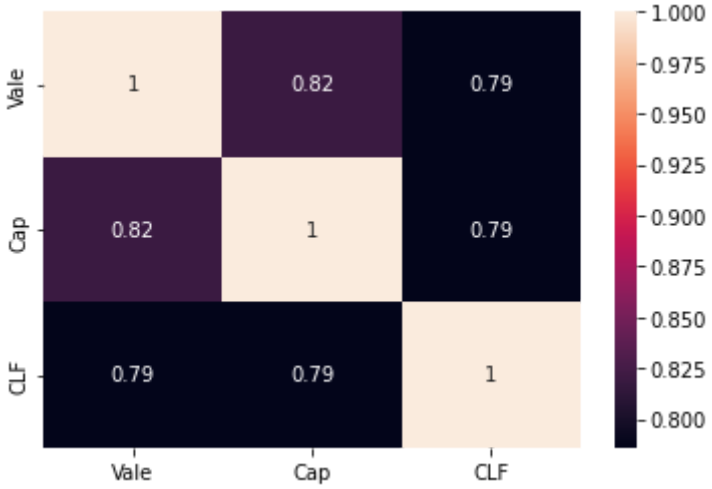
```
            yticklabels=data_corr.columns,
            cmap='coolwarm'
           )
```

Out[60]:
```
<AxesSubplot:>
```



In [61]:
```
#Matriz de correlación
sns.heatmap(data.corr(), annot=True)
plt.show
```

Out[61]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [62]:
```
corr = pg.pairwise_corr(data, method='pearson')
corr.sort_values(by=['p-unc'])[['X', 'Y', 'n', 'r', 'p-unc']]
```

Out[62]:

	X	Y	n	r	p-unc
0	Vale	Cap	1465	0.819205	0.000000e+00
2	Cap	CLF	1465	0.786271	3.295233e-308
1	Vale	CLF	1465	0.785961	8.380432e-308

In [63]:
```
from sklearn.preprocessing import LabelEncoder
```

```
encoder = LabelEncoder()
```

In [64]:
```
X = data.drop(['Cap'], axis=1)
y = data['Cap']

X.shape, y.shape
```

Out[64]:
```
((1465, 2), (1465,))
```

In [65]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
```

In [66]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
```

In [67]:
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 1000, random_state = 42)
regressor.fit(X_train, y_train)
```

Out[67]:
```
▾ RandomForestRegressor

RandomForestRegressor(n_estimators=1000, random_state=42)
```

In [68]:
```
#Modelación regresión random forest
```

In [69]:
```
model_rf = RandomForestRegressor(random_state=42)
model_rf.fit(X_train, y_train)
#predicción modelo
pred_rf = model_rf.predict(X_test)
print(pred_rf)
print(pred_rf.shape)
```

```
[ 9969.0948  6692.0409  5258.2797  5210.3877  7509.468   6817.6746
  5316.2867  3045.9156  9733.9527  5307.1403  5925.403   6966.7613
  3373.9378  4628.9179  6210.5857  6349.0868  6557.5457  9403.4336
  7150.2247 12208.8218  8224.6323  3150.9918  7542.9932  6808.4433
  6516.5268  9371.074  12589.4917  8006.036   7138.5282  6354.3642
  7334.5408  7269.2096 10421.7528 12030.908   7606.3778  6464.2096
  4304.9835  7611.9821  7318.931   6390.354   6497.5092  6728.442
  7630.67    8848.6071  6360.831   5532.6835  6955.7507 12161.1327
  7201.146   5526.105   5272.547   6819.7136  5973.1368  6628.527
  7917.9727  6434.0495  6216.5135  5847.9393  6589.3637  5680.8589
  7742.13    6365.6132  7389.0319  6773.7692  6280.1084  7788.3666
  6260.6378 12257.4642  7064.5695  5929.2544  6362.7742  5448.1669
  5243.0398  7718.0362 12077.4077  7021.6794  7222.9212 12466.9876
  4825.0948  7980.8643  9110.593   8274.2034  5651.5614  7299.1863
  7002.7307 10011.2616  7234.7102 10412.6309  8558.5701  6784.636
 10598.8317  7128.111   5074.7166  4468.0529  5238.989   9749.0105
  6721.4604  7412.4088  6603.2337  8096.0648  6258.7155  6587.914
  6490.3478  6562.658   7283.5239  9651.5665  6242.9906  6386.7768
  8007.398   4337.0743  6470.5078  7101.028   5886.472   8092.019
  7607.786   5944.6607 10468.4128  6649.6676  6145.2789  6096.6108
  7165.8921  9544.2031  5919.5103  7102.3341  6803.4179 11816.1954
  7216.1294  8978.343   6964.6858  6204.7809  6032.7424  6306.5958
  6232.8798  6361.8507  5919.982   7706.166   9164.3672 12662.1415
  6887.2191  6272.618   7607.33    8581.2388  3355.9196  7632.3524
  6992.437   7452.125   6628.6497  7769.0052  5186.82   10842.69
  7650.4776 12242.3721  6648.3107  3138.447   7368.153   7142.3112
  9242.1263  7130.001   4805.0298  4262.528   2649.2673  7071.0867
  7655.14    4045.2731 12038.1181  7093.2744  7037.412   7155.9552
 12281.1973 10149.3645  5564.8732  9833.6343  6326.009   5930.9767
  7026.5373  6723.9234  9788.8128  7618.299   6196.5235  7334.0929
  6183.6935  5959.4964  6870.7135  7323.162   8767.8591  6331.1802
  3076.7918 12242.2571  8012.4996  6969.1615  6292.1819  7948.7914
  5335.9136  7364.4628  8680.0848  5837.0058 11440.4901  6356.2993
  7921.6283  7435.6837 12597.5455  7451.6585  6675.1195  6943.7541
  6725.3022  6592.9208  8895.212   6951.95    6421.5712  7854.557
  9954.3658 12153.2403  6512.0693  9395.9951  7091.582   3025.0582
  2415.5303  9401.3342  7143.3832  6841.3083  6128.5469 12458.1255
 12281.1604  6407.54    6770.7517  7636.05    4890.7857  7349.939
  6464.4877  7323.942   6664.6133  7855.3719  6482.1768  6738.1068
  4726.3794  7161.183   8898.712   6139.3739  4593.0349  7108.0757
  5603.7082  6884.0516  6332.873   5971.5588 12107.5884  4618.1803
  6290.4266 10591.9617  7385.545  12018.3282  6888.218   6494.7806
  3411.8447 10136.653  11093.51    7154.3766  6781.0286 10064.0494
 12433.6213  6953.3949  6493.468   5968.7248  7112.7663  3862.5798
  7584.047   6388.943   4797.2175  6827.1062  6468.7955  7905.08
  7557.8448  7103.285   7382.271   7057.0204  5746.5956  7046.2252
  8726.7525  7472.79   12038.8012  5493.965   6602.6758  3051.4021
  6931.8436  7107.5758  6688.7828  7668.47    7327.3017  6356.6522
  7816.2832  6783.313   7593.2992  7604.832   6598.7753  7134.0809
 10122.2829 10565.9605 12212.652   6617.402   6182.6857  8623.167
  6603.754   3427.4045  6902.377   6326.0868  7341.243   8183.718
  6223.05    7065.6284  8788.4229  4469.711   5362.5268  6999.662
  7611.4874  7090.4666  7265.5122  6922.9601  5956.8399  7382.5271
  6901.5683  7483.0577  7518.1452  9432.4489  9318.8472  6318.4413
  6228.1724  5733.8051  5829.9725 12145.0909  7075.1127  9252.7121
  8580.7901  6591.5794  6811.9451  6885.1198 10083.3988  6153.4227
  8354.519   5113.0151  6506.9779  5461.0926  8971.4704  7315.1038
  4823.3898  5680.8588  7100.2281 10010.6714  5271.2392  8636.807
  6151.5563 12335.936   3464.665   6279.7159  6747.4033  7642.88
 12232.0792  6239.7915  9713.4358  7482.6034 10289.05    6735.5866
```

```
  7310.6861  6901.267   6941.7598  7211.3815  3567.9438  5344.6066
  7225.0098  5857.7995  9605.8911  6807.202   6309.0467 11623.77
  6226.1763 12261.5942  8227.75    7489.84    5717.7214 12252.2948
  6794.3588  6221.3563  5093.4325  5908.6275  5903.7598  8197.0064
  5637.6394  7495.868   6225.9764  6790.7269 10834.3853  8866.971
  7569.5169  8192.26    7769.33    6314.6657  6109.7365  8949.0909
  9807.8898  6044.7037  7079.1448  6750.177  12482.3042 10639.3202
  7360.101   7644.24    9357.0191  6353.4045  7463.739   7738.2752
  6595.1646  5062.3752  4956.1465  7930.665   7365.638   5998.6055
  7375.7836  5672.3972  7350.429   7572.1425  9758.7222  3014.1972
  7543.7325  3023.939   6891.8687  9597.974   6715.2872  5995.0577
 12301.4135  3365.235  12022.189   7146.5664  6530.6958  7066.194
  9492.5244  5874.8821  6555.676  12021.8626  5747.1132  6794.0239
  8260.8524  4511.1908]
(440,)
```

In [70]:

```
y_pred = model_rf.predict(X_test)
```

In [71]:

```
#Comparación datos reales con proyectados
data=pd.DataFrame({'Cap':y_test, 'Predicción':y_pred})
data
```

Out[71]:

	Cap	Predicción
994	8124.89	9969.0948
438	7026.90	6692.0409
746	4529.72	5258.2797
655	5839.00	5210.3877
31	7920.20	7509.4680
...	...	...
1130	12375.82	12021.8626
712	4550.02	5747.1132
10	5676.90	6794.0239
1031	9165.51	8260.8524
692	5033.02	4511.1908

440 rows × 2 columns

In [72]:

```
from sklearn import metrics
errors = abs(y_pred - y_test)
print('Testing R2 Score: ', r2_score(y_test, pred_rf)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y
mape = 100 * (errors / y_test)
accuracy = 100 - np.mean(mape)
print('Certeza:', round(accuracy, 2), '%.')
```

```
Testing R2 Score:  83.95820998844307
Mean Absolute Error: 612.8856674999996
Mean Squared Error: 706370.1369337585
Root Mean Squared Error: 840.4582898239261
Certeza: 90.91 %.
```

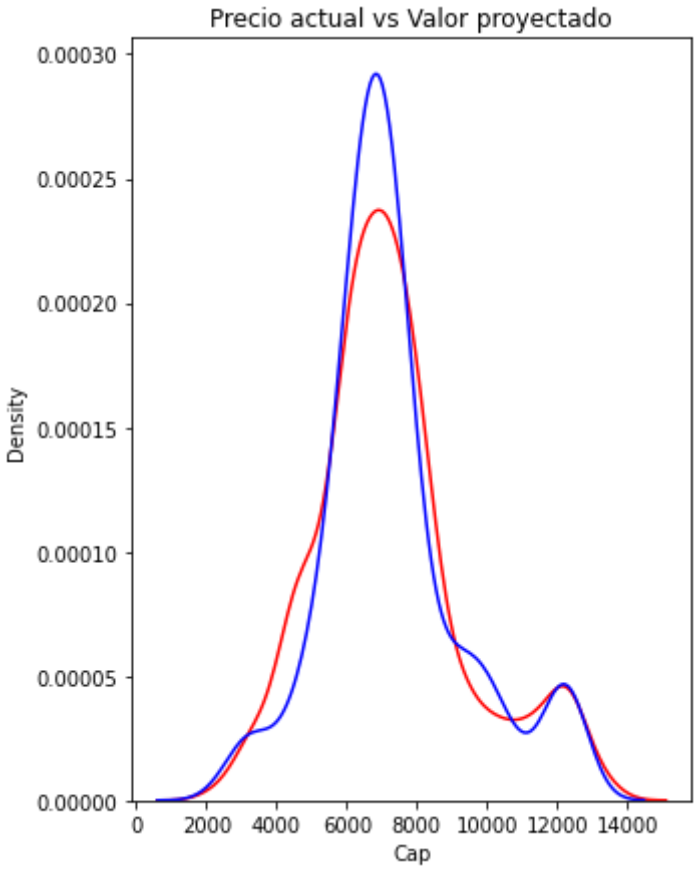
In [73]:

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


ax = sns.distplot(y, hist=False, color="r", label="Precio actual")
sns.distplot(y_pred, hist=False, color="b", label="Valor proyectado" , ax=ax)


plt.title('Precio actual vs Valor proyectado')


plt.show()
plt.close()
```



In [ ]:

```
import pydot
# Pull out one tree from the forest
Tree = regressor.estimators_[5]
# Export the image to a dot file
from sklearn import tree
plt.figure(figsize=(700,50))
tree.plot_tree(Tree,filled=True,
               rounded=True,
               fontsize=10);
```

In [76]:
```
#Proyección precio Cap, con valores de Vale y CLF17-11-2022 y 18-11-2022
prueba= pd.DataFrame({"Vale":[15.33, 15.07], "CLF":[14.75, 14.33]})
prueba
```

Out[76]:

	Vale	CLF
0	15.33	14.75
1	15.07	14.33

In [77]:
```
#Resultados
#Precio reales de Cap para las fechas proyectadas: 6201, 6310
print(model_rf.predict(prueba))
```

```
[6220.2156 6260.6378]
```

In [ ]:
```

```