In [223…

```
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 [224…

```
data = pd.read_excel("clp.xlsx")
data
```

Out[224]:

	USDCLP	DXY	Cobre
0	592.90	90.36	3.3660
1	594.00	90.61	3.3530
2	594.70	90.32	3.3595
3	595.10	89.94	3.3305
4	591.70	90.08	3.3255
...	...	...	...
1228	890.46	106.66	3.8335
1229	890.70	106.40	3.8200
1230	886.50	106.28	3.7735
1231	911.75	106.67	3.6880
1232	921.20	106.97	3.6320

1233 rows × 3 columns

In [151…

```
data.shape
```

Out[151]:

```
(1236, 3)
```

In [154…

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1233 entries, 0 to 1232
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   USDCLP  1233 non-null   float64
 1   DXY     1233 non-null   float64
 2   Cobre   1233 non-null   float64
dtypes: float64(3)
memory usage: 29.0 KB
```

In [202…

```
data.head()
```

Out[202]:

	USDCLP	DXY	Cobre
0	592.9	90.36	3.3660
1	594.0	90.61	3.3530
2	594.7	90.32	3.3595
3	595.1	89.94	3.3305
4	591.7	90.08	3.3255

In [203…

```
data.keys()
```

Out[203]:

```
Index(['USDCLP', 'DXY', 'Cobre'], dtype='object')
```

In [204…

```
data.describe()
```

Out[204]:

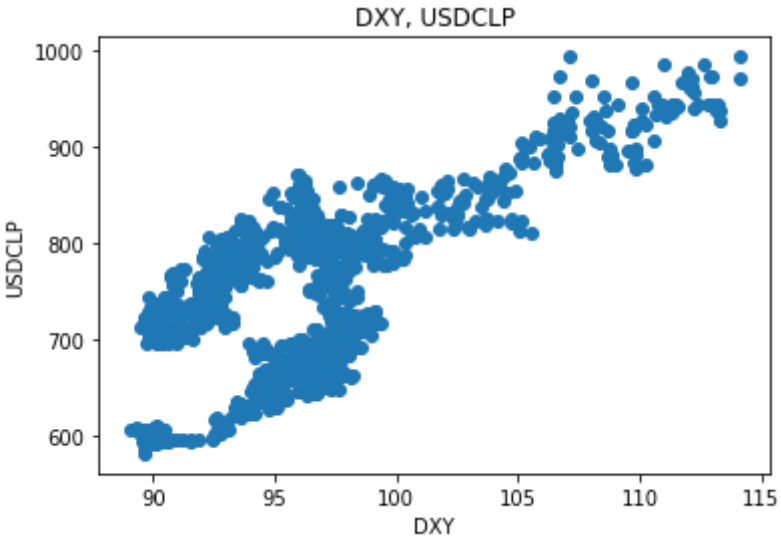
	USDCLP	DXY	Cobre
count	1233.000000	1233.000000	1233.000000
mean	753.501769	96.662685	3.369645
std	84.731032	4.943758	0.678179
min	581.100000	89.030000	2.175000
25%	685.850000	93.190000	2.800000
50%	751.000000	96.360000	3.186500
75%	808.100000	98.160000	4.040500
max	993.510000	114.110000	4.865500

In [205…

```
plt.scatter(data['DXY'],data ['USDCLP'])
plt.title ("DXY, USDCLP")
plt.xlabel ("DXY")
plt.ylabel ("USDCLP")
```

Out[205]:

```
Text(0, 0.5, 'USDCLP')
```

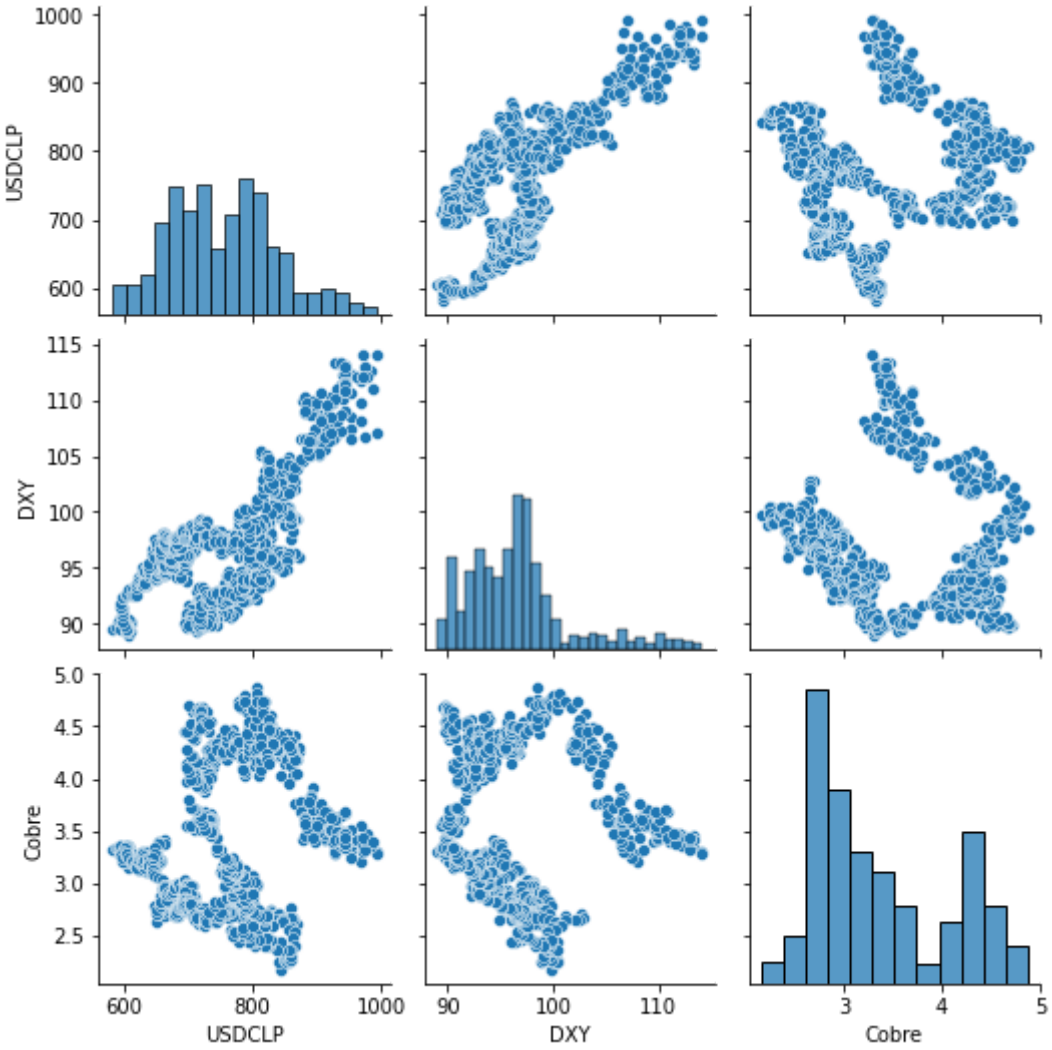


In [159…

```
sns.pairplot(data)
```

Out[159]:

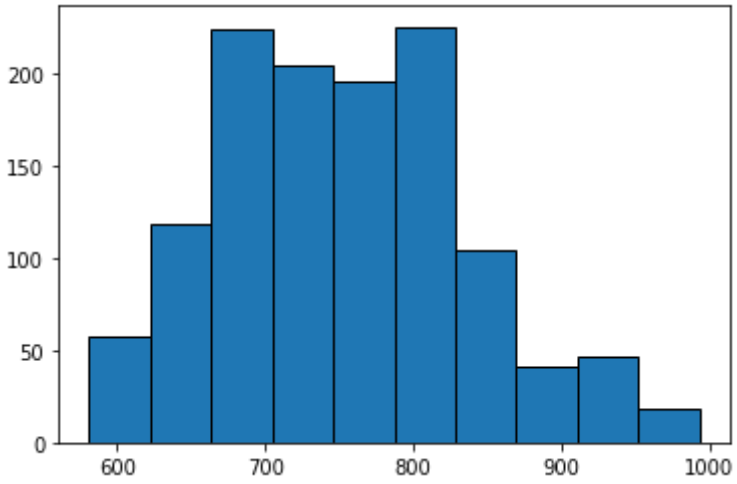
```
<seaborn.axisgrid.PairGrid at 0x12bacb4c0>
```



In [161…

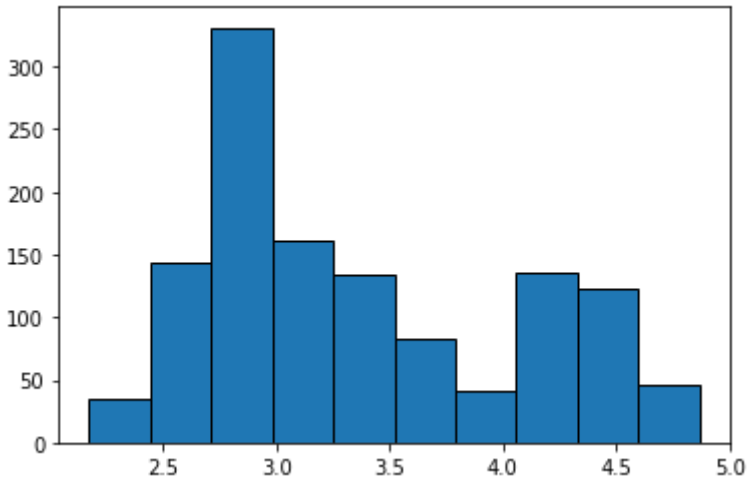
```
plt.hist(data['USDCLP'], edgecolor = 'black', linewidth=1)
```

Out[161]:
```
(array([ 57., 118., 224., 204., 195., 225., 104.,  41.,  47.,  18.]),
 array([581.1  , 622.341, 663.582, 704.823, 746.064, 787.305, 828.546,
        869.787, 911.028, 952.269, 993.51 ]),
 <BarContainer object of 10 artists>)
```



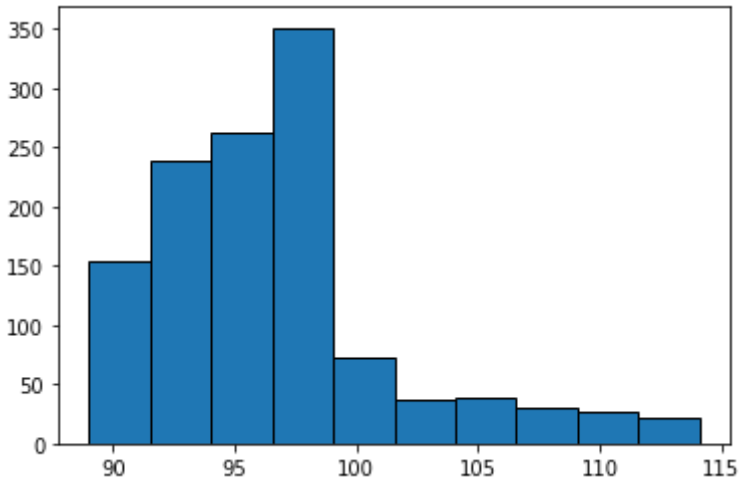
In [163…
```
plt.hist(data['Cobre'], edgecolor = 'black', linewidth=1)
```

Out[163]:
```
(array([ 35., 143., 331., 161., 134.,  83.,  42., 136., 122.,  46.]),
 array([2.175  , 2.44405, 2.7131 , 2.98215, 3.2512 , 3.52025, 3.7893 ,
        4.05835, 4.3274 , 4.59645, 4.8655 ]),
 <BarContainer object of 10 artists>)
```



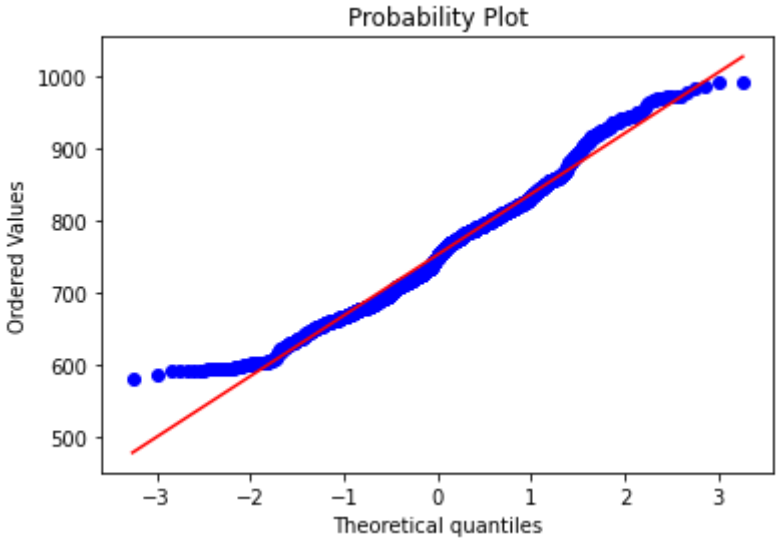
In [164…
```
plt.hist(data['DXY'], edgecolor = 'black', linewidth=1)
```

Out[164]:
```
(array([154., 239., 262., 351.,  73.,  37.,  38.,  31.,  27.,  21.]),
 array([ 89.03 ,  91.538,  94.046,  96.554,  99.062, 101.57 , 104.078,
        106.586, 109.094, 111.602, 114.11 ]),
 <BarContainer object of 10 artists>)
```



In [165…

```
stats.probplot (data['USDCLP'], dist="norm", plot=pylab)
pylab.show()
```



In [167…

```
estadistico, p_value = shapiro(data['USDCLP'])
print('Estadisticos=%.3f, p=%.3f' %  (estadistico, p_value))
```

```
Estadisticos=0.982, p=0.000
```

In [168…

```
data_corr = data.corr(method='pearson')
data_corr
```

Out[168]:

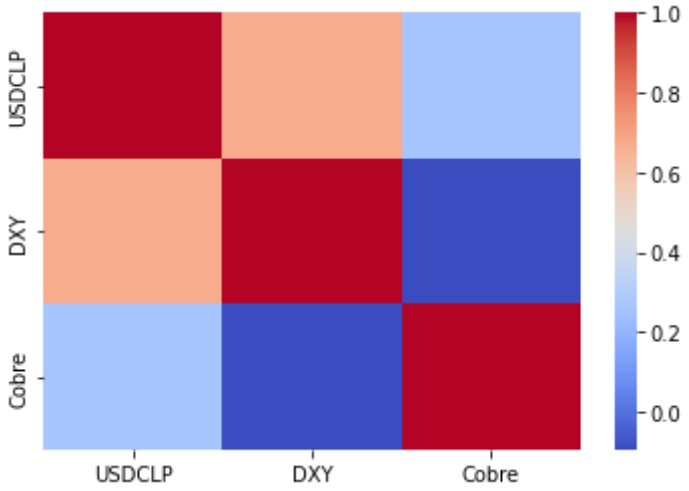
	USDCLP	DXY	Cobre
USDCLP	1.000000	0.674744	0.259269
DXY	0.674744	1.000000	-0.093952
Cobre	0.259269	-0.093952	1.000000

In [206…

```
#Mapa de calor
sns.heatmap(data_corr,
            xticklabels=data_corr.columns,
```

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

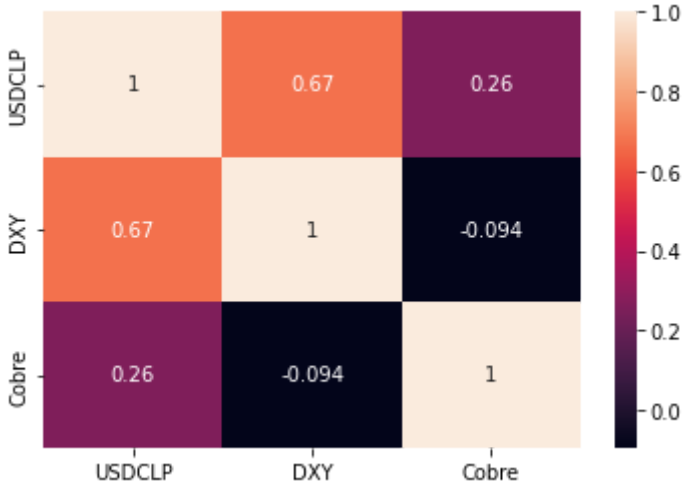
Out[206]: <AxesSubplot:>



In [207…

```
#Matriz de correlación
sns.heatmap(data.corr(), annot=True)
plt.show
```

Out[207]: <function matplotlib.pyplot.show(close=None, block=None)>



In [208…

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

Out[208]:

	X	Y	n	r	p-unc
0	USDCLP	DXY	1233	0.674744	1.387236e-164
1	USDCLP	Cobre	1233	0.259269	2.173979e-20
2	DXY	Cobre	1233	-0.093952	9.565777e-04

In [209…

```
from sklearn.preprocessing import LabelEncoder
```

```
encoder = LabelEncoder()
```

In [210…

```
X = data.drop(['USDCLP'], axis=1)
y = data['USDCLP']

X.shape, y.shape
```

Out[210]:

```
((1233, 2), (1233,))
```

In [225…

```
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 [226…

```
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 [227…

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 1000, random_state = 42)
regressor.fit(X_train, y_train)
```

Out[227]:

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

In [228…

```
#Modelación regresión random forest
```

In [229…

```
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)
```

```
[606.379    674.2557   718.286989 705.197    831.878992 751.313
 728.2511   658.1213   704.2475   778.048    600.7618   795.6652
 672.6026   706.141    717.856    641.726    775.602    703.81
 950.9612   790.1773   735.989474 707.005    724.315309 728.190121
 784.165    860.9574   741.098157 723.699    807.126371 952.5552
 810.404163 855.885    837.90724  827.207336 856.7313   761.4516
 821.081672 718.217    773.91707  848.281    945.4174   750.7842
 675.8298   837.1124   668.988    844.872    748.6961   833.6625
 739.971    777.183    783.161882 710.4141   820.286    814.3604
 722.504    939.2639   796.4744   716.989    767.043583 706.7372
 807.506808 689.745    794.2128   947.8127   603.391    805.7587
 832.842    805.421756 780.467    718.5824   741.723844 645.402
 836.864    684.1119   892.6234   671.953    741.902003 736.056454
 798.578    690.4481   782.762    790.02     671.738    705.308
 694.654    823.683292 720.444    679.3243   671.1383   684.647895
 683.197498 778.969    790.3422   666.2596   801.1939   657.88
 799.6442   684.8571   686.9234   692.21163  629.296    956.1589
 957.9721   703.236    733.99127  729.818316 886.565    858.1491
 735.138257 848.4701   700.2352   962.146    950.4625   709.424769
 887.2785   777.5537   599.612    779.041    926.504    714.669292
 819.883058 763.0998   684.3926   676.536    698.219    729.136
 724.378    763.287    739.198291 718.403    684.4505   843.5386
 823.232864 747.955549 714.219441 953.6756   765.458613 685.02
 731.264    892.3147   804.2722   805.0956   717.570576 952.716
 762.0433   715.159    839.615    683.4579   787.908543 779.184
 679.0673   739.98967  752.6373   716.506    668.969    790.127
 931.7913   682.8493   707.026712 719.8507   600.9092   718.914
 692.5129   754.9136   766.935    720.207    697.365994 709.781
 779.077    804.131881 718.5913   790.7172   957.5155   775.348
 613.853    800.950428 700.609856 691.9628   730.2065   771.281773
 710.181598 805.498799 683.876    739.264057 837.2615   613.347
 682.9935   686.4893   827.6442   801.1939   810.18354  838.9522
 735.603753 923.9279   756.038    937.0451   914.5076   768.013
 722.299104 739.037257 906.1731   835.291    728.091    748.222612
 806.192688 718.1      718.852    685.0067   863.9975   807.0749
 637.907    782.7033   821.157408 601.087    723.22     665.316
 679.0059   747.108048 753.1654   735.115181 599.5828   637.
 703.6887   680.4064   689.2018   724.880679 801.4696   726.227948
 830.9463   721.598    803.408856 777.546    722.828059 849.235
 860.7366   805.083156 853.2199   726.1249   765.795806 684.2965
 947.5063   792.905    717.093    701.8003   789.069    769.9714
 712.637527 768.9671   756.0101   831.87534  680.3011   740.386568
 653.181    840.642    716.992    690.661865 759.2062   749.308064
 633.0506   801.801157 738.872912 722.175    814.404348 677.1767
 815.1406   640.928    702.416963 655.027    798.401595 847.794
 719.366    854.6077   754.5494   944.4756   645.298    773.390363
 729.782    710.4995   714.468    964.6649   960.2518   706.047
 724.8405   705.9001   747.080199 701.315    686.9469   791.4407
 678.717    777.502    764.95581  657.5921   807.0083   751.881
 816.4113   770.0963   675.5079   719.326    722.175292 690.6332
 824.8909   712.423    685.542    636.749    688.3936   709.505
 861.1439   673.5448   671.538    678.1655   965.3069   726.833
 774.371    771.019    781.595    854.5566   690.038202 763.519547
 637.       704.287265 636.007    703.589    823.495624 799.132169
 782.328357 698.9596   643.348    914.9654   847.338    928.0423
 775.120236 956.8627   641.194    779.67307  674.3941   730.088541
 721.231234 927.6242   609.692    716.985    722.8365   712.227
 680.9153   684.697    922.2701   854.0111   717.639    787.818
 740.737508 896.3      678.5312   658.408    841.8555   807.6911
 733.984632 777.754932 725.054    782.685214 849.693    791.881
```

```
 798.4446   681.8095   739.445565 644.914    725.351053 759.6922
 732.4721   677.7324   795.832771 674.429    ]
(370,)
```

In [215…]
```
y_pred = model_rf.predict(X_test)
```

In [216…]
```
#Comparación datos reales con proyectados
data=pd.DataFrame({'USDCLP':y_test, 'Predicción':y_pred})
data
```

Out[216]:

	USDCLP	Predicción
18	600.9000	606.379000
342	695.6700	674.255700
467	787.4713	718.286989
852	729.5000	705.197000
980	811.7100	831.878992
...	...	...
880	735.8800	759.692200
795	736.0000	732.472100
272	657.3000	677.732400
1068	788.6100	795.832771
165	675.5000	674.429000

370 rows × 2 columns

In [231…]
```
from sklearn import metrics
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), '%.')
```

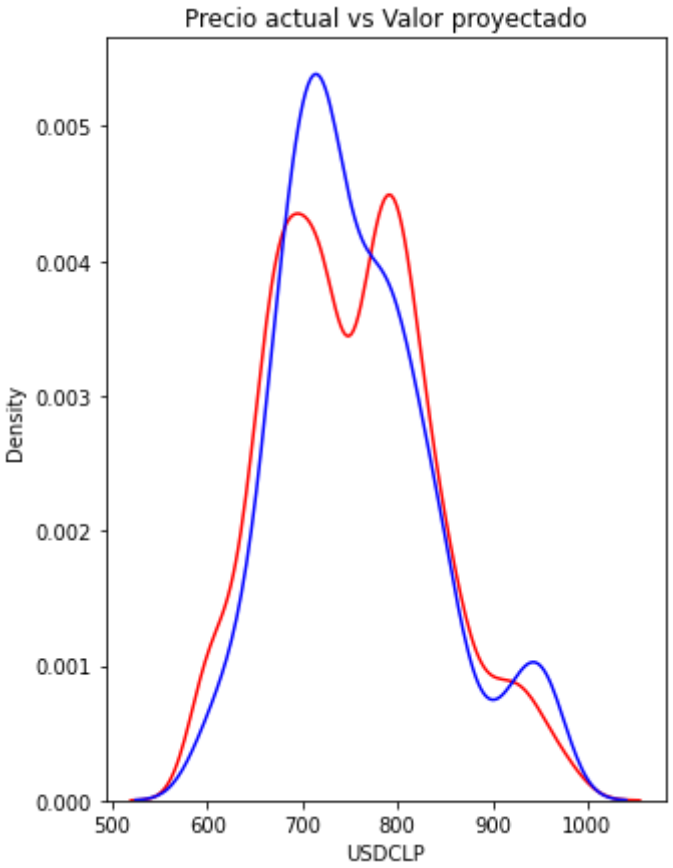
```
Testing R2 Score:  82.06514705390137
Mean Absolute Error: 23.477404670270243
Mean Squared Error: 1334.0632360841435
Root Mean Squared Error: 36.524830404591114
Certeza: 96.87 %.
```

In [219…]
```
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 [232…

```
#Proyección precio USDCLP, con valores de Cobre y DXY 17-11-2022 y 18-11-2022
prueba= pd.DataFrame({"Cobre":[3.688, 3.632], "DXY":[106.67, 106.97]})
prueba
```

Out[232]:

	Cobre	DXY
0	3.688	106.67
1	3.632	106.97

In [234…

```
#Resultados
#Precio reales de USDCLP para las fechas proyectadas: 911.75, 921.2
print(model_rf.predict(prueba))
```

```
[708.395 708.395]
```

In [ ]: