```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


##carga de dataset limpiando las columnas
df = pd.read_csv('/content/mediciones.csv',sep=';',
                 header=None,
                 names=['indice','id_medidor','fecha','consumo','consumo_reactivo'],
                 )
##elimiar primera fila
df = df.drop([0],axis=0)
df = df.drop(['indice'],axis=1)

df.head(5)
```

```
/usr/local/lib/python3.8/dist-packages/IPython/core/interactiveshell.py:3326: Dty
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	id_medidor	fecha	consumo	consumo_reactivo
1	439950	2020-02-01 00:15:00	466.35223	21.632
2	439950	2020-02-01 00:30:00	463.45178	20.944
3	439950	2020-02-01 00:45:00	461.77527	20.83
4	439950	2020-02-01 01:00:00	462.09753	21.186
5	439950	2020-02-01 01:15:00	460.12772	21.273

```
df.shape
```

```
(1585035, 4)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1585035 entries, 1 to 1585035
Data columns (total 4 columns):
 #   Column            Non-Null Count    Dtype 
---  ------            --------------    ----- 
 0   id_medidor        1585035 non-null  object
 1   fecha             1585035 non-null  object
 2   consumo           1585035 non-null  object
 3   consumo_reactivo  1574389 non-null  object
dtypes: object(4)
memory usage: 60.5+ MB
```

```
df.isnull().sum()
```

```
id_medidor              0
fecha                   0
consumo                 0
consumo_reactivo    10646
dtype: int64
```

```
##cambiar tipo de dato
df = df.astype({'id_medidor' : 'str',
                'consumo' : 'float32',
                'consumo_reactivo' : 'float32'})

df['fecha'] = pd.to_datetime(df['fecha'])



df = df.fillna(0)
```

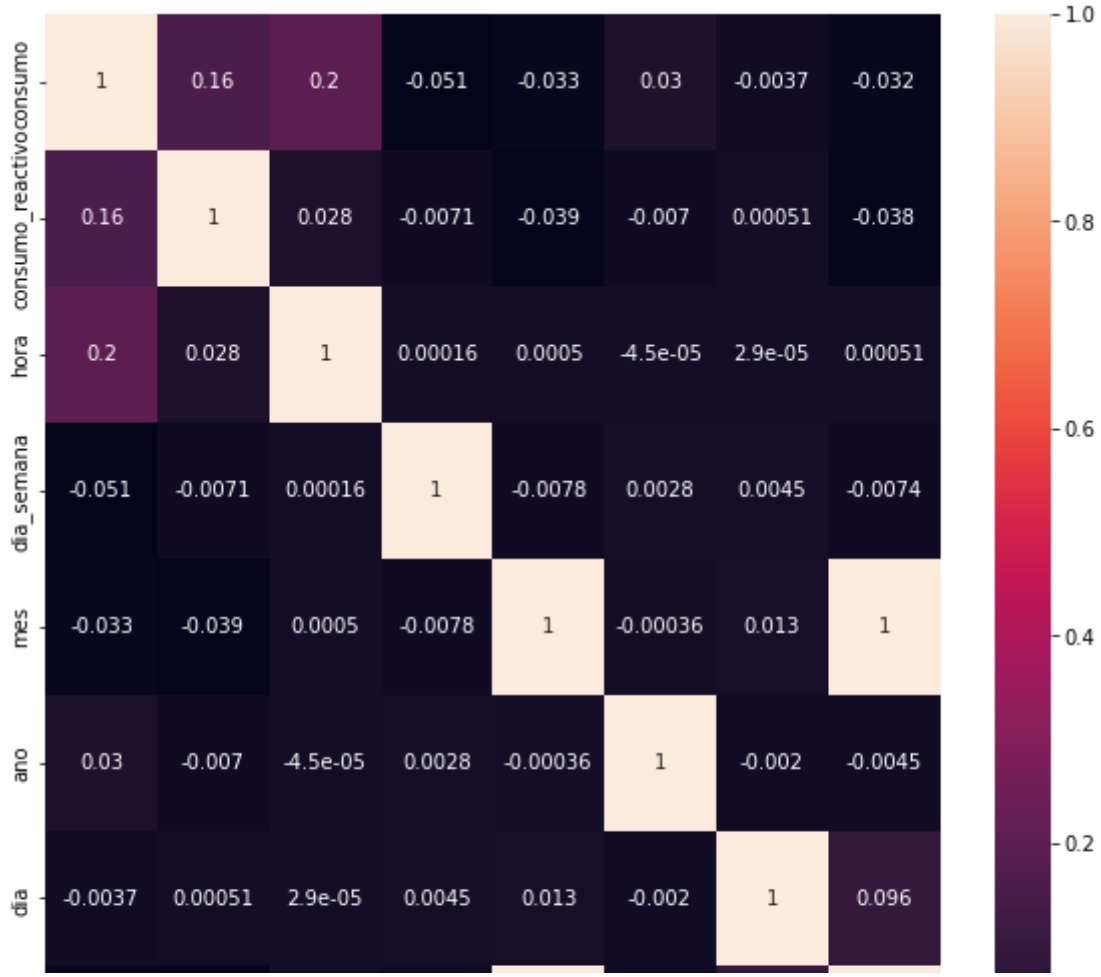
Descripcion de los datos

```
df.describe().T
```

	count	mean	std	min	25%	
consumo	1585035.0	327.938477	220.071274	-337.538025	153.266068	301.719
consumo_reactivo	1585035.0	3.013422	100.012482	-297.052979	-11.758000	0.806

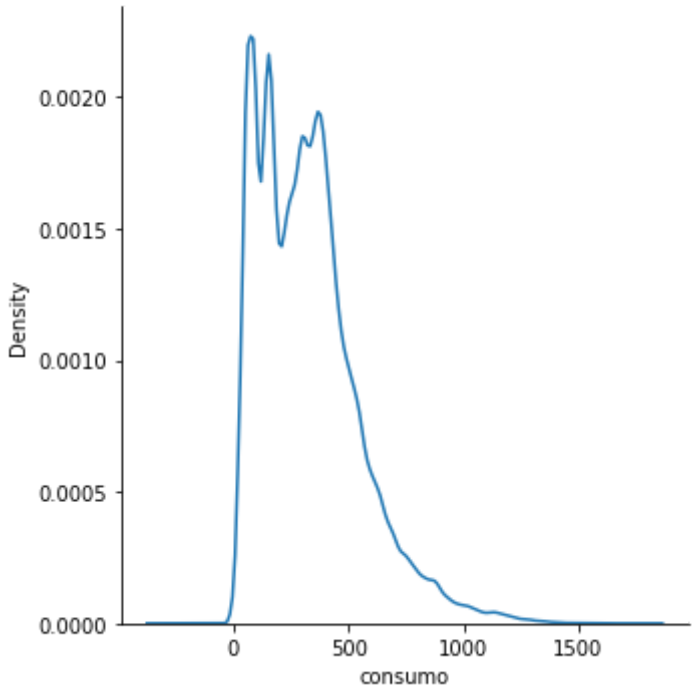
```
df_copia = df.copy()
df_copia['hora'] = pd.DatetimeIndex(df['fecha']).hour
df_copia['dia_semana'] = pd.DatetimeIndex(df['fecha']).dayofweek
df_copia['mes'] = pd.DatetimeIndex(df['fecha']).month
df_copia['ano'] = pd.DatetimeIndex(df['fecha']).year
df_copia['dia'] = pd.DatetimeIndex(df['fecha']).day
df_copia['dia_ano'] = pd.DatetimeIndex(df['fecha']).dayofyear


plt.figure(figsize=(10,10))
sns.heatmap(df_copia.corr(),annot=True)
```

```
<AxesSubplot:>
```



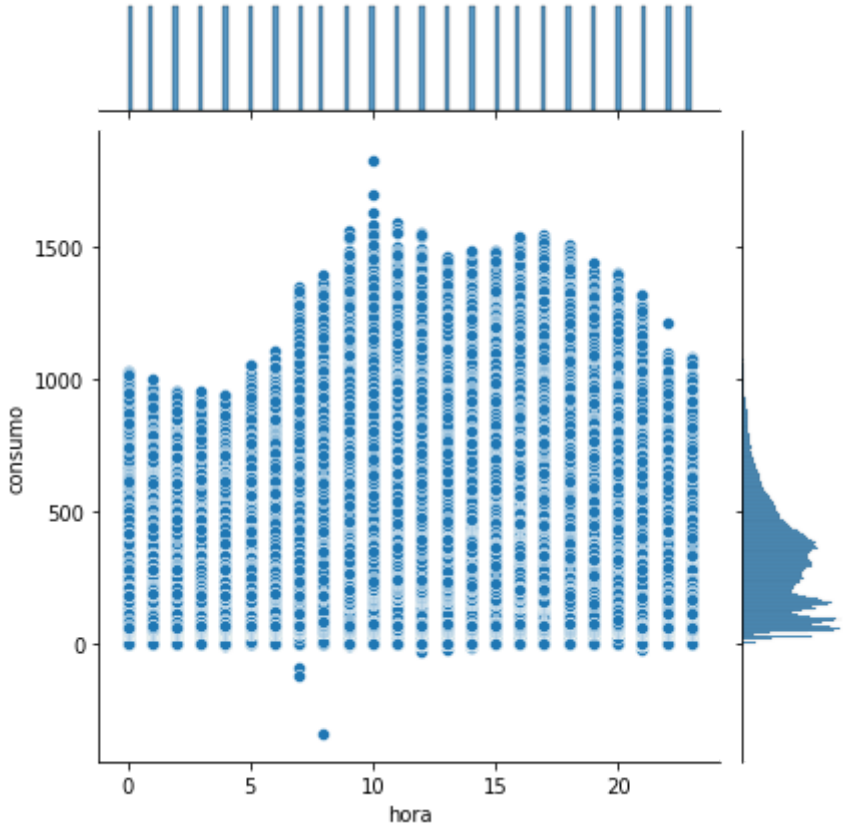

```
##seleccionar columnas
sns.displot(df,x='consumo',kind='kde')
```

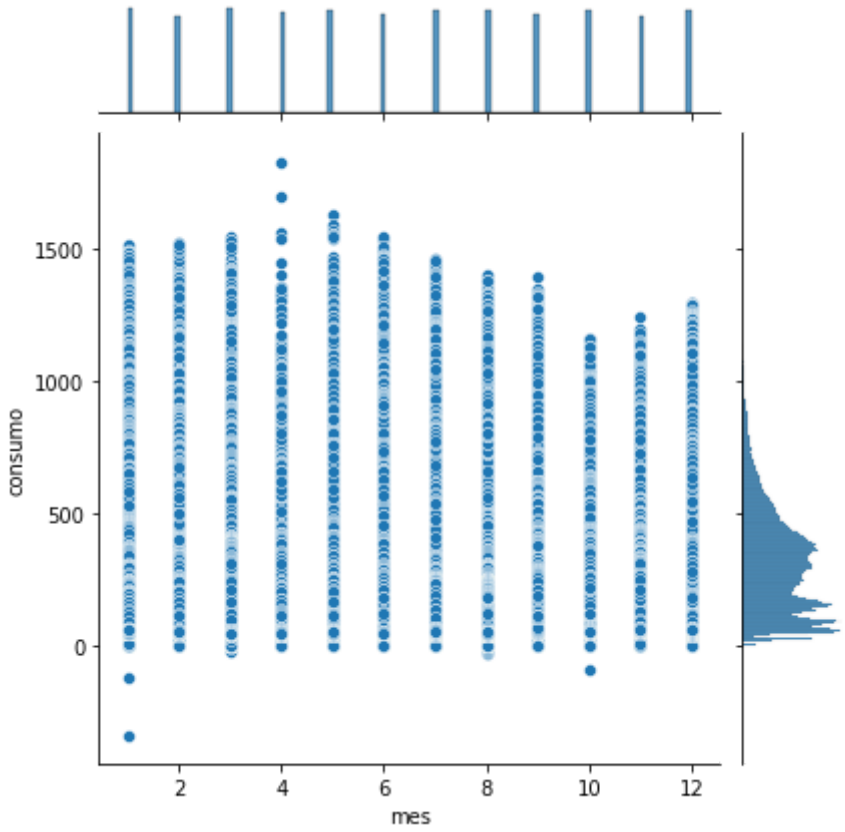
```
<seaborn.axisgrid.FacetGrid at 0x7efddcf4a340>
```



```
##distribuciones de hora y mes
sns.jointplot(data=df_copia,x='hora',y='consumo')
sns.jointplot(data=df_copia,x='mes',y='consumo')
```

```
<seaborn.axisgrid.JointGrid at 0x7efd58e46730>
```





IMMPLEMENTACION KMEANS

Aqui buscare crear 3 grupos de cluster para los medidores buscaremos crear tipo BAJO,MEDIO,ALTO consumo.

```
## implementacion de cluster para clasificar 3 tipos de medidores
from sklearn.cluster import KMeans


##seleccion de features
X = df.drop(['fecha','id_medidor'],axis=1)


##clasficar en 3 tipos de medidores por su uso de energia
model = KMeans(n_clusters=3).fit(X)


model.predict(X)
```

```
array([2, 2, 2, ..., 0, 0, 0], dtype=int32)
```

```
model.cluster_centers_
```

```
array([[140.62172  ,  -6.1364594],
       [758.245    ,  49.996254 ],
       [397.32562  ,  -0.7969482]], dtype=float32)
```

```
df['grupo'] = model.predict(X)


##TODO: cambiar esta consulta por anho
df.groupby(['grupo'])[['consumo','consumo_reactivo']].mean()
```

grupo	consumo	consumo_reactivo
0	140.839706	-6.143421
1	759.042297	50.127384
2	397.768829	-0.755224

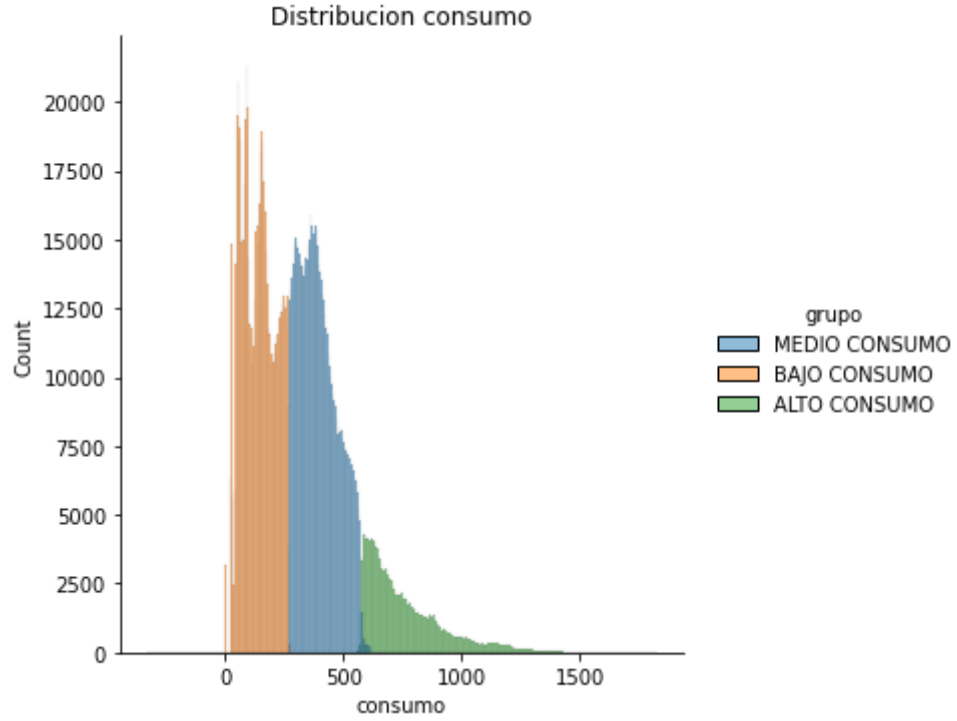
```
##cambiar valores de 0,1,2 a medio,bajo,alto
df['grupo'] = df['grupo'].map({2 : 'MEDIO CONSUMO', 1 : 'ALTO CONSUMO', 0 : 'BAJO CONS
df.sample(10)
```

	id_medidor	fecha	consumo	consumo_reactivo	grupo
514120	621831	2021-02-05 01:00:00	84.209976	-1.645000	BAJO CONSUMO
1529172	760809	2020-06-01 09:30:00	257.196014	-14.883000	BAJO CONSUMO
486546	621831	2020-07-21 14:30:00	374.733673	30.413000	MEDIO CONSUMO
653270	631218	2021-02-03 03:45:00	149.688522	6.876000	BAJO CONSUMO
775816	651903	2020-08-07 16:15:00	560.031128	-254.160004	MEDIO CONSUMO
		2021-07-11			BAJO

Distribucion de consumo por grupo

```
plt.figure(figsize=(15,15))
sns.displot(df,x='consumo',hue='grupo')
plt.title('Distribucion consumo')
plt.show()
```

```
<Figure size 1080x1080 with 0 Axes>
```



Consumo hora año 2020

```
##graficar linea de consumo promedio de los 3 tipos por anho
df2 = df.copy()
```

```
df2 = df2.astype({'id_medidor' : 'str',
                  'fecha' : 'str',
                  'consumo' : 'float32',
                  'consumo_reactivo' : 'float32'})


df2['hora'] = df2['fecha'].str.split(' ',expand=True)[1]
```

Haz doble clic (o pulsa Intro) para editar

```
df2.fecha = pd.to_datetime(df2.fecha)


##Agregar dia y mes

df2['mes'] = pd.DatetimeIndex(df2['fecha']).month
df2['dia'] = pd.DatetimeIndex(df2['fecha']).day

##Cambiar valores numericos por nombre de los meses
df2['mes'] = df2['mes'].map({
    1 : "Enero",
    2 : "Febrero",
    3 : "Marzo",
    4 : "Abril",
    5 : "Mayo",
    6 : "Junio",
    7 : "Julio",
    8 : "Agosto",
    9 : "Septiembre",
    10 : "Octubre",
    11 : "Noviembre",
    12 : "Diciembre"
})
```

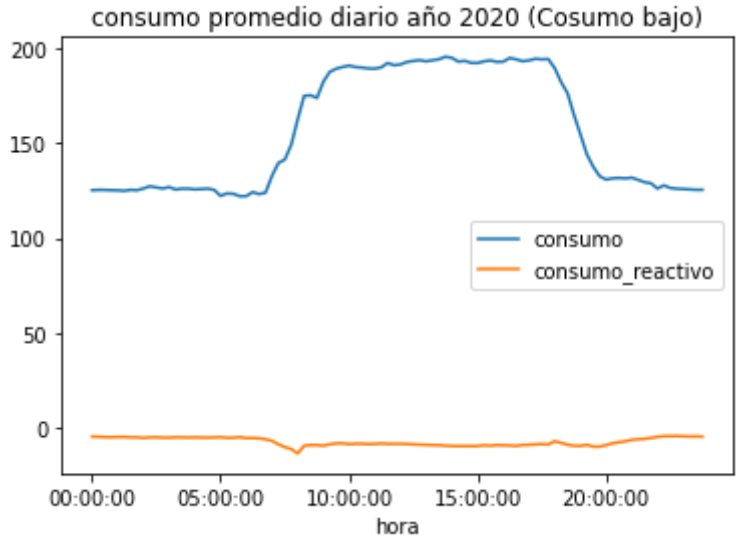
Año 2020

COMSUO BAJO - PROMEDIO HORAS 2020

```
q = df2.query('(fecha < 2021) and grupo == "Bajo consumo"')


q = q.groupby(['hora'])[['consumo','consumo_reactivo']].mean()


plt.figure(figsize=(10,10))
q.plot()
```

```
plt.title('consumo promedio diario año 2020 (Cosumo bajo)')
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```



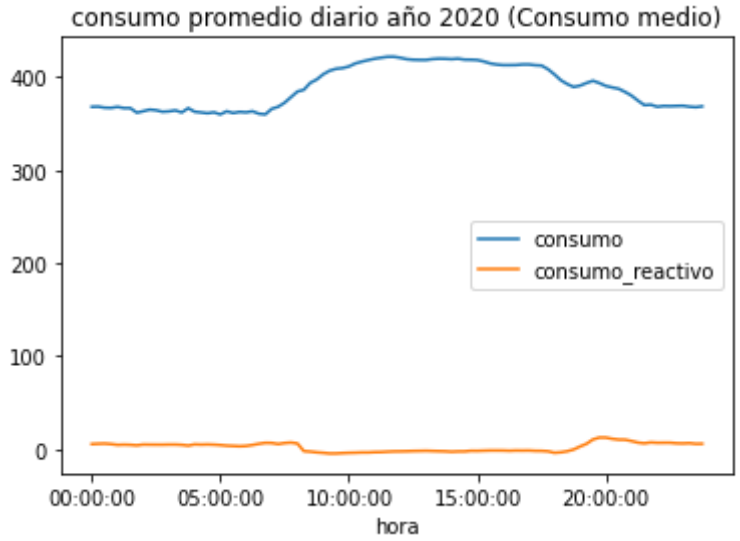
CONSUMO MEDIO - PROMEDIO HORAS 2020

```
q = df2.query('(fecha < 2021) and grupo == "Medio consumo"')
```

```
q = q.groupby(['hora'])[['consumo','consumo_reactivo']].mean()
```

```
plt.figure(figsize=(10,10))
q.plot()
plt.title('consumo promedio diario año 2020 (Consumo medio)')
plt.show()
```

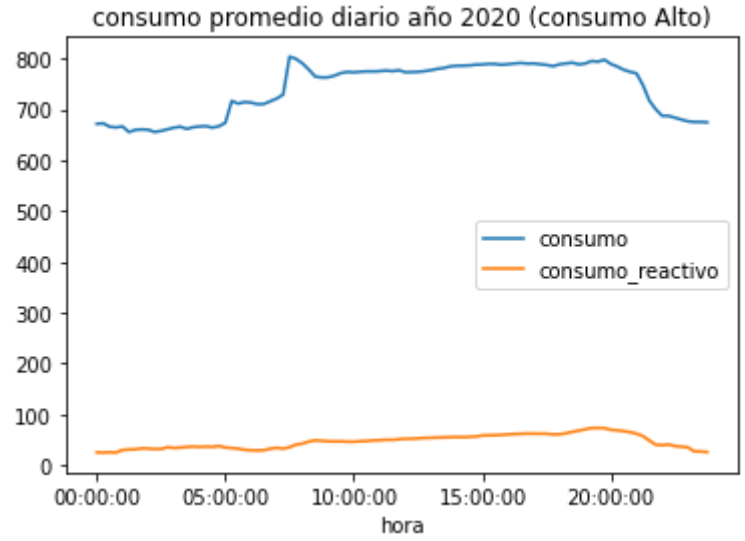
```
<Figure size 720x720 with 0 Axes>
```



CONSUMO ALTO - PROMEDIO HORAS 2020

```
q = df2.query('(fecha < 2021) and grupo == "Alto consumo"')


q = q.groupby(['hora'])[['consumo','consumo_reactivo']].mean()


plt.figure(figsize=(10,10))
q.plot()
plt.title('consumo promedio diario año 2020 (consumo Alto)')
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```



SERIE TEMPORAL 2020 MESES

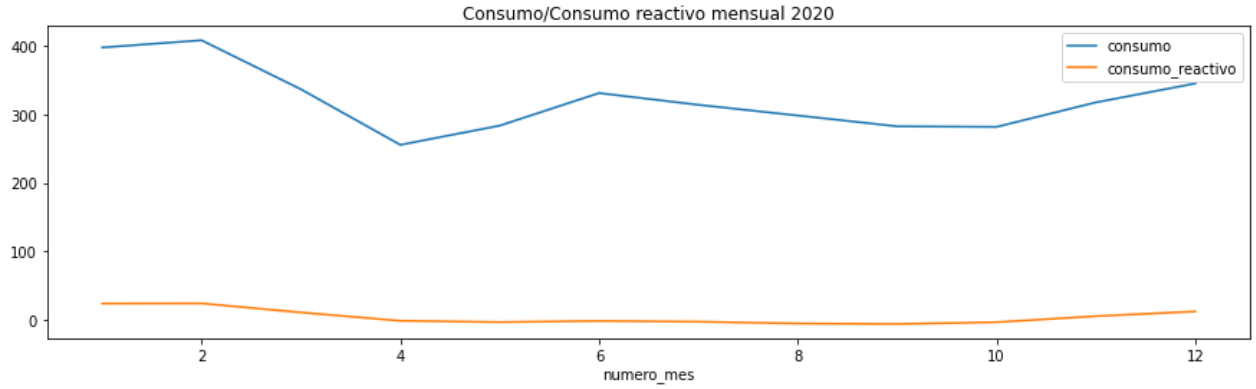
```
q = df2.query('fecha < 2021')
q['numero_mes'] = pd.DatetimeIndex(q.fecha).month
q = q.sort_values('numero_mes')
q = q.groupby(['numero_mes'])[['consumo','consumo_reactivo']].mean()
```

```
<ipython-input-30-68326f1fad70>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
  q['numero_mes'] = pd.DatetimeIndex(q.fecha).month
```

```
q
```

numero_mes	consumo	consumo_reactivo
1	398.187164	23.171957
2	408.782135	23.357054
3	337.011963	10.075310
4	255.601898	-2.116577
5	283.734100	-4.046050
6	331.459534	-2.400309
7	314.225403	-3.468721
8	298.548218	-6.070608
9	282.733612	-6.795469

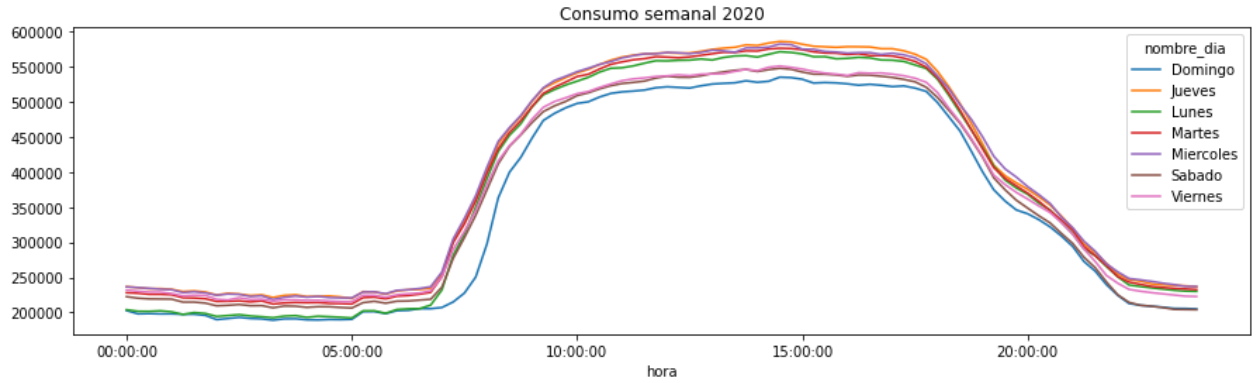
```
##consumo mensual
_ = q.pivot_table(index=['numero_mes'],
                  values=['consumo','consumo_reactivo'],
                  aggfunc='sum').plot(figsize=(15,4),
                  title='Consumo/Consumo reactivo mensual 2020')
```

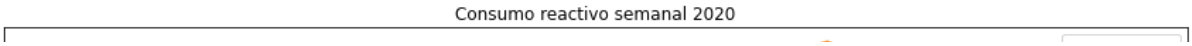



```
##Agrear dia

df2['nombre_dia'] = df2['fecha'].dt.strftime("%A")


df2.sample(5)
```

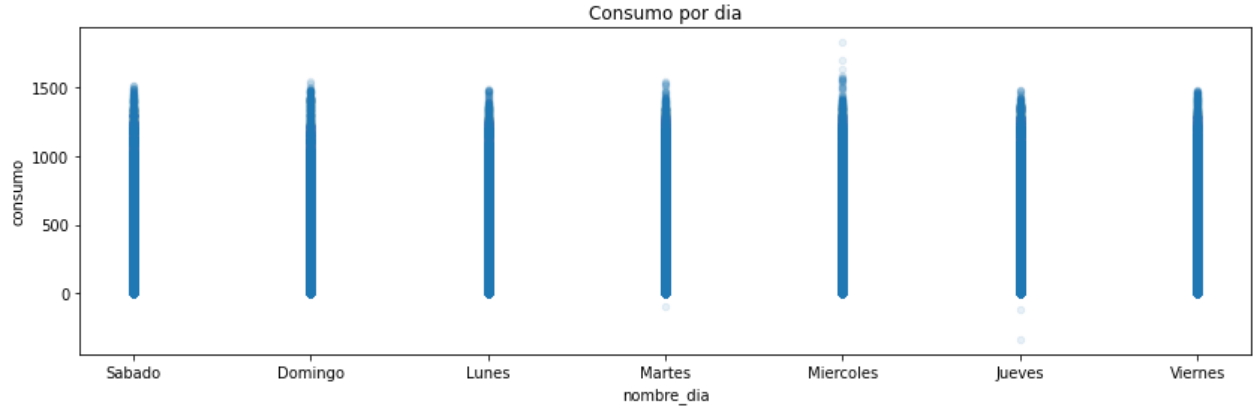
	id_medidor	fecha	consumo	consumo_reactivo	grupo	hora	
1367687	739788	2021-06-13 04:45:00	124.236519	19.011999	Bajo consumo	04:45:00	
355736	585669	2020-10-24 00:00:00	180.684280	13.869000	Bajo consumo	00:00:00	O

```
##cambiar nombres en ingles a espanhol
df2['nombre_dia'] = df2['nombre_dia'].map({'Monday' : 'Lunes',
                                           'Tuesday' : 'Martes',
                                           'Wednesday' : 'Miercoles',
                                           'Thursday' : 'Jueves',
                                           'Friday' : 'Viernes',
                                           'Saturday' : 'Sabado',
                                           'Sunday' : 'Domingo'})


q = df2.query('fecha < 2021')
q

_ = q.pivot_table(index=q['hora'],
                    columns='nombre_dia',
                    values='consumo',
                    aggfunc='sum').plot(figsize=(15,4),
                    title='Consumo semanal 2020')
_ = q.pivot_table(index=q['hora'],
                    columns='nombre_dia',
                    values='consumo_reactivo',
                    aggfunc='sum').plot(figsize=(15,4),
                    title='Consumo reactivo semanal 2020')
```



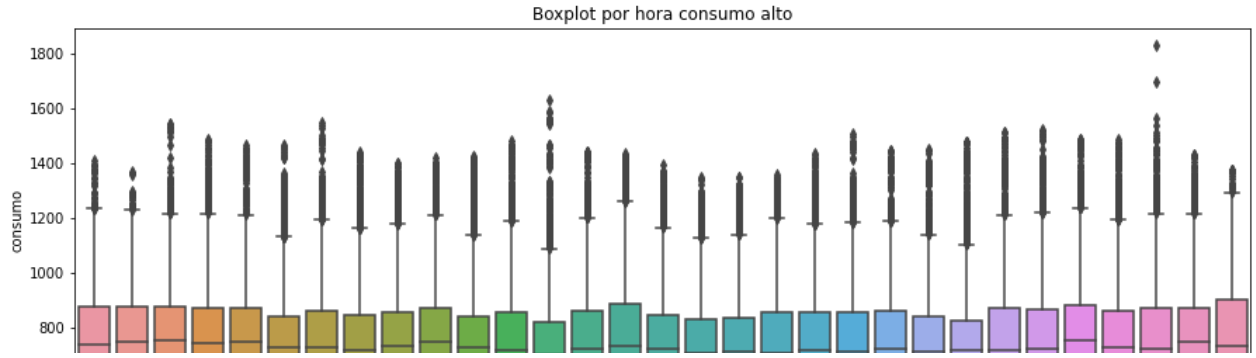



```
_ = q[['consumo','nombre_dia']].plot(x='nombre_dia',
                                     y='consumo',
                                     kind='scatter',
                                     figsize=(14,4),
                                     title='Consumo por dia',
                                     alpha=0.1)
```



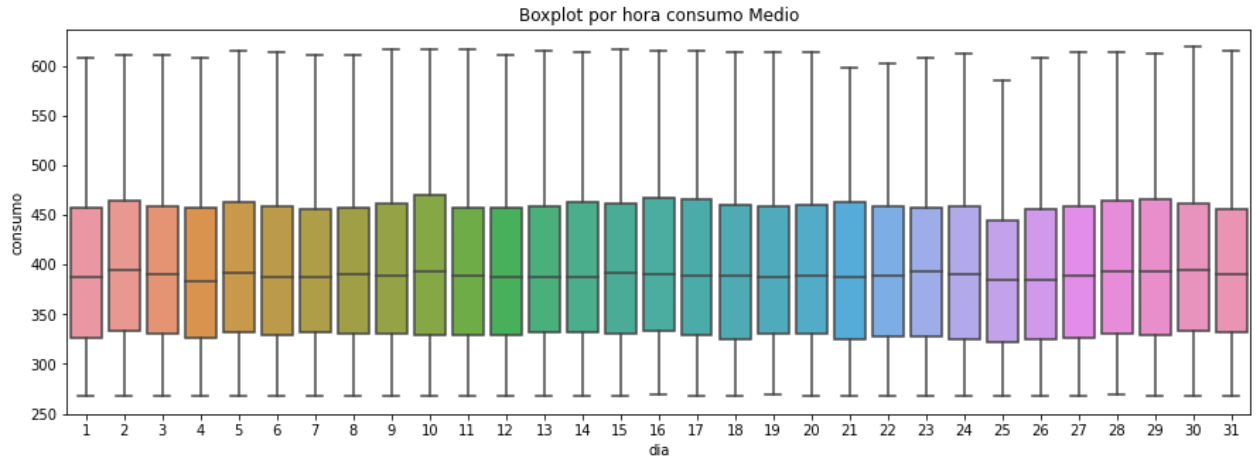

```
##BOXPLOT CONSUMO ALTO
fig,ax = plt.subplots(figsize=(15,5))
sns.boxplot(q.loc[df2['grupo']=='Alto consumo'].dia, q.loc[df2['grupo']=='Alto consumo
ax.set_title('Boxplot por hora consumo alto')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
  warnings.warn(
```



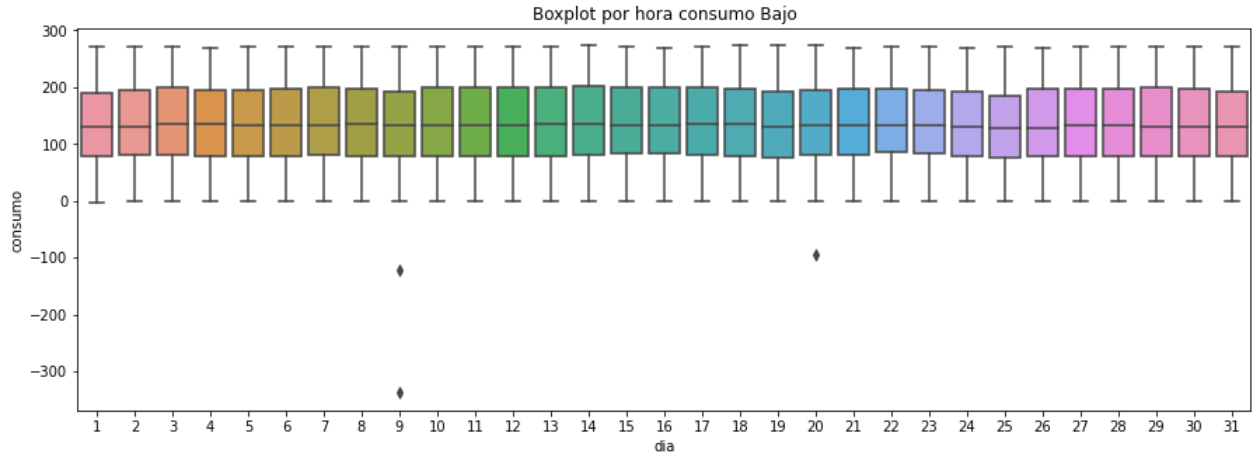

```
##BOXPLOT CONSUMO MEDIO
fig,ax = plt.subplots(figsize=(15,5))
sns.boxplot(q.loc[df2['grupo']=='Medio consumo'].dia, q.loc[df2['grupo']=='Medio consu
ax.set_title('Boxplot por hora consumo Medio')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
  warnings.warn(
```




```
##BOXPLOT CONSUMO BAJO
fig,ax = plt.subplots(figsize=(15,5))
sns.boxplot(q.loc[df2['grupo']=='Bajo consumo'].dia, q.loc[df2['grupo']=='Bajo consumo
ax.set_title('Boxplot por hora consumo Bajo')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
  warnings.warn(
```



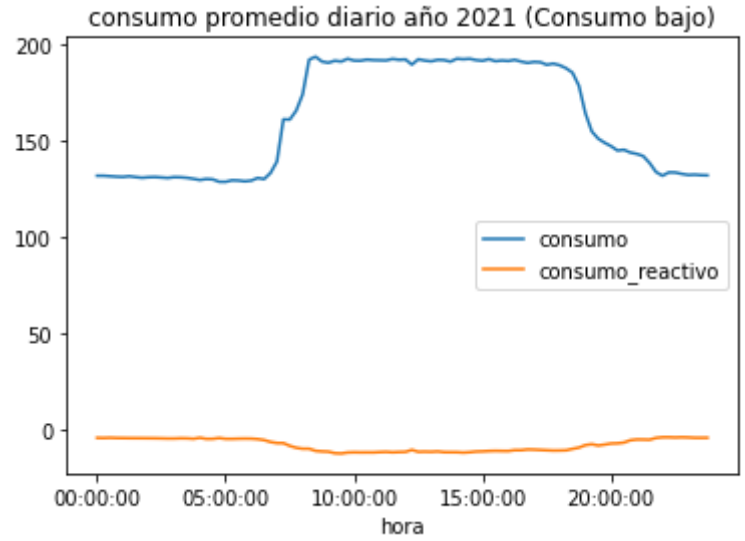
Año 2021

CONSUMO BAJO - HORA PROMEDIO

```
q2 = df2.query('(fecha >= 2021) and grupo == "Bajo consumo"')
q2 = q2.groupby(['hora'])[['consumo','consumo_reactivo']].mean()
```

```
plt.figure(figsize=(10,10))
q2.plot()
plt.title('consumo promedio diario año 2021 (Consumo bajo)')
plt.show()
```

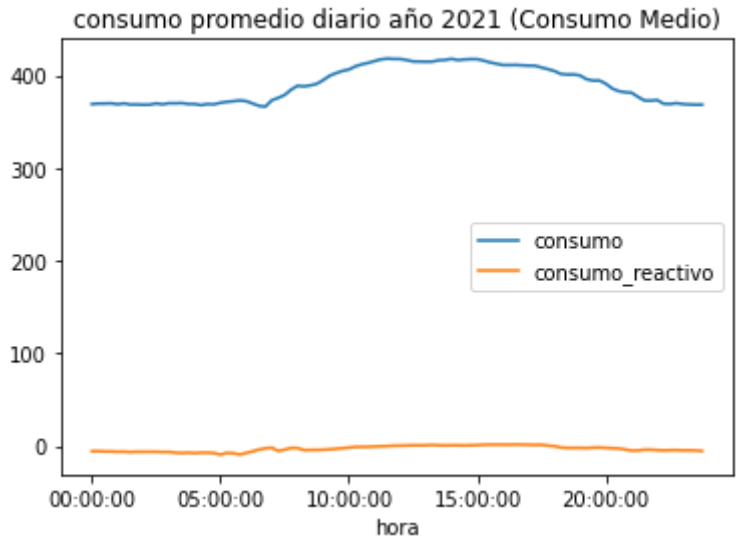
```
<Figure size 720x720 with 0 Axes>
```



CONSUMO MEDIO - HORA PROMEDIO 2021

```
q2 = df2.query('fecha >= 2021 and grupo == "Medio consumo"')
q2 = q2.groupby(['hora'])[['consumo','consumo_reactivo']].mean()


plt.figure(figsize=(10,10))
q2.plot()
plt.title('consumo promedio diario año 2021 (Consumo Medio)')
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```



CONSUMO ALTO - HORA PROMEDIO 2021

```
q2 = df2.query('(fecha >= 2021) and grupo == "Alto consumo"')
q2 = q2.groupby(['hora'])[['consumo','consumo_reactivo']].mean()


plt.figure(figsize=(10,10))
q2.plot()
plt.title('consumo promedio diario año 2021 (Consumo Alto)')
plt.show()
```

```
<Figure size 720x720 with 0 Axes>
```

SERIE TEMPORAL 2021

```
q2 = df2.query('fecha >= 2021')
q2['numero_mes'] = pd.DatetimeIndex(q2.fecha).month
q2 = q2.sort_values('numero_mes')
q2 = q2.groupby(['numero_mes'])[['consumo','consumo_reactivo']].mean()
```

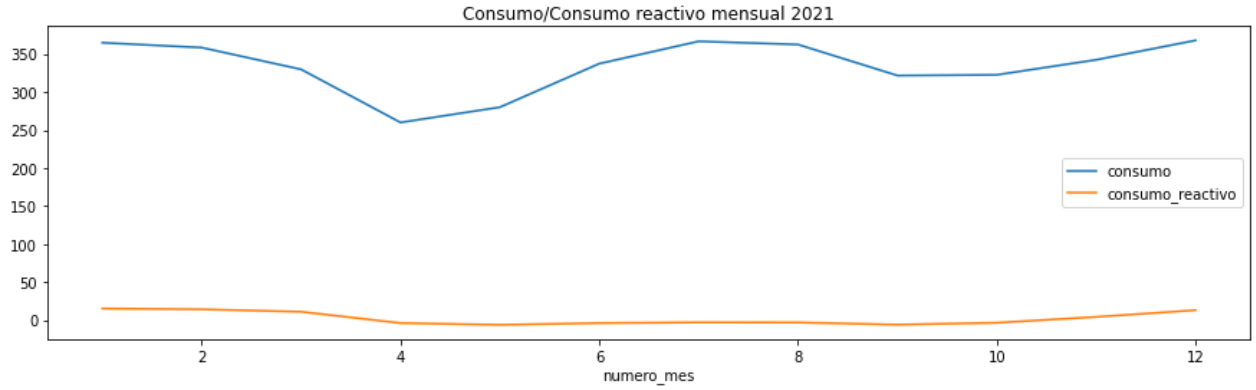
```
<ipython-input-52-94efffd75eed>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
  q2['numero_mes'] = pd.DatetimeIndex(q2.fecha).month
```

```
q2
```

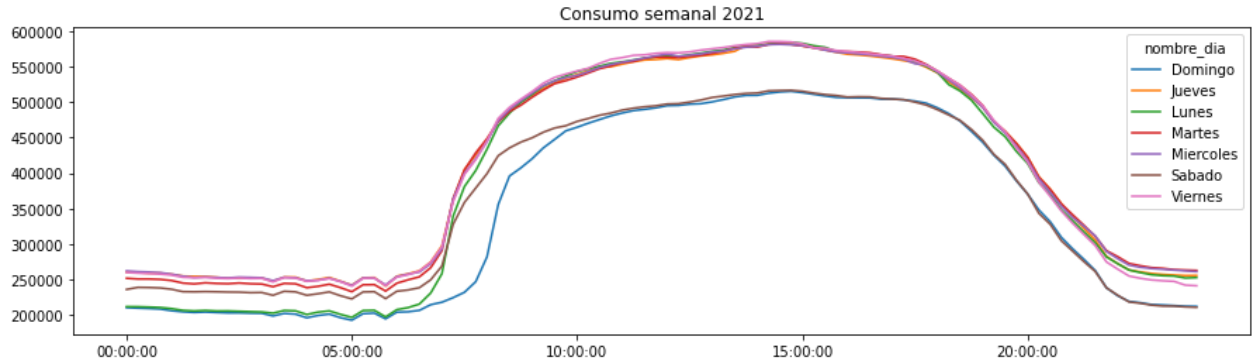
numero_mes	consumo	consumo_reactivo
1	364.719910	15.236846
2	358.323395	14.190757
3	329.679321	10.975924
4	259.859131	-3.909520
5	279.931366	-6.190374
6	337.192505	-3.906322
7	366.558746	-2.983154
8	362.339172	-3.080586
9	321.604614	-5.982365
10	322.505371	-3.482473
11	342.248169	4.184866
12	367.784546	13.049210

```
##consumo mensual 2021
_ = q2.pivot_table(index=['numero_mes'],
                   values=['consumo','consumo_reactivo'],
                   aggfunc='sum').plot(figsize=(15,4),
```

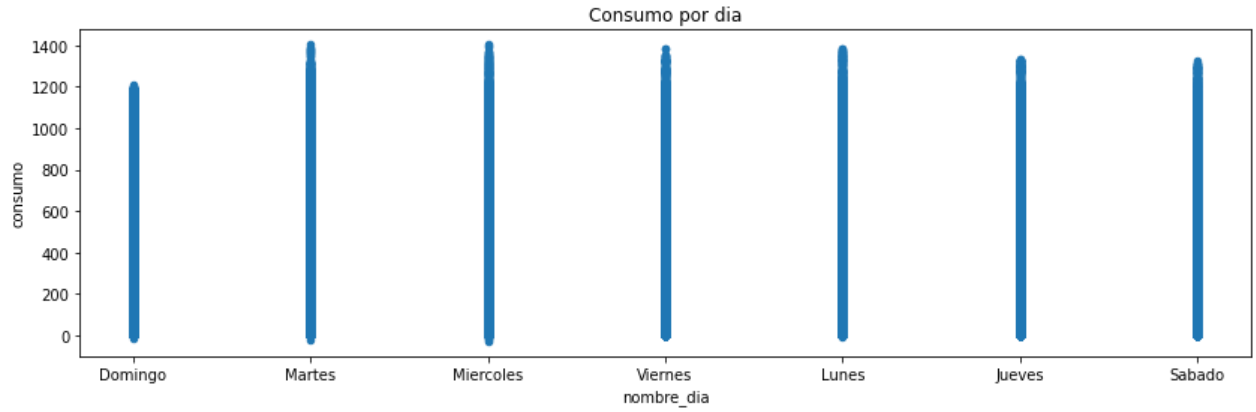
```
                title='Consumo/Consumo reactivo mensual 2021')
```




```
q = df2.query('fecha >= 2021')

_ = q.pivot_table(index=q['hora'],
                  columns='nombre_dia',
                  values='consumo',
                  aggfunc='sum').plot(figsize=(15,4),
                  title='Consumo semanal 2021')
_ = q.pivot_table(index=q['hora'],
                  columns='nombre_dia',
                  values='consumo_reactivo',
                  aggfunc='sum').plot(figsize=(15,4),
                  title='Consumo reactivo semanal 2021')
```

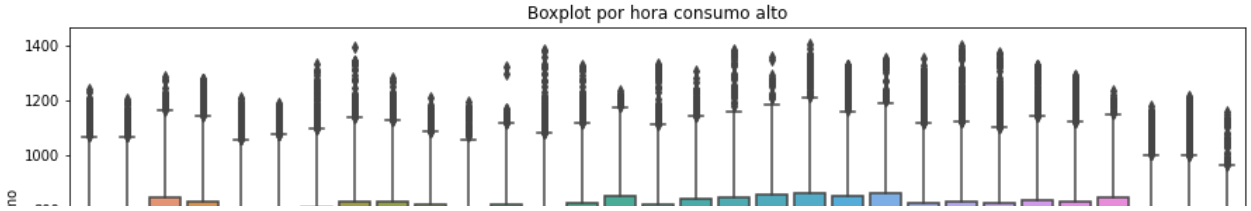



```
_ = q[['consumo','nombre_dia']].plot(x='nombre_dia',
                                     y='consumo',
                                     kind='scatter',
                                     figsize=(14,4),
                                     title='Consumo por dia')
```



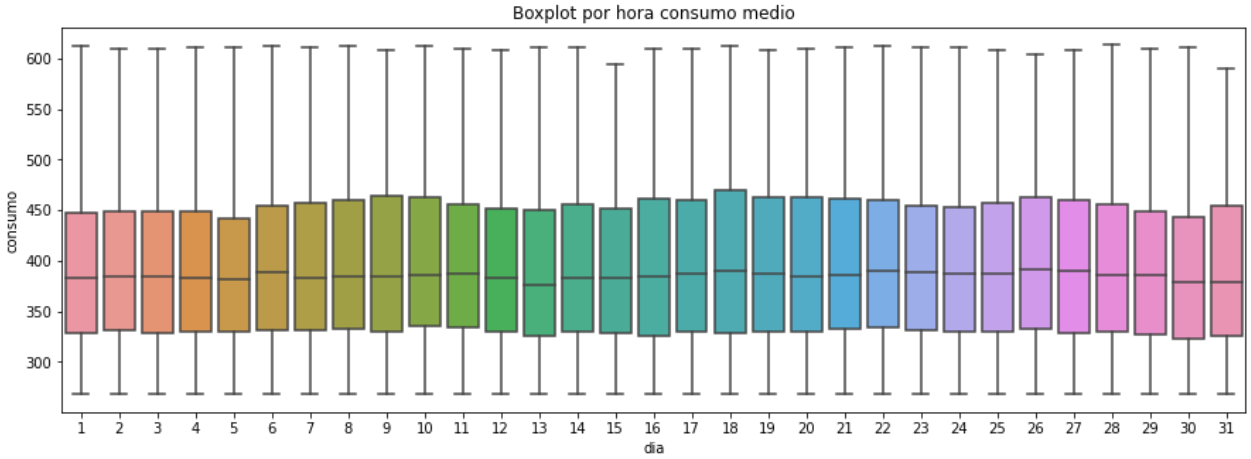

```
fig,ax = plt.subplots(figsize=(15,5))
sns.boxplot(q.loc[df2['grupo']=='Alto consumo'].dia, q.loc[df2['grupo']=='Alto consumo
ax.set_title('Boxplot por hora consumo alto')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
  warnings.warn(
```



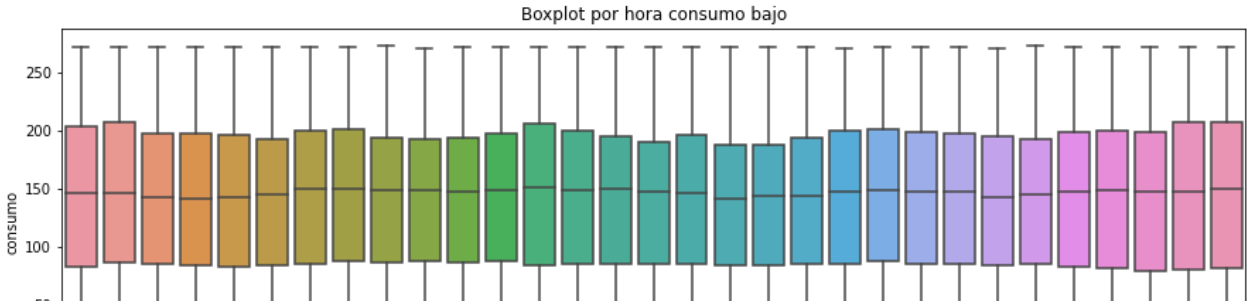

```
fig,ax = plt.subplots(figsize=(15,5))
sns.boxplot(q.loc[df2['grupo']=='Medio consumo'].dia, q.loc[df2['grupo']=='Medio consu
ax.set_title('Boxplot por hora consumo medio')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
  warnings.warn(
```




```
fig,ax = plt.subplots(figsize=(15,5))
sns.boxplot(q.loc[df2['grupo']=='Bajo consumo'].dia, q.loc[df2['grupo']=='Bajo consumo
ax.set_title('Boxplot por hora consumo bajo')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWar
  warnings.warn(
```



Comparacion 2020/2021

[ ] ↳ *7 celdas ocultas*

Implementacion de XGBOOST

```
df.sample()
```

	id_medidor	fecha	consumo	consumo_reactivo	grupo
1279237	731913	2020-01-03 15:45:00	165.389191	65.117004	BAJO CONSUMO

```
##CREACION DE LOS FEATURES
"""
TOMAREMOS LOS DATOS DE LA FECHA Y LA SEPARAREMOS PARA CREAR EL MODELO
"""

df3 = df.copy()

df3['hora'] = pd.DatetimeIndex(df3['fecha']).hour
df3['dia_semana'] = pd.DatetimeIndex(df3['fecha']).dayofweek
df3['mes'] = pd.DatetimeIndex(df3['fecha']).month
df3['ano'] = pd.DatetimeIndex(df3['fecha']).year
df3['dia'] = pd.DatetimeIndex(df3['fecha']).day
df3['dia_ano'] = pd.DatetimeIndex(df3['fecha']).dayofyear

df3
```

	id_medidor	fecha	consumo	consumo_reactivo	grupo	hora	di
1	439950	2020-02-01 00:15:00	466.352234	21.632000	MEDIO CONSUMO	0	
2	439950	2020-02-01 00:30:00	463.451782	20.944000	MEDIO CONSUMO	0	
3	439950	2020-02-01 00:45:00	461.775269	20.830000	MEDIO CONSUMO	0	
4	439950	2020-02-01 01:00:00	462.097534	21.186001	MEDIO CONSUMO	1	
5	439950	2020-02-01 01:15:00	460.127716	21.273001	MEDIO CONSUMO	1	
...	...	...	...	...	...	...	
1585031	760809	2021-01-02 23:00:00	59.075893	0.000000	BAJO CONSUMO	23	
1585032	760809	2021-01-02	57.619526	0.000000	BAJO CONSUMO	23	

```
df3 = df3.astype({
    'id_medidor' : 'str',
    'consumo' : 'float32',
    'consumo_reactivo' : 'float32'
})


##aplicar get dummies
df3 = pd.get_dummies(df3)


df3.sample(10)
```

	fecha	consumo	consumo_reactivo	hora	dia_semana	mes	ano	dia
436731	2020-06-16 04:30:00	63.722359	-2.718000	4	1	6	2020	1
141622	2020-03-18 07:00:00	619.712402	-3.287000	7	2	3	2020	1
152954	2020-09-12 09:00:00	483.066742	-18.303001	9	5	9	2020	1
608549	2021-10-25 06:00:00	264.753021	-30.738001	6	0	10	2021	2
972100	2020-03-13 16:00:00	497.829559	11.287000	16	4	3	2020	1
122710	2021-08-30 03:30:00	167.996292	-15.509000	3	0	8	2021	3
925869	2020-11-17 23:15:00	53.268532	0.319000	23	1	11	2020	1
1559936	2021-04-03 08:00:00	157.669205	-22.073000	8	5	4	2021	

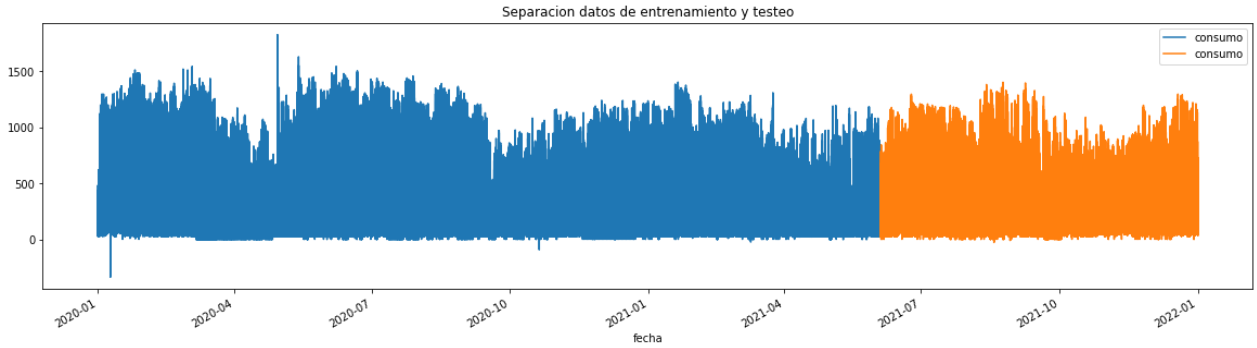
```
import xgboost as xgb
from sklearn.metrics import mean_squared_error
```

Separar data

```
train_df = df3.loc[df3.fecha < '2021-06-04']
test_df = df3.loc[df3.fecha>='2021-06-04']

train_df = train_df.sort_values('fecha')
test_df = test_df.sort_values('fecha')


train_df.shape,test_df.shape
```

```
((1130345, 35), (454690, 35))
```

```
fig,ax = plt.subplots(figsize=(20,5))
train_df[['fecha','consumo']].plot(ax=ax,x='fecha',figsize=(20,5),title="Separacion da
test_df[['fecha','consumo']].plot(ax=ax,x='fecha',figsize=(20,5))
plt.show()
```




```
##separacion de variables de entrenamiento y testeo
X_train = train_df.drop(['consumo','fecha','consumo_reactivo'],axis=1)
X_test = test_df.drop(['consumo','fecha','consumo_reactivo'],axis=1)

y_train = train_df[['consumo']]
y_test = test_df[['consumo']]



print(X_train.shape,X_test.shape,y_train.shape,y_test.shape)

X_test
```

```
(1130345, 32) (454690, 32) (1130345, 1) (454690, 1)
```

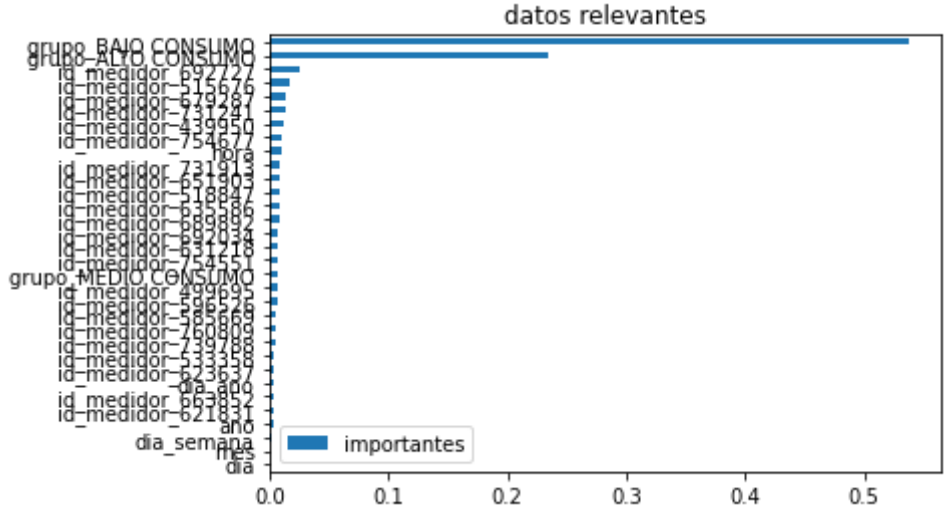
	hora	dia_semana	mes	ano	dia	dia_ano	id_medidor_439950	id_medi
1224606	0	4	6	2021	4	155	0	
1296958	0	4	6	2021	4	155	0	
1014931	0	4	6	2021	4	155	0	
734976	0	4	6	2021	4	155	0	
...	...	...	...	...	...	...	...	

```
modelo = xgb.XGBRegressor(n_estimators=500,booster='gbtree',max_depth=3,early_stopping
```

```
modelo.fit(X_train,y_train,eval_set=[(X_test,y_test)],verbose=100)
```

```
[17:38:40] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is
[0]     validation_0-rmse:384.529
[100]   validation_0-rmse:78.755
[200]   validation_0-rmse:76.2508
[300]   validation_0-rmse:75.3915
[400]   validation_0-rmse:75.1617
[499]   validation_0-rmse:75.338
XGBRegressor(early_stopping_rounds=50, learning_rate=0.05, n_estimators=500,
             objetive='reg:linear')
```

```
datos_importantes = pd.DataFrame(data=modelo.feature_importances_,
                                 index=modelo.get_booster().feature_names,
                                 columns=['importantes'])
datos_importantes.sort_values('importantes').plot(kind='barh',title='datos relevantes'
plt.show()
```




```
test_df['prediccion'] = modelo.predict(X_test)
```

```
test_df
```

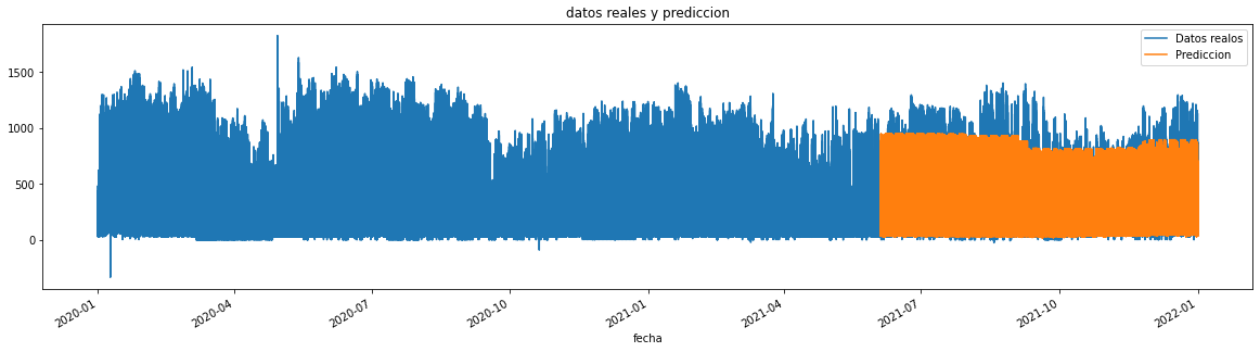
	fecha	consumo	consumo_reactivo	hora	dia_semana	mes	ano	di
1224606	2021-06-04 00:00:00	24.405540	-2.777000	0	4	6	2021	
1296958	2021-06-04 00:00:00	234.528549	46.447002	0	4	6	2021	
1014931	2021-06-04 00:00:00	134.625168	1.057000	0	4	6	2021	
734976	2021-06-04 00:00:00	90.282394	-3.876000	0	4	6	2021	
1087092	2021-06-04 00:00:00	166.786880	-17.849001	0	4	6	2021	
...	...	...	...	...	...	...	...	.
1316843	2021-12-31 23:45:00	193.321579	42.884003	23	4	12	2021	3
1386694	2021-12-31 23:45:00	178.381332	29.862999	23	4	12	2021	3
687831	2021-12-31 23:45:00	210.253357	1.126000	23	4	12	2021	3
897779	2021-12-31 23:45:00	228.495956	-10.571000	23	4	12	2021	3
1247062	2021-12-31 23:45:00	36.981274	-0.431000	23	4	12	2021	3

454690 rows × 36 columns

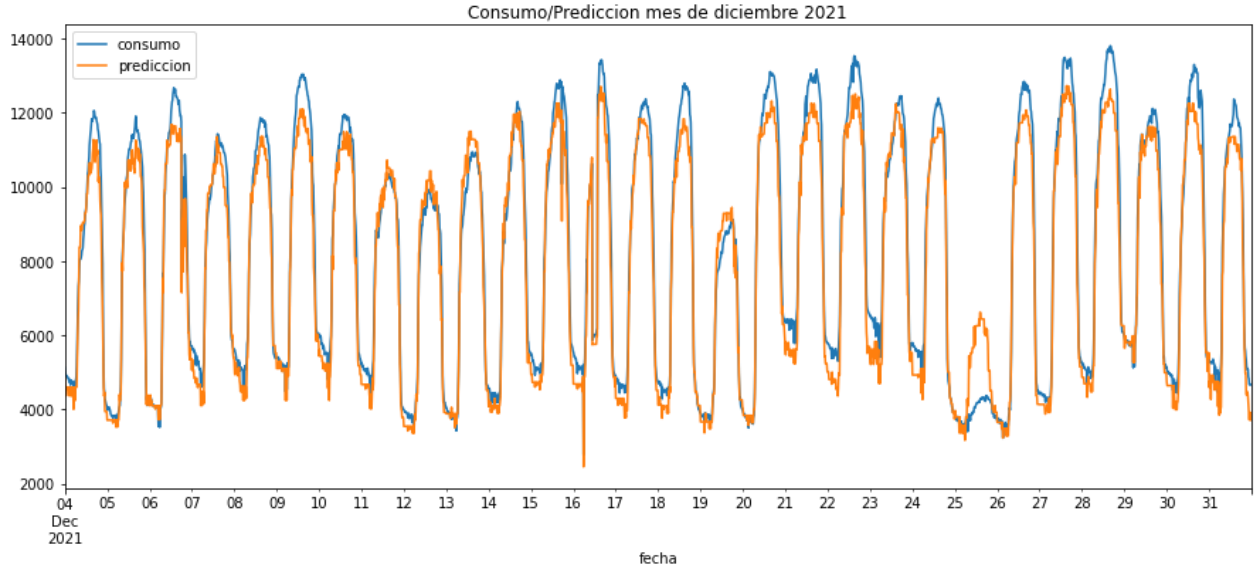
```
df_1 = df.sort_values('fecha').copy()
df_1 = df_1.merge(test_df[['prediccion']],how='left',left_index=True,right_index=True)
df_1
```

	id_medidor	fecha	consumo	consumo_reactivo	grupo	predicci
99362	499695	2020-01-01 00:00:00	229.564163	-2.000000	BAJO CONSUMO	N
1278982	731913	2020-01-01 00:00:00	36.481579	11.946000	BAJO CONSUMO	N
169334	515676	2020-01-01 00:00:00	411.012451	-13.108000	MEDIO CONSUMO	N
719667	635586	2020-01-01 00:00:00	142.714462	-20.728001	BAJO CONSUMO	N
1548854	760809	2020-01-01 00:00:00	178.309631	0.000000	BAJO CONSUMO	N
...	...	...	...	...	...	

```
fig,ax = plt.subplots(figsize=(20,5))
df_1[['fecha','consumo']].plot(ax=ax,x='fecha',figsize=(20,5),title="datos reales y pr
df_1[['fecha','prediccion']].plot(ax=ax,x='fecha')
plt.legend(['Datos realos','Prediccion'])
plt.show()
```




```
_ = df_1.query('fecha >= "2021-12-04"').pivot_table(index=['fecha'],
                  values=['consumo','prediccion'],
                  aggfunc='sum').plot(figsize=(15,6),
                  title='Consumo/Prediccion mes de diciembre 2021')
```




```
##obtener score

score = np.sqrt(mean_squared_error(y_test,test_df['prediccion']))
print(f'RMSE score on el set de testeo : {score:0.2f}')
```

```
    RMSE score on el set de testeo : 75.35
```

guardar csv

```
df.to_csv('mediciones_final.csv',encoding='utf-8')
```