Minería de datos G.571

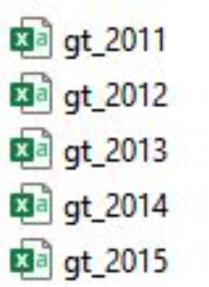
Proyecto final - Regresión: Gas Turbine CO and NOx Emission dataset

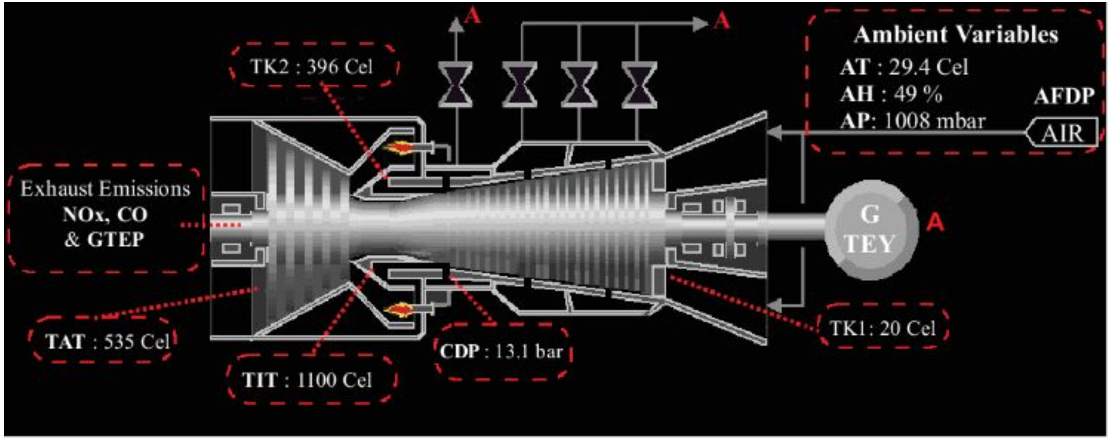
Minería de Datos

Zavala Roman Irvin Eduardo 1270771

Descripción dataset

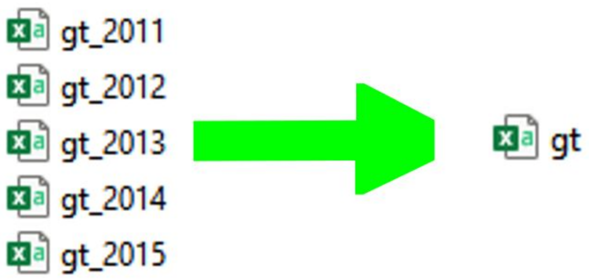
Consiste en 11 atributos que son mediciones de sensores en una turbina de gas en Turquía con variables de ambiente. El dataset está dividido en 5 años y aunque no es pronóstico los datos están ordenados cronológicamente para predecir ciertas emisiones.

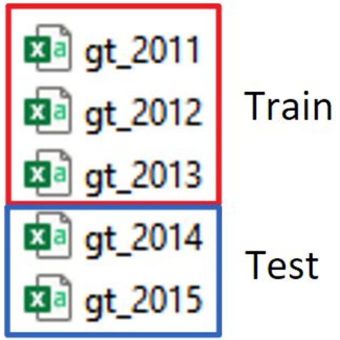




¿Qué hacer con 5 datasets separados?

Se plantean 2 opciones para la integración de los datasets, realizar un análisis de viabilidad para ver si simplemente pegar verticalmente los datos o usar unos datasets para entrenamiento y otros para prueba.

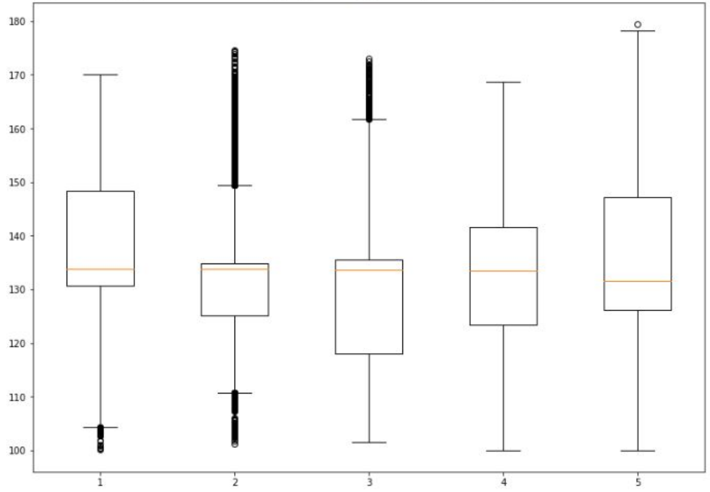




Probando si unir los datasets

```
dataset_2011 = pd.read_csv('https://raw.githubusercontent.com/EduardoZaRo/Mineria-de-datos-571/main/Proyecto/gt_2011.csv')
dataset_2012 = pd.read_csv('https://raw.githubusercontent.com/EduardoZaRo/Mineria-de-datos-571/main/Proyecto/gt_2012.csv')
dataset_2013 = pd.read_csv('https://raw.githubusercontent.com/EduardoZaRo/Mineria-de-datos-571/main/Proyecto/gt_2013.csv')
dataset_2014 = pd.read_csv('https://raw.githubusercontent.com/EduardoZaRo/Mineria-de-datos-571/main/Proyecto/gt_2014.csv')
dataset_2015 = pd.read_csv('https://raw.githubusercontent.com/EduardoZaRo/Mineria-de-datos-571/main/Proyecto/gt_2015.csv')
```

```
data = [dataset_2011['TEY'], dataset_2012['TEY'],dataset_2013['TEY'],dataset_2014['TEY'],dataset_2015['TEY']]
fig = plt.figure(figsize =(10, 7))
ax = fig.add_axes([0, 0, 1, 1])
bp = ax.boxplot(data)
plt.show()
```



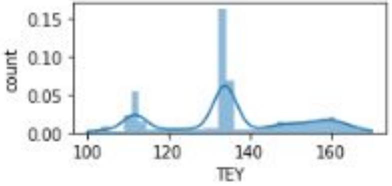

```
print(dataset_2011['TEY'].median())
print(dataset_2012['TEY'].median())
print(dataset_2013['TEY'].median())
print(dataset_2014['TEY'].median())
print(dataset_2015['TEY'].median())
```

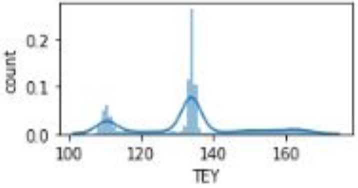
```
133.81
133.76
133.57
133.42
131.6
```

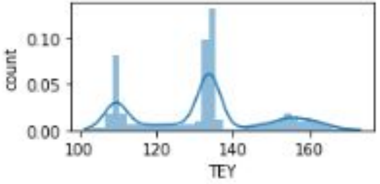
La mediana del target es prácticamente la misma los 5 años!

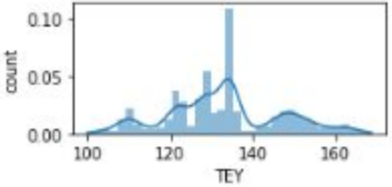
Probando si unir los datasets

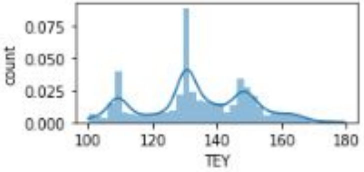
```
histogramas_TEY(dataset_2011)
histogramas_TEY(dataset_2012)
histogramas_TEY(dataset_2013)
histogramas_TEY(dataset_2014)
histogramas_TEY(dataset_2015)
```









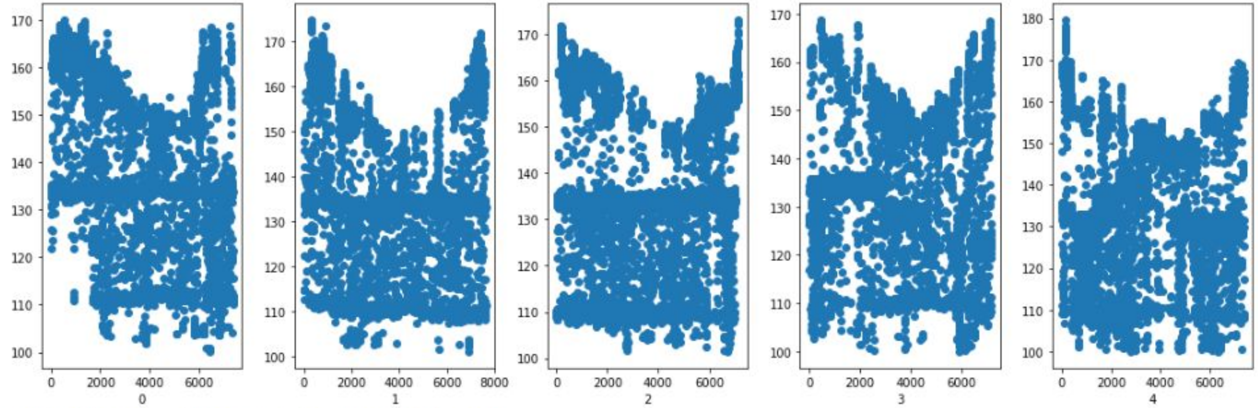


Los histogramas muestran un patrón en la distribución de los datos menos en 2014

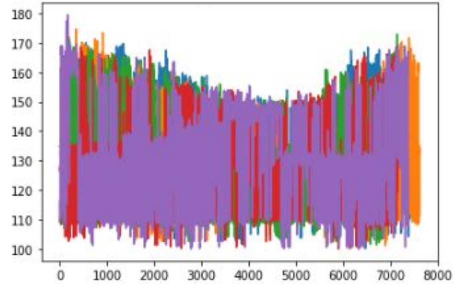
Probando si unir los datasets

```python
fig = plt.figure(figsize=(15, 5))
for i in range(5):
    plt.subplot (1, 5, i+1)
    plt.scatter(np.arange(len(data[i])),data[i])
    plt.xlabel (i)
fig.tight_layout()
plt.show()

dataset_2011['TEY'].plot()
dataset_2012['TEY'].plot()
dataset_2013['TEY'].plot()
dataset_2014['TEY'].plot()
dataset_2015['TEY'].plot()
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7fed5c846710>
```



El target muestra un patron de dispersion parecido los 5 años

¡Se decidió unirlos!

```
dataset = pd.concat([dataset_2011, dataset_2012, dataset_2013, dataset_2014, dataset_2015], axis=0)
dataset = dataset.reset_index()
dataset.pop('index')
```

```
dataset
```

	AT	AP	AH	AFDP	GTEP	TIT	TAT	TEY	CDP	CO	NOX
0	4.5878	1018.7	83.675	3.5758	23.979	1086.2	549.83	134.67	11.898	0.32663	81.952
1	4.2932	1018.3	84.235	3.5709	23.951	1086.1	550.05	134.67	11.892	0.44784	82.377
2	3.9045	1018.4	84.858	3.5828	23.990	1086.5	550.19	135.10	12.042	0.45144	83.776
3	3.7436	1018.3	85.434	3.5808	23.911	1086.5	550.17	135.03	11.990	0.23107	82.505
4	3.7516	1017.8	85.182	3.5781	23.917	1085.9	550.00	134.67	11.910	0.26747	82.028
...	...	...	...	...	...	...	...	...	...	...	...
36728	3.6268	1028.5	93.200	3.1661	19.087	1037.0	541.59	109.08	10.411	10.99300	89.172
36729	4.1674	1028.6	94.036	3.1923	19.016	1037.6	542.28	108.79	10.344	11.14400	88.849
36730	5.4820	1028.5	95.219	3.3128	18.857	1038.0	543.48	107.81	10.462	11.41400	96.147
36731	5.8837	1028.7	94.200	3.9831	23.563	1076.9	550.11	131.41	11.771	3.31340	64.738
36732	6.0392	1028.8	94.547	3.8752	22.524	1067.9	548.23	125.41	11.462	11.98100	109.240

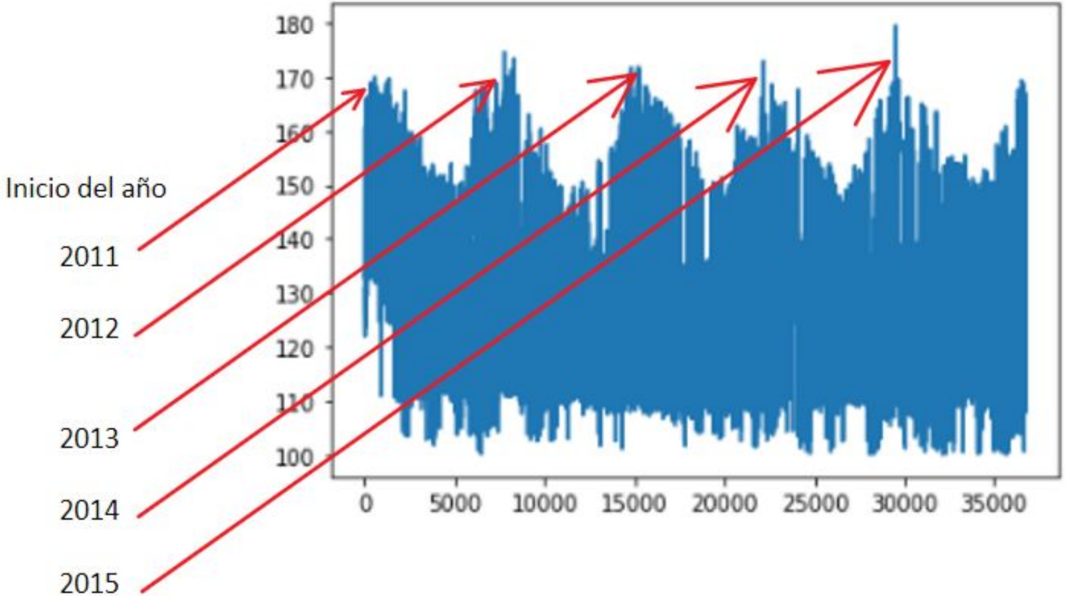
36733 rows × 11 columns

Se hizo un stack vertical para seguir la cronología de los datos

Seguimiento de TEY en el dataset unido

```
dataset['TEY'].plot()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fed5d190a50>



Estadísticas del dataset

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36733 entries, 0 to 36732
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   AT      36733 non-null  float64
 1   AP      36733 non-null  float64
 2   AH      36733 non-null  float64
 3   AFDP    36733 non-null  float64
 4   GTEP    36733 non-null  float64
 5   TIT     36733 non-null  float64
 6   TAT     36733 non-null  float64
 7   TEY     36733 non-null  float64
 8   CDP     36733 non-null  float64
 9   CO      36733 non-null  float64
 10  NOX     36733 non-null  float64
dtypes: float64(11)
memory usage: 3.1 MB
```

```
dataset.describe().T
```

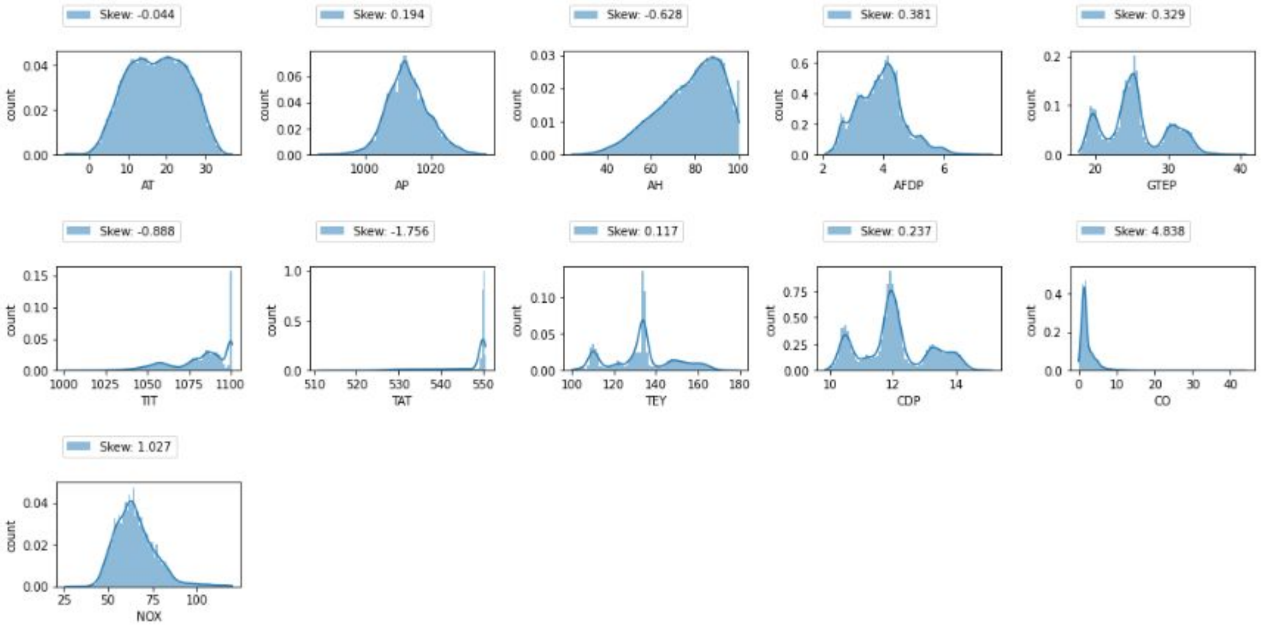
	count	mean	std	min	25%	50%	75%	max
AT	36733.0	17.712726	7.447451	-6.234800	11.7810	17.8010	23.6650	37.1030
AP	36733.0	1013.070165	6.463346	985.850000	1008.8000	1012.6000	1017.0000	1036.6000
AH	36733.0	77.867015	14.461355	24.085000	68.1880	80.4700	89.3760	100.2000
AFDP	36733.0	3.925518	0.773936	2.087400	3.3556	3.9377	4.3769	7.6106
GTEP	36733.0	25.563801	4.195957	17.698000	23.1290	25.1040	29.0610	40.7160
TIT	36733.0	1081.428084	17.536373	1000.800000	1071.8000	1085.9000	1097.0000	1100.9000
TAT	36733.0	546.158517	6.842360	511.040000	544.7200	549.8800	550.0400	550.6100
TEY	36733.0	133.506404	15.618634	100.020000	124.4500	133.7300	144.0800	179.5000
CDP	36733.0	12.060525	1.088795	9.851800	11.4350	11.9650	12.8550	15.1590
CO	36733.0	2.372468	2.262672	0.000388	1.1824	1.7135	2.8429	44.1030
NOX	36733.0	65.293067	11.678357	25.905000	57.1620	63.8490	71.5480	119.9100

```
dataset.isna().sum()
```

```
AT      0
AP      0
AH      0
AFDP    0
GTEP    0
TIT     0
TAT     0
TEY     0
CDP     0
CO      0
NOX     0
dtype: int64
```

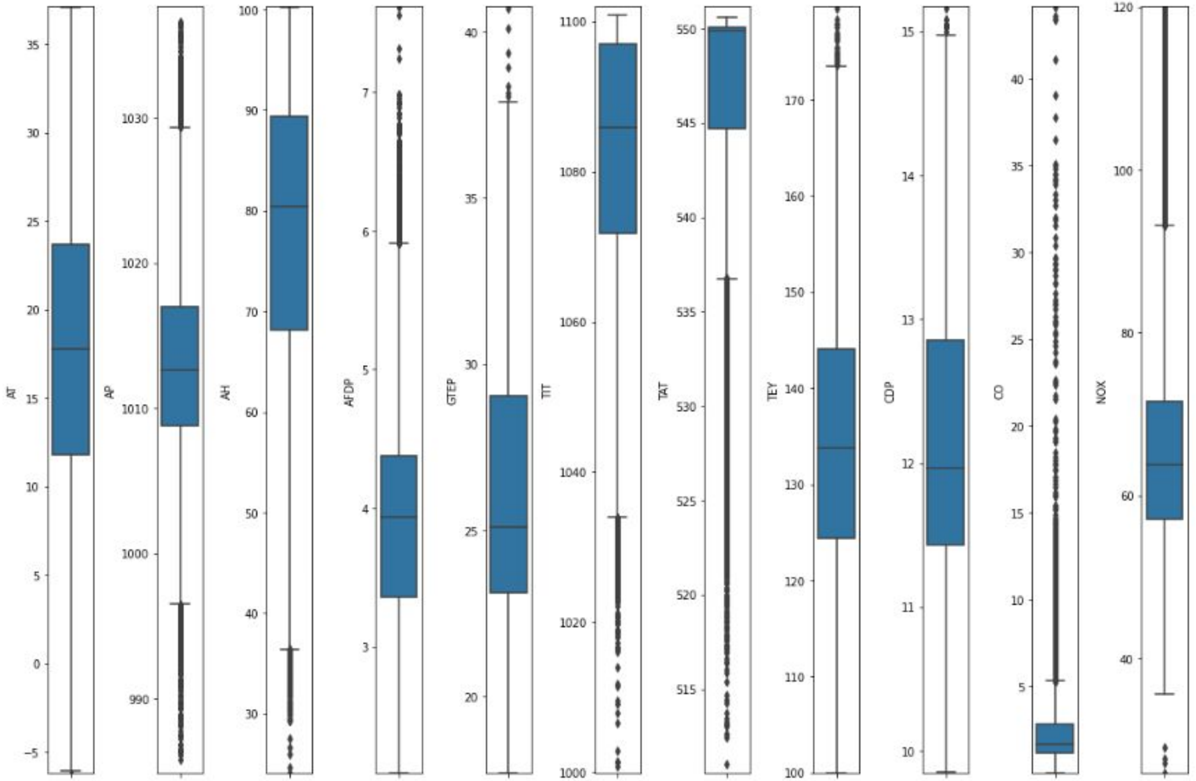
Histogramas de todos los atributos

```
def histogramas(dataset):
    fig = plt.figure(figsize=(15, 15))
    atributos = list(dataset.columns.values )
    for i in range(len(atributos)):
        plt.subplot (math.ceil(dataset.shape[1]/2), math.floor(dataset.shape[1]/2), i+1)
        #plt.hist(dataset[atributos[i]], bins=30, edgecolor='black')
        sn.histplot(dataset[atributos[i]], kde=True, stat="density", linewidth=0, label="Skew: {:.3f}".format(dataset[atributos[i]].skew()))
        plt.legend(loc = 'upper left', bbox_to_anchor=(0, 1.5))
        plt.xlabel (atributos[i])
        plt.ylabel ('count')
    fig.tight_layout()
    plt.show()
histogramas(dataset)
```



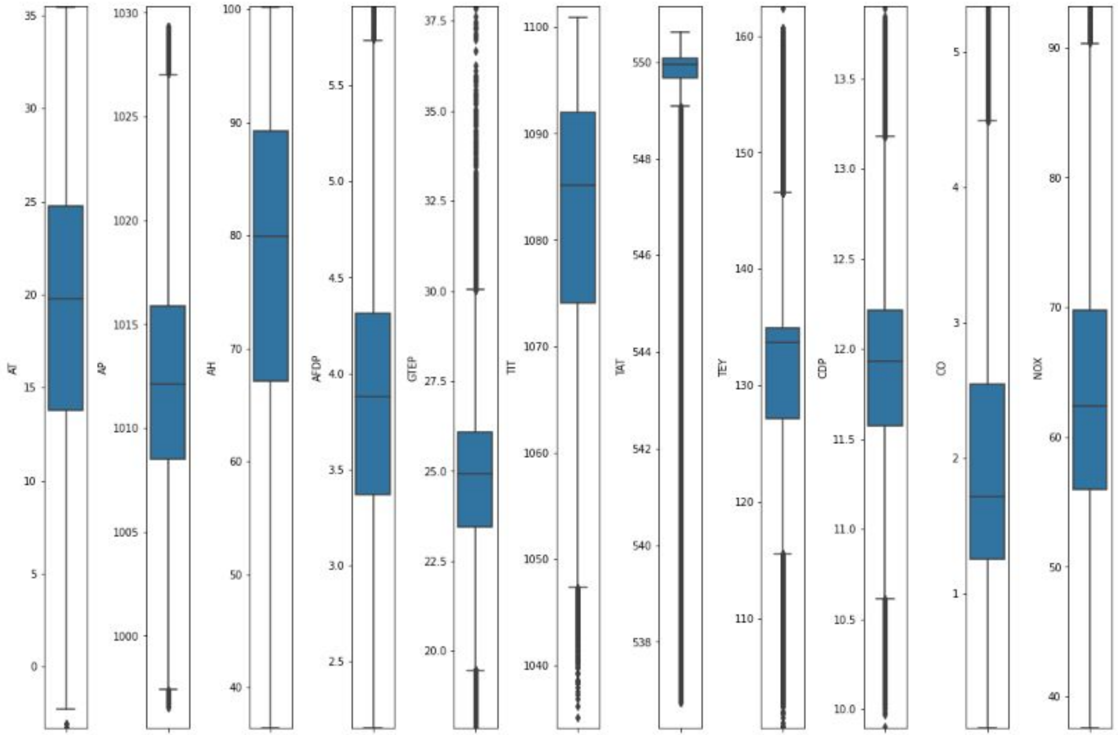
Box plots

```
def boxplots(dataset):
    columns = list(dataset.columns)
    fig, axes = plt.subplots(1, len(columns), figsize=(15, 10))
    for i, col in enumerate(columns):
        ax = sn.boxplot(y=dataset[col], ax=axes.flatten()[i])
        ax.set_ylim(dataset[col].min()-dataset[col].min()*0.001, dataset[col].max()+dataset[col].max()*0.001)
        ax.set_ylabel(col)
    fig.tight_layout()
    plt.show()
boxplots(dataset)
```



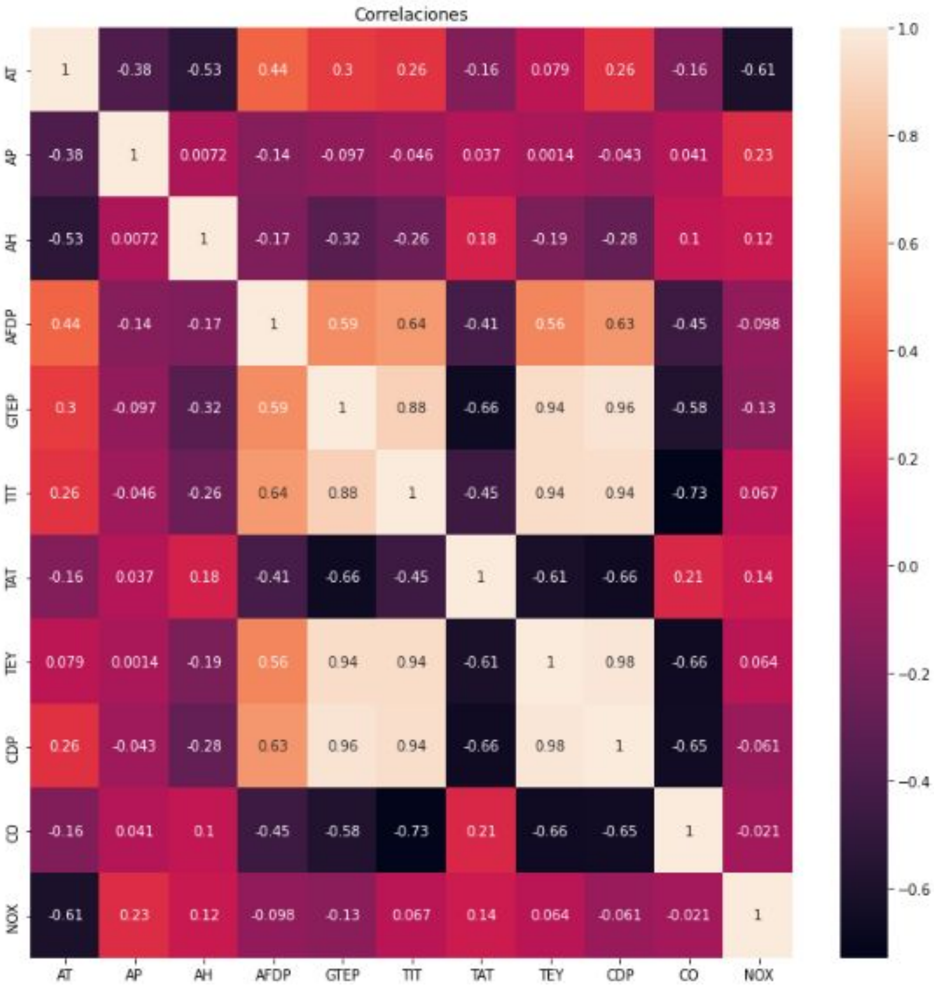
Removiendo outliers por rango intercuartil

```python
dataset_sin_anomalias = dataset
atributos = list(dataset_sin_anomalias.columns.values)
for i in atributos:
    if(i == "TEY"):
        continue
    q1 = dataset[i].quantile(0.25)
    q3  = dataset[i].quantile(0.75)
    qr = q3-q1
    q3 = q3+1.5*qr
    q1 = q1-1.5*qr
    dataset_sin_anomalias = dataset_sin_anomalias[(dataset_sin_anomalias[i] < q3) & (dataset_sin_anomalias[i] > q1)]
boxplots(dataset_sin_anomalias)
dataset = dataset_sin_anomalias
```



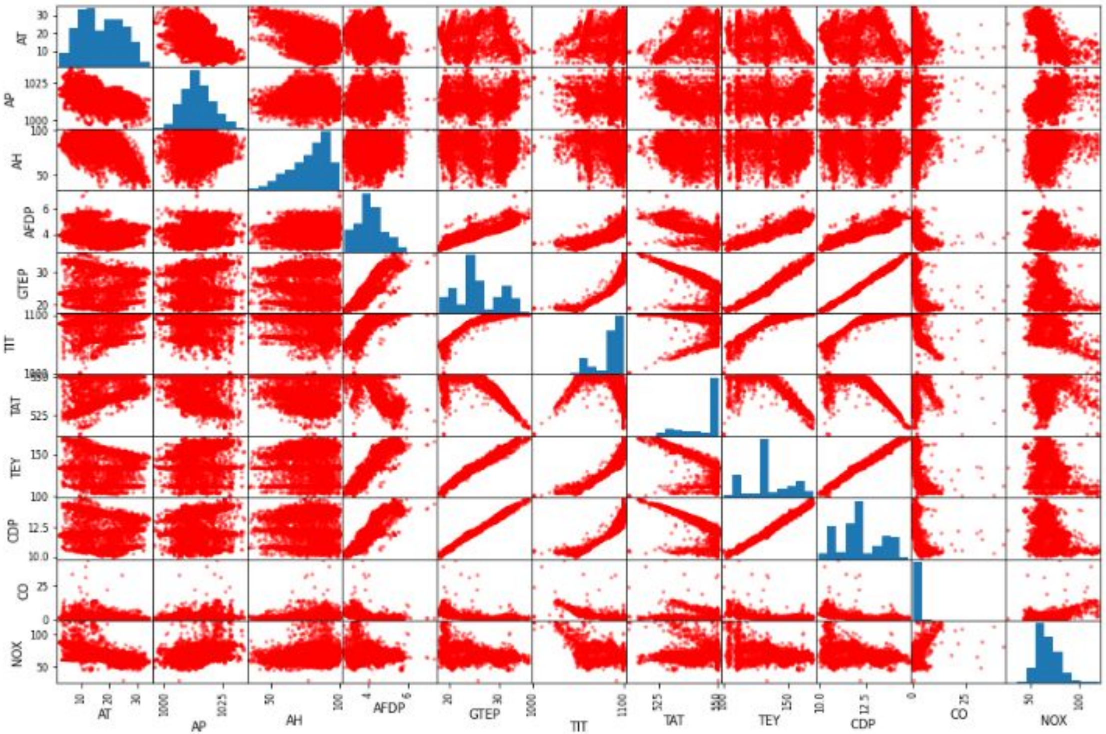
Correlaciones entre atributos

```python
corr_matrix = dataset.corr()
fig = plt.figure(figsize=(10,10))
ax = plt.axes()
sn.heatmap(corr_matrix,annot = True)
ax.set_title('Correlaciones')
fig.tight_layout()
plt.show()
```



Pairplot o scatter matrix

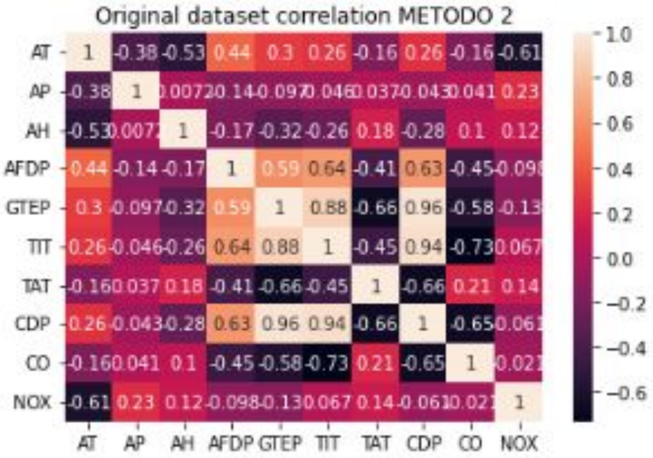
```
pd.plotting.scatter_matrix(dataset_2011, figsize = (15,10), color = 'Red')
plt.show()
```



```python
def corr_coef(dataset, threshold):
    corr_matrix = dataset.corr()
    ax = plt.axes()
    sn.heatmap(corr_matrix,annot = True)
    ax.set_title('Original dataset correlation METODO 2')
    plt.show()
    corr_matrix = features.corr().abs()
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
    to_drop = [column for column in upper.columns if any(upper[column] > threshold)]
    return to_drop
features = dataset.drop('TEY', axis = 1)
to_drop = corr_coef(features, 0.95)
print("Correlacion propone borrar:",to_drop)
```

Feature selection

Se escoge CDP ya que está altamente correlacionado con GTEP (96%), por lo que se puede considerar redundante




```
Correlacion propone borrar: ['CDP']
```

Feature selection

```
dataset_procesado = dataset.drop(to_drop, axis = 1)
print(dataset_procesado)
```

```
           AT      AP      AH    AFDP    GTEP     TIT     TAT     TEY
0      4.5878  1018.7  83.675  3.5758  23.979  1086.2  549.83  134.67
1      4.2932  1018.3  84.235  3.5709  23.951  1086.1  550.05  134.67
2      3.9045  1018.4  84.858  3.5828  23.990  1086.5  550.19  135.10
3      3.7436  1018.3  85.434  3.5808  23.911  1086.5  550.17  135.03
4      3.7516  1017.8  85.182  3.5781  23.917  1085.9  550.00  134.67
...       ...     ...     ...     ...     ...     ...     ...     ...
36707  2.8040  1028.5  85.691  3.3807  22.541  1072.4  549.83  127.91
36708  2.4584  1028.6  85.003  3.3503  22.643  1073.1  549.81  128.65
36709  2.6604  1028.7  85.115  3.8577  26.630  1085.2  543.74  143.26
36726  3.4218  1028.7  91.003  3.6911  22.859  1073.5  549.78  129.86
36731  5.8837  1028.7  94.200  3.9831  23.563  1076.9  550.11  131.41

            CO     NOX
0      0.32663  81.952
1      0.44784  82.377
2      0.45144  83.776
3      0.23107  82.505
4      0.26747  82.028
...        ...     ...
36707  3.54290  68.581
36708  3.64270  68.059
36709  3.45260  62.330
36726  3.67380  67.737
36731  3.31340  64.738

[28503 rows x 10 columns]
```

Correlación de atributos con target

```
dataset_procesado.corr()["TEY"].sort_values(ascending=False)
```

```
TEY     1.000000
TIT     0.938548
GTEP    0.935239
AFDP    0.561846
AT      0.078913
NOX     0.063745
AP      0.001396
AH     -0.188136
TAT    -0.610357
CO     -0.661107
Name: TEY, dtype: float64
```

Proceso de regresión

```
copy = dataset_procesado.copy()
copy.pop('TEY')
labels = dataset_procesado.columns
X = dataset_procesado.loc[:, copy.columns].values
y = dataset_procesado.loc[:, 'TEY'].values
print("Features:\n", X)
print("Target:\n", y)
```

```
Features:
 [[4.5878e+00 1.0187e+03 8.3675e+01 ... 5.4983e+02 3.2663e-01 8.1952e+01]
 [4.2932e+00 1.0183e+03 8.4235e+01 ... 5.5005e+02 4.4784e-01 8.2377e+01]
 [3.9045e+00 1.0184e+03 8.4858e+01 ... 5.5019e+02 4.5144e-01 8.3776e+01]
 ...
 [2.6604e+00 1.0287e+03 8.5115e+01 ... 5.4374e+02 3.4526e+00 6.2330e+01]
 [3.4218e+00 1.0287e+03 9.1003e+01 ... 5.4978e+02 3.6738e+00 6.7737e+01]
 [5.8837e+00 1.0287e+03 9.4200e+01 ... 5.5011e+02 3.3134e+00 6.4738e+01]]
Target:
 [134.67 134.67 135.1  ... 143.26 129.86 131.41]
```

Obteniendo los mejores regresores

```
split_methods = ['holdout', 'random_subsampling', 'kfold', 'kfold']
regression_methods = ['linear', 'decisiontree', 'kneighbors', 'sgd', 'randomforest', 'mlp']
```

_______________linear_______________

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.992	0.985	23141.7	1.072	2.03	1.425
random_subsampling	0.992	0.985	5782.8	1.076	2.028	1.424
kfold	0.992	0.985	11595.5	1.077	2.034	1.426
kfold	0.992	0.985	5799.7	1.077	2.035	1.426

_______________decisiontree_______________

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.996	0.993	11077.1	0.629	0.972	0.986
random_subsampling	0.997	0.994	2326.36	0.583	0.816	0.903
kfold	0.997	0.993	4961.64	0.593	0.87	0.933
kfold	0.997	0.994	2280.02	0.577	0.8	0.894

_______________kneighbors_______________

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.993	0.987	19664.5	0.845	1.725	1.313
random_subsampling	0.994	0.989	4295	0.776	1.506	1.227
kfold	0.994	0.988	8961.49	0.793	1.572	1.254
kfold	0.994	0.989	4265.85	0.776	1.497	1.223

sgd

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	-0.799	-2.77788e+26	4.24853e+32	1.92637e+14	3.72613e+28	1.93032e+14
random_subsampling	0.249	-1.15575e+27	4.34705e+32	3.4904e+14	1.52475e+29	3.49615e+14
kfold	-0.224	-1.52722e+27	1.15415e+33	4.37376e+14	2.02461e+29	4.37763e+14
kfold	0.246	-1.11835e+27	4.23095e+32	3.1194e+14	1.48448e+29	3.12954e+14

randomforest

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.998	0.996	5969.35	0.486	0.524	0.724
random_subsampling	0.998	0.996	1353.85	0.453	0.475	0.689
kfold	0.998	0.996	2915.48	0.469	0.511	0.715
kfold	0.998	0.996	1394.31	0.457	0.489	0.699

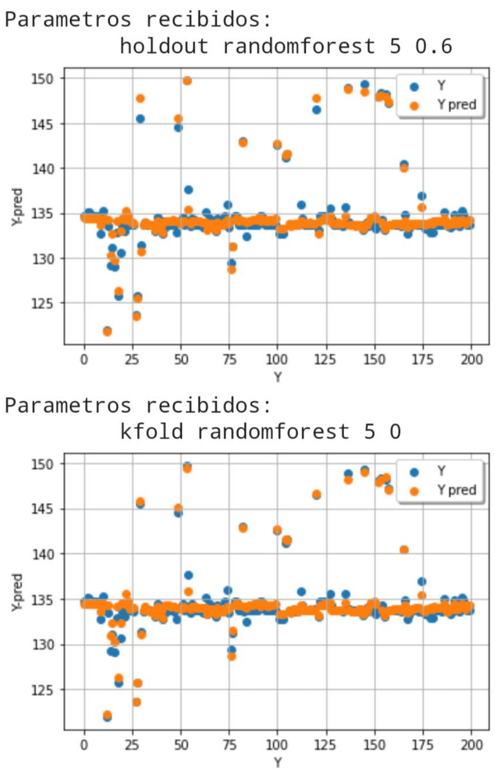
*random forest con n_estimators = 5

_______________mlp_______________

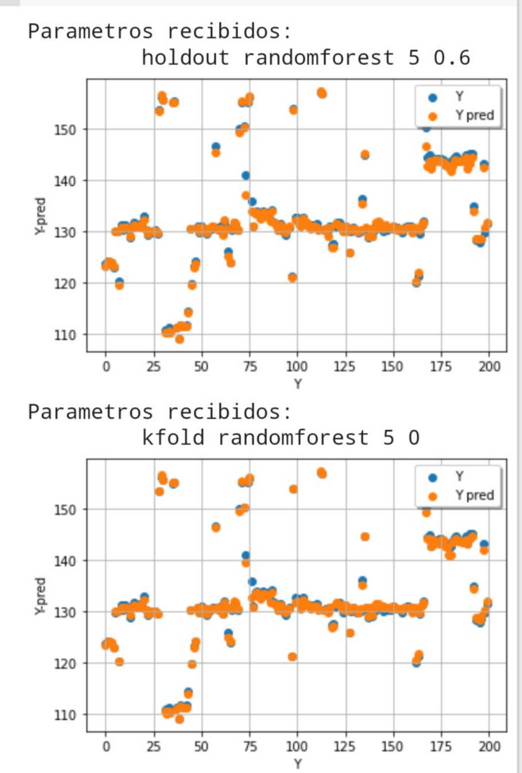
SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.99	0.981	28935.4	1.158	2.538	1.593
random_subsampling	0.991	0.975	9517.41	1.342	3.338	1.791
kfold	0.988	0.931	52918	2.311	9.283	2.837
kfold	0.992	0.973	10150.5	1.449	3.561	1.861

['random_subsampling', 'randomforest']

Y vs Y pred



Parametros recibidos:
holdout randomforest 5 0.6
Y
Y pred
Y-pred
Y
Parametros recibidos:
kfold randomforest 5 0
Y
Y pred
Y-pred
Y



Parametros recibidos:
holdout randomforest 5 0.6
Y
Y pred
Y-pred
Y
Parametros recibidos:
kfold randomforest 5 0
Y
Y pred
Y-pred
Y

Haciendo fine tunning

El método que dió mejores resultados es Random Forest, por lo que se va a buscar la mejor configuración con Random Search.

```
from sklearn.model_selection import RandomizedSearchCV
import random
X_train,X_test,y_train,y_test = train_test_split(X, y, train_size=0.6)

params = {
    "n_estimators": range(5, 20),
    "max_depth": range(0, 15),
    "min_samples_split": range(0, 20),
    "min_samples_leaf": range(0, 5),
    #"criterion": ['squared_error', 'absolute_error', 'poisson'],
    #"max_features": ['sqrt', 'log2'],
}
randomizedCV = RandomizedSearchCV(estimator=RandomForestRegressor(), param_distributions =params, verbose=2, cv = 5, n_iter = 5, n_jobs=-1)
randomizedCV.fit(X_train, y_train)
while(randomizedCV.best_score_ < 0.996):
    randomizedCV.fit(X_train, y_train)
    print(randomizedCV.best_score_)
randomizedCV.best_params_
```

Mejor configuración

Simplemente se podría aumentar la cantidad de árboles pero el tiempo de entrenamiento sería muy alto

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
Fitting 5 folds for each of 5 candidates, totalling 25 fits
0.9962057732133347
{'n_estimators': 47,
 'min_samples_split': 4,
 'min_samples_leaf': 1,
 'max_depth': 14}
```

randomforest

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.998	0.996	6027.77	0.493	0.529	0.727
random_subsampling	0.998	0.997	1320.19	0.463	0.463	0.68
kfold	0.998	0.996	2695.24	0.466	0.473	0.688
kfold	0.998	0.997	1317.42	0.462	0.462	0.679

¿Es buena idea usar Random Forest?

Viendo los buenos resultados de los regresores, me iría por aquel que tiene un comportamiento más sencillo y que tenga un tiempo de ejecución menor. Con esto me iría con el regresor lineal ya que estamos hablando de una diferencia de tiempo bastante grande respecto a Random Forest sin bajar de 0.99 en R y 0.98 en R2.

_______________linear_______________

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.993	0.985	22598.3	1.07	1.982	1.408
random_subsampling	0.992	0.985	5769.27	1.074	2.024	1.422
kfold	0.992	0.985	11602.2	1.076	2.035	1.427
kfold	0.992	0.985	5794.31	1.076	2.033	1.426

Tiempo de ejecucion: 3.195369005203247 segundos

_______________randomforest_______________

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.998	0.997	5253.56	0.462	0.461	0.679
random_subsampling	0.998	0.997	1203.61	0.442	0.422	0.649
kfold	0.998	0.997	2462.8	0.448	0.432	0.657
kfold	0.998	0.997	1188.63	0.44	0.417	0.645

Tiempo de ejecucion: 597.2100353240967 segundos

Referencias

Kaya, H., Tüfekci, P., & Uzun, E. (2019). Predicting CO and NOxemissions from gas turbines: novel data and abenchmark PEMS. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, *27*(6), 4783–4796. https://www.semanticscholar.org/paper/Predicting-CO-and-NOxemissions-from-gas-turbines%3A-Kaya-Tüfekçi/260c33c3a77a83fe4c15cc95800d4890d1b64985/figure/1

Sklearn.Model_selection.RandomizedSearchCV. (s/f). Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

UCI machine learning repository: Gas turbine CO and NOx emission data set data set. (s/f). Uci.edu. https://archive.ics.uci.edu/ml/datasets/Gas+Turbine+CO+and+NOx+Emission+Data+Set

Notebook colab:
https://colab.research.google.com/drive/1rvUb9aNBHDnKsynbaOFAMFK6wlorYYVN?usp=sharing