Regresión: Decision Tree y SSE

Minería de Datos

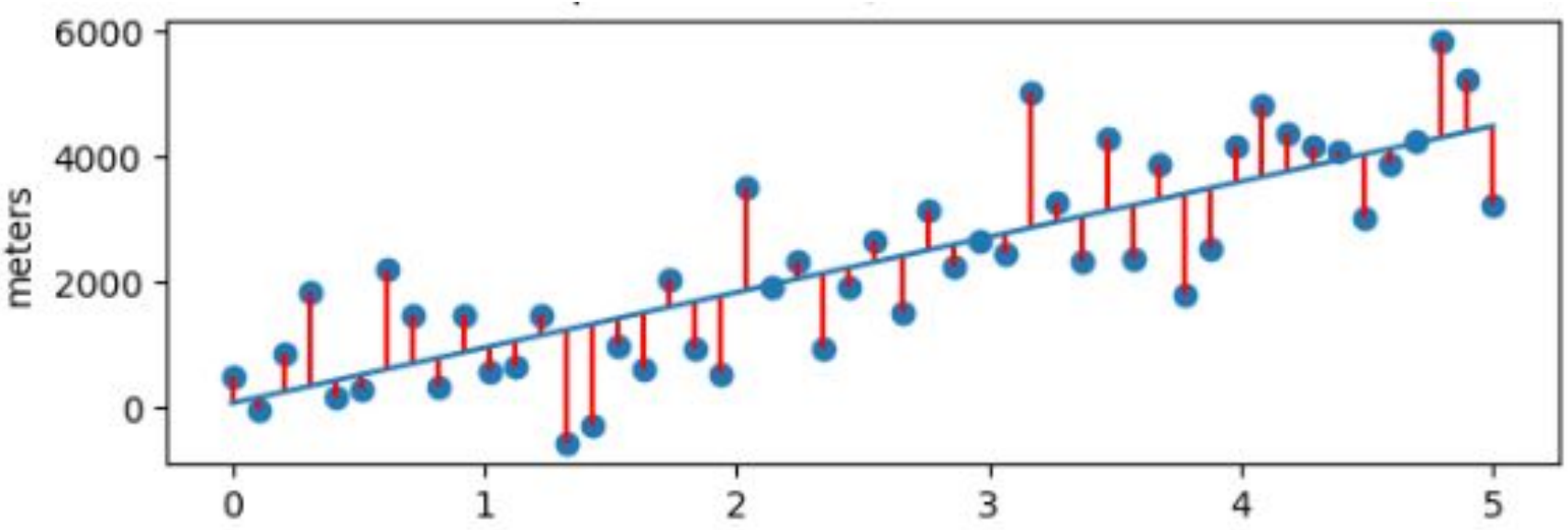
Zavala Roman Irvin Eduardo 1270771

SSE (Sum of Squared Errors)

Este índice representa la diferencia entre los valores reales y los valores predecidos, donde el mejor valor es 0. Si el error es menor significa que el modelo representa de manera precisa el comportamiento del fenómeno observado.

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

SSE (Sum of Squared Errors)



SSE (Sum of Squared Errors)

El SSE no existe en sklearn pero es fácil calcularlo con numpy.

```
sse = np.sum((y_test - y_pred)**2)
```

O se podría hacer un for sumando el cuadrado de las diferencias.

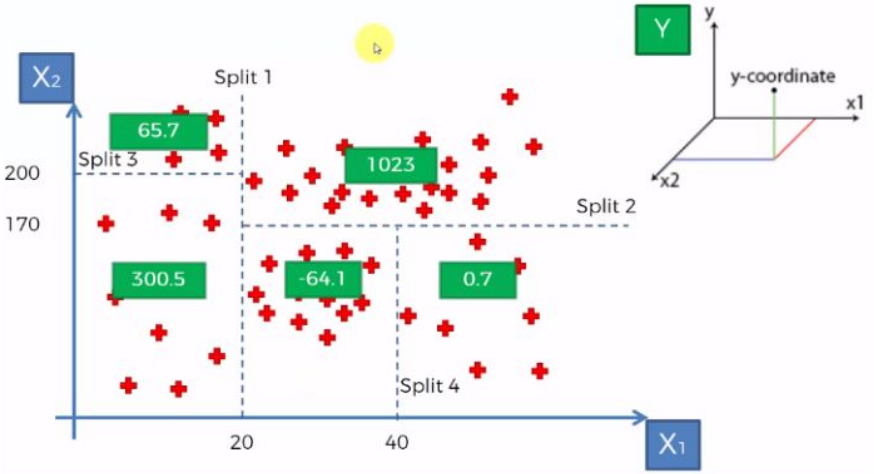
Observaciones

- El cuadrado hace que las diferencias que den negativo se hacen positivas.
- Al no tener un rango, SSE puede variar el valor “normal” respecto a dataset.
- Si el modelo se aleja bastante del fenómeno, gracias al cuadrado se nota bastante la diferencia
- Por sí solo no dice mucho
- Se suele calcular para calcular R2 que si tiene un rango delimitado.

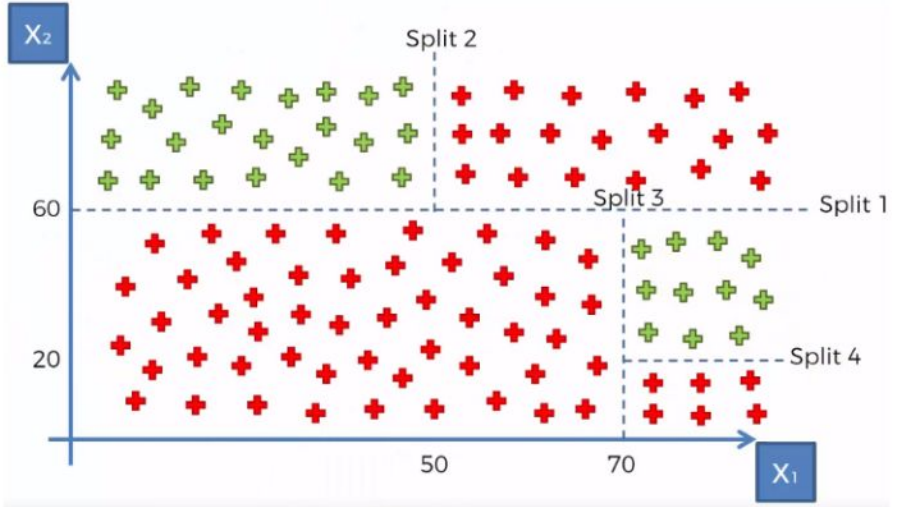
Decision Tree Regressor (sklearn.tree.DecisionTreeRegressor)

Simplemente es una adaptación del clasificador por árboles a datos continuos. La función tiene los mismos parámetros que el clasificador.

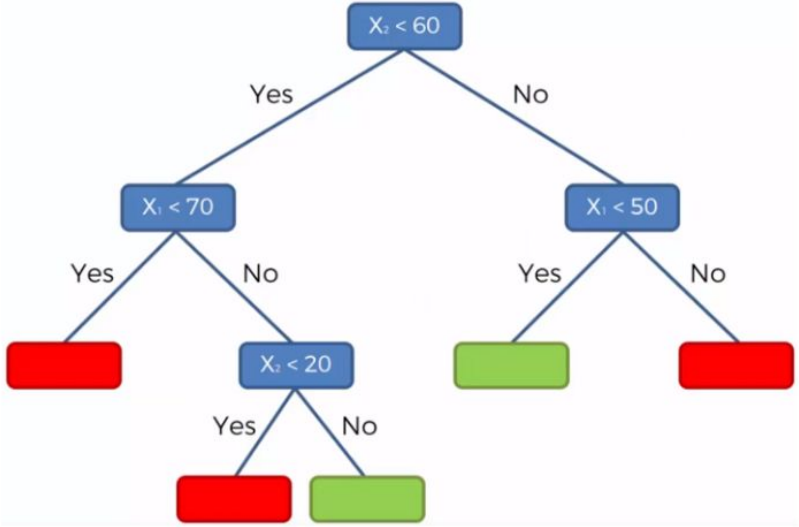
Decision Tree Regression:



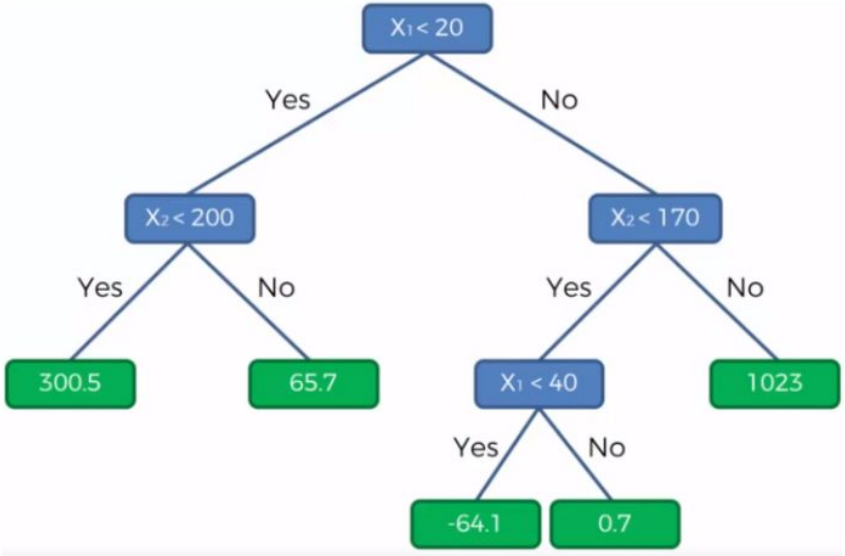
Decision Tree Classifications:



Decision Tree Regressor



Clasificación

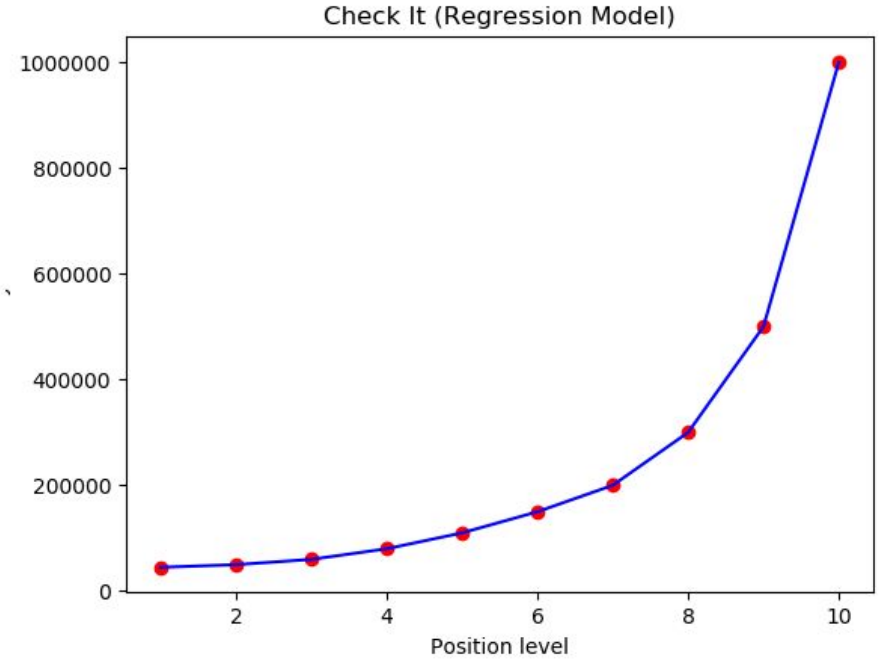


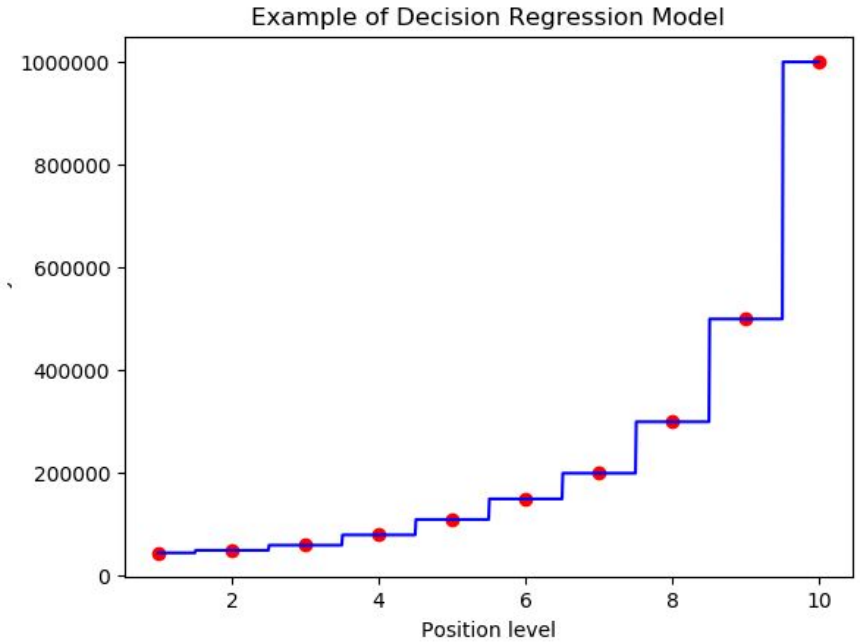
Regresión

Decision Tree Regressor (Código)

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
import pandas as pd
dataset = pd.read_csv("dataset.csv", on_bad_lines='skip')
labels = dataset.columns
X = dataset.loc[:, labels[:-1]].values
y = dataset.loc[:, labels[-1]].values
X_train,X_test,y_train,y_test = train_test_split(X, y, train_size=0.6)
model = DecisionTreeRegressor()
y_pred = model.fit(X_train, y_train).predict(X_test)
```

Decision Tree Regressor





Decision Tree Regressor (Observaciones)

- Las mismas ventajas que en clasificación
- Soporta multi-output
- Datos faltantes no afecta el árbol
- El entrenamiento puede ser tardado
- No se recomienda con valores continuos por la forma recursiva en la que se crea el árbol, haciendo overfitting

Datasets utilizados

- China (1 entrada | 1 salida)

  Relaciona años y un valor monetario

- Energy Eficciency (8 entradas | 2 salidas)

  Busca estudiar la eficiencia energética de construcciones

- scm20d(61 entradas | 16 salidas)

  Supply Chain Management, muy complicado pero se puede consultar:

  https://www.openml.org/search?type=data&sort=runs&id=41404

Metodos utilizados

Regresión:

- Lineal
- DecisionTree
- KNeighbors

Métricas: R, R^2, SSE, MAE, MSE, RMSE

Split methods: Holdout, random subsampling y kfold

Objeto creado

```
class regressor():
    """
    Clase para hacer regresiones
    Soporta los splitters: [holdout,random_subsampling,kfold]
    Soporta los regresores: [linear, decisiontree, kneighbors]

    regressor(self, X, y, split_method: str, regressor: str, k=None, train_size=None)

    Se tienen los metodos:
        regression: Hace el fit y predict
        getMetrics: Retorna r,r2,sse,mae,mse,rmse
    """
```

```
dataset = pd.read_csv("scm20d.csv", on_bad_lines='skip') #2 output dataset
labels = dataset.columns
X = dataset.loc[:, labels[1:-16]].values
y = dataset.loc[:, labels[-16:]].values
best_row = [0,0,0,0,0,0]
best_combination = ['split', 'regressor']
for regression_method in regression_methods:
    i = 0
    table = []
    head_table = ["SPLIT METHOD","r","r2", "sse", "mae","mse","rmse"]
    table.append(head_table)
    for split_method in split_methods:
        if(i == 0):
            rObj = regressor(X,y,split_method,regression_method,5, 0.6)
        if(i == 1):
            rObj = regressor(X,y,split_method,regression_method,30, 0)
        if(i == 2):
            rObj = regressor(X,y,split_method,regression_method,5, 0)
        if(i == 3):
            rObj = regressor(X,y,split_method,regression_method,10, 0)
        i+=1
        rObj.regression()
        r, r2, sse, mae, mse, rmse = rObj.getMetrics()
        row = [split_method,float("{:.3f}".format(r)),
               float("{:.3f}".format(r2)), float("{:.3f}".format(sse)),
               float("{:.3f}".format(mae)), float("{:.3f}".format(mse)), float("{:.3f}".format(rmse)) ]
        from statistics import mean
        import warnings
        warnings.filterwarnings('ignore')
        if(split_method == 'holdout'):
            best_row = row
        elif(mean([row[1],row[2]]) > mean([best_row[1],best_row[2]])and
             mean([row[3],row[4],row[5],row[6]]) < mean([best_row[3],best_row[4],best_row[5],best_row[6]])):
            best_row = row
            best_combination = [split_method, regression_method]
        table.append(row)
    print("                {}                ".format(regression_method))
    print(tabulate(table, headers='firstrow', tablefmt='fancy_grid'))
print(best_combination)
```

Dataset 1 salida

_____linear_____

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	1	0.437	6.07269e+25	1.34583e+12	2.76032e+24	1.66142e+12
random_subsampling	1	-9.153	1.99885e+25	1.35936e+12	3.33141e+24	1.70011e+12
kfold	1	-0.121	3.72116e+25	1.39914e+12	3.38287e+24	1.79232e+12
kfold	1	-128.26	1.82102e+25	1.40222e+12	3.44207e+24	1.70217e+12

_____decisiontree_____

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.72	0.956	6.96766e+24	3.07564e+11	3.16712e+23	5.62771e+11
random_subsampling	0.863	0.959	7.03012e+23	1.6e+11	1.17169e+23	2.80282e+11
kfold	0.827	0.952	2.48363e+24	2.24151e+11	2.25784e+23	4.01878e+11
kfold	0.851	0.956	8.37061e+23	1.95682e+11	1.56464e+23	3.15644e+11

_____kneighbors_____

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.753	0.918	9.45837e+24	1.98103e+11	4.29926e+23	6.55687e+11
random_subsampling	0.85	0.904	3.61907e+24	3.09972e+11	6.03178e+23	5.78165e+11
kfold	0.805	0.953	4.23738e+24	1.96449e+11	3.85217e+23	4.74558e+11
kfold	0.866	0.963	2.8456e+24	2.10321e+11	4.80832e+23	3.57784e+11

Dataset 1 salida: Mejor combinación

```
['kfold', 'kneighbors']
```

kneighbors

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.753	0.918	9.45837e+24	1.98103e+11	4.29926e+23	6.55687e+11
random_subsampling	0.85	0.904	3.61907e+24	3.09972e+11	6.03178e+23	5.78165e+11
kfold	0.805	0.953	4.23738e+24	1.96449e+11	3.85217e+23	4.74558e+11
kfold	0.866	0.963	2.8456e+24	2.10321e+11	4.80832e+23	3.57784e+11

Dataset 2 salidas

linear

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.105	0.895	6099.49	4.41	19.804	4.45
random_subsampling	0.099	0.897	1495.71	4.421	19.425	4.391
kfold	0.098	0.898	2952.98	4.368	19.23	4.38
kfold	0.096	0.897	1478.46	4.382	19.25	4.383

decisiontree

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.115	0.961	2212.51	1.784	7.183	2.68
random_subsampling	0.097	0.968	444.247	1.586	5.769	2.373
kfold	0.092	0.974	738.482	1.47	4.808	2.181
kfold	0.093	0.969	429.054	1.54	5.584	2.353

kneighbors

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.066	0.938	3609.67	3.541	11.72	3.423
random_subsampling	0.076	0.961	583.619	2.805	7.579	2.733
kfold	0.069	0.956	1306.11	2.988	8.505	2.914
kfold	0.069	0.959	596.492	2.817	7.772	2.778

Dataset 2 salidas: Mejor combinación

```
['random_subsampling', 'kneighbors']
```

kneighbors

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.066	0.938	3609.67	3.541	11.72	3.423
random_subsampling	0.076	0.961	583.619	2.805	7.579	2.733
kfold	0.069	0.956	1306.11	2.988	8.505	2.914
kfold	0.069	0.959	596.492	2.817	7.772	2.778

Dataset 16 salidas

linear

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.29	0.586	1.674e+09	2020.91	466685	683.144
random_subsampling	0.3	0.586	4.23947e+08	2028.78	472627	687.348
kfold	0.297	0.585	8.44496e+08	2024.32	470942	686.079
kfold	0.297	0.585	4.21977e+08	2023.77	470637	685.964

decisiontree

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.217	0.61	1.60106e+09	1557.28	446351	668.095
random_subsampling	0.228	0.693	3.09842e+08	1342.19	345420	587.591
kfold	0.228	0.68	6.49951e+08	1390.07	362455	602.006
kfold	0.229	0.693	3.1111e+08	1347.29	346988	589.013

kneighbors

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.243	0.833	6.77737e+08	1057.91	188943	434.675
random_subsampling	0.244	0.867	1.35113e+08	901.095	150627	387.721
kfold	0.243	0.858	2.85686e+08	936.059	159316	399.133
kfold	0.242	0.869	1.32809e+08	895.141	148126	384.147

Dataset 16 salidas: Mejor combinación

['random_subsampling', 'kneighbors']

kneighbors

SPLIT METHOD	r	r2	sse	mae	mse	rmse
holdout	0.243	0.833	6.77737e+08	1057.91	188943	434.675
random_subsampling	0.244	0.867	1.35113e+08	901.095	150627	387.721
kfold	0.243	0.858	2.85686e+08	936.059	159316	399.133
kfold	0.242	0.869	1.32809e+08	895.141	148126	384.147

Referencias

Error sum of squares. (s/f). Stanford.edu. https://hlab.stanford.edu/brian/error_sum_of_squares.html

Girgin, S. (2019). *Decision tree regression in 6 steps with python*. PursuitData. https://medium.com/pursuitnotes/decision-tree-regression-in-6-steps-with-python-1a1c5aa2ee16

Sklearn.tree.decisionTreeRegressor. (s/f). Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

Valchanov, I. (2018). *Sum of squares total, sum of squares regression and sum of squares error*. 365 Data Science. https://365datascience.com/tutorials/statistics-tutorials/sum-squares/

Zach. (2021). *A gentle guide to sum of squares: SST, SSR, SSE*. Statology. https://www.statology.org/sst-ssr-sse/