# XGBoost: Feature importance y aplicaciones

La feature importance puede decirte cuanto tus features (variables de entrada) están contribuyendo al modelo, o el nivel de impacto en el target.

Los niveles de feature importance pueden ser descrito como una serie de valores correspondientes a los features usados para ajustas el modelo. Mientras más grande el número, más importante.

Todas las implementaciones de XGBoost (API & native booster) proveen de métodos para graficar los feature importance.

Los valores del feature importance pueden ser calculados de diferentes maneras dependiendo en las mediciones del tree split (por lo que no es determinista), las maneras más populares son ‘gain’ y ‘weight’.

Los niveles de feature importance para XGBoost pueden ser usados, generalmente, para:

- Entender y explicar los key drivers del modelo,
- Feature selection para un re-modelado.

Ejemplos sobre como graficarlos:

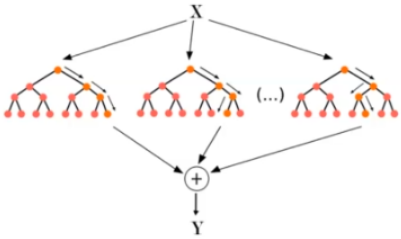
# Xgboost feature importance and application

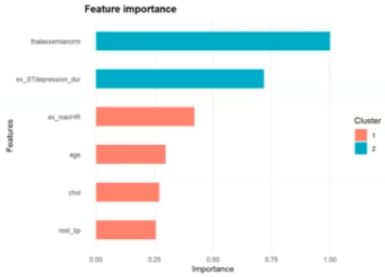
**XGBoostClassifier**

```
# by gain
sorted_idx = np.argsort(model.feature_importances_)[::-1]
for index in sorted_idx:
   print([X_train.columns[index], model.feature_importances_[index]])
```

```
imp_gain = clf.get_booster().get_score(importance_type="gain")
imp_weight = clf.get_booster().get_score(importance_type="weight")

plot_importance(model, importance_type='gain')
pyplot.show()
```




Ehemplos graficarlos en booster nativo:

# Xgboost feature importance and application

**XGBoost native booster**

```
imp_gain = model.get_score(importance_type="gain")
imp_weight = model.get_score(importance_type="weight")

important_values = list(imp_gain.values())
important_vars = list(imp_gain.keys())
sorted_idx = np.argsort(important_values)[::-1]
important_var_gain = [(important_vars[index], important_values[index]) for index in sorted_idx]

plot_importance(model, importance_type='gain')
pyplot.show()

plot_importance(model, importance_type='weight')
pyplot.show()
```