Adjustment of hyperparameters and parameters p, d, q for demand forecasting for a two weeks horizon

Scikit-learn Algorithms:

Random Forest, Light Gradient Boosting Machine, Decision Tree

Statsmodels Algorithms:

ARIMA

Hyperparameter Tuning for Scikit-Learn Algorithms

Hyperopt (Scikit-learn Algorithms)

- Provides automatic algorithm configuration of the Python scikit-learn machine learning library.
- Its objective is to minimize a function around several evaluations of a model with different combinations of hyperparameters.
- Hyperopt uses the Tree-Structured Parzen Estimator (TPE) search algorithm, which seeks to choose the best performing hyperparameters for the next evaluation step, leaving behind the values that do not infer a good result in the model.

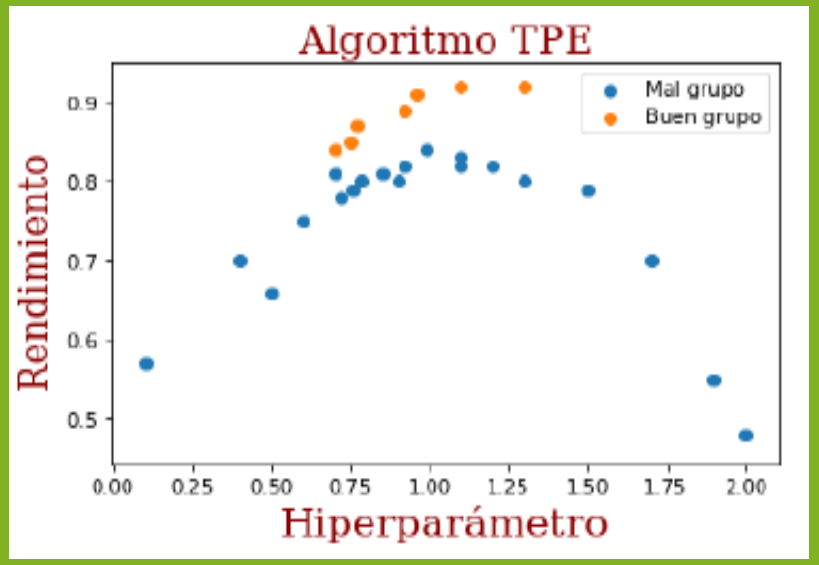
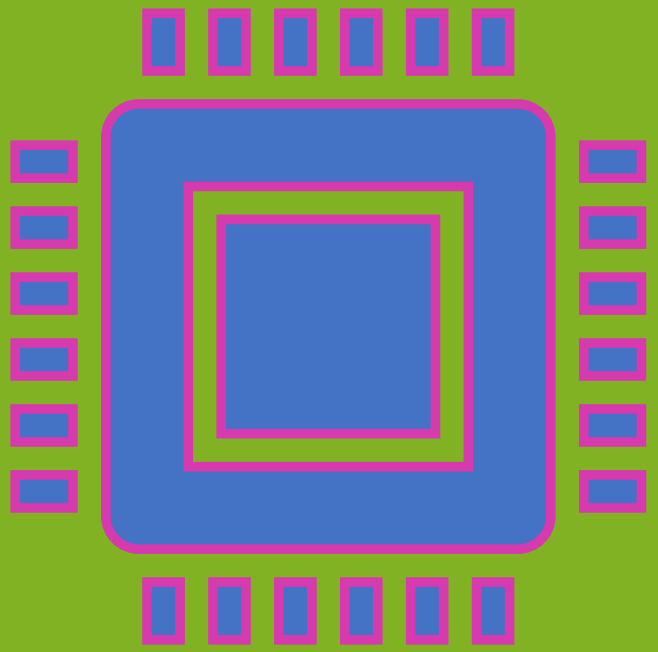


Fig. TPE search algorithm

➢**Stages for defining Hyperopt:**

1. Define a search space or domain
2. Define a function to minimize
3. Choose optimization algorithm
4. Call the function



Python syntax of algorithms Random Forest, LGBM, Decision Tree

Random Forest

RandomForestRegressor(n_estimators=100, *, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap35=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None)

Light Gradient Boosting Machine

LGBMRegressor(boosting_type='gbdt', num_leaves=31, max_depth=1,learning_rate=0.1, n_estimators=100, subsample_for_bin=200000, objective=None, class_weight=None, min_split_gain=0.0, min_child_weight=0.001, min_child_samples=20, subsample=1.0, subsample_freq=0, colsample_bytree=1.0, reg_alpha=0.0, reg_lambda=0.0, random_state=None, n_jobs=- 1, silent=True, importance_type='split')

Decision Tree

DecisionTreeRegressor(*, criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort='deprecated', ccp_alpha=0.0)

Parameter adjustment p, d, q for the ARIMA model

Grid Search

- GridSearch allows a model to be adjusted for all possible combinations of a given list of parameter values provided by the analyst; these combinations build a grid.
-

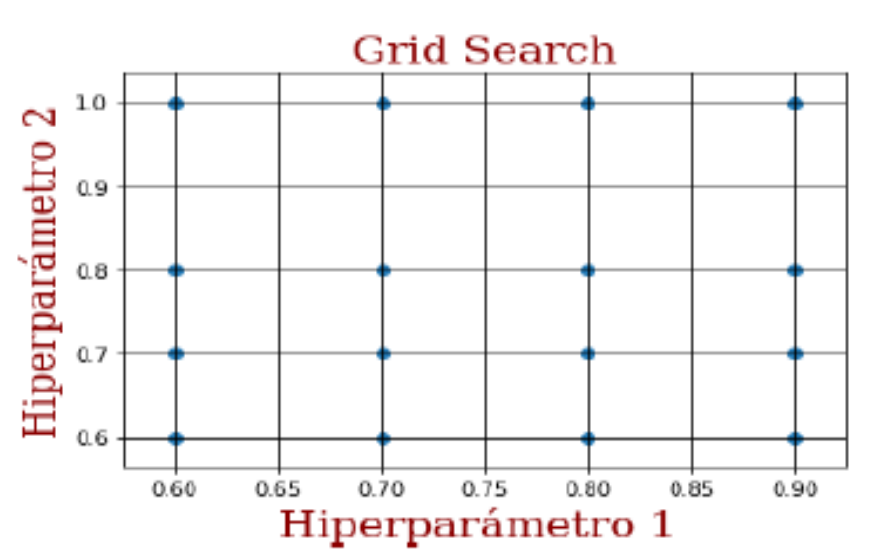


Fig. Búsqueda de Cuadrícula

➢**Stages to define the Grid Search function:**

1. Define a search space or domain
2. Define a function to minimize
3. Call the function

Syntax in Python

ARIMA(Historical Data, order=p, d, q)

Example

- Forecasting of the power demand of the Ecuadorian National Interconnected System from 17-06-2020 to 30-06-2020 (2 weeks).