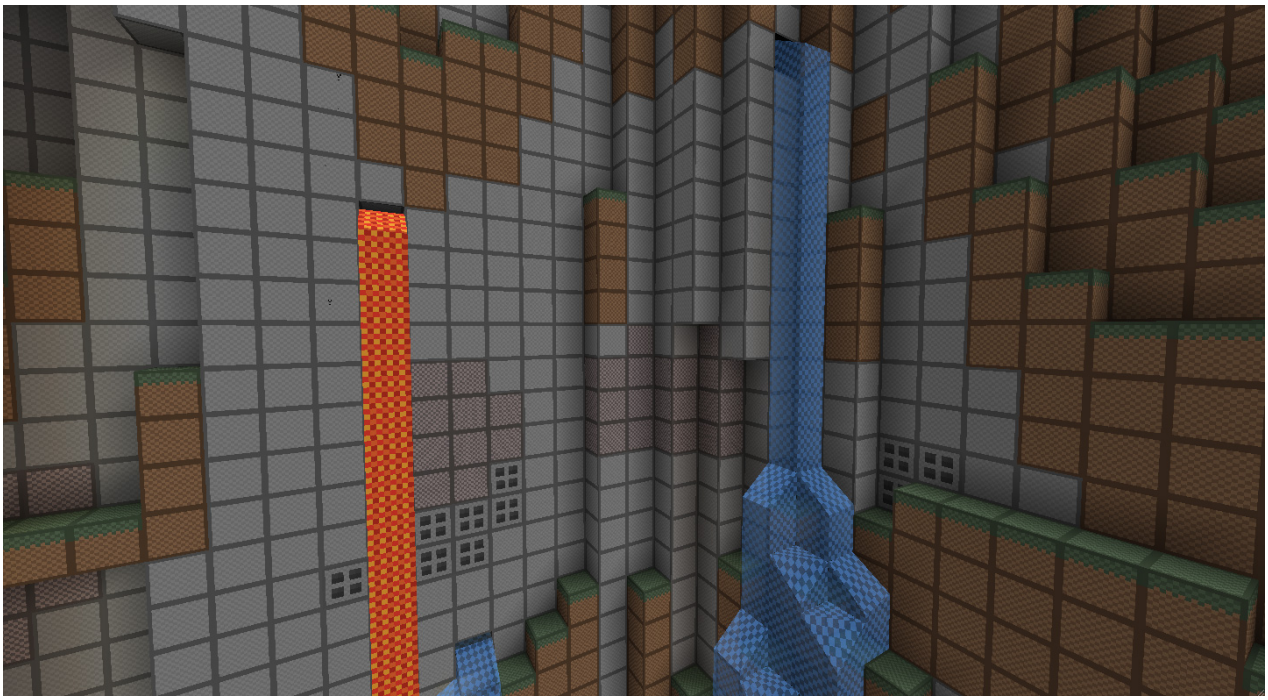

Grid Searching in Machine Learning: Quick Explanation and Python Implementation

Evan Lutins

Grid-searching is the process of scanning the data to configure optimal parameters for a given model. Depending on the type of model utilized, certain parameters are necessary. Grid-searching does NOT only apply to one model type. Grid-searching can be applied across machine learning to calculate the best parameters to use for any given model. It is important to note that Grid-searching can be extremely computationally expensive and may take your machine quite a long time to run. Grid-Search will build a model on each parameter combination possible. It iterates through every parameter combination and stores a model for each combination. Without further ado, lets jump into some examples and implementation.

For the sake of this article I will utilize **Decision Trees** to explain and implement Grid-Searching in Python.

. . .

> *Importing the necessary libraries*

```
# Importing the necessary libraries:
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
```

> *Instantiating the model*

```
# Instantiating the model:
model = DecisionTreeClassifier()
```

> *Establishing which parameters to Grid Search*

```
# Establishing the parameters to grid-search. It is important to note that these parameters change depending on what
# type of model we are building. Be sure to look up the model specific documentation for paramter explanation.
criterion = ['gini', 'entropy']
max_depth = [1, 3, 5, None]
splitter = ['best', 'random']
```

> *Running the Grid Search. This step may take a long time to run depending on the data and number of parameters.*

```
# Setting up the GridSearch. Note how the parameters are wrapped within a dictionary.
grid = GridSearchCV(estimator=model, cv=3, param_grid=dict(criterion=criterion, max_depth= max_depth,
                                                           splitter=splitter))
```

> *Fitting the Grid Search*

```
# Fitting the Grid-Search to our Data
grid.fit(X, y)
```

. . .

Methods to Run on Grid-Search

```
# Depending on the designated paremeter for measuring the model, will print the best score throughout the Grid Search
print grid.best_score_


# Will print out the best parameters used for the highest score of the model.
print grid.best_params_
```