4222-SURYA GROUP OF INSTITUTION

VIKRAVANDI-605 652

NAAN MUDHALVAN PROJECT

# EARTHQUAKE PREDICTION MODEL USING PYTHON

PREPARED BY:

K.RAJ

REGNO:422221106014

DEP:ECE

# AI_PHASE2:

## Consider advanced techniques such as hyperparameter tuning and feature engineering to improve the prediction model's performance.

### INTRODECTION:

Hyperparameters define the configuration or settings of the model. They not learned from the data, but instead we provide them as inputs before training the model. Hyperparameters guide the learning process and impact how the model behaves during training and prediction.

### DATA PREPROCESSING:

Before we can do any feature engineering, we need to preprocess the data to get it in a form suitable for analysis. The data we used in the course was a bit simpler than the competition data. For the Ames competition dataset, we'll need to:

Load the data from CSV files

Clean the data to fix any errors or inconsistencies

Encode the statistical data type (numeric, categorical)

Impute any missing values

### SET OF HYPERPARAMETER:

1.n_estimators = number of trees in the foreset.

2.max_features = max number of features considered for splitting a node.

3.max_depth = max number of levels in each decision tree.

4.min_samples_split = min number of data points placed in a node before the node is split.

GRIDE

SEARCH

A grid is a network of intersecting lines that forms a set of squares or rectangles like the image above. In grid search, each square in a grid has a combination of hyperparameters and the model has to train itself on each combination. For a clearer understanding, suppose that we want to train a Random Forest Classifier with the following set of hyperparameters.

## DATA SOURCE:

| n_estimators | | | |
|---|---|---|---|
| max_depth | (100, 20) | (150, 20) | (200, 20) |
| | (100, 30) | (150, 30) | (200, 30) |
| | (100, 40) | (150, 40) | (200, 40) |

Implementation of Grid Search in Python

Scikit-learn library in Python provides us with an easy way to implement grid search in just a few lines of code. Have a look at the example below

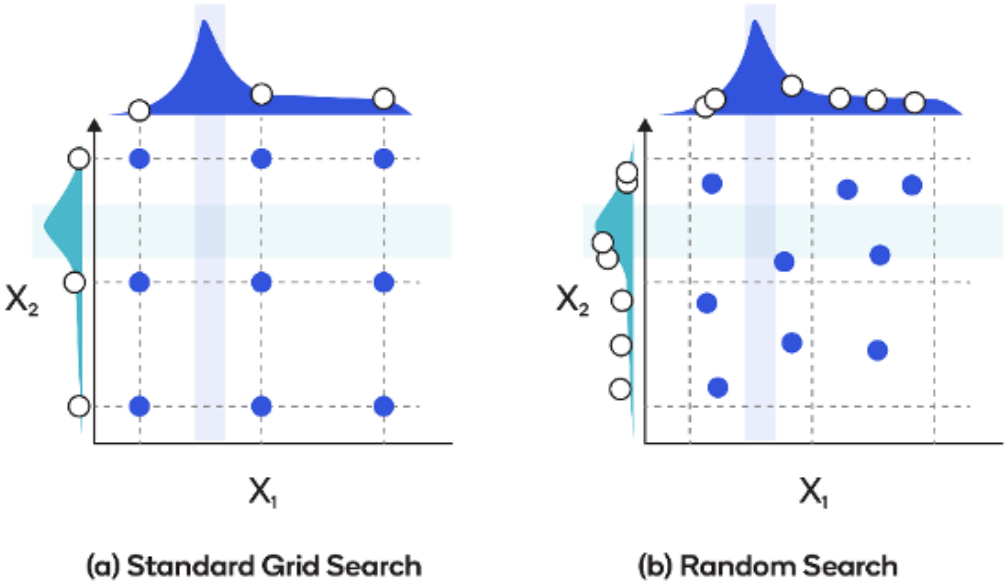
## EXAMPLE:

```
from sklearn.model_selection import GridSearchCV

model = LogisticRegression()

grid_vals = {'penalty': ['l1','l2'], 'C': [0.001,0.01,0.1,1]}

grid_lr = GridSearchCV(estimator=model, param_grid=grid_vals, scoring='accuracy',
                       cv=6, refit=True, return_train_score=True)


#Training and Prediction

grid_lr.fit(X_train, y_train)

preds = grid_lr.best__.predict(X_test)
```

## RANDOM SEARCH:

In grid search, it is required to define the range of values for each hyperparameter that needs to be tuned. In contrast, using a random search the distribution or range of values is defined for each hyperparameter that needs to be tuned. The random search algorithm then samples combinations of

hyperparameter values from these distributions and evaluates the model's performance. By iteratively sampling and evaluating a specified number of combinations, random search aims to identify the hyperparameter settings that yield the best performance.




## RANDOM SEARCH IN PYTHON

```
from sklearn.model_selection import RandomizedSearchCV

model = RandomForestClassifier()

param_vals = {'max_depth': [200, 500, 800, 1100], 'n_estimators': [100, 200, 300, 400],
                    'learning_rate': [0.001, 0.01, 0.1, 1, 10]]}

random_rf = RandomizedSearchCV(estimator=model, param_distributions=param_vals,
                                        n_iter=10, scoring='accuracy', cv=5,
                                        refit=True, n_jobs=-1)

#Training and prediction

random_rf.fit(X_train, y_train)
```

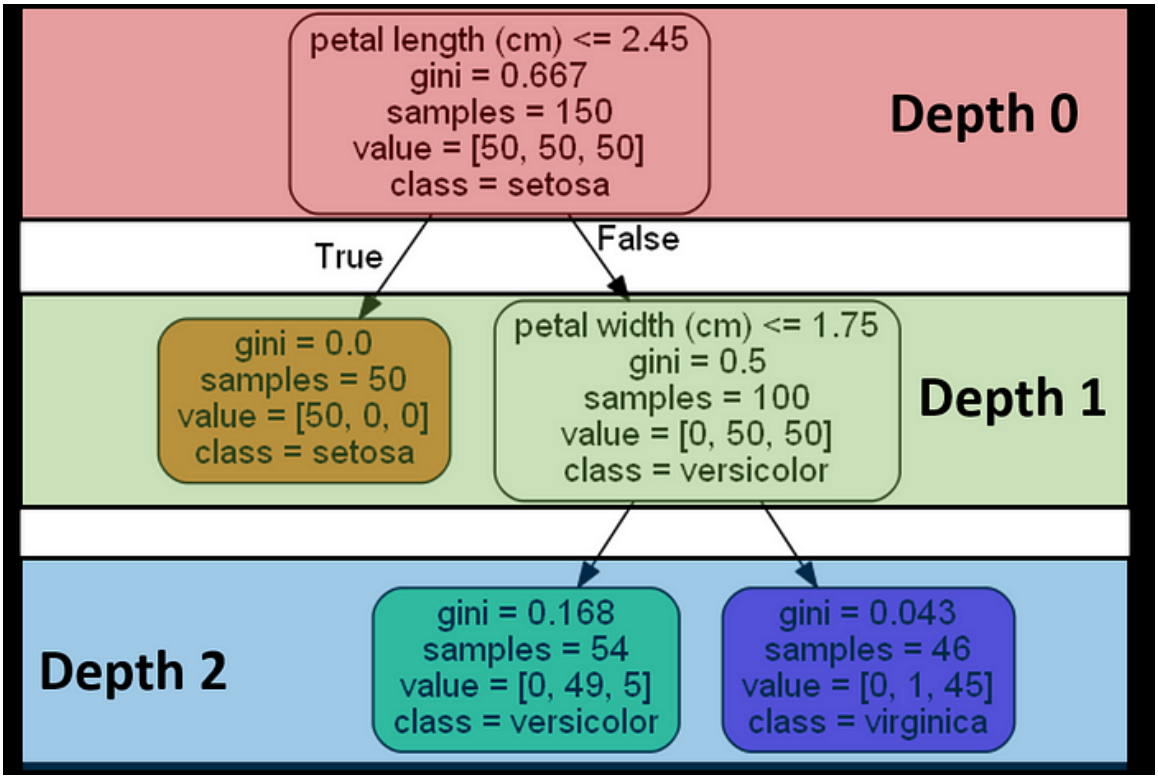
```
preds = random_rf.best_estimator_.predict(X_test)
```

## Hyperparameter Tuning with Grid Search:

Grid search involves exhaustively searching a predefined set of hyperparameter values to find the combination that yields the best performance. A grid search systematically explores the hyperparameter space by creating a grid or a Cartesian product of all possible hyperparameter values and evaluating the model for each combination.

In grid search, it is required to define the range of values for each hyperparameter that needs to be tuned. The grid search algorithm then trains and evaluates the model using all possible combinations of these values. The performance metric, such as accuracy or mean squared error, is used to determine the best set of hyperparameters.

For example, consider a Random Forest classifier. The hyperparameters that can be tuned using grid search include the number of trees in the forest, the maximum depth of each tree, and the minimum number of samples required to split a node. By defining a grid of values for each hyperparameter, such as [100, 200, 300] for the number of trees and [5, 10, 15] for the maximum depth, grid search explores all possible combinations. The model is trained and evaluated for each combination, and the best-performing set of hyperparameters is selected.



## CONCLUSION:

Hyperparameter tuning is a crucial step in developing accurate and robust machine learning models. Manual tuning, grid search, random search, and Bayesian optimization are popular techniques for exploring the hyperparameter space.Each method offers its own advantages and considerations. Machine learning practitioners work to find optimal configurations for a given model. Choosing the appropriate technique depends on factors such as search space size and computational resources.