4222-SURYA GROUP OF INSTITUTION

VIKRAVANDI-605 652

NAAN MUDHALVAN PROJECT

EARTHQUAKE PREDICTION MODEL USING PYTHON

AI_PHASE2:

Consider advanced techniques such as hyperparameter tuning and feature

Engineering to improve the prediction model's performance.

PREPARED BY

S. SUBASRI

REGNO:422221106310

DEP:BE.ECE

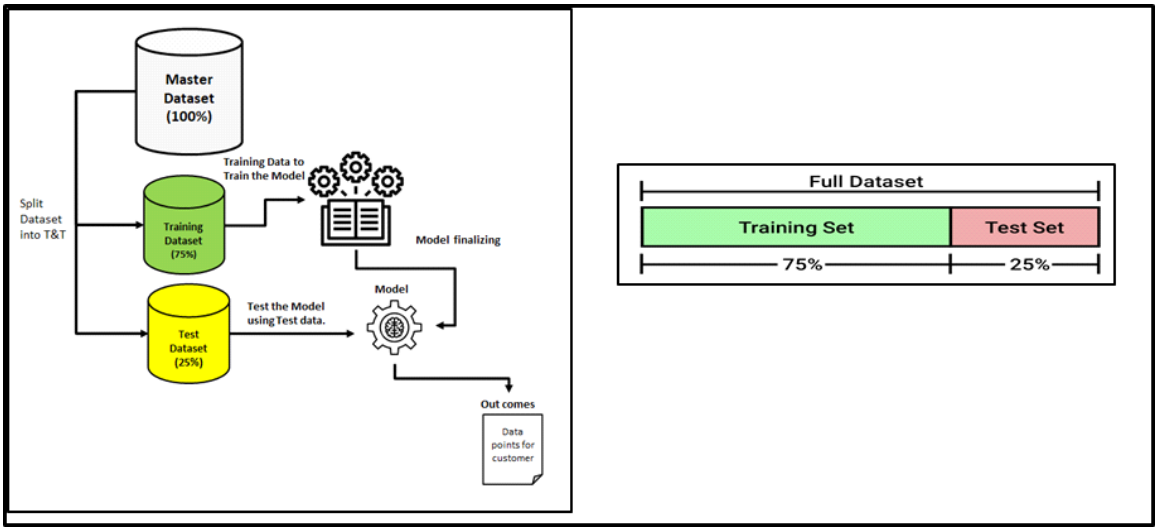
INTRODECTION:

Every ML Engineer and Data Scientist must understand the significance of **"Hyperparameter Tuning.** while selecting your right machine/deep learning model and improving the performance of the model(s).Make it simple, for every single machine learning model selection is a major exercise and it is purely dependent on selecting the equivalent set of hyperparameters, and all these are indispensable to train a model. It is always referring to the parameters of the selected model and be remember it cannot be learnt from the data, and it needs to be provided before the model gets into the training stage, ultimately the performance of the machine learning model improves with a more acceptable choice of hyperparameter tuning and selection techniques. The main intention of this article is to make you all aware of hyperparameter tuning.

HYPERPARAMETER TUNING:

As we know that there are parameters that are internally learned from the given dataset and derived from the dataset, they are represented in making predictions, classification and etc., These are so-called **Model Parameters,** and they are varying with respect to the nature of the data we couldn't control this since it depends on the data. Like '**m**' and '**C**' in linear equation, which is the value of coefficients learned from the given dataset.Some set of parameters that are used to control the behaviour of the model/algorithm and adjustable in order to obtain an improvised model with optimal performance is so-called **Hyperparamer.**

HYPERPARAMETER LIFE CYCLE:



Hyperparameter Space

As we know that there is a list of HPs for any selected algorithm(s) and our job is to figure out the best combination of HPs and to get the optimal results by tweaking them strategically, this process will be providing us with the platform for **Hyperparameter Space** and this combination leads to provide the best optimal results, no doubt in that but finding this combo is not so easy, we have to search throughout the space. Here every combination of selected HP value is said to be the "**MODEL**" and have to evaluate the same on the spot. For this reason, there are two generic approaches to search effectively in the HP space are **GridSearch CV** and **RandomSearch CV.** Here CV denotes **Cross-Validation.**

DATA LEAKAGE:

Well! Now quickly will understand what is Data leakage in ML, this is mainly due to not following some of the recommended best practices during the Data Science/Machine Learning life cycle.

The resulting

is Data Leakage, that’s fine, what is the issue here, after successful testing with perfect accuracy followed by training the model then the model has been planned to move into production. At this moment ALL Is Well.

Still, the actual/real-time data is applied to this model in the production environment, you will get poor scores. By this time, you may think that why did this happen and how to fix this. This is all because of the data that we split data into training and testing subsets. During the training the model has the knowledge of data, which the model is trying to predict, this results in inaccurate and bad prediction outcomes after the model is deployed into production.

Causes of Data Leakage

Data Pre-processing

The major root cause is doing all EDA processes before splitting the dataset into test and train

<!--[endif]-->Doing straightforward normalizing or rescaling on a given dataset

<!--[endif]-->Performing Min/Max values of a feature

<!--[endif]-->Handling missing values without reserving the test and train

<!--[endif]-->Removing outliers and Anomaly on a given dataset

<!--[endif]-->Applying standard scaler, scalin

Data Leakage

Well! Now quickly will understand what is Data leakage in ML, this is mainly due to not following some of the recommended best practices during the Data Science/Machine Learning life cycle. The resulting

is Data Leakage, that’s fine, what is the issue here, after successful testing with perfect accuracy followed by training the model then the model has been planned to move into production. At this moment ALL Is Well.

Still, the actual/real-time data is applied to this model in the production environment, you will get poor scores. By this time, you may think that why did this happen and how to fix this. This is all because of the data that we split data into training and testing subsets. During the training the model has the knowledge of data, which the model is trying to predict, this results in inaccurate and bad prediction outcomes after the model is deployed into production.

Causes of Data Leakage

Data Pre-processing

The major root cause is doing all EDA processes before splitting the dataset into test and train

<!--[endif]-->Doing straightforward normalizing or rescaling on a given dataset

<!--[endif]-->Performing Min/Max values of a feature

<!--[endif]-->Handling missing values without reserving the test and train

<!--[endif]-->Removing outliers and Anomaly on a given dataset

<!--[endif]-->Applying standard scaler, scaling, assert normal distribution on the full dataset

Bottom line is, we should avoid doing anything to our training dataset that involves having knowledge of the test dataset. So that our model will perform in production as a generalised model.

will go through the available Hyperparameters across the various algorithms and how we could implement all these factors and impact the model.

Steps to Perform Hyperparameter Tuning

Select the right type of model.

<!--[endif]-->Review the list of parameters of the model and build the HP space

<!--[endif]-->Finding the methods for searching the hyperparameter space

<!--[endif]-->Applying the cross-validation scheme approach

GRIDE SEARCH:

The **Grid Search** one that we have discussed above usually increases the complexity in terms of the computation flow, So sometimes GS is considered inefficient since it attempts all the combinations of given hyperparameters. But the **Randomized Search** is used to train the models based on random hyperparameters and combinations. obviously, the number of training models are small column than grid search.

In simple terms, In Random Search, in a given grid, the list of hyperparameters are trained and test our model on a random combination of given hyperparameters.

Getting RandomForestClassifier object for my operation.

```
from sklearn.model_selection import RandomizedSearchCV

from sklearn.ensemble import RandomForestClassifier

from scipy.stats import randint as sp_randint
```

Assigning my Train and Test spilt to my RandomForestClassifier object

```
# build a RandomForestClassifier
clf = RandomForestClassifier(n_estimators=50)
```

Specifying the list of parameters and distributions

```
param_dist = {"max_depth": [3, None],
              "max_features": sp_randint(1, 11),
              "min_samples_split": sp_randint(2, 11),
              "min_samples_leaf": sp_randint(1, 11),
              "bootstrap": [True, False],
              "criterion": ["gini", "entropy"]}
```

Defining the sample, distributions and cross-validation

```
samples = 8  # number of random samples
randomCV     =     RandomizedSearchCV(clf,     param_distributions=param_dist,
n_iter=samples,cv=3)
```

All parameters are set and, let's do the fit model

```
randomCV.fit(X, y)
print(randomCV.best_params_)
```

Output

```
{'bootstrap': False, 'criterion': 'gini', 'max_depth': 3, 'max_features': 3,
'min_samples_leaf': 7, 'min_samples_split': 8}
```

As per the Cross-Validation process, will figure out the mean and get the results

```
randomCV.cv_results_['mean_test_score']
```

Output

```
array([0.73828125, 0.69010417, 0.7578125 , 0.75911458, 0.73828125,
            nan,       nan, 0.7421875 ])
```

Best accuracy from training

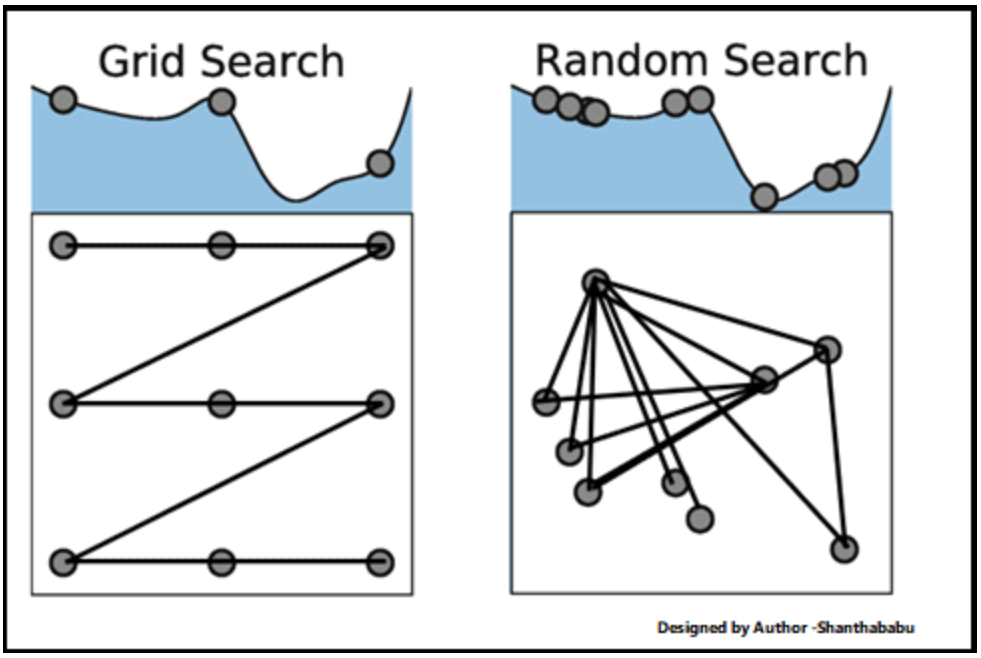
```
print(randomCV.score(X_test,y_test))
```

Output

```
0.8744588744588745
```

You may have a question, now which technique is best to go. The straight answer is RandomSearshCV, let's

The blow pictorial representation would give you the best understanding of GridSearchCV and RandomSearshCV



Conclusion

Guys! So far we have discussed in a detailed study of Hyperparameter visions with respect to the Machine Learning point of view, please remember a few things before we go

- Each model has a set of hyperparameters, so we have carefully chosen them and tweaked them during hyperparameter tuning. I mean building the HP space.
- All hyperparameters are NOT equally important and no defined rules for this. try to use continuous values instead of discrete values.
- Make sure to use K-Fold while using Hyperparameter tuning to improvise your hyperparameter tuning and coverage of hyperparameter space.
- Go with a better combination for hyperparameters and build strong results.