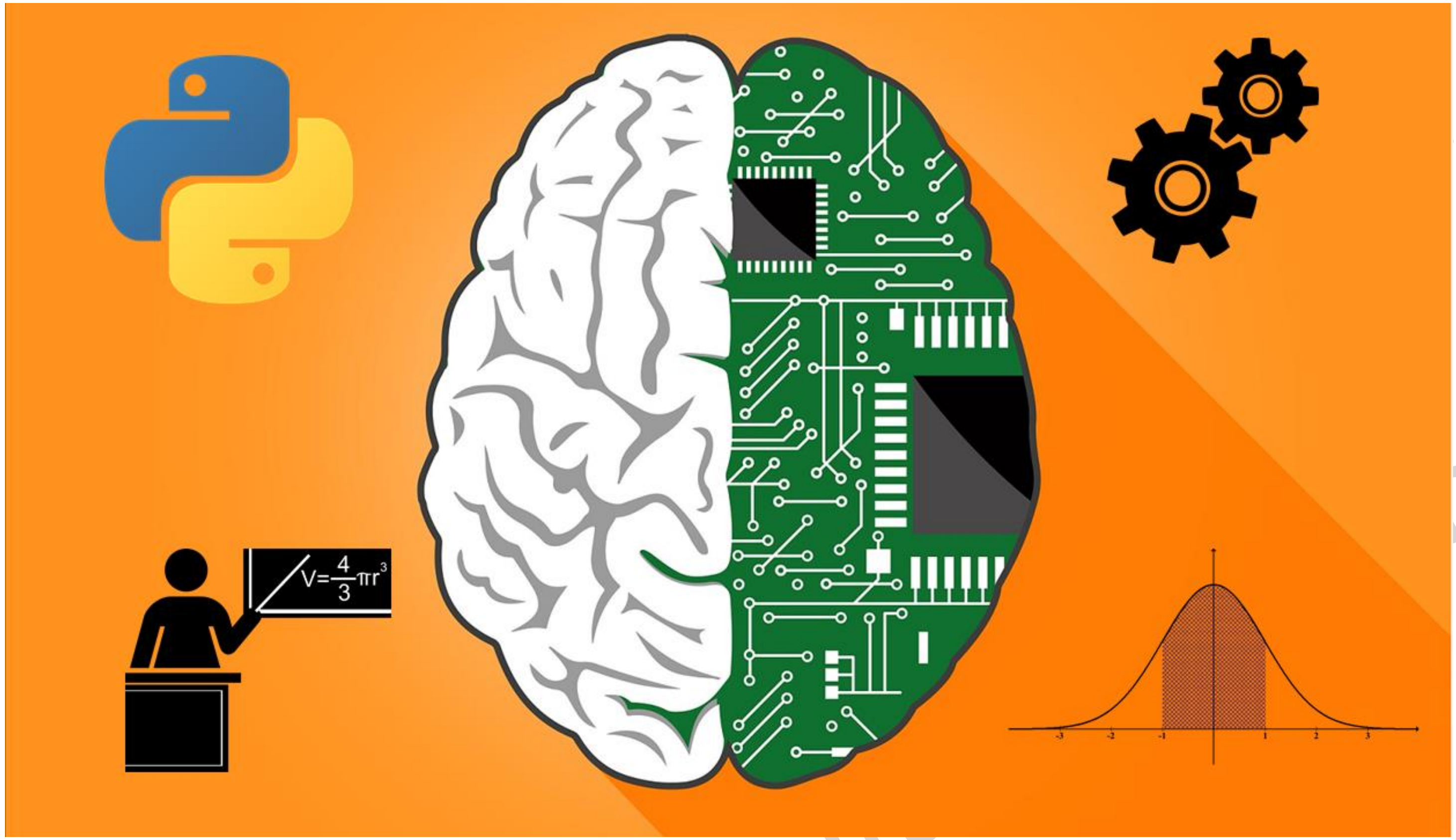
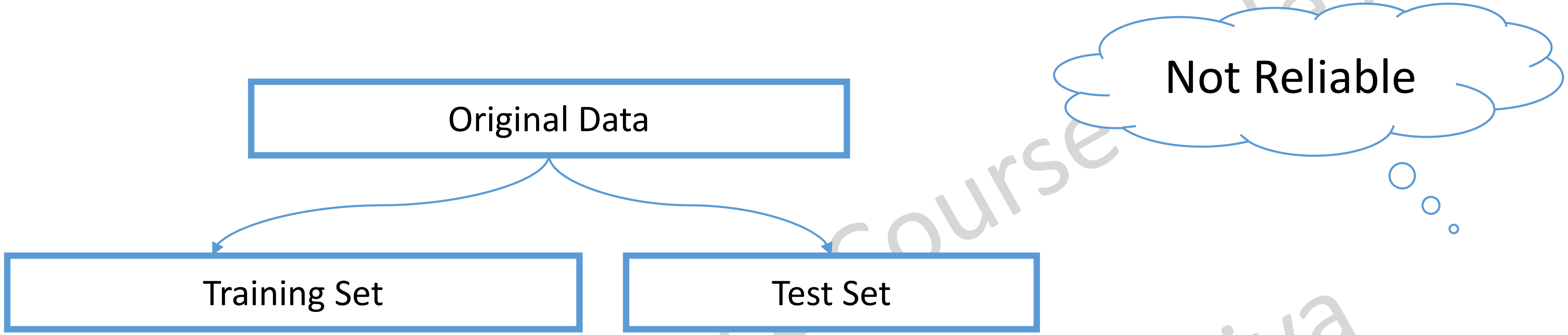


Complete Data Science and Machine Learning Using Python

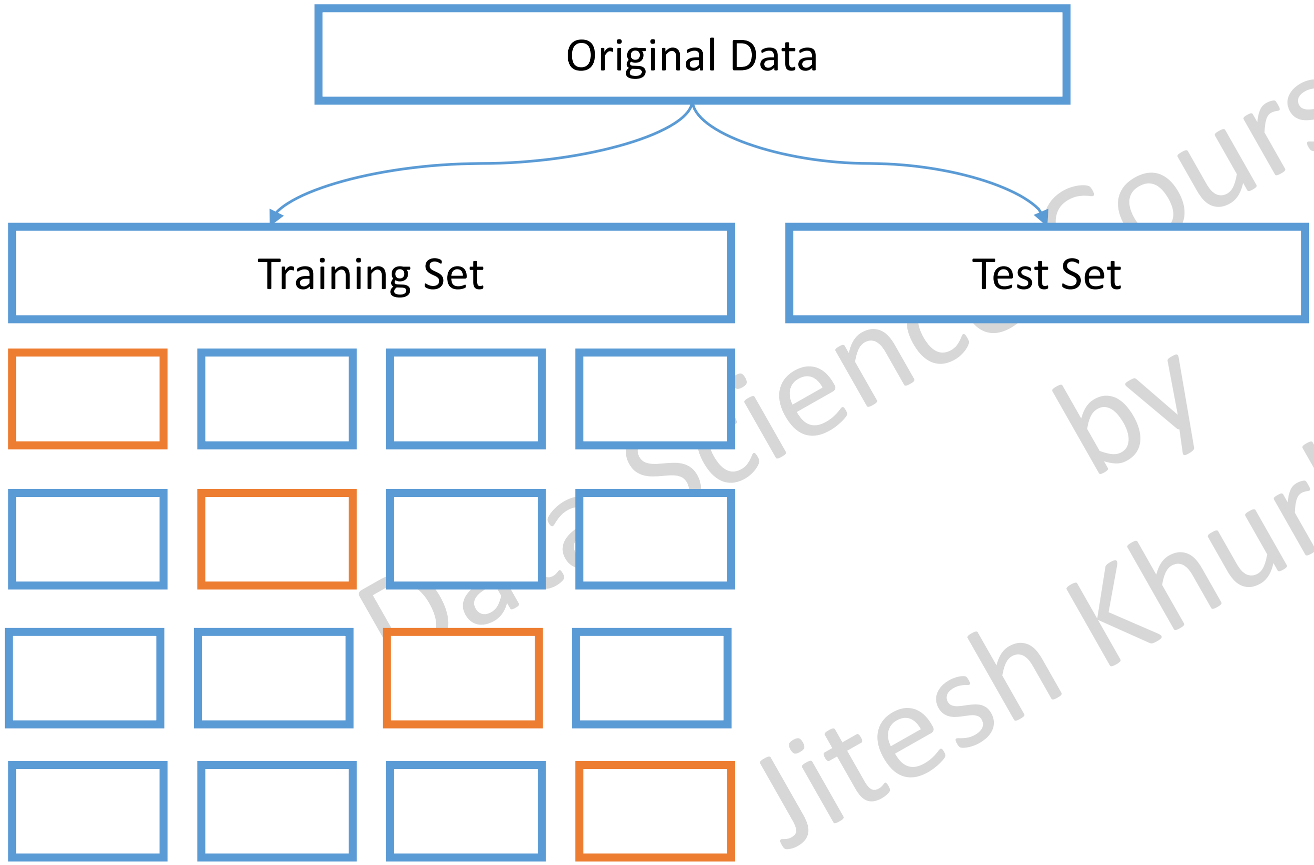
By
Jitesh Khurkhuriya

Hyperparameter Tuning

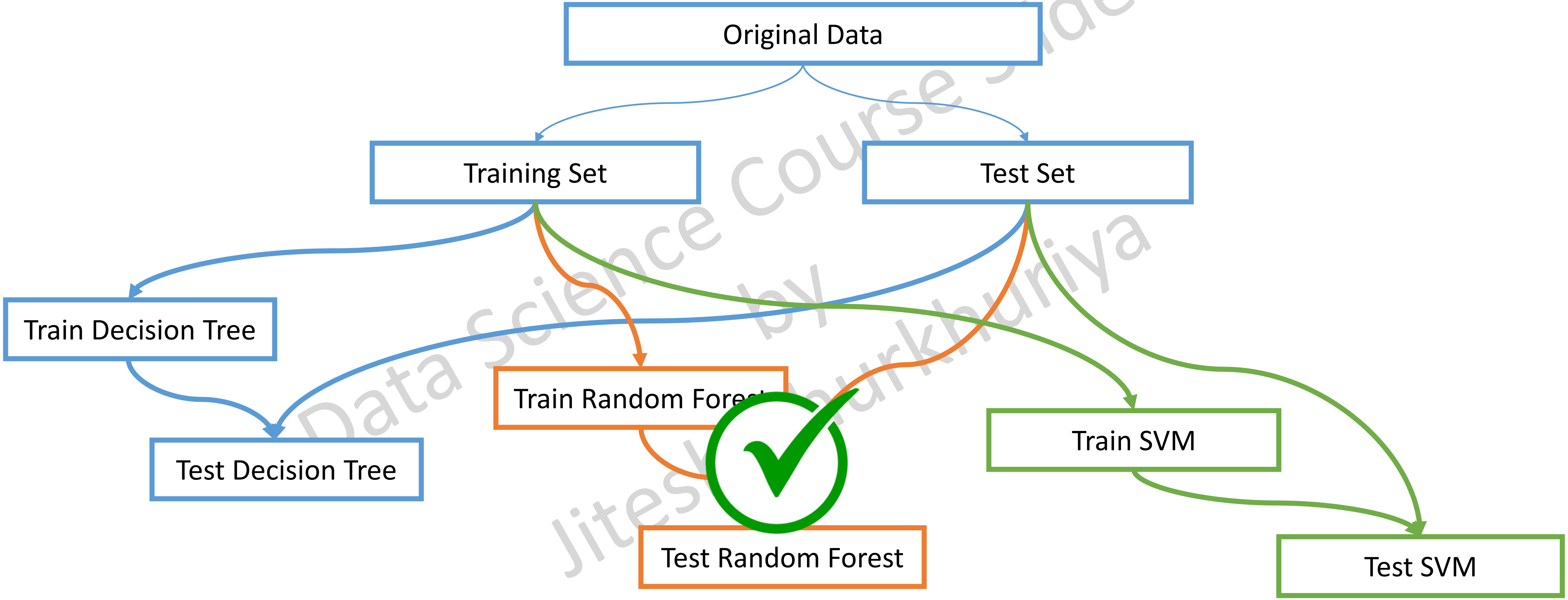
Model Selection



Cross Validation



Model Selection



Model Selection

```
              Decision Tree    Random Forest    Support Vector
              ---------------  ---------------  -----------------
Test   :      0.7816           0.7948           0.8036
Train  :      0.9044           0.8972           0.8745
```

Model Selection

	Decision Tree	Random Forest	Support Vector	Logistic Regression
Test :	0.7816	0.7948	0.8036	0.8133
Train :	0.9044	0.8972	0.8745	0.8141

What are Hyperparameter and Tuning?

As per Wikipedia,

- Hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm.

- Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data.

- A hyperparameter is a parameter whose value is used to control the learning process.

Cross Validation

```
# Import and train Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier(random_state=1234)

# Import and train Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=1234)

# Import and train Support Vector Classifier
from sklearn.svm import SVC
svc = SVC(kernel='rbf', gamma=0.5)

# Import and perform cross validation
from sklearn.model_selection import cross_validate
cv_results_dtc = cross_validate(dtc, X, Y, cv=10, return_train_score=True)
cv_results_rfc = cross_validate(rfc, X, Y, cv=10, return_train_score=True)
cv_results_svc = cross_validate(svc, X, Y, cv=10, return_train_score=True)
```

Decision Tree Classifier

- criteria
- splitter
- max_depth
- min_samples_split
- min_samples_leaf
- min_weight_fraction_leaf
- max_features
- max_leaf_nodes
- min_impurity_decrease
- min_impurity_split
- class_weight
- presort

Random Forest Classifier

- criterion
- n_estimators
- max_depth
- min_samples_split
- min_samples_leaf
- min_weight_fraction_leaf
- max_features
- max_leaf_nodes
- min_impurity_decrease
- min_impurity_split
- class_weight

Support Vector Classifier

- C – penalty parameter
- kernel
- degree
- gamma
- coef0
- shrinking
- probability
- tol
- cache_size
- class_weight
- max_iter

Logistic Regression

- penalty
- dual
- tol
- C – penalty parameter
- l1_ratio
- fit_intercept
- intercept_scaling
- class_weight
- solver
- max_iter

Random Forest Classifier

- criterion — default="gini"
- n_estimators — default=10
- max_depth
- min_samples_split — default=2
- min_samples_leaf — default=1
- min_weight_fraction_leaf
- max_features — default="auto"
- max_leaf_nodes
- min_impurity_decrease
- min_impurity_split
- class_weight

```
Random Forest
---------------
0.7948
0.8972
```

Support Vector Classifier

- C – penalty parameter — default=1.0
- kernel — default='rbf'
- degree
- gamma — gamma=0.5
- coef0
- shrinking
- probability
- tol — default=1e-3
- cache_size
- class_weight
- max_iter

```
Support Vector
----------------
0.8036
0.8745
```

What are Hyperparameter and Tuning?

As per Wikipedia,

- Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data.

Finding the **best combination** of the **hyperparameter values** that **minimizes errors** and provides the best/optimal model.

- criterion
- n_estimators
- max_depth
- min_samples_split
- min_samples_leaf
- min_weight_fraction_leaf
- max_features
- max_leaf_nodes
- min_impurity_decrease
- min_impurity_split
- class_weight

- C – penalty parameter
- kernel
- degree
- gamma
- coef0
- shrinking
- probability
- tol
- cache_size
- class_weight
- max_iter

Hyperparameter Tuning Approaches

GridSearchCV

Parameter 1 →

↓ Parameter 2

	1	2	3
A	A, 1	A, 2	A, 3
B	B, 1	B, 2	B, 3
C	C, 1	C, 2	C, 3
D	D, 1	D, 2	D, 3

RandomizedSearchCV

Parameter 1 →

↓ Parameter 2

	1	2	3
A	A, 1	A, 2	A, 3
B	B, 1	B, 2	B, 3
C	C, 1	C, 2	C, 3
D	D, 1	D, 2	D, 3

Random Forest Classifier

criterion	n_estimators	min_samples_leaf

criteria → (gini, entropy)

n_estimators → (5, 10, 15)

min_samples_leaf → (1, 2, 5)

Support Vector Classifier

kernel	C	gamma

kernel → (rbf, linear)

C → (0.25, 0.5, 1.0)

gamma → (0.25, 0.5, 1.0)

Random Forest Classifier

criterion	n_estimators	min_samples_leaf
gini	10	1
gini	10	2
gini	10	5
gini	5	1
gini	5	2
gini	5	5
gini	15	1
gini	15	2
gini	15	5

2 x 3 x 3 = 18

Support Vector Classifier

kernel	C	gamma
rbf	1.0	0.5
rbf	1.0	0.25
rbf	1.0	1
rbf	0.5	0.5
rbf	0.5	0.25
rbf	0.5	1
rbf	0.25	0.5
rbf	0.25	0.25
rbf	0.25	1

2 x 3 x 3 = 18

Random Forest Classifier

criterion	n_estimators	min_samples_leaf
gini	10	1
gini	10	2
gini	10	5
gini	5	1
gini	5	2
gini	5	5
gini	15	1
gini	15	2
gini	15	5

2 x 5 x 5 = 50

Support Vector Classifier

kernel	C	gamma
rbf	1.0	0.5
rbf	1.0	0.25
rbf	1.0	1
rbf	0.5	0.5
rbf	0.5	0.25
rbf	0.5	1
rbf	0.25	0.5
rbf	0.25	0.25
rbf	0.25	1

4 x 5 x 5 = 100

Hyperparameter Tuning Approaches

GridSearchCV

Parameter 1 →

↓ Parameter 2

	1	2	3
A	A, 1	A, 2	A, 3
B	B, 1	B, 2	B, 3
C	C, 1	C, 2	C, 3
D	D, 1	D, 2	D, 3

RandomizedSearchCV

Parameter 1 →

↓ Parameter 2

	1	2	3
A	A, 1	A, 2	A, 3
B	B, 1	B, 2	B, 3
C	C, 1	C, 2	C, 3
D	D, 1	D, 2	D, 3

What is a Grid?

- Cartesian Product of Parameters
- Parameter 1 → 1, 2, 3
- Parameter 2 → A, B, C, D

Parameter 1 →

↓ Parameter 2

	1	2	3
A	A, 1	A, 2	A, 3
B	B, 1	B, 2	B, 3
C	C, 1	C, 2	C, 3
D	D, 1	D, 2	D, 3

GridSearchCV

Parameter 1 →

↓ Parameter 2

	1	2	3
A	A, 1	A, 2	A, 3
B	B, 1	B, 2	B, 3
C	C, 1	C, 2	C, 3
D	D, 1	D, 2	D, 3

criterion	n_estimators	min_samples_leaf
gini	10	1
gini	10	2
gini	10	5
gini	5	1
gini	5	2
gini	5	5
gini	15	1
gini	15	2
gini	15	5

RandomizedSearchCV

Parameter 1 →

← Parameter 2

	1	2	3
A	A, 1	A, 2	A, 3
B	B, 1	B, 2	B, 3
C	C, 1	C, 2	C, 3
D	D, 1	D, 2	D, 3

criterion	n_estimators	min_samples_leaf
gini	10	1
gini	10	2
gini	10	5
gini	5	1
gini	5	2
gini	5	5
gini	15	1
gini	15	2
gini	15	5

Model Selection

Model Selection

Only Cross Validation

	Decision Tree	Random Forest	Support Vector	Logistic Regression
Test :	0.7816	0.7948	0.8036	0.8133
Train :	0.9044	0.8972	0.8745	0.8141

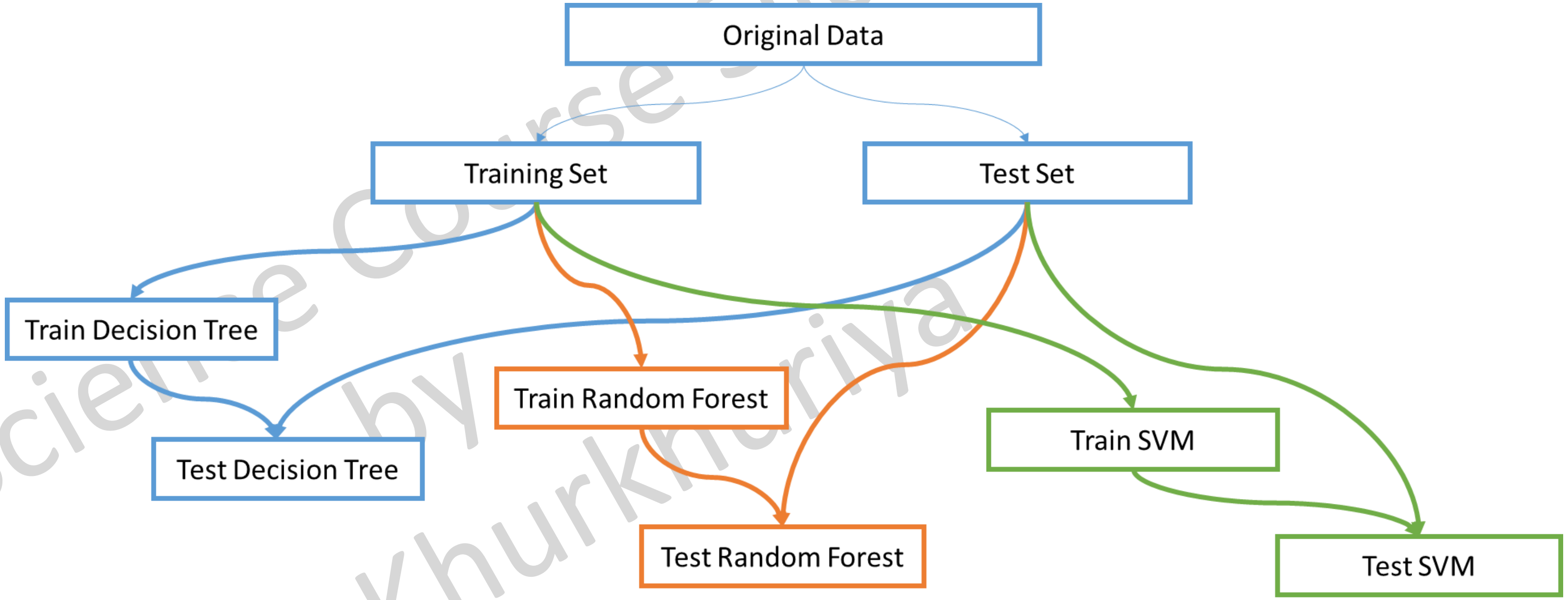
With Hyperparameter Tuning

Test : 0.8181
Train : 0.8424

Model Selection

Train Algorithm

Select the best scored



Model Test Score for Adult Income Prediction

Train Algorithm

~~Select the best scored~~

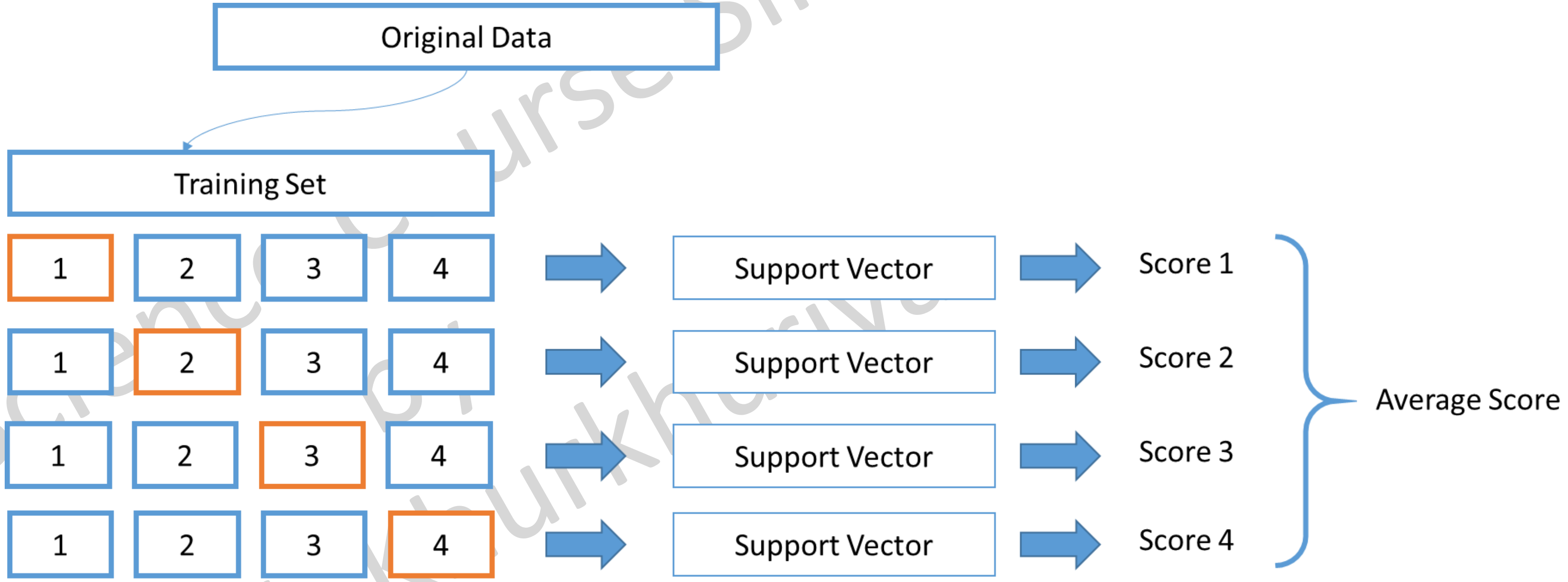
Split Random Seed	Split Size	Decision Tree	Random Forest	SVM
0	0.2	77.08%	79.18%	80.24%
123	0.2	78.39%	79.15%	**80.54%**
456	0.2	78.32%	78.57%	80.41%
999	0.2	**76.93%**	78.67%	79.73%
0	0.33	77.10%	79.30%	80.10%
123	0.33	77.81%	79.03%	79.46%
456	0.33	78.11%	79.31%	79.93%
999	0.33	77.70%	**78.39%**	79.49%
0	0.4	77.34%	78.96%	79.88%
123	0.4	78.44%	**79.87%**	79.63%
456	0.4	78.34%	79.01%	79.88%
999	0.4	77.43%	79.01%	79.79%
0	0.45	77.59%	79.30%	79.59%
123	0.45	78.06%	79.20%	**79.43%**
456	0.45	**78.50%**	79.29%	79.87%
999	0.45	77.20%	79.00%	79.71%

K-Fold Cross Validation

Train Algorithm

~~Select the best scored~~

Cross Validation



K-Fold Cross Validation

Train Algorithm

~~Select the best scored~~

Cross Validation

	Decision Tree	Random Forest	Support Vector	Logistic Regression
Test :	0.7816	0.7948	0.8036	0.8133
Train :	0.9044	0.8972	0.8745	0.8141

- Train Algorithm
- ~~Select the best scored~~
- Cross Validation
- Tune Parameters

Random Forest Classifier

Parameter	Default
criterion	default="gini"
n_estimators	default=10
max_depth	
min_samples_split	default=2
min_samples_leaf	default=1
min_weight_fraction_leaf	
max_features	default="auto"
max_leaf_nodes	
min_impurity_decrease	
min_impurity_split	
class_weight	

```
Random Forest
---------------
0.7948
0.8972
```

Support Vector Classifier

Parameter	Default
C – penalty parameter	default=1.0
kernel	default='rbf'
degree	
gamma	gamma=0.5
coef0	
shrinking	
probability	
tol	default=1e-3
cache_size	
class_weight	
max_iter	

```
Support Vector
-----------------
0.8036
0.8745
```

Train Algorithm

~~Select the best scored~~

Cross Validation

Tune Parameters

```
# define parameters for Random Forest
rfc_param = {'n_estimators':[10,15,20],
             'min_samples_split':[8,16],
             'min_samples_leaf':[1,2,3,4,5]
            }
```

```
# define parameters for Support Vector Classifier
svc_param = {'C':[0.01, 0.1, 0.5, 1, 2, 5, 10],
             'kernel':['rbf', 'linear'],
             'gamma':[0.1, 0.25, 0.5, 1, 5]
            }
```

```
# define parameters for Logistic Regression
lrc_param = {'C':[0.01, 0.1, 0.5, 1, 2, 5, 10],
             'penalty':['l2'],
             'solver':['liblinear','lbfgs', 'saga']
            }
```

		Random Forest	Logistic Regression	Support Vector
Mean Test Score	:	0.8181	0.8146	0.8145
Mean Train Score	:	0.8424	0.8148	0.8429

Train Algorithm

~~Select the best scored~~

Cross Validation

Tune Parameters

cv_results_svc - DataFrame

Index	mean_fit_time	std_fit_time	mean_score_time	std_score_time
40	20.1890	0.3919	0.9205	0.0235
50	26.0500	0.7167	0.8961	0.0311
30	18.3952	0.5654	0.9599	0.0272

cv_results_rfc - DataFrame

Index	mean_fit_time	std_fit_time	mean_score_time	std_score_time
23	0.2411	0.0330	0.0121	0.0010
18	0.1517	0.0270	0.0095	0.0034

High Score Time

Higher Train-Test Score difference

```
                      Random Forest      Logistic Regression     Support Vector
                      ---------------    -------------------     ---------------
Mean Test Score   :   0.8181             0.8146                  0.8145
Mean Train Score  :   0.8424             0.8148                  0.8429
```

cv_results_lrc - DataFrame

Index	mean_fit_time	std_fit_time	mean_score_time	std_score_time
3	0.1381	0.0163	0.0021	0.0008
13	0.1808	0.0077	0.0016	0.0005

Train Algorithm

~~Select the best scored~~

Cross Validation

Tune Parameters

cv_results_svc - DataFrame

Index	mean_fit_time	std_fit_time	mean_score_time	std_score_time
40	20.1890	0.3919	0.9205	0.0235
50	26.0500	0.7167	0.8961	0.0311
30	18.3952	0.5654	0.9599	0.0272

cv_results_rfc - DataFrame

Index	mean_fit_time	std_fit_time	mean_score_time	std_score_time
23	0.2411	0.0330	0.0121	0.0010
18	0.1517	0.0270	0.0095	0.0034

High Score Time
Higher Train-Test Score difference

```
                        Random Forest     Logistic Regression    Support Vector
                        ---------------   -------------------    ---------------
Mean Test Score    :    0.8181            0.8146                 0.8145
Mean Train Score   :    0.8424            0.8148                 0.8429
```

- **Faster Predictions**
- **High Accuracy**
- **Less overfitting**

cv_results_lrc - DataFrame

Index	mean_fit_time	std_fit_time	mean_score_time	std_score_time
3	0.1381	0.0163	0.0021	0.0008
13	0.1808	0.0077	0.0016	0.0005

Train Algorithm

~~Select the best scored~~

Cross Validation

Tune Parameters

	Random Forest	Logistic Regression	Support Vector
Mean Test Score :	0.8181	0.8146	0.8145
Mean Train Score :	0.8424	0.8148	0.8429

Random Forest:

- **Highest Accuracy**

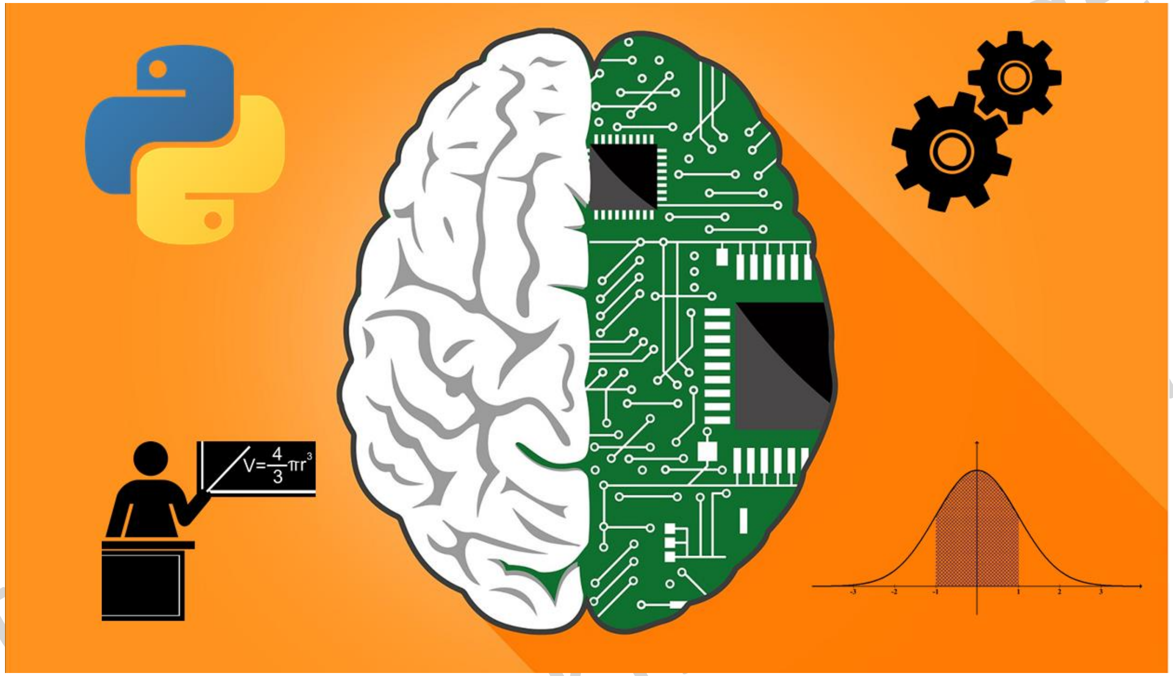
Logistic Regression:

- **High Accuracy**
- **Less overfitting**

Support Vector:

High Score Time
Higher Overfitting

Complete Data Science and Machine Learning Using Python



Thank You!