24. Data Science – Machine Learning – K Fold Cross Validation

Contents

24. Data Science – Machine Learning – K Fold Cross Validation

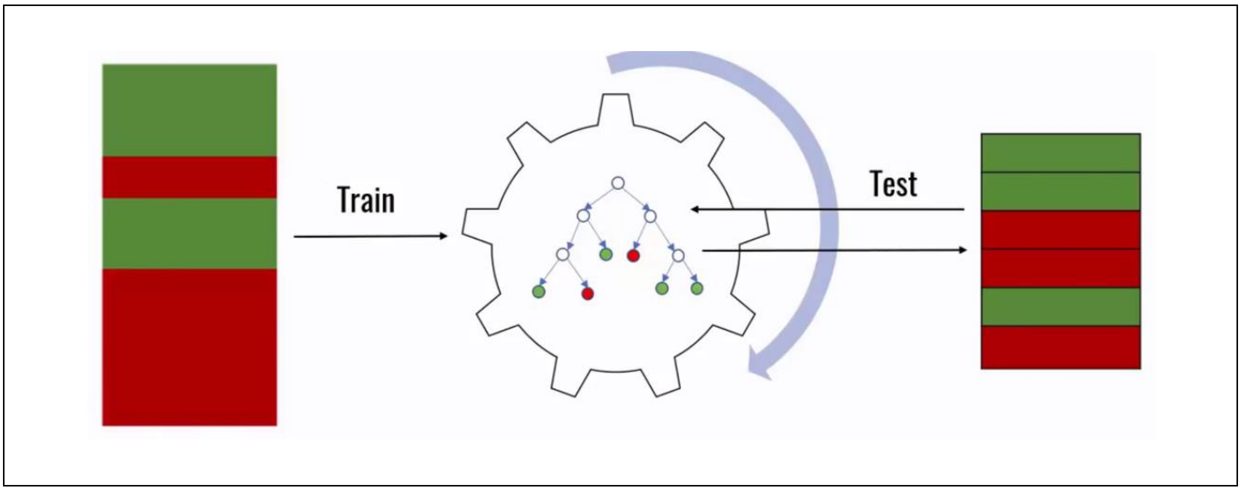
- ✓ To solve a problem we do have different machine learning algorithm for same problem
- ✓ So, we need to understand clearly which model is the best to use

1. K-fold cross validation

- ✓ Cross-validation is a technique, it evaluate the model performance.

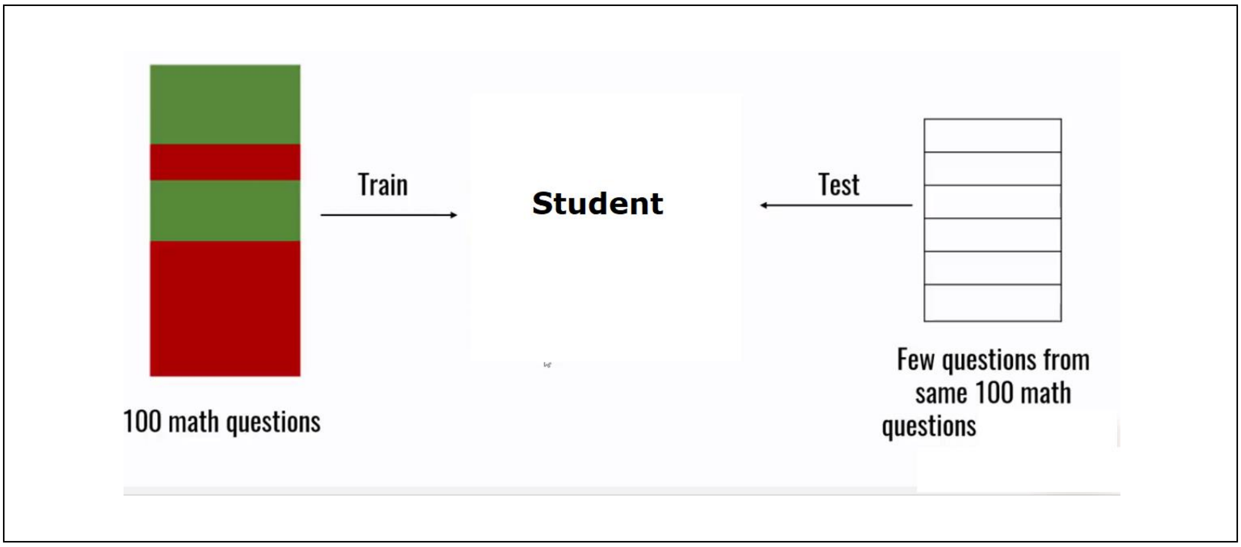
2. Ways to training the model

- ✓ So far we learned to spilt the data into train and test datasets
- ✓ Once the model got trained then we need to test the model



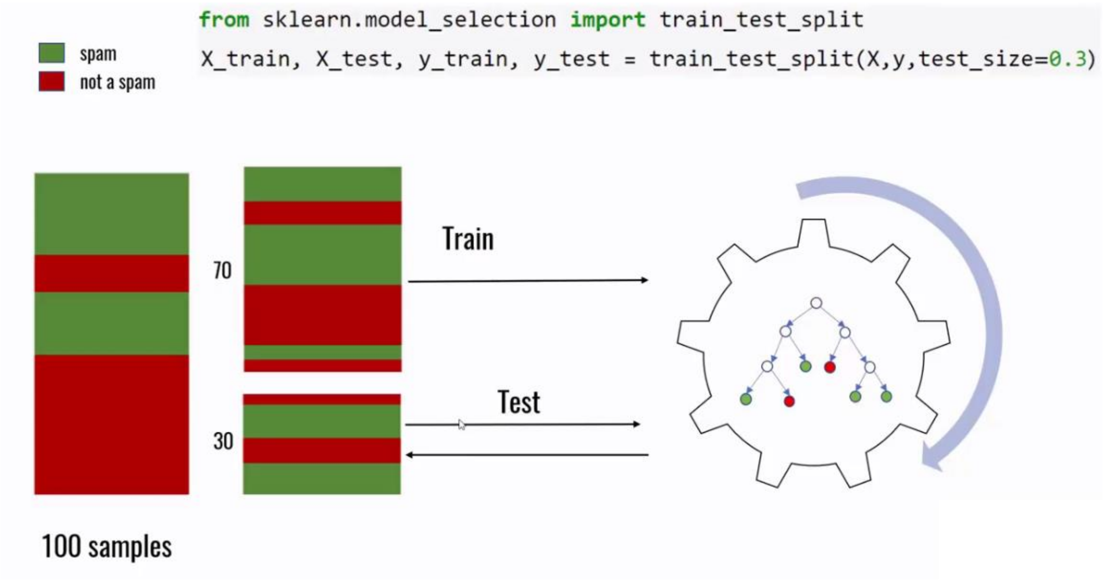
3. Scenario 1

- ✓ Use all dataset to train and test the model

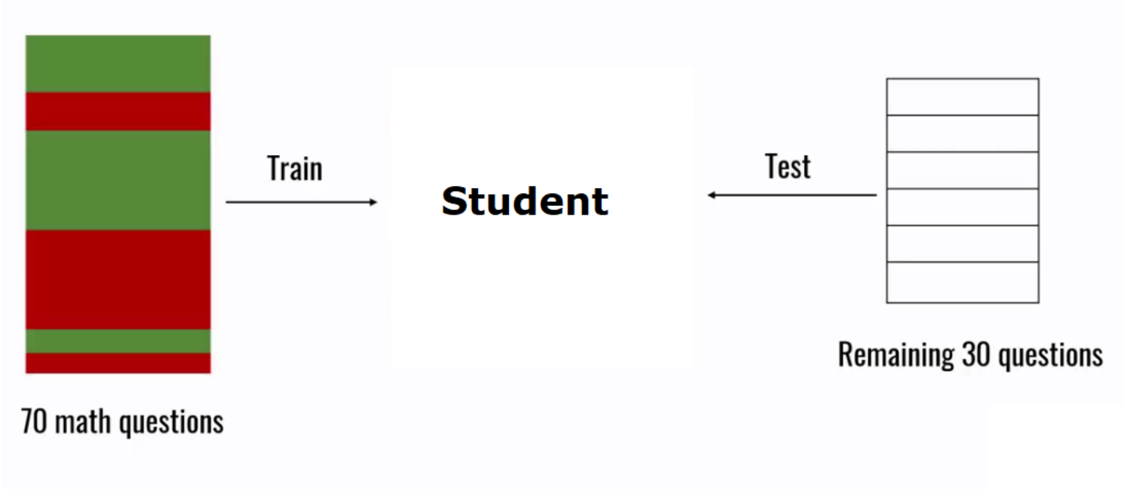


4. Scenario 2

- ✓ Split available dataset into training and testing to test the model



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```



5. Limitation

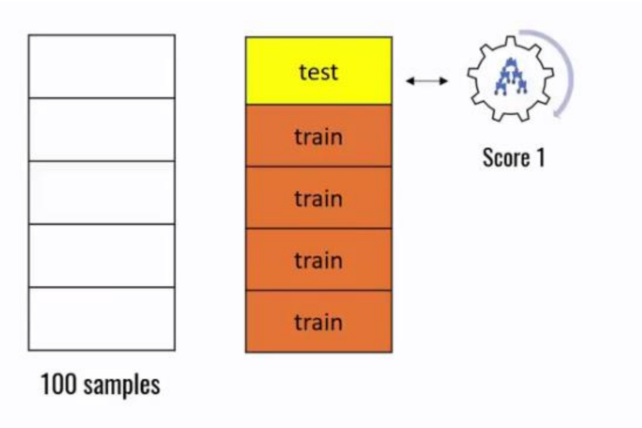
- ✓ During testing the model with test dataset, model may fail in few scenarios because model need to face new scenarios right

6. K fold cross validation

- ✓ Cross-validation is a technique, it evaluate the model performance.
- ✓ We used to divide 100 samples into folds, each contains 20 samples
- ✓ Then now we can start iterations to test the model in different ways

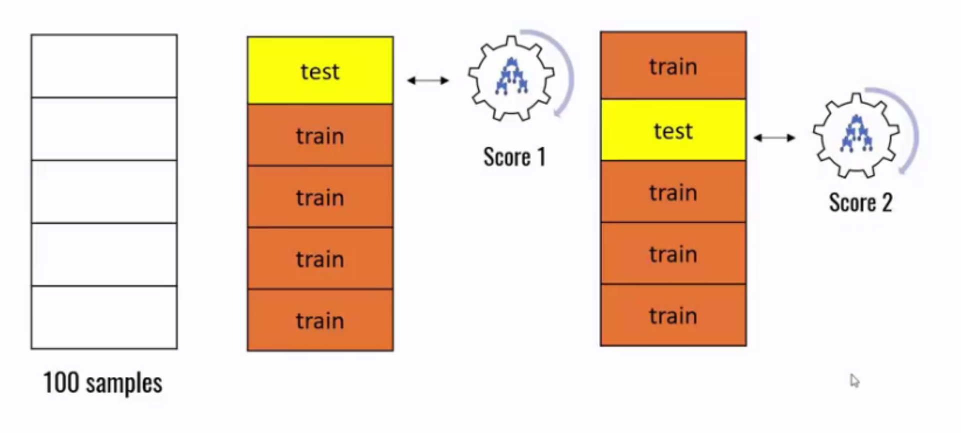
7. Iteration 1

- ✓ Use the first fold to test the model
- ✓ Use the remaining folds to training



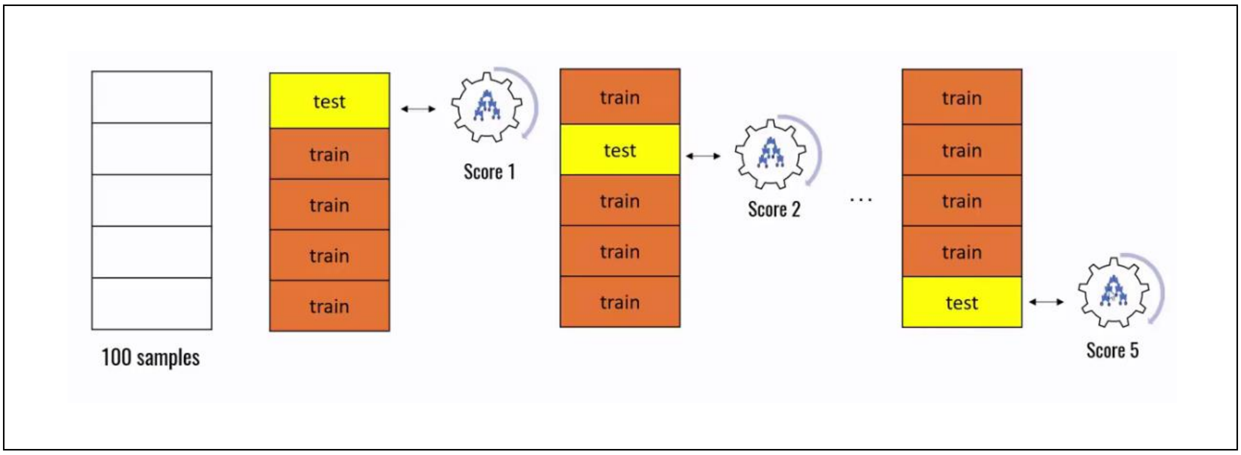
8. Iteration 2

- ✓ Use the second fold to test the model
- ✓ Use the remaining folds to training



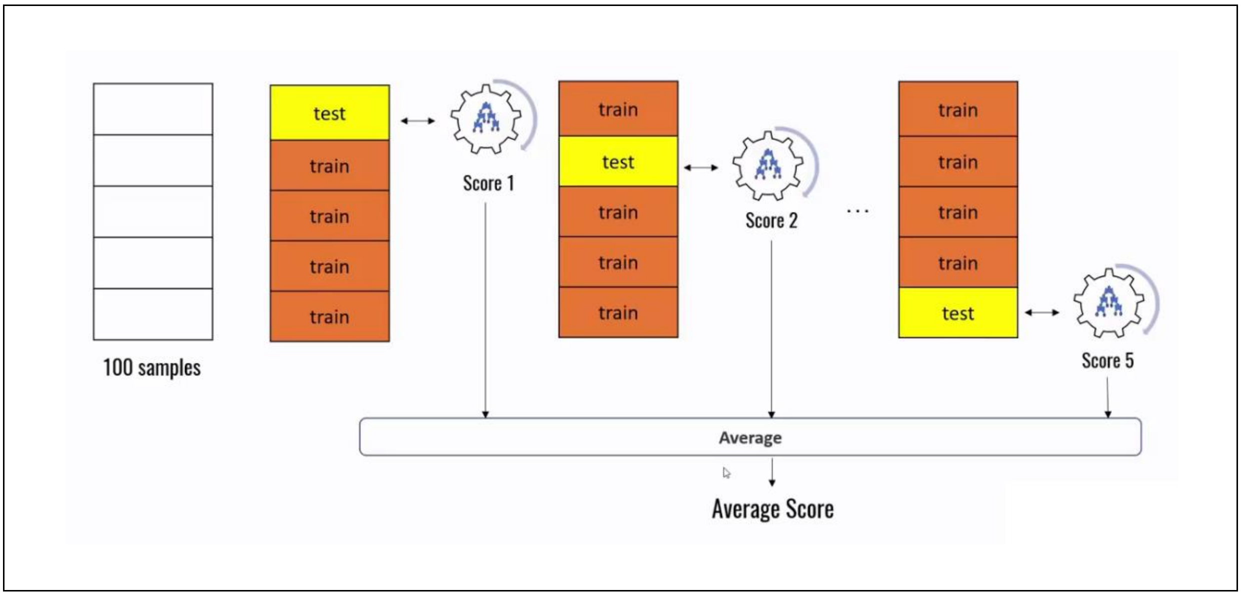
9. Last iteration

✓ Repeat the process till to the last fold



10. Average of scores

✓ This technique is good to calculate the average of all iterations scores



Program Name

Dataset is loading
demo1.py

```
from sklearn.datasets import load_digits

digits = load_digits()

print("Dataset is loading")
```

Output

```
Dataset is loading
```

Program Name

Splitting the data into train and test datasets
demo2.py

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

digits = load_digits()
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size = 0.3)

print("Splitting the data into train and test")
```

Output

```
Splitting the data into train and test
```

Program Name

Applying logistic regression
demo3.py

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

digits = load_digits()
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.3)

lr = LogisticRegression(solver = 'lbfgs', max_iter = 3000)
lr.fit(X_train, y_train)
print(lr.score(X_test, y_test))
```

Output

```
0.95
```

Program Name

Applying SVM algorithm
demo4.py

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

digits = load_digits()
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.3)

svm = SVC()
svm.fit(X_train, y_train)

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

Output

```
0.9907407407407407
```

Program Name

Applying Random forest algorithm
demo5.py

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

digits = load_digits()
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.3)

rf = RandomForestClassifier(n_estimators=40)
rf.fit(X_train, y_train)

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

Output

```
0.975925925925926
```

Program Name

K fold cross validation
demo6.py

```
from sklearn.model_selection import KFold

kf = KFold(n_splits=3)

print(kf)
```

Output

```
KFold(n_splits=3, random_state=None, shuffle=False)
```

Program Name

K fold cross validation: Example
demo7.py

```
from sklearn.model_selection import KFold

kf = KFold(n_splits=3)

for train_index, test_index in kf.split([1,2,3,4,5,6,7,8,9]):
        print(train_index, test_index)
```

Output

```
[3 4 5 6 7 8] [0 1 2]
[0 1 2 6 7 8] [3 4 5]
[0 1 2 3 4 5] [6 7 8]
```

Program Name

Logistic regression model performance using cross_val_score
demo8.py

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.3)

a = LogisticRegression(solver = 'lbfgs', max_iter = 5000)
scores = cross_val_score(a, digits.data, digits.target, cv=3)

print(scores)
```

Output

```
[0.92153589 0.94156928 0.91652755]
```

Program Name

SVM model performance using cross_val_score
demo9.py

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.3)

b = SVC()
scores = cross_val_score(b, digits.data, digits.target, cv=3)

print(scores)
```

Output

```
[0.92153589 0.94156928 0.91652755]
```

Program Name

Random forest model performance using cross_val_score
demo10.py

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.3)

c = RandomForestClassifier(n_estimators=40)
scores = cross_val_score(c, digits.data, digits.target, cv=3)

print(scores)
```

Output

```
[0.93823038 0.94156928 0.92821369]
```

Program Name

Checking average of all model scores
demo11.py

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
import numpy as np

digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.3)

a  = LogisticRegression(solver = 'lbfgs', max_iter = 5000)
b = SVC()
c = RandomForestClassifier(n_estimators=40)

scores1 = cross_val_score(a, digits.data, digits.target, cv=3)
scores2 = cross_val_score(b, digits.data, digits.target, cv=3)
scores3 = cross_val_score(c, digits.data, digits.target, cv=3)

print(np.average(scores1))
print(np.average(scores2))
print(np.average(scores3))
```

Output

```
0.9265442404006677
0.9699499165275459
0.9315525876460767
```