

Lesson 15-21

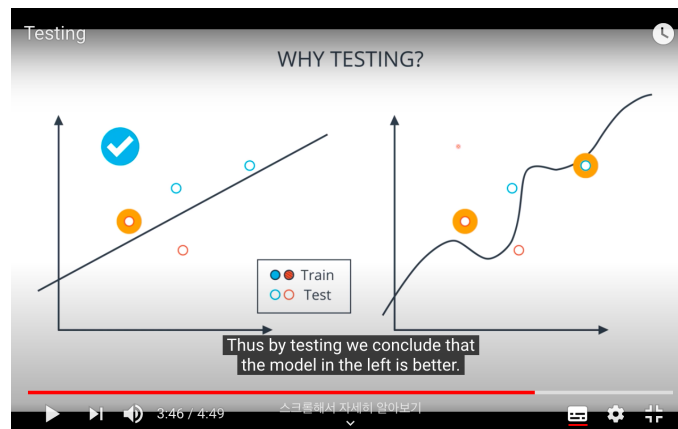
Files	
Tags	

Lesson 17. Model Testing and Evaluation

problem 문제 / tool 도구(모델) / measurement tool (측정 모델)

1. Testing : 모델이 잘 데이터를 대표하는지를 검사

- training / test dataset 으로 구분 - training 은 모델 만들고 test는 모델 평가
- (오른쪽 그림 overfitting)
- ** 절대 test data를 training에 쓰면 안됨!



2. confusion matrix : 모델 평가 지표 (TP,TN,FP,FN)

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Confusion Matrix and ROC Curve

		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

Model Performance

Accuracy = $(TN+TP)/(TN+FP+FN+TP)$

Precision = $TP/(FP+TP)$

Sensitivity = $TP/(TP+FN)$

Specificity = $TN/(TN+FP)$

TN True Negative
 FP False Positive
 FN False Negative
 TP True Positive

3. Accuracy(정확도) : 얼마나 정확하게 예측했냐?

맞게 예측한 수 / 전체 예측 수

하지만 accuracy 는 만능이 아니다 → 우연적 사건을 평가할 수 없다 (ex. 신용카드 거래 중 사기 건수 0.01%)

4. Recall (재현율)로 구하면 된다

= 통계학에서는 sensitivity, hit rate

: 실제 True인 것 중에서 모델이 True라고 예측한 것의 비율

$TP/(TP+FN)$

[실제 정답(data)의 입장]

5. precision(정밀도)

: True라고 분류한 것 중에서 실제 True인 것의 비율







$TP/(TP+FP)$

6. under/overfitting

04 L Types Of Errors

전체화면을 종료하려면 **esc** 을(를) 누르세요.

TRADEOFF

High bias (underfitting)	Good Model	High variance (overfitting)
<p>Not animals</p>  <p>Animals</p> 	<p>No dogs</p>  <p>Dogs</p> 	<p>No dogs who wag their tails</p>  <p>Dogs who wag their tails</p> 
<p>Oversimplify the problem</p> <p>Bad on training set</p> <p>Bad on testing set</p>	<p>Good model</p> <p>Good on training set</p>	<p>Overcomplicate the problem</p> <p>Great on training set</p>

And so does the high variance model.

3:56 / 4:04

7. 모델 복잡도 모형 model complexity graph

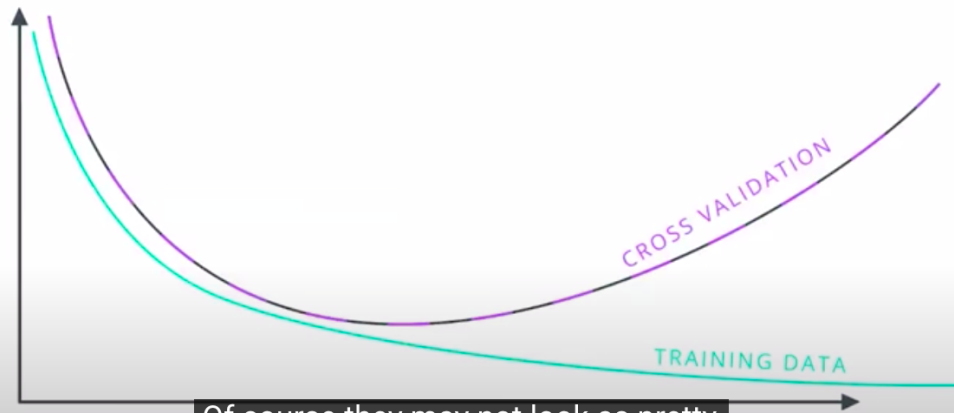
데이터 많을수록, 복잡도 증가

under/overfitting 의 절충점

모델의 일반화가 잘 됐는지 & 예측 정확도가 좋은지 평가



MODEL COMPLEXITY GRAPH



Of course they may not look as pretty,
but in real life you see that most of

- regularization : 복잡도를 잡는 방법
- ex) drop-out : 특정 변수나 일부 파라미터를 없애고 학습
- ex) Pruning , feature selection, Ensemble
- <https://bahnsville.tistory.com/1140>

8. k-fold validation

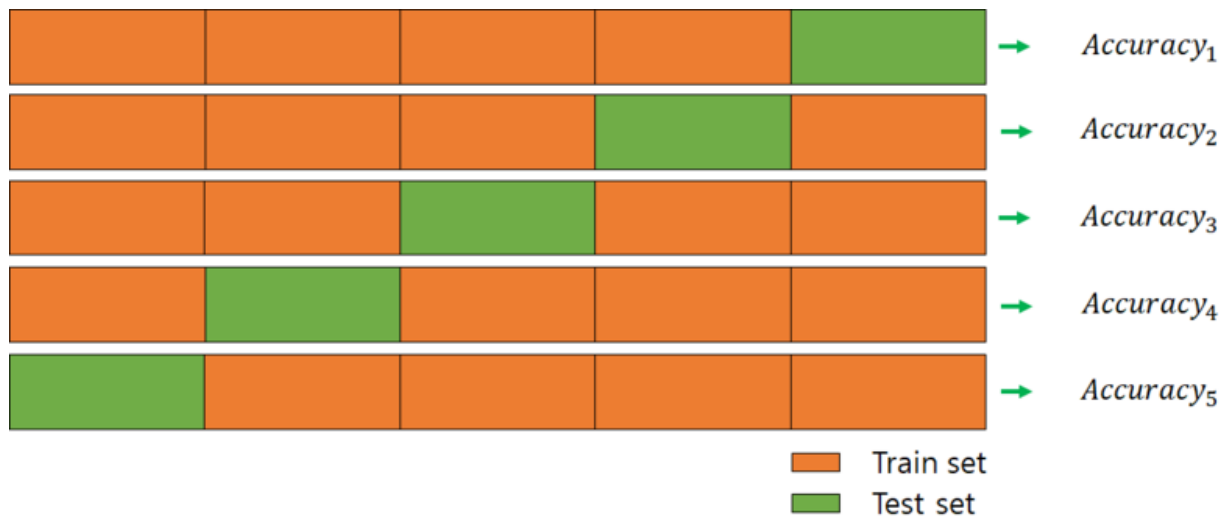
일반적 label 분류 : train, test set → overfitting



k-fold 교차검증 (k-fold cross validation)

: 전체 데이터 셋을 k개의 subset으로 나누고, k번의 평가를 실행 + 이 때 test set을 중복 없이 바꾸어가면서 평가를 진행

k개의 평가 지표(이 경우는 accuracy로 예를 들)를 평균(때에 따라 평균이 아닌 방법을 사용할 수도 있음)내어서 최종적으로 모델의 성능 평가



Lesson 18

1. Ensemble Model

- 여러개 모델이 투표해서, 또는 신호를 합성해서

2. Perturbations on Columns - Random Subspace

- 몇개 칼럼만 써서 모델 만들기
- 어떻게 고를지는 랜덤

3. Perturbations on Rows - Bagging (bootstrap aggregating)

- Bootstrapping - resampling with replacement "복원추출"
- 적당한 숫자를 고르고
- Improve unstable procedure (DNN, RF), might degrade stable procedure (kNN)
- cf. Boosting
 - 여러 약한 learner를 합성해서 좀 더 강한 learner를
- 부스팅이든 배깅이든 이런 여러개 섞어서 예측성능 향상시키는걸 앙상블이라고 합니다
- (시험문제 내기 좋다...)

4. Out-of-bag Estimate

- 가로세로 둘다
- rows 뿐만아니라 columns에서도 subsampling을 하는 이유는 생성되는 트리간의 correlation 제거를 위해서

5. Forest of Randomised Trees

- 별다른 설정 안하고, 전체 데이터크기와 bag크기를 같게 한다면 2/3정도가 유니크하게 들어가고 나머지는 중복... 그러면 나머지는 포함이 안 돼 있겠죠?
- (Udacity랑 위키랑 설명이 다른데)
- 그 포함이 안 된 애들을 각 모델이 예측한 것이 oob estimate
- oob estimate를 크게 하는 방향으로 최적화를 하면 될거같아용

6. Random Forest Hyperparameters

- <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

7. Choosing Hyperparameter Values

- Grid Search, Cross-validation
- Finance → Overfitting?

8. Random Forests for Alpha Combination

- Create Random Forest with alpha factors (columns) & additional features (columns)

Lesson 21. Feature Importance

1. Feature Importance

모델을 이해하기 위해서는 각 Feature들이 모델의 예측에 얼마나 기여하는지가 중요함

특정 Feature을 제거했을 때 모델에 끼치는 영향을 확인하여 이를 측정할 수 있음(Ablation Study)

당연히 이런 이해가 늘면 예측 모델의 성능을 올릴 수 있음

→ eXplainable AI

2. Feature Importance in sklearn

tree-based model (random forest)

→ 분리(Splitting)를 통한 불순물 제거(impurity removing)

▼ Impurity : 'disorganized' 정도 (like entropy)

$$Impurity = \sum_{i=1}^C -freq_i * (1 - freq_i)$$

$freq_i$: frequency of label i

C : the number of unique labels

▼ Node Importance : 해당 노드와 왼쪽 or 오른쪽 자식 노드와의 Impurity 차이

$$NodeImportance = w_i Impurity_i - (w_{left} Impurity_{left} + w_{right} Impurity_{right})$$

w : num of data

3. Shapley Additive Explanations

$$\phi_i = \sum_{S \subseteq M \setminus i} \frac{|S|!(|M|-|S|-1)!}{|M|!} [f(S \cup i) - f(S)]$$

A key part of this is the difference between the model's prediction with the feature i , and the model's prediction without feature i .

S refers to a subset of features that doesn't include the feature for which we're calculating ϕ_i .

$S \cup i$ is the subset that includes features in S plus feature i .

$S \subseteq M \setminus i$ in the Σ symbol is saying, all sets S that are subsets of the full set of features M , excluding feature i .

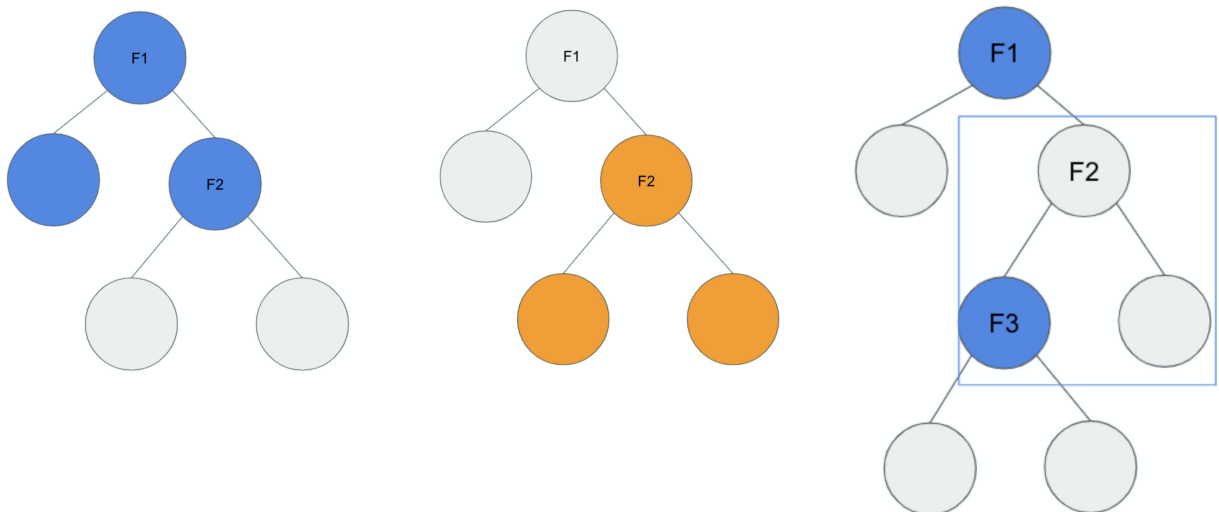
Ablation Study : 해당 Feature가 있고 없고에 따른 성능 차 확인

해당 Feature를 제외한 모든 조합을 다 해보고 평균]

M : full set of features, S : subset of features

local feature → global Feature을 이끌어낼 것임

4. Tree Shap



Algorithm 1 Estimating $E[f(x) \mid x_S]$

```
procedure EXPVALUE( $x, S, tree = \{v, a, b, t, r, d\}$ )
  procedure  $G(j, w)$ 
    if  $v_j \neq \text{internal}$  then
      return  $w \cdot v_j$ 
    else
      if  $d_j \in S$  then
        return  $G(a_j, w)$  if  $x_{d_j} \leq t_j$  else  $G(b_j, w)$ 
      else
        return  $G(a_j, wr_{a_j}/r_j) + G(b_j, wr_{b_j}/r_j)$ 
      end if
    end if
  end procedure
  return  $G(1, 1)$ 
end procedure
```

Handwritten notes: "leaf" (under return w · v_j), "in" (under if d_j ∈ S), "out" (under end if)

G : Tree Node

w : weight for each node

v : prediction of a leaf node

r_{a_j}, r_{b_j} : # of data points in the left and right child nodes of the nodes

$j \cdot r_j$: # of data points in node j

5. Rank Features

local feature를 global feature에 잘 적용하기 위해 local feature의 가치를 제대로 측정하고 이를 평균 내야함

→ sklearn or Shap Library로 측정

→ Feature Importance에 따른 수정 및 성능 향상

→ 가지치기 ㄱㄱ

```
# 1. Feature & Target 구성
tmp = all_factors.copy()
tmp[target_label] = targets_df[target_label]
tmp = tmp.dropna()
X = tmp[features]
y = tmp[target_label]

X_train, X_valid, X_test = split_by_index(X, 0, [0.6, 0.2, 0.2])
y_train, y_valid, y_test = split_by_index(y, 0, [0.6, 0.2, 0.2])
```

```
# 2. Random Forest Classifier 구성, fitting
clf = RandomForestClassifier(
    n_estimators=10,
    max_features='sqrt',
    min_samples_split=5000,
    bootstrap=True,
    oob_score=True,
    n_jobs=-1,
    criterion='entropy',
    verbose=0,
    random_state=0
)
clf.fit(X_train, y_train)
```



```

# 3. importance 및 rank 구성
def model_importances(m, features):
    # TODO: get the feature importances from the model
    importances = m.feature_importances_

    # TODO: sort the importances in descending order, and store the indices of that sort
    indices = np.argsort(importances)[::-1]
    """
    Iterate through the features, starting with the ones with the highest feature importances
    """
    features_ranked = []
    for f in range(X_train.shape[1]):
        print("%d. %s (%d) (%f)" % (f+1, features[indices[f]], indices[f], importances[indices[f]]))
        features_ranked.append(features[indices[f]])

    return features_ranked

features_skl = model_importances(clf, features)

```