Best Model?

실습 코드 : https://bit.ly/3QdGwGq

2022.06.14.

한국에너지기술연구원 계산과학연구실

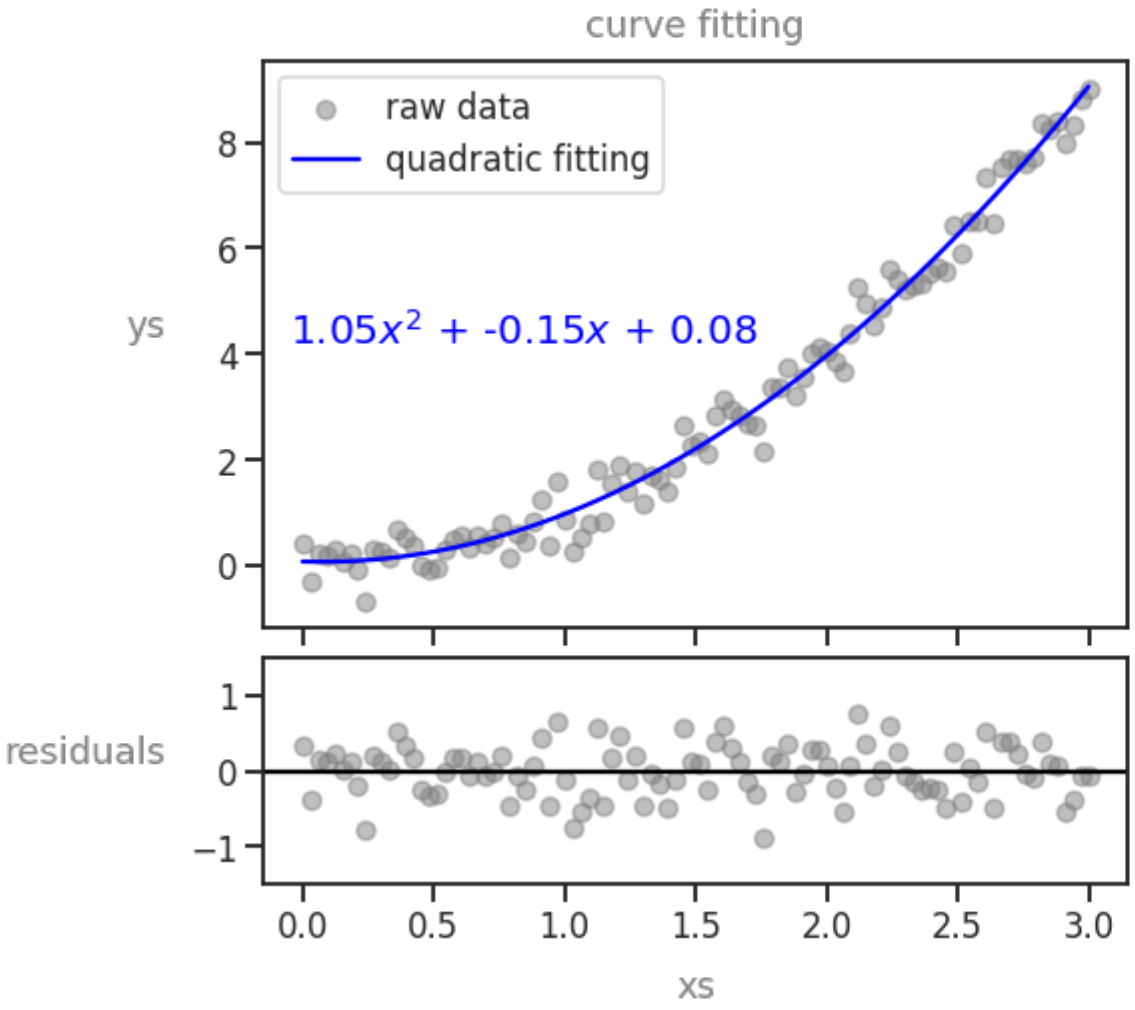
이제현

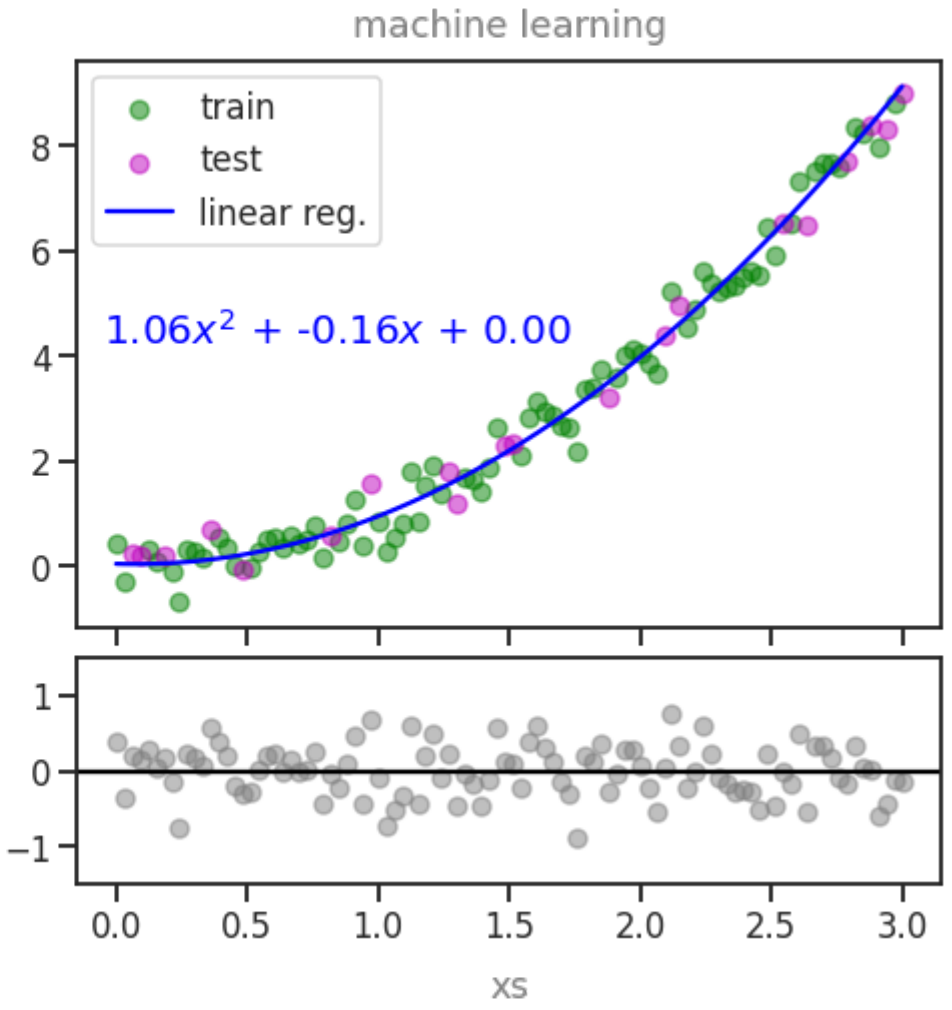
2022 에너지+AI 학습 조직 : 총 9회 + α

2차 모임 (5월) **김준우** **(기술경영)**	**3차 모임 (6월)** **한광우** **(에너지 시스템 운영)**	**4차 모임 (7월)** **윤민혜** **(전문가 초청)**	**5차 모임 (8월)** **손은국** **(연구노트)**
6차 모임 (8월) **김보영** **(주제 미정)**	**7차 모임 (9월)** **주영환** **(위상설계 최적화)**	**8차 모임 (10월)** **이효진** **(실험 설계)**	**9차 모임 (11월)** **오명찬** **(전문가 초청)**

- **발표자 :** 1. 개인 일정에 따라 당사자간 합의 하에 양도 가능,
  2. 외부 연사 초빙 가능.
- **발표 :** 자료 및 발표 영상 원내 공개 기본. 발표자 재량에 따라 비공개 및 유튜브 등 외부 공개 가능.

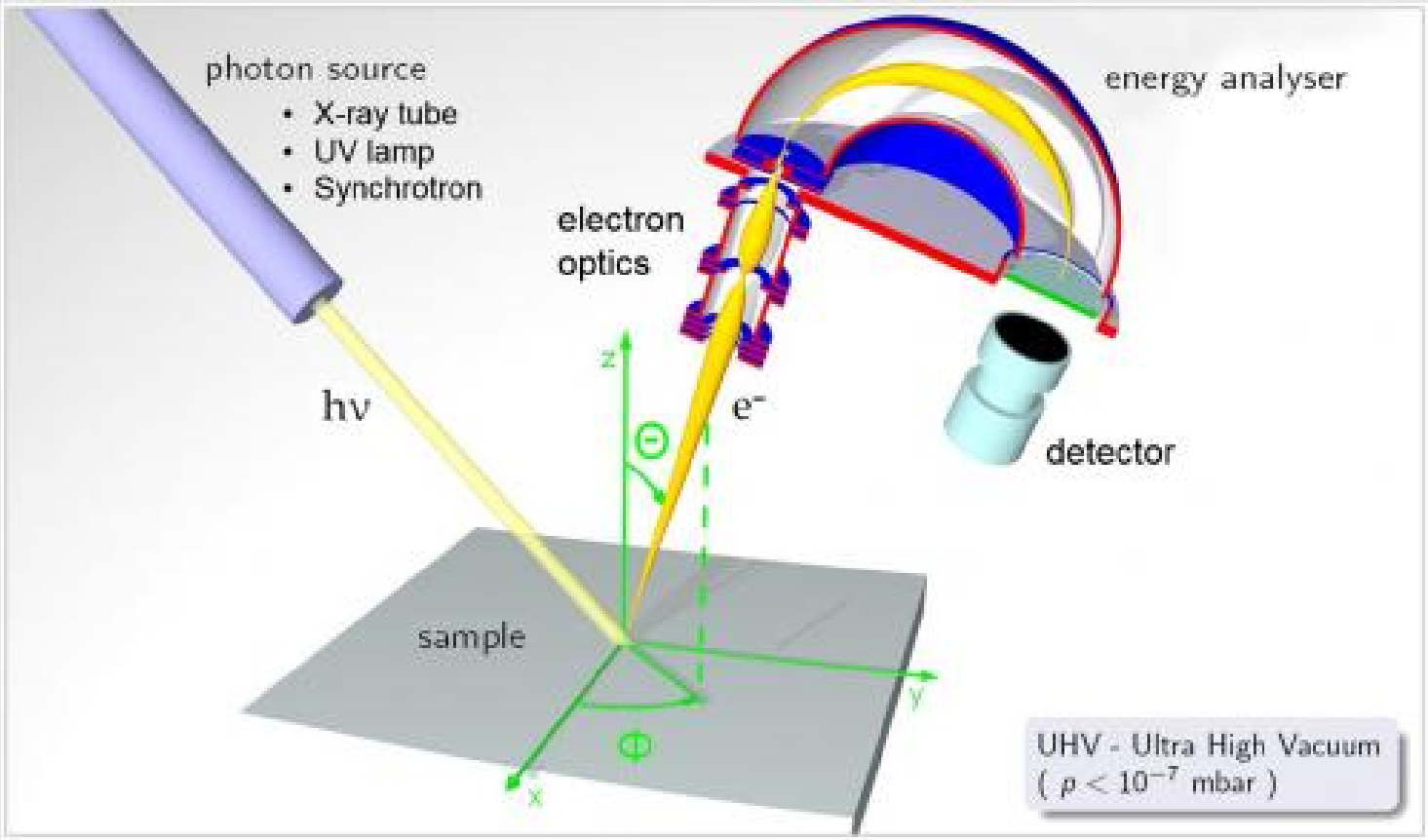
Fitting vs Machine learning





Fitting

- equation = 물리, 화학 등 분야별 전문 지식 활용. 데이터로 coefficient만 결정
  - **XPS peak : 물질 표면에 X-ray를 쏘아 결합이 끊어진 전자**secondary electron**의 에너지 = 결합 에너지 측정**



https://ywcmatsci.yale.edu/xps

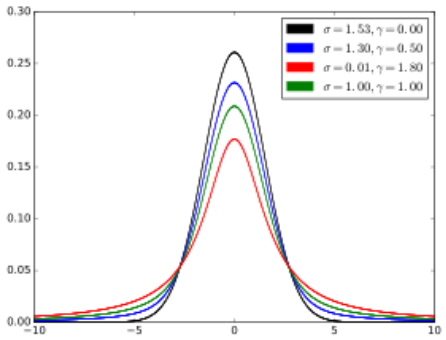
Fitting

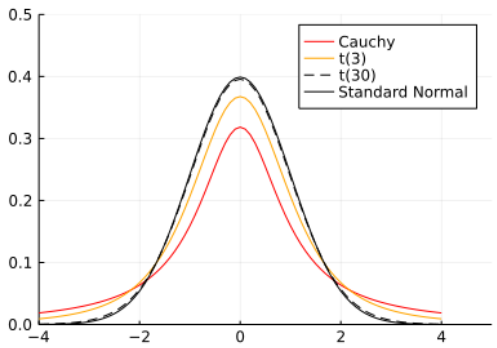
- equation = 물리, 화학 등 분야별 전문 지식 활용. 데이터로 coefficient만 결정
  - **XPS peak : Voigt** (convolution of **Gaussian**Phonon broadening & **Lorentzian**Uncertainty principle)

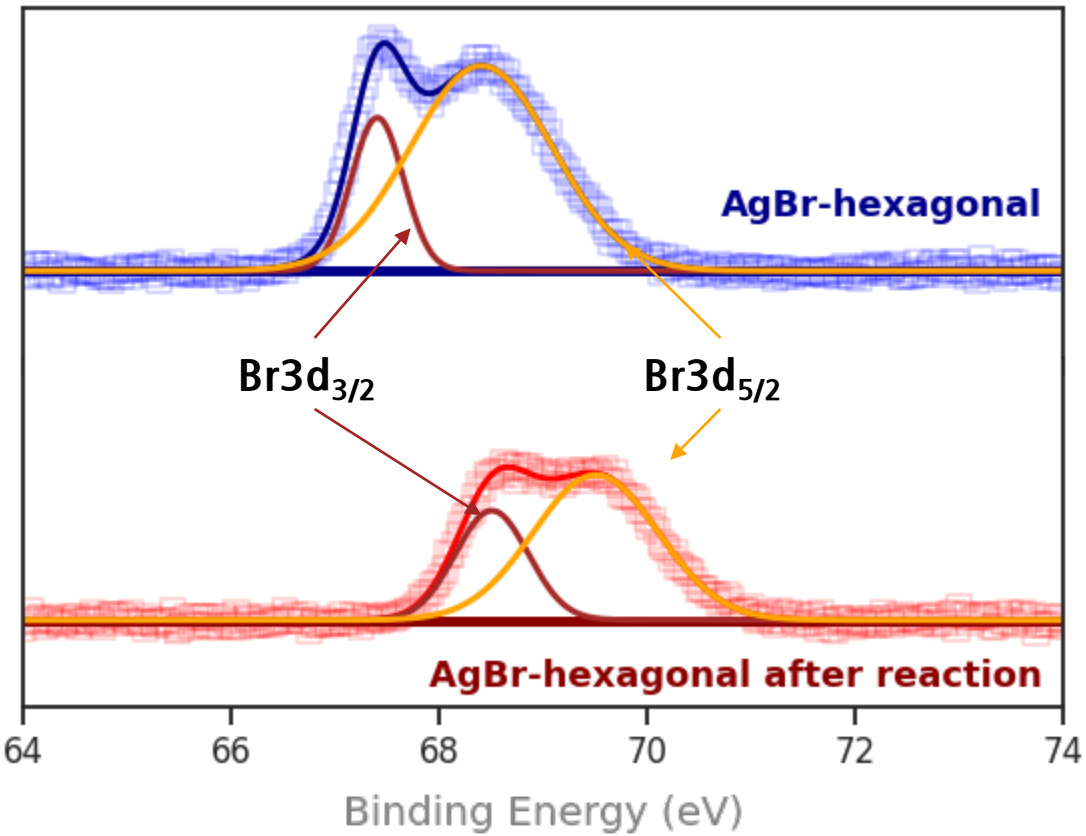
$$V(x;\sigma,\gamma) \equiv \int_{-\infty}^{\infty} G(x';\sigma)L(x-x';\gamma)\,dx',$$

$$G(x;\sigma) \equiv \frac{e^{-x^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}},$$

$$L(x;\gamma) \equiv \frac{\gamma}{\pi(x^2+\gamma^2)}.$$





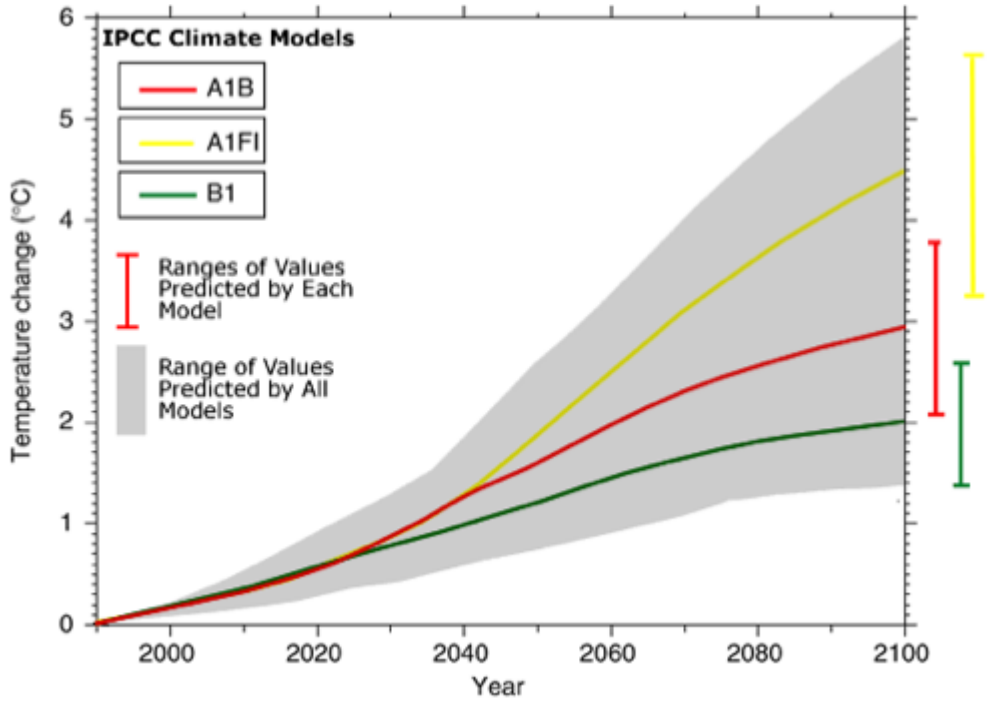


H. Zhang et al., RSC Advances 7 25725 (2017), DOI: 10.1039/c7ra02354e Fig 2c를 다시 그림, https://en.wikipedia.org/wiki/Voigt_profile, https://en.wikipedia.org/wiki/Cauchy_distribution

Fitting

- Model 활용 분석, 예측 : 설명력 강화

기후 변화 모델



http://wareslab.genetics.uga.edu/resources/Honors-Evol-Climate/pisaster.genetics.uga.edu/pisaster.genetics.uga.edu/groups/evolution3000/wiki/2ecdd/Climate_Change_Models.html

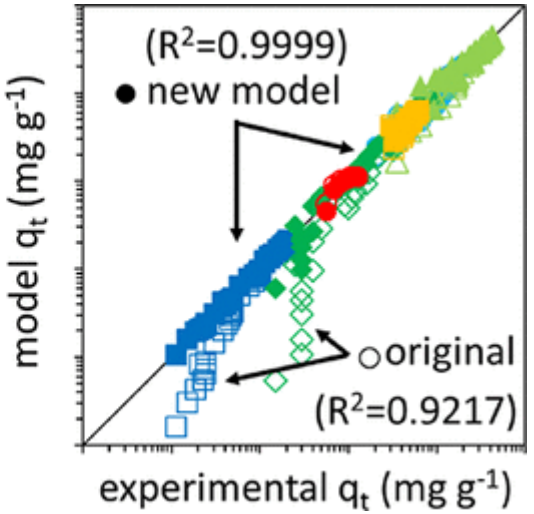
화학 반응 모델

original PSO model

$$\frac{dq_t}{dt} = k_2(q_e - q_t)^2$$

revised PSO model

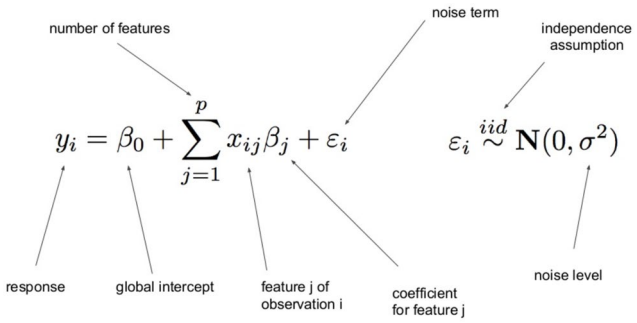
$$\frac{dq_t}{dt} = k'C_t\left(1 - \frac{q_t}{q_e}\right)^2$$

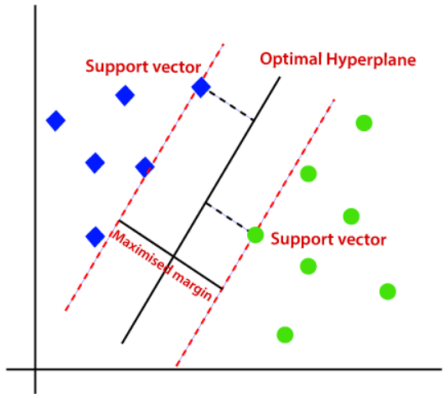


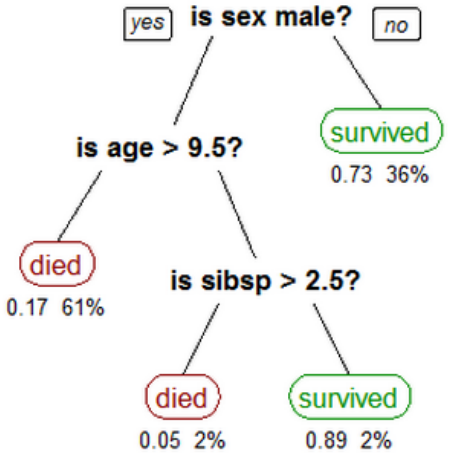
J. C. Bullen et al., Langmuir 2021, 37, 10, 3189-3201, https://doi.org/10.1021/acs.langmuir.1c00142

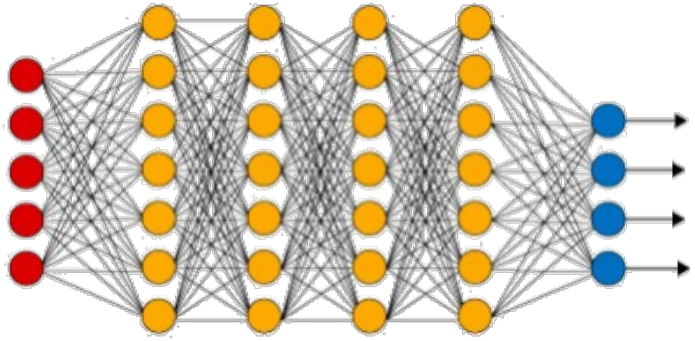
Machine Learning

- **모델 자체가 미지수.** 데이터의 패턴을 가장 잘 설명하는 모델과 hyperparameter 조합 탐색.
  - 지배 방정식이 존재하는 현상이라면 적절한 **파생 변수**를 만들어 Fitting처럼 방정식 활용 가능
  - **Linear Regression :** X 인자들의 선형 결합으로 y를 설명
  - **Support vector :** X 데이터가 존재하는 공간을 가상의 면hyperplane으로 나누어 y를 설명
  - **Tree Model :** X 데이터의 인자와 값에 따라 가지를 치며 y를 찾아감
  - **Neural Network :** X 인자의 선형 변환과 비선형 변환은 반복하여 y를 도출함







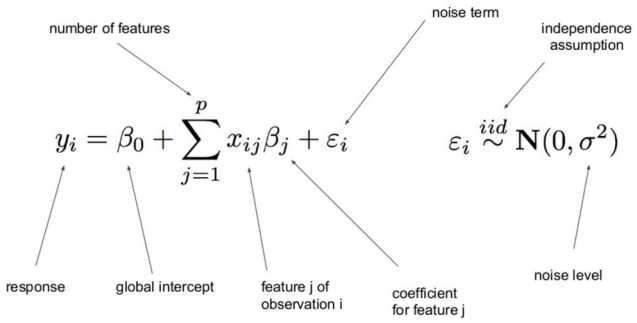


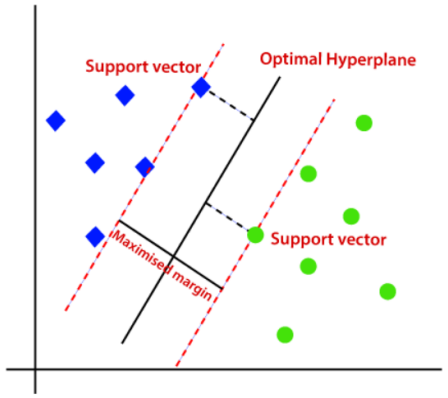
Machine Learning

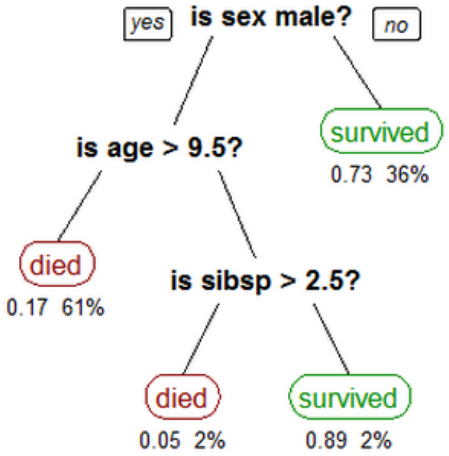
- **모델 자체가 미지수.** 데이터의 패턴을 가장 잘 설명하는 모델과 hyperparameter 조합 탐색.
  - 어떤 모델을 선택하건 가설을 수립하고 데이터로 학습하여 검증하는 단계로 구성됨
  - **가설 수립 : "이 데이터의 패턴은 이 모델이 잘 찾을 거야" → model selection**
  - **데이터 학습 : "모델의 parameter를 데이터에 맞게 맞춰 보자" → *"fitting"***
  - **가설 검증 : "학습에 사용하지 않은 데이터도 잘 설명하는지 확인해 볼까?" → validation**

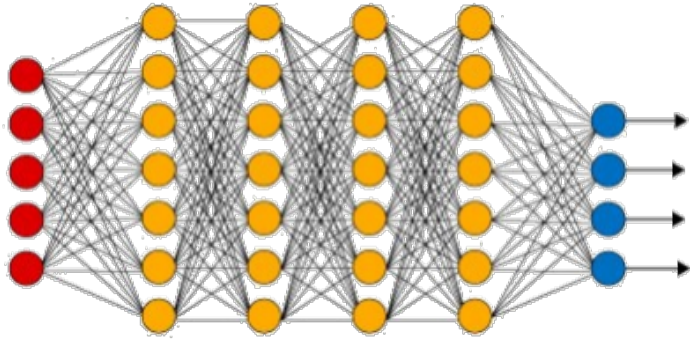
Q1. 어떤 기준으로 확인하지? R2? RMSE? MAE? → "metric"

Q2. 전체 데이터 중 어떤 데이터로 학습하고 어떤 데이터로 검증하지? → "split"





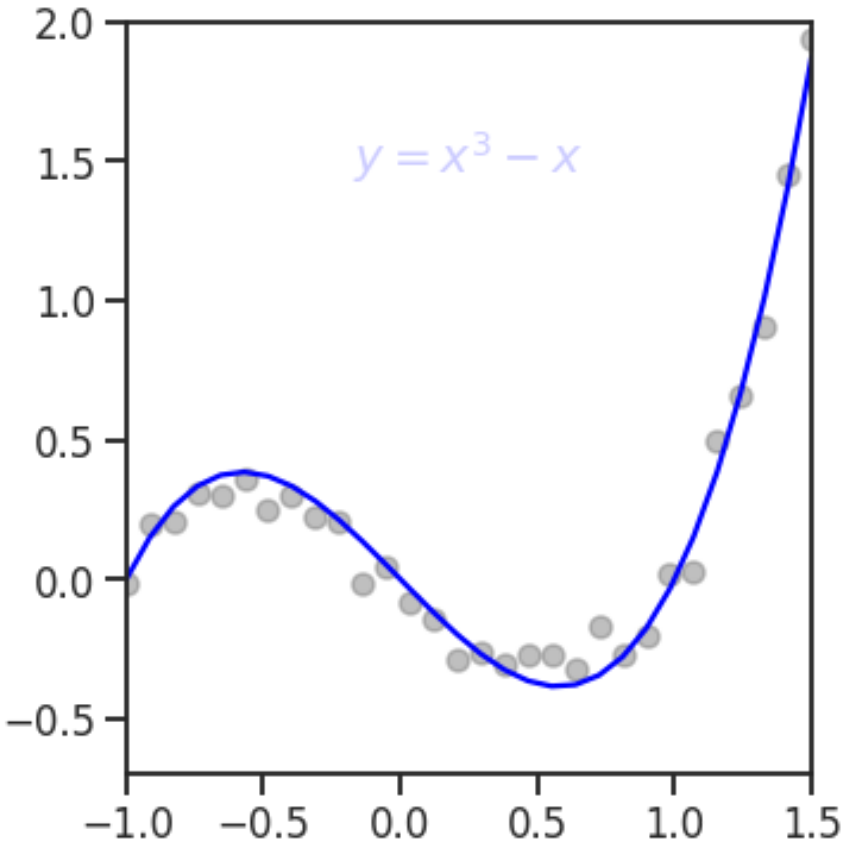




그리하여, 오늘의 주제 등장

fitting & validation

이런 데이터의 패턴을 찾고 있습니다.

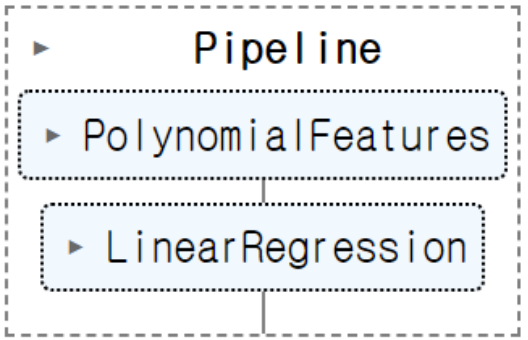


데이터 30개가 있습니다.
80%로 학습하고 20%로 검증합니다.

다항식 같습니다. 선형 모델 파이프라인을 만듭니다.

- 차수는 변수로 바꿀 수 있게 만들었습니다.
- Metric은 RMSE로 합니다. 작을수록 좋습니다.

```
def get_model(degree=1):
    model  = Pipeline([("polynomial", PolynomialFeatures(degree=degree)),
                       ("linearregression", LinearRegression())])
    return model
```

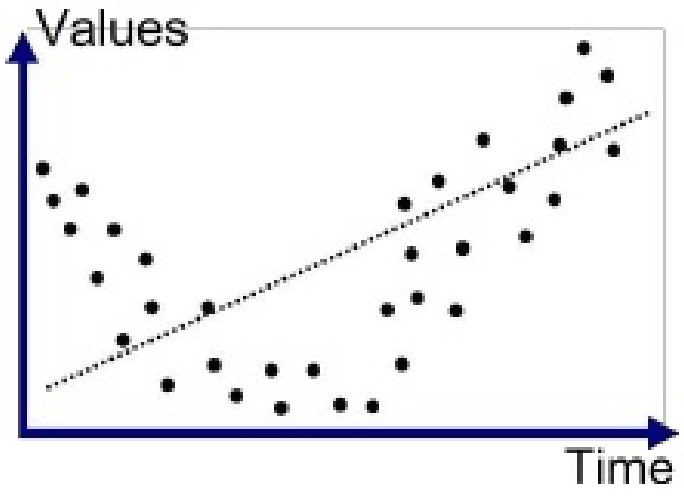


$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

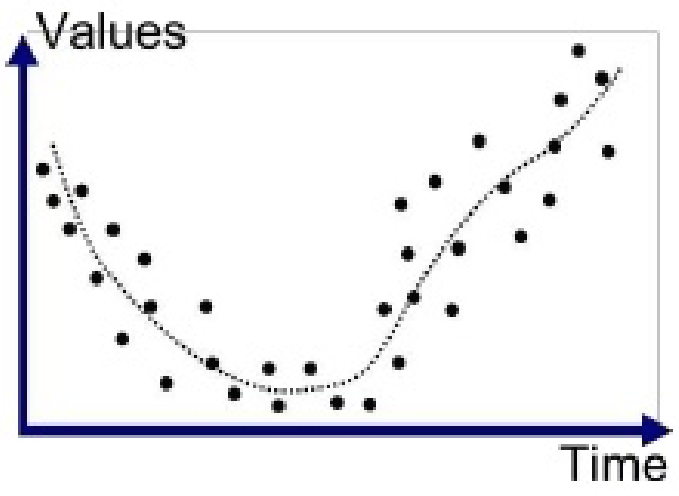
차수를 높여가며 찾아 봅니다.

- **1차 함수 :** 학습과 검증 데이터셋 모두 성능이 부족합니다. **Underfitting**이라고 합니다.
- **3차 함수 :** 학습과 검증 데이터셋 모두 성능이 우수합니다. **Just fit**이라고 합니다.
- **20차 함수 :** 학습은 기가 막힙니다. 그런데 검증이 문제입니다. **overfitting**이라고 합니다.

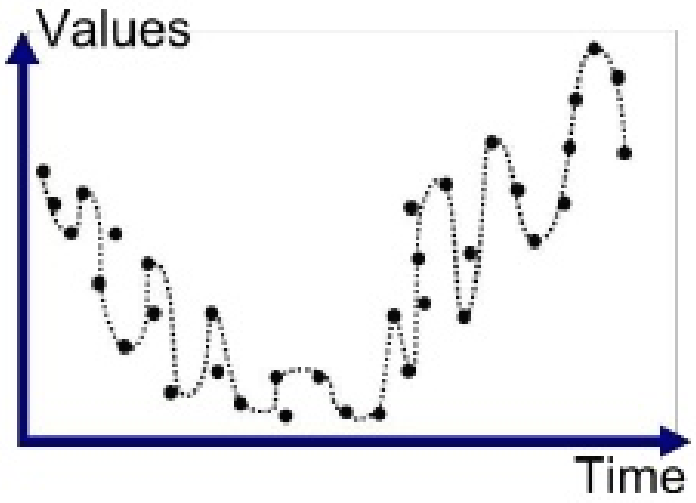
다른 표현으로 : Bias & Variance



high bias
low variance

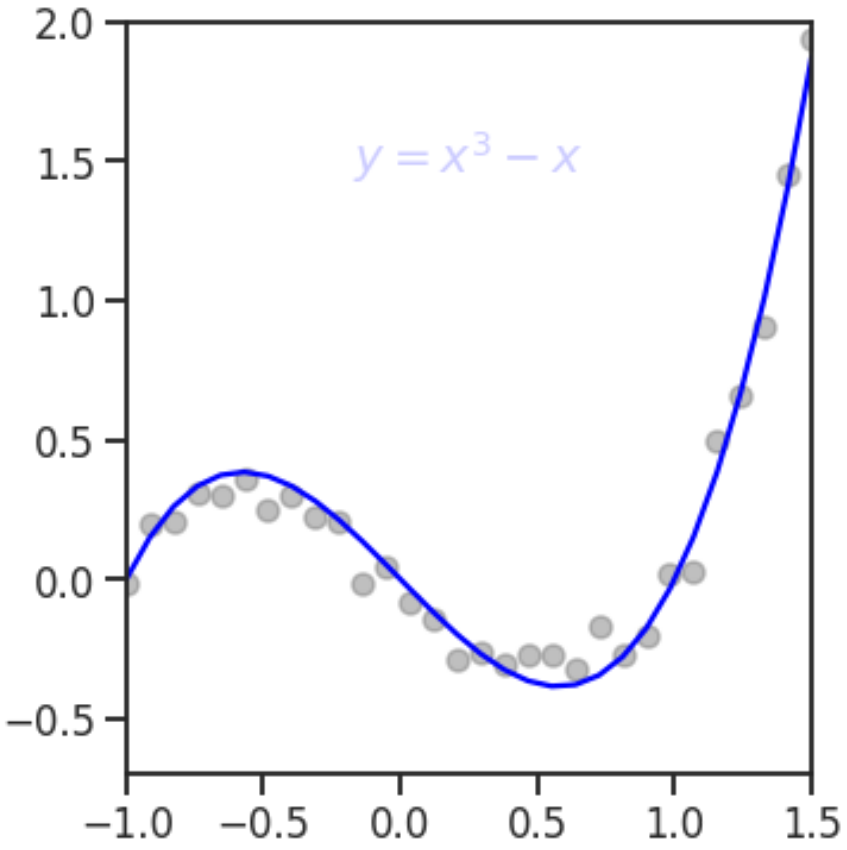


medium bias
medium variance



low bias
high variance

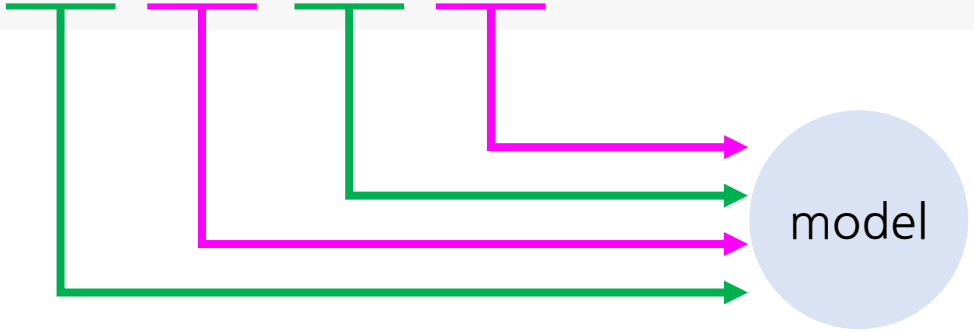
잠깐 아까 이 페이지를 다시 봅니다.



데이터 30개가 있습니다.
80%로 학습하고 20%로 검증합니다.

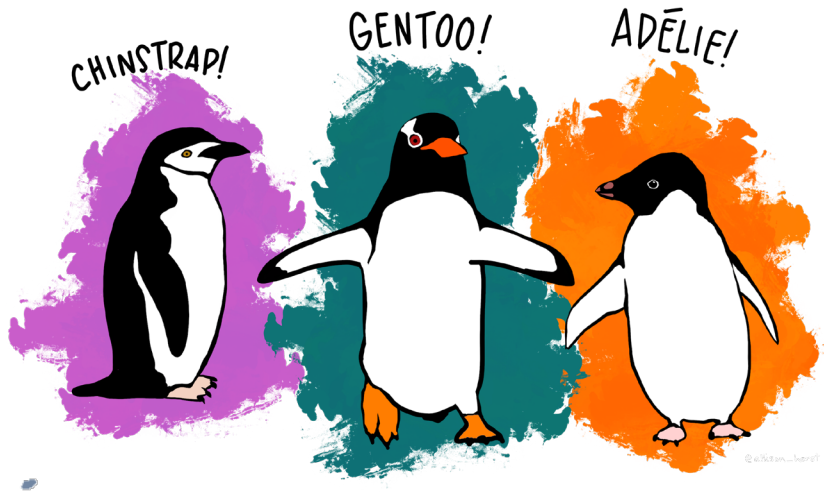
보통 random sampling 으로 train과 test 분할

```
from sklearn.model_selection import train_test_split

x_train, x_val, y_train, y_val = train_test_split(xs, ys, test_size=0.2)
```

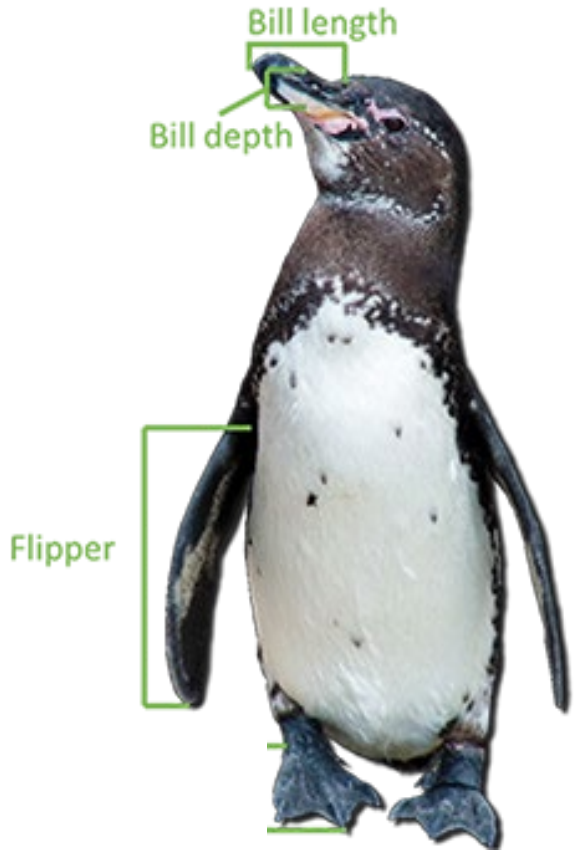


모델 성능이 불충분하면 hyperparameter를 바꾸고 학습합니다.
"degree" parameter: *a*, *b*, *c*, …
학습 데이터와 검증 데이터 분할에 따라 학습 모델이 달라집니다.
→ 검증이 제대로 됐다고 볼 수 있을까요?

예제 : 펭귄 데이터셋 분할



https://rpubs.com/julianlavila/penguins



펭귄 데이터셋 분할 = 8:2

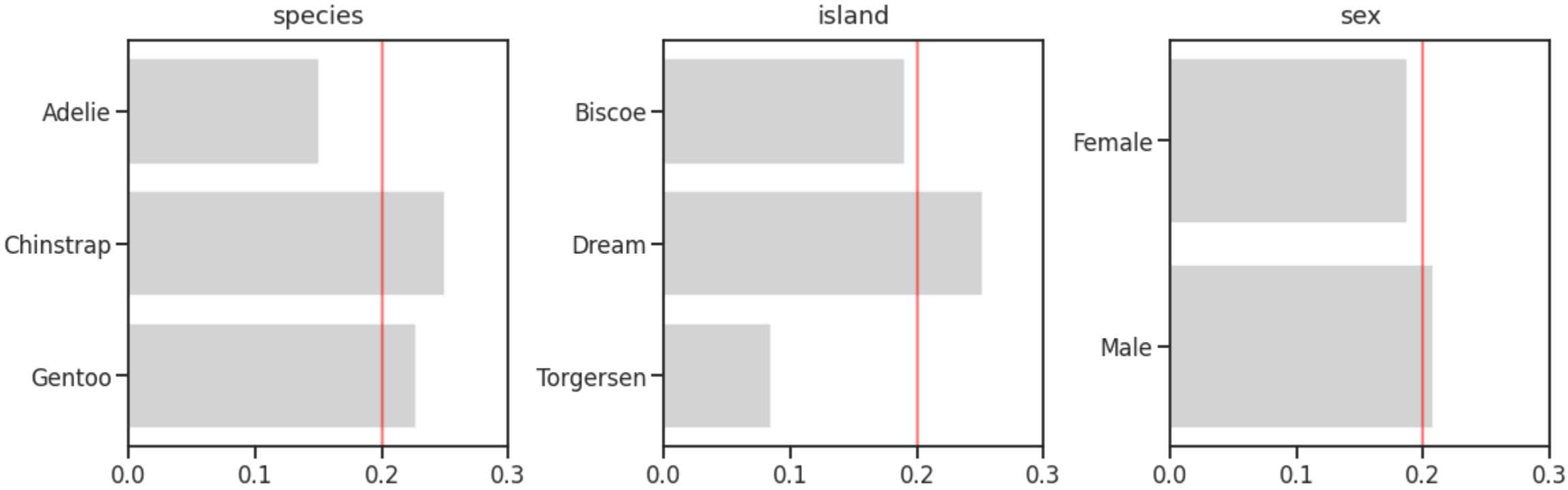
```
df_peng = sns.load_dataset("penguins")
df_peng = df_peng.dropna()
df_peng.head()
```

	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	Male

```
y = df_peng["body_mass_g"]
X = df_peng.drop("body_mass_g", axis=1).reset_index()   # 결측치 제거 후 index reset

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2)  # 8:2 분할
```

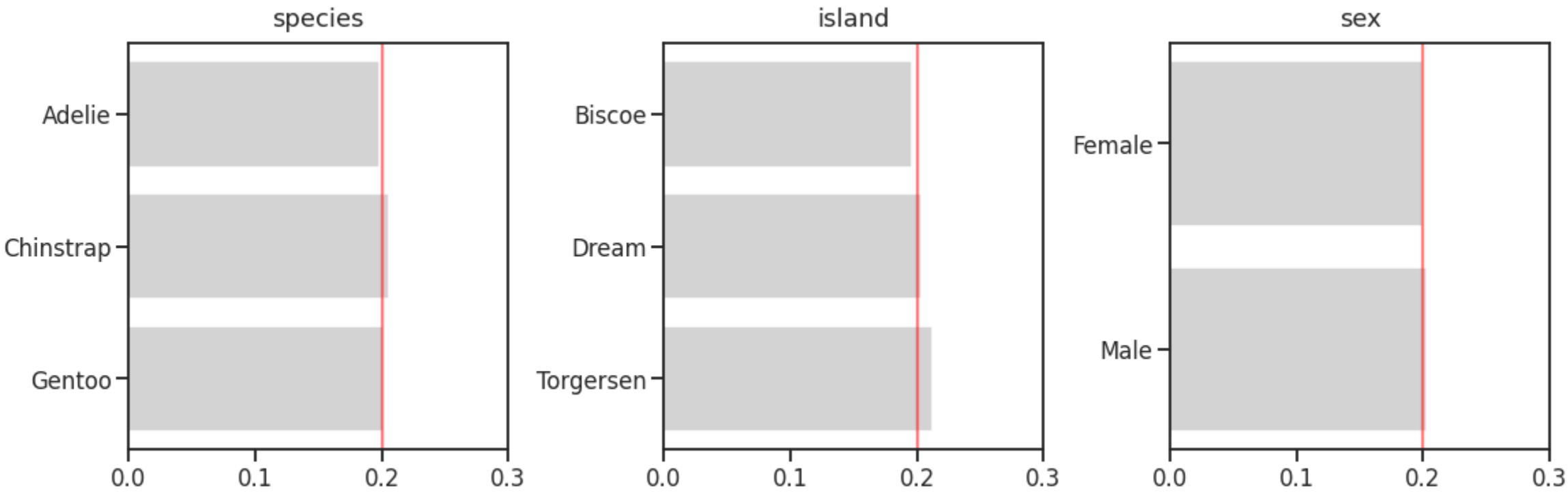
펭귄 데이터셋 분할 = 8:2 ?

- train : validation 데이터간 불균형 발생 → 제대로 된 학습과 평가가 어려움
- class 비율을 최대한 맞추어 데이터를 분할해야 함.



Stratified Sampling : 층화 추출

```
1 X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
2                                                   stratify=X[["species", "island", "sex"]])
```



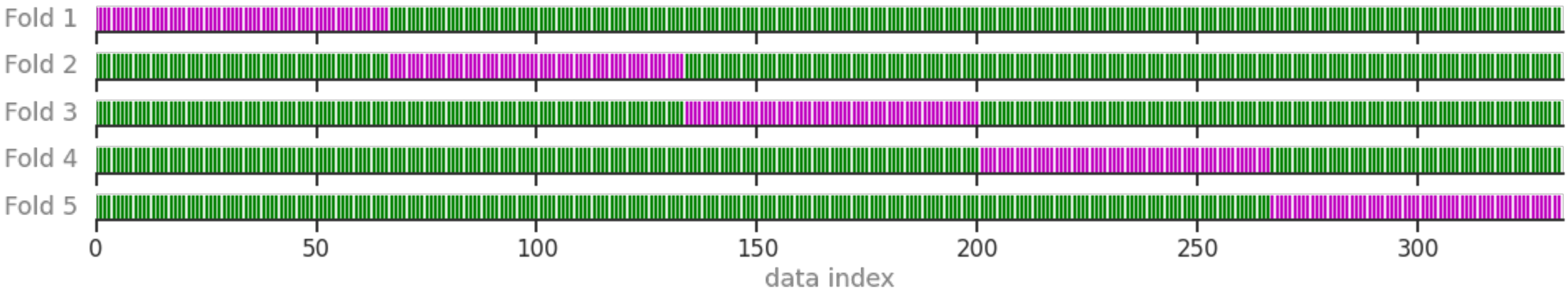
Cross Validation : 교차 검증

- 모든 데이터를 학습에 활용
  - 80%를 학습에 투입 + 20%로 학습 결과를 검증

Cross Validation 데이터 분할

- scikit-learn의 KFold 사용

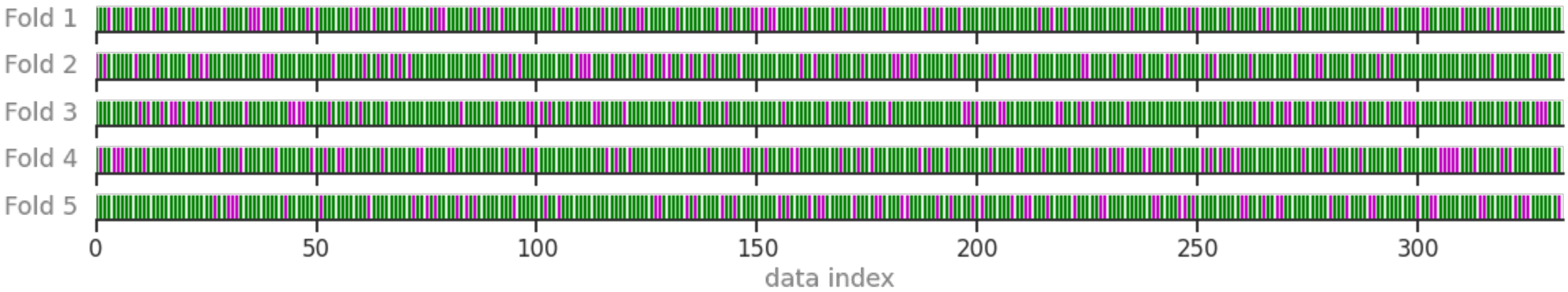
```python
from sklearn.model_selection import KFold

kf = KFold(n_splits=5)                  # 5-Fold 데이터 분할

X_trains, X_vals = [], []
y_trains, y_vals = [], []

for idx_train, idx_val in kf.split(X):  # 분할: index
    X_train, X_val = X.iloc[idx_train], X.iloc[idx_val]
    y_train, y_val = y.iloc[idx_train], y.iloc[idx_val]

    X_trains.append(X_train)
    X_vals.append(X_val)
    y_trains.append(y_train)
    y_vals.append(y_val)
```



Cross Validation 데이터 혼합 분할

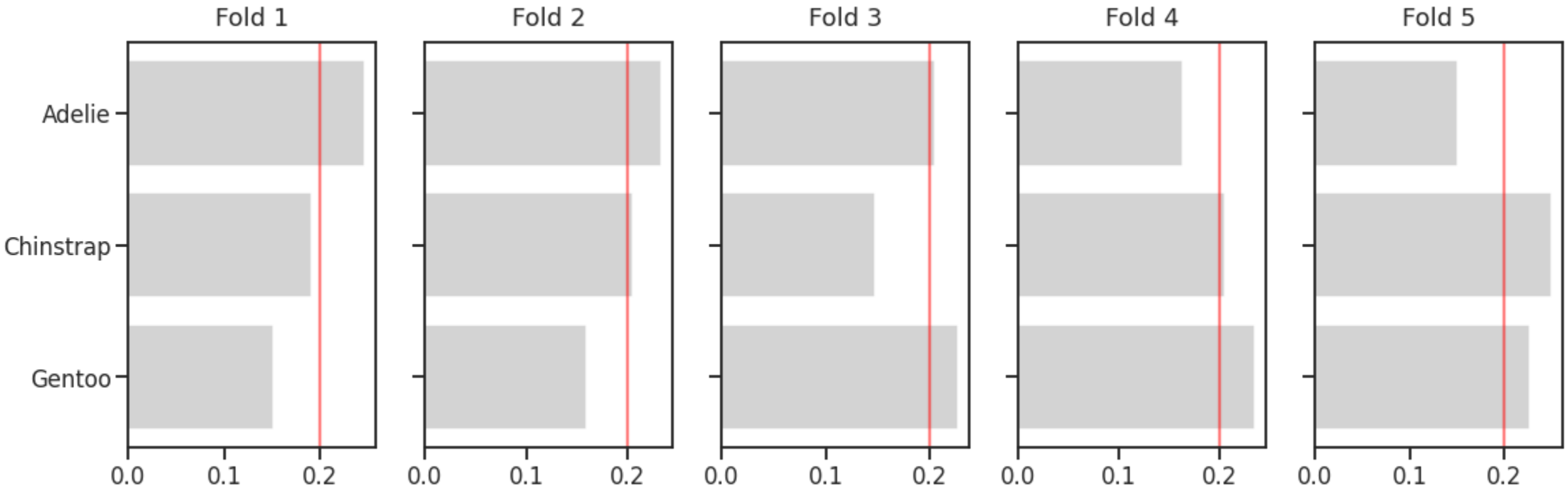
- scikit-learn의 KFold 사용, shuffle=True

```python
from sklearn.model_selection import KFold

kf = KFold(n_splits=5, shuffle=True)    # 5-Fold 데이터 분할, 데이터 섞기

X_trains, X_vals = [], []
y_trains, y_vals = [], []

for idx_train, idx_val in kf.split(X):  # 분할: index
    X_train, X_val = X.iloc[idx_train], X.iloc[idx_val]
    y_train, y_val = y.iloc[idx_train], y.iloc[idx_val]

    X_trains.append(X_train)
    X_vals.append(X_val)
    y_trains.append(y_train)
    y_vals.append(y_val)
```



Cross Validation 데이터 혼합 분할

- Fold별 species class 비율



Cross Validation 데이터 stratified sampling

- scikit-learn의 StratifiedKFold 사용, shuffle=True, stratifying features 입력

```python
from sklearn.model_selection import StratifiedKFold

skf = StratifiedKFold(n_splits=5, shuffle=True) # 5-Fold 데이터 분할, 데이터 섞기

X_trains, X_vals = [], []
y_trains, y_vals = [], []

for idx_train, idx_val in skf.split(X, X[["species", "island", "sex"]]): # 분할: index
    X_train, X_val = X.iloc[idx_train], X.iloc[idx_val]
    y_train, y_val = y.iloc[idx_train], y.iloc[idx_val]

    X_trains.append(X_train)
    X_vals.append(X_val)
    y_trains.append(y_train)
    y_vals.append(y_val)
```

오류 발생: 여러 column 동시 입력 불가

```
3 frames
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py in _make_test_folds(self, X, y)
    652             raise ValueError(
    653                 "Supported target types are: {}. Got {!r} instead.".format(
--> 654                     allowed_target_types, type_of_target_y
    655                 )
    656             )

ValueError: Supported target types are: ('binary', 'multiclass'). Got 'multiclass-multioutput' instead.

SEARCH STACK OVERFLOW
```

Cross Validation 데이터 stratified sampling

- scikit-learn의 StratifiedKFold 사용, shuffle=True, stratifying features 입력

```python
from sklearn.model_selection import StratifiedKFold

skf = StratifiedKFold(n_splits=5, shuffle=True)     # 5-Fold 데이터 분할, 데이터 섞기

X_trains, X_vals = [], []
y_trains, y_vals = [], []
X_stratify = X[["species", "island", "sex"]].apply(lambda x: f"{x[0]}_{x[1]}_{x[2]}", axis=1)

for idx_train, idx_val in skf.split(X, X_stratify): # 분할: index
    X_train, X_val = X.iloc[idx_train], X.iloc[idx_val]
    y_train, y_val = y.iloc[idx_train], y.iloc[idx_val]

    X_trains.append(X_train)
    X_vals.append(X_val)
    y_trains.append(y_train)
    y_vals.append(y_val)
```

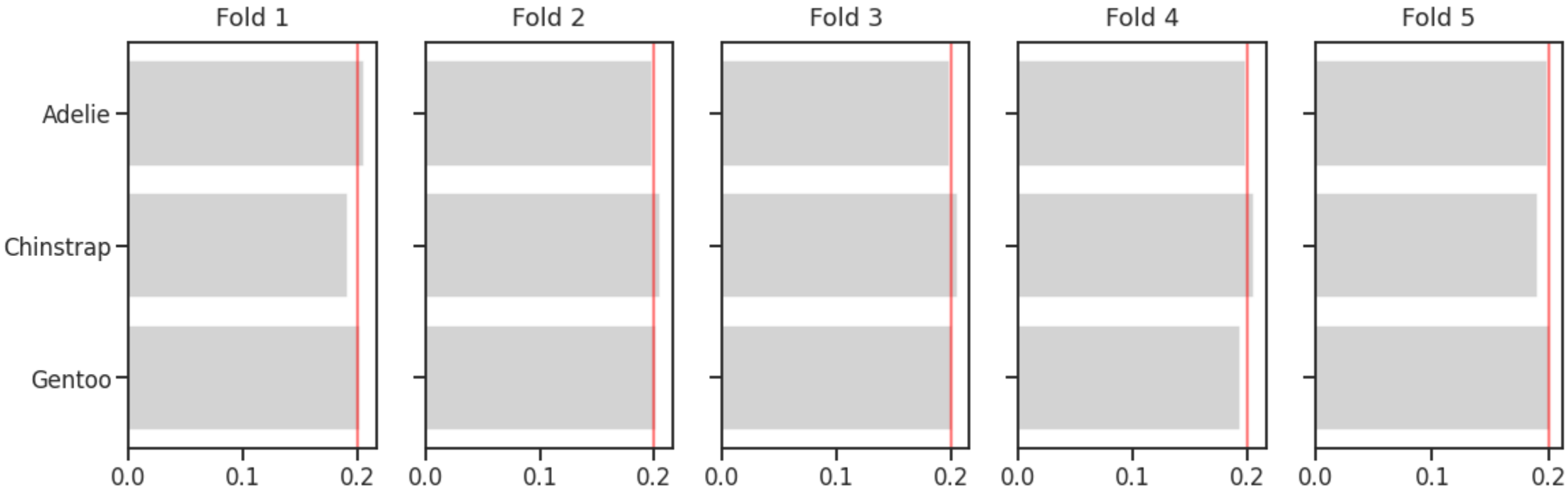
오류 회피: 여러 column 병합 데이터 생성

```python
X_stratify = X[["species", "island", "sex"]].apply(lambda x: f"{x[0]}_{x[1]}_{x[2]}", axis=1)
X_stratify
```

```
0         Adelie_Torgersen_Male
1       Adelie_Torgersen_Female
2       Adelie_Torgersen_Female
3       Adelie_Torgersen_Female
4         Adelie_Torgersen_Male
                 ...
328       Gentoo_Biscoe_Female
329       Gentoo_Biscoe_Female
330         Gentoo_Biscoe_Male
331       Gentoo_Biscoe_Female
332         Gentoo_Biscoe_Male
Length: 333, dtype: object
```

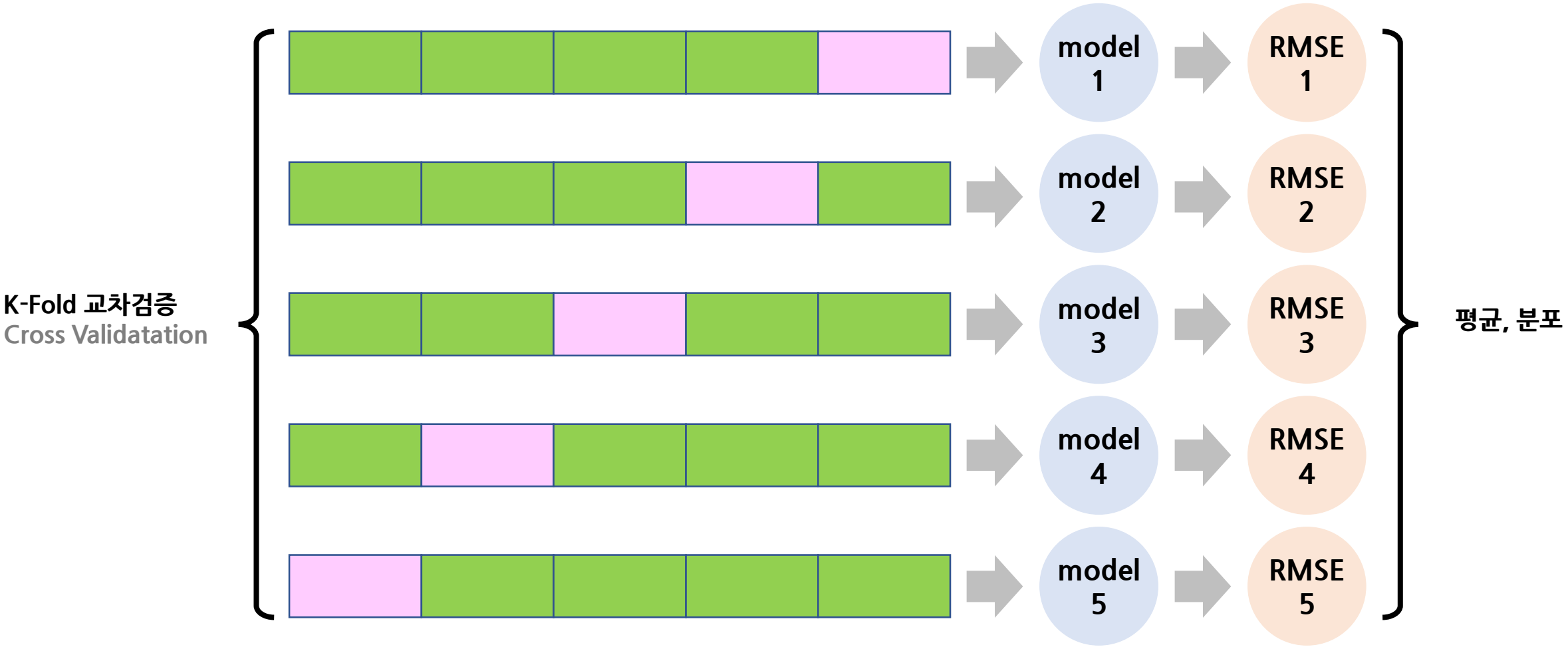
Cross Validation 데이터 혼합 분할

- Fold별 species class 비율



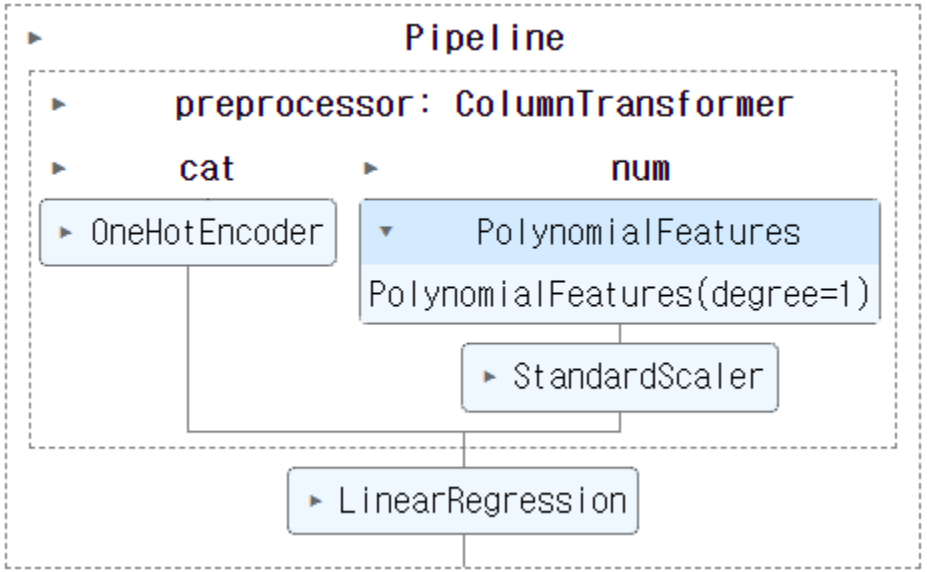
Cross Validation 평가 1

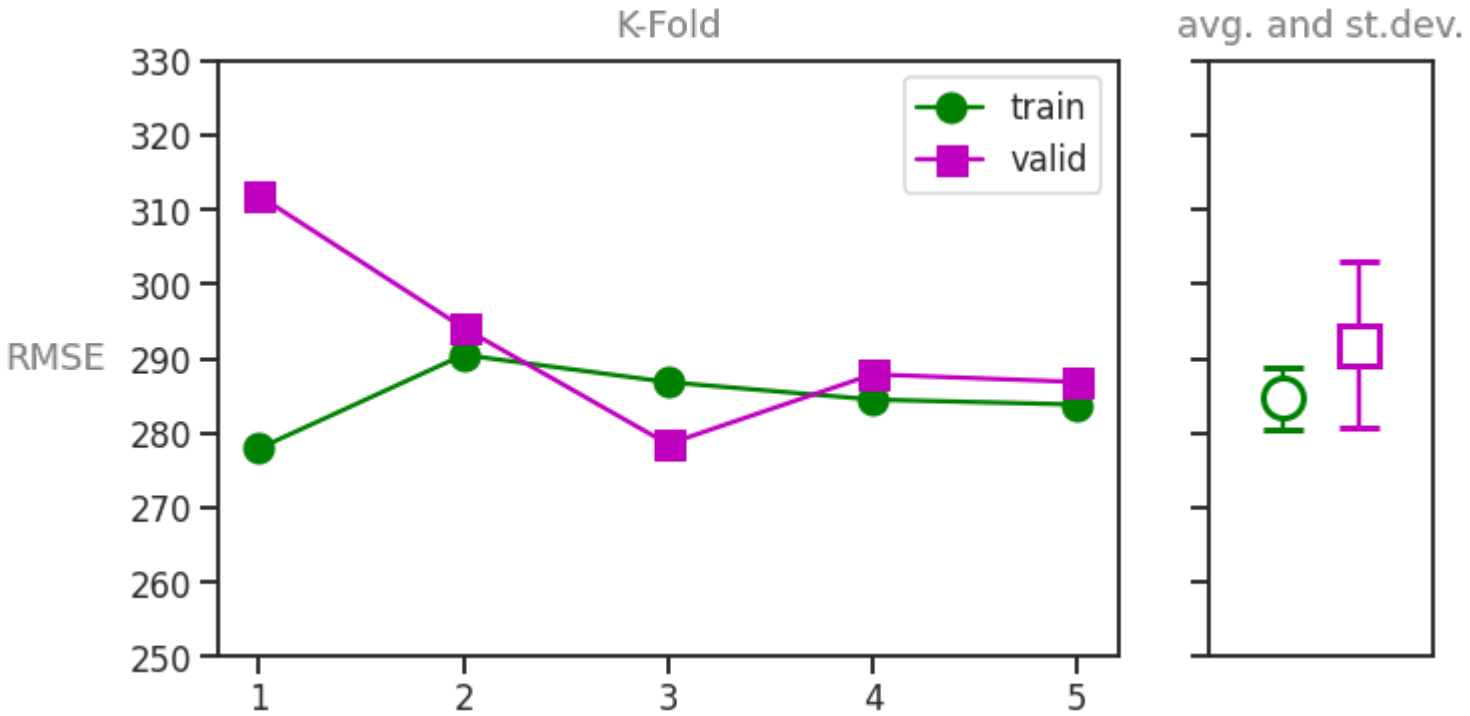
- Fold 수만큼 다른 모델이 훈련됩니다.



펭귄 체중 예측 모델 : degree = 1

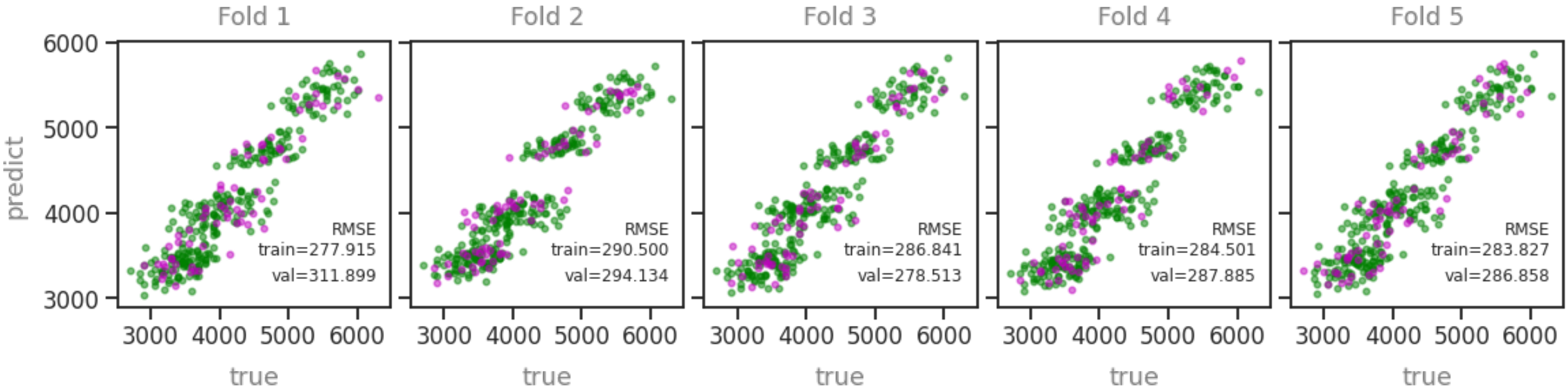
```
model_deg1 = get_model(1)                          # Pipeline으로 모델 생성
model_deg1.fit(X_trains[0], y_trains[0])           # 학습
y_pred_train = model_deg1.predict(X_train)         # train data 예측
y_pred_val = model_deg1.predict(X_val)             # test data 예측
```





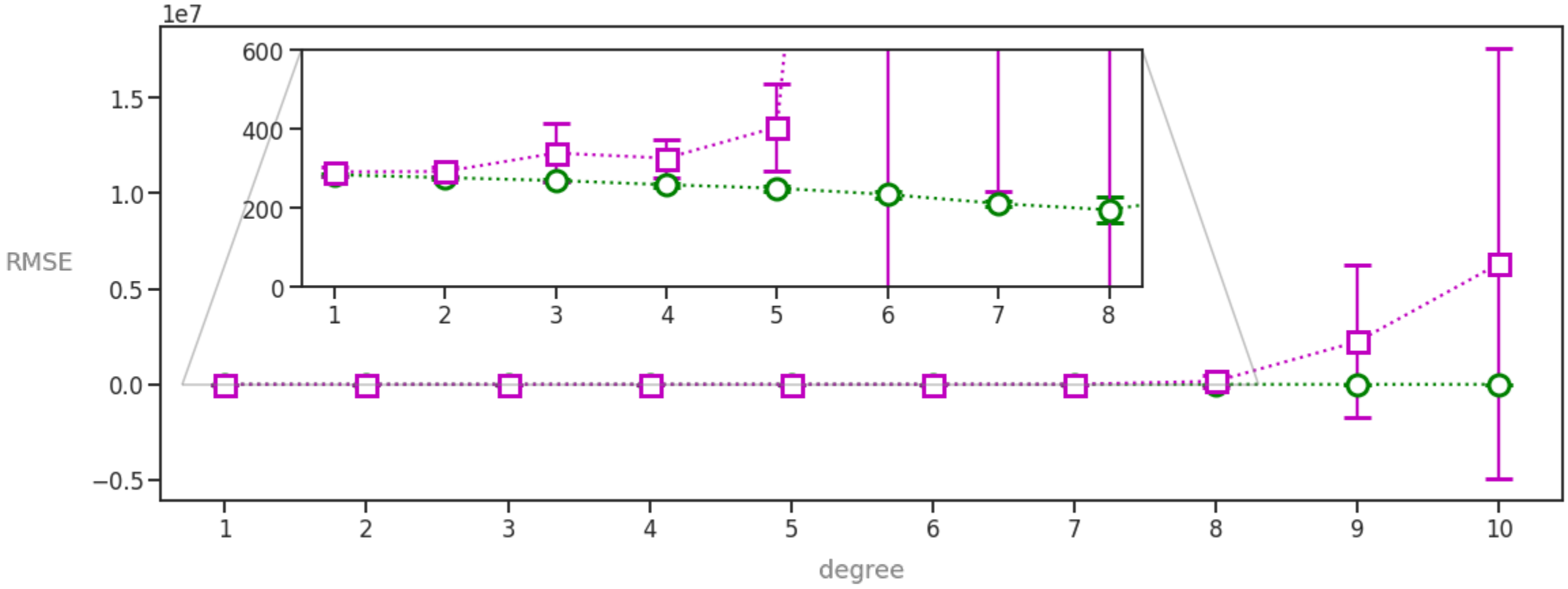
펭귄 체중 예측 모델 : degree = 1

```
model_deg1 = get_model(1)                          # Pipeline으로 모델 생성
model_deg1.fit(X_trains[0], y_trains[0])           # 학습
y_pred_train = model_deg1.predict(X_train)         # train data 예측
y_pred_val = model_deg1.predict(X_val)             # test data 예측
```



좋은 모델 찾기 : hyperparameter 찾기

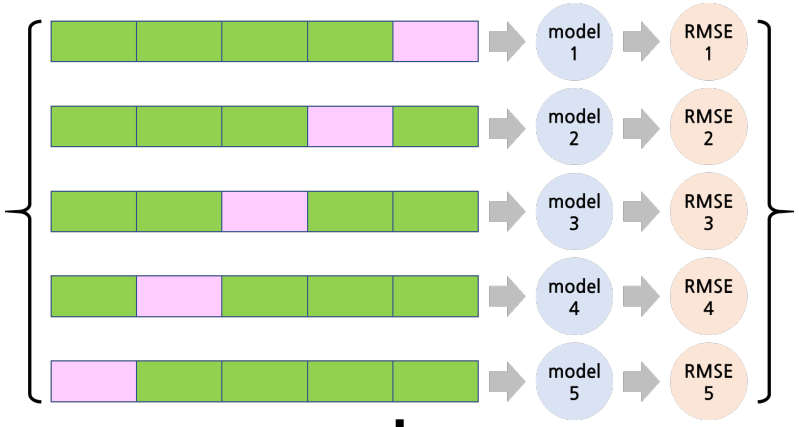
- degree = 1 ~ 10



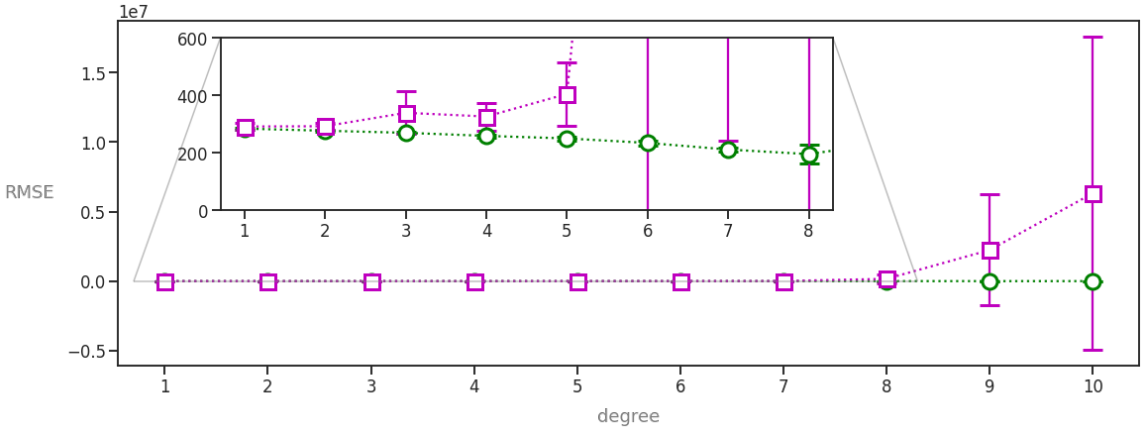
좋은 모델 찾기 : hyperparameter 찾기

- 적용한 hyperparameter 중 validation metric이 가장 좋은 것 선택 : degree = 1

K-Fold 교차검증
Cross Validatation



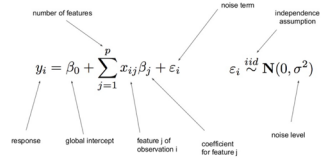
평균, 분포

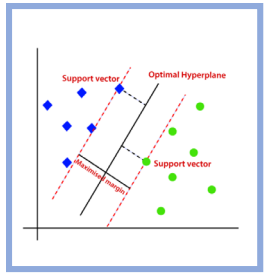


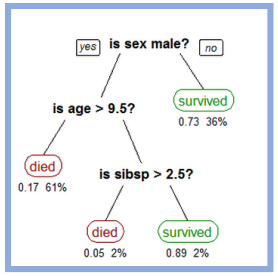
degree 1	degree 2	degree 3	degree 4	degree 5	degree 6	degree 7	degree 8	degree 9	degree 10

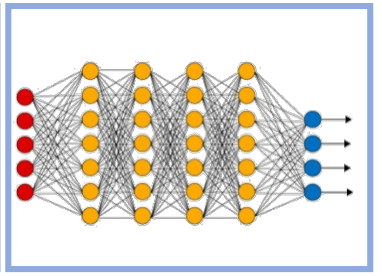
① 재학습
"refit"

② 모델 교체



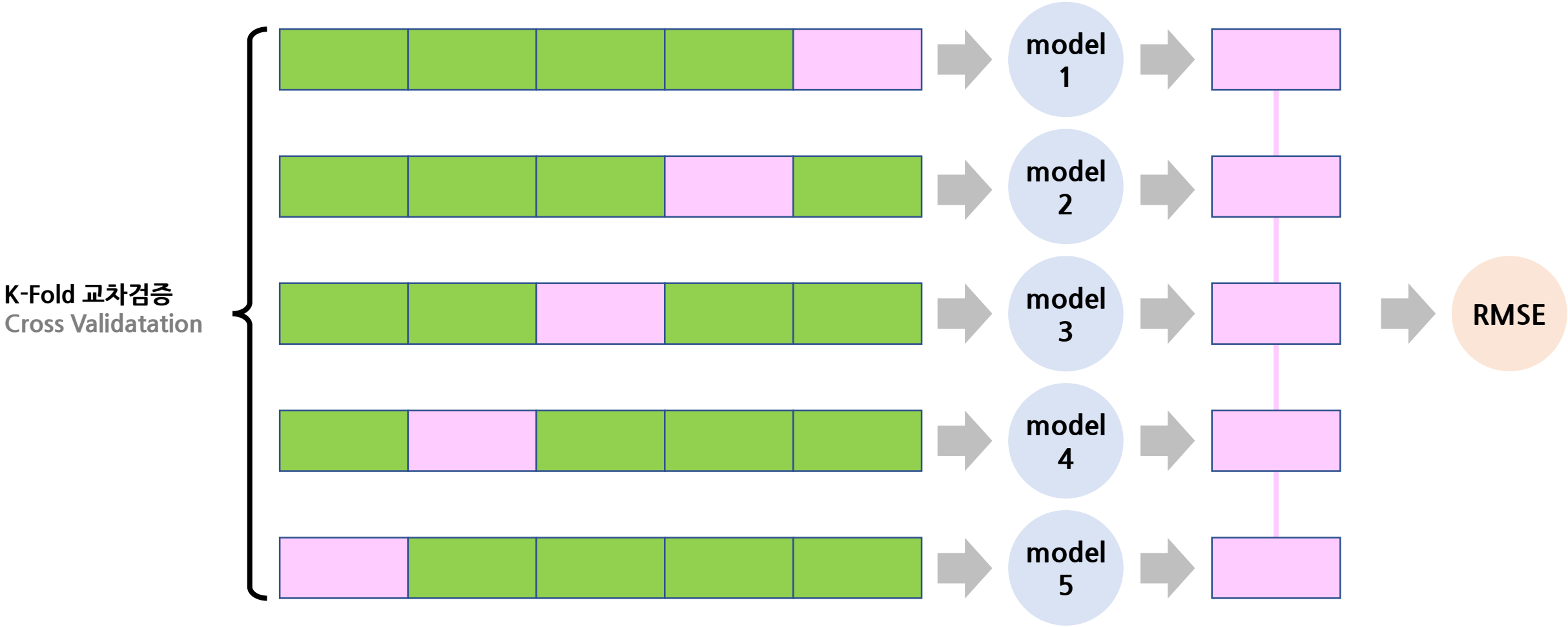






Cross Validation 평가 2

- Validation set 평가 결과 결합



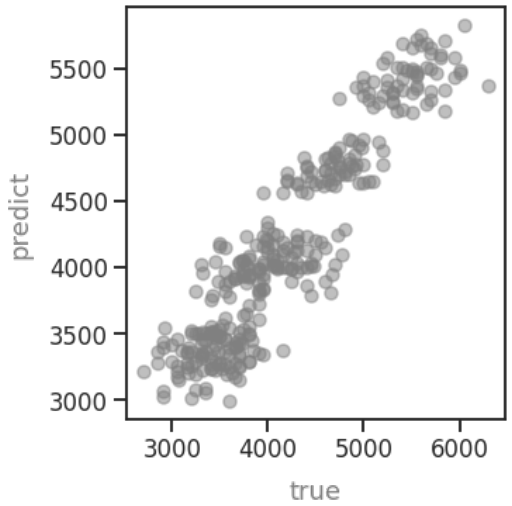
Cross Validation 평가

- 1. 모델별 평가

```python
from sklearn.model_selection import cross_val_score

model1 = get_model(1)
-cross_val_score(model1, X, y, cv=5, scoring="neg_root_mean_squared_error")
```

```
array([286.90723447, 315.77703262, 338.76071156, 306.27358314,
       232.49771288])
```

- 2. Fold별 평가, 취합




```python
from sklearn.model_selection import cross_val_predict

y_pred_cv = cross_val_predict(model1, X, y, cv=5)
fig, ax = plt.subplots(figsize=(5, 5), constrained_layout=True)
ax.scatter(y, y_pred_cv, c="gray", alpha=0.5)
ax.set_xlabel("true", color="gray", labelpad=12)
ax.set_ylabel("predict", color="gray", labelpad=12)
```

Homework

- Cross Validation 훑기
  - Machine Learning Mastery, k-fold cross validation (https://bit.ly/3Hp76Zo)

    - **Train/Test Split**: Taken to one extreme, k may be set to 2 (not 1) such that a single train/test split is created to evaluate the model.
    - **LOOCV**: Taken to another extreme, k may be set to the total number of observations in the dataset such that each observation is given a chance to be the held out of the dataset. This is called leave-one-out cross-validation, or LOOCV for short.
    - **Stratified**: The splitting of data into folds may be governed by criteria such as ensuring that each fold has the same proportion of observations with a given categorical value, such as the class outcome value. This is called stratified cross-validation.
    - **Repeated**: This is where the k-fold cross-validation procedure is repeated n times, where importantly, the data sample is shuffled prior to each repetition, which results in a different split of the sample.
    - **Nested**: This is where k-fold cross-validation is performed within each fold of cross-validation, often to perform hyperparameter tuning during model evaluation. This is called nested cross-validation or double cross-validation.

머신 러닝 강좌

- base : Scikit-learn MOOC @inria
  - 머신 러닝 위주. 딥 러닝은 skip (기회가 되면 한번쯤은 다룰 수도..?)
  - 소스 코드 포함 강의 자료 : 원내 게시판 공개
  - 강의 영상 : KIER-Tube & Youtube 공개

1차 모임 (4월) 머신러닝 기본 개념			
2차 모임 (5월) Modeling pipeline	3차 모임 (6월) Best Model?	4차 모임 (7월) Hyperparameter	5차 모임 (8월) Linear Models
6차 모임 (8월) Tree Models	7차 모임 (9월) Ensemble Models	8차 모임 (10월) CV & metrics	9차 모임 (11월) 마무리