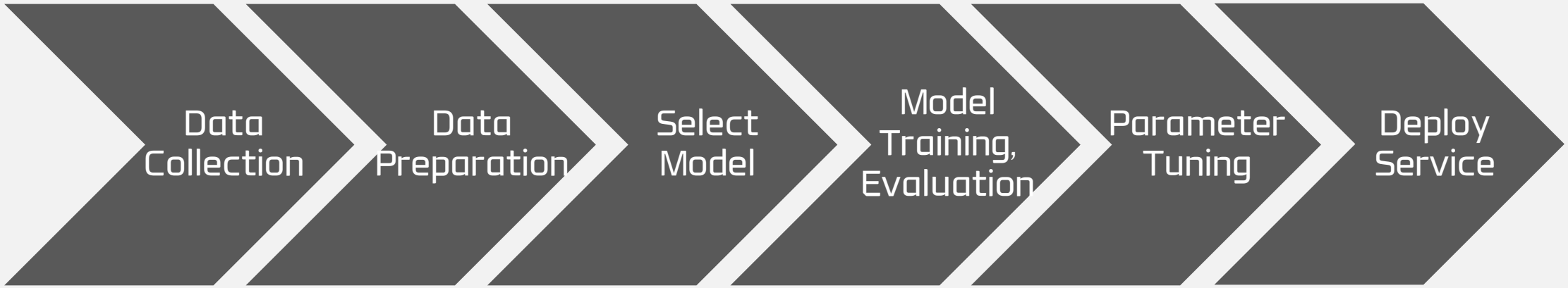
Hands On Machine Learning

Chap.02 머신러닝 프로젝트 처음부터 끝까지

Sangwon Lee

TAVE Research

Step for machine learning project



Before Start Project

문제 정의	• 비즈니스의 목적 • 예측에 사용할 특성 파악
성능 측정 지표 선택	• 예측값의 벡터와 타깃값의 벡터 사이의 거리 측정 • RMSE와 MSE가 대표적
가정 검사	• 가정을 나열 후 검사

Load Data

```
import os
import tarfile
import urllib

DOWNLOAD_ROOT = 'https://raw.githubusercontent.com/ageron/handson-ml2/master/'
HOUSING_PATH = os.path.join('datasets', 'housing')
HOUSING_URL = DOWNLOAD_ROOT + 'datasets/housing/housing.tgz'


def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    os.makedirs(housing_path, exist_ok=True)
    tgz_path = os.path.join(housing_path, 'housing.tgz')
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()
```

- Fetch_housing_data() 함수를 사용해 datasets/housing 폴더를 생성
- housing.tgz 파일을 내려받고 압축을 풀어 csv파일을 생성

Load Data

```python
import pandas as pd

def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, 'housing.csv')
    return pd.read_csv(csv_path)
```

- load_housing_data() 함수를 이용해 판다스의 데이터프레임 객체 반환

- head() 메서드와 info() 메서드를 이용해 간단하게 데이터를 확인

Load Data

```
from download_data import *

housing = load_housing_data()
print(housing.head())
print(housing.info())
------------------------------
longitude  latitude  ...  median_house_value  ocean_proximity
0    -122.23     37.88  ...            452600.0         NEAR BAY
1    -122.22     37.86  ...            358500.0         NEAR BAY
2    -122.24     37.85  ...            352100.0         NEAR BAY
3    -122.25     37.85  ...            341300.0         NEAR BAY
4    -122.25     37.85  ...            342200.0         NEAR BAY

[5 rows x 10 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
None
```

- load_housing_data() 함수를 이용해 판다스의 데이터프레임 객체 반환

- head() 메서드와 info() 메서드를 이용해 간단하게 데이터를 확인

Load Data

```python
ocean_proximity = housing['ocean_proximity'].value_counts()
print(ocean_proximity)
```

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

```python
housing.describe()
```

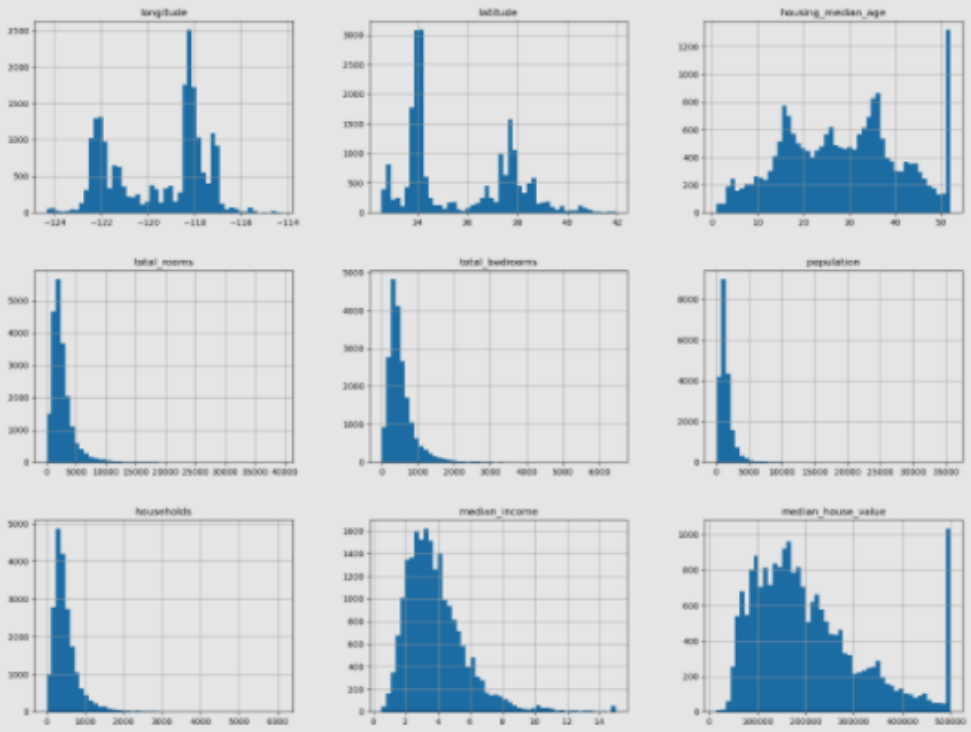
	longitude	latitude	...	median_income	median_house_value
count	20640.000000	20640.000000	...	20640.000000	20640.000000
mean	-119.569704	35.631861	...	3.870671	206855.816909
std	2.003532	2.135952	...	1.899822	115395.615874
min	-124.350000	32.540000	...	0.499900	14999.000000
25%	-121.800000	33.930000	...	2.563400	119600.000000
50%	-118.490000	34.260000	...	3.534800	179700.000000
75%	-118.010000	37.710000	...	4.743250	264725.000000
max	-114.310000	41.950000	...	15.000100	500001.000000

[8 rows x 9 columns]

- info()에서 확인한 ocean_proximity의 데이터 타입은 object(텍스트)
- 텍스트 데이터 타입이므로 특성이 categorical한 것을 예상 가능
- value_counts()를 사용해 각 카테고리가 얼마나 존재하는지 확인
- describe()를 사용해 숫자형 특성을 가진 데이터의 요약 확인

Load Data

```python
housing.hist(bins=50, figsize=(20, 15))
plt.show()
```



- matplotlib를 import한 후 hist()를 이용해 데이터의 히스토그램 확인

Test data set

```python
def train_test():
    '''소득 카테고리를 기반으로 계층 샘플링'''
    housing['income_cat'] = pd.cut(housing['median_income'],
                                   bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
                                   labels=[1, 2, 3, 4, 5])

    strat_test_set, strat_train_set = [], []
    split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
    for train_index, test_index in split.split(housing, housing['income_cat']):
        strat_train_set = housing.loc[train_index]
        strat_test_set = housing.loc[test_index]

    for set_ in (strat_train_set, strat_test_set):
        set_.drop('income_cat', axis=1, inplace=True)

    return strat_train_set, strat_test_set
```

- 계층적 샘플링을 통해 각 그룹을 대표할 수 있는 데이터들을 비율에 맞춰 선정

- 테스트 세트의 패턴을 인지하고 특정 머신러닝 모델을 사용하게되는 데이터 스누핑 편향 방지

Correlation

```python
corr_matrix = housing.corr()
house_price = corr_matrix['median_house_value'].sort_values(ascending=False)
----------------------------
median_house_value    1.000000
median_income         0.688075
total_rooms           0.134153
housing_median_age    0.105623
households            0.065843
total_bedrooms        0.049686
population           -0.024650
longitude            -0.045967
latitude             -0.144160
Name: median_house_value, dtype: float64
```

- corr()을 이용해 중간 주택 가격과 다른 특성 사이의 상관관계 크기 확인 가능

- 상관관계의 범위는 -1부터 1까지이며 1에 가까울수록 양의 상관관계를 가짐

Handling missing value

```python
housing.dropna(subset=['total_bedrooms'])  # 옵션 1
housing.drop('total_bedrooms', axis=1)  # 옵션 2

# 중간값
median = housing['total_bedrooms'].median()  # 옵션 3
housing['total_bedrooms'].fillna(median, inplace=True)
```

```python
# 중간값으로 대체
imputer = SimpleImputer(strategy='median')
# 수치형 특성에만 계산될 수 있으므로 object 특성 제외
housing_num = housing.drop('ocean_proximity', axis=1)

imputer.fit(housing_num)
X = imputer.transform(housing_num)
housing_tr = pd.DataFrame(X, columns=housing_num.columns, index=housing_num.index)
```

- **누락된 값 처리 방법**
  - 해당 구역 제거
  - 전체 특성을 삭제
  - 값으로 채우기(0, 평균, 중간값)
- 중간값을 저장하는것은 이후에 시스템 평가와 실제 운영시 발생하는 누락값을 채우기 위함
- SimpleImputer를 이용해 누락된 값을 처리

Handling categorical data

```python
ordinal_encoder = OrdinalEncoder()
housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat)
------------------------------
[[3.]
 [3.]
 [3.]
 [3.]
 [3.]]

cat_encoder = OneHotEncoder()
housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
--------------------------------
[[0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0.]
 ...
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0.]] # toarray() 사용해서 출력한 것
```

- OrdinalEncoder는 카테고리를 텍스트에서 숫자로 변환해서 표현
- OneHotEncoder는 1이 하나이고 나머지가 0인 배열을 반환
- toarray() 메서드 호출로 넘파이 배열 반환
- categories_ 메서드로 각 인코더에 어떤 카테고리가 있는지 알 수 있음

Converter

```python
rooms_ix, bedrooms_id, population_ix, households_ix = 3, 4, 5, 6

class CombineAttributesAdder(BaseEstimator, TransformerMixin):
    def __init__(self, add_bedrooms_per_room=True):
        self.add_bedrooms_per_room = add_bedrooms_per_room

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        rooms_per_household = X[:, rooms_ix] / X[:, households_ix]
        population_per_household = X[:, population_ix] / X[:, households_ix]
        if self.add_bedrooms_per_room:
            bedrooms_per_room = X[:, bedrooms_id] / X[:, households_ix]
            return np.c_[
                X, rooms_per_household, population_per_household, bedrooms_per_room]
        else:
            return np.c_[X, rooms_per_household, population_per_household]

attr_adder = CombineAttributesAdder(add_bedrooms_per_room=False)
housing_extra_attribs = attr_adder.transform(housing.values)
```

- 사이킷런은 다양한 변환기를 제공하지만 어떤 특성을 조합하는 등의 작업이 필요한 경우 변환기를 직접 생성 가능

- Class 내에 fit과 transform()을 구현

Feature scaling

min-max scaling

- 0과 1 범위에 들도록 값을 이동, 스케일 조정
- 사이킷런의 MinMaxScaler 변환기를 사용

standardization

- 평균을 뺀 후 표준편차로 나눠 계산
- 범위의 상한과 하한이 없음
- 사이킷런의 StandardScaler 변환기 사용

Convert Pipeline

```python
num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('attribs_adder', CombineAttributesAdder()),
    ('std_scaler', StandardScaler())
])
# 파이프라인의 fit 메서드 실행
housing_num_tr = num_pipeline.fit_transform(housing_num)
```

```python
from sklearn.compose import ColumnTransformer

num_attribs = list(housing_num)
cat_attribs = ['ocean_proximity']

full_pipeline = ColumnTransformer([
    ('num', num_pipeline, num_attribs),
    ('cat', OneHotEncoder(), cat_attribs)
])

housing_prepared = full_pipeline.fit_transform(housing)
```

- Pipline은 연속된 단계를 나타내는 이름/추정기 쌍의 목록을 입력으로 받음
- 마지막 단계에는 변환기와 추정기를 모두 사용할 수 있음
- fit() 메서드 호출 시 모든 변환기의 fit_transform() 메서드를 순서대로 호출, 마지막 단계에서는 fit() 메서드만 호출
- ColumnTransformer를 이용해 하나의 변환기로 각 열 마다 적절하게 변환을 적용해 모든 열을 처리

Train model & Evaluate model

```python
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(housing_prepared, housing_labels)
# RMSE: 68628.19819848923
```

```python
from sklearn.model_selection import cross_val_score

scores = cross_val_score(tree_reg, housing_prepared, housing_labels,
                         scoring="neg_mean_squared_error", cv=10)
tree_rmse_scores = np.sqrt(-scores)
# RMSE: 0.0
```

```python
from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor(n_estimators=100, random_state=42)
forest_reg.fit(housing_prepared, housing_labels)
# RMSE: 18603.515021376355
```

- housing_prepared와 housing_labels을 각 모델의 입력으로 사용하여 모델 훈련

- RMSE 지표를 사용해 모델이 예측한 값과 실제 환경에서 관찰되는 값의 차이를 계산

Save model

```python
import joblib

my_model = ''
joblib.dump(my_model, 'my_model.pkl')
my_model_loaded = joblib.load('my_model.pkl')
```

- Joblib을 사용해서 모델을 .pkl 파일로 저장

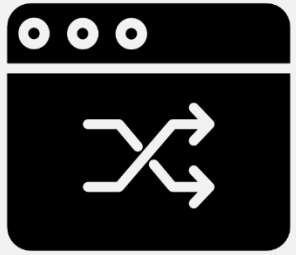
Model tuning

```python
from sklearn.model_selection import GridSearchCV

param_grid = [
    # 12(=3×4)개의 하이퍼파라미터 조합을 시도합니다.
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    # bootstrap은 False로 하고 6(=2×3)개의 조합을 시도합니다.
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
  ]

forest_reg = RandomForestRegressor(random_state=42)
# 다섯 개의 폴드로 훈련하면 총 (12+6)*5=90번의 훈련이 일어납니다.
grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)
grid_search.fit(housing_prepared, housing_labels)
```

```python
grid_search.best_params_
-------------------------
{'max_features': 8, 'n_estimators': 30}

grid_search.best_estimator_
-------------------------
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features=8, max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=30, n_jobs=None, oob_score=False,
                      random_state=42, verbose=0, warm_start=False)
```

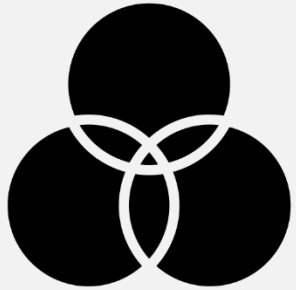
- GridSearchCV을 사용해 랜덤 포레스트 모델의 가능한 모든 하이퍼파라미터 조합에 대해 교차 검증을 사용해 평가
- best_params_ 로 최적의 조합을 확인
- best_estimator_ 로 최적의 추정기를 확인
- GridSearchCV의 refit=True로 초기화되었다면 교차 검증으로 최적의 추정기를 찾고 전체 훈련 세트로 다시 훈련

Model tuning



랜덤 탐색

- 하이퍼파라미터 탐색 공간이 클 경우 RandomizedSearchCV 사용



- 각 반복마다 하이퍼파라미터에 임의의 수를 대입하여 지정 횟수만큼 평가

앙상블 기법

- 최상의 모델을 연결하는 방법

- 결정 트리의 앙상블인 랜덤 포레스트가 결정 트리 하나보다 좋은 성능을 보일 수 있음

Evaluate system by testset

```python
final_model = grid_search.best_estimator_

X_test = strat_test_set.drop("median_house_value", axis=1)
y_test = strat_test_set["median_house_value"].copy()

X_test_prepared = full_pipeline.transform(X_test)
final_predictions = final_model.predict(X_test_prepared)

final_mse = mean_squared_error(y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
```

- 테스트 세트에서 훈련을 하지 않도록 fit_transform()이 아닌 transform()을 호출

```python
from scipy import stats

confidence = 0.95
squared_errors = (final_predictions - y_test) ** 2
np.sqrt(stats.t.interval(confidence, len(squared_errors) - 1,
                         loc=squared_errors.mean(),
                         scale=stats.sem(squared_errors)))
```

- scipy.stats.t.interval()을 사용해 일반화 오차의 95% 신뢰 구간을 계산

- 테스트 세트에서는 성능을 향상시키기위한 하이퍼파라미터 튜닝 불가

Thank you

Site: https://s-wlii.github.io/

Contact: marshmello.sw@gmail.com