그리드 서치 실습

In [145…
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import mean_squared_error
```

In [147…
```
# 데이터 로드 및 전처리
df = pd.read_csv('datasets/Clean_Dataset.csv')
```

✈️ 항공권 가격 데이터셋 설명

이 데이터셋은 **항공권 가격 정보**를 담고 있으며, 항공편의 다양한 특성(출발지, 도착지, 시간대, 남은 일수 등)을 기반으로 **항공권 가격 분석 및 예측**에 활용할 수 있습니다.

🔢 열(Column) 설명

열 이름	설명
`airline`	항공사 이름 (예: SpiceJet, AirAsia, Vistara 등)
`flight`	항공편 번호 (예: SG-8709)
`source_city`	출발 도시 (예: Delhi)
`departure_time`	출발 시간대 (예: Evening, Early_Morning 등)
`stops`	경유 횟수 (`zero`는 직항)
`arrival_time`	도착 시간대 (예: Morning, Night 등)
`destination_city`	도착 도시 (예: Mumbai)
`class`	좌석 등급 (예: Economy)
`duration`	총 비행 시간 (단위: 시간, 예: 2.17 → 약 2시간 10분)
`days_left`	출발일까지 남은 일 수 (예: 1)
`price`	항공권 가격 (예: 5953)

In [150…
```
# 데이터 확인
df.head()
```

Out[150…

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai

In [152…

```
# 학습에 필요 없는 문자열 열 제거
df = df.drop(['flight', 'departure_time', 'stops', 'arrival_time'], axis=1)
```

In [120…

```
# 데이터 전처리
df = pd.get_dummies(df, columns=['airline', 'source_city', 'destination_city', '
X = df.drop('price', axis=1) # 독립 변수
y = df['price'] # 종속 변수 (티켓 가격)
```

In [122…

```
# 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

In [124…

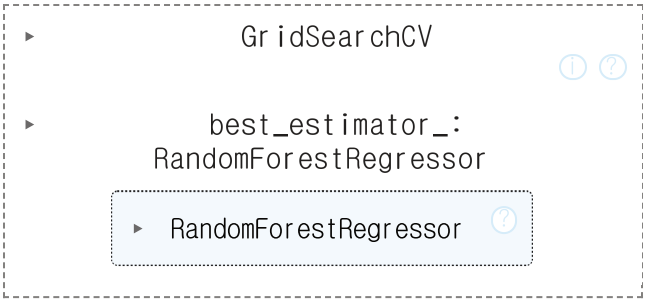
```
# 랜덤 포레스트 모델
rf = RandomForestRegressor(random_state=42)

# 하이퍼파라미터 그리드 설정
param_grid = {
                                'n_estimators': [50, 100, 200], # 트리 개수
                                'max_depth': [5, 10, None], # 트리의 최대 깊이
                                'min_samples_split': [2, 5, 10], # 최소 샘플 분할
                                'min_samples_leaf': [1, 2, 4] # 최소 리프 샘플 수
}
```

In [126…

```
# 그리드 서치 실행
grid_search = GridSearchCV(
                            estimator=rf,
                            param_grid=param_grid,
                            cv=3, # 교차 검증 폴드 수
                            scoring='neg_mean_squared_error', # MSE를 음수 값으로
                            n_jobs=-1 # 병렬 처리
)
grid_search.fit(X_train, y_train)
```

Out[126…



In [127…

```
# 최적 하이퍼파라미터 확인
best_params = grid_search.best_params_
print("Best Parameters:", best_params)
```

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split':
10, 'n_estimators': 200}
```

In [128…

```
# 테스트 데이터 성능 평가
best_model = grid_search.best_estimator_
predictions = best_model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("Test MSE:", mse)
```

```
Test MSE: 9456645.15948964
```

랜덤 서치

In [130…

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint
```

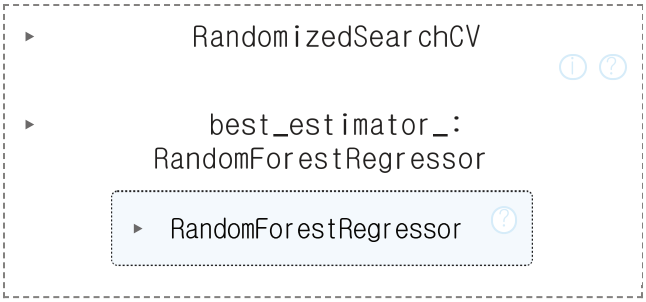
In [131…

```
# 하이퍼파라미터 분포 설정
param_dist = {
                                'n_estimators': randint(50, 200),
                                'max_depth': randint(3, 15),
                                'min_samples_split': randint(2, 10),
                                'min_samples_leaf': randint(1, 5)
}
```

In [136…

```
# 랜덤 서치 실행
random_search = RandomizedSearchCV(
                                        estimator=rf,
                                        param_distributions=param_dist,
                                        n_iter=20, # 시도할 조합 수
                                        cv=3,
                                        scoring='neg_mean_squared_error',
                                        n_jobs=-1, random_state=42
)
random_search.fit(X_train, y_train)
```

Out[136…



In [137…

```
# 최적 하이퍼파라미터 확인
best_params = random_search.best_params_
print("Best Parameters:", best_params)
```

```
Best Parameters: {'max_depth': 14, 'min_samples_leaf': 3, 'min_samples_split': 5,
'n_estimators': 57}
```

In [138…

```
# 테스트 데이터 성능 평가
best_model = random_search.best_estimator_
predictions = best_model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print("Test MSE:", mse)
```

```
Test MSE: 14375270.783725154
```