위스콘신 유방암 데이터의 하이퍼 파라미터 튜닝

학습 목표

- 배깅 방식의 알고리즘인 랜덤 포레스트를 이용하여 모델을 만들어봅니다.
- GridSearchCV를 이용하여 하이퍼 파리미터 튜닝을 수행해 봅니다.

학습 내용

- 위스콘신 유방암 데이터 세트에 대한 랜덤포레스트 모델 생성 및 학습
- 하이퍼 파라미터 튜닝해 보기

데이터 로드 및 전처리

In [25]:

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib

from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_breast_cancer
```

In [26]:

```
cancer = load_breast_cancer()

cancer_df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
cancer_df.head()
```

Out[26]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mear symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.241
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.181
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.206
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.259
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.180

5 rows × 30 columns

In [27]:

```
cancer_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

In [28]:

```
print( cancer_df.shape)
```

```
(569, 30)
```

데이터 설명

- 위스콘신 유방암 데이터 세트는 유방암의 악성 종양, 양성 종양 여부를 결정하는 이진 분류
- 종양의 크기, 모양 등의 형태와 관련한 많은 피처를 가지고 있음.
- 569개의 행과, 30개의 피처로 이루어진 데이터
- null 값이 없음. 값들은 실수로 되어 있음.

데이터 나누기

In [41]:

```
# 피처와 레이블를 지정.
X = cancer_df[:]
y = cancer.target

X.shape, y.shape
```

Out[41]:

```
((569, 30), (569,))
```

In [45]:

```
from sklearn.model_selection import train_test_split

X_train , X_test, y_train, y_test = train_test_split(X, y,
                                                     test_size=0.2, random_state=0)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[45]:

```
((455, 30), (114, 30), (455,), (114,))
```

GridSearchCV를 이용한 랜덤 포레스트의 하이퍼 파라미터 튜닝

In [46]:

```
### 기본 RandomForest 모델
```

In [47]:

```
# 모델 선택
model = RandomForestClassifier()
# 학습
model.fit(X_train, y_train)
# 예측
pred = model.predict(X_test)

print("예측 정확도 : {0:.4f}".format(accuracy_score(y_test, pred) ))
```

```
예측 정확도 : 0.9561
```

In [48]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
```

In [49]:

```
params = {
    "n_estimators" : [100],
    "max_depth":[6,8,10,12],
    "min_samples_leaf":[8,12,18],
    "min_samples_split":[8,16,20]
}
```

In [50]:

```
# RandomForestClassifier 모델 객체 생성 후, GridSearch 수행
model_rf = RandomForestClassifier(random_state=0, n_jobs=-1)
grid_cv = GridSearchCV(model_rf, param_grid=params, cv=2, n_jobs=-1)
grid_cv.fit(X_train, y_train)

print("최적의 하이퍼 파리미터 : ₩n", grid_cv.best_params_)
print("최고의 정확도 : {0:.4f}".format(grid_cv.best_score_ ))
```

```
최적의 하이퍼 파리미터 :
 {'max_depth': 6, 'min_samples_leaf': 8, 'min_samples_split': 8, 'n_estimators': 10
0}
최고의 정확도 : 0.9451
```

- 위의 내용을 이용하여 최종 모델 만들기

In [51]:

```
# 모델 선택
model_lrf = RandomForestClassifier(n_estimators= 100, max_depth= 6,
                                   min_samples_leaf= 8, min_samples_split= 8, random_state=0)

model_lrf.fit(X_train, y_train)
pred = model_lrf.predict(X_test)

print("예측 정확도 : {0:.4f}".format(accuracy_score(y_test, pred) ))
```

```
예측 정확도 : 0.9649
```

02. 모델 수행 후, feature(피처)의 중요도 확인해보기

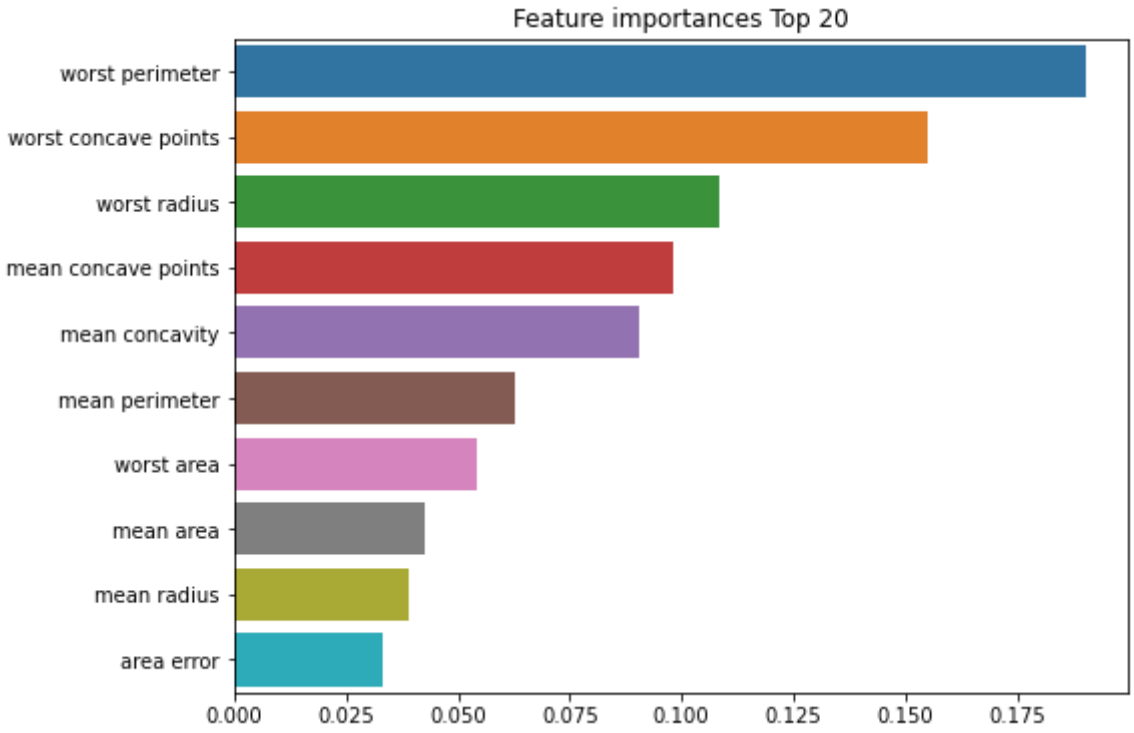
In [55]:

```
import seaborn as sns
import matplotlib.pyplot as plt

f_imp_values = model_lrf.feature_importances_
f_importances = pd.Series(f_imp_values, index=X_train.columns)
f_top10 = f_importances.sort_values(ascending=False)[:10]       # 10위까지의 중요도 확인
```

In [57]:

```
plt.figure(figsize=(8, 6))
plt.title("Feature importances Top 20")
sns.barplot(x=f_top10, y=f_top10.index)
plt.show()
```



정리

- GridSearchCV를 활용하여 하이퍼 파리미터 튜닝이 가능하다.