MM20B007 Tutorial 10

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.ensemble import BaggingClassifier
```

```
data = datasets.load_wine(as_frame= True)
X = data.data
Y = data.target
```

Dividing the Dataset into 3 parts : Train, Validation and Test

```
x, X_test, y, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 22)
X_train, X_val, y_train, y_val = train_test_split(x, y, test_size = 0.25, random_state = 22)
```

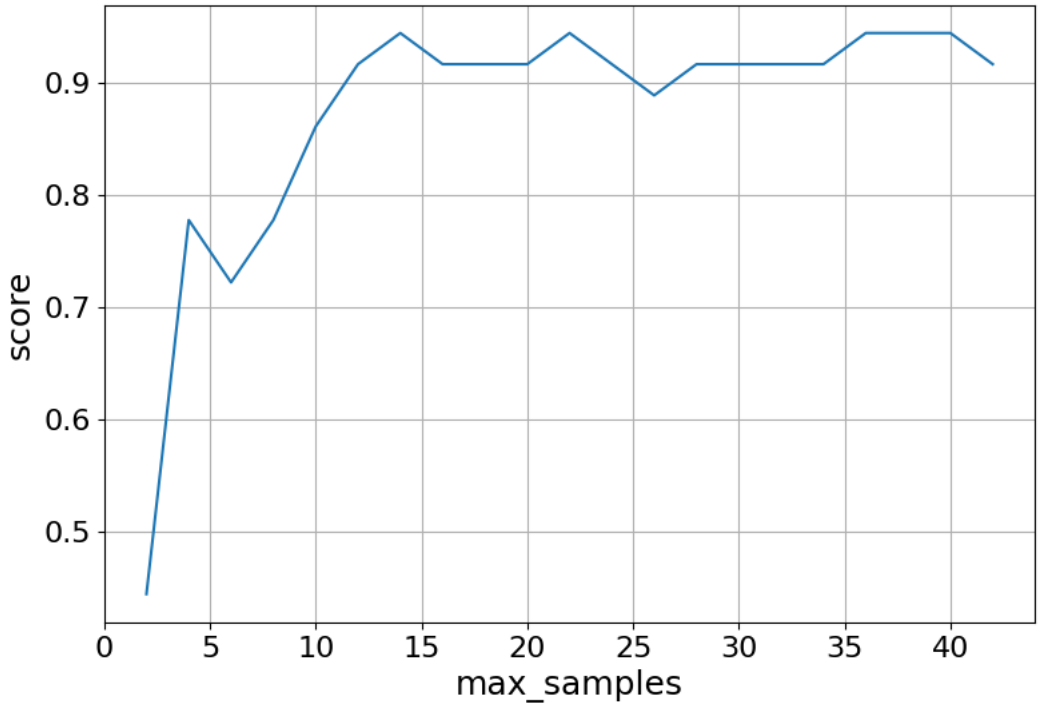
```
print(f'The size of train set is: {X_train.shape}')
print(f'The size of validation set is: {X_val.shape}')
print(f'The size of test set is: {X_test.shape}')
```

```
The size of train set is: (106, 13)
The size of validation set is: (36, 13)
The size of test set is: (36, 13)
```

Optimizing the max_sample parameter

```
models_max_samples = []
scores_max_samples = []
# this is for validation of appropriate no. of max_samples to be used
max_samples_range = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42]
for max_samples in max_samples_range:
  clf1 = BaggingClassifier(max_samples = max_samples, random_state = 22)
  clf1.fit(X_train, y_train)
  models_max_samples.append(clf1)
  scores_max_samples.append(accuracy_score(y_true = y_val, y_pred = clf1.predict(X_val)))
```

```
# Generate the plot of scores against number of estimators
plt.figure(figsize=(9,6))
plt.plot(max_samples_range, scores_max_samples)
# Adjust labels and font (to make visable)
plt.xlabel("max_samples", fontsize = 18)
plt.ylabel("score", fontsize = 18)
plt.tick_params(labelsize = 16)
plt.grid()
# Visualize plot
plt.show()
```



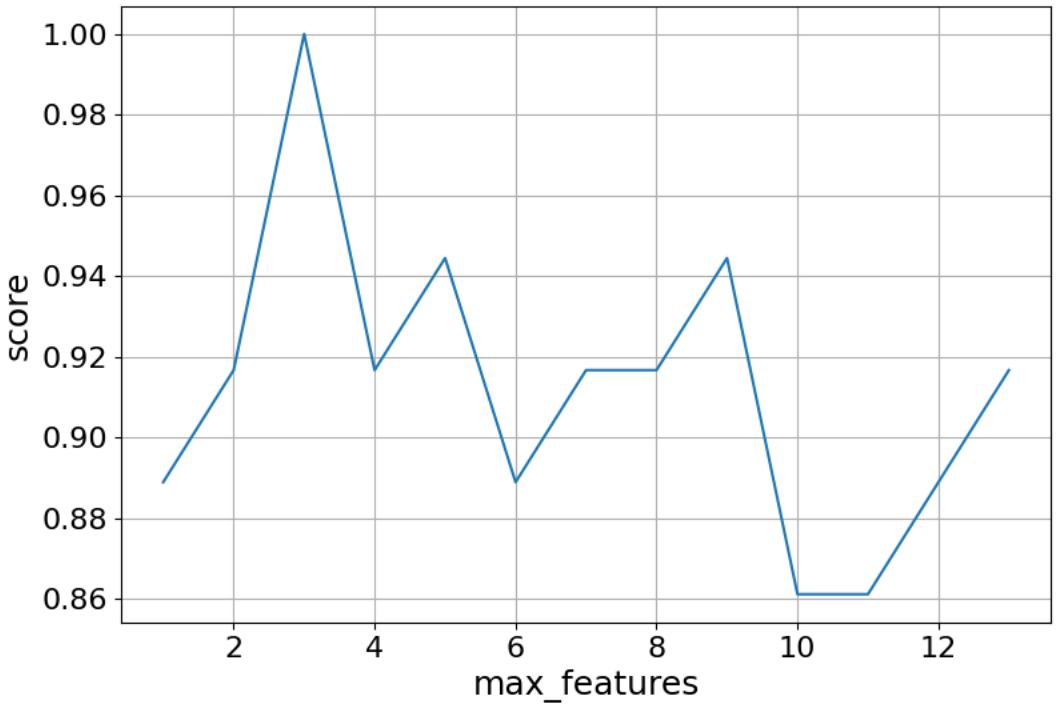
```
# Taking the value with maximum score
clf_max_samples = BaggingClassifier(max_samples = 42, random_state = 22)
clf_max_samples.fit(X_train, y_train)
print(f'The accuracy of the model on test data when max_features = 6 is: {accuracy_score(y_true = y_test, y_pred = clf_max_samples.predict(X_test))}')
```

```
The accuracy of the model on test data when max_features = 6 is: 0.9722222222222222
```

Optimizing the max_features parameter

```
models_max_features = []
scores_max_features = []
# this is for validation of appropriate no. of max_samples to be used
max_features_range = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
for max_features in max_features_range:
  clf2 = BaggingClassifier(max_features = max_features, random_state = 22)
  clf2.fit(X_train, y_train)
  models_max_features.append(clf2)
  scores_max_features.append(accuracy_score(y_true = y_val, y_pred = clf2.predict(X_val)))
```

```
# Generate the plot of scores against number of estimators
plt.figure(figsize=(9,6))
plt.plot(max_features_range, scores_max_features)
# Adjust labels and font (to make visable)
plt.xlabel("max_features", fontsize = 18)
plt.ylabel("score", fontsize = 18)
plt.tick_params(labelsize = 16)
plt.grid()
# Visualize plot
plt.show()
```




```
# Taking the value with maximum score
clf_max_features = BaggingClassifier(max_features = 6, random_state = 22)
clf_max_features.fit(X_train, y_train)
print(f'The accuracy of the model on test data when max_features = 6 is: {accuracy_score(y_true = y_test, y_pred = clf_max_features.predict(X_test))}')
```

```
The accuracy of the model on test data when max_features = 6 is: 0.9722222222222222
```

Optimizing by Grid Search

```
parameters = {
    'max_samples': [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42],
    'max_features': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
}

clf_new = BaggingClassifier()
gridsearch = GridSearchCV(estimator=clf_new, param_grid = parameters, scoring = 'accuracy')
gridsearch.fit(X_train, y_train)
```