we observe though the model with both 150 and 200 trees produce similar accuracy score, the complexity and computation time for 150 trees will be less than 200. Hence, Gradient Bossting with 150 trees works best.

Question 2 - Naive Bayes (Unprocessed)

[30]:
```python
from sklearn.naive_bayes import GaussianNB

# find best value for var_smoothing parameter using gridsearchcv
param_grid = {
    'var_smoothing'  :[1e-10, 1e-9, 1e-5, 1e-3, 1e-1]
      }

tree = GaussianNB()
grid_search = GridSearchCV(estimator=tree, param_grid=param_grid, cv=10)
grid_search.fit(X_train_val, y_train_val)
grid_search.best_params_
```

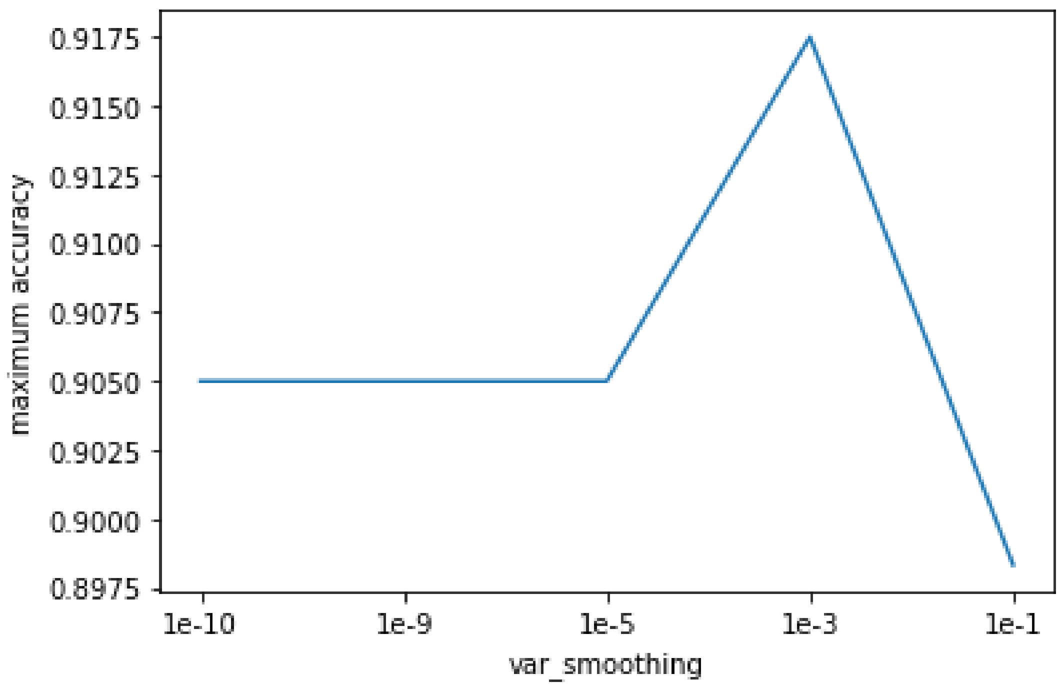
[30]:
```
{'var_smoothing': 0.001}
```

[31]:
```python
# using cross validation on train set to fine tune the var_smoothing parameter
var_smoothing = [1e-10, 1e-9, 1e-5, 1e-3, 1e-1]
Scores = []
max_acc=0
best_var=0
for var in var_smoothing:
    print(var)
    gnb = GaussianNB(var_smoothing=var)
    gnb.fit(X_train_val, y_train_val)
    accuracy = cross_val_score(gnb, X_train_val, y_train_val, cv=10,␣
 ↪scoring='accuracy')
    print(accuracy.mean())
    Scores.append(accuracy.mean())
    if(accuracy.mean() > max_acc):
            max_acc=accuracy.mean()
            best_var=var
print('The maximum accuracy value is ', max_acc)
print('The best value of var_smoothing is ', best_var)
```

```
1e-10
0.905
1e-09
0.905
1e-05
0.905
0.001
0.9175000000000001
0.1
0.8983333333333334
The maximum accuracy value is  0.9175000000000001
The best value of var_smoothing is  0.001
```

[32]:
```
# plotting the mean accuracy versus the number of estimators
plt.xlabel("var_smoothing")
plt.ylabel("maximum accuracy")
xticks = ['1e-10', '1e-9', '1e-5', '1e-3', '1e-1']
plt.plot(xticks, Scores)
```

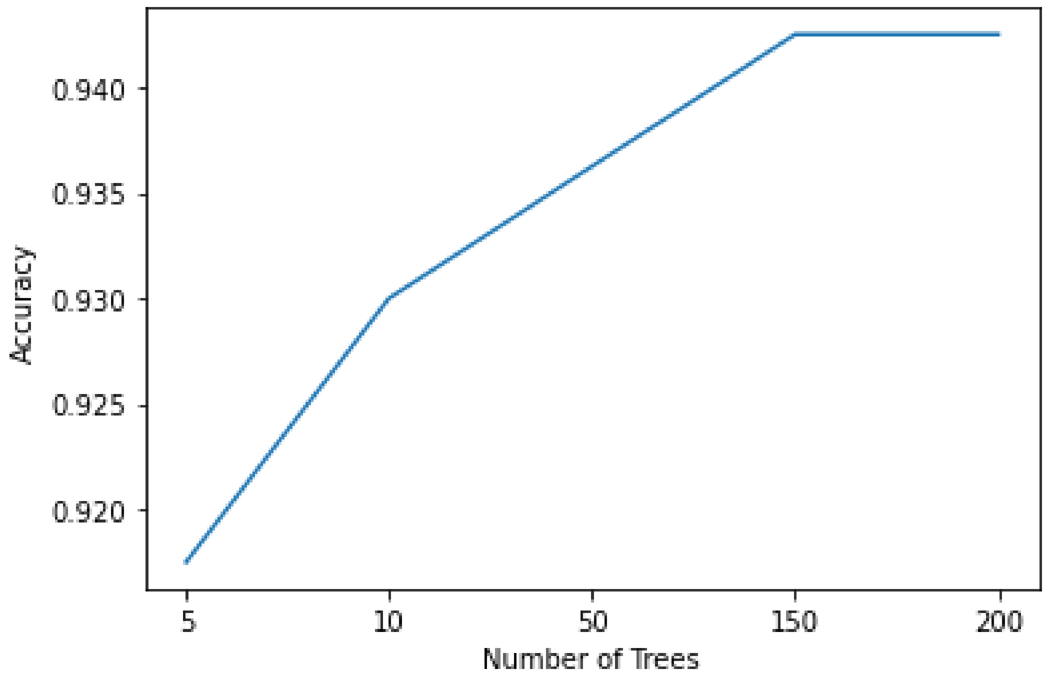
[32]:
```
[<matplotlib.lines.Line2D at 0x20c72c6b9a0>]
```



We observe maximum accuracy for variance smoothing parameter 1e-3 (0.001). Smoothing allows Naive Bayes to better handle cases where evidence has never appeared for a particular category i.e. the problem of zero probability. Var_smoothing is the portion of the largest variance of all features that is added to variances for calculation stability i.e if the predicted value is too small. We observe the accuracy is same as the var_smoothing parameter increases from 1e-10, 1e-9, 1e-5, with peak accuracy at 1e-3, and further decreasing at 1e-1

[33]:
```
%%time
# applying the best value of var_smoothing on test set
gnb = GaussianNB(var_smoothing=best_var)
gnb.fit(X_train_val, y_train_val)
gnb.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: ', accuracy)
f_score = f1_score(y_test, y_pred, average = 'macro')
print('f-score:', f_score)
```

```
Accuracy:  0.95
f-score: 0.9484126984126985
Wall time: 12 ms
```



we observe though the model with both 150 and 200 trees produce similar accuracy score, the complexity and computation time for 150 trees will be less than 200. Hence, Gradient Bossting with 150 trees works best.

[62]:
```
%%time
# applying the best value of number of trees on test set
gt_clf = GradientBoostingClassifier(n_estimators=max_trees, random_state=0)
gt_clf.fit(X_train_val, y_train_val)
y_pred = gt_clf.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: ', accuracy)
f_score = f1_score(y_test, y_pred, average = 'macro')
print('f-score:', f_score)
```

```
Accuracy:  0.925
f-score: 0.9265428824049513
Wall time: 387 ms
```

Question 2 - Naive Bayes (Processed Dataset)

[63]:
```
from sklearn.naive_bayes import GaussianNB

# find best value for var_smoothing parameter using gridsearchcv
param_grid = {
    'var_smoothing'  :[1e-10, 1e-9, 1e-5, 1e-3, 1e-1]
      }

tree = GaussianNB()
grid_search = GridSearchCV(estimator=tree, param_grid=param_grid, cv=10)
```

```
grid_search.fit(X_train_val, y_train_val)
grid_search.best_params_
```

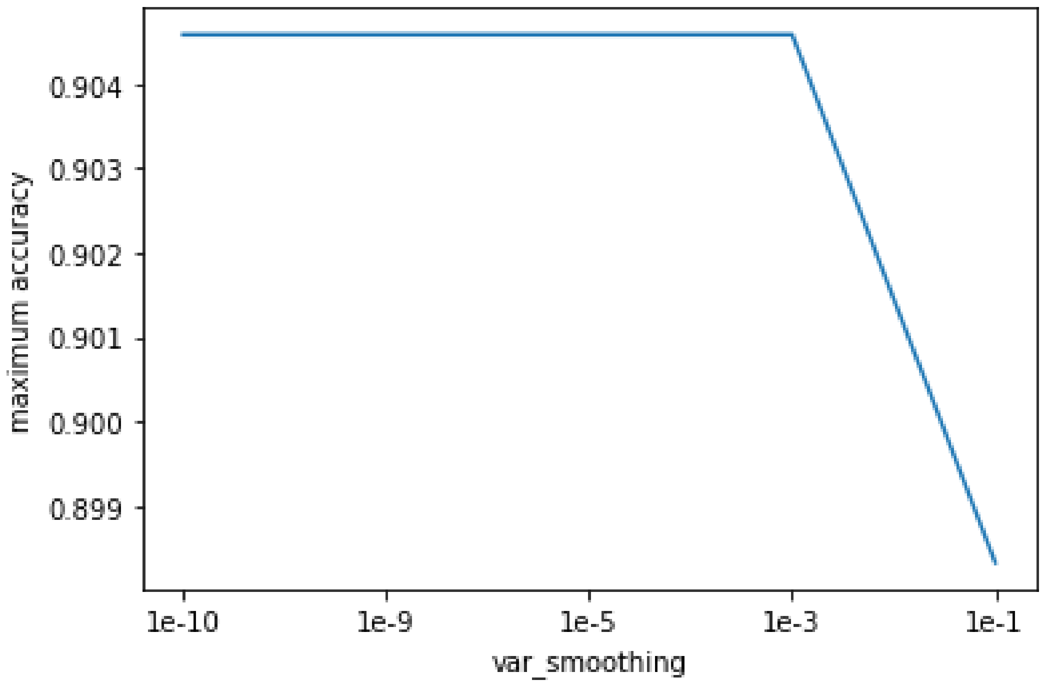
[63]:
```
{'var_smoothing': 1e-10}
```

[64]:
```
# using cross validation on train set to fine tune the var_smoothing parameter
var_smoothing = [1e-10, 1e-9, 1e-5, 1e-3, 1e-1]
Scores = []
max_acc=0
best_var=0
for var in var_smoothing:
    print(var)
    gnb = GaussianNB(var_smoothing=var)
    gnb.fit(X_train_val, y_train_val)
    accuracy = cross_val_score(gnb, X_train_val, y_train_val, cv=10,␣
 ↪scoring='accuracy')
    print(accuracy.mean())
    Scores.append(accuracy.mean())
    if(accuracy.mean() > max_acc):
            max_acc=accuracy.mean()
            best_var=var
print('The maximum accuracy value is ', max_acc)
print('The best value of var_smoothing is ', best_var)
```

```
1e-10
0.9045833333333334
1e-09
0.9045833333333334
1e-05
0.9045833333333334
0.001
0.9045833333333334
0.1
0.8983333333333334
The maximum accuracy value is  0.9045833333333334
The best value of var_smoothing is  1e-10
```

[65]:
```
# plotting the mean accuracy versus the number of estimators
plt.xlabel("var_smoothing")
plt.ylabel("maximum accuracy")
xticks = ['1e-10', '1e-9', '1e-5', '1e-3', '1e-1']
plt.plot(xticks, Scores)
```

[65]:
```
[<matplotlib.lines.Line2D at 0x20c751d4910>]
```



We observe same maximum accuracy for variance smoothing parameters 1e-10, 1e-9, 1e-5, 1e-3. Smoothing allows Naive Bayes to better handle cases where evidence has never appeared for a particular category i.e. the problem of zero probability. Var_smoothing is the portion of the largest variance of all features that is added to variances for calculation stability i.e if the predicted value is too small. The smoothing parameter doesn't have a major impact here.This can be attributed to the type of the features , as all the features are numeric in the dataset.

[66]:
```
%%time
# applying the best value of var_smoothing on test set
gnb = GaussianNB(var_smoothing=best_var)
gnb.fit(X_train_val, y_train_val)
gnb.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: ', accuracy)
f_score = f1_score(y_test, y_pred, average = 'macro')
print('f-score:', f_score)
```

```
Accuracy:  0.925
f-score: 0.9265428824049513
Wall time: 7.98 ms
```

Impact of preprocessing:

- we observed pre processing of the data (normalization , feature selection etc.) didn't improve the performance of the Tree based classifiers significantly. In fact to the contrary, the accuracy of the ensembe methods were observed to have gone down by a small amount.