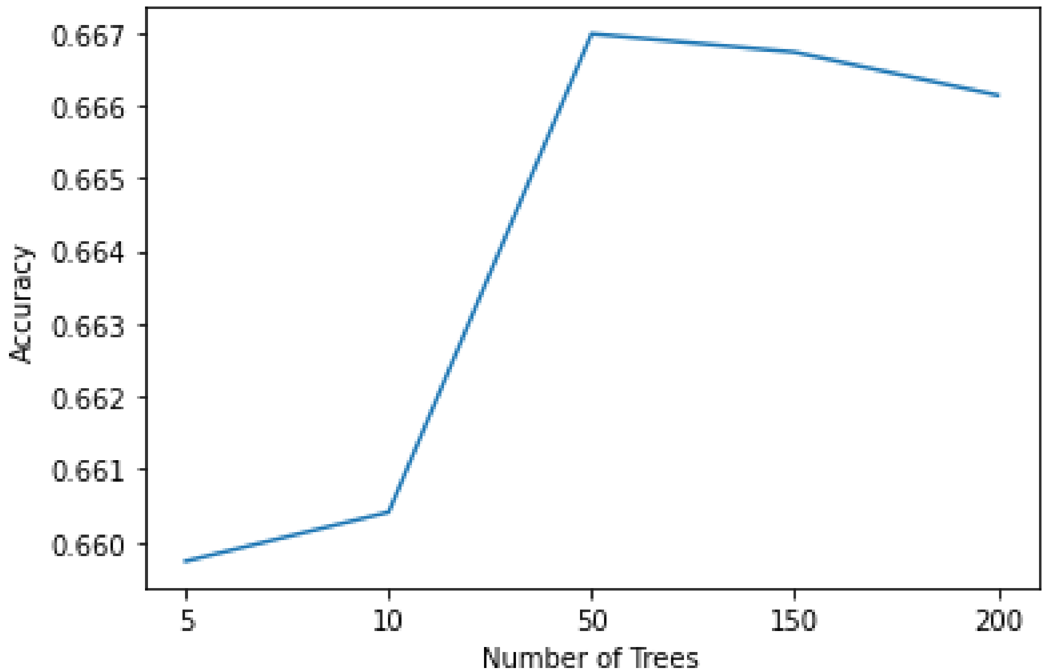


We observe maximum accuracy for number of trees 50.

[27]:
```
%%time
# applying the best value of number of trees on test set
clf = GradientBoostingClassifier(n_estimators=max_trees, random_state=0)
clf.fit(X_train_val, y_train_val)
y_pred = 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:  0.6675648366453352
f-score: 0.6552896231570426
Wall time: 2.44 s
```

Question 2 - Naive Bayes (Unprocessed)

[28]:
```
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_
```

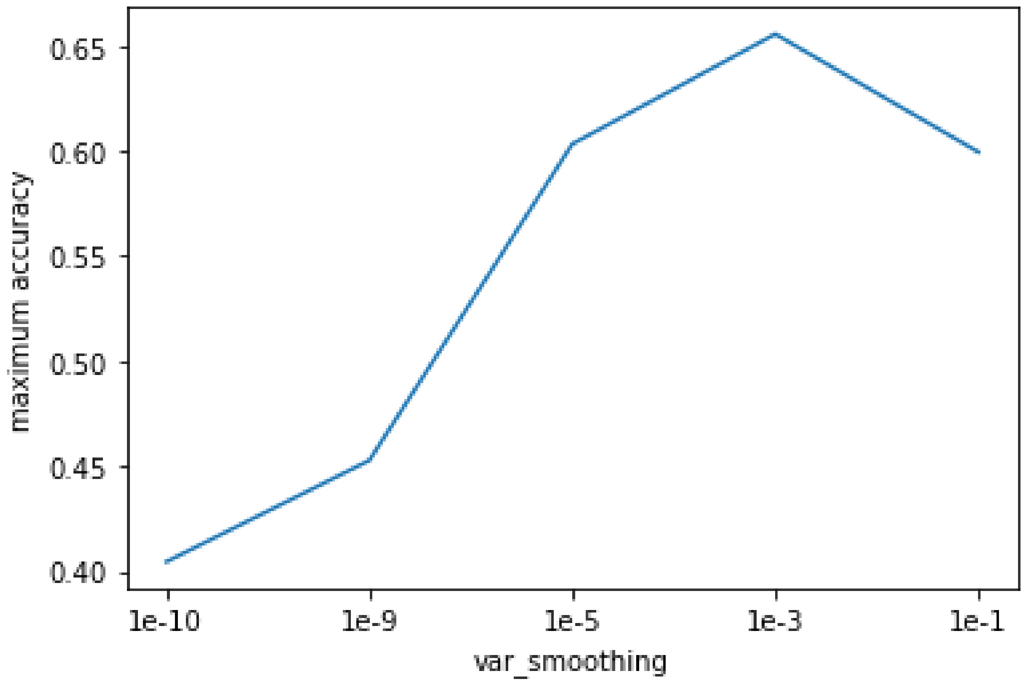
[28]: {'var_smoothing': 0.001}

[29]:
```
# 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
0.404520563682316
1e-09
0.45276870076785836
1e-05
0.6033197036356261
0.001
0.6556105140140523
0.1
0.5995321085043074
The maximum accuracy value is  0.6556105140140523
The best value of var_smoothing is  0.001
```

[30]:
```
# 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)
```

[30]: [<matplotlib.lines.Line2D at 0x27e04c7e2e0>]



We observe maximum accuracy for variance smoothing parameters 0.001 (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. We observe with increasing smoothing parameter, the accuracy of the model increases to a maximum at 1e-3 , after which it decreases.

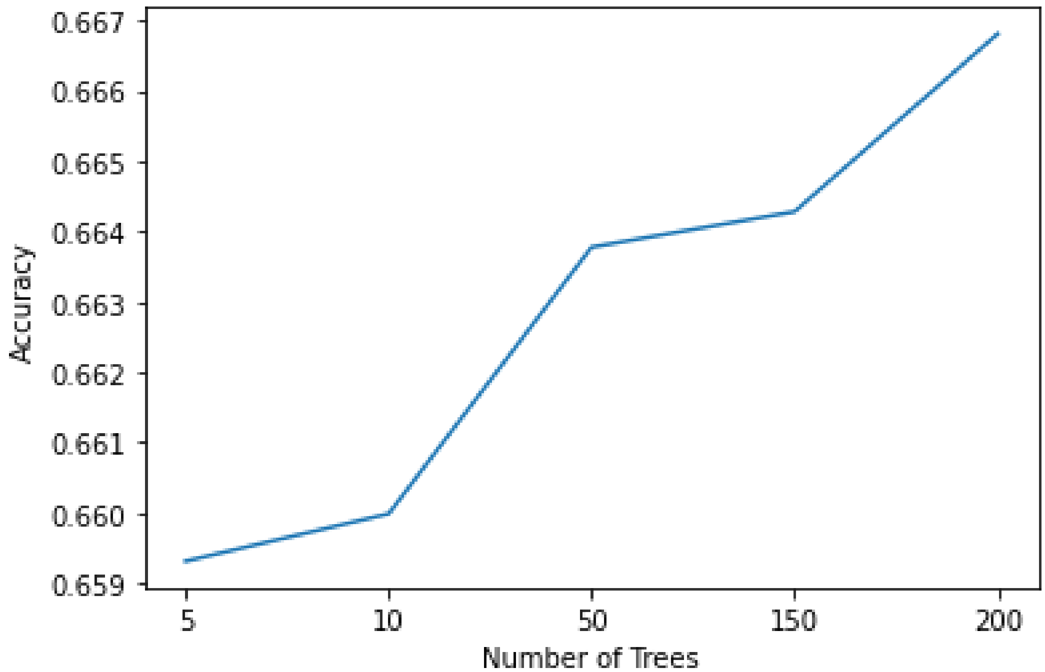
[31]:
```
%%time
# applying the best value of var_smoothing on test set
gnb = GaussianNB(var_smoothing=best_var)
gnb.fit(X_train_val, y_train_val)
gnb.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: ', accuracy)
f_score = f1_score(y_test, y_pred, average = 'macro')
print('f-score:', f_score)
```

```
Accuracy:  0.6675648366453352
f-score: 0.6552896231570426
Wall time: 46.9 ms
```

Preprocessing

[32]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

[33]:
```
df = pd.read_csv('covid_train.csv')
```



We observe maximum accuracy at number of trees 200

[60]:
```
%%time
# applying the best value of number of trees on test set
clf = GradientBoostingClassifier(n_estimators=max_trees, random_state=0)
clf.fit(X_train_val, y_train_val)
y_pred = 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:  0.6722802290333446
f-score: 0.6624100366524396
Wall time: 6.88 s
```

Question 2 - Naive Bayes (Processed Dataset)

[61]:
```
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_
```

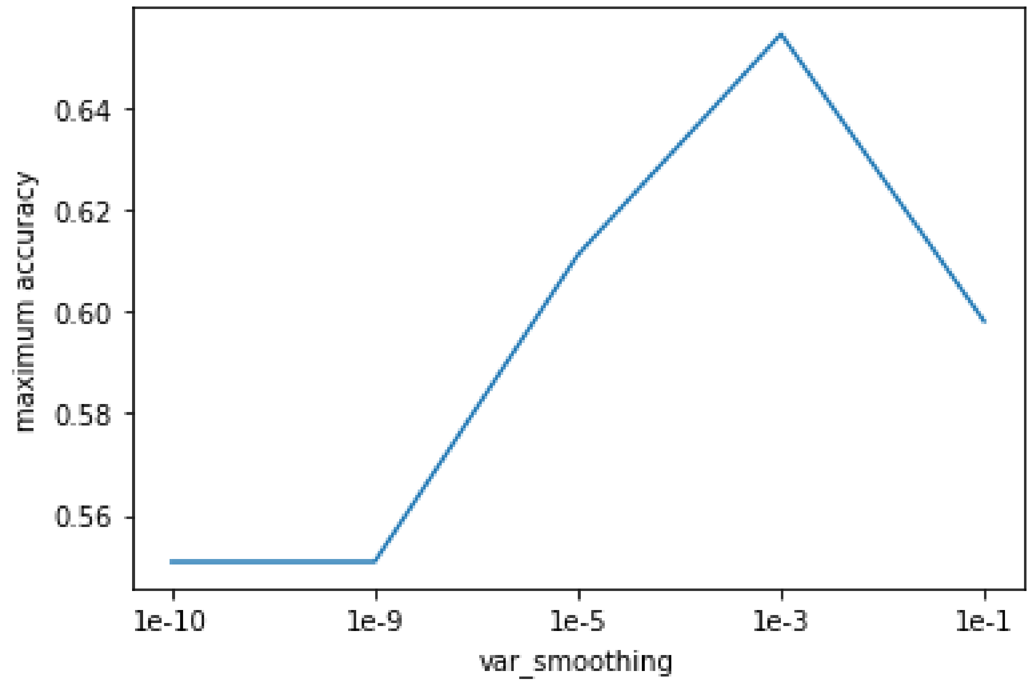
[61]: {'var_smoothing': 0.001}

[62]:
```
# 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
0.5508603303464297
1e-09
0.5508603303464297
1e-05
0.6111514612567688
0.001
0.6545165215763362
0.1
0.5981007775026309
The maximum accuracy value is  0.6545165215763362
The best value of var_smoothing is  0.001
```

[63]:
```
# 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)
```

[63]: [<matplotlib.lines.Line2D at 0x27e04dcd0a0>]



We observe same maximum accuracy for variance smoothing parameters 0.001 (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. We observe with increasing smoothing parameter, the accuracy of the model remains constant, peaks at 0.001 and again decreases.

[64]:
```
%%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:  0.6722802290333446
f-score: 0.6624100366524396
Wall time: 22.9 ms
```

NB learned Parameters theta_ (mean) and sigma_ (variance)

[65]:
```
# mean
gnb.theta_
```

[65]:
```
array([[ 3.84338492e+01,  4.37274733e+01, -7.95909554e+01,
          4.97254119e-01,  6.24063904e-03,  1.59760359e-01,
          2.05941088e-01,  1.75736395e-01,  1.72241638e-02,
          9.98502247e-04,  2.09685472e-01],
        [ 4.02877131e+01,  4.37725211e+01, -7.96265344e+01,
          4.90714831e-01,  9.41236327e-03,  3.56652251e-01,
```