

# optimization

September 29, 2021

## 1 Example of optimizing lowess fit and max and redundant set selection

```
[1]: import os,errno
import dRFEtools
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.datasets import make_regression
from sklearn.datasets import make_classification
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
```

```
[2]: def mkdir_p(directory):
    try:
        os.makedirs(directory)
    except OSError as e:
        if e.errno != errno.EEXIST:
            raise
```

### 1.1 Optimize classification

```
[3]: outdir = 'classification/'
mkdir_p(outdir)
```

```
[4]: # Create a dataset with only 10 informative features
X, y = make_classification(
    n_samples=500, n_features=20000, n_informative=100, n_redundant=300,
    n_repeated=0, n_classes=2, n_clusters_per_class=1, random_state=13
)

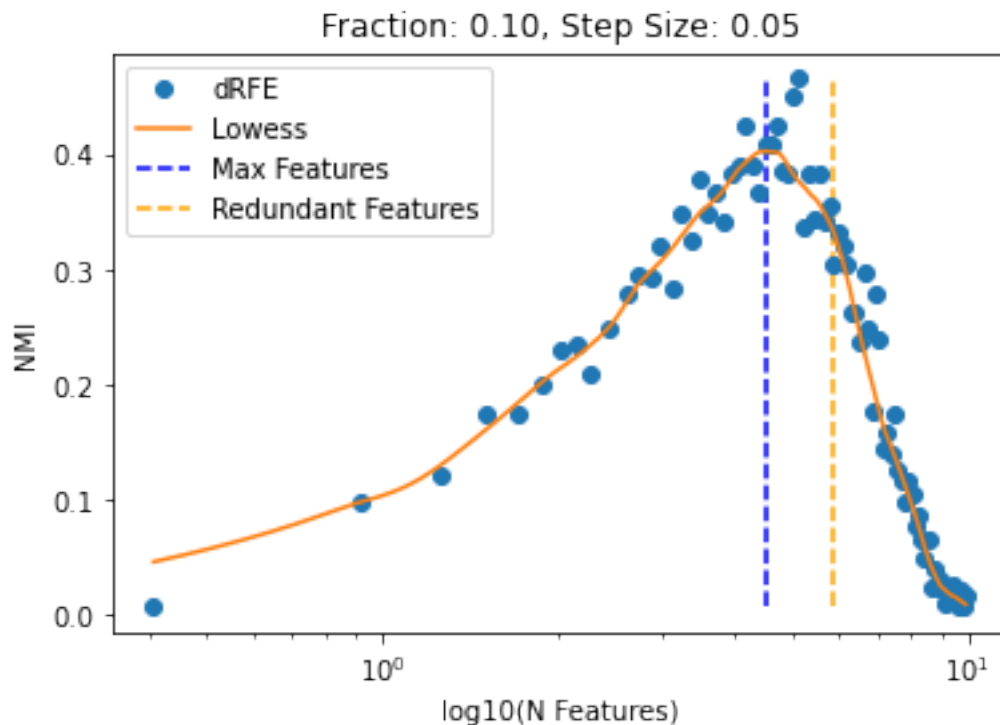
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=13)
#cla = dRFEtools.LogisticRegression(max_iter=1000, n_jobs=-1)
cla = dRFEtools.RandomForestClassifier(n_estimators=100, n_jobs=-1,
                                       oob_score=True, random_state=13)
```

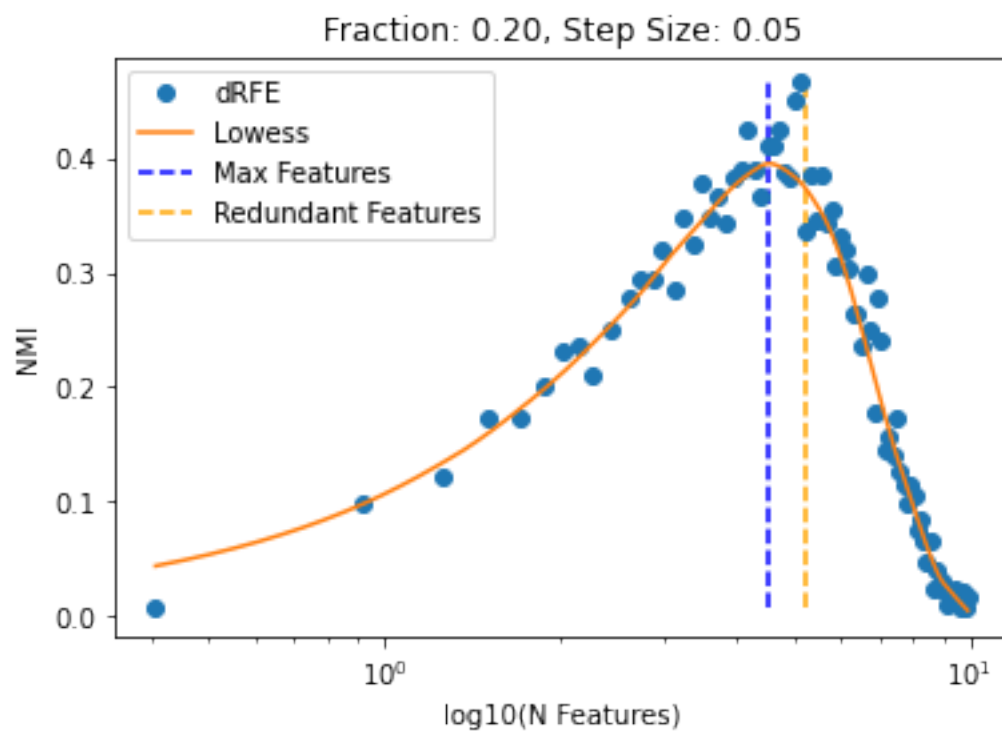
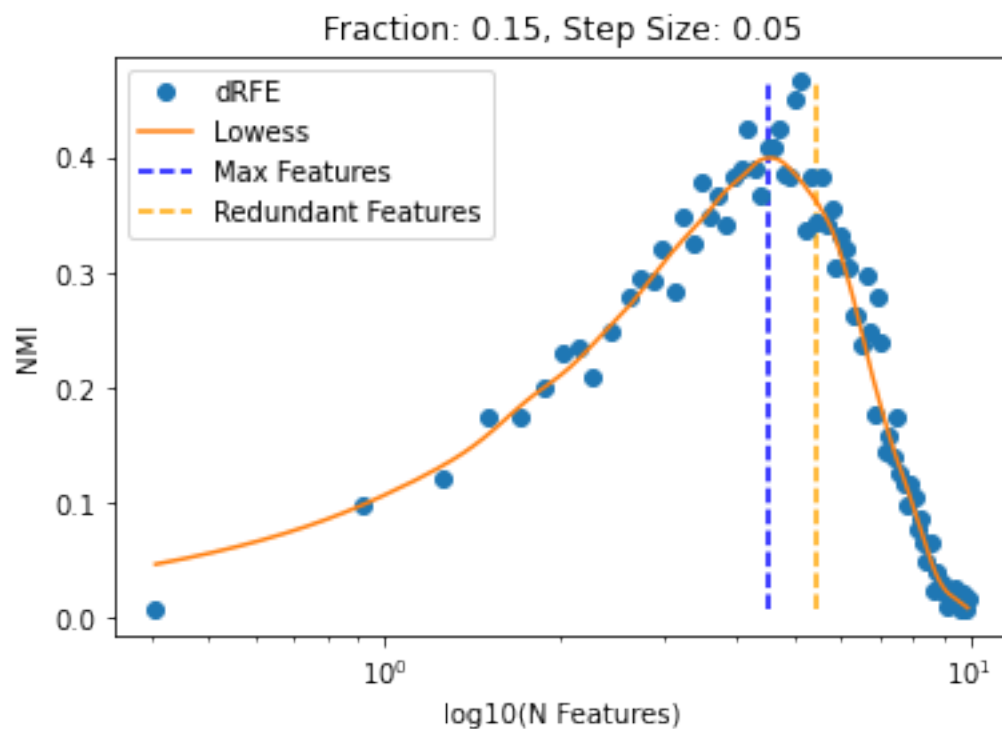
```
[5]: fold = 1
for train_index, test_index in cv.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    fold += 1
fold -= 1

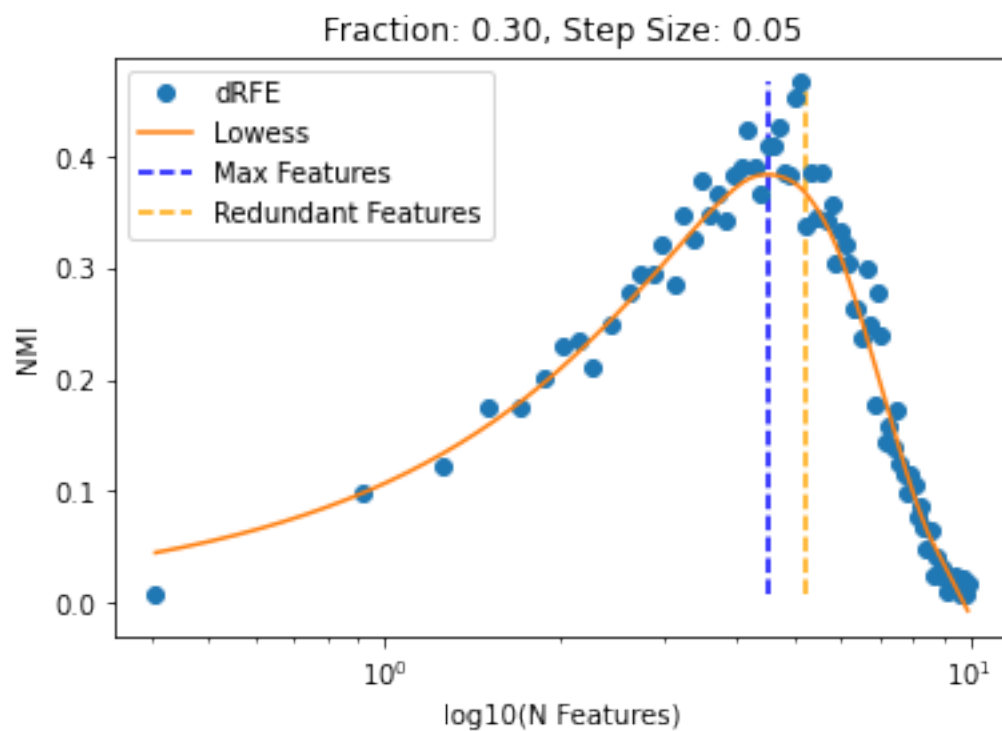
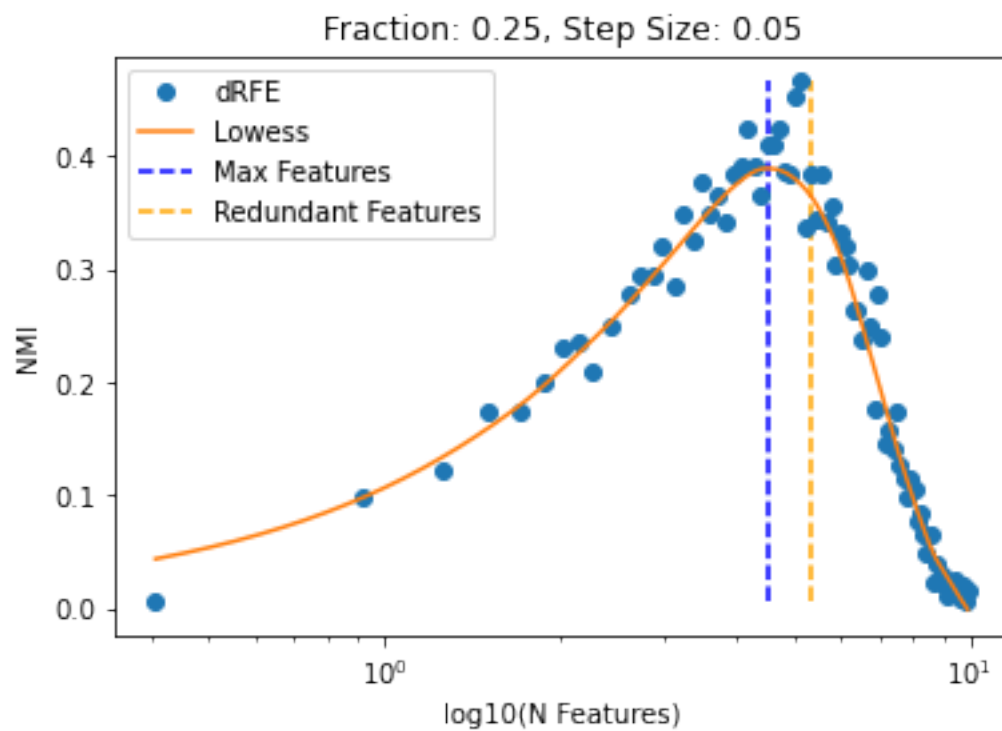
features = ["feature_%d" % x for x in range(X_train.shape[1])]
d, pfirst = dRFEtools.rf_rfe(cla, X_train, y_train, np.array(features),
                             fold, outdir, elimination_rate=0.1, RANK=False)
```

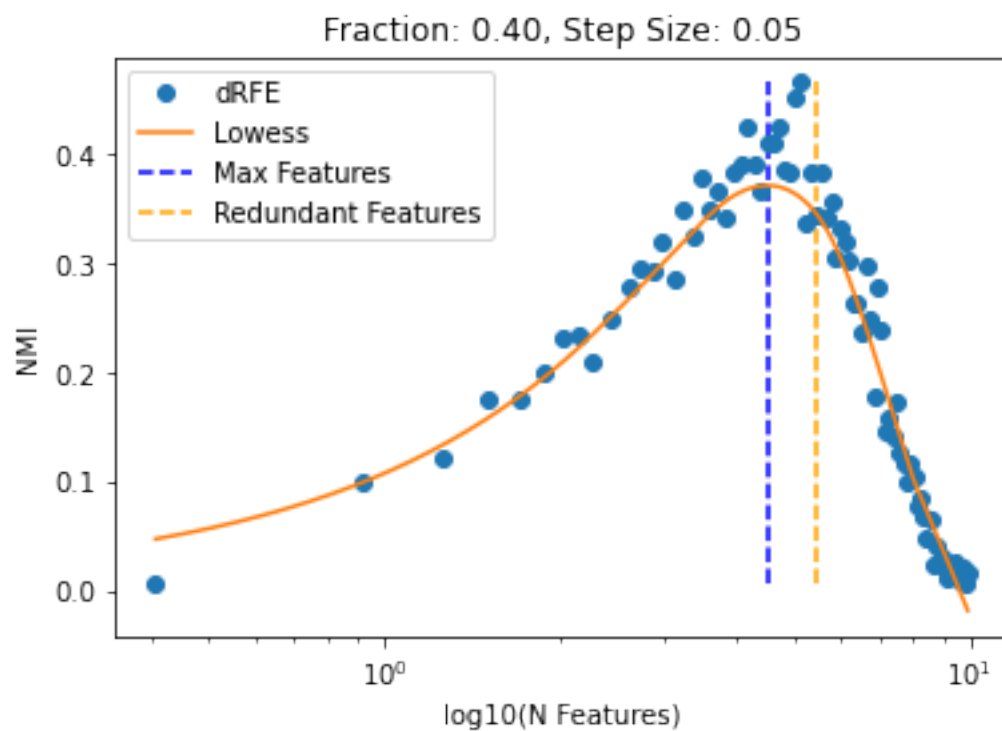
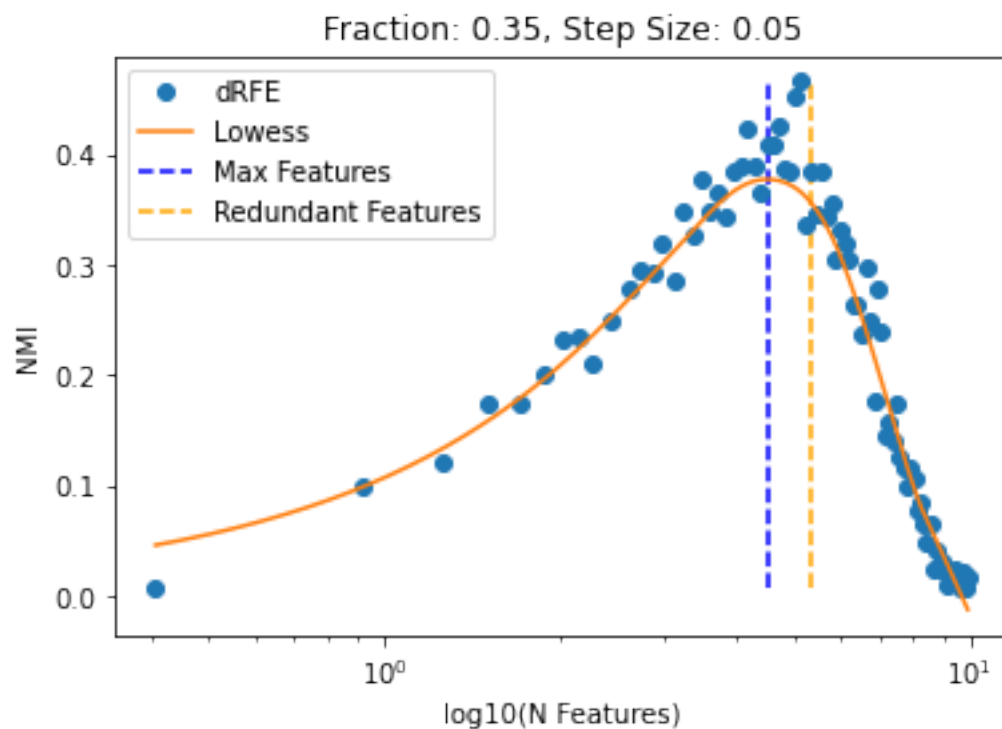
### 1.1.1 Optimize lowess fit: fraction of data to use when estimating y-values

```
[6]: ## Single Fold examination
for frac in [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4]:
    dRFEtools.optimize_lowess_plot(d, fold, outdir, frac=frac, step_size=0.05,
                                   classify=True, save_plot=True)
```



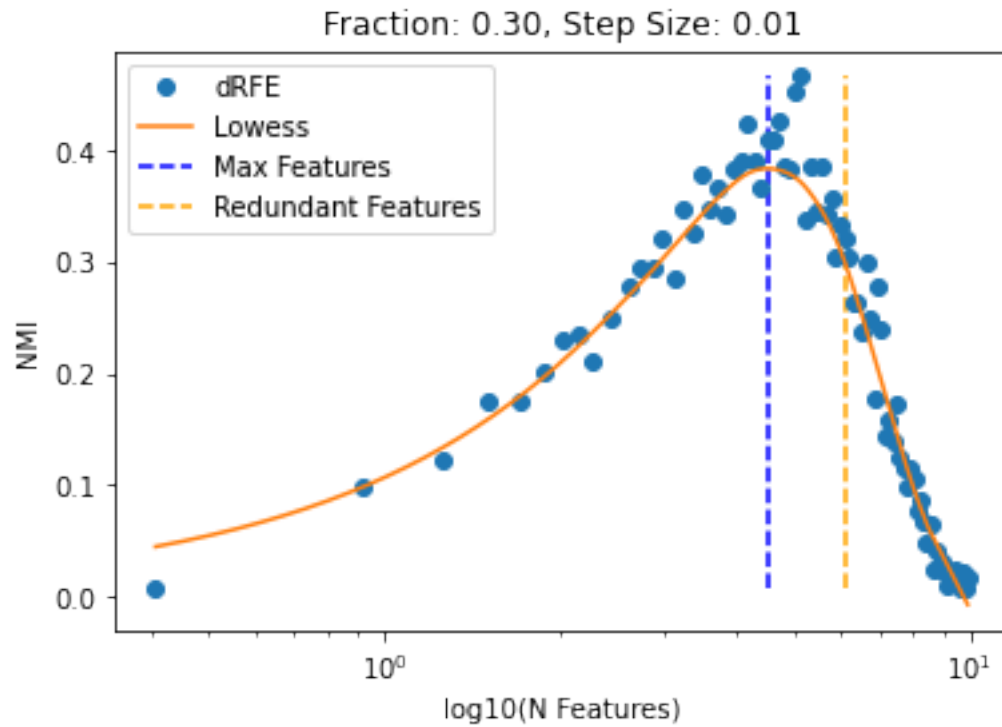


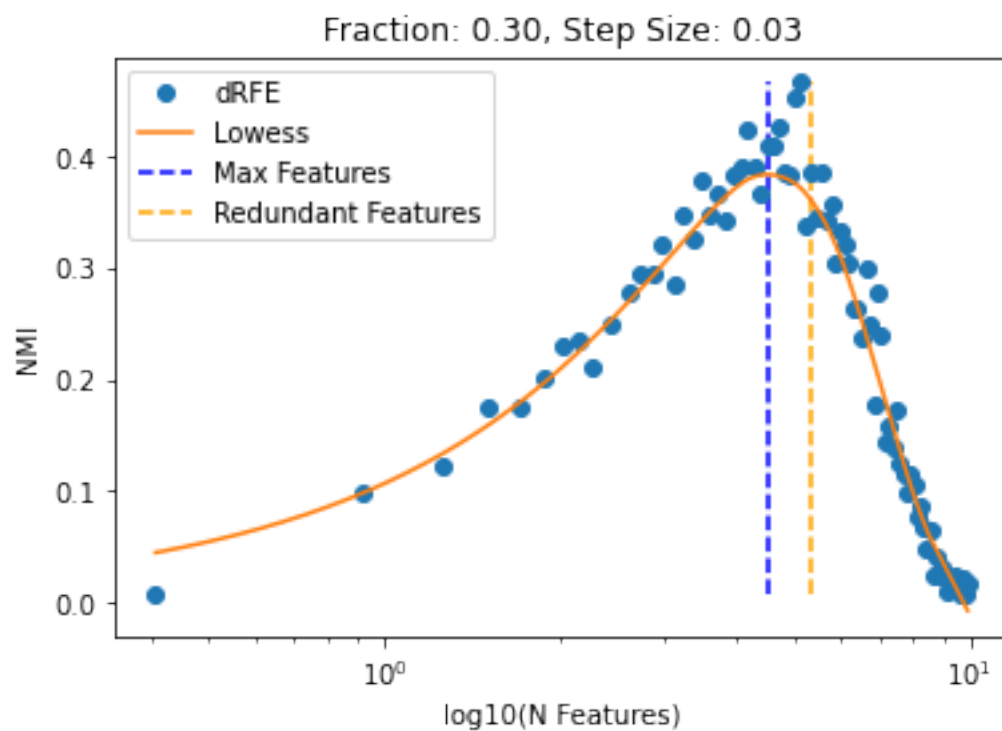
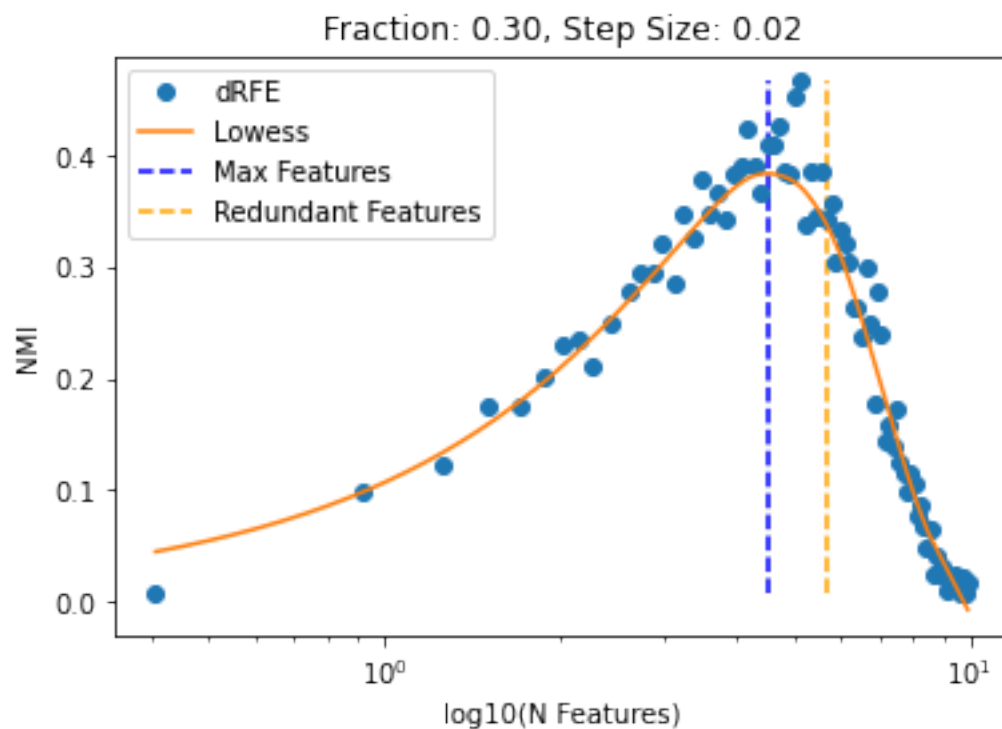


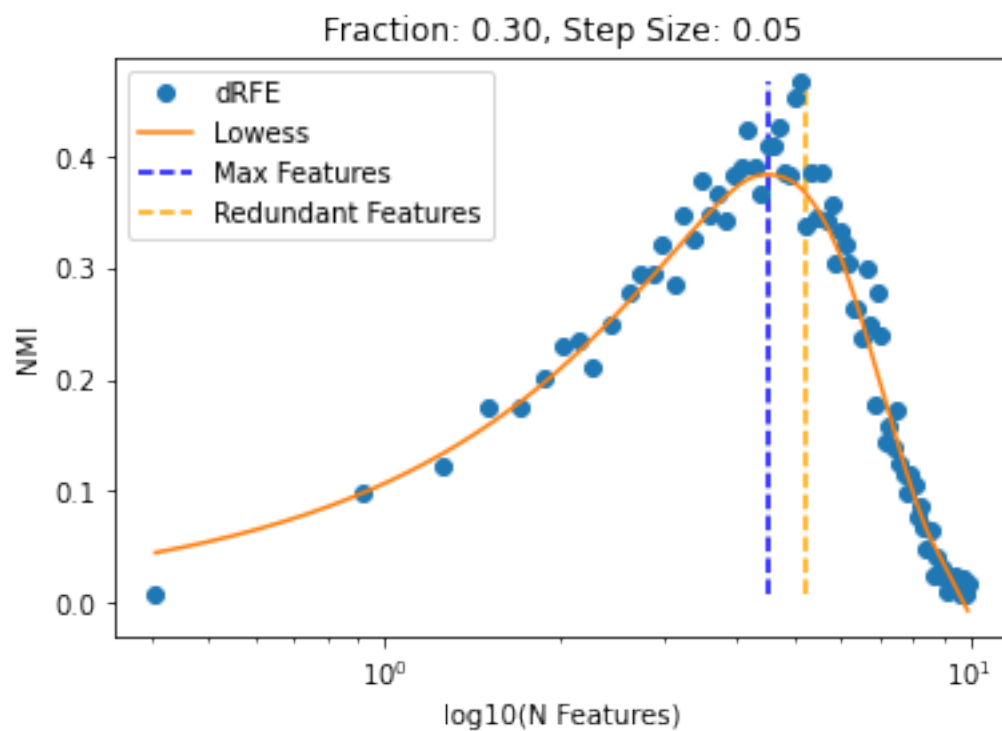
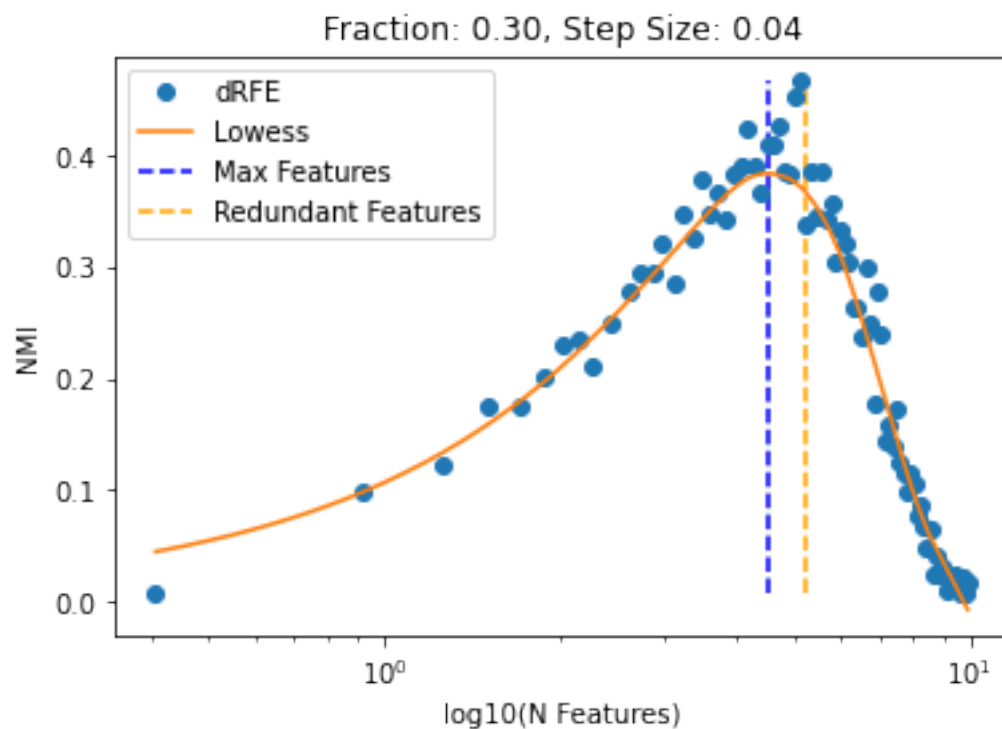


### 1.1.2 Optimize redundant selection: examine rate change

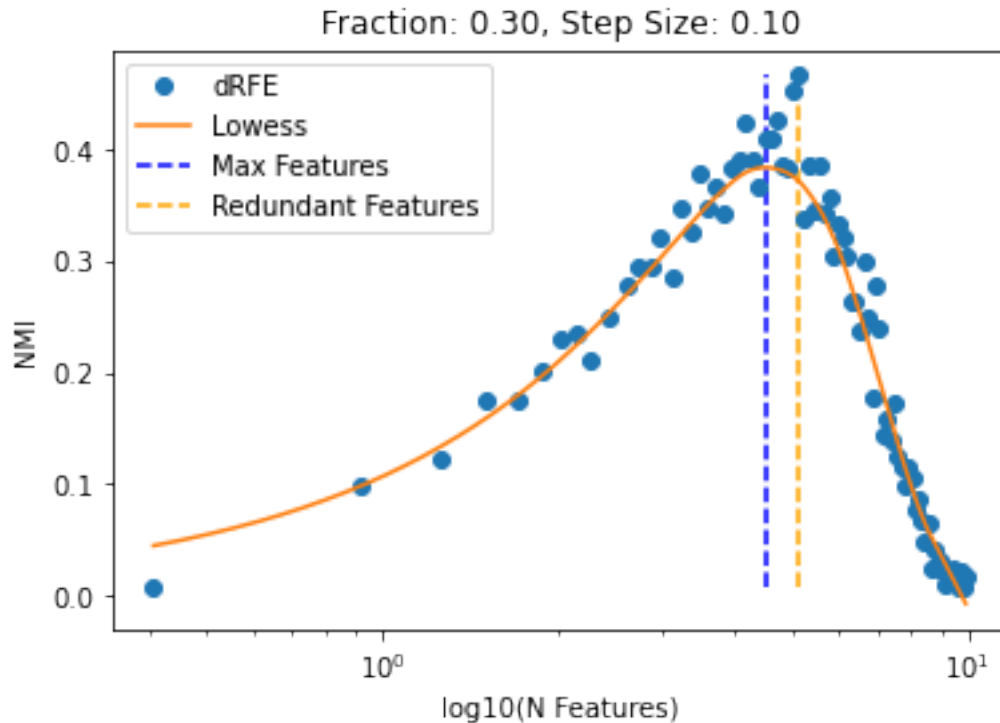
```
[7]: for step_size in [0.01, 0.02, 0.03, 0.04, 0.05, 0.1]:  
      dRFEtools.optimize_lowess_plot(d, fold, outdir, frac=0.3,   
      ↪ step_size=step_size,  
                                     classify=True, save_plot=True)
```











```
[8]: ## Highest value (without smoothing)
max(d, key=lambda x: d[x][1])
```

```
[8]: 170
```

```
[9]: print(dRFEtools.extract_max_lowess(d, frac=0.30))
## Using a conservative step size
dRFEtools.extract_redundant_lowess(d, frac=0.30, step_size=0.02)
```

```
(89, 4.4942386252808095)
```

```
[9]: (291, 5.675040005790547)
```

## 1.2 Optimize regression

```
[10]: outdir = 'regression/'
mkdir_p(outdir)
```

```
[11]: # Create a dataset with only 10 informative features
X, y = make_regression(
    n_samples=500, n_features=20000, n_informative=200, bias=0.2,
    n_targets=1, noise=10, random_state=13
)
cv = KFold(n_splits=5, shuffle=True, random_state=13)
```

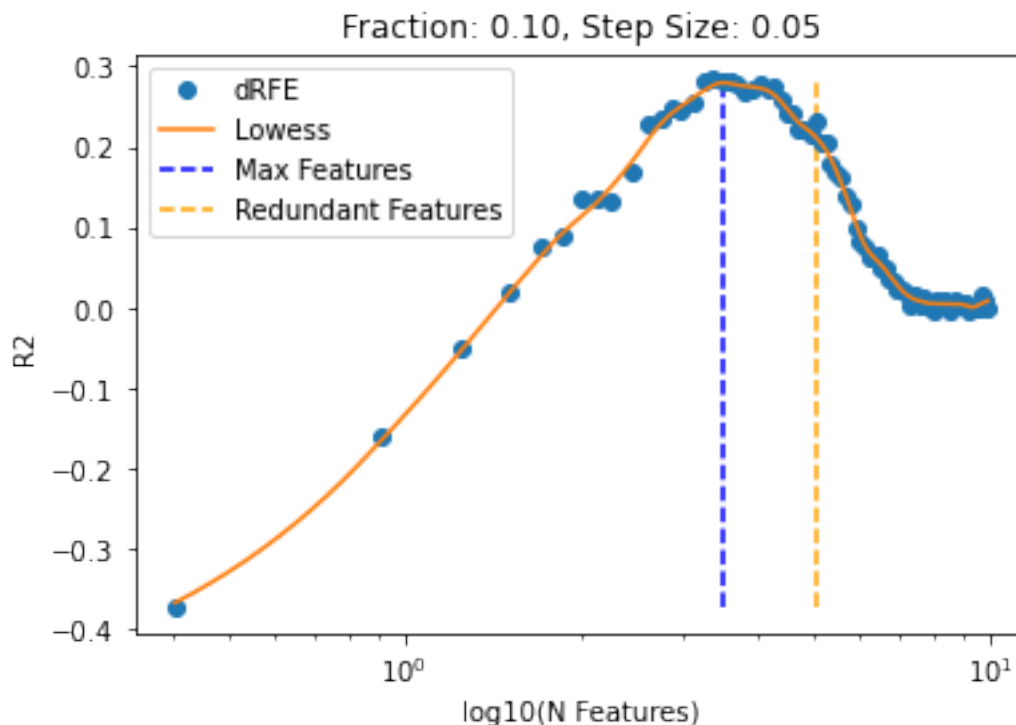
```
regr = dRFEtools.RandomForestRegressor(n_estimators=100, oob_score=True,
                                       n_jobs=-1, random_state=13)
```

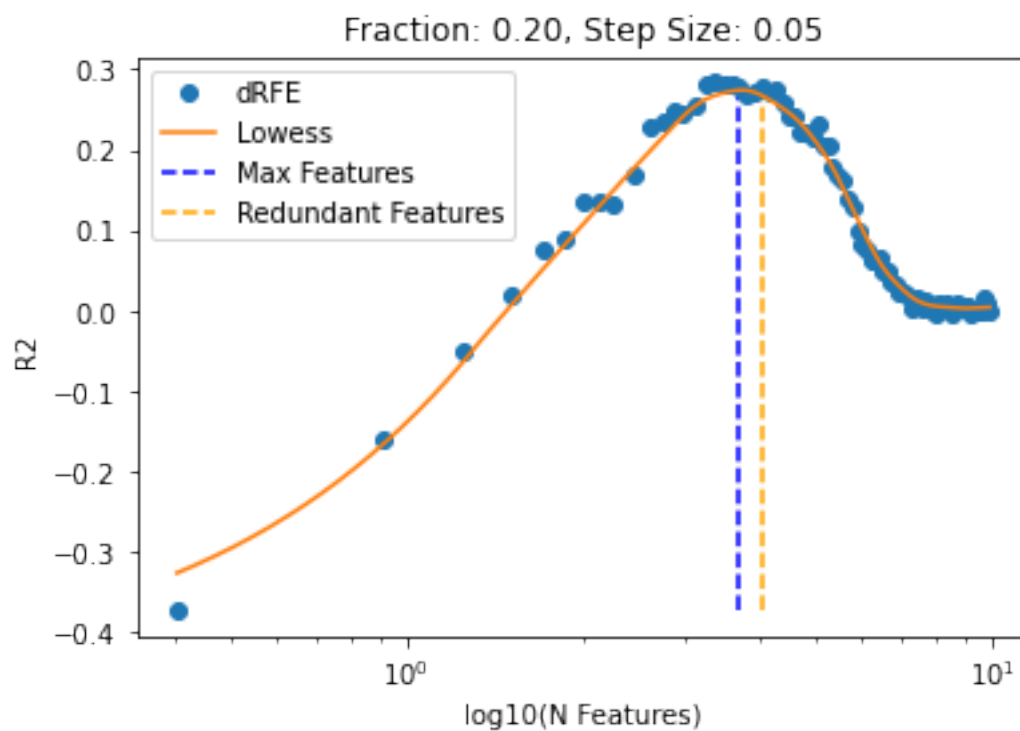
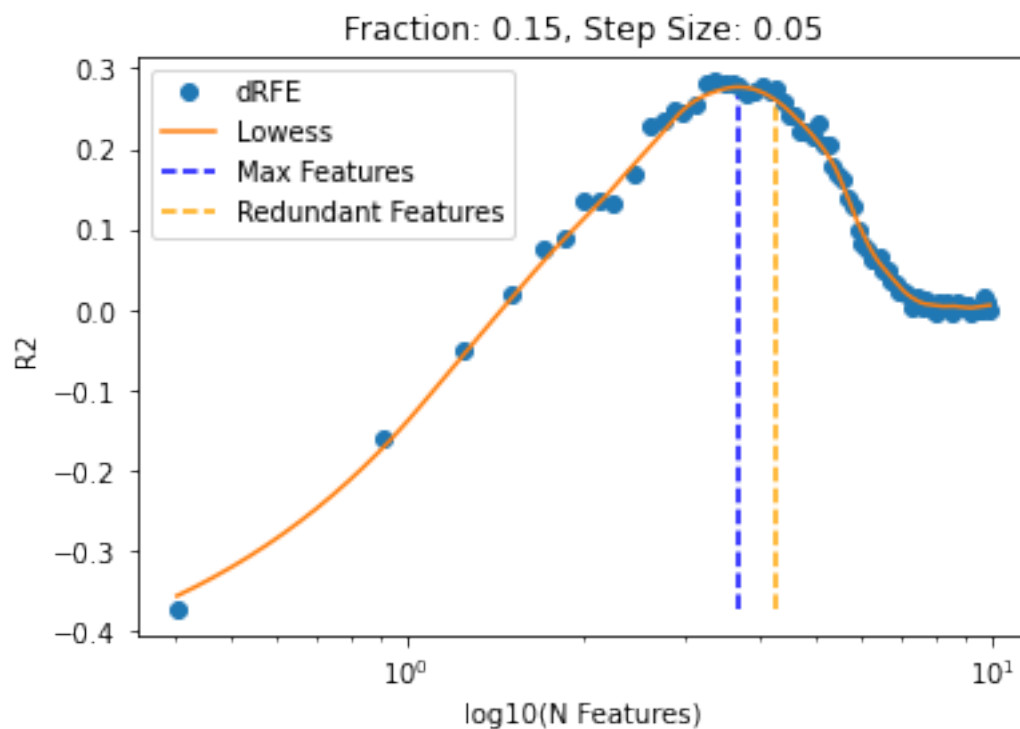
```
[12]: fold = 1
for train_index, test_index in cv.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    fold += 1
fold -= 1

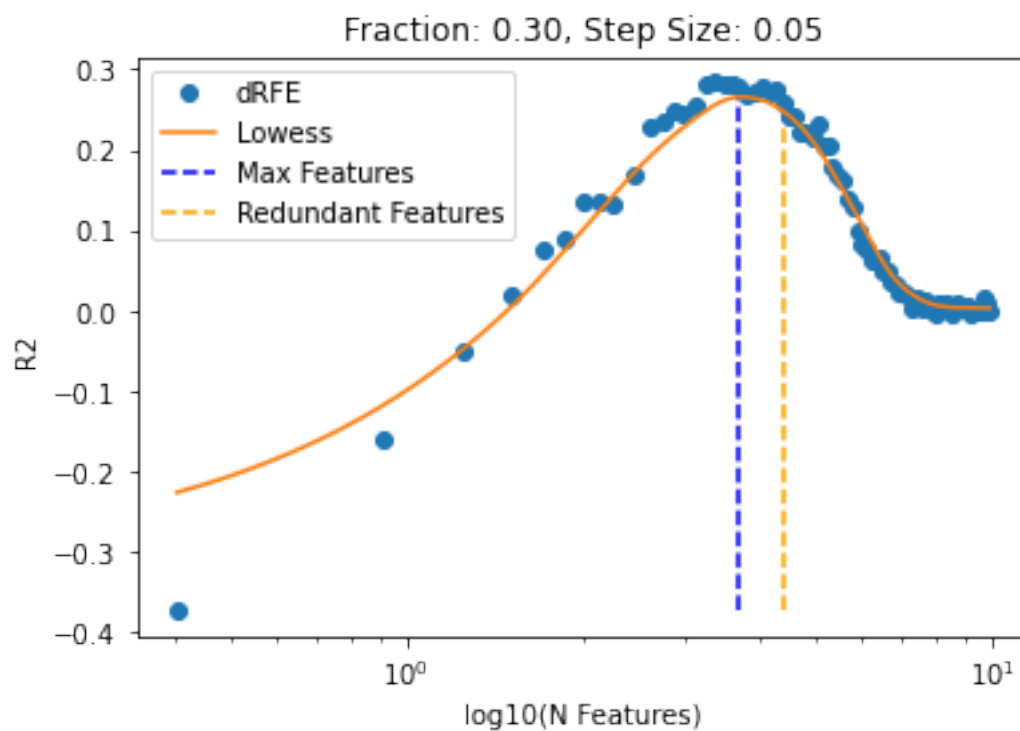
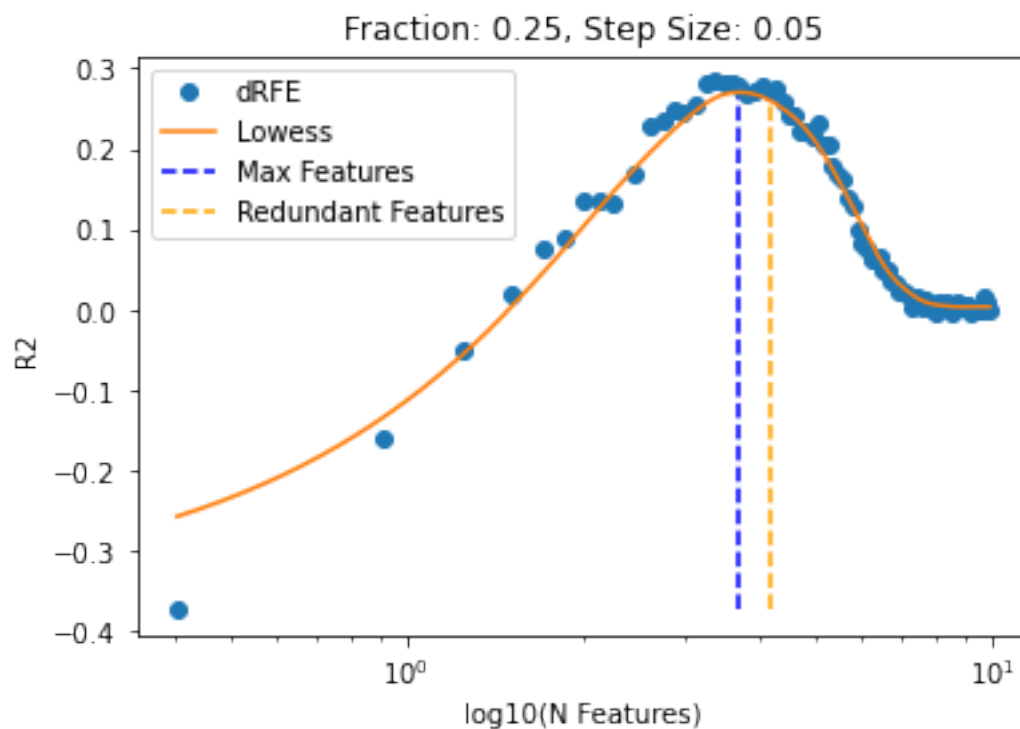
features = ["feature_%d" % x for x in range(X_train.shape[1])]
d, pfirst = dRFEtools.rf_rfe(regr, X_train, y_train, np.array(features),
                           fold, outdir, elimination_rate=0.1, RANK=False)
```

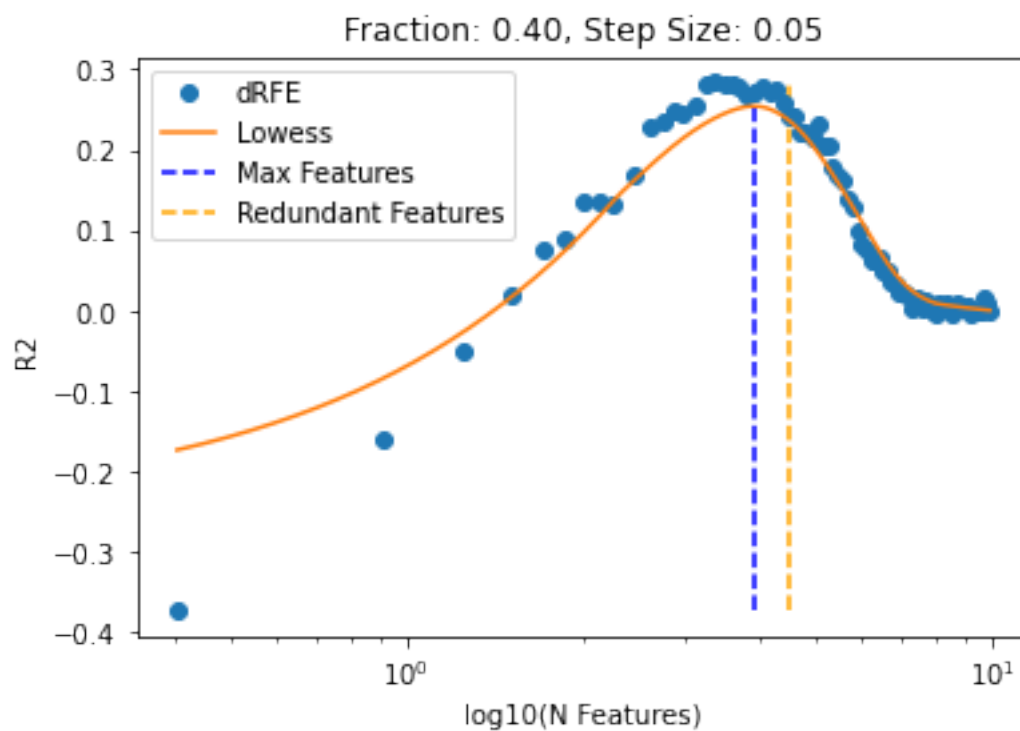
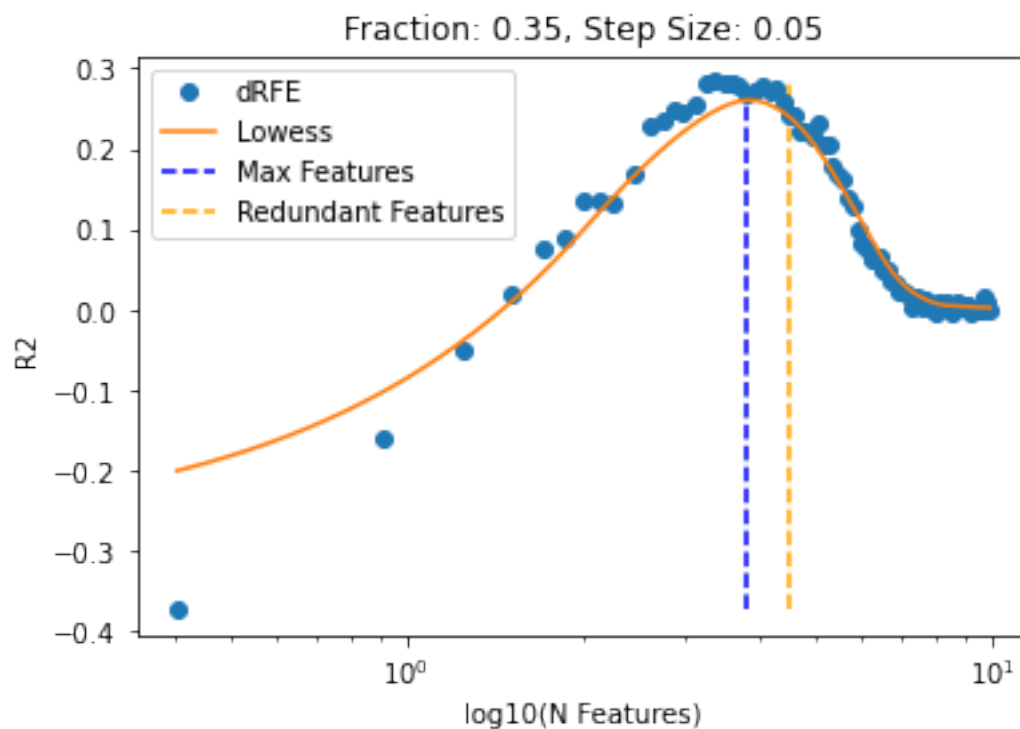
### 1.2.1 Optimize lowess fit: fraction of data to use when estimating y-values

```
[13]: ## Single Fold examination
for frac in [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4]:
    dRFEtools.optimize_lowess_plot(d, fold, outdir, frac=frac, step_size=0.05,
                                classify=False, save_plot=True)
```



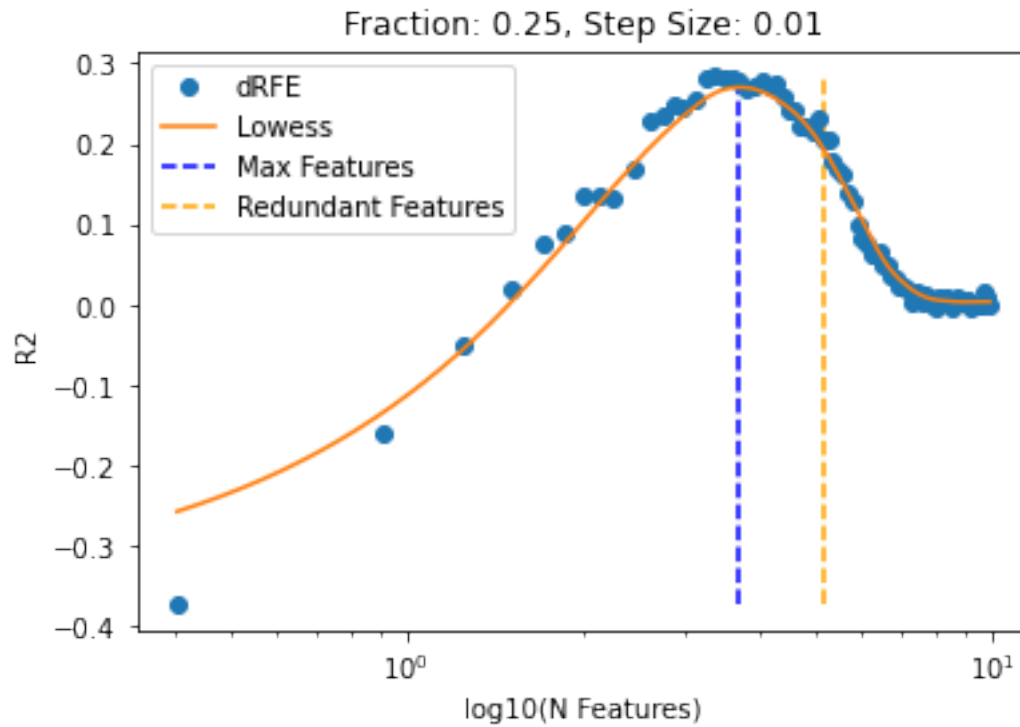


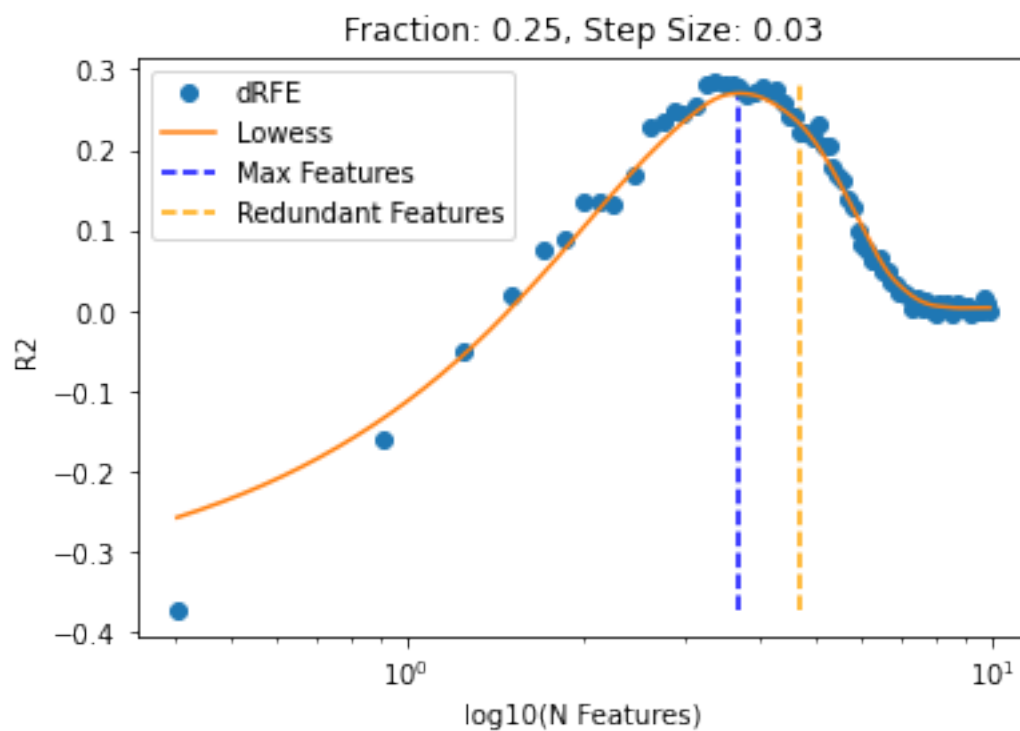
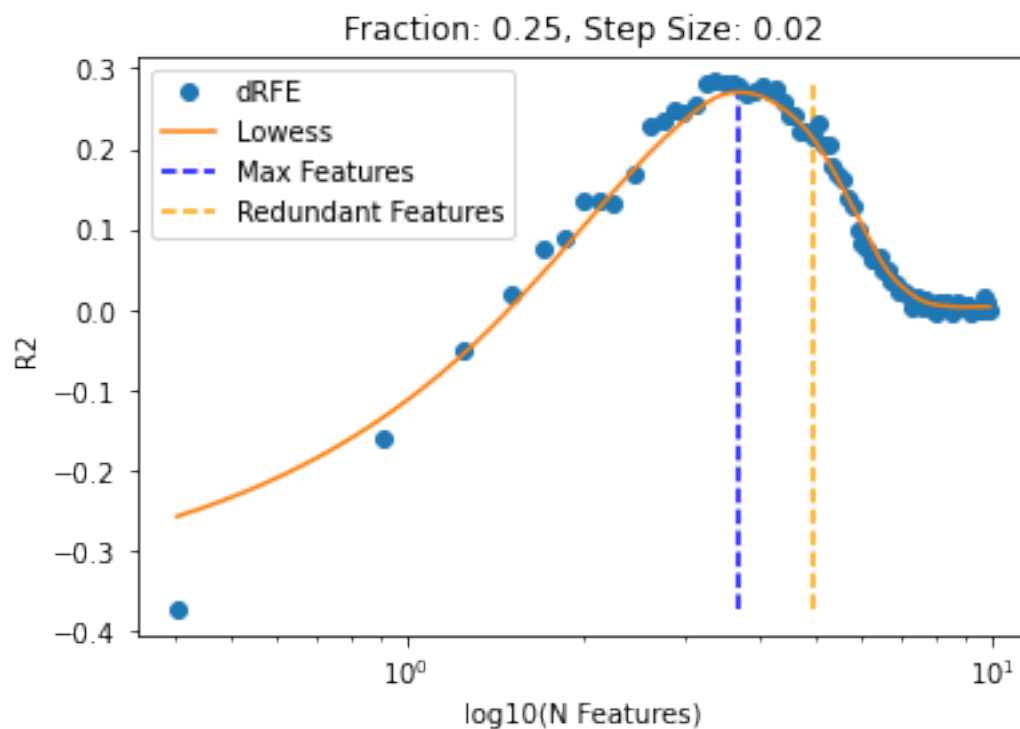


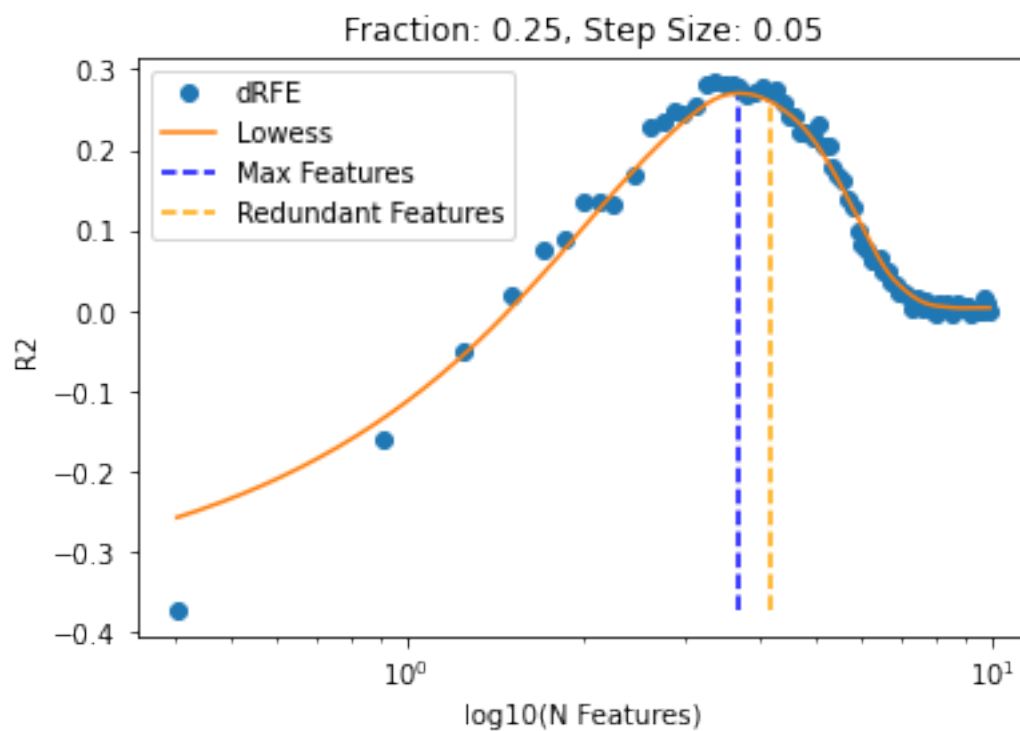
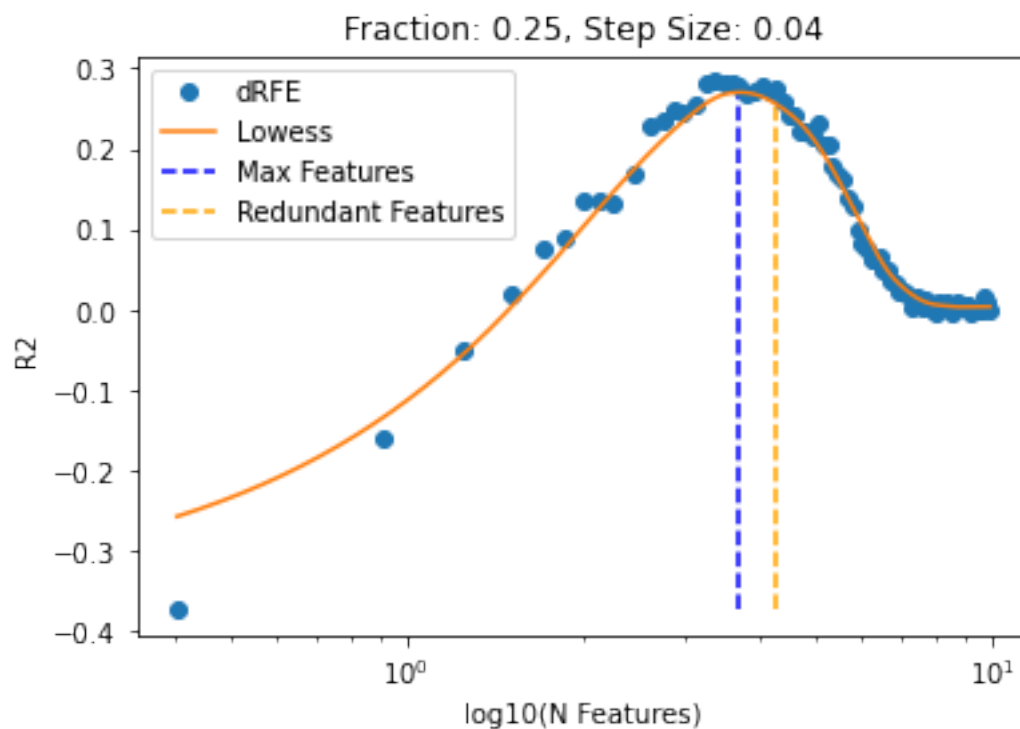


### 1.2.2 Optimize redundant selection: examine rate change

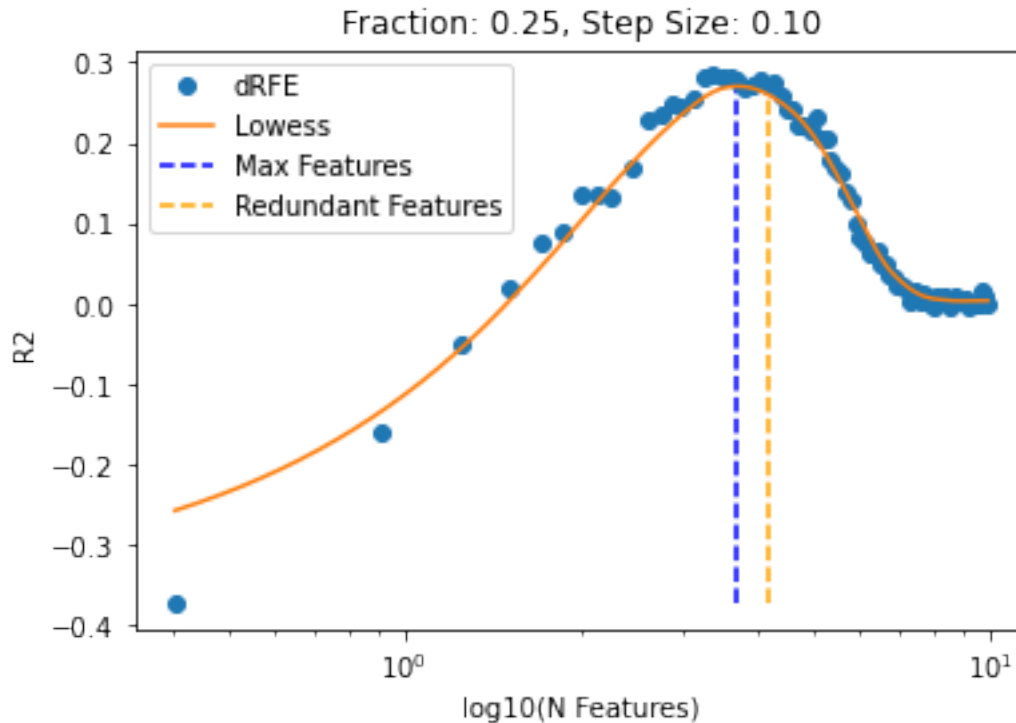
```
[14]: for step_size in [0.01, 0.02, 0.03, 0.04, 0.05, 0.1]:  
        dRFEtools.optimize_lowess_plot(d, fold, outdir, frac=0.25,   
        ↪step_size=step_size,   
                                     classify=False, save_plot=True)
```











```
[15]: ## Highest values without smoothing
max(d, key=lambda x: d[x][1])
```

[15]: 28

```
[16]: print(dRFEtools.extract_max_lowess(d, frac=0.25))
dRFEtools.extract_redundant_lowess(d, frac=0.25, step_size=0.02)
```

(40, 3.7013019741124933)

[16]: (137, 4.923623917106626)

### 1.3 Optimize classification: multi-class

```
[17]: outdir = 'multiclass/'
mkdir_p(outdir)
```

```
[18]: # Create a dataset with only 10 informative features
X, y = make_classification(
    n_samples=500, n_features=20000, n_informative=100, n_redundant=300,
    n_repeated=0, n_classes=4, n_clusters_per_class=1, random_state=13
)

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=13)
```

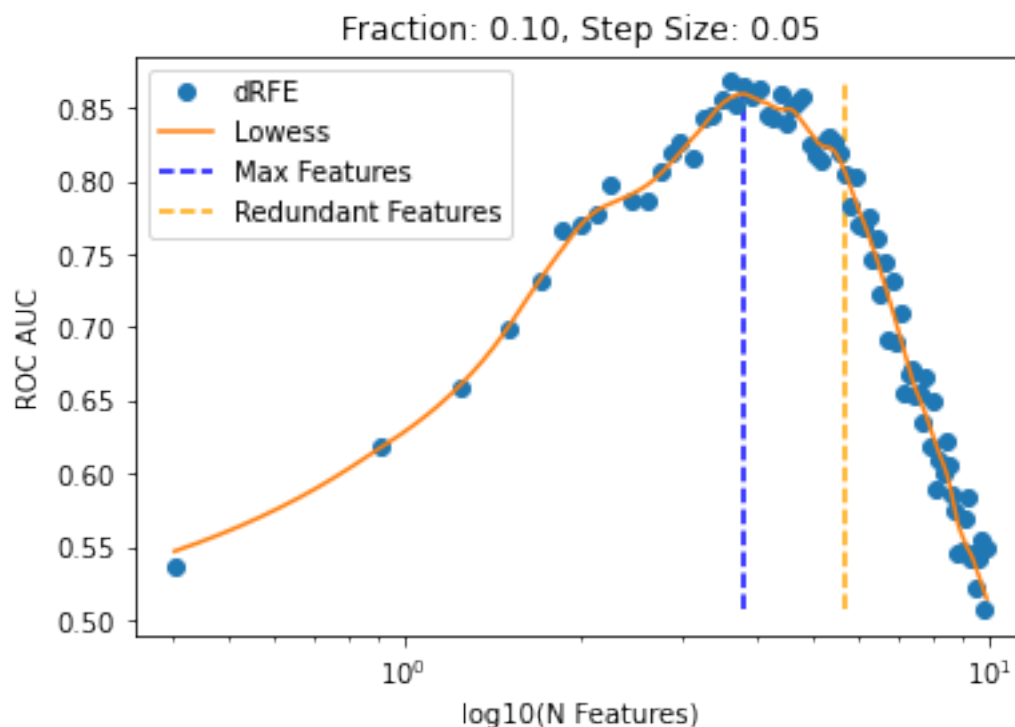
```
cla = dRFEtools.RandomForestClassifier(n_estimators=100, n_jobs=-1,
                                       oob_score=True, random_state=13)
```

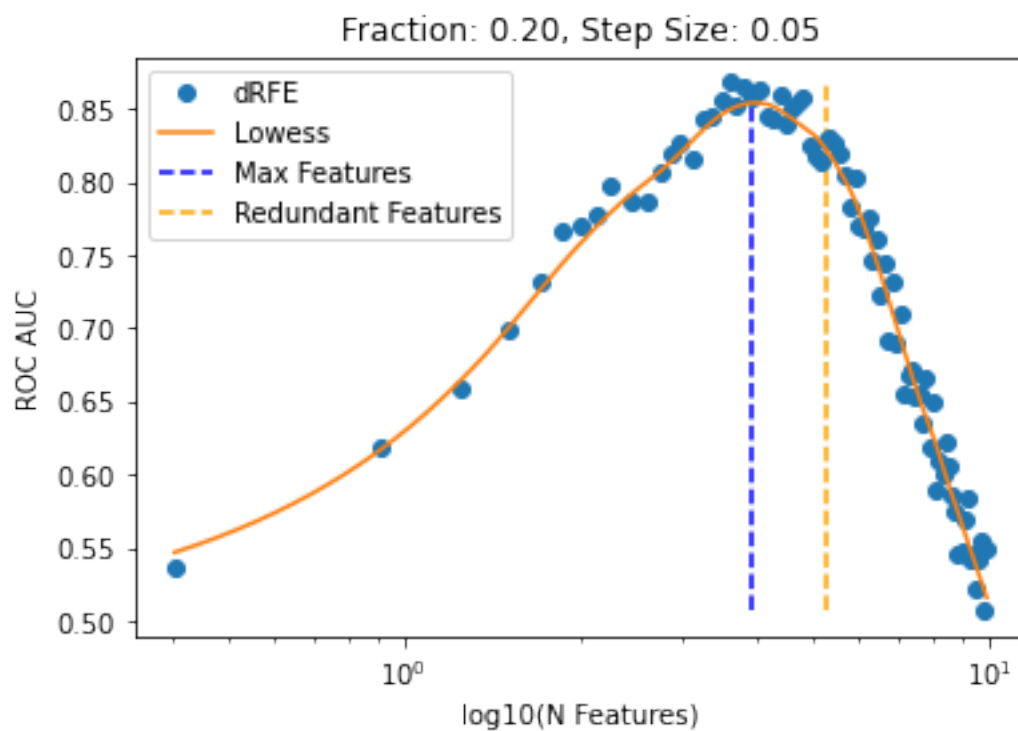
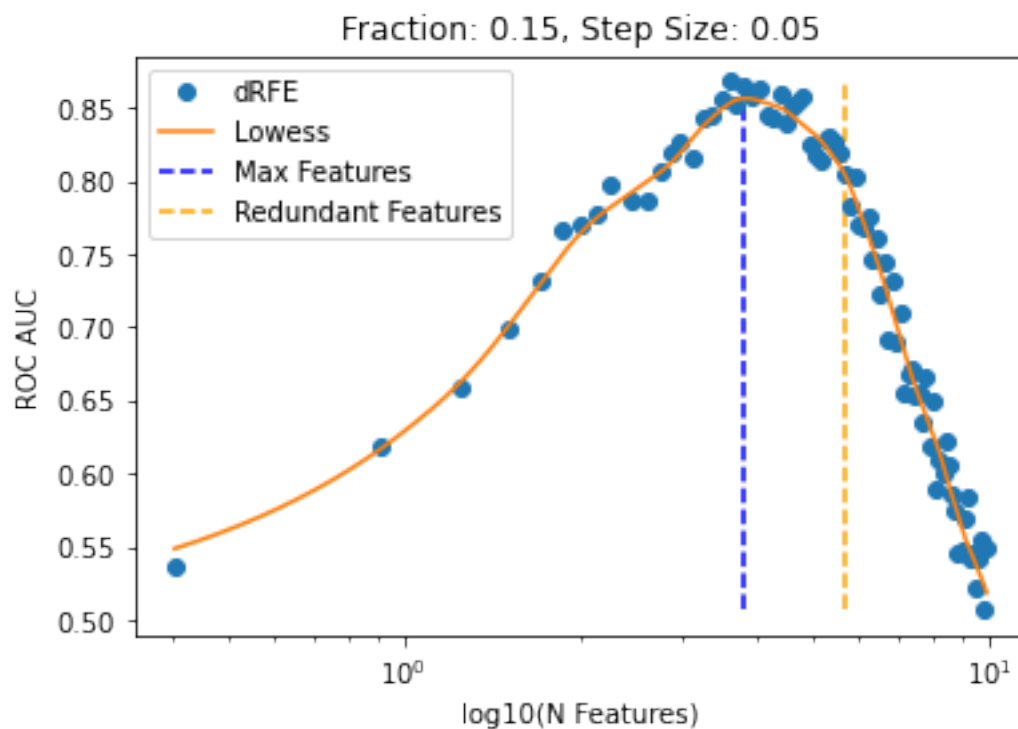
```
[19]: fold = 1
for train_index, test_index in cv.split(X, y):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]
    fold += 1
fold -= 1

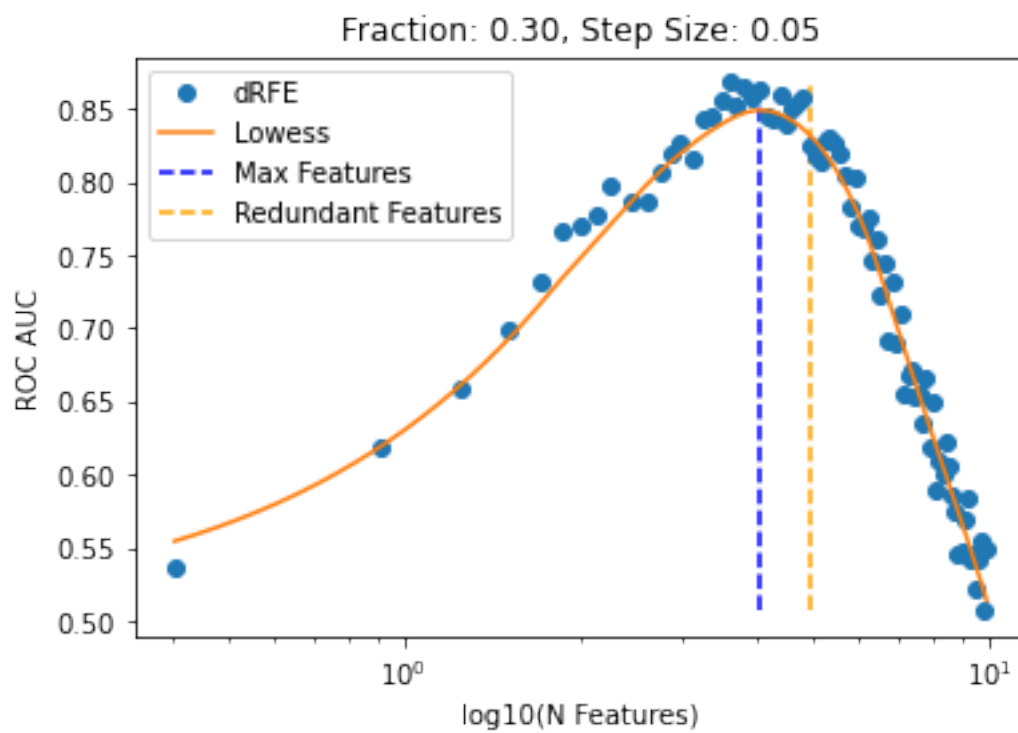
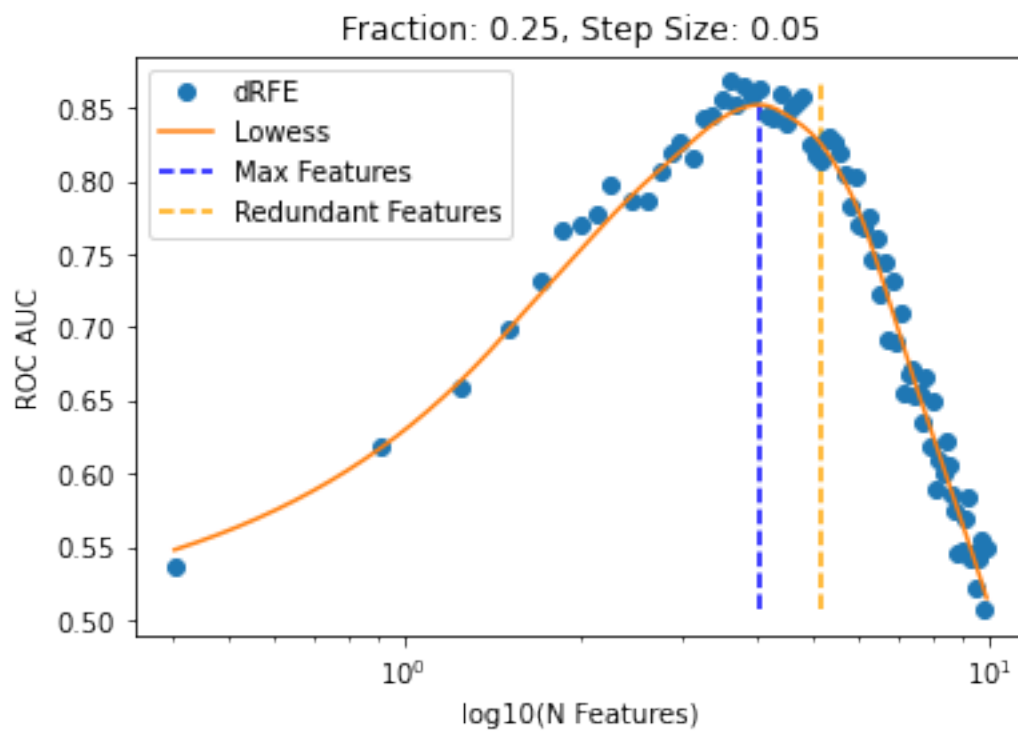
features = ["feature_%d" % x for x in range(X_train.shape[1])]
d, pfirst = dRFEtools.rf_rfe(cla, X_train, y_train, np.array(features),
                             fold, outdir, elimination_rate=0.1, RANK=False)
```

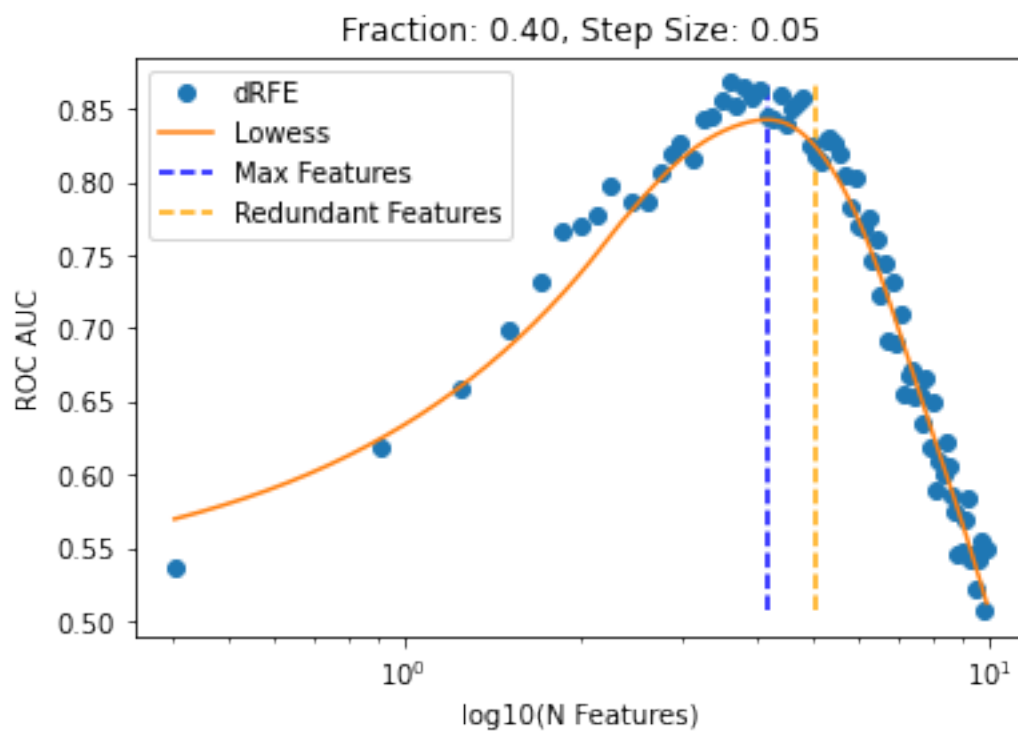
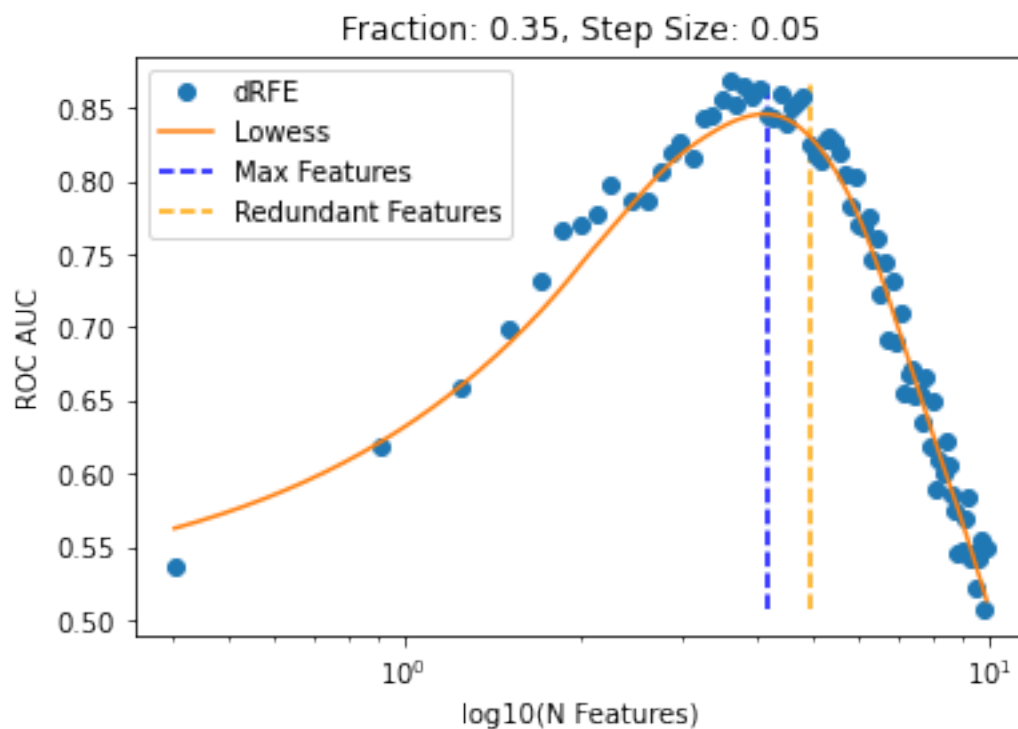
### 1.3.1 Optimize lowess fit: fraction of data to use when estimating y-values

```
[20]: ## Single Fold examination
for frac in [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4]:
    dRFEtools.optimize_lowess_plot(d, fold, outdir, frac=frac, step_size=0.05,
                                   classify=True, multi=True, save_plot=True)
```



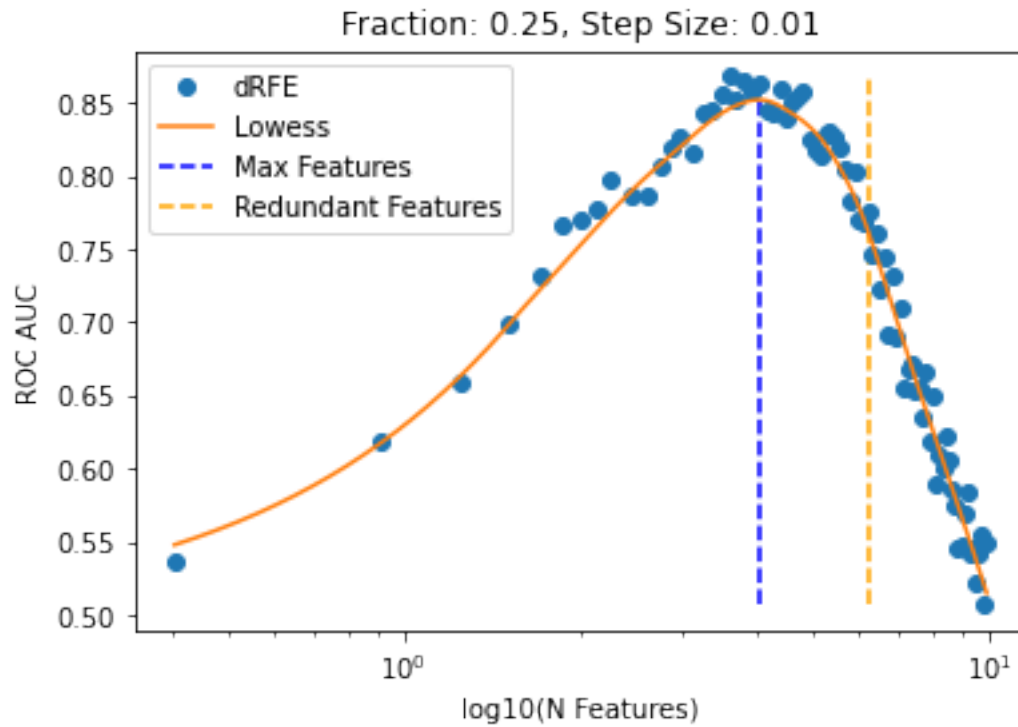


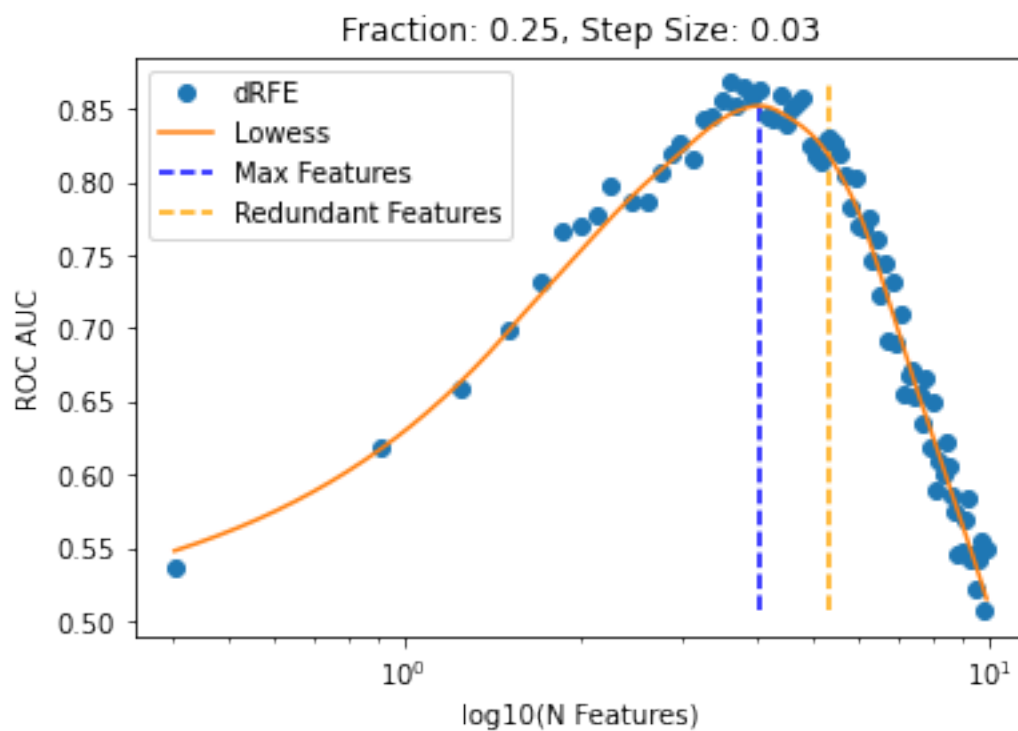
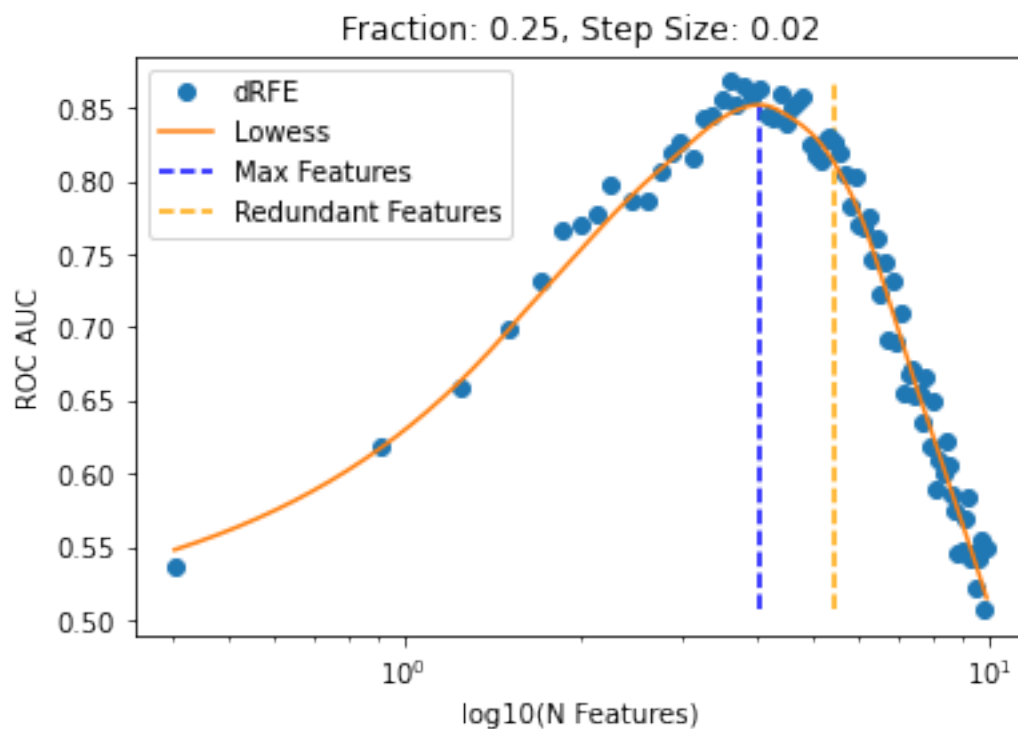


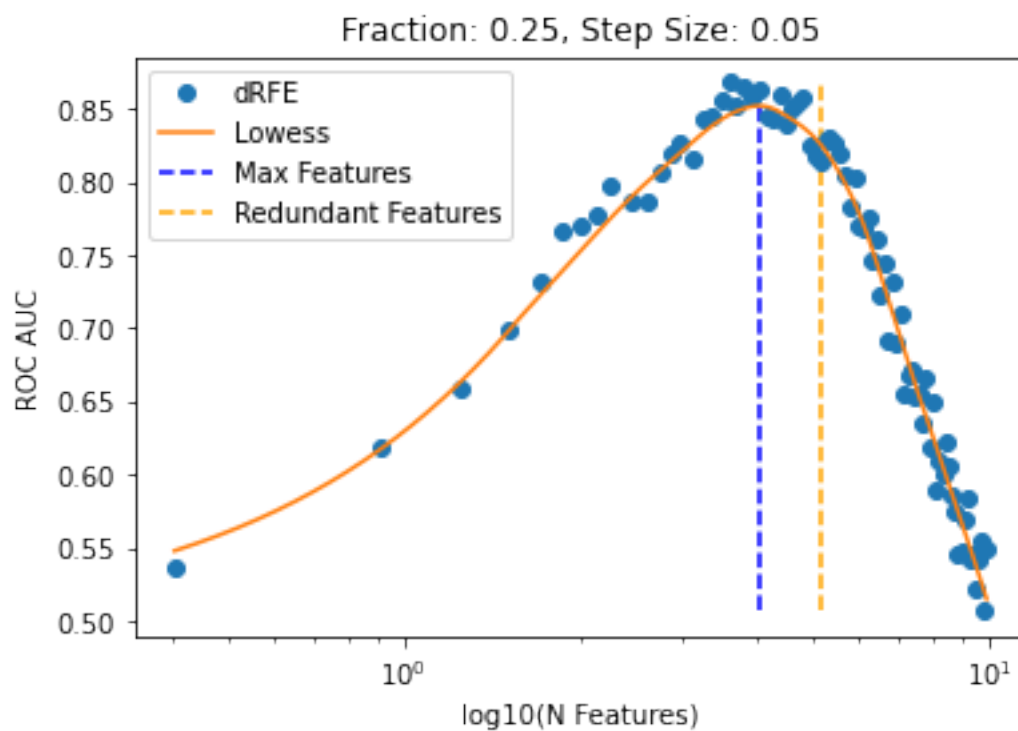
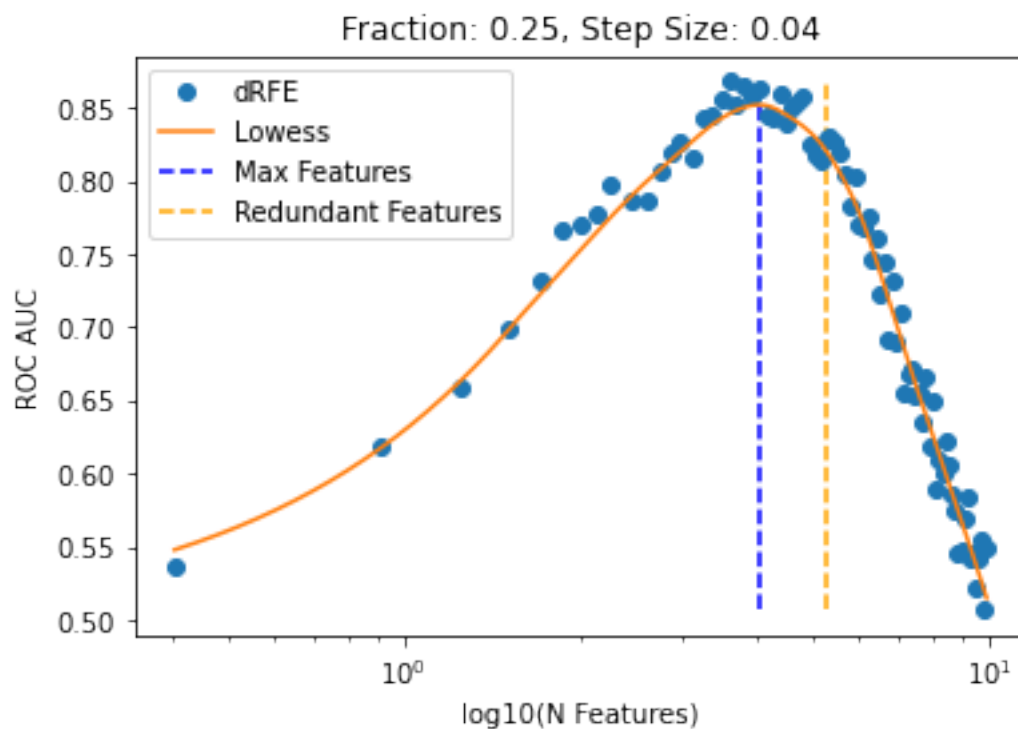


### 1.3.2 Optimize redundant selection: examine rate change

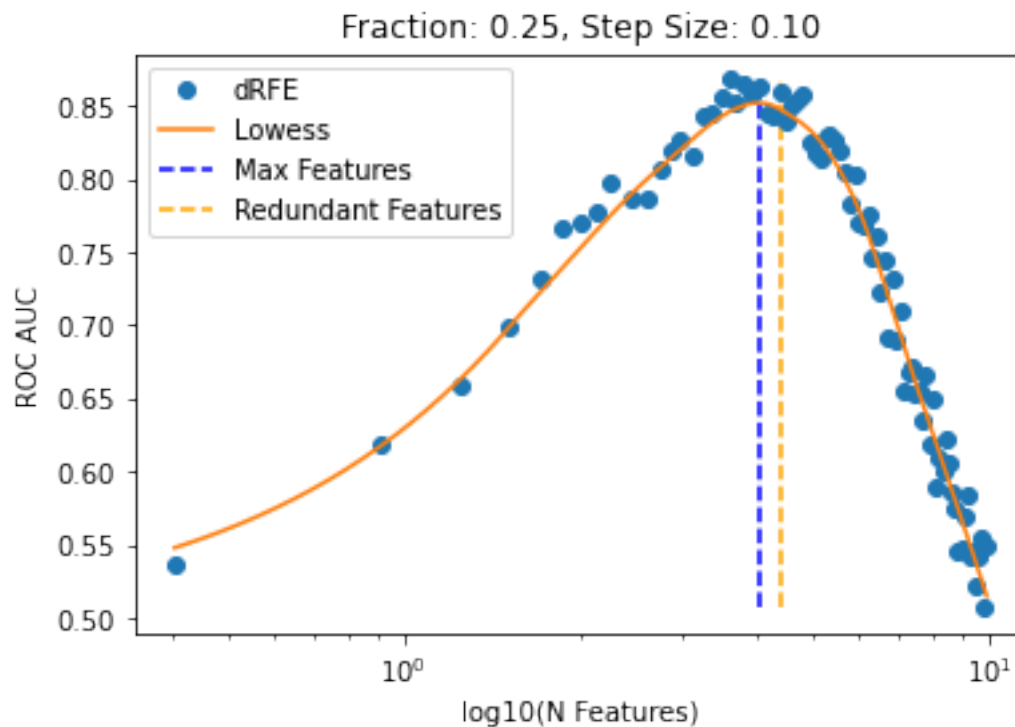
```
[21]: for step_size in [0.01, 0.02, 0.03, 0.04, 0.05, 0.1]:  
        dRFEtools.optimize_lowess_plot(d, fold, outdir, frac=0.25,  
        ↪step_size=step_size,  
                                     classify=True, multi=True, save_plot=True)
```











```
[22]: ## Highest value (without smoothing)
      max(d, key=lambda x: d[x][1])
```

```
[22]: 28
```

```
[23]: print(dRFETools.extract_max_lowess(d, frac=0.25))
      dRFETools.extract_redundant_lowess(d, frac=0.25, step_size=0.015)
```

```
(25, 3.2386784521643803)
```

```
[23]: (72, 4.283586561860629)
```

```
[ ]:
```