Part (a):

Checking the Test Accuracy and Time Taken(secs) of our Random Forests Implementation

Checking the Test Accuracy and Time Taken(secs) of our Random Forests Implementation ¶

In [7]:
```
#Took about 5 mins to run on my system
start_time = time.time()

forest = random_forest_algo(train_df,num_trees=100,bootstrap=0.80,max_feat=5,depth=20)
pred = rf_predictions(test_df.drop(57,axis=1),forest)
acc = rf_calculate_accuracy(test_df,pred)*100

end_time=time.time()

print("Test Accuracy: {} %".format(acc))
print("Time Taken: {} secs".format(round(end_time-start_time,3)))
```

```
Test Accuracy: 94.5691527878349 %
Time Taken: 379.75 secs
```

Test Accuracy : 94.56%

Time Taken : 379.75 secs

Checking Test Accuracy using Sklearn's RandomForestClassifier

Checking Test Accuracy using Sklearn's RandomForestClassifier

In [8]:
```
X = df.drop(57,axis=1)
y = df[57]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=101)

start_time = time.time()

rfc = RandomForestClassifier(n_estimators=100,max_features=5,max_depth=20,bootstrap=False,
                             max_samples=0.8,criterion='entropy',min_samples_split=3)
rfc.fit(X_train,y_train)
preds=rfc.predict(X_test)
acc = accuracy_score(preds,y_test)*100

end_time=time.time()

print("Test Accuracy: {} %".format(acc))
print("Time Taken: {} secs".format(round(end_time-start_time,3)))
```

```
Test Accuracy: 95.1484431571325 %
Time Taken: 1.0 secs
```

Test Accuracy : 95.14%

Time Taken : 1.0 secs

- So, based on test accuracy our model performs almost as good as the sklearn's model but the main differnce is the time taken by both the models. Sklearn's model is approx 380X faster than our model.

Part (b):

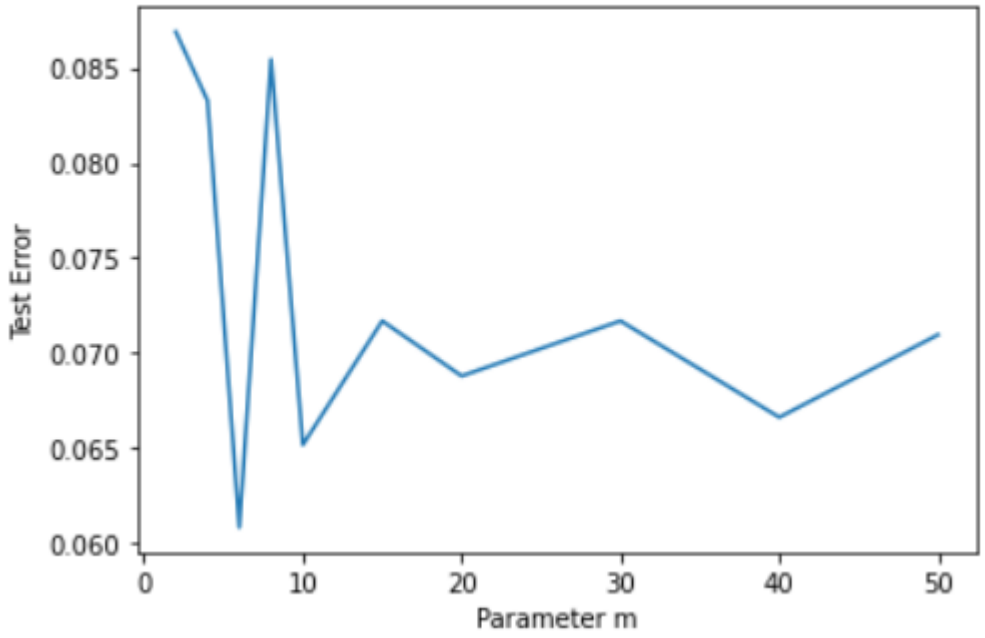
Note: Here due to Computation Infeasibility we are limiting the number of trees and depth to 10 (on full depth and 100 trees it takes hours to plot the behaviour) but that doesn't affect the sensitivity of Random Forests but accuracy of the model does get affected.

Sensitivity of Random Forests to the parameter m (Our Implementation of Random Forests)

	Number of Features	Test Error(%)
0	2	8.689356
1	4	8.327299
2	6	6.082549
3	8	8.544533
4	10	6.517017
5	15	7.168718
6	20	6.879073
7	30	7.168718
8	40	6.661839
9	50	7.096307

Plotting the same now,

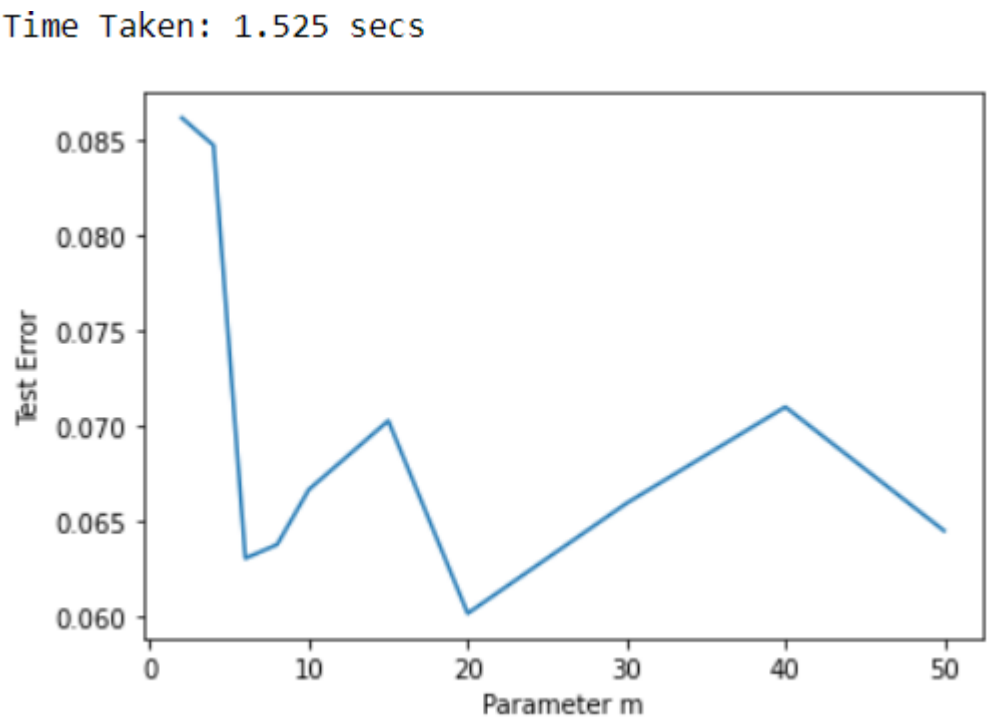
```
Time Taken: 890.746 secs
```



Sensitivity of Random Forests to the parameter m (Sklearn's RandomForestClassifier)

	Number of Features	Test Error(%)
0	2	8.616944
1	4	8.472122
2	6	6.299783
3	8	6.372194
4	10	6.661839
5	15	7.023896
6	20	6.010138
7	30	6.589428
8	40	7.096307
9	50	6.444605

Plotting the same now,



As can be seen from our models, [5-8] seems like the best values for the number of features and it gives least test error, Typically sqrt(m), log2(m), m/2, m/3 features works the best and gives the least test error as can be seen from the models.

Time Taken by our model : 890.746 secs

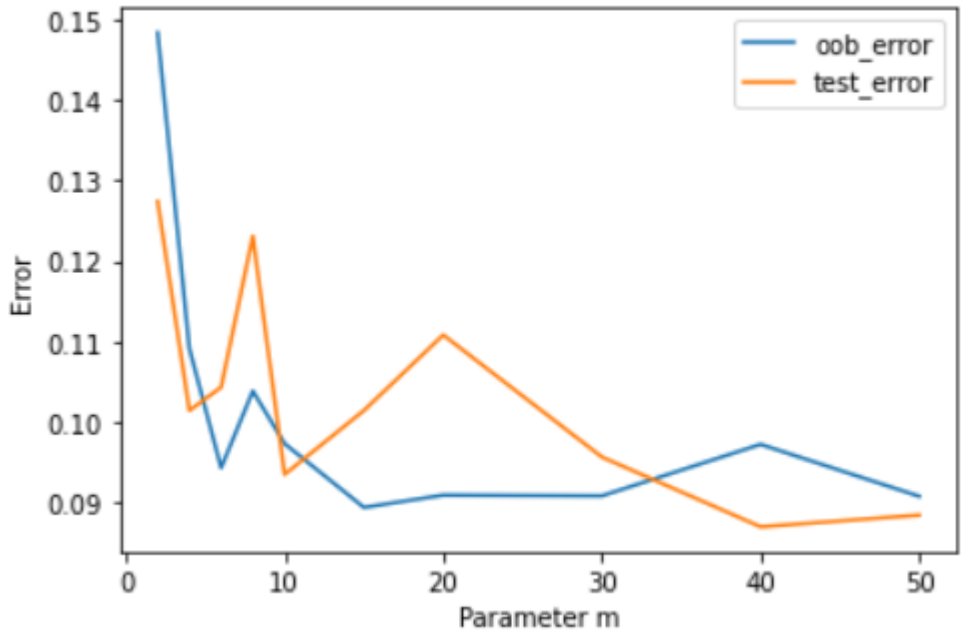
Time Taken by sklearn's model : 1.525 secs

Part (c):

Plot the OOB (out-of-bag) error and the Test Error against m (Our Implementation of Random Forests and OOB function)

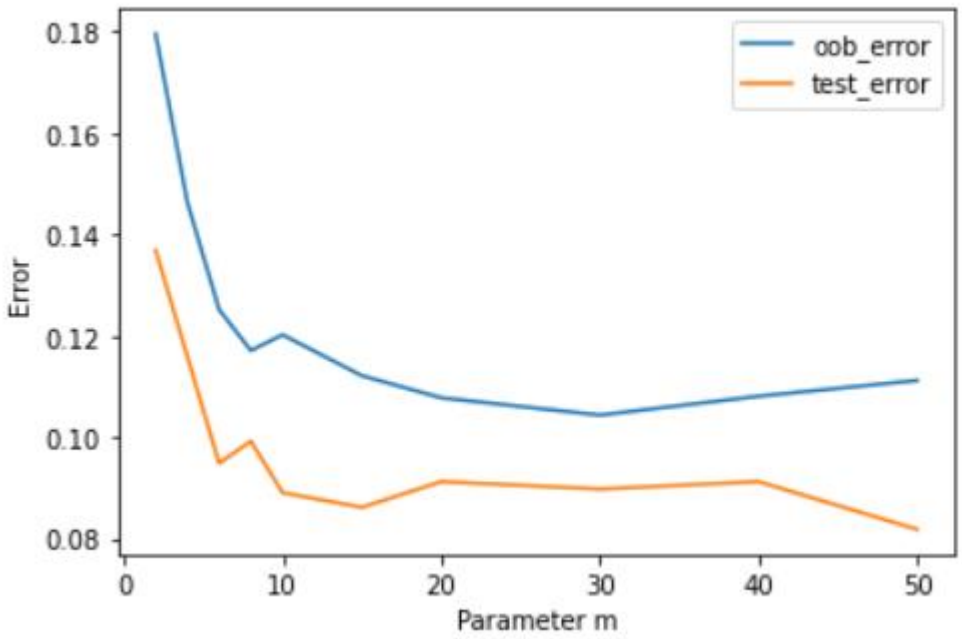
Note: Here due to Computation Infeasibility we are limiting the number of trees and depth to 5 (on full depth and 100 trees it takes hours to plot the behaviour) but that doesn't affect the sensitivity of Random Forests but accuracy of the model does get affected.

```
Time Taken: 1314.423 secs
```



Plot the OOB (out-of-bag) error and the Test Error against m (Sklearn's RandomForestClassifier and OOB function)

```
Time Taken: 0.702 secs
```



As can be seen from my model the oob-error and the test error is almost overlapping but test error gives less error for suitable values of m as it should be, as oob-error is calculated on less data compared to test error so it is not as robust and doesn't give a good estimate for the model's performance.

Time Taken by our model : 1314.423 secs

Time Taken by sklearn's model : 0.702 secs