QUESTION 5.a

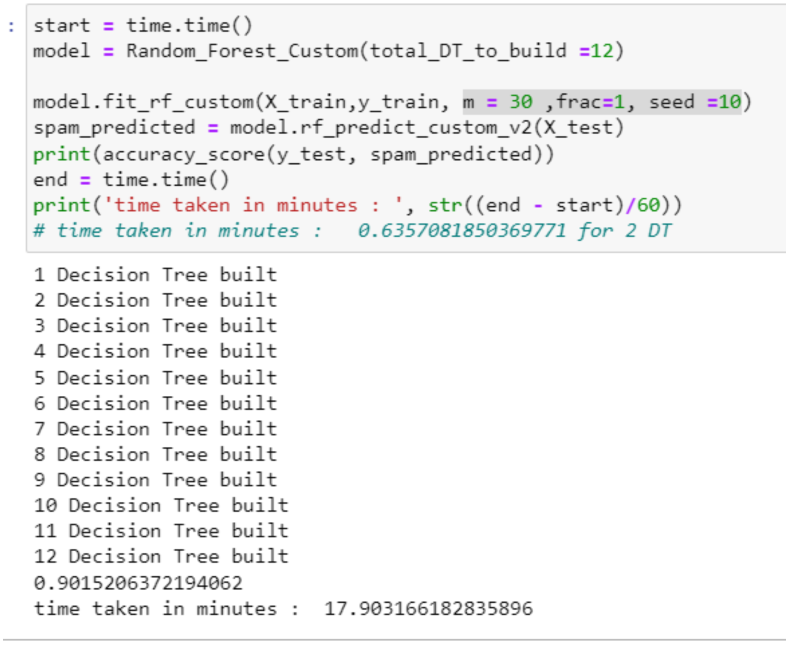
Please check the below notebooks notebooks-

1.**NOTEBOOK 1** (THE MAIN NOTEBOOK YOU SHOULD RUN to verify my code) : **Q5_ML_Assignment_RF_CS21MDS14025_SELECTED.ipynb** for one optimal result of Custom RF. (RUNS QUICKLY) – (here Question 5.b implemented in sklearn)

2. OPTIONALLY YOU CAN CHECK - **NOTEBOOK 2** - **Q5_ML_Assignment_RF_CS21MDS14025_DETAILED_NOTEBOOK.ipynb** all the different hyperparameter results. (here Question 5.b implemented with Custom RF function written by me, rest all are same)

From Notebook 2, the best accuracy is for - **m** = 30 ,**frac**=1(i.e. n out of n records were selected with replacement) , seed =10.

For 12 trees, **sklearn gives 94% accuracy, my custom implementation gives 90%**

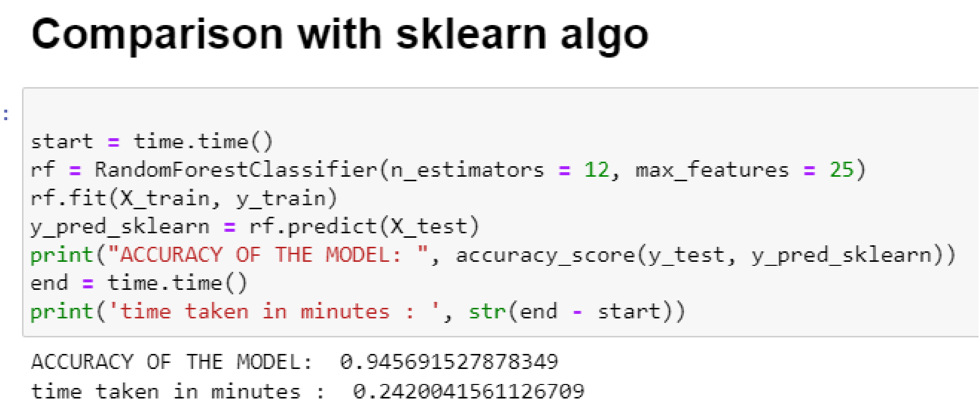


```
start = time.time()
model = Random_Forest_Custom(total_DT_to_build =12)

model.fit_rf_custom(X_train,y_train, m = 30 ,frac=1, seed =10)
spam_predicted = model.rf_predict_custom_v2(X_test)
print(accuracy_score(y_test, spam_predicted))
end = time.time()
print('time taken in minutes : ', str((end - start)/60))
# time taken in minutes :   0.6357081850369771 for 2 DT
```

```
1 Decision Tree built
2 Decision Tree built
3 Decision Tree built
4 Decision Tree built
5 Decision Tree built
6 Decision Tree built
7 Decision Tree built
8 Decision Tree built
9 Decision Tree built
10 Decision Tree built
11 Decision Tree built
12 Decision Tree built
0.9015206372194062
time taken in minutes :  17.903166182835896
```

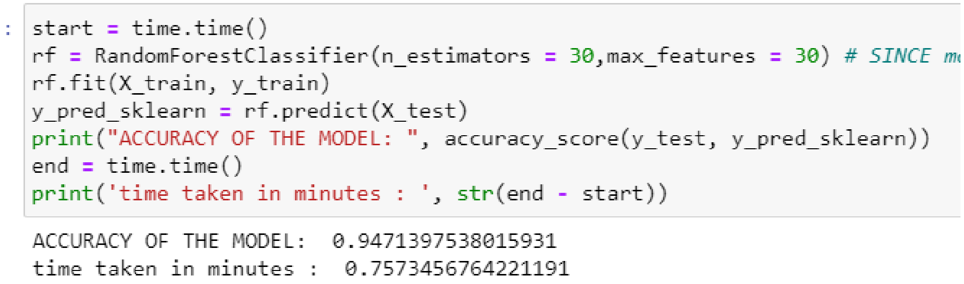
SKLEARN performance ---



Comparison with sklearn algo

```
start = time.time()
rf = RandomForestClassifier(n_estimators = 12, max_features = 25)
rf.fit(X_train, y_train)
y_pred_sklearn = rf.predict(X_test)
print("ACCURACY OF THE MODEL: ", accuracy_score(y_test, y_pred_sklearn))
end = time.time()
print('time taken in minutes : ', str(end - start))
```

```
ACCURACY OF THE MODEL:  0.945691527878349
time taken in minutes :  0.2420041561126709
```



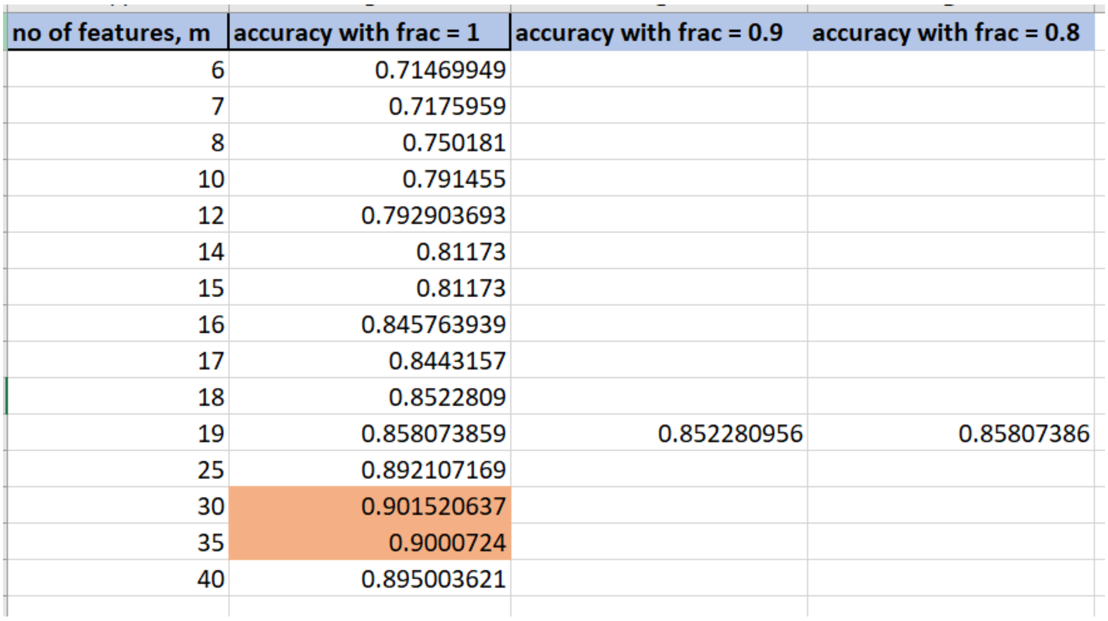
```
start = time.time()
rf = RandomForestClassifier(n_estimators = 30,max_features = 30) # SINCE m
rf.fit(X_train, y_train)
y_pred_sklearn = rf.predict(X_test)
print("ACCURACY OF THE MODEL: ", accuracy_score(y_test, y_pred_sklearn))
end = time.time()
print('time taken in minutes : ', str(end - start))
```

```
ACCURACY OF THE MODEL:  0.9471397538015931
time taken in minutes :  0.7573456764221191
```

QUESTION 5.b

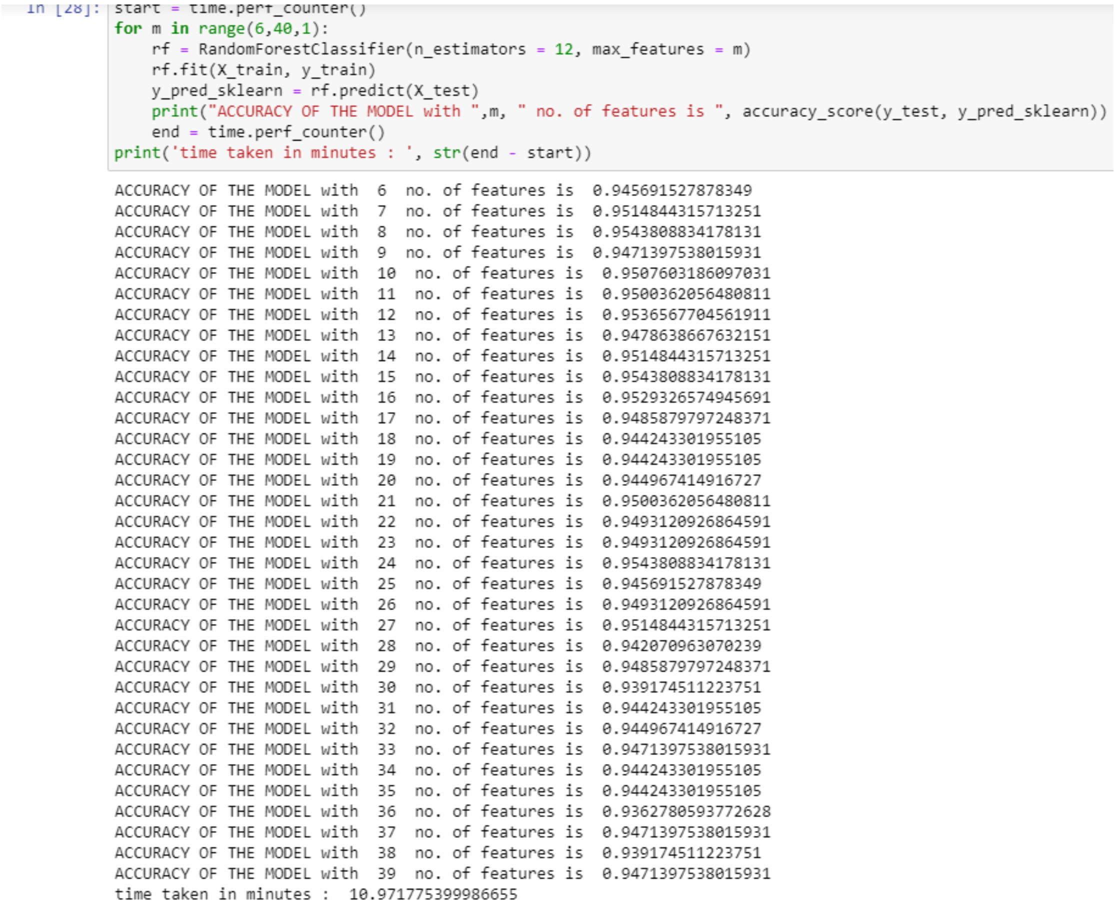
We can see as we increase the m, **the no of features considered for splitting for each DT- the accuracy increases…at some point reaches a peak, and then starts decreasing**

1. Using Custom RF Implementation – NOTEBOOK 2 - **Q5_ML_Assignment_RF_CS21MDS14025_DETAILED_NOTEBOOK**.ipynb



no of features, m	accuracy with frac = 1	accuracy with frac = 0.9	accuracy with frac = 0.8
6	0.71469949		
7	0.7175959		
8	0.750181		
10	0.791455		
12	0.792903693		
14	0.81173		
15	0.81173		
16	0.845763939		
17	0.8443157		
18	0.8522809		
19	0.858073859	0.852280956	0.85807386
25	0.892107169		
30	0.901520637		
35	0.9000724		
40	0.895003621		

2. Using SKLEARN RF – Cannot see any patterns- **Q5_ML_Assignment_RF_CS21MDS14025_SELECTED.ipynb**



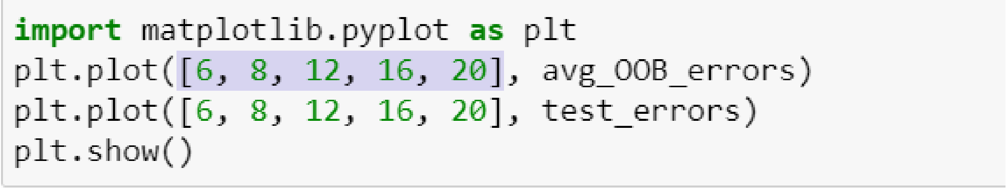
```
In [28]: start = time.perf_counter()
for m in range(6,40,1):
    rf = RandomForestClassifier(n_estimators = 12, max_features = m)
    rf.fit(X_train, y_train)
    y_pred_sklearn = rf.predict(X_test)
    print("ACCURACY OF THE MODEL with ",m, " no. of features is ", accuracy_score(y_test, y_pred_sklearn))
    end = time.perf_counter()
print('time taken in minutes : ', str(end - start))
```

```
ACCURACY OF THE MODEL with  6  no. of features is  0.945691527878349
ACCURACY OF THE MODEL with  7  no. of features is  0.9514844315713251
ACCURACY OF THE MODEL with  8  no. of features is  0.9543808834178131
ACCURACY OF THE MODEL with  9  no. of features is  0.9471397538015931
ACCURACY OF THE MODEL with  10  no. of features is  0.9507603186097031
ACCURACY OF THE MODEL with  11  no. of features is  0.9500362056480811
ACCURACY OF THE MODEL with  12  no. of features is  0.9536567704561911
ACCURACY OF THE MODEL with  13  no. of features is  0.9478638667632151
ACCURACY OF THE MODEL with  14  no. of features is  0.9514844315713251
ACCURACY OF THE MODEL with  15  no. of features is  0.9543808834178131
ACCURACY OF THE MODEL with  16  no. of features is  0.9529326574945691
ACCURACY OF THE MODEL with  17  no. of features is  0.9485879797248371
ACCURACY OF THE MODEL with  18  no. of features is  0.944243301955105
ACCURACY OF THE MODEL with  19  no. of features is  0.944243301955105
ACCURACY OF THE MODEL with  20  no. of features is  0.944967414916727
ACCURACY OF THE MODEL with  21  no. of features is  0.9500362056480811
ACCURACY OF THE MODEL with  22  no. of features is  0.9493120926864591
ACCURACY OF THE MODEL with  23  no. of features is  0.9493120926864591
ACCURACY OF THE MODEL with  24  no. of features is  0.9543808834178131
ACCURACY OF THE MODEL with  25  no. of features is  0.945691527878349
ACCURACY OF THE MODEL with  26  no. of features is  0.9493120926864591
ACCURACY OF THE MODEL with  27  no. of features is  0.9514844315713251
ACCURACY OF THE MODEL with  28  no. of features is  0.942070963070239
ACCURACY OF THE MODEL with  29  no. of features is  0.9485879797248371
ACCURACY OF THE MODEL with  30  no. of features is  0.939174511223751
ACCURACY OF THE MODEL with  31  no. of features is  0.944243301955105
ACCURACY OF THE MODEL with  32  no. of features is  0.944967414916727
ACCURACY OF THE MODEL with  33  no. of features is  0.9471397538015931
ACCURACY OF THE MODEL with  34  no. of features is  0.944243301955105
ACCURACY OF THE MODEL with  35  no. of features is  0.944243301955105
ACCURACY OF THE MODEL with  36  no. of features is  0.9362780593772628
ACCURACY OF THE MODEL with  37  no. of features is  0.9471397538015931
ACCURACY OF THE MODEL with  38  no. of features is  0.939174511223751
ACCURACY OF THE MODEL with  39  no. of features is  0.9471397538015931
time taken in minutes :  10.971775399986655
```

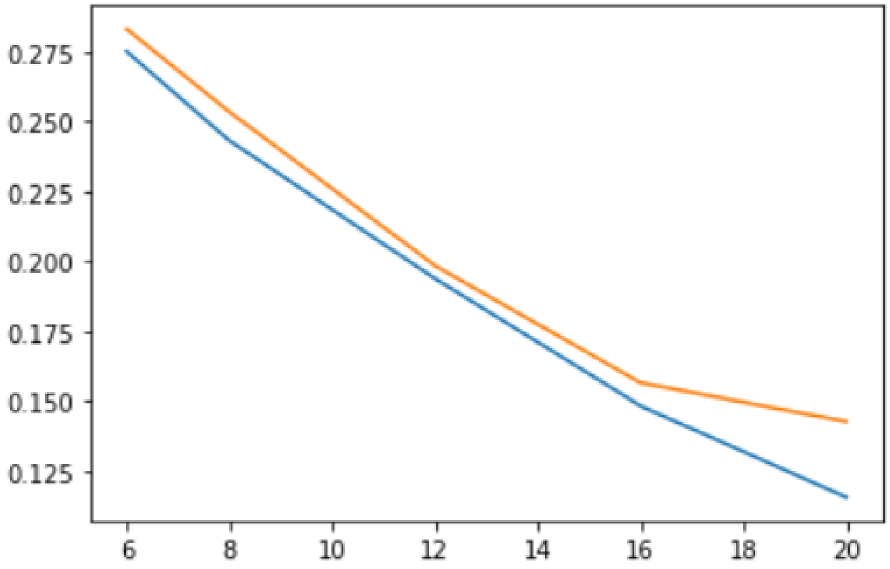
QUESTION 5.c

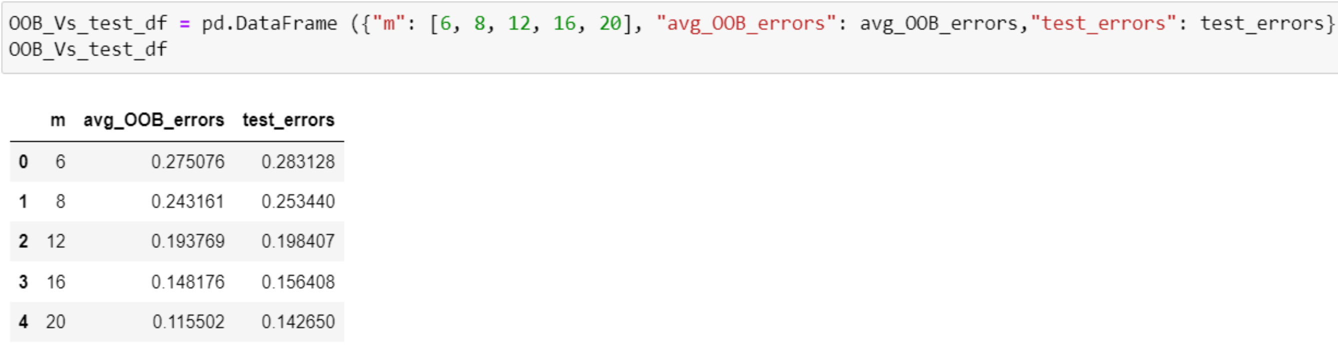
OOB Errors with m= [6, 8, 12, 16, 20]

Please check **Q5_ML_Assignment_RF_CS21MDS14025_SELECTED** Ipython notebook



```
import matplotlib.pyplot as plt
plt.plot([6, 8, 12, 16, 20], avg_OOB_errors)
plt.plot([6, 8, 12, 16, 20], test_errors)
plt.show()
```





```
OOB_Vs_test_df = pd.DataFrame ({"m": [6, 8, 12, 16, 20], "avg_OOB_errors": avg_OOB_errors,"test_errors": test_errors}
OOB_Vs_test_df
```

	m	avg_OOB_errors	test_errors
0	6	0.275076	0.283128
1	8	0.243161	0.253440
2	12	0.193769	0.198407
3	16	0.148176	0.156408
4	20	0.115502	0.142650