Feature Selection

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.

F - score

The F-score, also called the F1-score, is a measure of a model's accuracy on a dataset. It is used to evaluate binary classification systems, which classify examples into 'positive' or 'negative'.

The F-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model's precision and recall.

$$\begin{aligned} F_1 &= \frac{2}{\frac{1}{\mathrm{recall}} \times \frac{1}{\mathrm{precision}}} = 2 \times \frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \\ &= \frac{\mathrm{tp}}{\mathrm{tp} + \frac{1}{2}(\mathrm{fp} + \mathrm{fn})} \end{aligned}$$

The chi2 test returns 2 values : F-score and p - value. Based on the F-score for each feature, we will check the accuracy while considering different number of features for training at a time. Features with high F-score value are of importance.

Importing the required libraries

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import math
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


from sklearn.model_selection import train_test_split
def split(df,label):
    X_tr, X_te, Y_tr, Y_te = train_test_split(df, label, test_size=0.25, random_state=4
2)
    return X_tr, X_te, Y_tr, Y_te


from sklearn.feature_selection import chi2

def feat_select(df,f_score_val,num):
    feat_list = list(f_score_val["Feature"][:num])
    return df[feat_list]


from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold, cross_val_score

classifiers = ['LinearSVM', 'RadialSVM',
               'Logistic',  'RandomForest',
               'AdaBoost',  'DecisionTree',
               'KNeighbors','GradientBoosting']

models = [svm.SVC(kernel='linear'),
          svm.SVC(kernel='rbf'),
          LogisticRegression(max_iter = 1000),
          RandomForestClassifier(n_estimators=200, random_state=0),
          AdaBoostClassifier(random_state = 0),
          DecisionTreeClassifier(random_state=0),
          KNeighborsClassifier(),
          GradientBoostingClassifier(random_state=0)]


def f_score(df,label):
    chi_values=chi2(df,label)
    score = list(chi_values[0])
    feat = df.columns.tolist()
    fscore_df = pd.DataFrame({"Feature":feat, "Score":score})
    fscore_df.sort_values(by="Score", ascending=False,inplace = True)
    fscore_df.reset_index(drop=True, inplace=True)
    return fscore_df


def acc_score(df,label):
    Score = pd.DataFrame({"Classifier":classifiers})
    j = 0
    acc = []
    X_train,X_test,Y_train,Y_test = split(df,label)
    for i in models:
        model = i
        model.fit(X_train,Y_train)
        predictions = model.predict(X_test)
        acc.append(accuracy_score(Y_test,predictions))
        j = j+1
    Score["Accuracy"] = acc
    Score.sort_values(by="Accuracy", ascending=False,inplace = True)
    Score.reset_index(drop=True, inplace=True)
    return Score


def acc_score_num(df,label,f_score_val,feat_list):
    Score = pd.DataFrame({"Classifier":classifiers})
    df2 = None
    for k in range(len(feat_list)):
        df2 = feat_select(df,f_score_val,feat_list[k])
        X_train,X_test,Y_train,Y_test = split(df2,label)
        j = 0
        acc = []
        for i in models:
            model = i
            model.fit(X_train,Y_train)
            predictions = model.predict(X_test)
            acc_val = accuracy_score(Y_test,predictions)
            acc.append(acc_val)
            j = j+1
        feat = str(feat_list[k])
        Score[feat] = acc
    return Score


def plot2(df,l1,l2,p1,p2,c = "b"):
    feat = []
    feat = df.columns.tolist()
    feat = feat[1:]
    plt.figure(figsize = (16, 18))
    for j in range(0,df.shape[0]):
        value = []
        k = 0
        for i in range(1,len(df.columns.tolist())):
            value.append(df.iloc[j][i])
        plt.subplot(4, 4,j+1)
        ax = sns.pointplot(x=feat, y=value,color = c ,markers=["."])
        plt.text(p1,p2,df.iloc[j][0])
        plt.xticks(rotation=90)
        ax.set(ylim=(l1,l2))
        k = k+1


def highlight_max(data, color='aquamarine'):
    attr = 'background-color: {}'.format(color)
    if data.ndim == 1:
        is_max = data == data.max()
        return [attr if v else '' for v in is_max]
    else:
        is_max = data == data.max().max()
        return pd.DataFrame(np.where(is_max, attr, ''),
                             index=data.index, columns=data.columns)
```

Function Description

1. split():

Splits the dataset into training and test set.

2. feat_select():

Returns the dataframe with first 'n' features.

3. f_score():

Returns the dataframe with the F-score for each feature.

4. acc_score():

Returns accuracy for all the classifiers.

5. acc_score_num():

Returns accuracy for all the classifiers for the specified number of features.

6. plot2():

For plotting the results.

The following 3 datasets are used:

1. Breast Cancer
2. Parkinson's Disease
3. PCOS

Plan of action:

- Looking at dataset (includes a little preprocessing)
- F-score (Displaying F-score for each feature)
- Checking Accuracy (comparing accuracies for different number of features)
- Visualization (Plotting the graphs)

Breast Cancer

1. Looking at dataset

In [2]:

```
data_bc = pd.read_csv("C:/Leina/Data_sets/Breast_cancer/data.csv")
label_bc = data_bc["diagnosis"]
label_bc = np.where(label_bc == 'M',1,0)
data_bc.drop(["id","diagnosis","Unnamed: 32"],axis = 1,inplace = True)

print("Breast Cancer dataset:\n",data_bc.shape[0],"Records\n",data_bc.shape[1],"Feature
s")
```

```
Breast Cancer dataset:
 569 Records
 30 Features
```

In [3]:

```
display(data_bc.head())
print("All the features in this dataset have continuous values")
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

5 rows × 30 columns

```
All the features in this dataset have continuous values
```

2. F-score

In [4]:

```
f_score_bc = f_score(data_bc,label_bc)
f_score_bc
```

Out[4]:

	Feature	Score
0	area_worst	112598.431564
1	area_mean	53991.655924
2	area_se	8758.504705
3	perimeter_worst	3665.035416
4	perimeter_mean	2011.102864
5	radius_worst	491.689157
6	radius_mean	266.104917
7	perimeter_se	250.571896
8	texture_worst	174.449400
9	texture_mean	93.897508
10	concavity_worst	39.516915
11	radius_se	34.675247
12	concavity_mean	19.712354
13	compactness_worst	19.314922
14	concave points_worst	13.485419
15	concave points_mean	10.544035
16	compactness_mean	5.403075
17	symmetry_worst	1.298861
18	concavity_se	1.044718
19	compactness_se	0.613785
20	smoothness_worst	0.397366
21	concave points_se	0.305232
22	symmetry_mean	0.257380
23	fractal_dimension_worst	0.231522
24	smoothness_mean	0.149899
25	texture_se	0.009794
26	fractal_dimension_se	0.006371
27	smoothness_se	0.003266
28	symmetry_se	0.000080
29	fractal_dimension_mean	0.000074

3. Checking Accuracy

In [5]:

```
score1 = acc_score(data_bc,label_bc)
score1
```

Out[5]:

	Classifier	Accuracy
0	RandomForest	0.972028
1	KNeighbors	0.965035
2	LinearSVM	0.958042
3	Logistic	0.958042
4	GradientBoosting	0.958042
5	RadialSVM	0.951049
6	AdaBoost	0.951049
7	DecisionTree	0.930070

In [6]:

```
num_feat1 = list(range(8,26))
classifiers = score1["Classifier"].tolist()
score_bc = acc_score_num(data_bc,label_bc,f_score_bc,num_feat1)
score_bc.style.apply(highlight_max, subset = score_bc.columns[1:], axis=None)
```

Out[6]:

	Classifier	8	9	10	11	12	13	14	
0	RandomForest	0.979021	0.979021	0.979021	0.979021	0.972028	0.979021	0.965035	0.
1	KNeighbors	0.944056	0.944056	0.944056	0.944056	0.951049	0.951049	0.951049	0.
2	LinearSVM	0.972028	0.972028	0.965035	0.972028	0.972028	0.965035	0.965035	0.
3	Logistic	0.965035	0.965035	0.965035	0.979021	0.979021	0.979021	0.979021	0.
4	GradientBoosting	0.944056	0.951049	0.958042	0.965035	0.958042	0.965035	0.965035	0.
5	RadialSVM	0.909091	0.944056	0.944056	0.951049	0.930070	0.951049	0.944056	0.
6	AdaBoost	0.951049	0.965035	0.965035	0.965035	0.965035	0.965035	0.965035	0.
7	DecisionTree	0.944056	0.965035	0.958042	0.979021	0.965035	0.972028	0.972028	0.

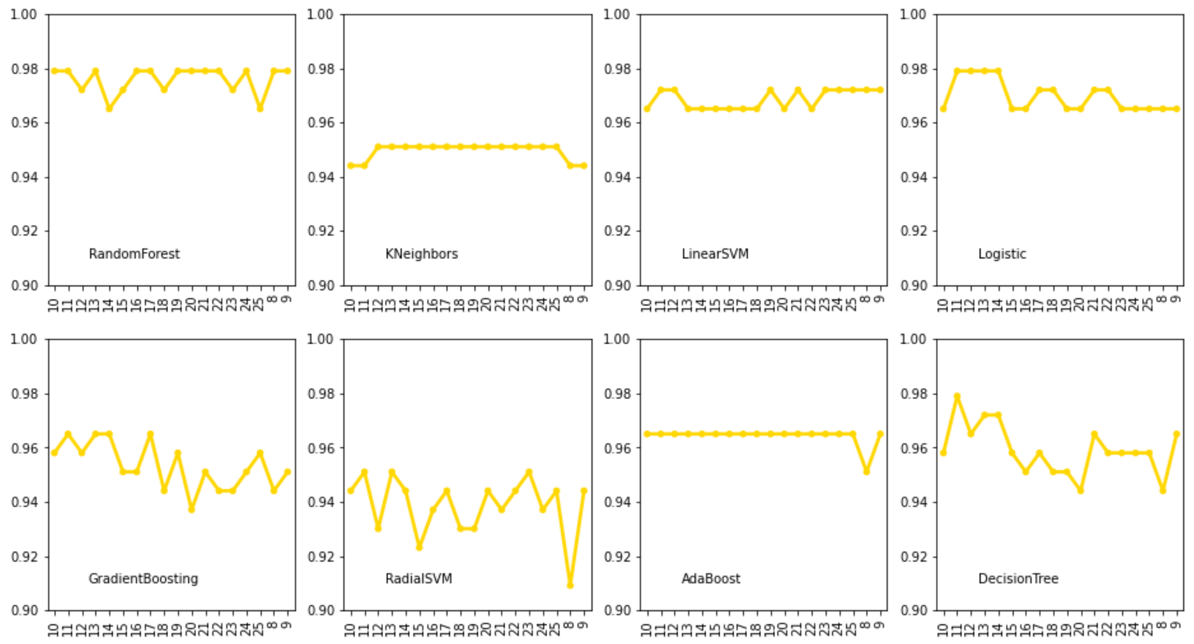
Best Accuracy with all features : RandomForest Classifier - 0.972

Best Accuracy for multiple classifiers for different number of features - 0.979

4. Visualization

In [7]:

```
plot2(score_bc,0.90,1,2.5,0.91,c = "gold")
```



Parkinson's disease

1. Looking at dataset

In [8]:

```
data_pd = pd.read_csv("C:/Leina/Data_sets/Breast_cancer/Parkinsson disease.csv")
label_pd = data_pd["status"]
data_pd.drop(["status","name"],axis = 1,inplace = True)
#Dropping columns with negative value as it does not work for chi2 test
for i in data_pd.columns:
    neg = data_pd[i]<0
    nsum = neg.sum()
    if nsum > 0:
        data_pd.drop([i],axis = 1,inplace = True)

print("Parkinson's disease dataset:\n",data_pd.shape[0],"Records\n",data_pd.shape[1],"F
eatures")
```

```
Parkinson's disease dataset:
 195 Records
 21 Features
```

In [9]:

```
display(data_pd.head())
print("All the features in this dataset have continuous values")
```

	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP
0	119.992	157.302	74.997	0.00784	0.00007	0.00370
1	122.400	148.650	113.819	0.00968	0.00008	0.00465
2	116.682	131.111	111.555	0.01050	0.00009	0.00544
3	116.676	137.871	111.366	0.00997	0.00009	0.00502
4	116.014	141.781	110.655	0.01284	0.00011	0.00655

5 rows × 21 columns

```
All the features in this dataset have continuous values
```

2. F-score

In [10]:

```
f_score_pd = f_score(data_pd,label_pd)
f_score_pd
```

Out[10]:

	Feature	Score
0	MDVP:Flo(Hz)	456.626628
1	MDVP:Fo(Hz)	316.985398
2	MDVP:Fhi(Hz)	227.402656
3	HNR	22.691579
4	MDVP:Shimmer(dB)	3.210348
5	PPE	2.151107
6	D2	1.381600
7	spread2	1.232614
8	Shimmer:DDA	0.462793
9	NHR	0.457699
10	RPDE	0.400299
11	MDVP:Shimmer	0.313475
12	MDVP:APQ	0.307076
13	Shimmer:APQ5	0.193435
14	Shimmer:APQ3	0.154276
15	Jitter:DDP	0.110222
16	MDVP:Jitter(%)	0.056742
17	DFA	0.044425
18	MDVP:RAP	0.036749
19	MDVP:PPQ	0.035713
20	MDVP:Jitter(Abs)	0.000614

3. Checking Accuracy

In [11]:

```
score3 = acc_score(data_pd,label_pd)
score3
```

Out[11]:

	Classifier	Accuracy
0	Logistic	0.918367
1	DecisionTree	0.897959
2	RandomForest	0.877551
3	LinearSVM	0.877551
4	GradientBoosting	0.877551
5	RadialSVM	0.877551
6	KNeighbors	0.836735
7	AdaBoost	0.836735

In [12]:

```
num_feat3 = list(range(7,21))
classifiers = score3["Classifier"].tolist()
score_pd = acc_score_num(data_pd,label_pd,f_score_pd,num_feat3)
score_pd.style.apply(highlight_max, subset = score_pd.columns[1:], axis=None)
```

Out[12]:

	Classifier	7	8	9	10	11	12	13	
0	Logistic	0.877551	0.877551	0.877551	0.877551	0.877551	0.877551	0.877551	0.
1	DecisionTree	0.836735	0.836735	0.836735	0.836735	0.836735	0.836735	0.836735	0.
2	RandomForest	0.877551	0.877551	0.877551	0.877551	0.877551	0.877551	0.877551	0.
3	LinearSVM	0.877551	0.918367	0.918367	0.918367	0.918367	0.918367	0.897959	0.
4	GradientBoosting	0.836735	0.836735	0.836735	0.857143	0.857143	0.877551	0.857143	0.
5	RadialSVM	0.877551	0.877551	0.897959	0.918367	0.897959	0.897959	0.918367	0.
6	KNeighbors	0.836735	0.836735	0.836735	0.836735	0.836735	0.836735	0.836735	0.
7	AdaBoost	0.877551	0.877551	0.918367	0.918367	0.918367	0.897959	0.918367	0.

Best Accuracy with all features : RandomForest Classifier - 0.918

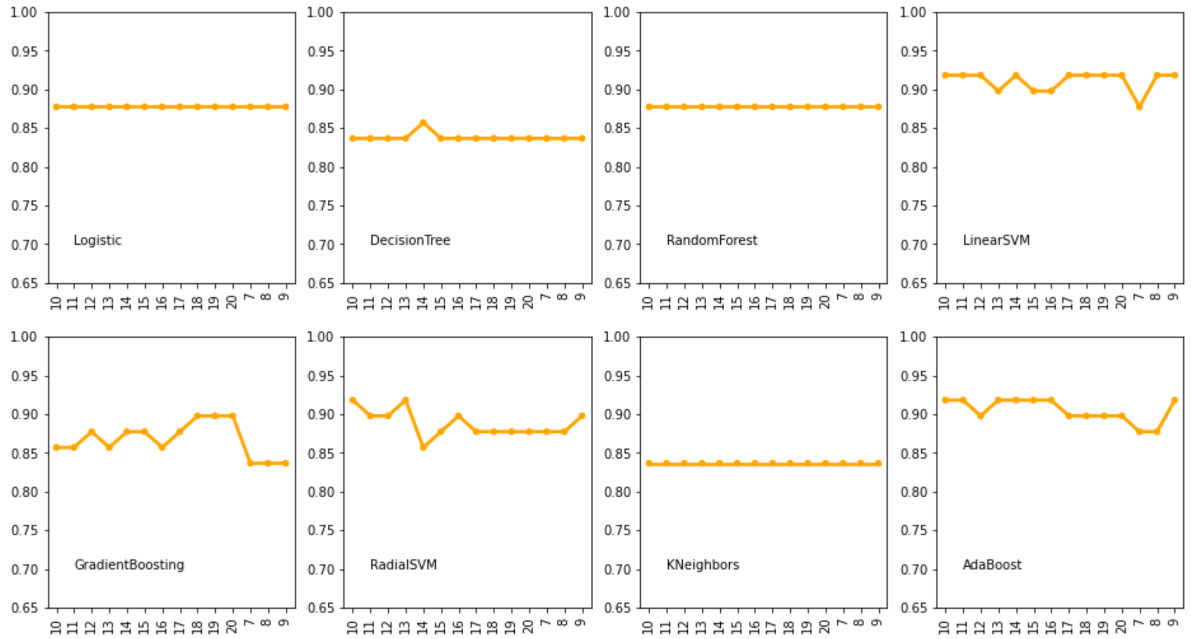
Best Accuracy for multiple classifiers for different number of features - 0.918

Here we see no improvement.

4. Visualization

In [13]:

```
plot2(score_pd,0.65,1.0,1,0.7,c = "orange")
```



PCOS

1. Looking at dataset

In [14]:

```
data_pcos = pd.read_csv("C:/Leina/Data_sets/Breast_cancer/PCOS_data.csv")
label_pcos = data_pcos["PCOS (Y/N)"]
data_pcos.drop(["Sl. No","Patient File No.","PCOS (Y/N)","Unnamed: 44","II    beta-HCG
(mIU/mL)","AMH(ng/mL)"],axis = 1,inplace = True)
data_pcos["Marraige Status (Yrs)"].fillna(data_pcos['Marraige Status (Yrs)'].describe()
.loc[['50%']][0], inplace = True)
data_pcos["Fast food (Y/N)"].fillna(1, inplace = True)

print("PCOS dataset:\n",data_pcos.shape[0],"Records\n",data_pcos.shape[1],"Features")
```

```
PCOS dataset:
 541 Records
 39 Features
```

In [15]:

```
display(data_pcos.head())
print("The features in this dataset have both discrete and continuous values")
```

	Age (yrs)	Weight (Kg)	Height(Cm)	BMI	Blood Group	Pulse rate(bpm)	RR (breaths/min)	Hb(g/dl)	Cycle(R/I)	leng
0	28	44.6	152.0	19.3	15	78	22	10.48	2	
1	36	65.0	161.5	24.9	15	74	20	11.70	2	
2	33	68.8	165.0	25.3	11	72	18	11.80	2	
3	37	65.0	148.0	29.7	13	72	20	12.00	2	
4	25	52.0	161.0	20.1	11	72	18	10.00	2	

5 rows × 39 columns

```
The features in this dataset have both discrete and continuous values
```

2. F-score

In [16]:

```
f_score_pcos = f_score(data_pcos,label_pcos)
f_score_pcos
```

Out[16]:

	Feature	Score
0	Vit D3 (ng/mL)	9477.648952
1	I beta-HCG(mIU/mL)	6950.525631
2	LH(mIU/mL)	2558.471157
3	FSH(mIU/mL)	1601.145511
4	Follicle No. (R)	672.789402
5	Follicle No. (L)	573.647927
6	FSH/LH	96.831682
7	Skin darkening (Y/N)	84.870716
8	hair growth(Y/N)	84.854623
9	Weight gain(Y/N)	65.554147
10	Weight (Kg)	49.466423
11	Fast food (Y/N)	37.076527
12	Cycle(R/I)	27.681419
13	PRG(ng/mL)	24.638020
14	Pimples(Y/N)	22.587803
15	Marraige Status (Yrs)	20.702561
16	BMI	14.568227
17	Age (yrs)	14.284370
18	Hair loss(Y/N)	8.846546
19	Avg. F size (L) (mm)	8.090830
20	Cycle length(days)	7.750342
21	Hip(inch)	5.894434
22	Waist(inch)	5.593108
23	RBS(mg/dl)	4.459089
24	Avg. F size (R) (mm)	3.673590
25	Endometrium (mm)	3.397687
26	No. of abortions	2.934677
27	Reg.Exercise(Y/N)	1.737463
28	Pulse rate(bpm)	1.219952
29	Height(Cm)	0.585219
30	BP _Diastolic (mmHg)	0.315466
31	Hb(g/dl)	0.276315
32	TSH (mIU/L)	0.262652

	Feature	Score
33	Pregnant(Y/N)	0.254536
34	Blood Group	0.175974
35	PRL(ng/mL)	0.131618
36	RR (breaths/min)	0.109112
37	BP _Systolic (mmHg)	0.016198
38	Waist:Hip Ratio	0.000190

3. Checking Accuracy

In [17]:

```
score4 = acc_score(data_pcos,label_pcos)
score4
```

Out[17]:

	Classifier	Accuracy
0	LinearSVM	0.889706
1	GradientBoosting	0.860294
2	AdaBoost	0.860294
3	Logistic	0.852941
4	RandomForest	0.852941
5	RadialSVM	0.838235
6	DecisionTree	0.698529
7	KNeighbors	0.676471

Best Accuracy with all features : RandomForest Classifier - 0.889

Best Accuracy for first (12,20,25) features : DecisionTree Classifier - 0.904

Here we can see an improvement of ~1.5%.

In [18]:

```
num_feat4 = list(range(12,28))
classifiers = score4["Classifier"].tolist()
score_pcos = acc_score_num(data_pcos,label_pcos,f_score_pcos,num_feat4)
score_pcos.style.apply(highlight_max, subset = score_pcos.columns[1:], axis=None)
```

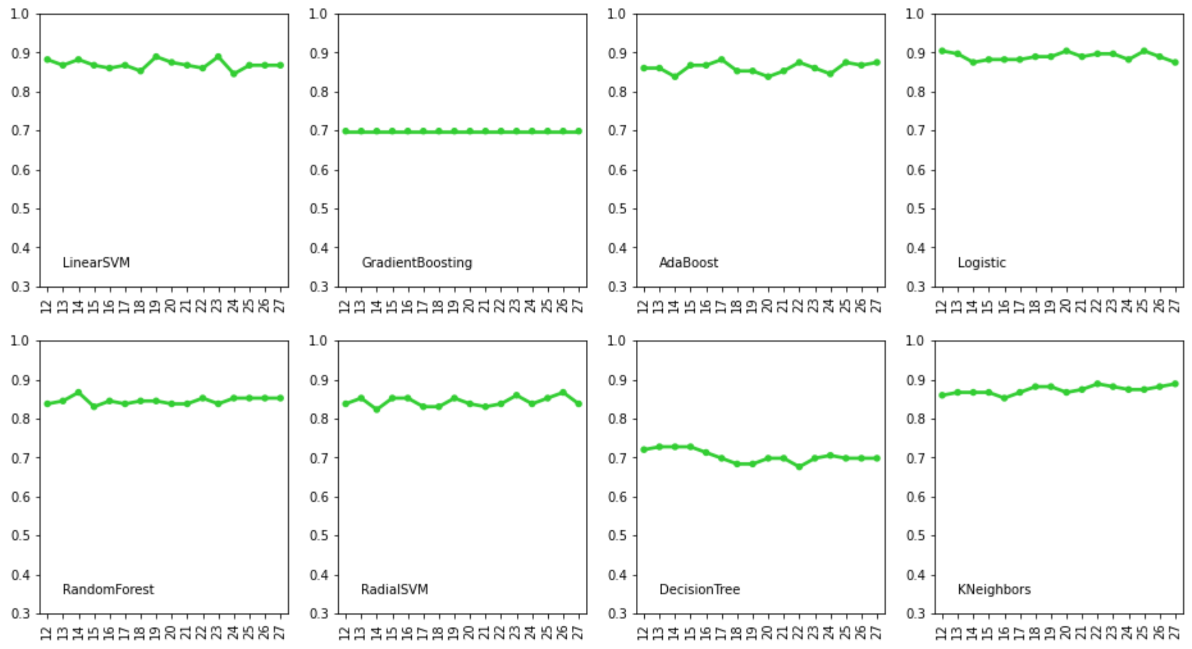
Out[18]:

	Classifier	12	13	14	15	16	17	18	
0	LinearSVM	0.882353	0.867647	0.882353	0.867647	0.860294	0.867647	0.852941	0.
1	GradientBoosting	0.698529	0.698529	0.698529	0.698529	0.698529	0.698529	0.698529	0.
2	AdaBoost	0.860294	0.860294	0.838235	0.867647	0.867647	0.882353	0.852941	0.
3	Logistic	0.904412	0.897059	0.875000	0.882353	0.882353	0.882353	0.889706	0.
4	RandomForest	0.838235	0.845588	0.867647	0.830882	0.845588	0.838235	0.845588	0.
5	RadialSVM	0.838235	0.852941	0.823529	0.852941	0.852941	0.830882	0.830882	0.
6	DecisionTree	0.720588	0.727941	0.727941	0.727941	0.713235	0.698529	0.683824	0.
7	KNeighbors	0.860294	0.867647	0.867647	0.867647	0.852941	0.867647	0.882353	0.

4. Visualization

In [19]:

```
plot2(score_pcos,0.3,1.0,1,0.35,c = "limegreen")
```


LinearSVM
GradientBoosting
AdaBoost
Logistic
RandomForest
RadialSVM
DecisionTree
KNeighbors