In [2]:

```
from math import sqrt
from numpy import hstack
from numpy import vstack
from numpy import asarray
import pandas as pd
from sklearn import metrics
from pandas import DataFrame
import numpy as np
from numpy import mean
from numpy import std
import mlens
from mlens.metrics import make_scorer
import dataframe_image as dfi
from bokeh.io import export_png, export_svgs
from bokeh.models import ColumnDataSource, DataTable, TableColumn
from pandas.plotting import table
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from mlens.ensemble import SequentialEnsemble
from mlens.ensemble import BlendEnsemble
from sklearn.datasets import make_regression
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split


from mlens.ensemble import SuperLearner
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score


import seaborn as sns; sns.set_theme(color_codes=True)
import matplotlib.pyplot as plt


import sklearn
from matplotlib import pyplot
import warnings
warnings.filterwarnings('ignore')
from mlens.ensemble import SuperLearner
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import AdaBoostClassifier


from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
```

```
[MLENS] backend: threading
```

In [2]:

```
# all Data 5 input and 1 output
data = pd.DataFrame(pd.read_csv(r'F:\\IT Master\\Protocol\\Dataset\\Final_DataSet.csv'))
data.describe()

# 4 input and 1 output
#Model_1 = pd.DataFrame(pd.read_csv(r'F:\\IT Master\\Protocol\\Dataset\\Model_1.csv'))
#Model_1.describe()

# 4 input and 1 output
#Model_2 = pd.DataFrame(pd.read_csv(r'F:\\IT Master\\Protocol\\Dataset\\Model_2.csv'))
#Model_2.describe()

# 3 input and 1 output
#Model_3 = pd.DataFrame(pd.read_csv(r'F:\\IT Master\\Protocol\\Dataset\\Model_3.csv'))
#Model_3.describe()
```

Out[2]:

| | Total IT expenditure | Fixed Assets | deposit | percentage of perform loans | profit accued from investing in securities | DEA_Efficiency |
|---|---|---|---|---|---|---|
| **count** | 1.100000e+02 | 1.100000e+02 | 1.100000e+02 | 110.000000 | 1.100000e+02 | 110.000000 |
| **mean** | 1.264739e+09 | 4.195205e+08 | 5.084247e+08 | 60.889273 | 6.263769e+08 | 0.727273 |
| **std** | 3.352885e+09 | 1.858692e+09 | 1.498717e+09 | 22.554861 | 2.149072e+09 | 0.447400 |
| **min** | 1.099000e+03 | 5.697000e+03 | 2.011000e+03 | 16.000000 | 1.615200e+04 | 0.000000 |
| **25%** | 2.016740e+06 | 1.799975e+05 | 3.800550e+05 | 44.427500 | 3.126575e+05 | 0.000000 |
| **50%** | 5.634808e+06 | 5.628785e+05 | 4.163952e+06 | 66.540000 | 3.606246e+06 | 1.000000 |
| **75%** | 2.416401e+08 | 2.562696e+07 | 1.366982e+08 | 78.000000 | 3.597403e+07 | 1.000000 |
| **max** | 1.852275e+10 | 1.514369e+10 | 1.088096e+10 | 95.400000 | 1.144372e+10 | 1.000000 |

In [3]:

```
X = data.drop(['DEA_Efficiency'], axis=1, inplace=False)
print('X Data is \n' , X.head())
print('X shape is ' , X.shape)

#y Data
y = data['DEA_Efficiency']
print('y Data is \n' , y.head())
print('y shape is ' , y.shape)


X, X_val, y, y_val = train_test_split(X, y, test_size= 0.2, random_state=43, shuffle =True)
print('Train', X.shape, y.shape, 'Test', X_val.shape, y_val.shape)
```

```
X Data is
    Total IT expenditure  Fixed Assets  deposit  percentage of perform loans  \
0                 866056         78246   160927                        77.82
1               10159088         84418   149217                        80.16
2               17776187         79162    83921                        85.63
3               46518892         49750   137766                        78.61
4               45319766         31409    64779                        69.53

   profit accued from investing in securities
0                                    43762724
1                                      922832
2                                      450697
3                                      402067
4                                      496045
X shape is  (110, 5)
y Data is
 0    1
1    0
2    0
3    0
4    0
Name: DEA_Efficiency, dtype: int64
y shape is  (110,)
Train (88, 5) (88,) Test (22, 5) (22,)
```

In [4]:

```
def cross_val(model):
    pred = cross_val_score(model, X, y, cv=10)
    return pred.mean()

def print_evaluate(y_val, y_pred):
    acc = metrics.accuracy_score(y_val, y_pred)
    ps = metrics.precision_score(y_val, y_pred)
    rs = metrics.recall_score(y_val, y_pred)
    f1 = metrics.f1_score(y_val, y_pred)
    #cm = confusion_matrix(y_val, y_pred)
    print('acc:', acc)
    print('ps:', ps)
    print('rs:', rs)
    print('f1:', f1)
    #print('cm', cm)
    print('_____________________________________')

def evaluate(y_val, y_pred):
    acc = metrics.accuracy_score(y_val, y_pred)
    ps = metrics.precision_score(y_val, y_pred)
    rs = metrics.recall_score(y_val, y_pred)
    f1 = metrics.f1_score(y_val, y_pred)
    #cm = confusion_matrix(y_val, y_pred)
```

```
    return acc, ps, rs, f1
#cm
```

In [9]:

```
cv = KFold(n_splits=10,random_state=43,shuffle=True)
clf = KNeighborsClassifier(n_neighbors= 4 , weights ='uniform', algorithm='auto')
scores = cross_val_score(clf,X, y,scoring='f1_macro',cv=cv)
clf.fit(X, y)
y_pred1 = clf.predict(X_val)

print('Test set evaluation:\n_____________________________________')
print_evaluate(y_val, y_pred1)
results_df = pd.DataFrame(data=[["KNN", *evaluate(y_val, y_pred1) ,
                                 cross_val(KNeighborsClassifier(n_neighbors= 5,weights ='distance', algorithm
                          columns=['Model', 'acc', 'ps', 'rs', 'f1', "Cross Validation"])
round(results_df,4)
```

```
Test set evaluation:
_____________________________________
acc: 0.5909090909090909
ps: 0.8333333333333334
rs: 0.5882352941176471
f1: 0.6896551724137931
_____________________________________
```

Out[9]:

| | Model | acc | ps | rs | f1 | Cross Validation |
|---|---|---|---|---|---|---|
| **0** | KNN | 0.5909 | 0.8333 | 0.5882 | 0.6897 | 0.7722 |

In [10]:

```
cv = KFold(n_splits=10,random_state=43,shuffle=True)
clf1 = RandomForestClassifier(n_estimators= 50,criterion= 'gini')
scores = cross_val_score(clf1,X, y,scoring='f1_macro',cv=cv)
clf1.fit(X, y)
y_pred2 = clf1.predict(X_val)


print('Test set evaluation:\n_____________________________________')
print_evaluate(y_val, y_pred2)
results_df_2 = pd.DataFrame(data=[["random", *evaluate(y_val, y_pred2) ,
                                   cross_val(RandomForestClassifier(n_estimators=10,criterion= 'gini', random
                           columns=['Model', 'acc', 'ps', 'rs', 'f1', "Cross Validation"])
results_df = results_df.append(results_df_2, ignore_index=True)
round(results_df,4)
```

```
Test set evaluation:
_____________________________________
acc: 0.8636363636363636
ps: 0.8888888888888888
rs: 0.9411764705882353
f1: 0.9142857142857143
_____________________________________
```

Out[10]:

| | Model | acc | ps | rs | f1 | Cross Validation |
|---|---|---|---|---|---|---|
| **0** | KNN | 0.5909 | 0.8333 | 0.5882 | 0.6897 | 0.7722 |
| **1** | random | 0.8636 | 0.8889 | 0.9412 | 0.9143 | 0.7597 |

In [11]:

```
cv = KFold(n_splits=10,random_state=43,shuffle=True)
clf2 = svm.SVC(C= 1.0, kernel='poly', degree= 3,  gamma='scale',random_state=43)
scores = cross_val_score(clf2,X, y,scoring='f1_macro',cv=cv)
clf2.fit(X, y)
y_pred3 = clf2.predict(X_val)


print('Test set evaluation:\n_____________________________________')
print_evaluate(y_val, y_pred3)
results_df_2 = pd.DataFrame(data=[["svm", *evaluate(y_val, y_pred3) ,
                                   cross_val(svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto',random_state
                           columns=['Model', 'acc', 'ps', 'rs', 'f1', "Cross Validation"])
results_df = results_df.append(results_df_2, ignore_index=True)
round(results_df,4)
```

```
Test set evaluation:
_____________________________________
acc: 0.7272727272727273
ps: 0.7619047619047619
rs: 0.9411764705882353
f1: 0.8421052631578947
_____________________________________
```

Out[11]:

| | Model | acc | ps | rs | f1 | Cross Validation |
|---|---|---|---|---|---|---|
| **0** | KNN | 0.5909 | 0.8333 | 0.5882 | 0.6897 | 0.7722 |
| **1** | random | 0.8636 | 0.8889 | 0.9412 | 0.9143 | 0.7597 |
| **2** | svm | 0.7273 | 0.7619 | 0.9412 | 0.8421 | 0.7167 |

In [12]:

```
cv = KFold(n_splits=10,random_state=43,shuffle=True)
clf3 = AdaBoostClassifier(n_estimators= 10, learning_rate = 0.5,random_state=43)
scores = cross_val_score(clf3,X, y,scoring='f1_macro',cv=cv)
clf3.fit(X, y)
y_pred3 = clf3.predict(X_val)


print('Test set evaluation:\n_____________________________________')
print_evaluate(y_val, y_pred3)
results_df_2 = pd.DataFrame(data=[["Ada", *evaluate(y_val, y_pred3) ,
                                   cross_val(AdaBoostClassifier(n_estimators=50, learning_rate =0.5, random_s
                            columns=['Model', 'acc', 'ps', 'rs', 'f1', "Cross Validation"])
results_df = results_df.append(results_df_2, ignore_index=True)
round(results_df,4)
```

```
Test set evaluation:
_____________________________________
acc: 0.8181818181818182
ps: 0.8823529411764706
rs: 0.8823529411764706
f1: 0.8823529411764706
_____________________________________
```

Out[12]:

| | Model | acc | ps | rs | f1 | Cross Validation |
|---|---|---|---|---|---|---|
| **0** | KNN | 0.5909 | 0.8333 | 0.5882 | 0.6897 | 0.7722 |
| **1** | random | 0.8636 | 0.8889 | 0.9412 | 0.9143 | 0.7597 |
| **2** | svm | 0.7273 | 0.7619 | 0.9412 | 0.8421 | 0.7167 |
| **3** | Ada | 0.8182 | 0.8824 | 0.8824 | 0.8824 | 0.8278 |

In [13]:

```
ensemble = SuperLearner(scorer=accuracy_score, random_state=43)
ensemble.add([KNeighborsClassifier(n_neighbors= 4 , weights ='uniform', algorithm='auto'),
              RandomForestClassifier(n_estimators= 50  , criterion= 'gini',random_state=43),
              svm.SVC(C= 1.0, kernel='poly', degree= 3,  gamma='scale',random_state=43),
              AdaBoostClassifier(n_estimators= 10,learning_rate = 0.5,random_state=43)])
ensemble.add_meta(LogisticRegression())
ensemble.fit(X, y)
print(ensemble.data)
ensemble_results =pd.DataFrame(ensemble.data)
y_pred4 = ensemble.predict(X_val)
print('Test set evaluation:\n_____________________________________')
print_evaluate(y_val, y_pred4)
results_df_2 = pd.DataFrame(data=[["Super Learner", *evaluate(y_val, y_pred4)]],
                            columns=['Model', 'acc', 'ps', 'rs', 'f1'])
results_df = results_df.append(results_df_2, ignore_index=True)
round(results_df,4)
```

```
                                  score-m  score-s  ft-m  ft-s  pt-m  pt-s
layer-1  adaboostclassifier          0.72     0.08  0.28  0.06  0.01  0.00
layer-1  kneighborsclassifier        0.66     0.05  0.02  0.01  0.04  0.00
layer-1  randomforestclassifier      0.78     0.03  0.53  0.01  0.02  0.00
layer-1  svc                        0.69     0.01  0.01  0.00  0.01  0.00

Test set evaluation:
_____________________________________
acc: 0.9545454545454546
ps: 0.9444444444444444
rs: 1.0
f1: 0.9714285714285714
_____________________________________
```

Out[13]:

| | Model | acc | ps | rs | f1 | Cross Validation |
|---|---|---|---|---|---|---|
| **0** | KNN | 0.5909 | 0.8333 | 0.5882 | 0.6897 | 0.7722 |
| **1** | random | 0.8636 | 0.8889 | 0.9412 | 0.9143 | 0.7597 |
| **2** | svm | 0.7273 | 0.7619 | 0.9412 | 0.8421 | 0.7167 |
| **3** | Ada | 0.8182 | 0.8824 | 0.8824 | 0.8824 | 0.8278 |

| | Model | acc | ps | rs | f1 | Cross Validation |
|---|---|---|---|---|---|---|
| **4** | Super Learner | 0.9545 | 0.9444 | 1.0000 | 0.9714 | NaN |

In [15]:

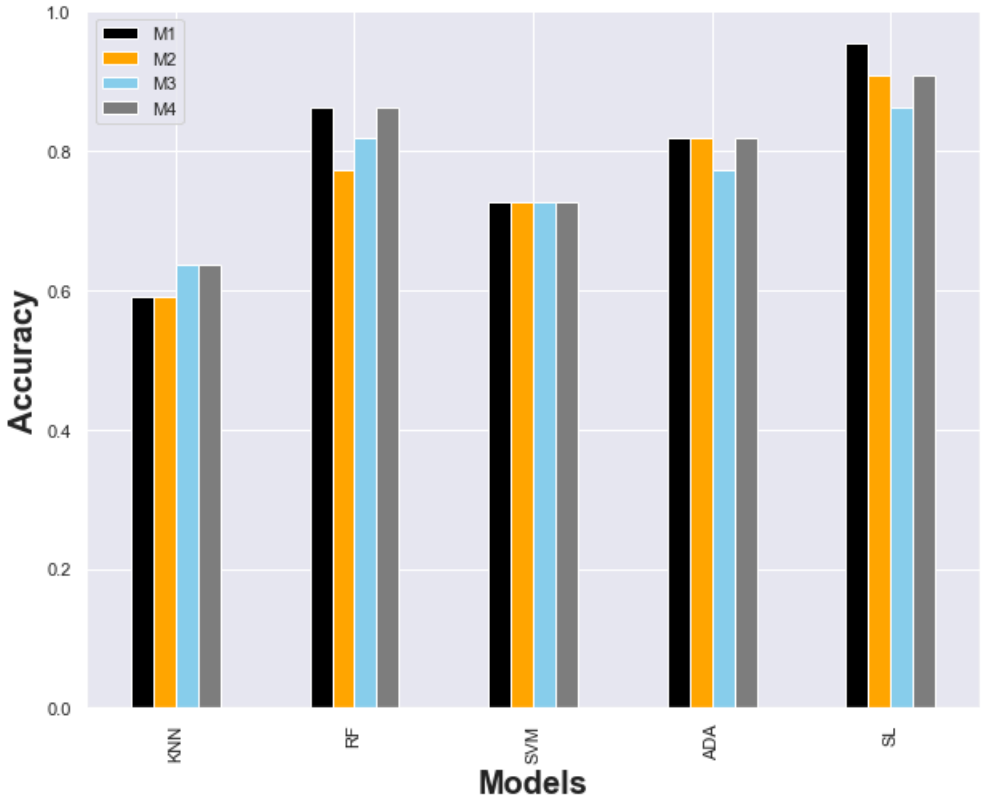
```
data=[["KNN", 0.5909, 0.5909, 0.6364, 0.6364],
      ["RF", 0.8636, 0.7727, 0.8182, 0.8636],
      ["SVM", 0.7273, 0.7273, 0.7273, 0.7273],
      ["ADA", 0.8182, 0.8182, 0.7727, 0.8182],
      ['SL', 0.9545, 0.9091, 0.8636, 0.9091]
     ]

# Plot multiple columns bar chart

df=pd.DataFrame(data,columns=["Models","M1","M2","M3", "M4"])

df.plot(x="Models", y=["M1","M2","M3", "M4"], kind="bar",figsize=(10,8),
        color=['black','orange', 'skyblue','gray'])


plt.xlabel("Models", fontsize=20, fontweight='bold')
plt.ylabel("Accuracy", fontsize=20, fontweight='bold')
plt.legend()
plt.show()
```




# Bagging Classifier

In [11]:

```
from sklearn.ensemble import BaggingClassifier
```

In [12]:

```
# all Data 5 input and 1 output
data = pd.DataFrame(pd.read_csv(r'F:\\IT Master\\Protocol\\Dataset\\Final_DataSet.csv'))
data.describe()

# 4 input and 1 output
#Model_1 = pd.DataFrame(pd.read_csv(r'F:\\IT Master\\Protocol\\Dataset\\Model_1.csv'))
#Model_1.describe()

# 4 input and 1 output
#Model_2 = pd.DataFrame(pd.read_csv(r'F:\\IT Master\\Protocol\\Dataset\\Model_2.csv'))
#Model_2.describe()

# 3 input and 1 output
#Model_3 = pd.DataFrame(pd.read_csv(r'F:\\IT Master\\Protocol\\Dataset\\Model_3.csv'))
#Model_3.describe()
```

```
# Efficiency CRS and VRS
#d = pd.DataFrame(pd.read_csv(r'F:\\IT Master\\Protocol\\Dataset\\Models.csv'))
#d
```

Out[12]:

| | Total IT expenditure | Fixed Assets | deposit | percentage of perform loans | profit accued from investing in securities | DEA_Efficiency |
|---|---|---|---|---|---|---|
| **count** | 1.100000e+02 | 1.100000e+02 | 1.100000e+02 | 110.000000 | 1.100000e+02 | 110.000000 |
| **mean** | 1.264739e+09 | 4.195205e+08 | 5.084247e+08 | 60.889273 | 6.263769e+08 | 0.727273 |
| **std** | 3.352885e+09 | 1.858692e+09 | 1.498717e+09 | 22.554861 | 2.149072e+09 | 0.447400 |
| **min** | 1.099000e+03 | 5.697000e+03 | 2.011000e+03 | 16.000000 | 1.615200e+04 | 0.000000 |
| **25%** | 2.016740e+06 | 1.799975e+05 | 3.800550e+05 | 44.427500 | 3.126575e+05 | 0.000000 |
| **50%** | 5.634808e+06 | 5.628785e+05 | 4.163952e+06 | 66.540000 | 3.606246e+06 | 1.000000 |
| **75%** | 2.416401e+08 | 2.562696e+07 | 1.366982e+08 | 78.000000 | 3.597403e+07 | 1.000000 |
| **max** | 1.852275e+10 | 1.514369e+10 | 1.088096e+10 | 95.400000 | 1.144372e+10 | 1.000000 |

In [13]:

```
X = data.drop(['DEA_Efficiency'], axis=1, inplace=False)
print('X Data is \n' , X.head())
print('X shape is ' , X.shape)

#y Data
y = data['DEA_Efficiency']
print('y Data is \n' , y.head())
print('y shape is ' , y.shape)


X, X_val, y, y_val = train_test_split(X, y, test_size= 0.2, random_state=43, shuffle =True)
print('Train', X.shape, y.shape, 'Test', X_val.shape, y_val.shape)
```

```
X Data is
    Total IT expenditure  Fixed Assets  deposit  percentage of perform loans  \
0                 866056         78246   160927                        77.82
1               10159088         84418   149217                        80.16
2               17776187         79162    83921                        85.63
3               46518892         49750   137766                        78.61
4               45319766         31409    64779                        69.53

   profit accued from investing in securities
0                                    43762724
1                                      922832
2                                      450697
3                                      402067
4                                      496045
X shape is  (110, 5)
y Data is
 0    1
1    0
2    0
3    0
4    0
Name: DEA_Efficiency, dtype: int64
y shape is  (110,)
Train (88, 5) (88,) Test (22, 5) (22,)
```

In [29]:

```
def print_evaluate(y_val, y_pred):
    acc = metrics.accuracy_score(y_val, y_pred)
    ps = metrics.precision_score(y_val, y_pred)
    rs = metrics.recall_score(y_val, y_pred)
    f1 = metrics.f1_score(y_val, y_pred)
    #cm = confusion_matrix(y_val, y_pred)
    print('acc:', acc)
    print('ps:', ps)
    print('rs:', rs)
    print('f1:', f1)
    #print('cm', cm)
    print('_____________________________________')
```

In [30]:

```
clf = BaggingClassifier(base_estimator= DecisionTreeClassifier(),n_estimators=10)
clf.fit(X, y)

y_pred = clf.predict(X_val)
```

```
print_evaluate(y_val, y_pred)
```

```
acc: 0.8636363636363636
ps: 0.8888888888888888
rs: 0.9411764705882353
f1: 0.9142857142857143
_____________________________________
```

In [16]:

```
clf = BaggingClassifier(base_estimator= AdaBoostClassifier(),n_estimators=10)
clf.fit(X, y)

y_pred = clf.predict(X_val)

print_evaluate(y_val, y_pred)
```

```
acc: 0.8636363636363636
ps: 0.8888888888888888
rs: 0.9411764705882353
f1: 0.9142857142857143
_____________________________________
```

In [17]:

```
clf = BaggingClassifier(base_estimator= RandomForestClassifier(),n_estimators=10)
clf.fit(X, y)

y_pred = clf.predict(X_val)

print_evaluate(y_val, y_pred)
```

```
acc: 0.9090909090909091
ps: 0.8947368421052632
rs: 1.0
f1: 0.9444444444444444
_____________________________________
```

In [18]:

```
clf = BaggingClassifier(base_estimator= KNeighborsClassifier(),n_estimators=10)
clf.fit(X, y)

y_pred = clf.predict(X_val)

print_evaluate(y_val, y_pred)
```

```
acc: 0.7727272727272727
ps: 0.8333333333333334
rs: 0.8823529411764706
f1: 0.8571428571428571
_____________________________________
```

In [19]:

```
clf = BaggingClassifier(base_estimator=SVC(),n_estimators=10)
clf.fit(X, y)

y_pred1 = clf.predict(X_val)

print_evaluate(y_val, y_pred1)
```

```
acc: 0.7727272727272727
ps: 0.7727272727272727
rs: 1.0
f1: 0.8717948717948718
_____________________________________
```