3.2 Random Forest:

Random forests are one the most popular machine learning algorithms. They are so successful because they provide in general a good predictive performance, low overfitting, and easy interpretability.

This interpretability is given by the fact that it is straightforward to derive the importance of each variable on the tree decision. In other words, it is easy to compute how much each variable is contributing to the decision.

Feature selection using Random forest comes under the category of Embedded methods. Embedded methods combine the qualities of filter and wrapper methods. They are implemented by algorithms that have their own built-in feature selection methods

How does Random forest select features?

Random forests consist of 4 –12 hundred decision trees, each of them built over a random extraction of the observations from the dataset and a random extraction of the features. Not every tree sees all the features or all the observations, and this guarantees that the trees are de-correlated and therefore less prone to over-fitting. Each tree is also a sequence of yes-no questions based on a single or combination of features. At each node (this is at each question), the tree divides the dataset into 2 buckets, each of them hosting observations that are more similar among themselves and different from the ones in the other bucket. Therefore, the importance of each feature is derived from how “pure” each of the buckets is.

In [4]:

```
import pandas as pd
import numpy as np


from sklearn.model_selection import train_test_split

from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [5]:

```
data = pd.read_csv('paribas.csv',nrows = 20000)
data.shape
```

Out[5]:

```
(20000, 133)
```

In [6]:

```
num = ['int16','int32','int64','float16','float32','float64']
num_cols = list(data.select_dtypes(include = num).columns)
data = data[num_cols]
data.shape
```

Out[6]:

```
(20000, 114)
```

In [7]:

```
correlated_features=set()
rows = set()
correlation_matrix = data.corr()
```

In [8]:

```
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i,j]) > 0.8:
            col_name = correlation_matrix.columns[i]
            row_name = correlation_matrix.index[j]
            rows.add(row_name)
            correlated_features.add(col_name)
```

In [9]:

```
X_train, X_test, y_train, y_test = train_test_split(
    data.drop(labels=['target', 'ID'], axis=1),
    data['target'],
    test_size=0.2,
    random_state=41)
X_train.shape, X_test.shape
```

Out[9]:

```
((16000, 112), (4000, 112))
```

In [10]:

```
X_train.drop(labels = correlated_features , axis =1 , inplace =True)
X_test.drop(labels = correlated_features , axis =1 , inplace =True)
```

In [11]:

```
X_train.shape, X_test.shape
```

Out[11]:

```
((16000, 57), (4000, 57))
```

In [55]:

```
from sklearn.ensemble import RandomForestClassifier as rfc
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import roc_auc_score
```

In [38]:

```
r = rfc(n_estimators = 100)
sfl = SelectFromModel(r)
sfl.fit(X_train.fillna(0) , y_train)
```

Out[38]:

```
SelectFromModel(estimator=RandomForestClassifier(bootstrap=True,
                                                  class_weight=None,
                                                  criterion='gini',
                                                  max_depth=None,
                                                  max_features='auto',
                                                  max_leaf_nodes=None,
                                                  min_impurity_decrease=0.
0,
                                                  min_impurity_split=None,
                                                  min_samples_leaf=1,
                                                  min_samples_split=2,
                                                  min_weight_fraction_leaf=
0.0,
                                                  n_estimators=100, n_jobs=
None,
                                                  oob_score=False,
                                                  random_state=None, verbos
e=0,
                                                  warm_start=False),
                max_features=None, norm_order=1, prefit=False, threshold=N
one)
```

In [53]:

```
sfl.get_support()
features_selected = X_train.columns[(sfl.get_support())]
len(features_selected)
```

Out[53]:

```
4
```

In [56]:

```
clf = rfc(n_estimators = 100 , random_state = 41 , max_depth = 3)
clf.fit(X_train[features_selected].fillna(0) , y_train)

train_pred = clf.predict_proba(X_train[features_selected].fillna(0))
print('accuracy on training data: {}'.format(roc_auc_score(y_train , train_pred[:,1])))

test_pred = clf.predict_proba(X_test[features_selected].fillna(0))
print('accuracy on test data  :{}'.format(roc_auc_score(y_test , test_pred[:,1])))
```

```
accuracy on training data: 0.7035399696485769
accuracy on test data  :0.7112540120793788
```

Links for the inbuilt methods used in the above code:

SelectFromModel() : https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel. (https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.

RandomForestClassifier() : https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.h (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.h