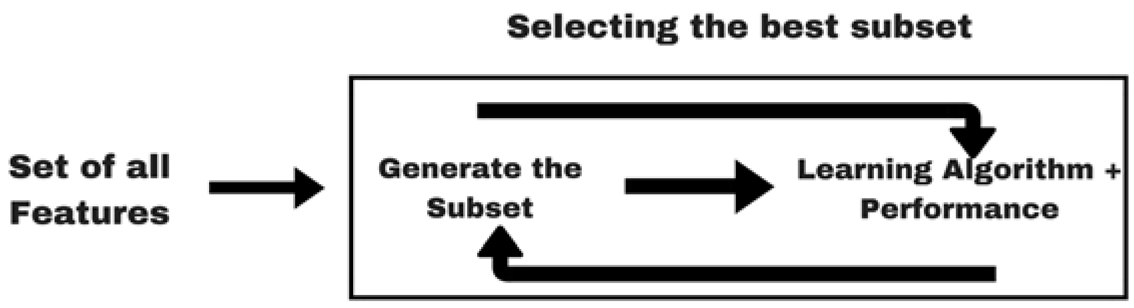
3. Embeded Method

3 .Embeded Method

Embedded methods have been recently proposed that try to combine the advantages of both previous methods. A learning algorithm takes advantage of its own variable selection process and performs feature selection and classification simultaneously.



3.1 Logistic Regression L1

L1 Regularization: L1 regularization technique is called Lasso Regression

Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds “absolute value of magnitude” of coefficient as penalty term to the loss function.

$$L(x,y) \equiv \sum_{i=1}^{n}(y_i - h_\theta(x_i))^2 + \boxed{\lambda \sum_{i=1}^{n}|\theta_i|}$$

Lasso shrinks the less important feature’s coefficient to zero thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features.

In [7]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.model_selection import train_test_split

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

In [9]:

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

Out[9]:

```
(20000, 133)
```

In [10]:

```
data.head()
```

Out[10]:

	ID	target	v1	v2	v3	v4	v5	v6	v7	v8	...	
0	3	1	1.335739	8.727474	C	3.921026	7.915266	2.599278	3.176895	0.012941	...	8
1	4	1	NaN	NaN	C	NaN	9.191265	NaN	NaN	2.301630	...	
2	5	1	0.943877	5.310079	C	4.410969	5.326159	3.979592	3.928571	0.019645	...	9
3	6	1	0.797415	8.304757	C	4.225930	11.627438	2.097700	1.987549	0.171947	...	7
4	8	1	NaN	NaN	C	NaN	NaN	NaN	NaN	NaN	...	

5 rows × 133 columns

In [15]:

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

Out[15]:

```
(20000, 114)
```

In [27]:

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

In [28]:

```
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 [32]:

```
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[32]:

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

In [34]:

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

In [35]:

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

Out[35]:

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

In [37]:

```
sel_ = SelectFromModel(LogisticRegression(penalty='l1'))
sel_.fit(np.array(X_train.fillna(0)), y_train)
```

```
c:\users\dell\appdata\local\programs\python\python36\lib\site-packages\skl
earn\linear_model\logistic.py:432: FutureWarning: Default solver will be c
hanged to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[37]:

```
SelectFromModel(estimator=LogisticRegression(C=1, class_weight=None, dual=
False,
                                             fit_intercept=True,
                                             intercept_scaling=1, l1_ratio
=None,
                                             max_iter=100, multi_class='wa
rn',
                                             n_jobs=None, penalty='l1',
                                             random_state=None, solver='wa
rn',
                                             tol=0.0001, verbose=0,
                                             warm_start=False),
                max_features=None, norm_order=1, prefit=False, threshold=N
one)
```

In [40]:

```
features_selected = X_train.columns[sel_.get_support()]
```

In [44]:

```
print('total features: {}'.format((X_train.shape[1])))
print('selected features: {}'.format(len(features_selected)))
print('features with coefficients shrank to zero: {}'.format(
      np.sum(sel_.estimator_.coef_ == 0)))
```

```
total features: 57
selected features: 54
features with coefficients shrank to zero: 3
```

In [41]:

```
# from boruta import BorutaPy
from sklearn.metrics import roc_auc_score
from sklearn.ensemble import RandomForestClassifier as rfc
```

In [42]:

```
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: 0.7038005963224865
accuracy on test data  :0.7048169111302847
```

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.

LogisticRegression() : https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.htr (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.htr