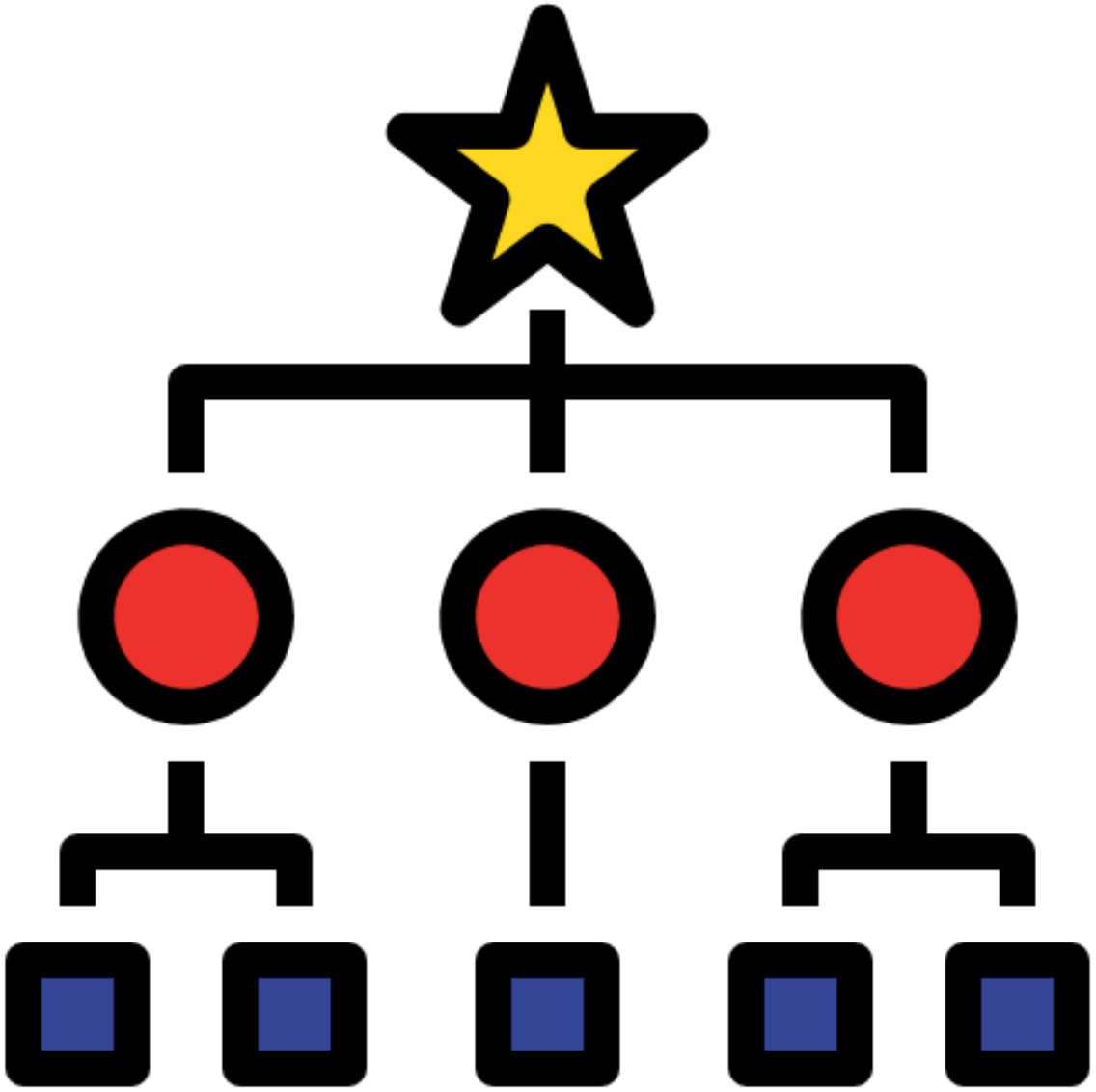
RANDOM FOREST CLASSIFIER

INDEX NUMBER: 19000792



University of Colombo School of Computing

Random Forest Classifier

Decision Trees

The supervised learning algorithms family includes the decision tree algorithm. The decision tree technique, in contrast to other supervised learning methods, is capable of handling both classification and regression issues.By learning straightforward decision rules derived from previous data, a Decision Tree is used to build a training model that may be used to predict the class or value of the target variable (training data).In decision trees, we begin at the tree's root when anticipating a record's class label. We contrast the root attribute's values with that of the attribute on the record. We follow the branch that corresponds to that value and go on to the next node based on the comparison.

Depending on the kind of target variable we have, several decision trees can be used. It comes in two varieties:

- **Categorical Variable Decision Tree**: When a decision tree has a categorical target variable, it is referred to as a Categorical Variable Decision Tree.
- **Continuous Variable Decision Tree**: When a decision tree has a continuous target variable, it is referred to as a Continuous Variable Decision Tree.

Random forests

A random forest is a machine learning method for tackling classification and regression issues. It makes use of ensemble learning, a method for solving complicated issues by combining a number of classifiers.In a random forest algorithm, there are many different decision trees. The random forest algorithm creates a "forest" that is trained via bagging or bootstrap aggregation. The accuracy of machine learning algorithms is increased by bagging, an ensemble meta-algorithm.Based on the predictions of the decision trees, the (random forest) algorithm determines the result. It makes predictions by averaging or averaging out

the results from different trees. The accuracy of the result grows as the number of trees increases.

Bank Marketing Data Set

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

Dataset contains following columns,

Input variables:
bank client data:

1. age (numeric)
2. job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3. marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4. education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5. default: has credit in default? (categorical: 'no','yes','unknown')
6. housing: has housing loan? (categorical: 'no','yes','unknown')
7. loan: has personal loan? (categorical: 'no','yes','unknown')
8. contact: contact communication type (categorical: 'cellular','telephone')
9. month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10. duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

11. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
12. pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
13. previous: number of contacts performed before this campaign and for this client (numeric)
14. poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
15. emp.var.rate: employment variation rate - quarterly indicator (numeric)

Output variable (desired target):

y - has the client subscribed a term deposit? (binary: 'yes','no')

Apply machine Learning algorithm for the data set

The first step is to load the data set to our Jupyter notebook from Pandas library. For that we use read_csv() function.
We can use the head function to receive valuable information about the data set.

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df=pd.read_csv('bank.csv',names=['age','job','marital','education','default','
                    'housing','loan','contact','day','month','duration','camp
df.head()
```

Out[2]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may

Data preprocessing

We can see, there is no null values, in the dataset, by checking null values in the dataset.

```
In [5]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        45211 non-null  int64
 1   job        45211 non-null  int32
 2   marital    45211 non-null  int32
 3   education  45211 non-null  int32
 4   default    45211 non-null  int32
 5   balance    45211 non-null  int64
 6   housing    45211 non-null  int32
 7   loan       45211 non-null  int32
 8   contact    45211 non-null  int32
 9   day        45211 non-null  int64
 10  month      45211 non-null  int32
 11  duration   45211 non-null  int64
 12  campaign   45211 non-null  int64
 13  pdays      45211 non-null  int64
 14  previous   45211 non-null  int64
 15  poutcome   45211 non-null  int32
 16  y          45211 non-null  int32
dtypes: int32(10), int64(7)
memory usage: 4.1 MB
```

Now we convert the data set to Numeric values to fit for Random Forest.

In [3]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
# df['age'].nunique()

df['age'] = le.fit_transform(df['age'])
df['job'] = le.fit_transform(df['job'])
df['marital'] = le.fit_transform(df['marital'])
df['education'] = le.fit_transform(df['education'])
df['default'] = le.fit_transform(df['default'])
df['balance'] = le.fit_transform(df['balance'])
df['housing'] = le.fit_transform(df['housing'])
df['loan'] = le.fit_transform(df['loan'])
df['contact'] = le.fit_transform(df['contact'])
df['day'] = le.fit_transform(df['day'])
df['month'] = le.fit_transform(df['month'])
df['duration'] = le.fit_transform(df['duration'])
df['campaign'] = le.fit_transform(df['campaign'])
df['pdays'] = le.fit_transform(df['pdays'])
df['previous'] = le.fit_transform(df['previous'])
df['poutcome'] = le.fit_transform(df['poutcome'])
df['y'] = le.fit_transform(df['y'])
df.head()
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration
0	40	4	1	2	0	3036	1	0	2	4	8	261
1	26	9	2	1	0	945	1	0	2	4	8	151
2	15	2	1	1	0	918	1	1	2	4	8	76
3	29	1	1	3	0	2420	1	0	2	4	8	92
4	15	11	2	3	0	917	0	0	2	4	8	198

Model fitting

Now we Split the data set to train set and test. Because we need to check your model work or not correctly.

In [9]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

In [10]:
```
X_test.shape
```

Out[10]: (13564, 16)

In [11]:
```
y_test.shape
```

Out[11]: (13564,)

Evaluation

Now we can apply the Random forest classifier. The random forest classifier is Sklearn. Ensemble library, we can import it and predict the test. We can get the score of the training test. The score is the probability of correct results. We can find the accuracy by taking it as a percentage.

```
In [12]: from sklearn.ensemble import RandomForestClassifier
         # n_estimators - how many decision trees
         model = RandomForestClassifier(n_estimators=500)
         model.fit(X_train, y_train)

Out[12]: RandomForestClassifier(n_estimators=500)

In [13]: pred = model.predict(X_test)
         pred

Out[13]: array([0, 0, 0, ..., 0, 0, 0])

In [15]: from sklearn.metrics import accuracy_score
         accuracy_score(y_test, pred)

Out[15]: 0.908876437629018

In [16]: from sklearn.metrics import confusion_matrix
         confusion_matrix(y_test, pred)

Out[16]: array([[11634,   338],
                [  898,   694]], dtype=int64)
```

In this case, some Person gen gets a loan or not prediction correctness is 90%.We can see Random forest algorithm gives very high accuracy for the given problem.

Github Link

https://github.com/wrlakshan/Random-Forest