Implementation of KNN in spark on Bank Subscription Dataset

About the dataset:

The dataset is available in the UCI repository and is available at the link https://archive.ics.uci.edu/ml/datasets/Bank+Marketing. The data is about the marketing campaign of a financial banking institution. The objective is to predict whether the customer is going to subscribe to a term deposit or not. There were 20 factors in total which determined the acceptance of the term deposit by the customers.

Attribute #	Name	Type
1	Age	Numeric
2	Job	Categorical (Admin, Student)
3	Marital	Categorical (Single, Married)
4	Education	Categorical (school, university)
5	Default – has credit in default	Categorical (no, yes, unknown)
6	Housing – has housing loan or not	Categorical (no, yes, unknown)
7	Loan – Has Personal loan or not	Categorical (no, yes, unknown)
8	Contact – Means of contact	Categorical (cell, telephone)
9	Month - last month of contact	Categorical (Jan, Feb, Mar..)
10	Day_of_week – last contact day	Categorical (Mon, Tue, Wed)
11	Duration – duration of last contact in seconds	Numeric
12	Campaign – Total # of contacts	Numeric
13	Pdays – Numer of days since last contact	Numeric
14	Previous – Previous number of contacts	Numeric
15	Poutcome – Outcome of the previous campaign	Categorical (Failure, Non-existent, Success)
16	Emp.var.rate – Employment variation rate	Numeric
17	Cons.price.idx – Consumer Price Index	Numeric
18	Cons.conf.idx – Consumer Confidence Index	Numeric
19	Euribor3m – 3 months rate of an institute	Numeric
20	Nr.employed – Number of employees	Numeric
Output	Y – whether the client has subscribed to the term deposit	Categorical (yes, no)

Execution Instructions:

Ensure that the data files in the semi-colon separated values and the python file/the notebook file are in the same directory. In case of the jupyter notebook, execute the code cell by cell. In case of the python file, open a command prompt in the current working directory and type the following command:

```
spark-submit knn.py > knnoutput.out
```

Implementation:

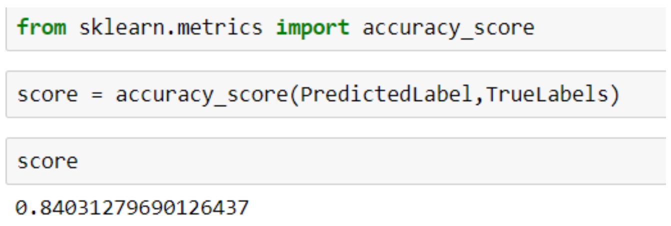
All the categorical variables have to be converted into numerical to build machine learning models. There are two ways in which it could be done: the first is by Label encoding and the second is by one hot encoding.

K-nearest neighbor is a simple classification algorithm where the label for the test sample is assigned based on the labels of the k closest training samples. A majority vote will determine the class to which the test sample belongs to.

A mapper essentially would compute the distance of the test sample with all the training samples and the reducer would sort the distances and compute the class label based on the value of 'k'. The implementation would require longer running time in case of bigger training dataset as the distance has to be computed for every test sample with each and every training sample.

Results and Analysis

The samples were split into training and testing in the ratio of 70:30. With the label encoding approach, an accuracy of almost 85% was obtained on the test dataset. There is no implementation of KNN available in spark MLLib. So another way standard way to verify the performance of the algorithm implemented was by using Scikit learn. The accuracy obtained with Scikit learn was almost the same. The labels were generated by setting the value of k to 5. An accuracy improvement of 3% was obtained by using one hot encoding format. Normalization of data with the label encoding method provided a 2% increase in accuracy.



```
from sklearn.metrics import accuracy_score
```

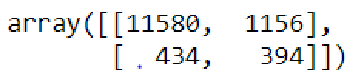
```
score = accuracy_score(PredictedLabel,TrueLabels)
```

```
score
```

```
0.84031279690126437
```

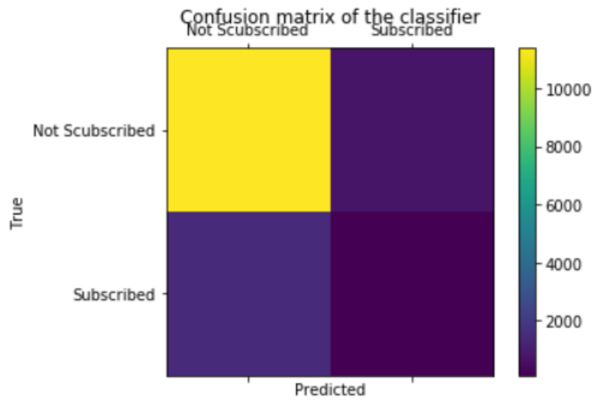
The confusion matrix obtained with the standard implementation is as shown below:

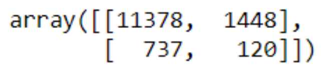
	Subscribed	Unsubscribed
Precision	0.9638	0.2545
Recall	0.9092	0.475
f1score	0.9375	0.3317



```
array([[11580,  1156],
       [  434,   394]])
```

On the other hand, the confusion matrix obtained with the implemented KNN algorithm is given below





```
array([[11378,  1448],
       [  737,   120]])
```

	Subscribed	Unsubscribed
Precision	0.9391	0.0765
Recall	0.8871	0.14
fscore	0.9123	0.0989

Further Work

- Do feature selection and feature importance
- Determine the optimal value of ‘k’
- Effectively scale the performance to larger datasets. It is highly in effective to compare every test sample with each and every training sample and measure distance.
- Use a different distance metric