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 Portuguese financial banking institution. The objective is to predict whether the customer is going to subscribe to a term deposit or not. There are four data files in the csv format of which two are full-sized documents and the other two are samples containing 10% of the records from the first two documents. 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 file named ‘bank-full.csv’ 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.

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StringIndexer,OneHotEncoder
```

```
def ConvertCategoricaltoNumeric(Dataframe,column):
    indexer = StringIndexer(inputCol=column, outputCol= "Numeric" + column)
    indexed = indexer.fit(Dataframe).transform(Dataframe)
    encoder = OneHotEncoder(inputCol="Numeric" + column, outputCol="categoryVec" + column)
    encoded = encoder.transform(indexed)
    return encoded
```

Figure 1 : Snapshot depicting the functionality of spark to achieve label encoding and 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 voting and inverse distance weighted voting are two of the techniques that could be used to determine the class label for the sample

A mapper essentially would compute the distance of the test sample with all the training samples and the reducer would sort the distances in the ascending order and compute the class label based on the value of ‘k’. In the implemented KNN, the mapper computes the Euclidean distance for a test data point with all the training samples using the distance function available in Scipy. 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

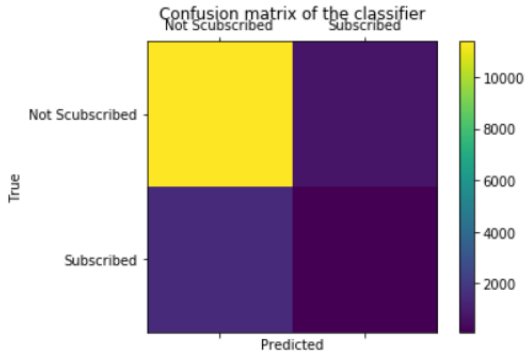
The samples were split into training and testing in the ratio of 70:30. With the label encoding approach, an accuracy of almost 85% was obtained on the test dataset. There is no implementation of KNN available in spark Machine learning library(MLlib). Another standard way to verify the performance of the algorithm implemented was by using Scikit learn. The accuracy obtained with Scikit learn was almost the same.

```
from sklearn.metrics import accuracy_score

score = accuracy_score(PredictedLabel,TrueLabels)

score

0.84031279690126437
```

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

Figure 2: Confusion Matrix

Figure 3 : Snapshot of the accuracy obtained with the KNN implemented in SPARK using Label Encoding Approach



	Not Subscribed	Subscribed
Precision	0.9391	0.0765
Recall	0.8871	0.14
fscore	0.9123	0.0989

Figure 4:Precision Recall and f1 score for the implemented KNN algorithm

Figure 5:Pictorial Representation of the confusion matrix

The accuracy report and the confusion matrix obtained with the standard implementation (scikit learn) using the labeled encoding approach is as shown below:

```
score = accuracy_score(PredictedLabels,TrueLabels)

score

0.88122972574461811
```

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

Figure 6:Confusion Matrix

Figure 7: Accuracy with scikit learn with label encoded categoric features

	Not Subscribed	Subscribed
Precision	0.9638	0.2545
Recall	0.9092	0.475
fscore	0.9375	0.3317

Figure 8: Precision,Recall and F1 score obtained with scikit learn implementation

Further Work

- There is accuracy to be gained with one hot encoding format. Once when the class labels are generated from the model built on the test data, it could be compared to see how it stands out when label encoding is performed.
- Analyze the features, look for similarity. Implement feature selection and find out the most important features in the given dataset.
- Determine the optimal value of ‘k’. Evaluate the implemented algorithm for different values of k and find out the value which results in better performance indicators on the given dataset
- Effectively scale the performance to larger datasets. It is highly ineffective to compare every test sample with each and every training sample and measure distance.
- Use a different distance metric. There are metrics like Manhattan, Cosine distance etc., that could be used to calculate the distance between the samples in vector space