Professor Eduardo Kugler Viegas
Frameworks de Big Data

Prática Spark e MLib

Considerando o caso a seguir, implemente as soluções em Python para extrair o conjunto de informações solicitadas

Descrição: Você foi contratado por uma empresa para desenvolver um sistema de detecção de alertas em dispositivos. Esta empresa possui uma base de dados sobre as medições obtidas de milhões de dispositivos. Sendo que, para cada dispositivo, as medições foram agrupadas e os seguintes dados foram extraídos:

Campo	Descrição
Hora	Hora média das medições
Minuto	Minuto médio das medições
Temp_minima	Temperatura minima das medições
Temp_maxima	Temperatura máxima das medições
Latitude_media	Latitude média das medições
Longitude_media	Longitude media das medições
Classe	Estado do medidor (Frio, Moderado, Quente, Alerta)

A base de dados foi fornecida no formato CSV, sendo que cada entrada (estatística sobre a medição do dispositivo), é representada por uma linha no arquivo. Enquanto cada linha possui os campos listados previamente, estes separados pelo caractere ",". A imagem a seguir exibe as 5 primeiras entradas presentes na base.

```
hora,minuto,temp_minima,temp_maxima,latitude_media,longitude_media,Classe
11.312758,30.169239,-1.859,27.495,36.169994,139.23022,Moderado
11.292323,29.638779,8.542997,36.177994,31.349113,73.50964,Quente
11.600304,29.64275,-1.861,27.695002,36.17158,139.22937,Moderado
11.462425,30.157314,9.776998,36.077995,31.351593,73.5104,Quente
```

Diante desse contexto, você foi encarregado pelo desenvolvimento de uma solução em Apache Spark e Apache Spark MLib que permita a identificação da classe Alerta nos dispositivos. Para tanto, complete as seguintes tabelas:

Dataset	***Classificador***	***ACC (%)***	***FP (%)***	***FN (%)***
4 classes	Árvore de Decisão			
	Random Forest (20 trees)			
	Random Forest (100 trees)			
	Ensemble dos 3 classificadores			
2 classes	Árvore de Decisão			
	Random Forest (100 trees)			
	Gradient-boosted tree (20 trees)			
	Ensemble dos 3 classificadores			

Matriz Confusão (4 classes), ensemble dos 3 classificadores				
Classe/Classificado	Frio	Moderado	Quente	Alerta
Frio				
Moderado				
Quente				
Alerta				

Matriz Confusão (2 classes), ensemble dos 3 classificadores		
Classe/Classificado como	Outros	Alerta
Outros		
Alerta		

Matriz Confusão (2 classes), ensemble dos 3 classificadores, com rejeição utilizando limiar de 90%			
Classe/Classificado como	Outros	Alerta	Rejeitado
Outros			
Alerta			

Busca exaustiva de conjunto de features (teste de todos os subconjuntos possíveis, 2^6)				
Dataset	***Melhor Resultado de acordo com ACC (Árvore)***	***ACC***	***FP (%)***	***FN (%)***
2 classes	1º melhor			
	2º melhor			
	3º melhor			
	4º melhor			

Para os testes:

- Utilize o dataset de treinamento (treinamento.csv) para geração dos modelos, e o dataset de testes (teste.csv) para avaliação dos modelos obtidos;
- Para a avaliação dos resultados converta o DataFrame com o resultado para um RDD;
- Implemente a técnica de ensemble através da manipulação dos resultados individuais no RDD;

Ordem recomendada dos passos para obtenção dos resultados:

Para cálculo das métricas de classificação individuais

1. Carregar os arquivos em um DataFrame
2. Criar os campos de features, label e índice nos DataFrames
3. Treinar os modelos
4. Avaliar os modelos e obter o DataFrame com os resultados
5. Converter o DataFrame com os resultados para RDD e calcular a matriz de confusão individual

Para cálculo das métricas de classificação do ensemble

6. Combinar os RDDs com o resultado individual de cada classificador em um único RDD. Utilize a função *join*, que é aplicada por um RDD<K,V> sobre outro RDD<K,V> e gera um RDD de saída com <K,[V,V]>
7. Calcule acurácia através do voto majoritário

Para cálculo das métricas de classificação individuais com duas classes

8. Atualize o campo do label para apenas duas classes nos DataFrames
9. Treinar os modelos
10. Avaliar os modelos e obter o DataFrame com os resultados
11. Converter o DataFrame com os resultados para RDD e calcular a matriz de confusão individual

Para cálculo das métricas de classificação do ensemble com duas classes

12. Combinar os RDDs com o resultado individual de cada classificador em um único RDD. Utilize a função *join*, que é aplicada por um RDD<K,V> sobre outro RDD<K,V> e gera um RDD de saída com <K,[V,V]>
13. Calcule acurácia através do voto majoritário

Para cálculo das métricas de classificação do ensemble com duas classes utilizando rejeição

14. Crie o DataFrame com os valores de probabilidades dos classificadores individuais
15. Combinar os RDDs com o resultado individual de cada classificador em um único RDD. Utilize a função *join*, que é aplicada por um RDD<K,V> sobre outro RDD<K,V> e gera um RDD de saída com <K,[V,V]>
16. Calcule acurácia através do voto majoritário, considerando a limiar de 90% para rejeição

Busca exaustiva de subconjunto de features

17. Boa sorte =)

Código Apache Spark e MLib (lembre-se de atualizar os IPs e caminhos dos arquivos)

Iniciar sessão

```
import os
os.environ['PYSPARK_PYTHON'] = '/usr/bin/python3'

from pyspark.sql import SparkSession
sc = SparkSession \
    .builder \
    .master('spark://172.18.0.11:7077') \
    .config('spark.executor.memory', '512mb') \
    .getOrCreate()
```

Criar DataFrame

```
df = sc.read \
    .option('delimiter', ',') \
    .option('header', 'true') \
    .option('inferschema', 'true') \
    .csv('hdfs://172.18.0.10:9000/treinamento.csv')
```

Criar campo com ID

```
from pyspark.sql.functions import monotonically_increasing_id

df.withColumn('id', monotonically_increasing_id()).take(5)
```

Gerar DataFrame com resultados do modelo

```
result = model.transform(df)
```

Criar campo com a label

```
from pyspark.ml.feature import StringIndexer

df = StringIndexer(inputCol="Classe", outputCol="label")\
    .fit(df).transform(df)
```

Selecionar campos do DataFrame e gerar RDD

```
resultSelectedRDD = result.select('field1', 'field2', 'fieldN').rdd;
```

Criar campo com as features

```
from pyspark.ml.feature import VectorAssembler

features = ['f1','f2']

df = VectorAssembler(inputCols=features, outputCol='features').transform(df)
```

Aplicar função sobre um campo do DataFrame

```
from pyspark.sql.functions import udf
from pyspark.sql.types import DoubleType

def funcaoX(x):
    if x % 2 == 0:
        return 1.0
    return 0.0

minha_udf = udf(funcaoX, DoubleType())

df = df.withColumn('CampoParaFuncao', minha_udf(df.CampoParaFuncao))
```

Treinar classificador (DecisionTreeClassifier, RandomForestClassifier (numTrees=X), GBTClassifier (maxIter=X))

```
from pyspark.ml.classification import DecisionTreeClassifier, RandomForestClassifier

tree = DecisionTreeClassifier(labelCol="label", featuresCol="features")
modeloDT = tree.fit(df)
```

Código Apache Spark

	Função	Entrada	Saída	Descrição
Transformações	map(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, deve gerar um valor de saída
	filter(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, gera um RDD apenas com valores que retornaram True
	flatMap(função)	RDD <Valores>	RDD <Valores>	Executa a função sobre todo valor de entrada, pode gerar 0 ou mais valores de saída
	sample(*withReplacement*, *fraction*)	RDD <Valores>	RDD <Valores>	Gera um RDD com *fraction* % do RDD de entrada
	distinct()	RDD <Valores>	RDD <Valores>	Gera um RDD com valores únicos do RDD de entrada
	groupBy(função)	RDD <Valores>	RDD <Chave, Valores>	Agrupa os valores de acordo com o valor retornado pela função
	groupByKey()	RDD <Chave, Valor>	RDD < Chave, Valores>	Agrupa os valores de acordo com a chave
	reduceByKey(função)	RDD <Chave, Valor>	RDD <Chave, Valor>	Gera um RDD com os valores agrupados de acordo com a chave (função recebe 2 parâmetros)
	sortBy(função)	RDD <Valores>	RDD <Valores>	Gera um RDD ordenado de acordo com os valores retornado pela função
	sortByKey()	RDD < Chave, Valores>	RDD <Chave, Valores>	Gera um RDD ordenado de acordo com a chave
Ações	reduce(função)	RDD <Valores>	Valor	Gera um valor agregando o RDD de entrada (função recebe 2 parâmetros)
	sampleStdev()	RDD <Valores>	Valor	Gera o desvio padrão do RDD de entrada
	max()	RDD <Valores>	Valor	Gera o maior valor do RDD de entrada
	min()	RDD <Valores>	Valor	Gera o menor valor do RDD de entrada
	collect()	RDD <Valores>	Lista (Valores)	Recebe todos os valores do RDD como uma lista no driver
	count()	RDD <Valores>	Inteiro	Recebe um inteiro com a quantidade de elementos no RDD
	first()	RDD <Valores>	Valor	Recebe o primeiro valor do RDD
	take(n)	RDD <Valores>	Lista (Valores)	Recebe os N primeiros valores do RDD em uma lista
	countByKey()	RDD <Chave, Valores>	Dicionário (chave, inteiro)	Recebe um dicionário com a quantidade de valores para cada chave
	saveAsText(caminho)	RDD <Valores>	-	Gera um arquivo no caminho especificado com o RDD de entrada (pode ser escrito no HDFS)