Votre numéro d'anonymat : | | | |

BDLE – MU5IN852

Examen réparti 1 du 26 Novembre 2021
Durée : 2 Heures (au total)
Documents autorisés

Les réponses doivent être inscrites dans les cadres réservés puis sur intercalaires s'il manque de la place. Le barème est donné à titre indicatif.

1 Questions de cours (2 pts)

Question 1 (2 points)

Quelle est la différence entre l'opération reduce de Hadoop Map-Reduce et celle de Spark RDD ?

Réponse :

Dans Spark RDD, quelle fonction d'ordre supérieur utiliser lorsque on veut réduire une liste à l'aide d'une fonction qui n'est pas associative ?

Réponse :

Dans Spark RDD, quelle est la différence entre *reduce* et *reduceByKey* ?

Réponse :

Donner deux limitations de l'inférence de schéma de Spark Dataframe pour les données CSV ?

Réponse :

2 Requêtes en Dataframe (6 pts)

On considère le dataframe *soumissions* qui décrit, au format JSON, des métadonnées de soumissions d'articles de

conférence. Le schéma inféré par Spark est fournis ci-dessous. Tous les attributs sont obligatoires sauf ceux indiqués comme optionnels et aucun array n'est vide.

```
root
 |-- authors: array
 |    |-- element: struct
 |    |    |-- affil: string
 |    |    |-- orcid: long
 |-- id: long
 |-- keywords: array
 |    |-- element: string
 |-- material: struct
 |    |-- appendix: string (optionnel)
 |    |-- tool: string (optionnel)
 |    |-- video: string (optionnel)
 |-- reviews: array
 |    |-- element: struct
 |    |    |-- expertise: string
 |    |    |-- note: long
 |    |    |-- orcid: long
 |-- track: string
```

Chaque object décrit les informations d'une soumission et possède les attributs suivants :

- *id* correspond à l'identifiant de la soumission,
- *authors* indique la liste des auteurs de la soumission en précisant, pour chacun, son affiliation *affil* et son identifiant *orcid*.
- *track* décrit le type de soumission qui peut être soit "demo", soit "article", soit "tutorial" ;
- *material* décrit le support joint à la soumission qui dépend de track comme suit :
  - lorsque track = demo, seul *video* est renseigné ;
  - lorsque track = article, seul *appendix* est renseigné ;
  - lorsque track = tutorial, seul *video* est renseigné ;
- *reviews* indique les évaluations de la soumission. Pour chaque évaluation on renseigne l'identifiant du reviewer *orcid*, son niveau d'*expertise*, ainsi que la *note* qu'il attribue.
- *keywords* contient une liste de mots-clés associés à la soumission.

Formuler, en utilisant l'API Dataframe en langage Python **impérativement** (sans avoir recours aux RDD), les requêtes qui retournent les informations suivantes.

Question 1 (1 point)

Les identifiants des auteurs qui ont une soumission dans un track dans lequel ils participent en tant que reviewer. Le schéma du résultat est (orcid Long).

Réponse :

Question 2 (1 point)

Les affiliations dont tous les articles soumis sont de type 'demo'. Le schéma du résultat est (affil String).

Réponse :

Question 3 (1 point)

Pour chaque soumission, la decision qui prend la valeur 'accept' si la moyenne des notes des reviews est >0, 'revise', si cette moyenne est =0 et 'reject' sinon (moyenne <0). On considère que la fonction utilisateur *decide* qui prend en entrée une note et retourne la décision associée a été définie. Le schéma du résultat est (id Long, moy Long, decision String).

Réponse :

Question 4 ($1\frac{1}{2}$ points)

Les paires de reviewers avec la moyenne de la différence des notes des articles qu'ils ont évalués en commun. Le schéma du résultat est (orcid Long, orcid1 Long, moyDiff Long).

Réponse :

Question 5 ($1\frac{1}{2}$ points)

Pour chaque combinaison de mot-clé et de track, la plus grande note donnée par les reviewers aux soumissions de cette combinaison. Le schéma du Dataframe résultat doit être (keyword String, article String, demo String, tutorial String).

Réponse :

3 Spark ML (2 pts)

On considère le dataframe

Voitures(motor String, volume Long, annee Long, couleur String, kilom Long, achat String)

qu'on voudrait utiliser pour construire un arbre de décision pour prédire les achats des voitures (problème de classification) en fonction de certaines caractéristiques (type de moteur, année mise en circulation, ...). On utilise la libraire Spark ML pour encoder les attributs de Voitures suivant le principe montré en cours en invoquant le pipeline composé des étapes suivantes : *StringIndexer* puis *VectorAssembler* puis *VectorIndexer*. Ensuite, on voudrait invoquer la classe *DecisionTreeClassifier* pour inférer le modèle.

Voici les hypothèses considérées dans cet exercice.

- Le nombre de lignes de *Voiutres* est 10.000.
- L'attribut *motor* possède 3 valeurs distinctes et est toujours renseigné
- L'attribut *volume* possède 5000 valeurs distinctes et est toujours renseigné
- L'attribut *annee* possède 10 valeurs distinctes et n'est pas toujours renseigné
- L'attribut *couleur* possède 6 valeurs distinctes et n'est pas toujours renseigné
- L'attribut *kilom* possède 1000 valeurs distinctes et est toujours renseigné
- aucune ligne de Voitures ne doit être enlevée à l'issue des étape d'encodage

Question 1 (1 point)

En se basant sur les statistiques et hypothèses ci-dessus, indiquer les paramètres devant être spécifiés lors des différentes étapes de l'encodage de features ainsi que les valeurs qu'ils doivent prendre.

Réponse :

- Le parametre ..... de l'étape ..... prend la valeur .....
- Le parametre ..... de l'étape ..... prend la valeur .....

Question 2 ($\frac{1}{2}$ point)

On considère un extrait de Voitures restreint aux attributs motor et couleur et on visualise le résultat de l'étape *StringIndexer* appliquée sur cet extrait. Comment a-t-on paramété cette étape pour obtenir ce résultat ?

```
+-----+-------+---------+-----------+
|motor|couleur|idx_motor|idx_couleur|
+-----+-------+---------+-----------+
|s    |blanc  |1.0      |2.0        |
|d    |rouge  |2.0      |1.0        |
|d    |null   |2.0      |3.0        |
|e    |bleu   |0.0      |0.0        |
|d    |blanc  |2.0      |2.0        |
|s    |blanc  |1.0      |2.0        |
+-----+-------+---------+-----------+
```

Réponse :

Question 3 ($\frac{1}{2}$ point)

On voudrait sélectionner le meilleur modèle en exploitant la classe *CrossValidator* en testant deux valeurs du paramètre maxBins sur 5 échantillons (folds). De combien de coeurs doit on disposer pour paralléliser la recherche du meilleur modèle ? Quel paramètre utiliser ?

Réponse :