

Project Scala Pipeline





**Prepared by : CHARHABIL SANAA**



Architecture Big Data

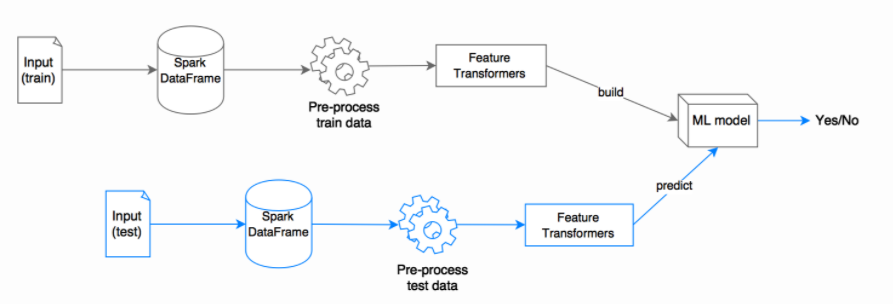
Année : 2020/2021

Professor : Mr KALLOUBi

1. Concepts et Définitions:

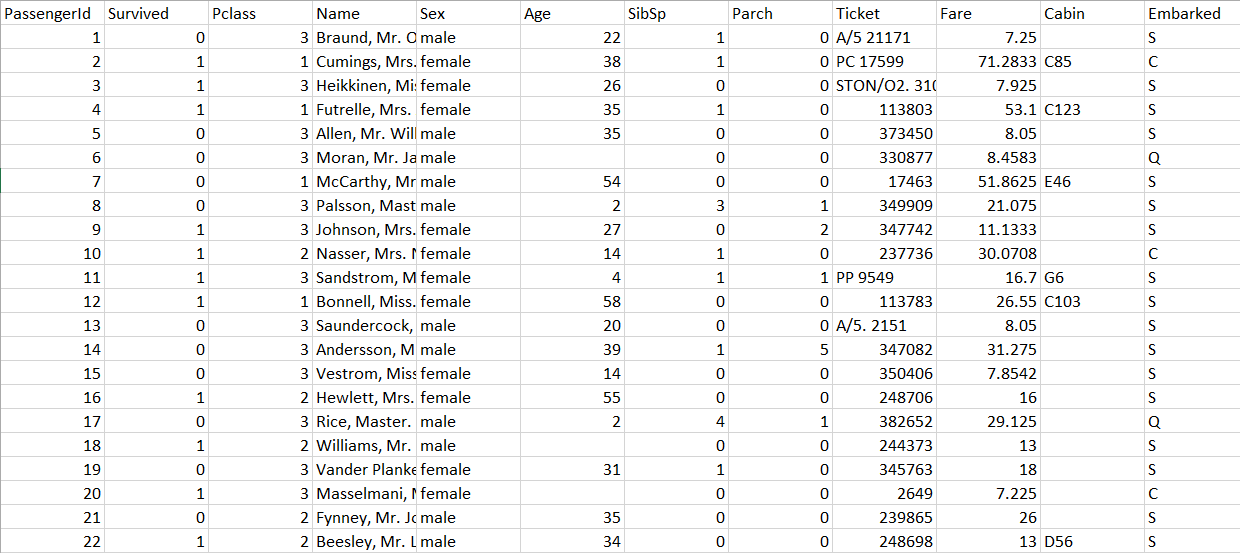
Principaux concepts dans les pipelines :

* **MLlib :** standardise les API pour les algorithmes d'apprentissage automatique afin de faciliter la combinaison de plusieurs algorithmes en un seul pipeline ou flux de travail. Cette section couvre les concepts clés introduits par l'API Pipelines, où le concept de pipeline est principalement inspiré du projet scikit-learn.
* **DataFrame:** cette API ML utilise DataFrame de Spark SQL en tant qu'ensemble de données ML, qui peut contenir une variété de types de données. Par exemple, un DataFrame peut avoir différentes colonnes stockant du texte, des vecteurs de caractéristiques, de vraies étiquettes et des prédictions.
* **Transformer:** Un Transformer est un algorithme qui peut transformer un DataFrame en un autre DataFrame. Par exemple, un modèle ML est un Transformer qui transforme un DataFrame avec des fonctionnalités en un DataFrame avec des prédictions.
* **Estimator:** Un Estimator est un algorithme qui peut être ajusté sur un DataFrame pour produire un Transformer. Par exemple, un algorithme d'apprentissage est un Estimator qui s'entraîne sur un DataFrame et produit un modèle.
* **Pipeline:** un pipeline enchaîne plusieurs transformateurs et estimateurs pour spécifier un flux de travail ML.
* **Paramètre:** tous les transformateurs et estimateurs partagent désormais une API commune pour la spécification des paramètres.



1. Création et insertion du Dataset au niveau de Spark:
2. **Charger les données MySQL:**
3. **Charger les données HIVE:**
4. Construction du JAR:

J’ai choisi le csv Titanic qui contient un ensemble de détails sur les passagers tels que le nom, le sexe, le tarif, la cabine, etc. et si la personne a survécu à la catastrophe du Titanic. Sur cette base, nous devons construire un modèle qui peut prédire, étant donné un autre passager, s'il est susceptible de survivre. Ceci est un exemple de classification binaire où il n'y a que deux classes possibles (1 si le passager survit et 0 sinon).

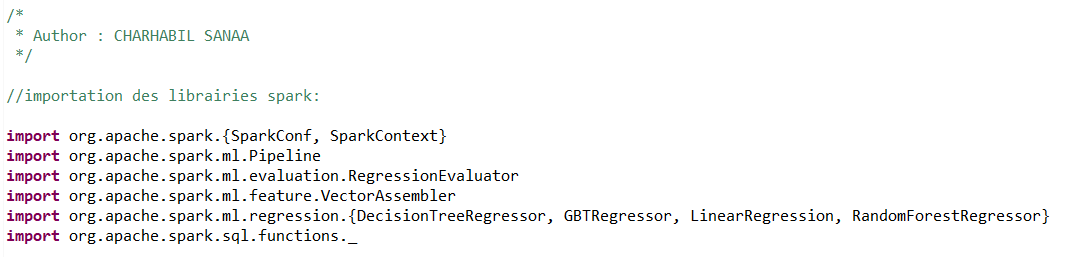


Voici les étapes qu’on va appliquer sur notre Dataset afin de conclure les prédictions :

* La première étape lorsque vous essayez de créer un modèle d'apprentissage automatique consiste à analyser et à comprendre les données dont vous disposez. Pour que vous puissiez décider quelles fonctionnalités doivent être utilisées pour construire le modèle, si les fonctionnalités sont numériques ou catégoriques, quelle est la moyenne, maximum ou minimum de vos fonctionnalités numériques, etc.
* Une fois les données analysées, l'étape suivante est la sélection des fonctionnalités où nous décidons quelles fonctionnalités sont pertinentes pour la construction du modèle
* Vient ensuite le prétraitement des données. La plupart du temps, les données d'entrée que vous recevez pour la modélisation ne seront pas de bonnes données. Au cours de cette étape, par exemple, nous pouvons décider quoi faire avec les valeurs manquantes - supprimer les lignes ayant des valeurs nulles, ou remplir celles avec la valeur moyenne de la fonctionnalité (si la fonctionnalité est numérique), ou remplir avec la valeur la plus fréquente de la caractéristique (si la caractéristique est catégorielle) etc.
* Vient ensuite l'étape de l'ingénierie des fonctionnalités et de la transformation des fonctionnalités. Dans l'ingénierie des fonctionnalités, nous dérivons de nouvelles fonctionnalités à partir de celles existantes et pendant la transformation des fonctionnalités, nous transformons les fonctionnalités existantes afin qu'elles puissent être utilisées pour créer le modèle.
* Enfin, nous construisons le modèle en utilisant les fonctionnalités sélectionnées et faisons des prédictions sur un nouvel ensemble de données.

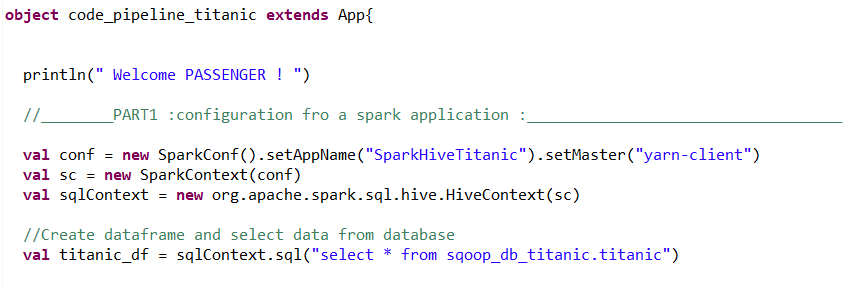
Nous implémenterons toutes les étapes ci-dessus à l'aide de Spark et Scala et créerons un pipeline d'apprentissage automatique - le flux global peut être illustré par le diagramme ci-dessous. La section grise du diagramme montre le flux de construction du modèle et la section bleue du diagramme montre le flux pour faire des prédictions.

**Importation des librairies :**



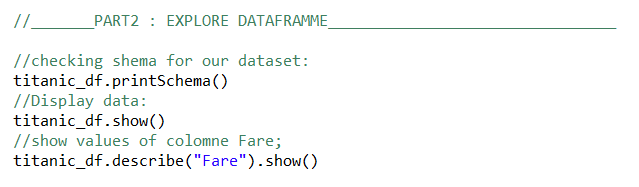
1. **Charger et analyser les données:**

**Configuration de l’application Spark et sélection des données à manipuler :**

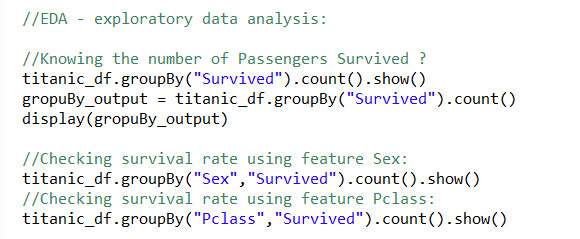


**Visualisation des données :**

Nous allons maintenant explorer le DataFrame chargé pour mieux comprendre les données. On peut vérifier le schéma des données chargées par exemple en appelant **printShema()**:



### **EDA - exploratory data analysis :**

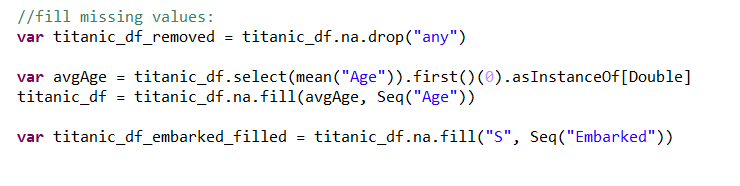


1. **Pré-traitement :**

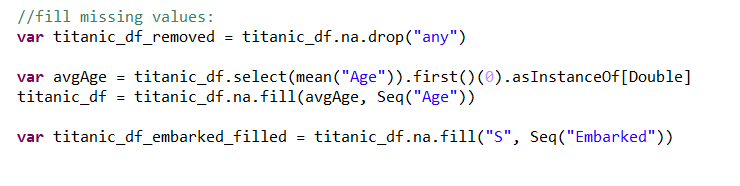
**Nettoyage des données-Remplacement des données manquantes :**

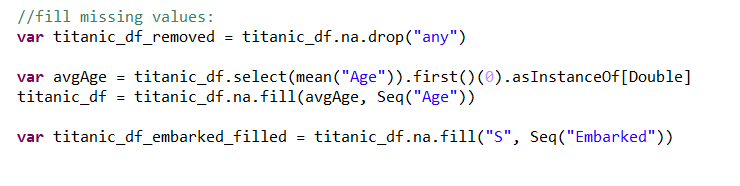
En analysant les données, vous pouvez y voir quelques irrégularités. Par exemple, il manque quelques valeurs dans la colonne Age. De même, il existe des valeurs nulles / manquantes dans Cabin, Fare et Embarked. Il existe plusieurs techniques pour remplir les valeurs manquantes. Vous pouvez :

1. Ignorez / supprimez les lignes ayant des valeurs manquantes. Cela peut être fait dans Spark en appelant :



1. Si la colonne est numérique, remplissez la valeur manquante avec la valeur moyenne / moyenne de la colonne. Nous allons remplacer les valeurs manquantes dans la colonne Age en utilisant cette méthode :

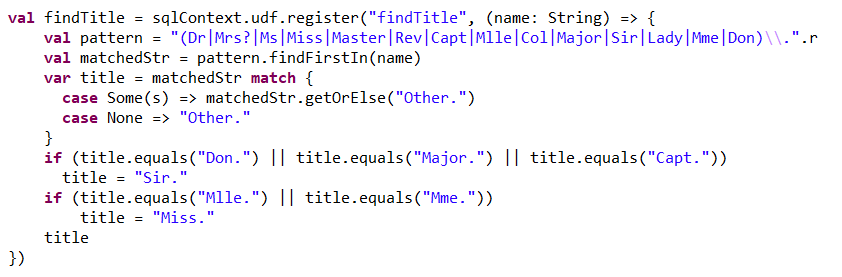


1. Si la colonne est catégorielle, remplissez avec la catégorie la plus courante : 

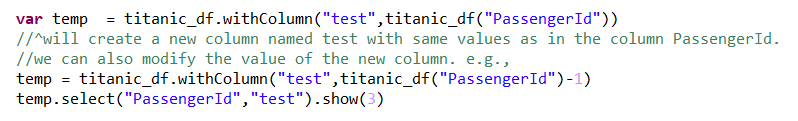
Créez un modèle d'apprentissage automatique qui peut prédire ces valeurs manquantes.

**Précision des Features :**

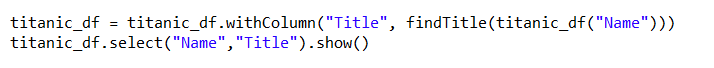
Dans de nombreux cas, il y aura des fonctionnalités dans vos données d'entrée qui peuvent être utilisées pour dériver de nouvelles fonctionnalités qui aideront à créer un meilleur modèle. Cela s'appelle également l'ingénierie des fonctionnalités. Par exemple, si vous regardez de plus près la colonne «Nom», vous pouvez voir que le format est FirstName Title. Nom de famille. Nous n’avons pas pu faire de prédiction en fonction du nom du passager, mais il se peut qu’il y ait une relation entre le titre et la survie du passager. Donc, nous allons extraire le titre de chaque nom et former une nouvelle colonne / fonctionnalité. L'udf findTitle est utilisé pour extraire le titre d'une chaîne donnée.



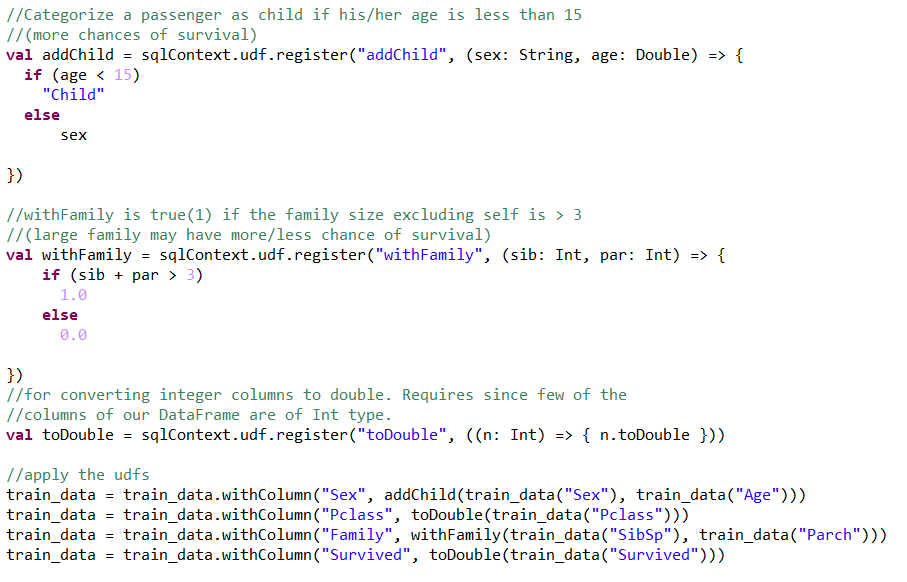
DataFrame fournit une méthode **withColumn()**  qui peut être utilisée pour ajouter / remplacer une colonne existante. Il prend deux paramètres - le nom de la nouvelle colonne et une colonne du DataFrame actuel. c'est-à-dire si vous appelez :



Nous allons maintenant appliquer la fonction **findTitle()** sur la colonne Nom pour extraire le titre et créer une nouvelle colonne - Titre.



De même, nous définirons 3 autres udfs, à l'aide desquels nous générerons de nouvelles fonctionnalités.

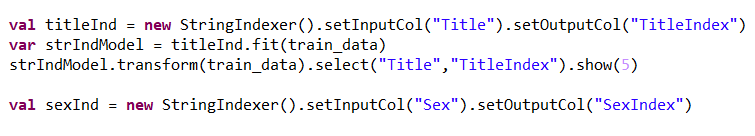


**Composants du pipeline :**

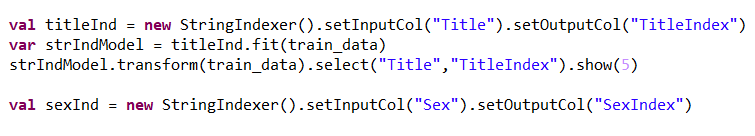
Le pipeline ML aura une séquence de composants de pipeline. Il existe deux types de composants: les transformateurs et les estimateurs. Transformers transforme le Dataframe d'entrée en un nouveau DataFrame à l'aide de la méthode **transform().** Un Estimator ajuste d'abord un modèle aux données, à l'aide de la méthode **fit(),** puis transforme. Celles-ci seront plus claires une fois que vous aurez parcouru les composants ci-dessous.

StringIndexer

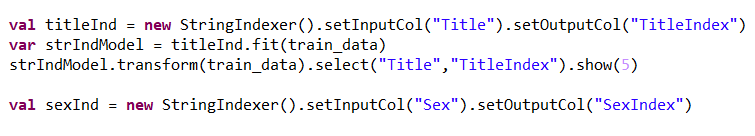
Pour créer un modèle dans Spark, les fonctionnalités doivent être du type Double mais nous avons quelques fonctionnalités qui sont du type String. Spark fournit un Feature Transformer - StringIndexer qui peut être utilisé pour cette transformation.



Ici, StringIndexer est un Estimator qui transforme la colonne Title, génère des indices pour les mots et crée une nouvelle colonne nommée TitleIndex. La méthode Fit de StringIndexer convertit la colonne en StringType (si ce n'est pas StringType), puis compte l'occurrence de chaque mot. Il trie ensuite ces mots par ordre décroissant de leur fréquence et attribue un index à chaque mot. La méthode StringIndexer.fit () renvoie un StringIndexerModel qui est un Transformer.

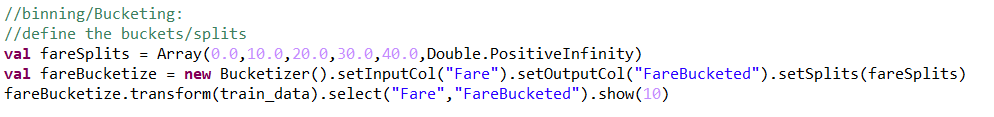


M. est le mot le plus fréquent dans ces données, il reçoit donc l'index 0. De même, nous allons également créer un indexeur pour la fonctionnalité – Sexe



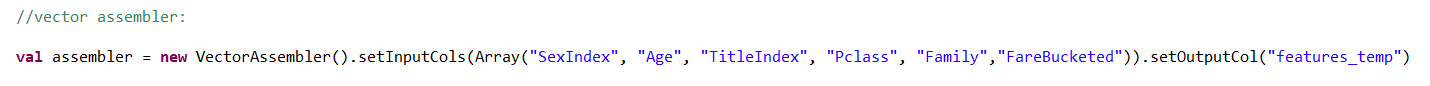
**Rangement / regroupement**

Pendant le Binning / Bukceting, une colonne avec des valeurs continues est convertie en buckets. Nous définissons la valeur de début et de fin de chaque bucket lors de la création du Bucketizer - qui est un Transformer. Nous allons regrouper la colonne "Tarif".



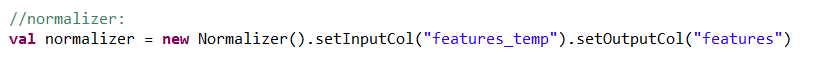
Assembleur de vecteur

VectorAssembler est utilisé pour assembler des entités dans un vecteur. Nous passerons toutes les colonnes que nous allons utiliser pour la prédiction au VectorAssembler et il créera une nouvelle colonne vectorielle.



Normaliseur

Ensuite, nous normaliserons ou normaliserons les données à l'aide du transformateur - Normalizer. Le normalisateur prendra la colonne créée par VectorAssembler, la normalisera et produira une nouvelle colonne.



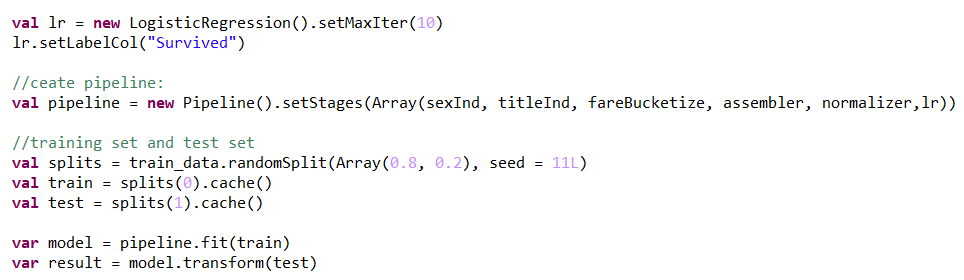
**Création et Application des modèles de machine learning :**

Modèle de construction et d'évaluation

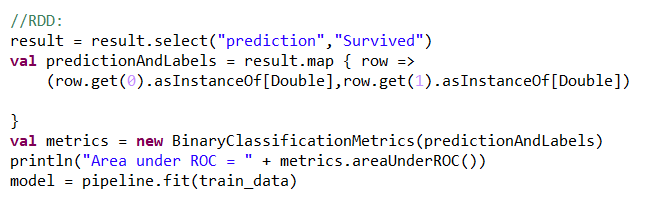
Nous allons construire notre modèle en utilisant l'algorithme LogisticRegression qui est utilisé pour la classification. La variable classée est appelée variable dépendante et les autres variables qui déterminent la valeur de la variable dépendante sont appelées variables indépendantes.

Dans la régression logistique, basée sur les valeurs des variables indépendantes, elle prédit la probabilité que la variable dépendante prenne l’une de ses valeurs catégorielles (classes). Dans notre exemple, il existe deux classes possibles 0 ou 1. Pour créer un composant LogitsticRegression,

* **Régression Logistique :**



Evaluation du model :



* **DecisionTreeClassifier:**
* **Régression Logistique :**
* **Support Vector Machine SVM :**

1. **Application de split sur les données :**
2. **Implémentation des algorithmes pour la prédiction :**
3. Le code: