Universidad de Buenos Aires
Facultad de Ingeniería

Seminario de Programación Funcional en Scala

Segundo cuatrimestre 2019

Trabajo Práctico

Martin Bosch - 96749

Informe

Ejecución del proyecto

Para ejecutar algunos de los módulos pedidos se debe parar en el directorio más alto y ejecutar

```
$ sbt run
```

y luego seleccionar el Main que corresponda al módulo que se quiera ejecutar.

Una mejora para este punto sería poder tener tres subproyectos separados y un sbt que los relacione y maneje sus dependencias.

Diseño de la solución

Primer módulo [paquete “db”]

Para el desarrollo del primer módulo implemente la siguiente solución:

- una función para leer el csv
- una función para insertar una fila del csv en la base
- una función para insertar todas las filas en la base

Para representar cada fila del csv use una case class cuyos parámetros son las columnas del csv. Una mejora sobre este punto sería crear una case class para representar cada columna del csv y así poder hacer validaciones para cada columna, por ej, en el caso de las columnas “dolar_bn” y “dolar_itau” se podría validar que sean mayores o iguales a 0.

Para la función que inserta todas las filas del csv en la base de datos en lugar de utilizar un loop for utilice una función recursiva que va insertando las filas en la base hasta que la lista quede vacía.

Segundo módulo [paquete “randomForest”]

Para el segundo módulo implemente la siguiente solución:

- una función para obtener todas las filas de la base de datos
- una función para entrenar el algoritmo de machine learning

Para entrenar el algoritmo de machine learning no pude utilizar todas las columnas de los datos porque había ciertas columnas que requerían ser transformadas para ser entendidas por el algoritmo de machine learning, como por ej, la columna fecha, y decidí empezar

utilizando las que más información podían brindarme (dolar_bn, dolar_itau, dif_sem, ol_dif, ol_vol, aj_dif), y en caso de llegar con el tiempo incluir las demás, aunque finalmente no llegue.

En cambio otras columnas no aportaban ninguna información como la columna id (por ser un simplemente un identificador), unidad y mon (porque tienen el mismo valor en todas las filas, entonces no agregan ninguna información valiosa). Las columnas open, high, low, last cambian de valor para una fila por lo tanto decidí incluirlas pero seguramente puedan excluirse también por no aportar información útil.

Tercer módulo [paquete “myserver”]

Para el tercer módulo implemente una función para que dados los valores pasados al servicio devuelva todas las filas en la base que coincidan con esos valores. Luego use pattern matching para chequear si estos valores ya estaban en la base o no. En caso de existir devuelvo la columna cierre de esa fila, y en caso de que no, predigo su valor utilizando el modelo entrenado con el anterior módulo, guardo el registro en la base (con el valor predecido para cierre) y devuelvo la predicción.

Evaluación del modelo entrenado

Para evaluar el modelo de machine learning utilice un cliente de http4s para pegarle al endpoint de predicción con los datos del archivo test.csv y comparar el campo cierre dado contra el predecido. Algunos valores que obtuve se muestran en la siguiente tabla:

Cierre dado	Cierre predecido
234.0	201.2716
236.0	184.2290
237.0	183.9115
238.0	173.3615

Se puede observar que el modelo no predice bien, ya que la diferencia entre el valor dado y el predecido es muy grande. Esto se puede deber a los valores dados a los parámetros pasados al constructor del modelo de machine learning. Se podría probar variandolos y ver si la predicción mejora o no.

Deudas

Puntos que no llegue a finalizar del trabajo práctico:

- Utilizar Docker para correr los módulos dentro de containers.
- Utilizar un hash para chequear si los datos pasados al servicio existen en la base.
- Utilizar más campos para el algoritmo de machine learning.