PySpark - Used UK Car Price Prediction

Michele Baglioni 330085, Eraldo Marku 334567

1 Introduzione

In questo progetto ci siamo posti il problema di fare delle predizioni sui prezzi delle auto in base ad alcuni parametri inviati da un utente. Le predizioni sono rese possibili grazie a un modello calcolato in modo distribuito utilizzando Spark integrato con Hadoop. Successivamente è stato fatto il deploy del modello su un server per renderlo fruibile al cliente finale. Poiché è richiesto l'utilizzo di librerie per il machine learning, per sviluppare il job da assegnare a Spark è stato utilizzato python tramite l'interfaccia PySpark.

2 Dataflow e tecnologie utilizzate

2.1 Dataset e formato di input

La sorgente dati utilizzata è il dataset car price uk composto dai seguenti campi:

- brand: marca dell'auto
- model: modello dell'auto
- year: anno di immatricolazione
- transimission: tipo di cambio dellauto
- Mileage: miglia percorse
- FuelType: tipologia di carburante
- Tax: road tax inglese
- Mpg: miglia per gallone
- EngineSize: cilindrata motore:
- Price: prezzo

Il dataset scaricato da Kaggle è stato elaborato in modo da ottenere un unico csv. Per lo storage è stato utilizzato HDFS nel quale abbiamo preso in input il dataset. Dopo aver elaborato i dati e creato la pipeline, che verrà approfondita nel paragrafo successivo, il modello allenato sarà esportato tramite HDFS.

2.2 Tecnologie utilizzate

Nel progetto sono state utilizzate le seguenti tecnologie:

- Hadoop
- Spark
- Python
- PySpark (Interfaccia per apache spark in python)
- MLlib (libreria python per Apache Spark)
- Pyspark2pmml (esporta Apache Spark ML pipeline in formato pmml)
- OpenScoring (REST-API web service per l'utilizzo di modelli pmml)

2.3 Architettura e dataflow

Inizialmente con gli appositi script .sh carichiamo il dataset su HDFS. Dopodichè viene lanciato Spark con il job scritto in python. Il job assegnato a Spark, tramite le librerie di pyspark esegue le seguenti operazioni in modo distribuito:

1. Esegue delle operazioni di EDA per pulire il dataset.
2. Genera un pipeline cosi fatta:
   - Trasforma le features in forma vettorizzata dato che MLlib richiede questo formato.
   - Normalizza le features numeriche utilizzando un MinMaxScaler.
   - Costruisce un one-hot-vector per le features categoriche.
   - Allena un modello di regressione lineare con il training-set.

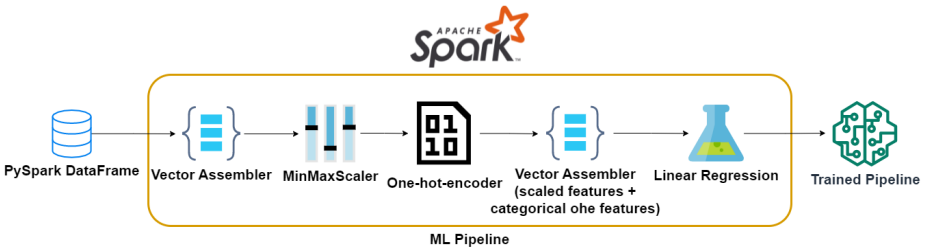

Figura 1: Pipeline utilizzata.

Successivamente il job ci restituisce l'R2 score in modo da valutare la qualità delle predizioni del modello. Siccome l'R2 score è soddisfacente (~ 0.87) viene riallenato il modello con l'intero dataset. L'ultima parte del job, tramite pyspark2pmml, esporta il modello allenato su HDFS in pmml. Questo formato permette tramite xml di descrivere modelli di analisi predittiva. Il file pmml viene salvato in locale dalla macchina addetta ad eseguire il server OpenScoring il quale permette tramite REST-API di fruire delle predizioni del modello. E' stato implementata anche un intefaccia utente in HTML+Javascript per simulare un caso d'uso in cui l'utente finale richiede la stima del prezzo della propria auto.

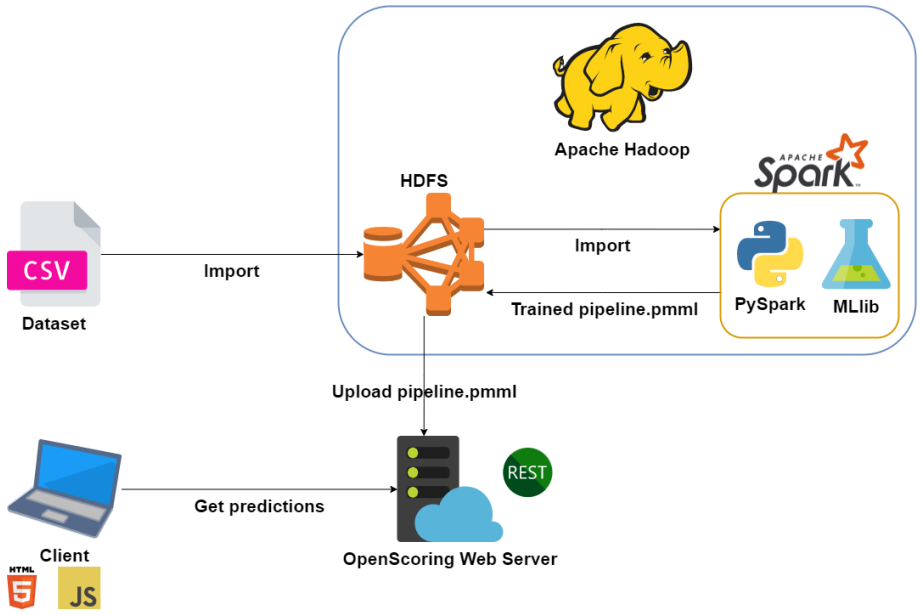

Figura 2: Architettura del sistema.

3 Use Case

Per eseguire il progetto sono stati implementati degli script .sh che automatizzano tutto il processo descritto in precedenza. Per prima cosa bisogna impostare dentro allo script 'PySpark/script/runSpark.sh' la seguente variabile in questo modo:

```
MY_PTH="your/path/to/PySpark-CarPricePrediction/PySpark"
```

Dopodiché per eseguire il progetto basta lanciare il seguente script:

```
sh runProject.sh
```

Quest'ultimo avvia Hadoop, esegue Spark con il job apposito, salva in locale da HDFS il modello allenato, avvia il server OpenScoring e carica il pmml all'interno. Inoltre tramite l'interfaccia web è possibile simulare un caso d'uso dell'utente finale che stima il prezzo della propria auto.

4 Limitazioni e possibili estensioni

Questo sistema potrebbe essere sicuramente migliorato su vari fronti. Innanzitutto la parte di machine learning potrebbe fornire migliori risultati effettuando una fase di exploratory data analysis più completa ed approfondita, inoltre si potrebbe anche provare ad addestrare diversi modelli di regressione e vedere quale ottiene una migliore performance.

Si potrebbe anche aggiungere un'ulteriore funzionalità del sistema: far si che un utente, o magari un amministratore del sistema, possa aggiungere nuovi campioni nel dataset. Considerando un'applicativo su larga scala e con grandi moli di dati si potrebbe tenere un contatore dei nuovi campioni inseriti. Una volta superata una certa threshold viene risottomesso il job in cui si ricalcola un nuova pipeline da mettere in produzione utilizzando un dataset contenente più informazione utile.