June 3, 2021

IDC Big Data Platforms - Final Project using Spark + ML, 2021

In the final project you will make use of different technologies we learned throughout the course with emphasis on using Spark for data ingestion, preprocessing, exploration and building a machine learning mode.

The project will be conducted in the framework of a private kaggle competition (Note: use this link to participate, but do not share it!).

Dataset

The dataset contains historical data about Uber rides in New York City. It can be loaded from here . The total size is about 1.5GB. The dataset has 7 columns for features (independent variables) + 1 column used for the label (dependent variable):

- Key - identifying each row in both the training and test sets. Unique integer. Required in your submission CSV. Not necessary in the training set, but could be useful to simulate a 'submission file' while doing cross-validation.
- Pickup_datetime is a `timestamp` value indicating when the Uber ride started.
- Pickup_longitude, Pickup_latitude, Dropoff_longitude, Dropoff_latitude - `floats` for longitude and latitude coordinates of where the Uber ride started and ended.
- Passenger_count - integer indicating the number of passengers.
- Fare ($) - float indicating the cost of the ride, used to create the label (class) for the classification model, or as the target for the regression model.

Here are a few sample rows with **fictitious** data (note missing and incorrect values!)

Key	Pickup_datetime	Pickup_ longitude	Pickup_ latitude	Dropoff_ longitude	Dropoff_ latitude	Passenger_ count	Fare ($)
101	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.84161	40.712278	1	4.5
102	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1	16.9
103	2011-08-18 00:35:00 UTC	-73.982738	40.76127	-73.991242	40.750562	2	
104	2012-04-21 04:30:42 UTC	-73.98713	40.733143	-290.99156	40.758092	1	7.7
105		0	0	0	0	3	5.5
105	2011-01-06 09:50:45 UTC	-74.000964	40.73163	-73.972892	40.758233	125	12.1
106	2009-01-09 16:10:00 UTC	-73.873027	40.773883	-73.984545	40.769545	3	-300

Task

1. You should build a classification machine learning model that tries to predict if the fare is lower or higher then $10. After training the model, given a set of (previously unseen) test data (w/o the fare) you should predict (classify) for each row if the fare is < 10 or >= 10.
2. You should write your code using Python and Spark, using MLlib for the ML code. Time all the steps (using %%time).
3. You can develop and run your code on an EC2 instance, and finally also run it on an EMR cluster. See tips for details.
4. Create a new boolean column "high_fare" for the training set from the fare amount (i.e. if fair is < 10 or >= 10). Use it as the label (dependent variable) for your classification model. Ignore the fare amount in the rest of this project, unless you want to try a regression model as an extra challenge.
5. Handle missing or incorrect data, e.g. eliminate negative or zero fares, for example replace missing values by the average or median value or delete them. Also read the pickup time in a way that it can be used. Use imputation where applicable.
6. Perform feature engineering. For example, create a ride_length feature that calculates the straight line distance of the ride. Also consider how to create useful features from the pickup time, such as day of week, hour of the day etc (search e.g. here).
7. Use plot or plotly with a Pandas dataframe (or another plotting package) to create the following plots (with a sample of 10,000):
    a. Distribution of ride fairs.
    b. Ride fare vs. ride length.
    c. Average ride length per pick up time hour of the day (x-axis = 24 hours).
8. Train your models using 80% of the training set, keeping 20% aside for validation (or cross-validation). Then test the models on the validation set. Output the model results - print the confusion matrix.
9. Try logistic regression first, decision tree and then either a random forest or a gradient-boosted tree or both. Use scaling where necessary. You need to achieve AUC (area under the curve) of at least 0.8 and hopefully better.
10. Add to your notebook code for handling of the test data (note you should avoid discarding any rows). Run your best model on the test data and create a submission file (prediction for idc_test.csv file, in a file that has only the key + high_fare label). Save the file and submit it to our private Kaggle competition with your team name, i.e. one or both of the team

members. You can have multiple submissions, according to the Kaggle limitations. The top 3 results (in the final public leaderboard) will get a bonus!

11. Your Jupyter notebook should have titles (markdown) for every section. E.g.: **cleaning the data**, **removing NULLs and NANs**, **Training Logistic Regression model**, **Testing Logistic Regression model** etc.
12. Copy your datasets and code to AWS EMR (a Spark cluster, see tips below). Run your code **on an EMR Notebook**, first with one worker, then with 8 workers. Print how long it took you to perform the major tasks, such as preprocessing, vectorization and training, in each case.
13. Make sure your notebook runs from beginning to end on an EC2 server or EMR server, without any errors. All outputs should be visible. Write in the beginning of the notebook if it should run on EC2 or EMR.
14. Submit your notebook, with output for all cells and running times (for all models, with 1 vs. 8 workers), in one or two Jupyter notebook.
15. If you want an extra challenge, try writing a regression model to predict the ride fare (ignore the $10 threshold).

Tips:

1. Install anaconda + pyspark + pandas + scikit-learn + pycm.
   (command: *pip3 install anaconda pyspark pandas scikit-learn pycm*)
2. On EC2 you also need to get spark + untar … into /opt/bin (only necessary if the cluster does not already have spark): see instructions here, up to Configure Spark Environment. On EMR just make sure the spark option is checked
3. Use the pycm package to print the confusion matrix.
4. The (unzipped) data file (.csv) is about 1.5 GB. Make sure to use an EC2 or EMR instance with at least 16GB of RAM and 16GB of storage (EBS volume), otherwise you may run out of space. For example m4.xlarge with a 16GB disk. Use the command "`df -h .`" to verify you have enough disk space.
5. Here is some code that will help you manage your spark memory, since the default settings are too low. It should be at the beginning of your notebook. I recommend that you set the size to at least half the RAM on the box (e.g. I used 16GB below on a box with 32GB RAM). Install and run htop in parallel (in another terminal) and watch the RAM utilization, and adjust these values if necessary.

```
from pyspark.sql import SparkSession, SQLContext

from pyspark import SparkConf, SparkContext
```

```
conf = SparkConf()

conf.set("spark.executor.memory", "16g")

conf.set("spark.driver.memory", "16g")

spark_context = SparkContext.getOrCreate(conf)

spark = SparkSession(spark_context)
```

6. Do not forget to release the spark memory at the end, by doing spark.stop(). You will notice in htop the memory is not reclaimed immediately. But if you restart the kernel it will actually go down.
7. To get the data follow the Kaggle guidelines. You will need to install Kaggle, get an API key from your Kaggle account and use the CLI to download the data to your AWS server. (commands:
   pip install kaggle
   mkdir .kaggle
   cd .kaggle
   vim kaggle.json (paste your credentials on 1st line, save the file)
   kaggle competitions download -f idc_train.csv idc-big-data-2020-ml-competition
   kaggle competitions download -f idc_test.csv idc-big-data-2020-ml-competition
8. Install unzip or a similar tool and unzip the data files.
   (commands:
   sudo apt install unzip
   unzip <idc_train.csv.zip>)
9. It may be a good idea to develop your code on an EC2 instance with a sample of the training set (e.g. 10%) in order for the process to run faster, and once the code is error-free retrain it on the entire training set to get a better model.
10. You can copy your data file from your EC2 instance to your EMR master node using scm:

    Copy your key pair file (.pem) to local directory, e.g.

    chmod 400 key-pair.pem

    scp -i key-pair.pem ubuntu@<EC2-remote-ip-or-hostname>:idc_train.csv .

11. **To run on a spark Cluster**:
    a. create an EMR cluster with a normal (non-Educate) AWS account. Use version 5.29, make sure to check Spark, JupyterHub and Livy. You can start with 1 master and 1 core (worker).
    b. Once the cluster is operational create an EMR notebook and attach to this cluster.

c. Open the notebook in Jupyter lab. Make sure to use the PySpark kernel. Then run your code. Resize the cluster to the number of workers you want, and run the code again to see the impact on running time.

d. You can upload the Kaggle data file (zip file with training + test data files) from your laptop, or use the wget command to copy it from another AWS server. Unzip it.

e. When working with the EMR notebook you may need to put the data on HDFS.

12. If you get an error message from Jupyter that it doesn't trust your notebook the then type this in a cmd prompt:

    jupyter trust <your-notebook.ipynb>