Comparing Machine Learning Algorithms Across Data Systems

Grace Davenport, Hayden French, Silas Hayes, Ryan Lipps

Motivation

- As **data scientists**, we are interested in using big data systems for the implementation of **scikit-learn** machine learning algorithms
- We are now familiar with several **distributed computing frameworks,** but have not directly applied them to the data science pipeline
- We are interested in using **Spark**, **Dask**, and **Ray** to implement some popular machine learning algorithms—**regression**, **classification**, and **clustering**—and compare compute times between each system

Data Preprocessing

- Dataset: **20 million flight searches** between June, 2022 and August, 2022 and is approximately **2 GBs**
- Variables: **(1)** flight search on a specific date, **(2)** returning base fare, **(3)** distance, **(4)** time traveled, and **(5)** number of seats remaining
- Prior to uploading our data to our cluster, we **normalized each variable** via **z-score scaling** and performed a random **80-20** train/test split
- We are more interested in the **overall compute times** than **model performance**, but still aim to mirror the traditional machine learning pipeline as closely as possible

Experimental Design

- We implemented each algorithm on **Dask**, **Spark**, and **Ray** hosted on **Amazon's EC2** instances

- For all algorithms, we used all **quantitative** variables as predictors. For KNN and Random Forest, our response variable was the **base price** of the flight. For K-Means, we performed **unsupervised clustering**

- We ran each algorithm on **differently-sized subsets** of our data and recorded the execution time for each

Sklearn + Joblib

- Parallelize sklearn with this **one neat trick!!**

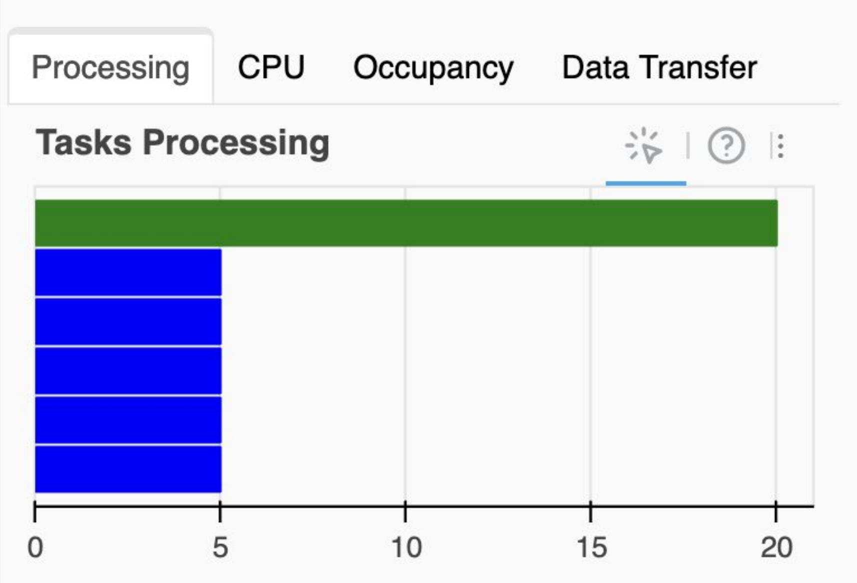
```
with joblib.parallel_backend('dask'):
    rf_model = RandomForestRegressor(max_depth=2)
    rf_model.fit(X_reg_train_subset, y_reg_train_subset)
    y_pred = rf_model.predict(X_reg_test_subset)
    metric = mean_squared_error(y_reg_test_subset, y_pred)
```

Joblib

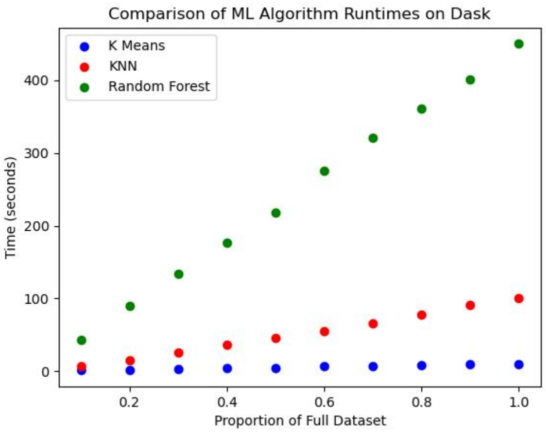
scikit learn

It’s not that simple.

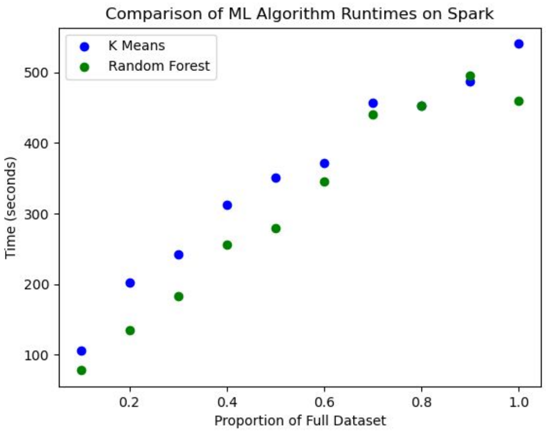
Sklearn + Joblib Actually Parallel?



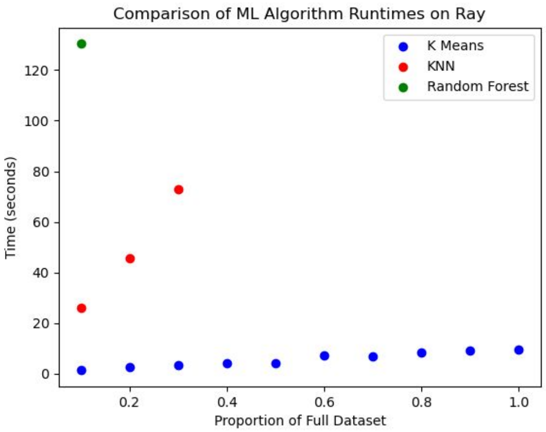
Discussion



Dask



Spark



Ray

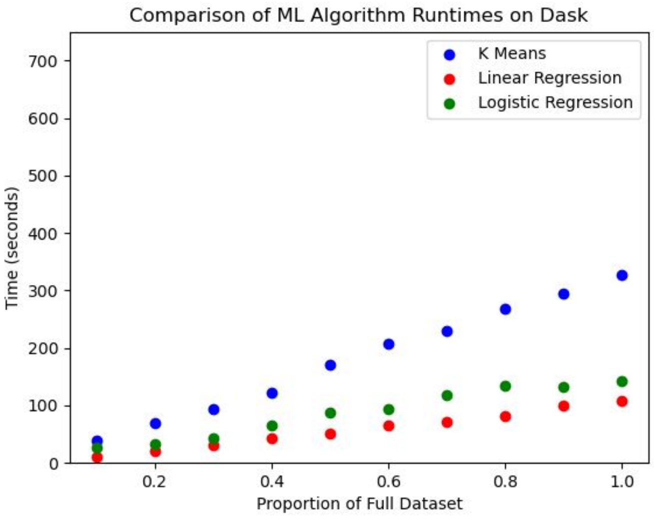
Modified Experimental Design

- We modified our approach to use Dask and Spark's distributed computing frameworks **own machine learning backend** instead of **sklearn**
- We selected algorithms which were supported by **both libraries** and representative of **different machine learning methods** (clustering vs. regression, e.g.)
- Again, we ran each algorithm on **differently-sized** subsets of the data to compare runtimes
- **Ray** does not have its own ML implementation for these algorithms though it is still useful for **RL** and **parallelized tuning**

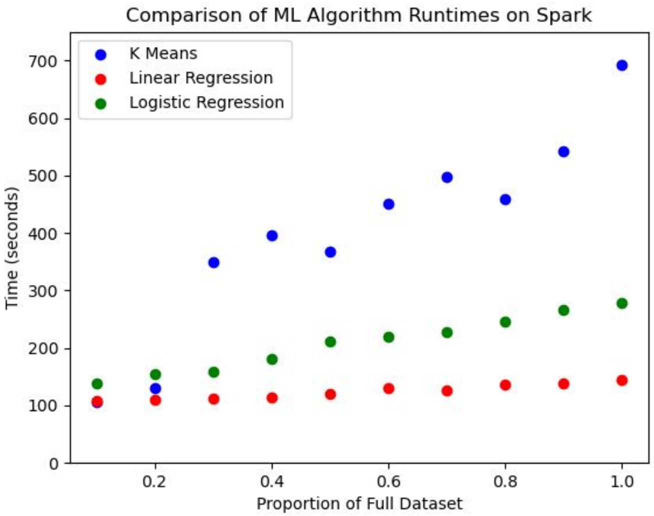
Results

	Dask	Spark
Linear Regression	108.04 s	144.91 s
Logistic Regression	142.80 s	279.03 s
K-Means	326.70 s	691.83 s

Discussion



Dask



Spark

Future Work

Custom Distribution

Assess these systems using custom sklearn distributing

Vary VM Numbers

Each system different number of VMs

More Precise Metrics

Calculate the mean and standard deviation

Thank you!