Machine Learning-Based Predictions for Optimal Job Scheduling

JOHNS HOPKINS UNIVERSITY

Amy He, Heather Han, Jennifer Baron, Sharmila Tamby
Department of Computer Science, Johns Hopkins University

Background

Many big data systems today must process large amounts of data in an inexpensive and efficient way. Specifically, predicting the resources needed for jobs is a key challenge for schedulers to maximize resource utilization and minimize job completion times. Current predictors lack precise information of the requirements of the services [1].

This project integrates novel approaches of both machine learning and progressive deadlines in order to dynamically predict CPU resources and improve resource prediction models.

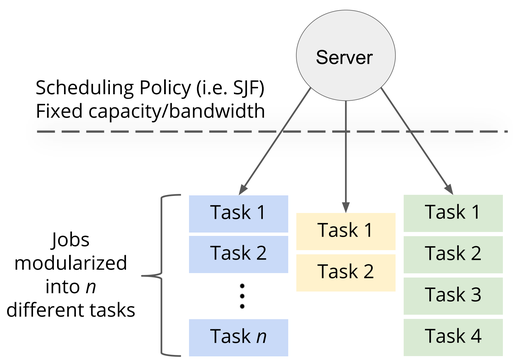


Fig 1. Visualization of progressive deadlines, in which jobs are modularized into specific "tasks" [2]. Using multiple deadlines corresponding to the progress of jobs mitigates issues that arise with the dynamic resource demands of jobs in a workflow.

Question

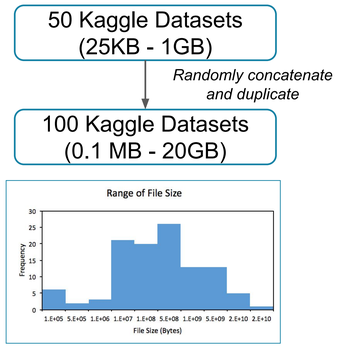
Will utilizing machine learning:

1) provide accurate resource predictions ?

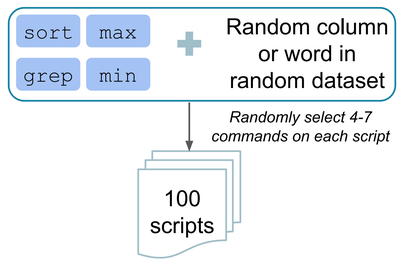
2) capture dynamic and specific information in cluster environments for progressive deadlines ?

Methods

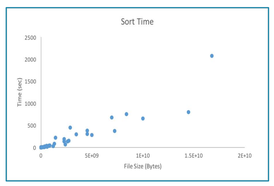
1. Generate Datasets



2. Generate Scripts

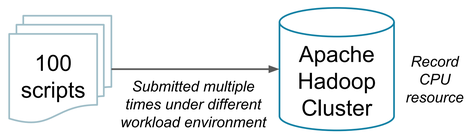


3. Select Features

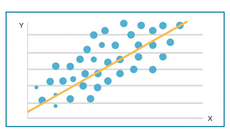


- File size, among other features, were tested for correlation with CPU resource

4. Run Scripts



5. Build and Evaluate ML Models



- Linear Regression
- Random Forest Regression

Results

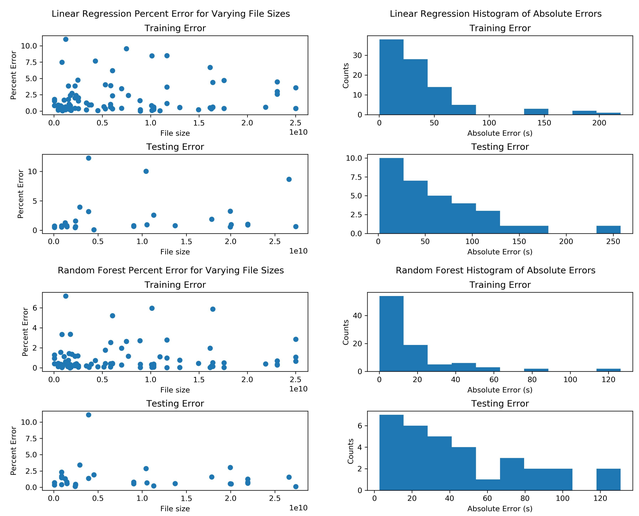


Fig 2. Error plots reflecting Linear and Tree Regression model performance on scripts each with 4-7 commands, where scripts were submitted multiple times, each time in a different cluster workload environment. File size is measured in bytes and error is measured in second.

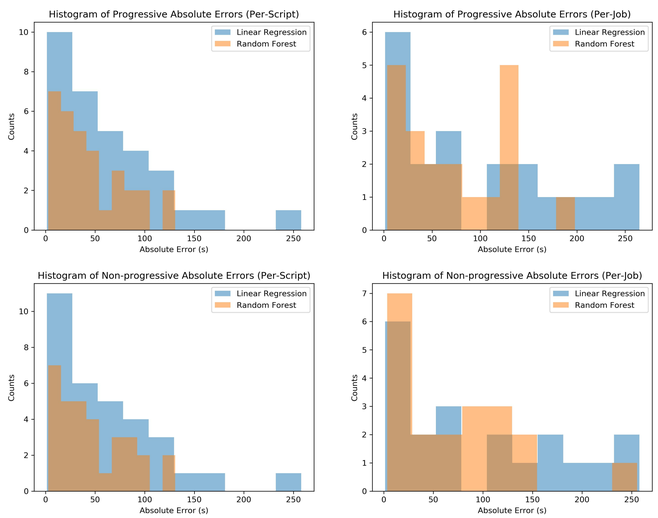


Fig 3. Histogram of absolute errors on scripts, where each script belongs to a certain job. The cluster workload feature for scripts is the first script's environment in a *non-progressive* job, and script-specific in a *progressive* job.

Conclusion

- Machine learning models and progressive deadlines are useful for dynamically predicting CPU resources
- Additional metrics needed to fully capture dynamic environments
- Understanding the dynamic resource needs of submitted jobs is a robust method to assist scheduling algorithms make informative and accurate decisions

Future Work

- Vary cluster configurations
- Incorporate larger, real-world datasets
- Build neural networks to approximate important features
- Prototype a scheduling algorithm (i.e. SJF) to incorporate our predictive methods

Acknowledgements

Thank you to Dr. Soudeh Ghorbani and the TAs of Cloud Computing for project guidance.

References

[1] Anderson, James H., Vasile Bud, and UmaMaheswari C. Devi. "An EDF-based scheduling algorithm for multiprocessor soft real-time systems." 17th Euromicro Conference on Real-Time Systems (ECRTS'05). IEEE, 2005.

[2] Gardner, Kristen, Sem Borst, and Mor Harchol-Balter. "Optimal scheduling for jobs with progressive deadlines." 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, 2015.

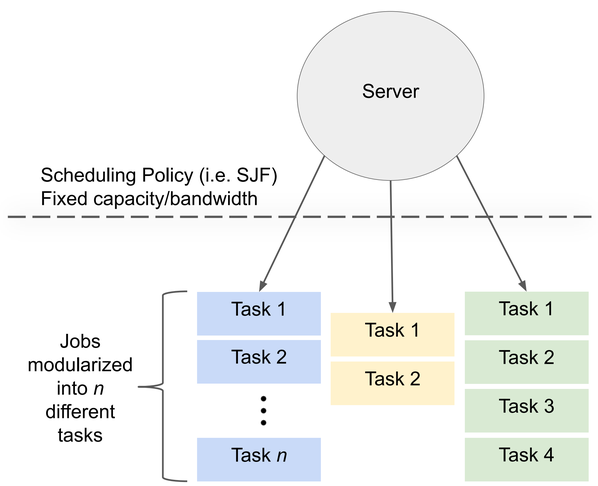
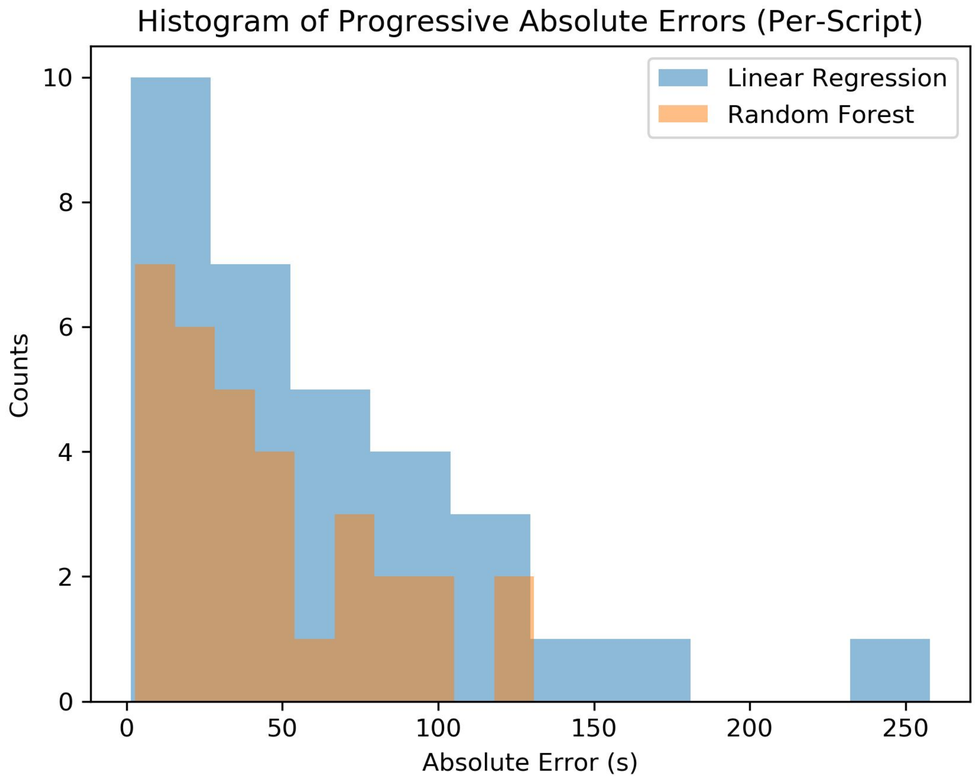
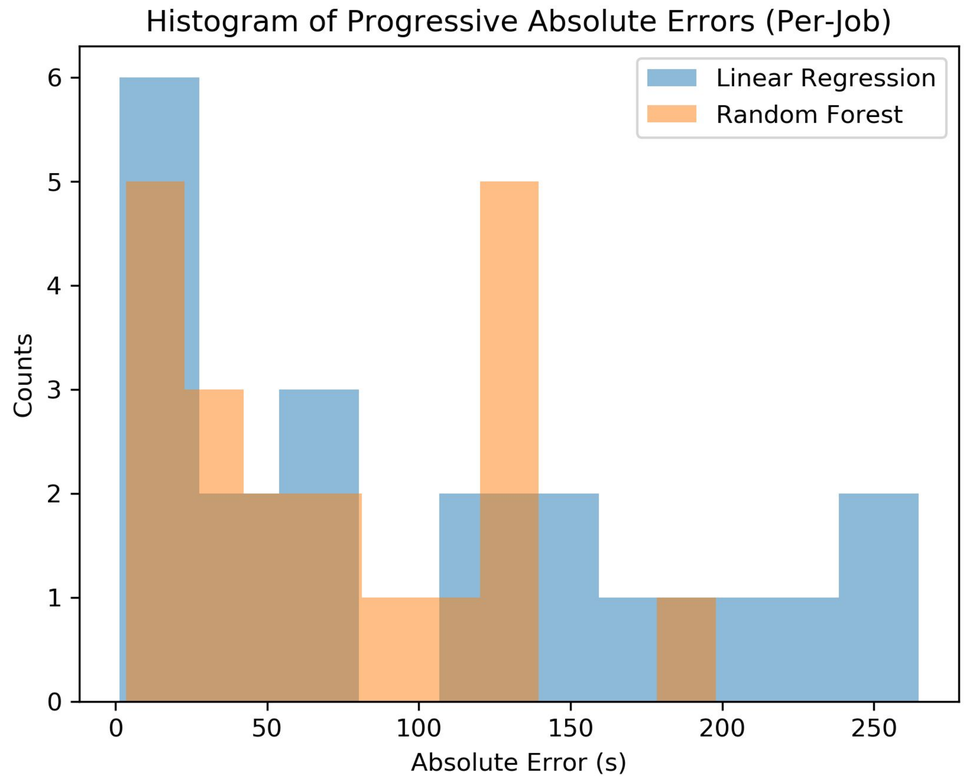
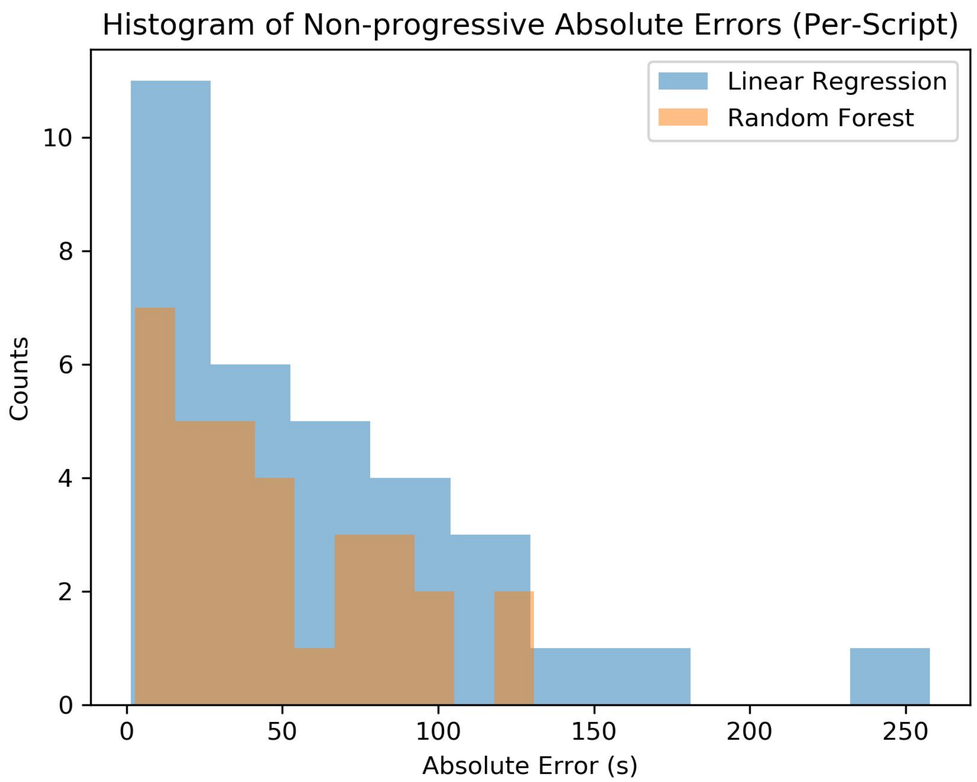
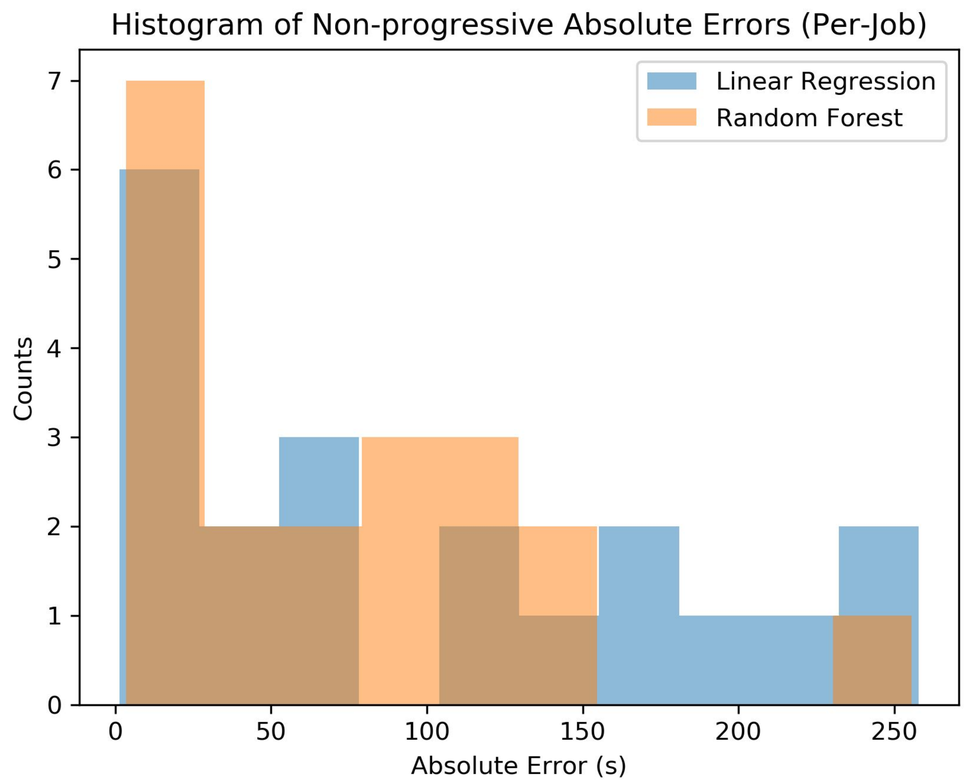


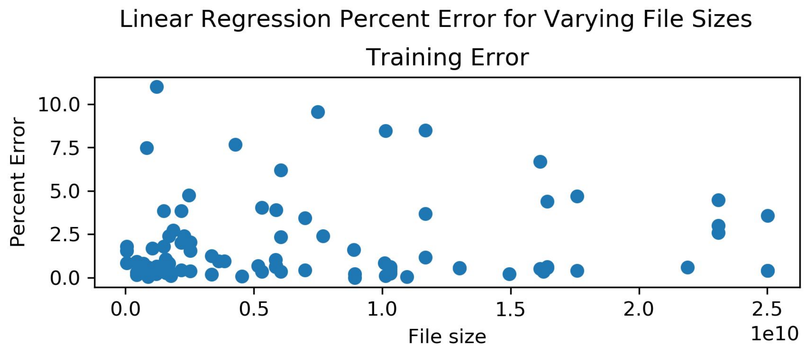
Fig 1. Visualization of progressive deadlines. Instead of scheduling an entire job once before entering the workflow, scheduling and resource allocation can instead be modularized into more specific "task"-specific levels, in which one job can have many tasks. Making predictions on a modular level and using multiple deadlines corresponding to the progress of jobs mitigates issues that arise with the dynamic resource demands of tasks and jobs in a workflow.

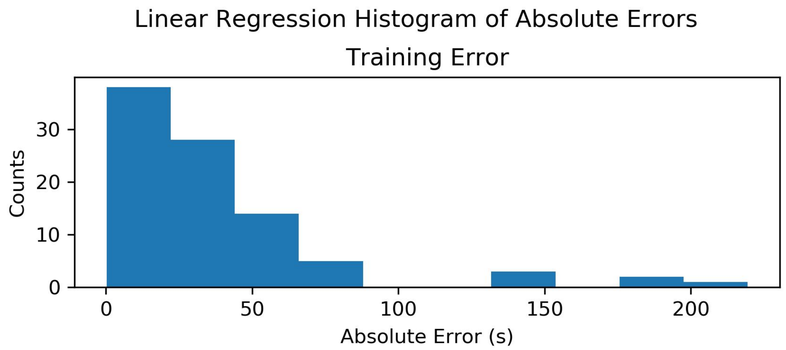

Histogram of Progressive Absolute Errors (Per-Script)
Linear Regression
Random Forest
Counts
Absolute Error (s)
0
2
4
6
8
10
0
50
100
150
200
250


Histogram of Progressive Absolute Errors (Per-Job)
Linear Regression
Random Forest
Counts
Absolute Error (s)
0
1
2
3
4
5
6
0
50
100
150
200
250

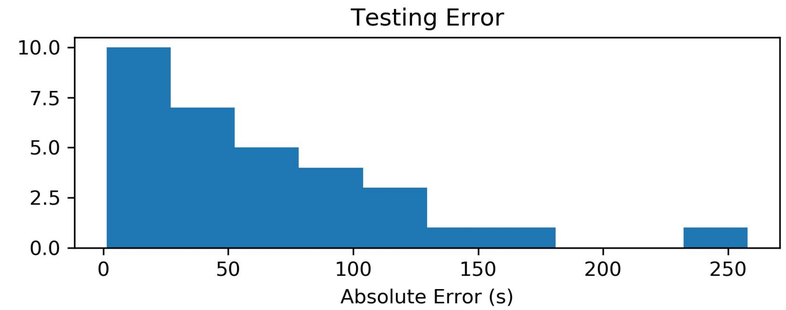

Histogram of Non-progressive Absolute Errors (Per-Script)
Linear Regression
Random Forest
Counts
Absolute Error (s)
0
2
4
6
8
10
0
50
100
150
200
250

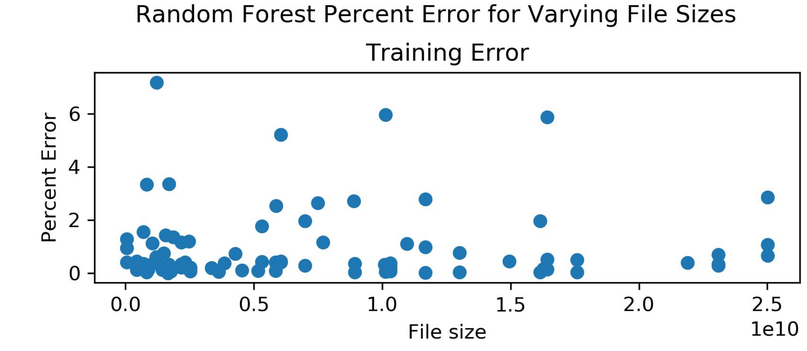

Histogram of Non-progressive Absolute Errors (Per-Job)
Linear Regression
Random Forest
Counts
Absolute Error (s)
0
1
2
3
4
5
6
7
0
50
100
150
200
250

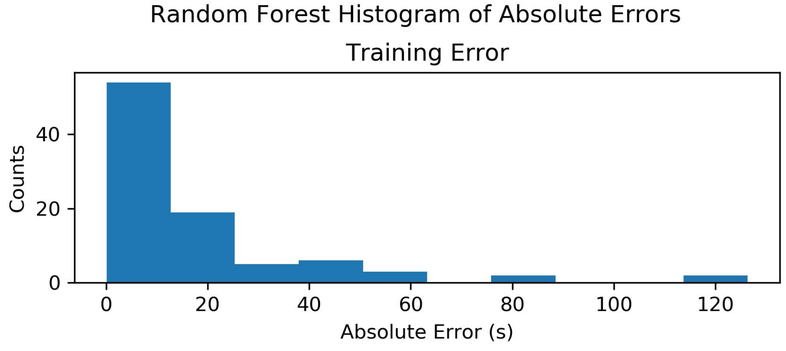

Linear Regression Percent Error for Varying File Sizes
Training Error
Percent Error
10.0
7.5
5.0
2.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
File size
1e10

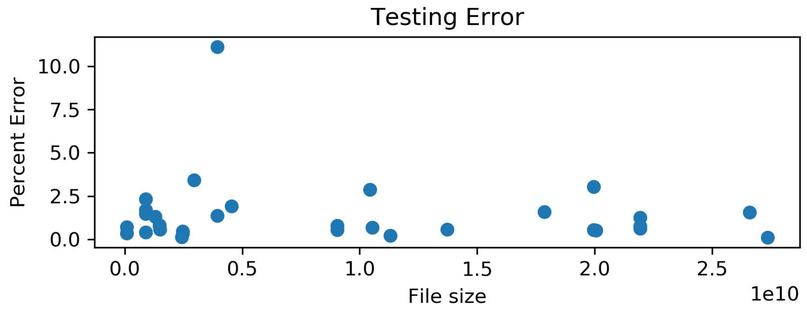

Linear Regression Histogram of Absolute Errors
Training Error
Counts
30
20
10
0
0
50
100
150
200
Absolute Error (s)

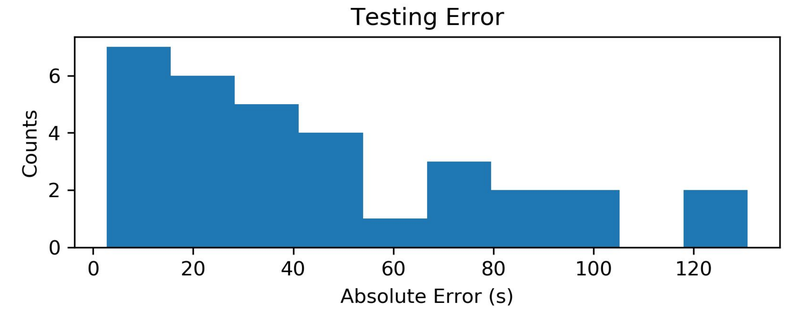

Testing Error
Percent Error
10
5
0
0.0
0.5
1.0
1.5
2.0
2.5
File size
1e10


Testing Error
10.0
7.5
5.0
2.5
0.0
0
50
100
150
200
250
Absolute Error (s)


Random Forest Percent Error for Varying File Sizes
Training Error
Percent Error
6
4
2
0
0.0
0.5
1.0
1.5
2.0
2.5
File size
1e10


Random Forest Histogram of Absolute Errors
Training Error
Counts
40
20
0
0
20
40
60
80
100
120
Absolute Error (s)


Testing Error
Percent Error
10.0
7.5
5.0
2.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
File size
1e10


Testing Error
Counts
6
4
2
0
0
20
40
60
80
100
120
Absolute Error (s)