Data Science Driven Retail

Differentiate and Win with Data Science in Hypercompetitive Retail Environment

W205 Project - 8/22/17

Maura Cullen,Bala Ganeshan, Jonah Smith, Mike Frazzini,

Differentiate and Win in Hypercompetitive Retail Environment with Data

Problem: Hyper Competitive Retail Environment

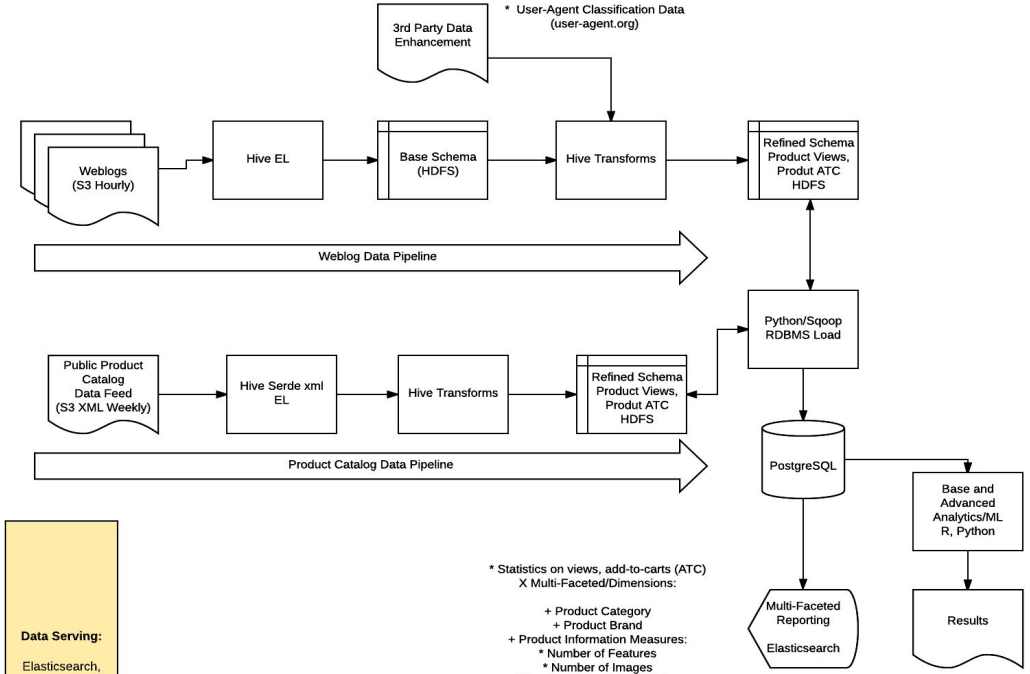
Assuming Competitive Pricing, and Competitive Product Assortment . . .

- Compete and Win Through Product Authority and Expertise
  - Auto-Detect Product Information Quality Opportunities
  - Leverage Data Science to Improve Product Information Quality
  - Apply Machine Learning for Advanced Product Attribution

Data Acquisition (ETL)

- Real eCommerce Site Raw Weblog Data
  - Filtered for Product Views and Cart Requests, PII Filtered and Cryptographically Hashed (Hive Sha256Hex).
  - Volume: 1 Week ~10GB, ~8M Records. Peak Hourly Volume: ~80MB, ~64K Records.
  - ETL: Cleansing and Regex Filtering and Transformations
- Real Third Party Weblog User-Agent Classifications (Spider, Bot signatures . . .). http://user-agents.org
  - Volume: 2,459 Records.
  - ETL: Left Outer Join with Weblog Data and Exclude Non-Human User-Agent Weblog Records.
- Real eCommerce Product Catalog XML Dump
  - Full dump containing 100’s of XML Lines for over 125K Products
  - Volume: ~1.2GB, ~32.5M XML Lines
  - ETL: Extensive XML Xpath Queries and Lateral Explodes to Create Product Catalog Schema with Aggregations

Solution Architecture



Data Ingest Solutions:

ETL, Sqoop

Query and Exploration:

SQL, Python, R, Elasticsearch

Data Cleaning:

Regex (Weblog Hive/Serde), XPath (XML)

Batch Processing Platforms:

M/R, Hive, SQL, Python

Data Definition:

Text Regex (Weblog Hive/Serde), XML, SQL DDL, Faceted Index (Elasticsearch)

Storage Systems:

Cloud (Amazon S3), HDFS, RDBMS

Data Serving:

Elasticsearch, Postgres SQL/ RDBMS

Compute Platforms:

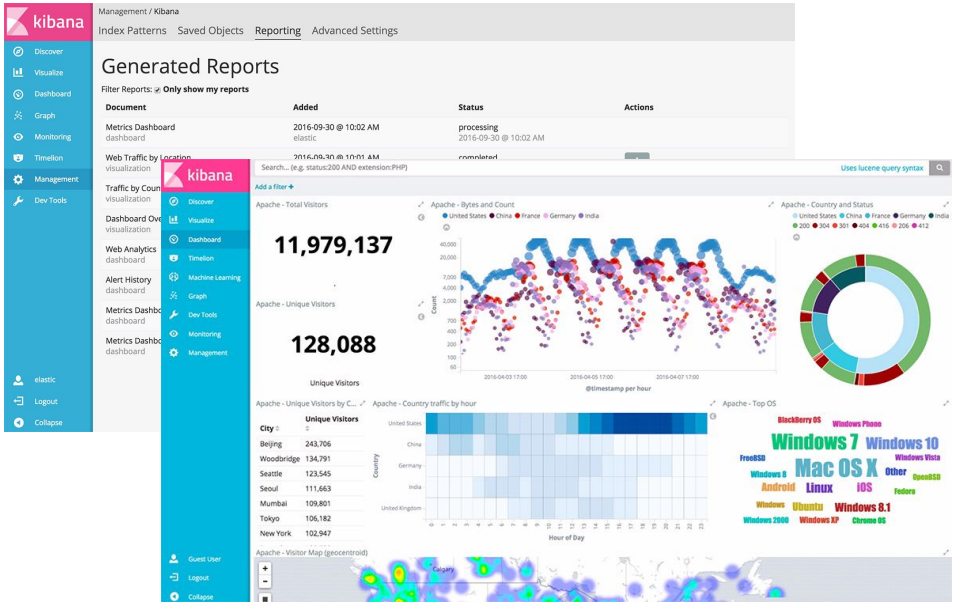
Psuedo Distributed Cloud (Amazon AWS), Scalable to Cluster

Implementation Details

- Based on the AMI provided by UC Berkeley (AMI ID: **ami-be0d5fd4**)
  - Provides Hadoop, Hive, Spark services
  - Python 2.6 and 2.7
  - Postgres and metstore made available through EBS volume created in “Lab 2”
- Patched OS version to apply latest patches that addresses security issues
- Reconfigured Hive to utilize Postgres DB (metastore) to store metadata about Hive schema
  - Created user “hiveuser” with appropriate authorizations to modify the database
- Installed Sqoop, asciidoc, xmlto and ant to compile and provide Sqoop services
- Ported some of the Python code to use Python version 2.6
- Created a database to store data processed by Hive in a relational table
  - Sqoop run as user “hiveuser” to load processed data
- Installed and configured “Elastic Search” database to utilize data from Postgres DB for analysis and visualization.
- Code and instructions to enable data pipeline available in the GIT repository

Serving Layer

- Elastic Search
  - Faceted search technology
  - Distributed approach
  - Open source
- Kibana
  - Create templated reporting
  - Simple web interface

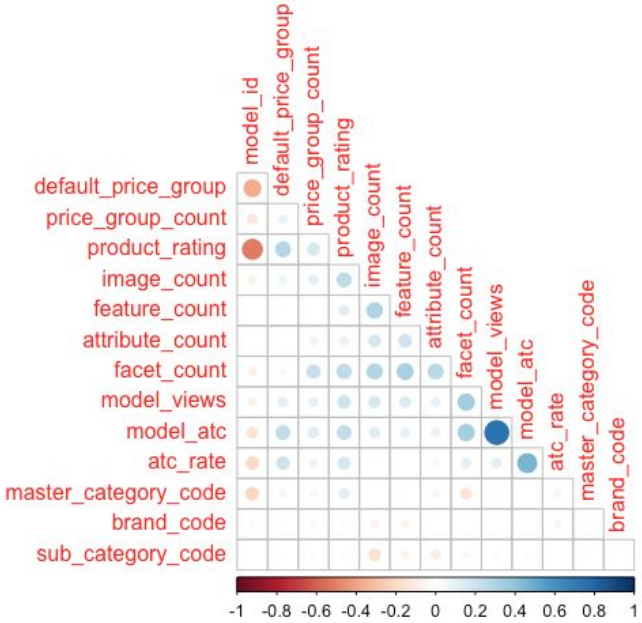


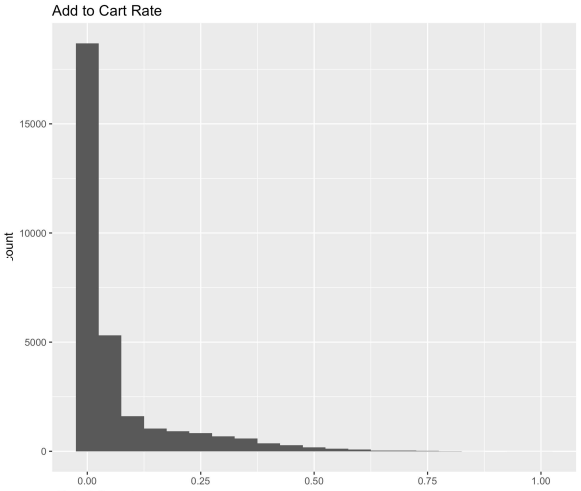
Statistical Results

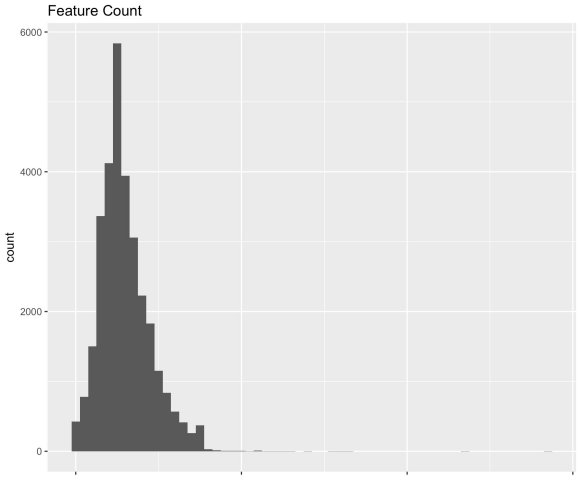
- Dataset overview:
  - **30,793** Product Models - All active products on eBags
  - **15** Master Product Categories such as Luggage, Travel Accessories, Technology, Handbags, Apparel, etc.
  - **115** Sub Product Categories
  - **932** Unique Product Brands
- Training Set
  - **21,548** product examples
  - **14** possible base predictors
- Test Set
  - **9,245** product examples

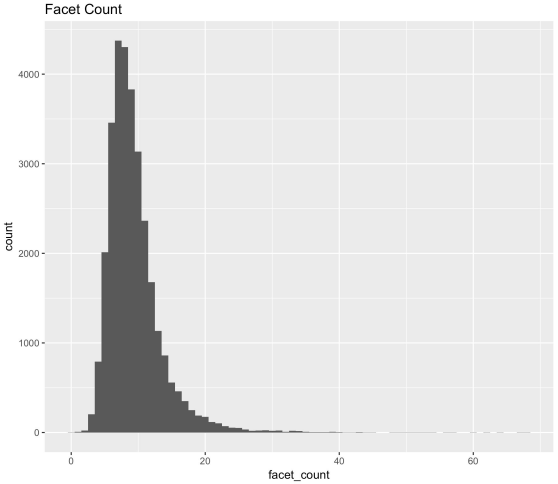
Statistical Results

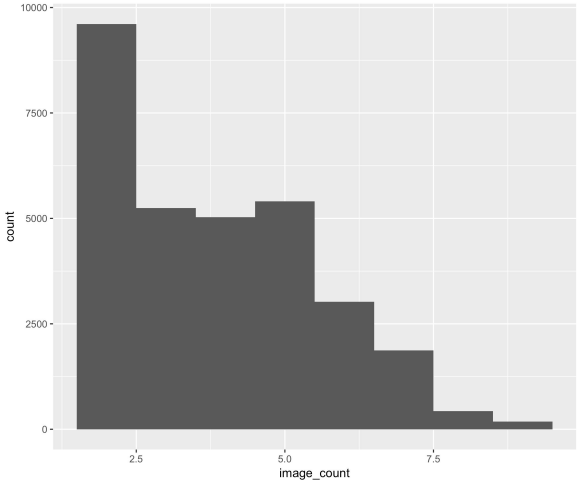
Correlation Matrix of Features





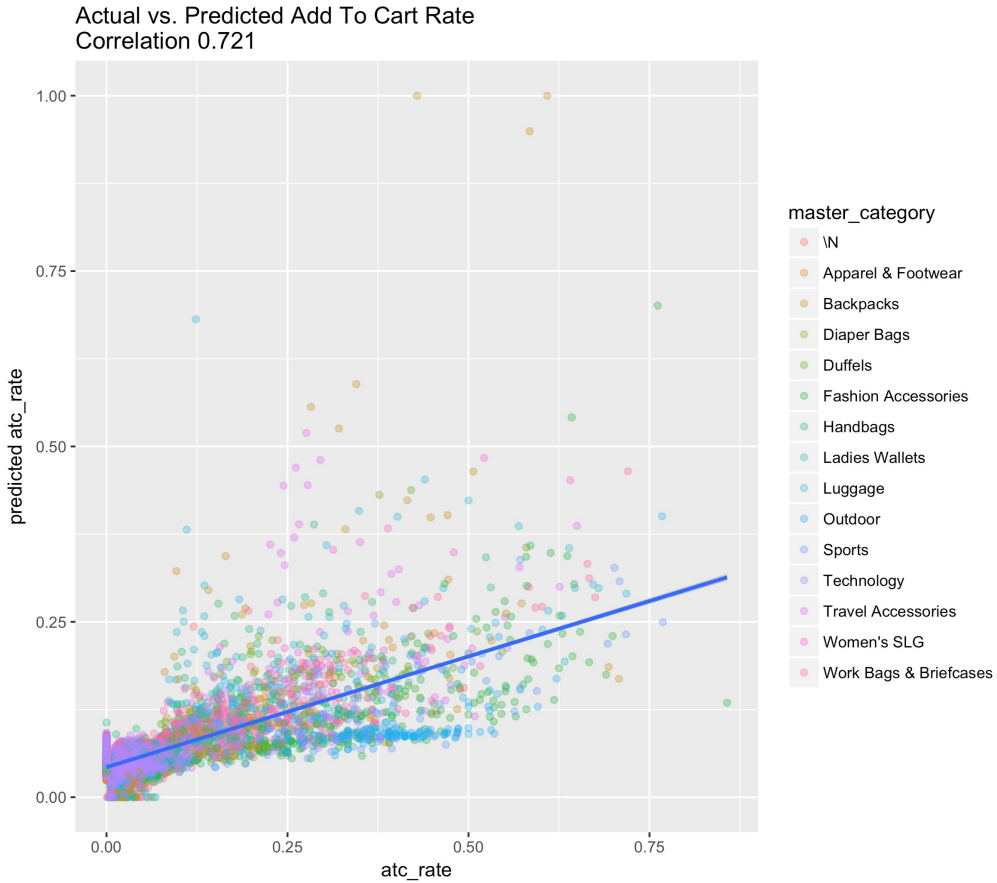






Statistical Results

- Predicting the “Add to Cart Rate” for each product based on product and weblog data features
- Fit linear regression model using a backward search method for feature selection
- 70/30 Train/Test set. Data points randomly selected within product category
- Product Features used for linear fit:
  - Product Brand
  - Product Rating
  - Master Category of Product
  - Number of Model Views
  - Facet Count
  - Number of Features in Product Description
  - Model ATC



Multiple R-squared: 0.3451, Adjusted R-squared: 0.3449

Project Outcome

- Data pipelines to support hourly performance statistics of product demand
  - Visitor based Product Views / Product Cart Adds
  - Optimizes focus and resource allocation for product information and attribution improvements weighted by demand and opportunity
- Statistical model for optimal product information and attributes
  - Optimal number of product images = 6+
  - Optimal number of product features = 6 to 10
  - Statistical evidence that increased facets drive increased product views
- Framework for Machine Learning Advanced Analytics
  - Product Information Score and Prediction of Improved Product Information and Attribution

Complexity, Storage Needs and Limitations

- UCB provided image is woefully outdated and hard to work with
  - Constraint of “instructor able to run code” forced us to use a standard UCB image
- Old image (CentOS 6.x, Python 2.6 etc.) implies lack of newer ETL and EDA tools and packages
- The data processing model is appropriate for Hive (batch processing)
  - However requires scale out implementation to process data in a reasonable time (constrained by above)
- Product catalog is a HUGE XML file (> 30M lines): takes a while to process and is cumbersome
  - Incremental updates to the product catalog will potential require processing of all records
- Hourly web log files are approximately 80MB and have to be processed in a reasonable time
  - “Cyber Weekend” volume typically 5X normal average
  - Expected to grow as the customer base increases with Samsonite acquisition and new partnerships
- Hive code developed processes drops existing tables and processes all files available
  - Need to update SQL scripts to process incrementally
  - Sqoop code can process data incrementally but was not tested
- Presentation layer requires further work to allow ad hoc analysis and visualization
- Integration Challenges - Each team member worked on isolated pieces and final integration rough