

Bills of Material Replace Material Recommendation System

Ayush Soni (aas427), Harshit Manchanda (hm545),
Mukul Shukla (ms3528), Rohith MD (rm977), Shubham Agrawal (sa2279),
Shubham Khandelwal (sk3266), Vidush Vishwanath (vv259)

Agenda :

- 1 Database and Data discussion
- 2 Feature encoding
- 3 Baseline model
- 4 Next ML model
- 5 Application MVC architecture
- 6 Static and Dynamic Verification
- 7 Next Steps and Timelines

Database:

- Schema created in MySQL
- Data imported to MySQL, following the SAP key constraints
- Primary tables as per Proprietary database – MARA, MAST,STKO,STPO,CDPOS, MISC
- The relations are maintained between the tables.
 - MARA --- MAST
 - MAST --- STKO
 - STKO --- STPO
 - STPO --- MISC
 - MARA--- CDPOS
 - STKO --- STPO

Data:

- Sample data rows were created together with the client.
- The rest of the rows for all the tables were created in accordance with the proprietary tables and compliant with the initial rows as provided.
- The rows were combined to prepare an intermediate table as required by the ML model.

MISC Table:

- This table is created as a combination of different tables as used by the client.
- This table contains information of items being used in different tables related to MRP , STORAGE LOCATION, MATERIAL GROUP etc.
- Since all the relative data is distributed among multiple tables which involve approximately joining ~20 tables under different LOB's, the design required to create a combination of all the values which contribute in the determination of best replacement item.

Recommendation Models

Feature Encoding

A number of features were identified using the MISC table for each BOM:

- Plant Location
- MRP Group
- Purchasing Group
- Count of Replacement
- Unit of Measure
- Volume
- Cost
- Scheduling
- Lot Size
- Storage Location

Feature Encoding

The feature objects are encoded using appropriate hash function for creating encodings of each feature of the object passed.

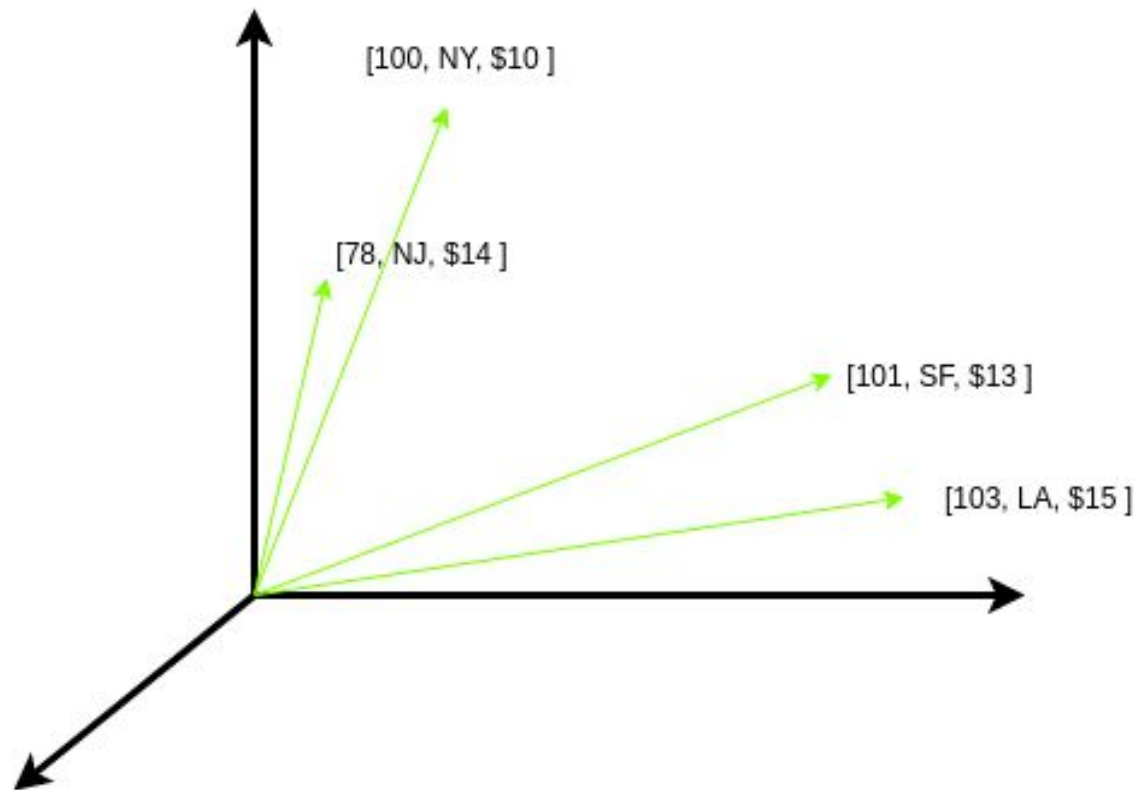
We tried two approaches for hashing:

- Hash based approach with feature count modulus for each feature
- Word2Vec models for word vector formation (word string encoding) and absolute difference for numeric entities.

We tried running different similarity computation techniques using these approaches.

Cosine Similarity

- We use the concept of cosine similarity.
- The encoded features are used to represent a vector for each BOM object.
- These features are then provided as input to the similarity function for getting the scores.



Output of the models

As mentioned we used two approaches for feature encoding and evaluated the model for both of them. The results observed are discussed below:

```
[Shubhams-MBP:machine_learning shubham$ python cosine_
ITEM to be replaced: PLATINUM AG3 BATTERY
ITEM                SCORES
AUTOMOTIVE BATTERY  0.922689071731669
EVERSTART LEAD ACID BAT  0.7753529327459576
DEKA 410 AMP MOWER BAT  0.7734385152048789
OPTIMA RED TOP BATTERY  0.7427720751557592
DURALAST            0.6939810229273553
BMW M3-75 STARTING BAT  0.6802292790180758
TOP DEEP CYCLE BATTERY  0.5506562749445612
NORTHSTAR           0.5465285453779151
BMW_FDZ BATTERY      0.5045687128075965
```

Similarity Scores: Word2Vec based features

```
Shubhams-MBP:machine_learning shubham$ python cosine_
ITEM to be replaced: PLATINUM AG3 BATTERY
ITEM                SCORES
AUTOMOTIVE BATTERY  0.8766351635601365
DURALAST            0.750704359745609
EVERSTART LEAD ACID BAT  0.7242811860050686
BMW M3-75 STARTING BAT  0.6969393864590663
OPTIMA RED TOP BATTERY  0.6730585169304361
NORTHSTAR           0.6547043622003653
DEKA 410 AMP MOWER BAT  0.6471334369693766
TOP DEEP CYCLE BATTERY  0.6331062392832385
BMW_FDZ BATTERY      0.4483722150329731
```

Similarity Scores: Hash based features

Weighted Similarity Computations

- We also explored about an approach for similarity computations on the basis of weights associated with each feature.
- Currently we use a rule-based weight computation technique, with more important features getting more weights.
- Distribution of weights followed importance levels. This helps for use case specific knowledge to be included.
- For incorporating based intuitions for feature importance in our machine learning models, the industry experts can give their feedback for the rules and we can upgrade our rule based model for better efficiencies.

Results:

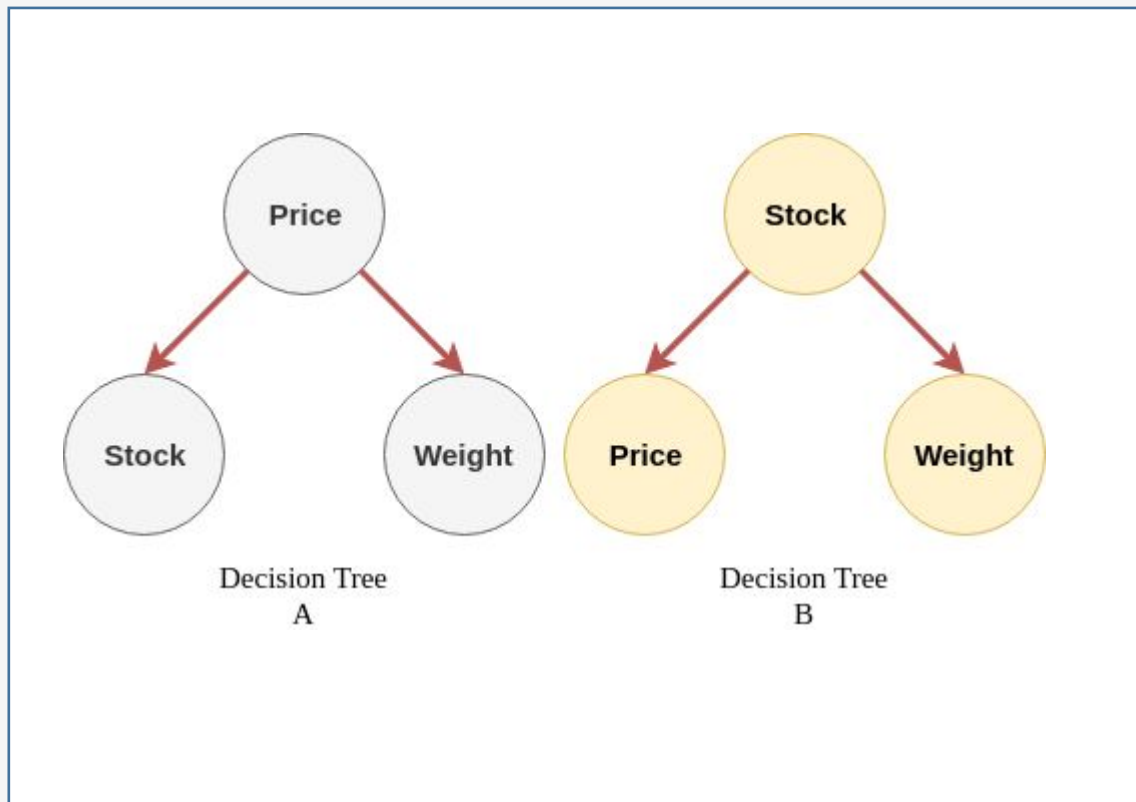
We evaluated our models on 10 different material groups and found the following results:

Average Accuracy Scores:

- Cosine Similarity (Hash based approach): 0.68978
- Cosine Similarity (Word2vec approach): 0.76473
- Weighted Similarity Computations: 0.82371

Next Steps: Random Forest

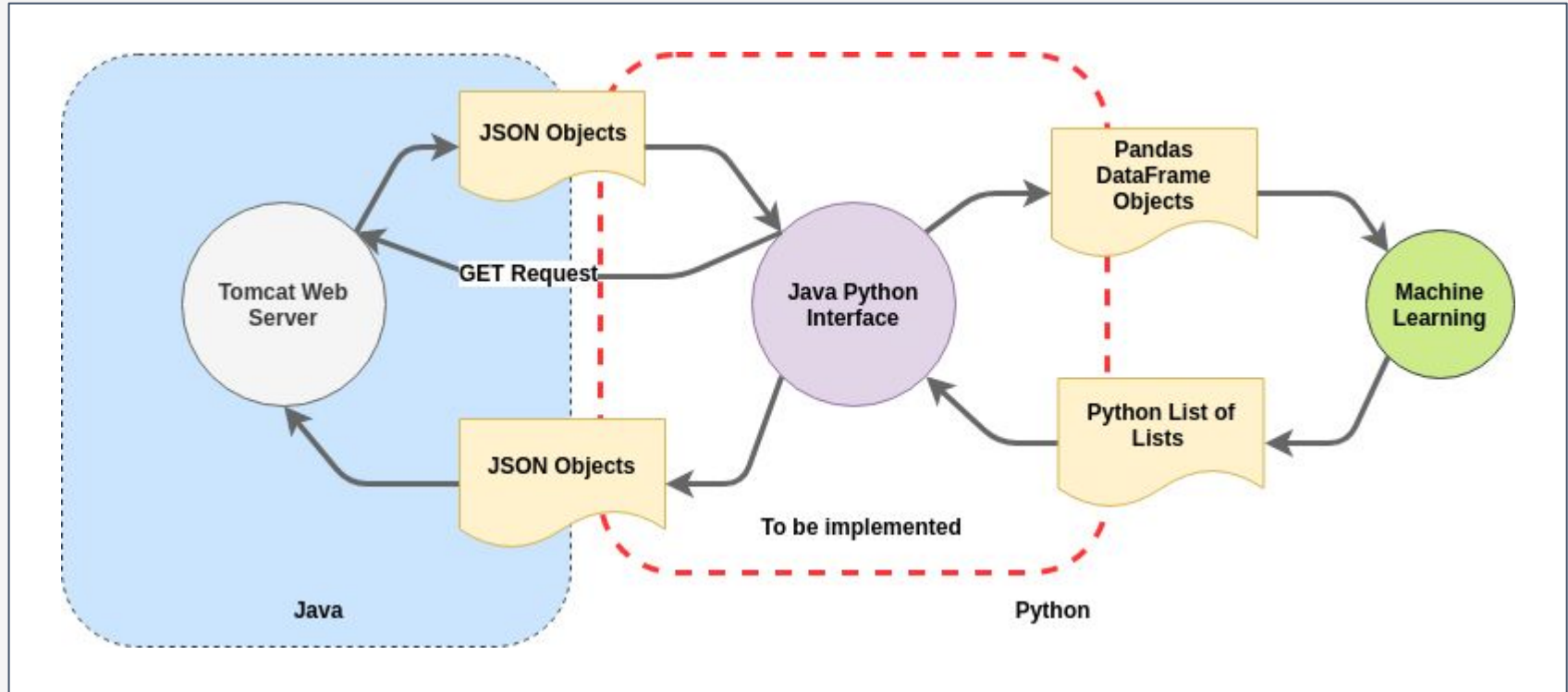
- 1 Sampling BOM data
- 2 Training models
- 3 Bootstrap
Aggregating



Motivation of Random Forest

- 1 Weights for features
- 2 Hyperparameter Tuning

Next Steps: Java Python Interface



Database/Repository/JPA : Work so far ...

- Data imported to MySQL, following the SAP key constraints
- Added dependencies for persistence, JPA and MySQL connector, Hibernate
- Implemented ORM POJOs

Service / Controller : Work so far ...

- Using basic repository methods, implemented sample service layer methods. (eg getMiscForCDPOS)
- Service layer implementation, interface separated, so that its extendible.
- We use Autowiring to enforce single instances of repository, service layer objects.
- Exposed sample API to test basic CRUD operations

Sample GET response (via PostMan)

dd

GET

localhost:8080/api/CDPOS

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth

Body

Cookies

Headers (3)

Test Results

Pretty

Raw

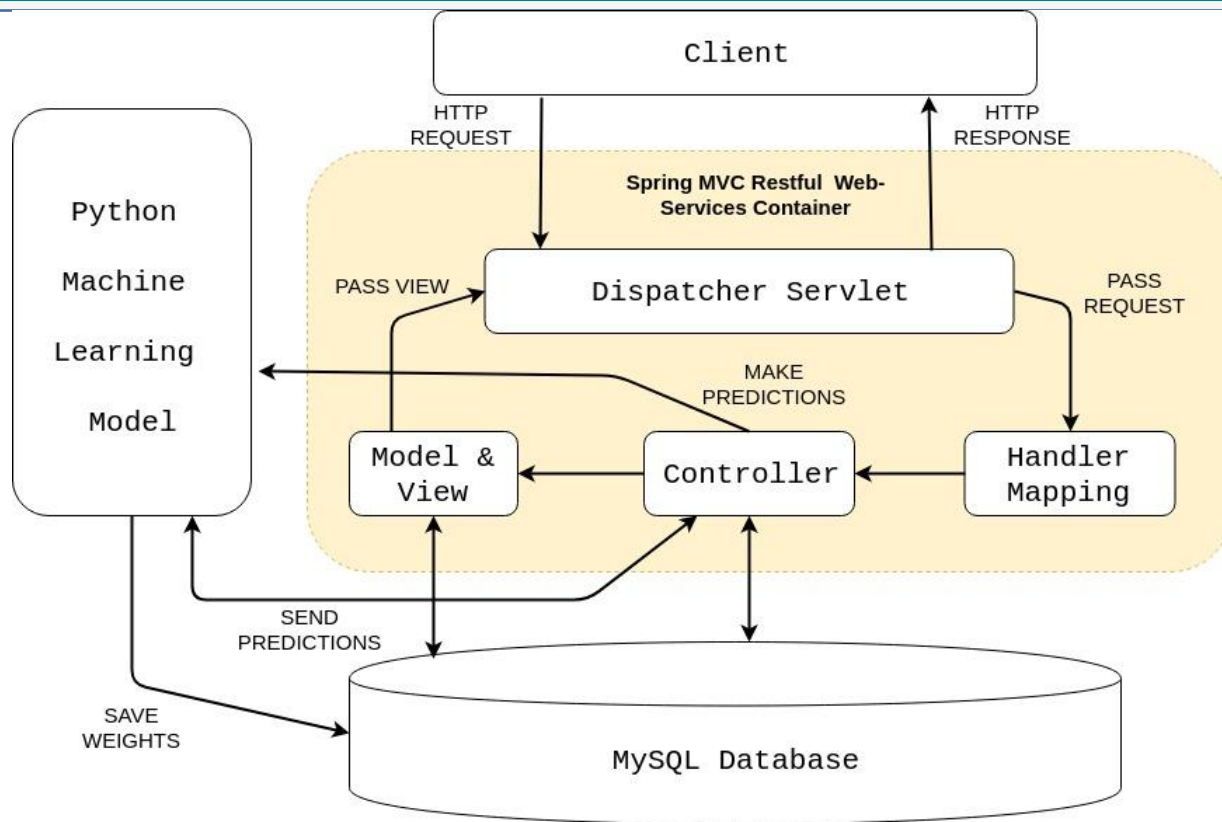
Preview

JSON



```
1  [
2  {
3    "OLD_MATNR": "AUTOMOTIVE BATTERY",
4    "OLD_UOM": "1",
5    "VALUE_OLD": "0",
6    "NEW_MATNR": "PLATINUM AG3 BATTERY",
7    "NEW_UOM": "1",
8    "VALUE_NEW": "0",
9    "AENNR": "\r"
10  },
11  {
12    "OLD_MATNR": "PLATINUM AG3 BATTERY",
13    "OLD_UOM": "1",
14    "VALUE_OLD": "0",
15    "NEW_MATNR": "EVERSTART LEAD ACID BAT",
16    "NEW_UOM": "2",
17    "VALUE_NEW": "0",
18    "AENNR": "EVER_PLAT\r"
19  },
20  {
21    "OLD_MATNR": "AUTOMOTIVE BATTERY",
22    "OLD_UOM": "1",
23    "VALUE_OLD": "0",
24    "NEW_MATNR": "PLATINUM AG3 BATTERY",
25    "NEW_UOM": "1",
```

Overall Pipeline



Static Verification

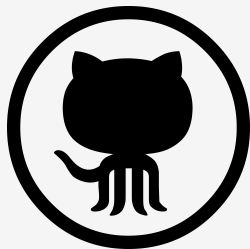
- Pair design and coding:
 - Database: Mukul and Rohith
 - Controllers: Harshit and Mukul
 - ML: Ayush, Shubham
 - Feature Engineering: Shubham, Vidush
 - Unit Testing and CI: Mukul, Vidush
- Code reviews for each addition
 - All PRs are reviewed by individual not committing

GitHub branch history:

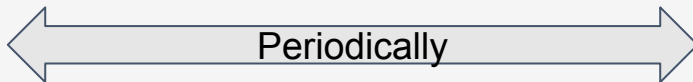
Next Steps – Development

- Use Random forests as ML Models
 - Evaluate on Data
- Create more data
 - Better model training
 - Reveal if the model works or not
 - For different use cases
- Unit testing
- Integration Testing
- For cogging: Python -> Java
 - Using intermediate results in data files
 - Want to use REST Base APIs

Deployment and Continuous Integration



 **GitHub**



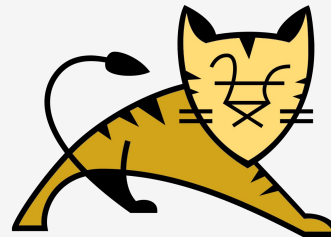
Maven™



Automation
Test Suite



Jenkins




Apache Tomcat


How it currently works - Continuous Integration


- Source code is hosted in GitHub.
- Jenkins is hosted on Azure
- We can tunnel into Azure VM to view repository health.
- Jenkins is not exposed via public IP. Github hooks not configured as GitHub cannot reach Jenkins via tunnel
- Jenkins runs maven tests and ML tests daily.


Jenkins screenshot


 **Jenkins**



Jenkins > BomRecommendation >


 [Back to Dashboard](#)






 [Status](#)



 [Changes](#)

 [GitHub](#)

 **Build History** [trend](#) 




 #5	28 Mar, 2019 6:54 PM
 #4	28 Mar, 2019 6:53 PM
 #3	28 Mar, 2019 6:50 PM
 #2	28 Mar, 2019 6:50 PM
 #1	28 Mar, 2019 6:47 PM

 [RSS for all](#)  [RSS for failures](#)

Project BomRecommendation

BomRecomCompilation

 [Recent Changes](#)

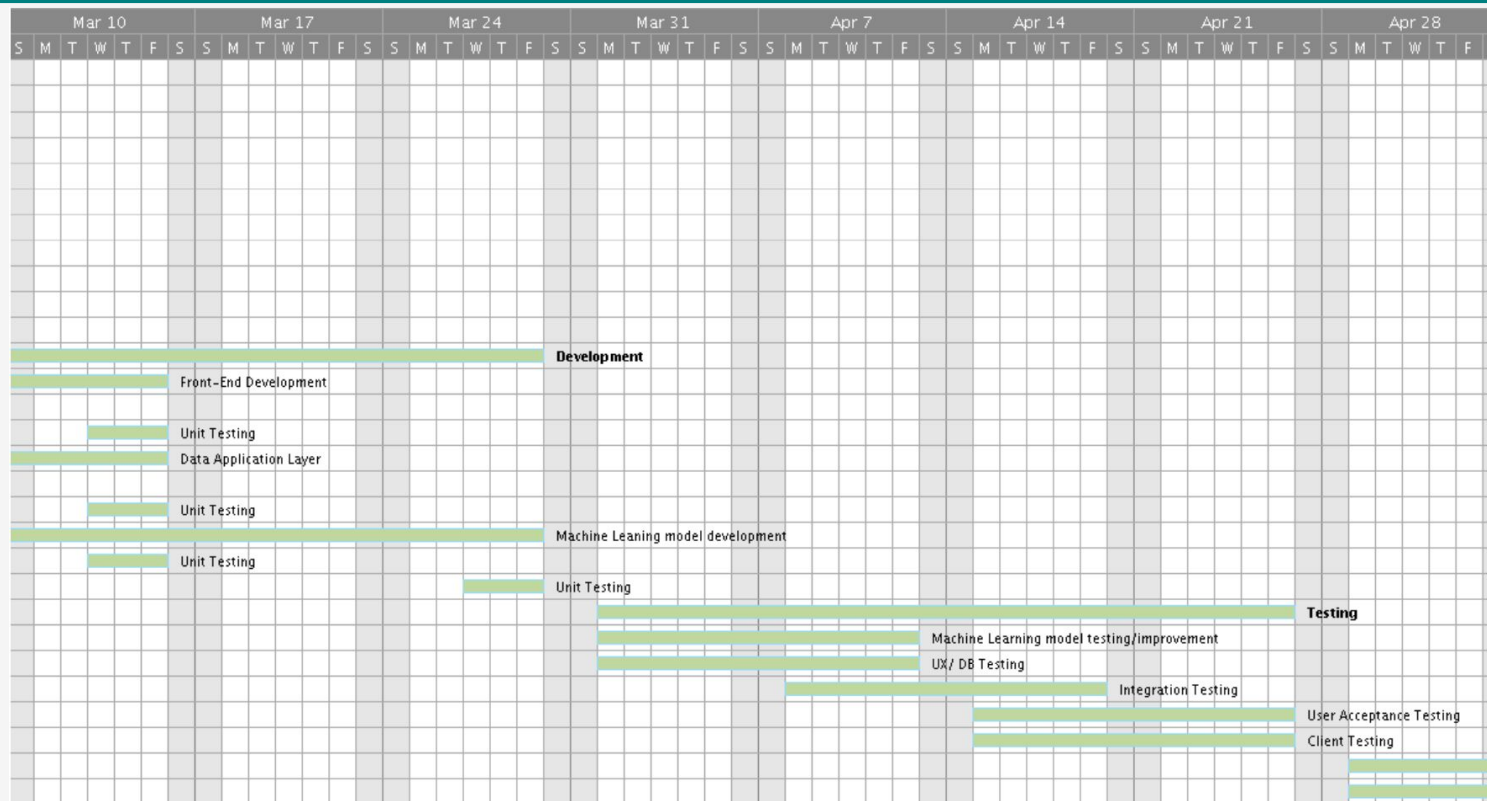
Permalinks

- [Last build \(#5\), 1 hr 38 min ago](#)
- [Last stable build \(#3\), 1 hr 41 min ago](#)
- [Last successful build \(#3\), 1 hr 41 min ago](#)
- [Last failed build \(#5\), 1 hr 38 min ago](#)
- [Last unsuccessful build \(#5\), 1 hr 38 min ago](#)
- [Last completed build \(#5\), 1 hr 38 min ago](#)

Dynamic Verification: Unit Testing

- Separate manual tests for ML in python
- Spring Boot Java Unit Test
- Java Unit Tests integrated with Maven
- All tests run via Jenkins periodically or on demand

Timeline



Work Completed

- Database Creation
- Data population
- Hibernate / JPA setup
- Basic CRUD controller
- Repository and Service layer
- ML Baseline Model (cosine similarity)
- Currently working on Random Forests

Questions?

?

Thank You

Clients: Abhijnan Saha, Rajiv Kumar
Professor: William Y Arms
Teaching Assistant: Alison Cooper