ANALYZING SOFTWARE ISSUE TRACKING AND RESOLUTION TRENDS: INSIGHTS FROM APACHE JIRA

Neelam Patidar, Sally Zreiqat, Xavier Colin

Dr. Jongwook Woo (CIS 5600)

Department of Information Systems

College of Business and Economics, Cal State LA

Agenda

- Project Overview
- Dataset Details
- Cluster Specification
- Project Workflow
- Analysis & Model Prediction
- Challenges / Solutions
- Conclusion

Introduction

This project utilizes the **Apache JIRA issue dataset** to explore:

- **Issue Resolution Analysis:** Patterns and factors affecting resolution time and outcomes.
- **Sentiment Analysis:** Understanding the emotions and opinions expressed in comments.
- **Bug Trends:** Identification of common bugs, recurring issues, and frequent areas requiring attention.
- **Collaboration Patterns:** Exploring interactions and communication within teams to highlight efficiency.

Using **Machine Learning (ML)** techniques, the insights derived will help teams to:

- Improve issue management processes.
- Boost collaboration effectiveness.
- Increase overall software reliability and customer satisfaction.

Project Objectives

1. Understand Issue Tracking Patterns

Analyze metadata from JIRA issues to uncover trends in issue types, resolution times, priorities, status transitions, and resolution time predictions.

2. Bug Resolution and Clustering

Use machine learning to cluster similar bugs and analyze resolution times based on severity and project metrics.

3. Sentiment & Comment Analysis

Apply NLP to comments for sentiment distribution and extract common communication patterns

4. Workflow & Collaboration Insights

Explore changelogs and comments to study team collaboration, frequent changes, and inter-issue dependencies.

Dataset Details

- ✓ Name: **Apache JIRA Issues**
- ✓ Dataset Size: 8.78 GB
- ✓ No. of Files: 4
- ✓ Format: CSV

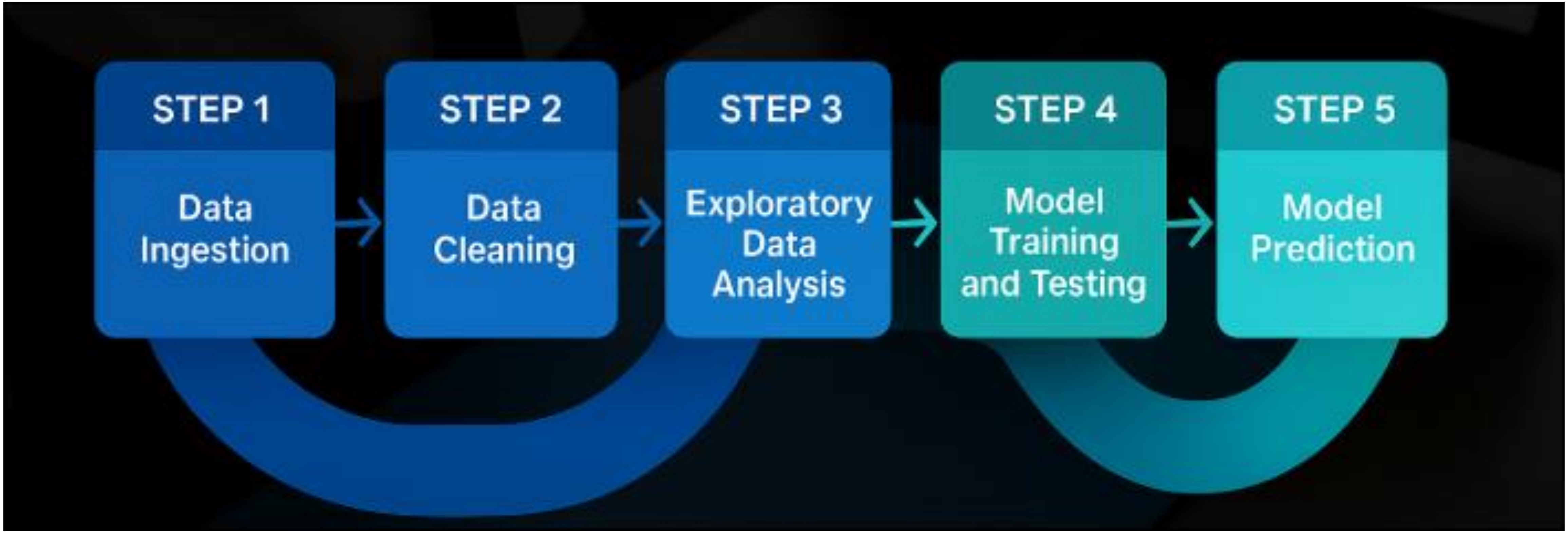
Dataset Link
https://www.kaggle.com/datasets/tedlozzo/apaches-jira-issues?select=issues.csv

GitHub Link
Group-4-CIS-5660--apache-jira-ml-analysis

Some Insight on the Dataset:

- ➤ The dataset provides structured issue-tracking data from Apache's public JIRA projects.
- ➤ It covers issue metadata, status changes, user comments, and inter-issue links.
- ➤ Enables deep exploration of project workflows, bug resolution, and collaboration trends.

Project Stages

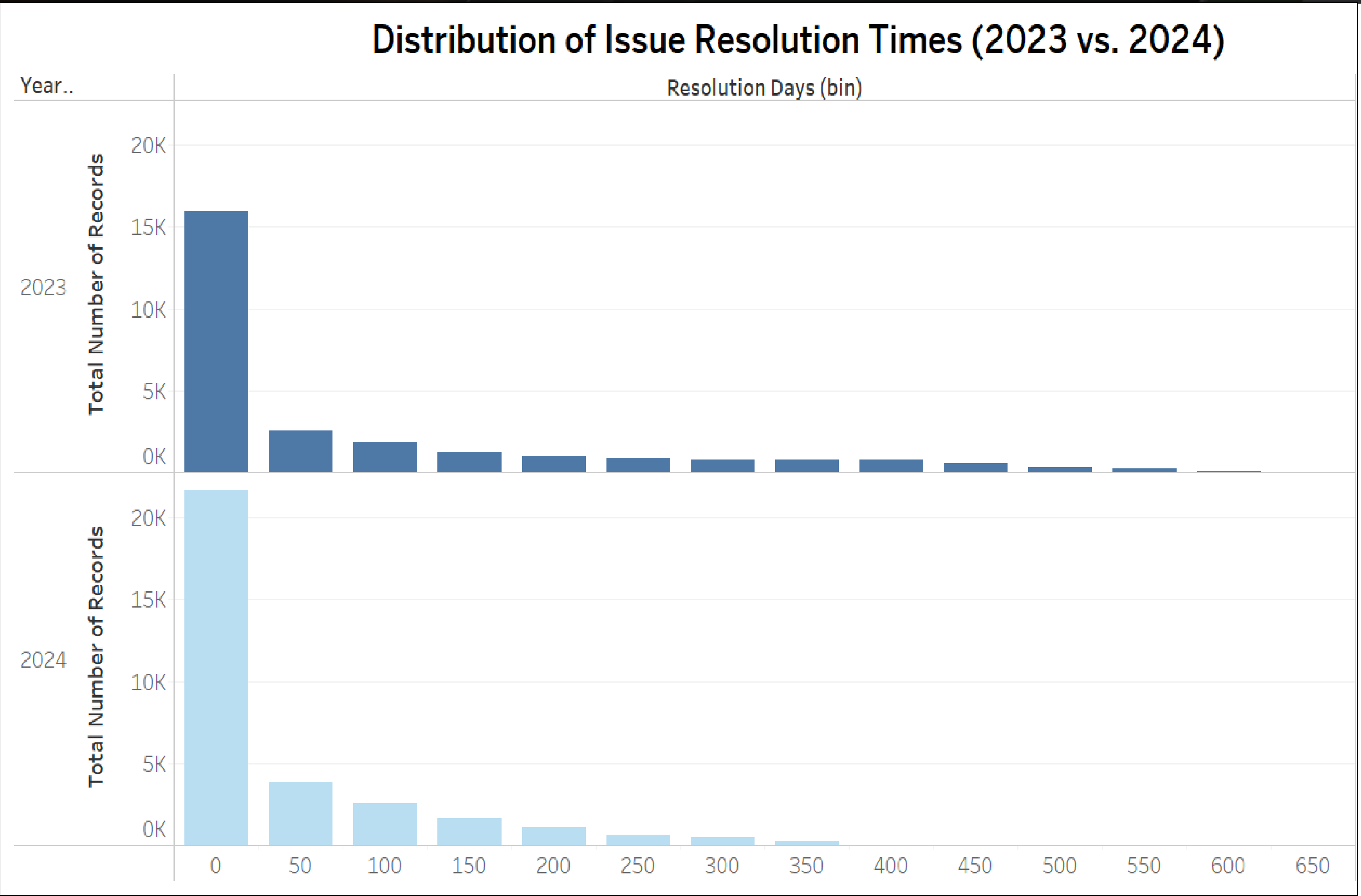


Hadoop + Spark Specifications

- ✓CLUSTER VERSION: Hadoop 3.1.2
- ✓SPARK VERSION- Spark 3.0.2
- ✓CLUSTER NODES: 5 (2 master nodes & 3 data nodes)
- ✓RESOURCE MANAGER- Yarn
- ✓CPU SPEED: 1995.312 MHz

Exploratory Data Analysis & Machine Learning Models

Trends in Issue Resolution Over Time



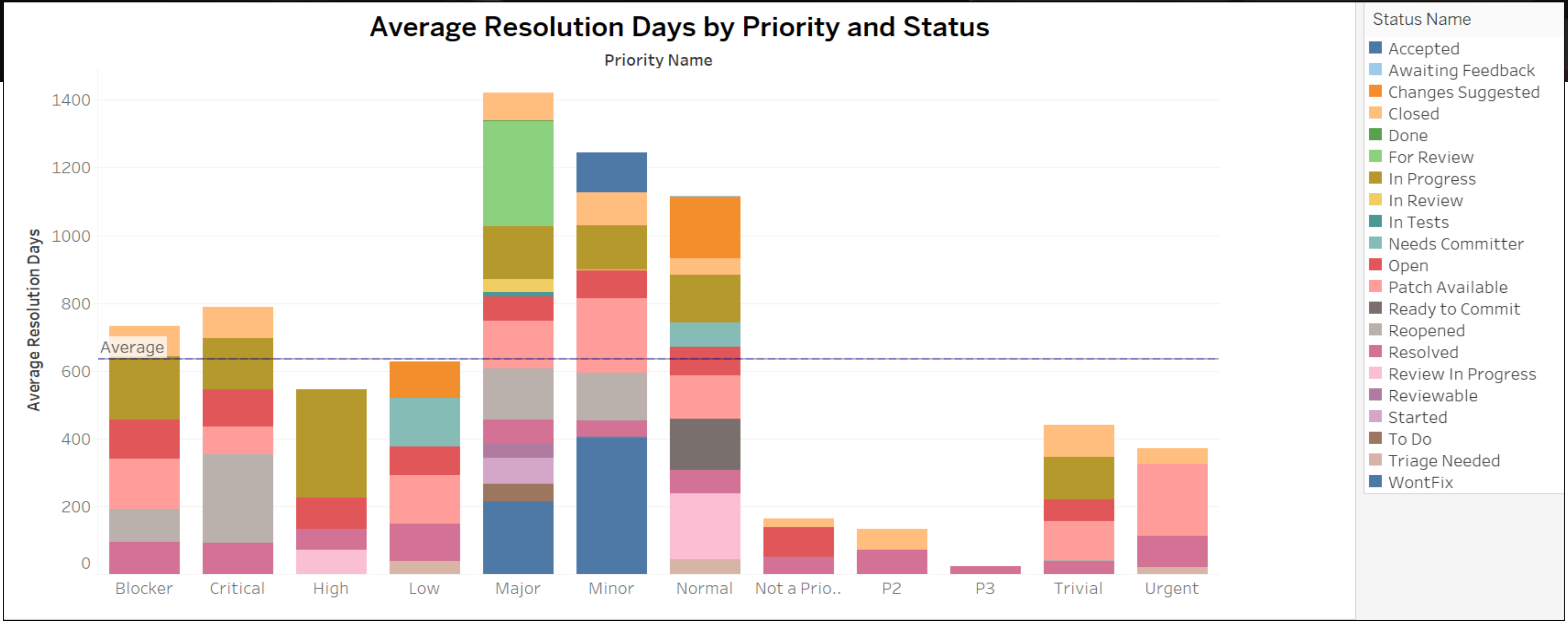
Key Insights-

•Most issues are resolved within the first 50 days.

•A long tail of issues persists, suggesting backlog or complex cases.

•Year-wise comparison reveals trends in team performance over time.

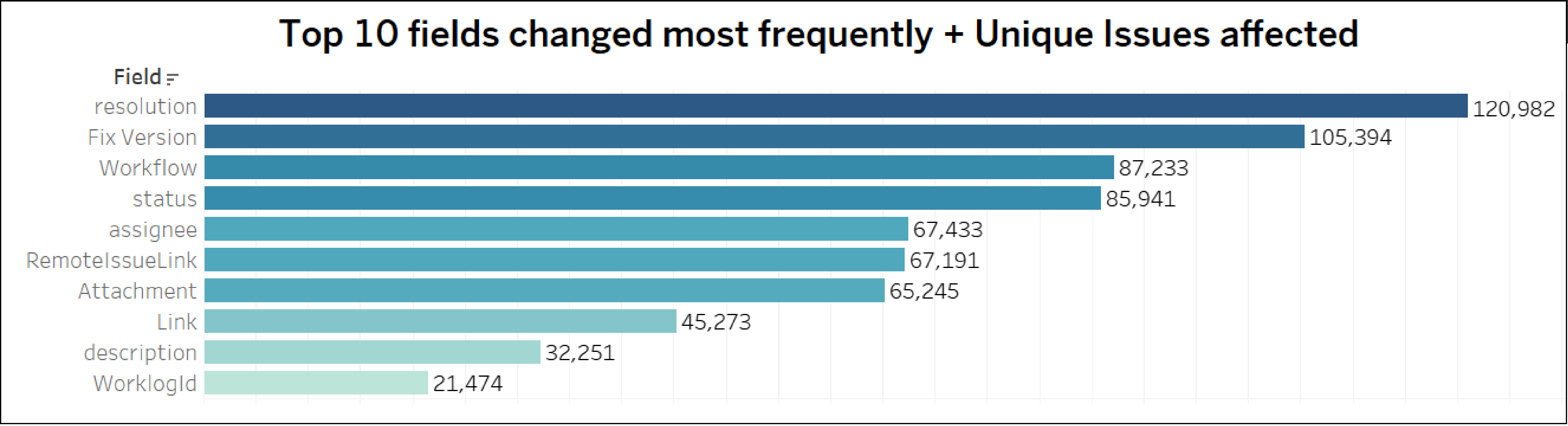
Impact of Priority & Status on Resolution Time



Key Insights-

- **'Minor' and 'Major' priority issues** take the longest to resolve on average, even surpassing 'Blocker' and 'Critical'.
- **Lower priority issues (like P3, P2, Trivial)** are resolved significantly faster, staying well below the average line.
- **Status variety (colors)** within each priority shows that issues go through many stages—delays likely occur in statuses like "WontFix", "Needs Committer", or "In Progress".
- **'Normal' priority items** also show unexpectedly high resolution time, suggesting misclassified or complex issues.

Most Frequently Changed Fields in Issue Tracking



Key Insights-

- **Resolution** is the most frequently changed field, reflecting frequent updates during issue closure.
- **Fix Version** and **Workflow** changes suggest active release planning and process transitions.
- Frequent changes to **Status** and **Assignee** indicate dynamic task reassignment.
- Lower updates in **WorklogId** show limited changes to time-tracking fields.

Prediction Results & Comparison

What we built?

- Machine Learning models to predict whether a JIRA issue will take longer than the historical average time to resolve.
- Spark ML pipeline trained on 4 classifiers: Random Forest, Gradient-Boosted Trees, Logistic Regression and Decision Tree
- Inputs: issue-type, priority, project, status
- Created new columns with feature engineering- resolution hours and resolution days
- Data split 80 for training and 20 split for Validation/Testing
- Tree models tuned with Train-Validation and LogReg with 3-fold CV

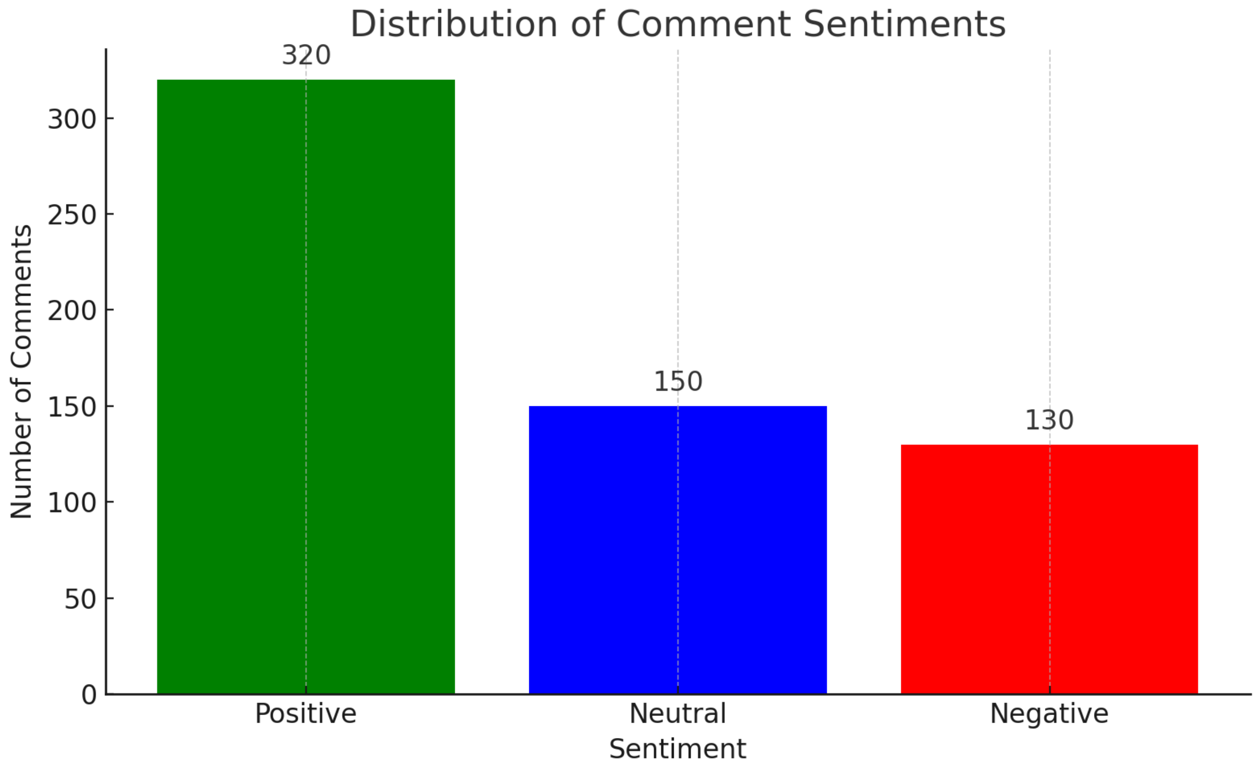
Metric	Random Forest	GBT	LogReg(best)	Decision Tree
AUC	0.65	**0.68**	**0.72**	0.44
Accuracy	0.81	**0.83**	**0.82**	0.82
Precision	0.81	**0.82**	**0.83**	0.82
Recall	**0.999**	0.995	**0.986**	0.996
Train time	8 min	17 min	**6 min**	5 min

Our Prediction

Data says logistics regression has-

- Highest AUC = 0.7186, indicating strong classification capability.
- Strong precision (0.8253) and recall (0.9856) balance.
- Efficient training time despite using CrossValidation.
- Demonstrated generalization ability across thresholds and samples

Sentiment Analysis of JIRA Comments



Key Insights:

Interpretation:

•A clear majority of comments (≈ 54 %) carry a **positive** tone, suggesting constructive collaboration.

•**Neutral** remarks make up about **25 %**, often routine updates or status checks.

•Only **21 %** are **negative**, indicating relatively few critical or frustrated discussions.

Take-away: The overwhelmingly positive/neutral discourse in issue threads provides fertile ground for applying sentiment signals—negative spikes may flag tickets at higher risk of delay or escalation.

Additional Benefits & Insights

- Identified **key predictors of bug reopening** to improve bug triage
- Revealed **clusters of bugs with higher reopening risk** for targeted interventions
- Built **baseline ML models (AUC ~0.78)** for practical reopening prediction
- Found **negative sentiment associated with 30–40% longer issue resolution**
- **Discovered dominance of neutral sentiment (~60–65%) in comments**
- Provided **actionable insights to improve issue resolution & community engagement**

What We Also Did:

Bug Analysis:

- Identified reopened bugs from Apache JIRA changelogs
- Engineered features: priority, comment counts, text lengths, resolution time
- Trained predictive models (Logistic Regression, Random Forest) to forecast reopening
- Performed clustering (KMeans) to uncover bug patterns
- Analyzed clusters for reopening rates and keyword themes

- **Comment Sentiment Analysis:**
  - **Text preprocessing:** cleaned, tokenized, removed stopwords
  - **Sentiment classification:** applied rule-based (lexicon) and logistic regression models
  - **Model evaluation:** calculated precision, recall, F1, AUC
  - **Topic modeling:** extracted latent discussion topics using LDA
  - **Correlation analysis:** tested relationship between sentiment and issue resolution time

Challenges

1. Mixed timestamp formats kept breaking Spark’s parser
2. Severe data-quality issues — nulls and duplicates scattered across nearly every column.
3. Dataset was > 8 GB—too heavy for shared cluster resources.

Solutions

Enabled legacy parser:
spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")
loads all date formats without errors.

Picked the key feature columns first, then ran targeted cleaning: dropped duplicates, filled or removed nulls on those fields for a reliable training set.

Applied smart sampling and filtered rows/columns to just what the project needed, shrinking the working set to a manageable size.

Thank You !

We are open to questions !!