# ASDS 6303 – Data Mining with Information Visualization

## Predictive Analytics for Incident Resolution Time

GROUP – 4

University of Texas at Arlington
Applied Statistics and Data Science

**Presented by:**
Trupti Shailendra Gandhi (1002202261)
Rushika Badri Prasad (1002235923)
Girish Sunkadakatte Chandrappa (1002212727)

# INDEX

## I. Introduction

- In modern IT Service Management (ITSM), the timely resolution of incidents is critical to maintaining service quality, client satisfaction, and compliance with Service Level Agreements (SLAs).
- Failure to meet SLAs can result in financial penalties, operational disruptions, and diminished client trust. Predicting incident resolution time is a complex challenge due to the dynamic and multi-faceted nature of service requests.
- The incident management process involves identifying, logging, categorizing, prioritizing, responding to, and resolving incidents to minimize their impact on business operations. Efficient execution of this process reduces downtime and ensures that IT issues are addressed swiftly and effectively.

## II. Problem Statement

- Service Level Agreements (SLAs) are critical components of IT Service Management, as they define the expected timeframes for resolving incidents based on urgency, impact, and priority.
- Failure to meet these SLAs can lead to financial penalties and erode client trust. However, accurately predicting resolution time is challenging due to the diverse, high-dimensional, and dynamic nature of incident attributes.
- Our project aims to analyze historical incident data and build a predictive model that classifies incidents resolution time required into different categories. By analyzing the factors affecting the resolution time.
- Our model enables IT teams to identify high-risk tickets early, allocate resources proactively, and optimize workflows. These insights will support data driven SLA discussions with clients and help teams to improve service quality and operational efficiency.

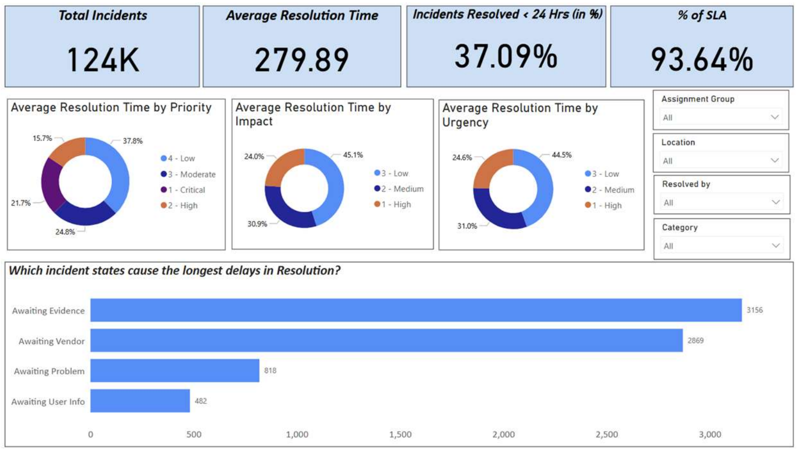
## III. Dataset Overview

The dataset consists of 141,712 observations and 37 features, including 4 numerical and 33 categorical variables. Key predictive features influencing resolution time include operational and organizational metrics, time-based and incident-related metadata, SLA risk indicators, and detailed incident categorization. These attributes collectively play a vital role in modeling the speed and efficiency with which IT service incidents are resolved.

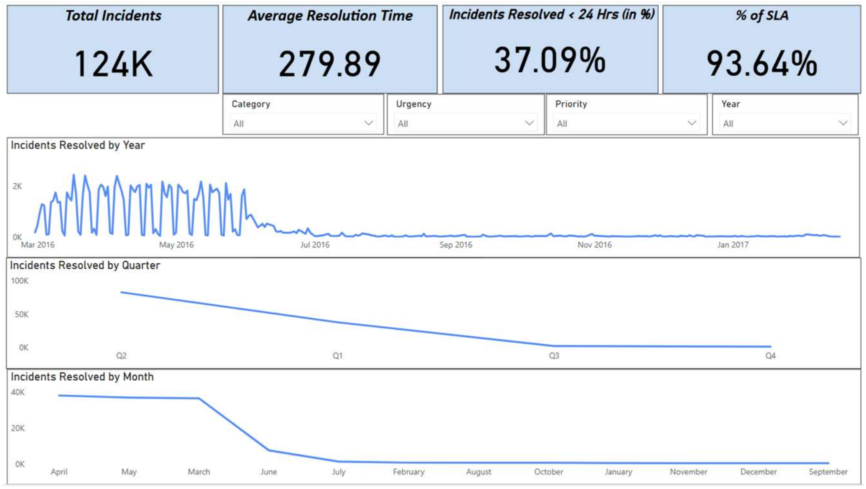
| **Category** | **Features** |
|---|---|
| Operational Metrics | reassignment_count, reopen_count, sys_mod_count, time_taken |
| Organizational Attributes | assignment_group, assigned_to, caller_id, opened_by, resolved_by, sys_created_by, sys_updated_by |
| Time Based Metrics | opened_at, resolved_at, closed_at, sys_created_at, sys_updated_at, closed_code |
| Incident Metadata | incident_state, active, made_sla, knowledge, u_priority_confirmation, notify, location, cmdb_ci, vendor |
| SLA Risk Indicators | impact, urgency, priority |
| Identifier | number |
| Target | time_group |
| Issue details | category, subcategory, u_symptom, contact_type, problem_id, rfc, caused_by |

## IV. Exploratory Data Analysis

Dashboard Link: https://app.powerbi.com/groups/me/reports/a649b80f-6a2b-4034-a81b-f9fd4dab3056/8ab36ebe4b76132c6a64?experience=power-bi

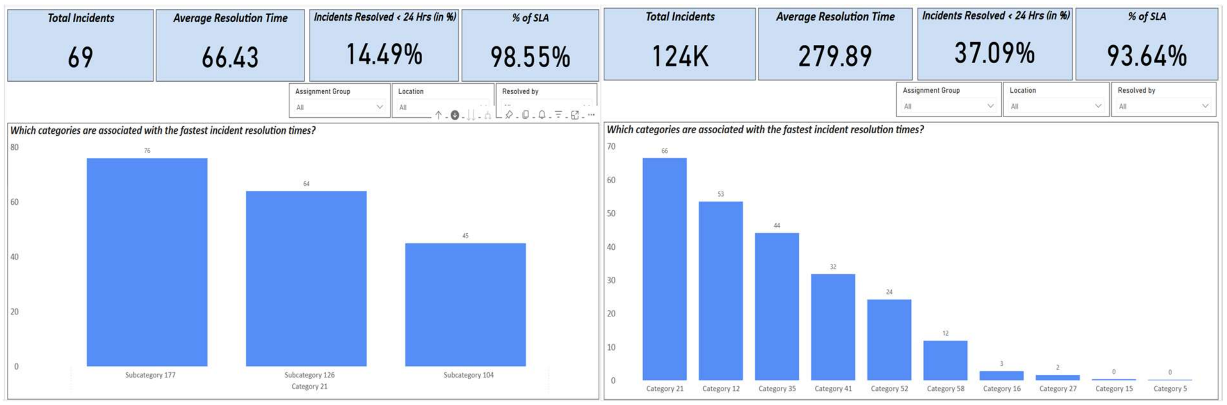


- The dashboard gives an overview of average resolution time taken for solving the incidents.
- There are some important metrics such as the Total number of incidents, Average resolution time, percentage of Incidents resolved within 24 hours and percentage of SLA being met.
- The average resolution time shows a split by Priority, Impact and Urgency. From the graph, the least resolution time is observed in High Priority (15.7%) followed by Critical, Moderate and Low Priority.
- There is a similar observation in Impact and Urgency as well. The incident states cause the longest delays in resolution graph talks about the incident states which consumes long time for resolution.
- The Awaiting Evidence (3156) has the highest resolution time followed by Awaiting Vendor (2869), Awaiting Problem (818) and Awaiting User Info (482). With the filters such as assignment group, location, resolved by and category we can observe for a particular case if applied.



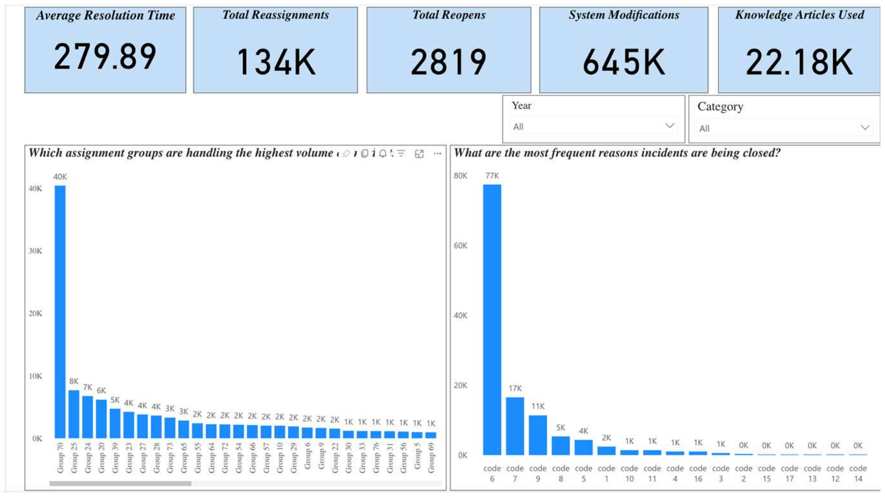
- This dashboard basically talks about the time-based trends for the incident resolution by year, quarter and month. According to the data we have data from March 2016 through Feb 2017.
- From the graphs, we can observe that there have been a good amount of incidents resolved with ups and downs but there is a sudden fall in June 2016 and since then it has decreased gradually.

➢ There are some filters such as category, year, urgency and priority to check for a particular case if required.



➢ This dashboard talks about the incidents split by Category and Sub-category. The fastest resolved tickets are seen first by Category.

➢ To see the sub-categories under each category we can drill down by clicking on the bar as seen in the screenshot above. There are filters such as assignment group, location and resolved by to check for a particular case.

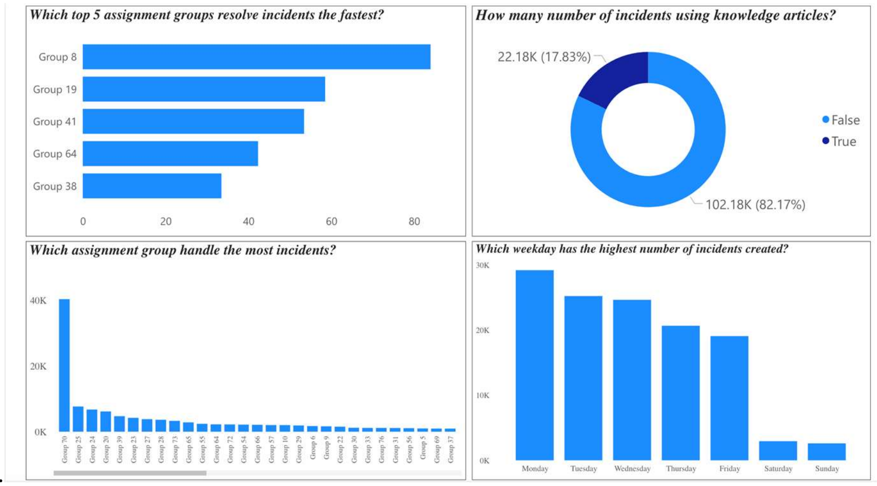


➢ The dashboard provides an overview of incident management performance which highlights average resolution time in hours, workload distribution, reassignment and reopen counts, system modifications, knowledge article used and closure trends. It also includes filters for year and category, allowing users to interactively explore incident trends across time and issue types.

➢ The average time taken is 279.89 hours, suggesting potential delays in addressing issues or having complex or low priority issues. A high number of reassignments (134K) indicates frequent handovers between teams.

➢ The 2819 reopen counts which show issues were not handled properly in the first attempt. The modifications that happened on the system for all incidents are 645K which indicates active management throughout lifecycle. We also observed that only over 22k leveraged knowledge articles.

➢ The workload doesn’t appear equal as one assignment group handles most incidents significantly. Finally, most incidents are getting close with code6 which might represent a standard code



➢ We observed that Group38 resolving incidents fastest within 34 hours followed by Group64 and Group41, while Group70 is handling most incidents.

➢ 82% of incidents were resolved by not leveraging knowledge articles while just around 18% of incidents resolved with the help of knowledge articles.

➢ Additionally, the volume of incidents peaks on Mondays, followed by Tuesday and Wednesday. The incidents are slowly decreasing as we move towards the weekend.

➢ This pattern may be due to reduced monitoring or response activity over weekends, leading to delayed logging and escalation until the start of the workweek.

**EDA Important Insights:**

➢ Our exploratory data analysis reveals several critical patterns in the incident management process. The average resolution time taken across all incidents is approximately 280 hours, though certain categories and groups show higher delays exceeding 500 hours.

➢ Group 70 and Group 75 handle the highest number of incidents, indicating potential overload or resource concentration, while Group 38, Group 64, and Group 41 are among the fastest in resolving incidents.

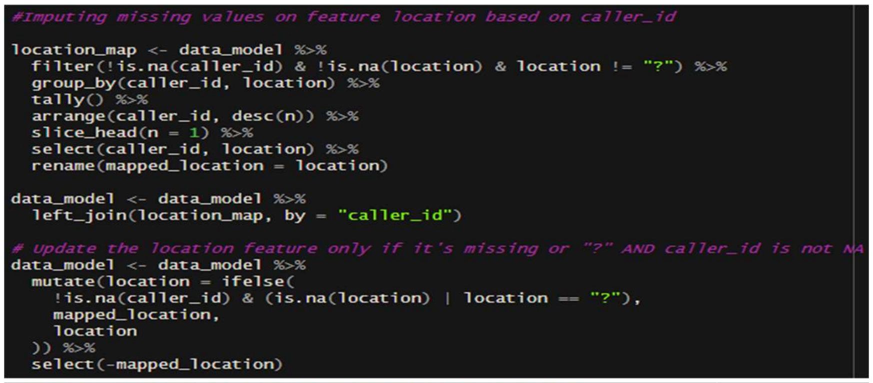
➢ Reassigning incidents are very high which suggests process inefficiencies or unclear routing logic. About 63% of the incidents take more than 24 hours to resolve with only 37% getting resolved within the threshold. Some states like Awaiting Evidence and Awaiting Vendor are the major contributors for resolution delay.

➢ The number of incidents resolved shows a fluctuating weekly trend but an overall declining pattern across months and quarters in 2016, suggesting a drop in resolution activity or incident volume over time for Category 22 with medium urgency and moderate priority.

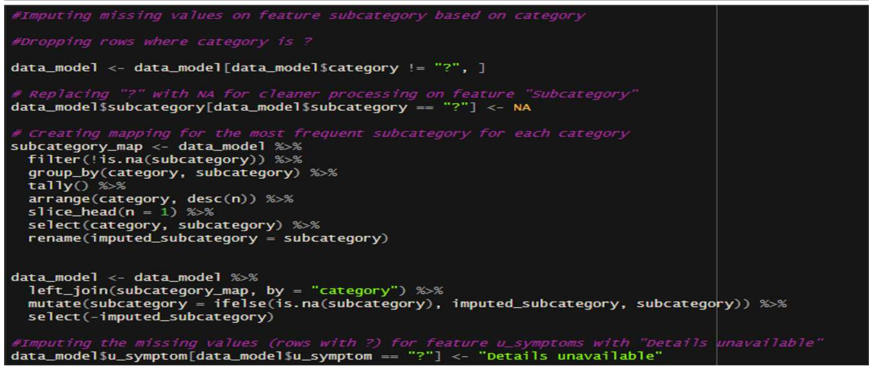
➢ Category 17 resolving the fastest incidents, based on Group 70, Location108, and Resolved by 103 which indicates strong performance under these specific conditions. Within Category 17, Subcategory 174 resolves incidents much faster compared to Subcategory 90, showing that resolution efficiency can vary significantly even within the same category.

- Incident volume is highest on weekdays, especially Mondays, with a sharp drop during weekends. Additionally, resolution time varies with priority, urgency, and impact, with low-priority incidents resolved faster, while critical and high-impact incidents face longer delays.
- The SLA compliance average rate is around 93–98%, though some segments fall below that, calling for targeted improvements.

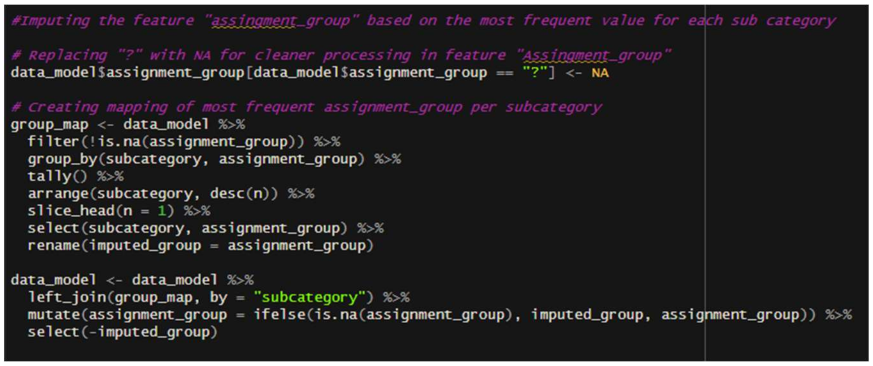
## V. Preprocessing and Feature Engineering



```r
#Imputing missing values on feature location based on caller_id

location_map <- data_model %>%
  filter(!is.na(caller_id) & !is.na(location) & location != "?") %>%
  group_by(caller_id, location) %>%
  tally() %>%
  arrange(caller_id, desc(n)) %>%
  slice_head(n = 1) %>%
  select(caller_id, location) %>%
  rename(mapped_location = location)

data_model <- data_model %>%
  left_join(location_map, by = "caller_id")

# Update the location feature only if it's missing or "?" AND caller_id is not NA
data_model <- data_model %>%
  mutate(location = ifelse(
    !is.na(caller_id) & (is.na(location) | location == "?"),
    mapped_location,
    location
  )) %>%
  select(-mapped_location)
```



```r
#Imputing missing values on feature subcategory based on category

#Dropping rows where category is ?

data_model <- data_model[data_model$category != "?", ]

# Replacing "?" with NA for cleaner processing on feature "Subcategory"
data_model$subcategory[data_model$subcategory == "?"] <- NA

# Creating mapping for the most frequent subcategory for each category
subcategory_map <- data_model %>%
  filter(!is.na(subcategory)) %>%
  group_by(category, subcategory) %>%
  tally() %>%
  arrange(category, desc(n)) %>%
  slice_head(n = 1) %>%
  select(category, subcategory) %>%
  rename(imputed_subcategory = subcategory)


data_model <- data_model %>%
  left_join(subcategory_map, by = "category") %>%
  mutate(subcategory = ifelse(is.na(subcategory), imputed_subcategory, subcategory)) %>%
  select(-imputed_subcategory)

#Imputing the missing values (rows with ?) for feature u_symptoms with "Details unavailable"
data_model$u_symptom[data_model$u_symptom == "?"] <- "Details unavailable"
```

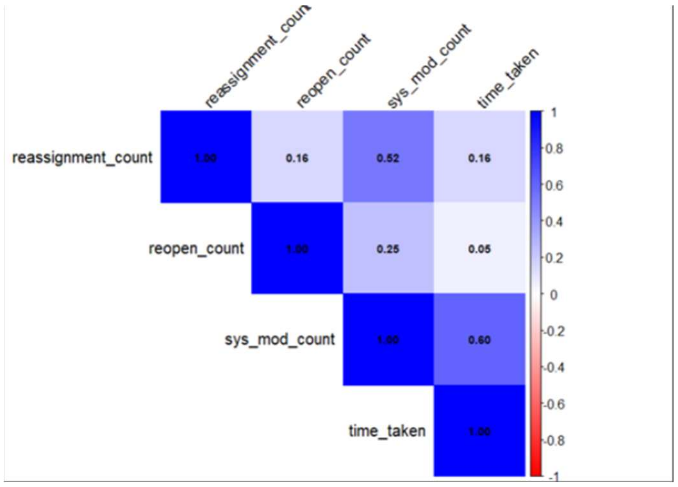


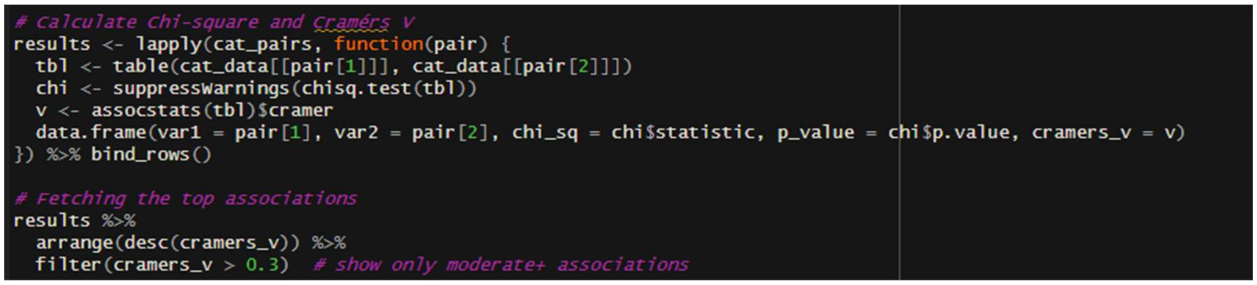
```r
#Imputing the feature "assingment_group" based on the most frequent value for each sub category

# Replacing "?" with NA for cleaner processing in feature "Assingment_group"
data_model$assignment_group[data_model$assignment_group == "?"] <- NA

# Creating mapping of most frequent assignment_group per subcategory
group_map <- data_model %>%
  filter(!is.na(assignment_group)) %>%
  group_by(subcategory, assignment_group) %>%
  tally() %>%
  arrange(subcategory, desc(n)) %>%
  slice_head(n = 1) %>%
  select(subcategory, assignment_group) %>%
  rename(imputed_group = assignment_group)

data_model <- data_model %>%
  left_join(group_map, by = "subcategory") %>%
  mutate(assignment_group = ifelse(is.na(assignment_group), imputed_group, assignment_group)) %>%
  select(-imputed_group)
```

- We have dropped number feature as it is a unique identifier and not useful for our analysis. We have removed missing values or bad data like ‘?’ from category feature.
- We have dropped duplicate values.
- We have addressed missing values to ensure quality and completeness of our dataset for below features:

a. **Location:** We have imputed it based on the caller ID assuming that each caller is associated with a specific location.
b. **Subcategory:** We have imputed it using the most common subcategory associated with the given category.
c. **Assignment Group:** We have filled the missing values based on the most frequent assignment group corresponding to the related subcategory and it ensures that each incident is assigned to the most appropriate group based on historical data.
d. **U_symptom:** We have replaced the missing entries with 'Details not available' since this feature captures the user's perception of service availability, which cannot be accurately inferred or imputed from other fields.

➢ **Correlation analysis**

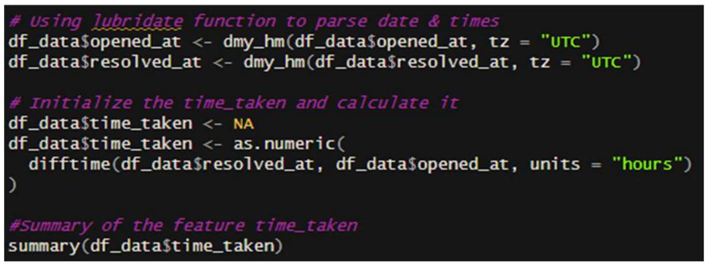




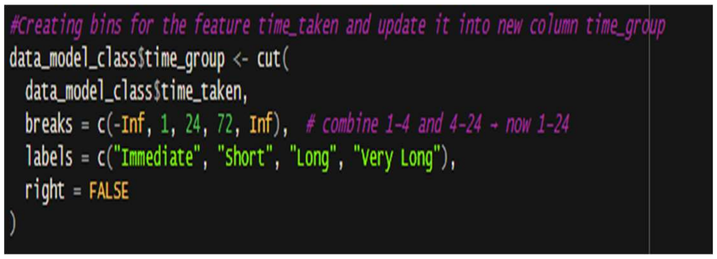
```
# Calculate Chi-square and Cramérs V
results <- lapply(cat_pairs, function(pair) {
  tbl <- table(cat_data[[pair[1]]], cat_data[[pair[2]]])
  chi <- suppressWarnings(chisq.test(tbl))
  v <- assocstats(tbl)$cramer
  data.frame(var1 = pair[1], var2 = pair[2], chi_sq = chi$statistic, p_value = chi$p.value, cramers_v = v)
}) %>% bind_rows()

# Fetching the top associations
results %>%
  arrange(desc(cramers_v)) %>%
  filter(cramers_v > 0.3)  # show only moderate+ associations
```

We plotted correlation matrix for numerical features, and we observed that there were no highly correlated features. We have performed chi square test to check association between categorical features and we found out that priority is highly associated with impact and urgency, so we have dropped the priority feature.

➢ **Target variable creation: time_group**



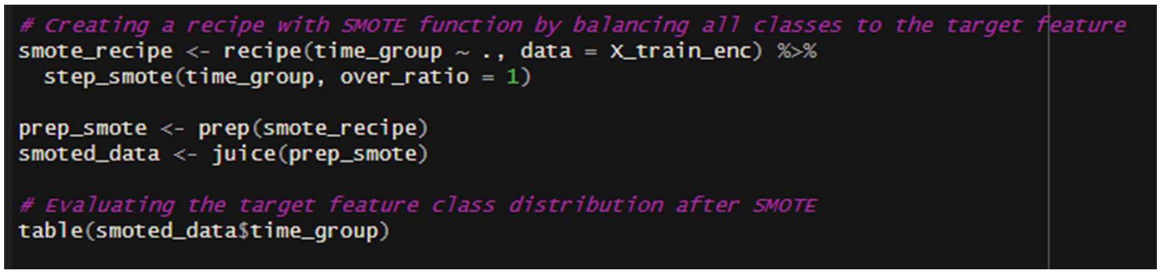
```
# Using lubridate function to parse date & times
df_data$opened_at <- dmy_hm(df_data$opened_at, tz = "UTC")
df_data$resolved_at <- dmy_hm(df_data$resolved_at, tz = "UTC")

# Initialize the time_taken and calculate it
df_data$time_taken <- NA
df_data$time_taken <- as.numeric(
  difftime(df_data$resolved_at, df_data$opened_at, units = "hours")
)

#Summary of the feature time_taken
summary(df_data$time_taken)
```



```
#Creating bins for the feature time_taken and update it into new column time_group
data_model_class$time_group <- cut(
  data_model_class$time_taken,
  breaks = c(-Inf, 1, 24, 72, Inf),  # combine 1-4 and 4-24 → now 1-24
  labels = c("Immediate", "Short", "Long", "Very Long"),
  right = FALSE
)
```

➢ We created a new feature time_taken, which captures the time taken to resolve the incident (in hours) by subtracting the opened_at timestamp from resolved_at timestamp.
➢ Also, we have converted this continuous variable to categorical target feature for our classification model.
➢ We have created our target variable 'time_group' by binning time_taken values to 4 groups:
➢ Immediate: less than 1 hour, Short: 1 to 24 hours, Long: 24 to 72 hours, Very long: more than72hours.

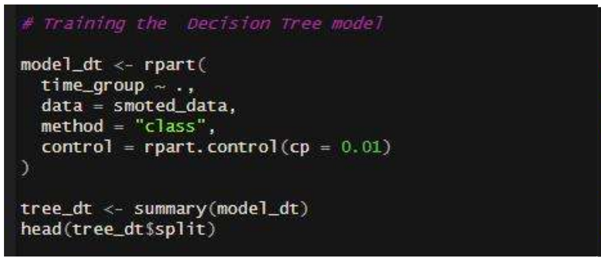
## VI. Model Performance and Evaluation



```
# Creating a recipe with SMOTE function by balancing all classes to the target feature
smote_recipe <- recipe(time_group ~ ., data = X_train_enc) %>%
  step_smote(time_group, over_ratio = 1)

prep_smote <- prep(smote_recipe)
smoted_data <- juice(prep_smote)

# Evaluating the target feature class distribution after SMOTE
table(smoted_data$time_group)
```

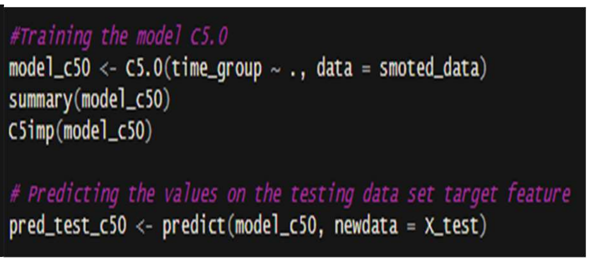
➢ We split our dataset into a training set (70%) and a testing set (30%) to evaluate model performance on unseen data. Since the target variable was imbalanced, we applied the SMOTE technique to the training data to balance the class distribution.
➢ We trained four classification models Decision Tree, C5.0, Random Forest, and XGBoost.

**Decision Tree**



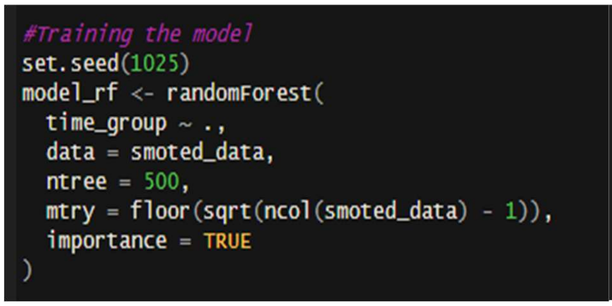
```
# Training the  Decision Tree model

model_dt <- rpart(
  time_group ~ .,
  data = smoted_data,
  method = "class",
  control = rpart.control(cp = 0.01)
)

tree_dt <- summary(model_dt)
head(tree_dt$split)
```

**C 5.0**



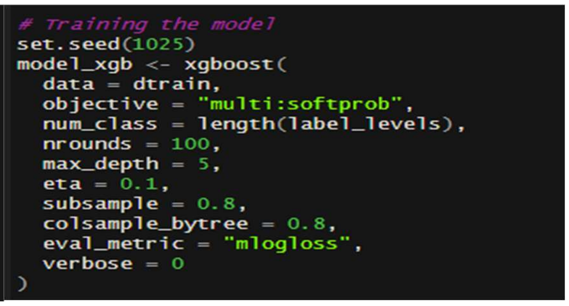
```
#Training the model C5.0
model_c50 <- C5.0(time_group ~ ., data = smoted_data)
summary(model_c50)
C5imp(model_c50)

# Predicting the values on the testing data set target feature
pred_test_c50 <- predict(model_c50, newdata = X_test)
```

**Random Forest**



```
#Training the model
set.seed(1025)
model_rf <- randomForest(
  time_group ~ .,
  data = smoted_data,
  ntree = 500,
  mtry = floor(sqrt(ncol(smoted_data) - 1)),
  importance = TRUE
)
```

**XG Boost**

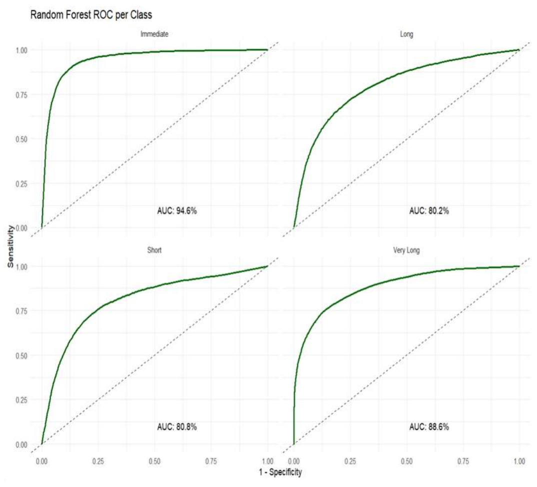


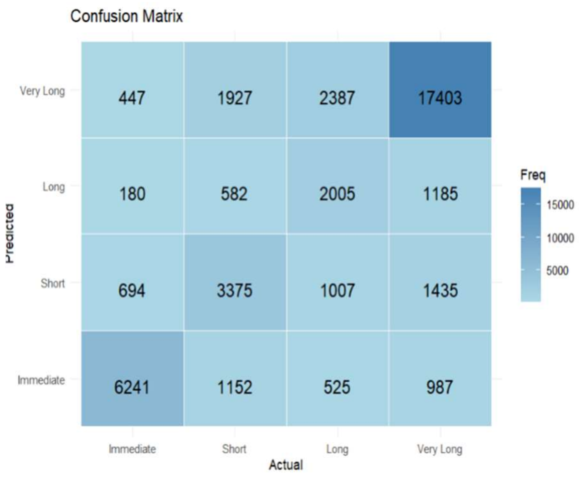
```
# Training the model
set.seed(1025)
model_xgb <- xgboost(
  data = dtrain,
  objective = "multi:softprob",
  num_class = length(label_levels),
  nrounds = 100,
  max_depth = 5,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8,
  eval_metric = "mlogloss",
  verbose = 0
)
```

- Random Forest and XGBoost both performed significantly better than the baseline models (Decision Tree and C5.0).
- Random Forest outperformed XGBoost across all key metrics—accuracy (69.90%), precision (62.80%), recall (61.80%), and F1 score (61.70%).
- The higher precision and recall values of Random Forest suggest better reliability in correctly identifying positive cases and minimizing false negatives.
- With an F1 score of 61.70%, Random Forest provides a more balanced performance between precision and recall than XGBoost, which had an F1 score of 54.50%.
- Given its consistently superior metrics, Random Forest is the preferred model for this classification task in terms of both accuracy and predictive strength.
- The higher F1-score and precision indicate that Random Forest delivered more accurate and reliable classifications, with less misclassification errors when compared with other models.

| Models | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Random Forest | 69.90% | 62.80% | 61.80% | 61.70% |
| XG Boost | 64.80% | 54.80% | 54.20% | 54.50% |
| Decision Tree | 53.50% | 47.40% | 50.00% | 46.30% |
| C 5.0 | 63.80% | 54.10% | 54.70% | 54.10% |

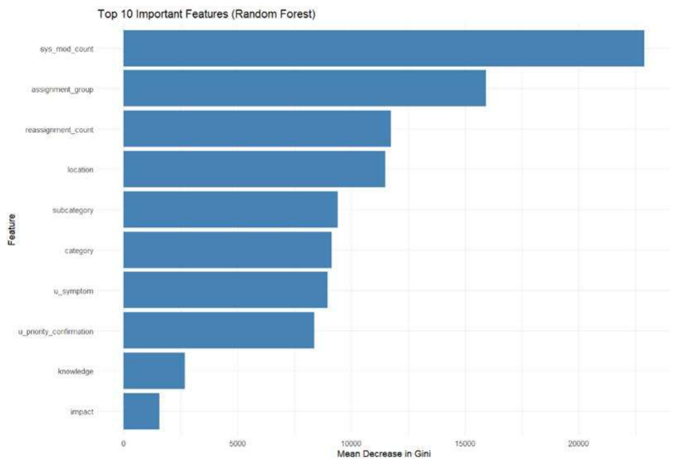
## Confusion Matrix and ROC Curve for top performer





- ➢ Model performs well for the "Immediate" (AUC 94.6%) and "Very Long" (AUC 88.6%) classes, indicating high predictive accuracy for early and prolonged outcomes.
- ➢ The "Long" (AUC 80.2%) and "Short" (AUC 80.8%) classes show moderately strong performance, suggesting room for improvement in mid-range predictions.
- ➢ Random Forest model demonstrates reliable classification across all classes, showcasing its robustness for multi-class prediction scenarios.
- ➢ Model frequently misclassifies "Immediate" instances as "Very Long" (6,241 cases) and "Very Long" as "Long" or "Short" indicating difficulty in distinguishing extremes.
- ➢ Model correctly predicts "Short" duration in 3375 cases, demonstrating better performance in this category compared to others.
- ➢ Class "Long" and "Very Long" categories show substantial confusion with each other and with adjacent classes, suggesting overlapping feature patterns or model limitations in boundary clarity

### Feature Importance for the top performer Random Forest



- ➢ We have plotted the feature importance graph using a Random Forest model. The model ranks features based on the mean decrease in Gini, which measures how much each feature improves the decision-making process.
- ➢ sys_mod_count is the top predictor, followed by assignment_group and reassignment_count, all of which have the greatest impact on predicting our target variable.
- ➢ Reducing reassignments and minimizing unnecessary modifications can help make the resolution process faster and smoother.

## VII. Summary and Conclusion

- ➢ In this project, we analyzed the enriched incident management process dataset to uncover patterns influencing incident resolution time and to build predictive models that can classify incidents based on expected resolution durations. Through detailed exploratory data analysis, we identified key operational factors such as assignment group, system modifications and reassignments, all of which have highest impact on predicting target variable.
- ➢ Notably, we observed that a few assignment groups handle most incidents, while others show faster resolution capabilities. Additionally, significant delay caused by certain external dependencies such as Awaiting Evidence and Awaiting Vendor.
- ➢ We performed multiple classification models including Decision Tree, C5.0, Random Forest, and XGBoost. Among these, Random Forest outperformed others, achieving an accuracy of 69.9% which shows balance between precision and recall. The model's ROC AUC scores also confirm its strong ability to distinguish between different classes in the target variable.

## VIII. Future Scope

- Deploy the predictive model into a live dashboard environment for real-time SLA monitoring and proactive alerting of high-risk incidents. Explore advanced feature selection and transformation techniques.
- Periodically retrain models on latest data to capture evolving incident trend and maintain prediction accuracy.
- Develop a recommendation system to suggest relevant knowledge articles based on incident metadata, improving usage and reducing resolution time.

## IX. References

**Dataset:**https://archive.ics.uci.edu/dataset/498/incident+management+process+enriched+event+log