**Merilytics Dataset Report**

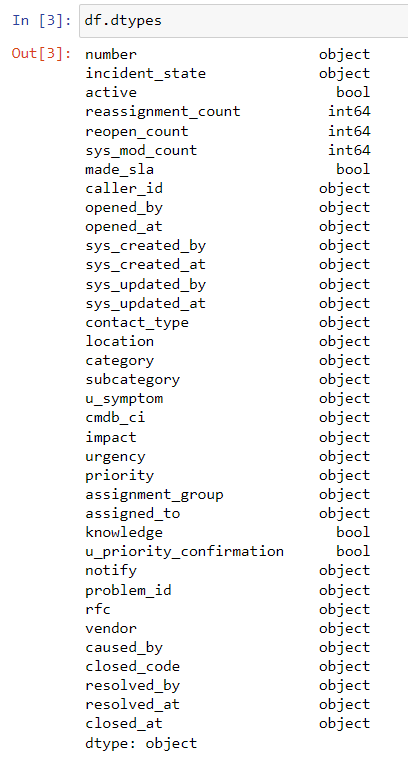
**Problem Statement:**

The Chief Operating Officer of a Software Service Provider company is looking to evaluate the efficacy of their incident/issue resolution process and have shared the details of the incident log data. They would like to perform the following analysis to improve their operations to drive better customer satisfaction.

* Understand the distribution of incidents to identify the spread by key attributes.
* Understand the alignment between urgency/priority of incidents against the resolution parameters/statistics.
* Build a predictive model using the data that can estimate the resolution time for incident raised in the future.
* Build a classification model that would bucket the incidents into high priority/ urgency buckets.
* Suggest recommendations to reduce resolution time.

**Exploratory Data Analysis and Data Pre-processing:**

1. **Types of features in the dataset**

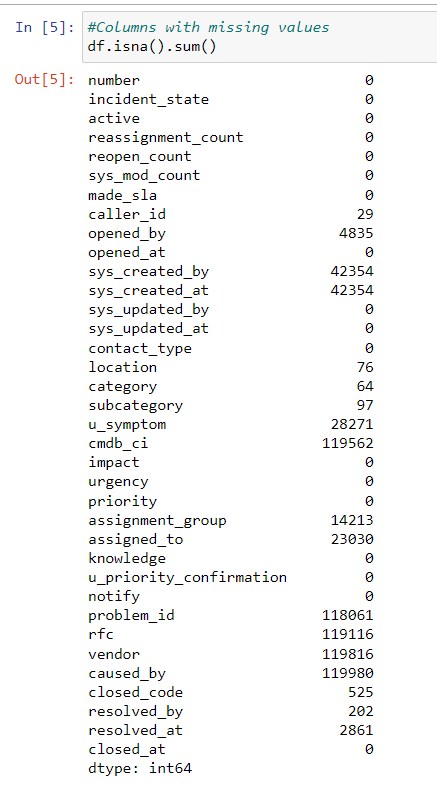


1. **Handling missing values:**

List of features with missing values and their corresponding count:

All the missing values have been handled by replacing with:

* **mode** if it contains categorical data
* **mean** for date/time relevant columns



1. **Data Sampling and Normalization:**

* Used Stratified split to select 1% of the data from the original dataset and to ensure the class balance is kept intact.
* Normalized the features using “MinMaxScaler” which had higher range to bring it to a uniform scale i.e between 0 and 1.

**Feature selection:**

1. Used corr() for detecting multicollinearity among features

* In case of multicollinearity, we might lose reliability in determining the effects of individual features in the model and hence, we remove these features.
* To resolve this, we use corr() method which returns a correlation score for each feature and visualized the same via a heatmap.
* This score determines the strength of correlation between the independent variables. All the features with a score of greater than 0.7 were removed from the dataset suggesting high multicollinearity.

Features: 'urgency', 'notify' were removed as a result of multi-collinearity.

**Model Building and Hyperparameter tuning:**

1. Choosing the predictor variable:

* For the classification model, chose the predictor variable as: “**Priority**”
* For the regression model, chose the predictor variable as: “**Resolution time**”

1. Deleted all the rows with “incident\_state” anything other than “NEW”, “ACTIVE”

This was done because the SLA clock of an incident runs only when the incident is in service operations queue which is represent by the incident states: NEW & ACTIVE.

1. We then calculate the following:

'timediff\_per\_incidentstate’ = ‘closed\_at' - 'sys\_updated\_at’

AND

1. Apply the group by() function based on “incident number” and sum the timediff\_per\_incidentstate for “NEW” & “ACTIVE” states to get the final resolution time per incident.
2. Machine Learning algorithm used:

* Applied "Decision Tree Classifier" and “Support Vector Classifier” for the classification model i.e to bucket the incidents into different priority incidents.
* Applied “Decision Tree Regressor” and “Random Forest Regressor” for the regression model i.e to predict the resolution time of the incidents.

Hyperparameter Tuning:

* + Applied GridSearchCV and RandomSearchCV for hyperparameter tuning of the models.

1. Evaluation metrics:

* I chose “weighted f1 score” to evaluate the accuracy of the classification model and also to tune the hyper parameters of the model.
* I chose “r2 score” to evaluate the accuracy of the regression model and also to tune the hyper parameters of the model.

|  |  |  |
| --- | --- | --- |
|  | **Classification** | **Weighted F1 score** |
| Before Hyperparameter tuning | DecisionTreeClassifier | 0.975 |
| After Hyperparameter tuning | DecisionTreeClassifier | 0.977 |
| Before Hyperparameter tuning | SVC | 0.897 |
| After Hyperparameter tuning | SVC | 0.959 |

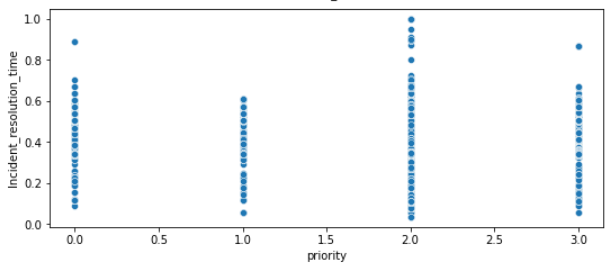
|  |  |  |
| --- | --- | --- |
|  | **Regression** | **R-squared** |
| Before Hyperparameter tuning | DecisionTreeRegressor | 0.851 |
| After Hyperparameter tuning | DecisionTreeRegressor | 0.827 |
| Before Hyperparameter tuning | RandomForestRegressor | 0.846 |
| After Hyperparameter tuning | RandomForestRegressor | 0.829 |

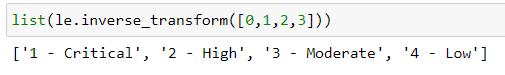
1. Additional points:
   * Removed all the records where the ticket life was less than 0 indicating re-opened incidents. Only one-time incidents were considered for the analysis of this dataset to reduce the complexity.

* For the regression model, only those features were considered where the records were similar for every incident. This was done so the groupby() function could be easily applied.

1. Observations and recommended suggestions to reduce the resolution time:
2. From the below graph “Priority” versus “incident\_resolution\_time”, we see that the incident resolution time for “1- Critical” priority is very high which should not be the case.

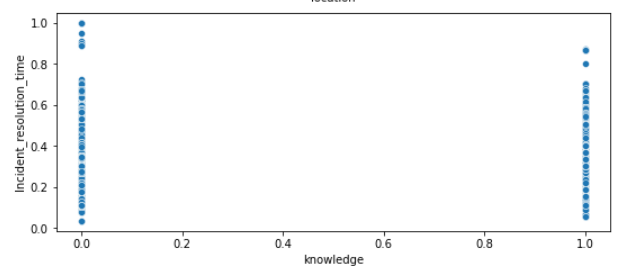
These kind of incidents should be given utmost priority and have to be resolved as early as possible.





1. From the below graph: “knowledge” versus “incident\_resolution\_time”, it can be observed, the incidents that made use of knowledge base documents had a lower resolution time as compared to the ones that did not.

So, creation of more knowledge base documents and usage of the same would definitely reduce the resolution time of the incidents.



1. From the below graph: “time\_creation\_sys\_timegap” versus “priority”, it can be observed that the time gap between the events: user opening the incident and system creating it is quite high.

This could impact the resolution time and the business especially for “Critical” and “High” incidents. Ensuring minimal gap between the two could save time and increase efficiency.

