**COMP-309 - 21F --Data Warehousing & Predictive Analysis**

**Group project #2**

**Group #2**

**Members:**

**Alvin Yap**

**Jarod Lavine**

**Anmoldeep Singh Gill**

**Michael Asemota**

**Girishanth Sivanesachselvan**

# 

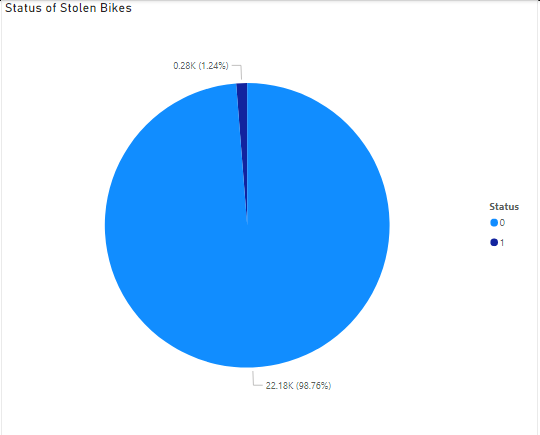
API:<https://comp309-group2-flaskapi.azurewebsites.net/>

Frontend:<https://datawarehousing-group2-frontend.azurewebsites.net/>

# **Executive Summary:**

The problem we are aiming to solve through the analysis of the Toronto Police Bike Thefts dataset is to predict if a certain bike that is stolen is likely to be returned or not. The solution we came up with was to create a decision tree that would classify if a bike would be returned based on 75 features that were extracted from the data set. What we found was that 99% of bikes stolen were not returned and that a wide variety of bikes are targeted in the city of Toronto.

**Key Findings:**

****

Only 1% of bikes that were stolen were recovered.

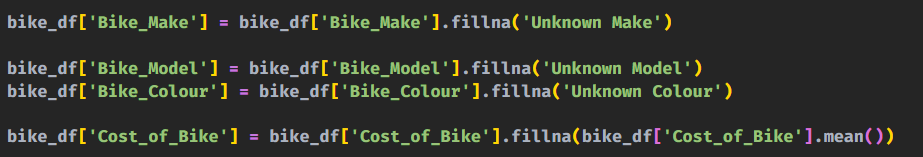
Given the dataset, it was found that a decision tree was the best model to perform the classification as it had the highest accuracy rate (91.9%) via cross validation compared to a logistic regression (76.5%) or a KNN nearest neighbour model (86.2%). The model on the test data has an accuracy of 88.1%.

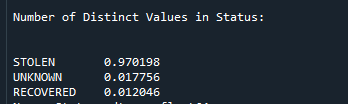
# **Data exploration and findings**

In the data exploration phase we realized that there were numerous null values that needed to be cleaned up. The dates in Occurrence date and Report Date were converted to “datetime” since they are easier to work with in pandas. We used a combination of matplotlib import pyplot, pandas, seaborn, numpy, and scipy’s chi2\_contigency module.

We then needed to take care of the missing values.

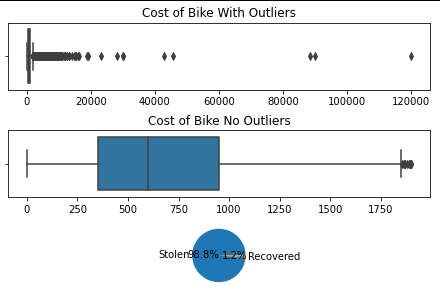
Some null values in the columns ‘Bike\_Make’, ‘Bike\_Model’, ‘Bike\_Colour’ and ‘Cost\_of\_bike’ were replaced with ‘Unknown Make’, ‘Unknown Model’ etc.





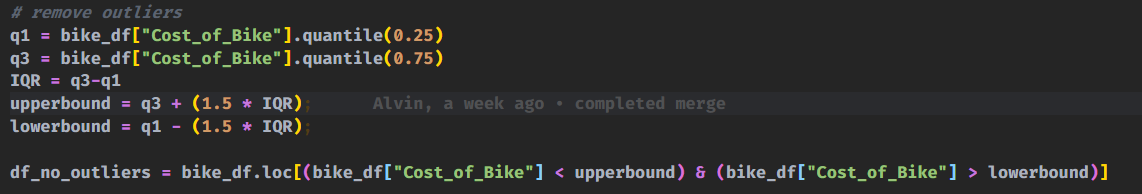
Next, we identified outliers in the data set.

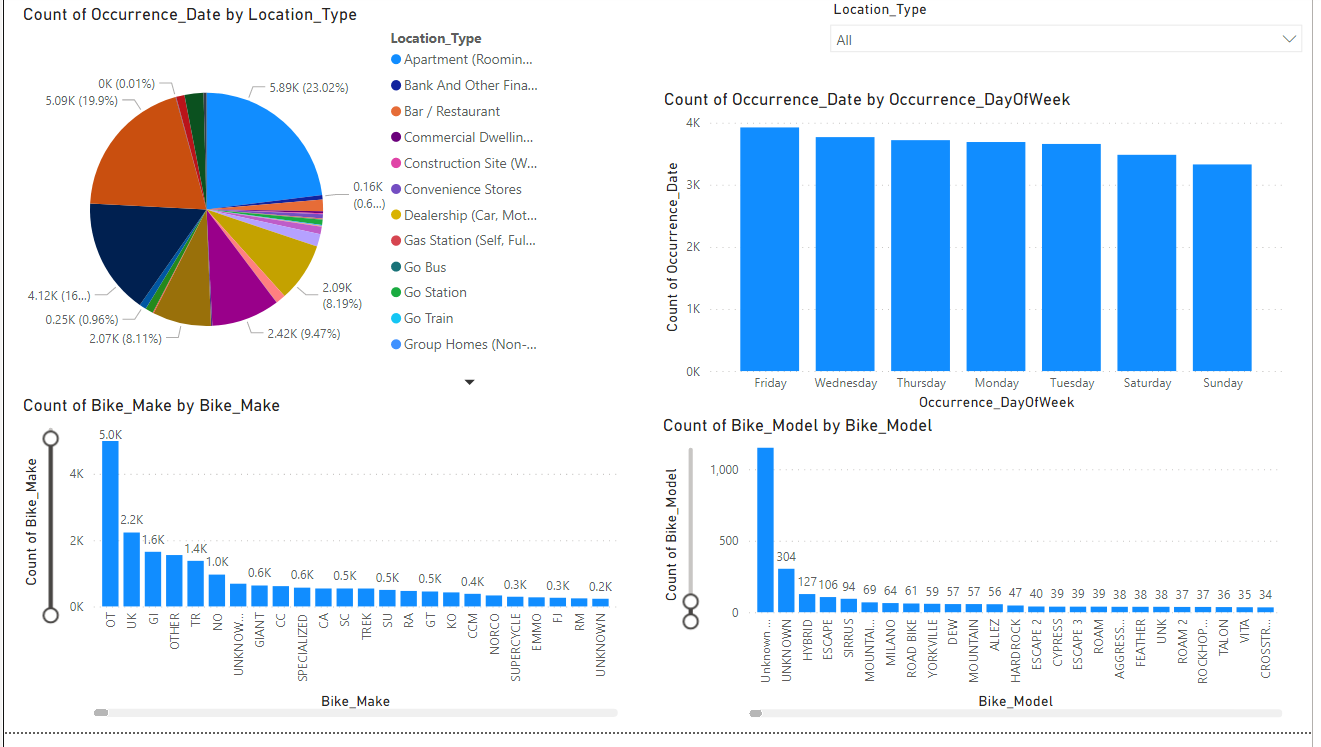
The most prominent outlier was the bike cost. There were a few bikes that were extremely expensive compared to most of the other bikes in the feature set. One bike had a value of 120,000! So we removed those outliers.

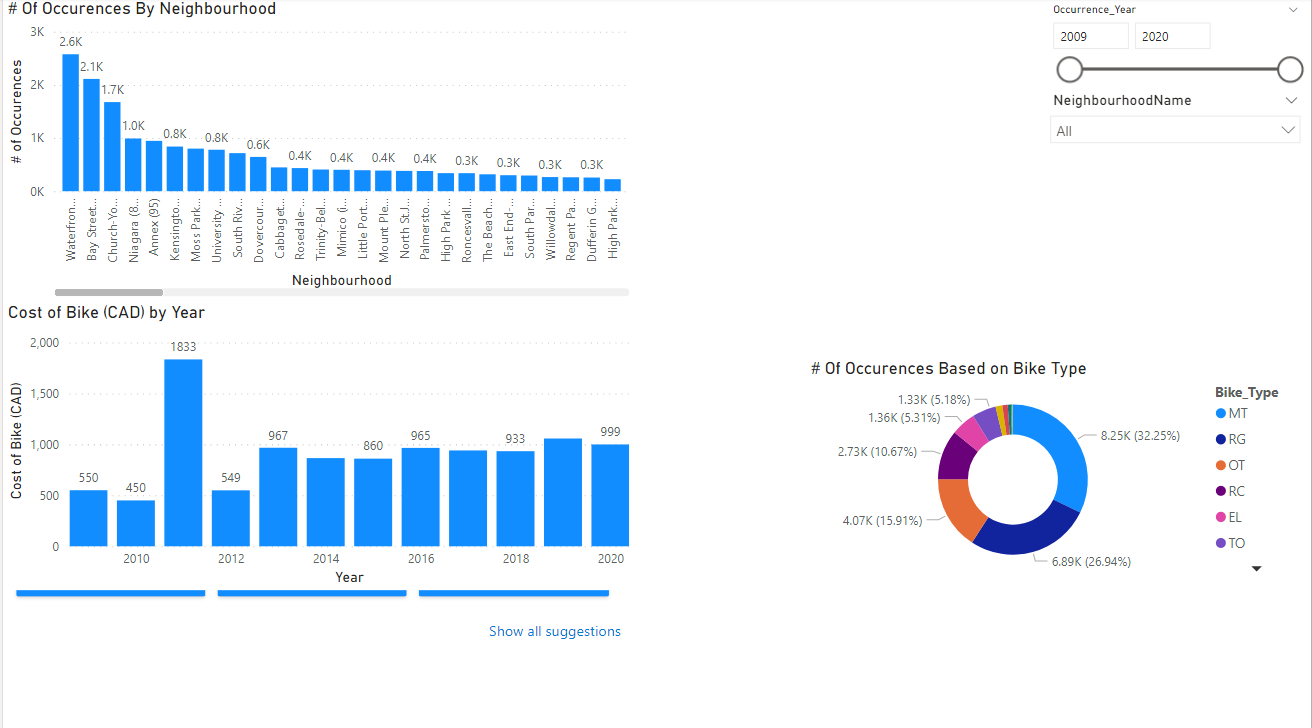


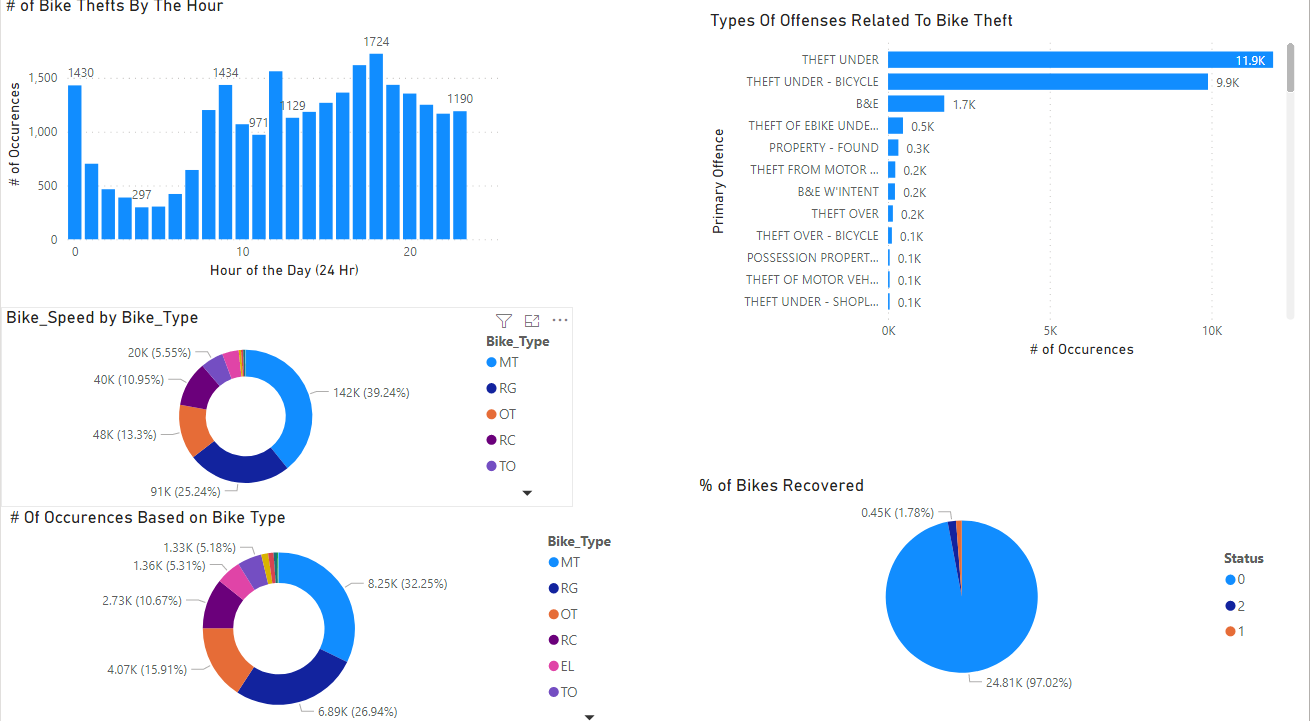
By using the box plot and calculation of the IQR, Q1,Q3 we were able to simply remove any rows that existed outside of the upper and lower bounds.

For our visualizations we used powerBI to see if there were any visual trends we could identify. The Trend of note is that there is a significant dip between the hours of 1 am to 7 am. Other trends is that the specific bike model and bike type are less important but the bike’s price and the neighbourhood that the bike is in affects if the bike is likely to be stolen.



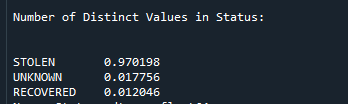






# **Feature selection**

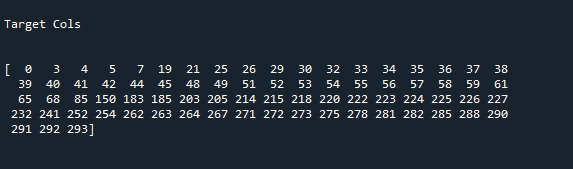
Since a status of “Unknown” indicates that no conclusion was reached or there was no follow-up on whether or not the bike was discovered, our options were to either add unknown to stolen or recovered or drop those rows all together. We felt that because of the ambiguity and that stolen is the prevailing we added the unknown to the stolen category. Testing both scenarios, there was a 2% increase for the decision tree’s accuracy and a 5% increase for the accuracy of KNN nearest neighbors by adding unknown to stolen.



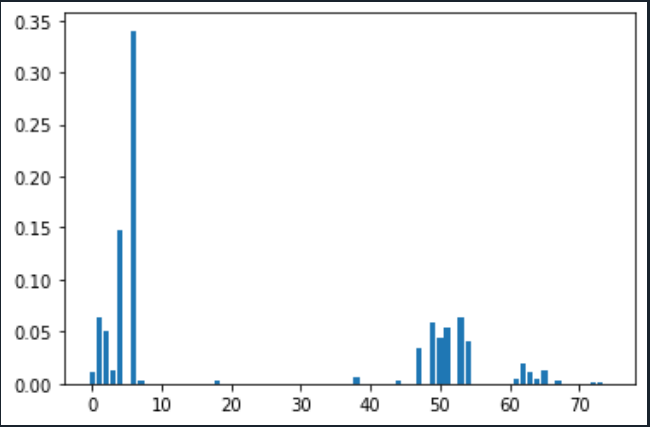
The bike's status is the feature we'll be concentrating on. We can't forecast the likelihood of a bike being stolen because all bikes are stolen in this data set. However, we can anticipate whether or not a stolen bike will be found. The time between report and incidence is an intriguing data piece that isn't readily available in this data set.

Thus we added the column named “Report\_Lag” to represent this time lag.

For the actual features, do the following. We first resample the target variable and all the features using the SMOTE library. Following this, we then use sklearns' SelectKbest and chi2 libraries to give us the scores for all the features in order of how they aid in predicting Status. Given the nature of our data being largely one hot encoded, we decided to take the top 15% (75 features) that helped predict our target status.

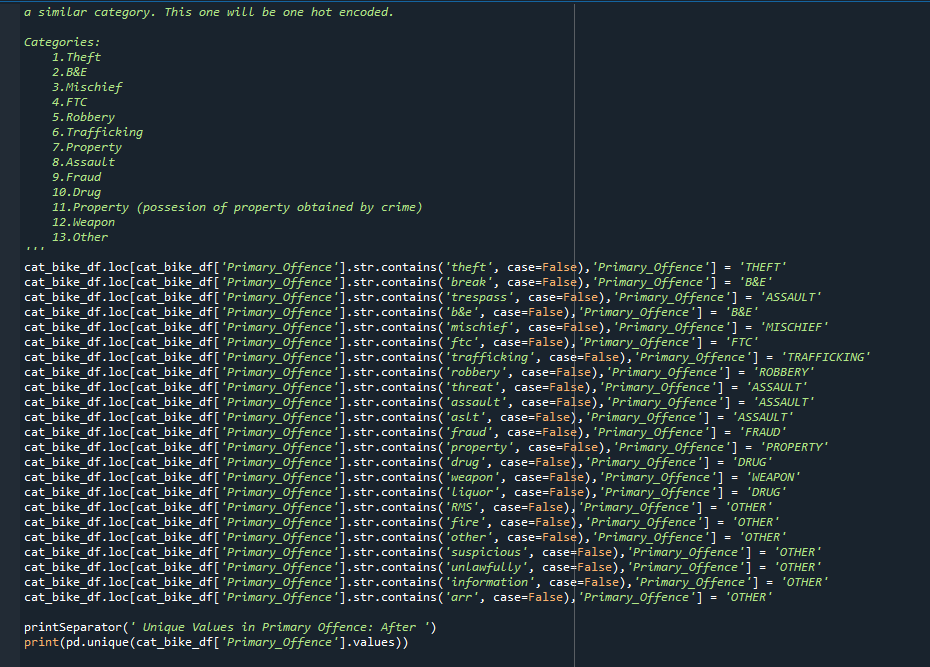


Getting the feature importance is the act of scoring the features on how related they are to a target variable, the higher the scores are the more important they are in predicting the target variable. In our case the target variable was the bike status. After running the feature importance the features that ended up being the highest correlated to predicting the feature were the cost of the bike, and report lag. The other features were important but not as important as those two mentioned.

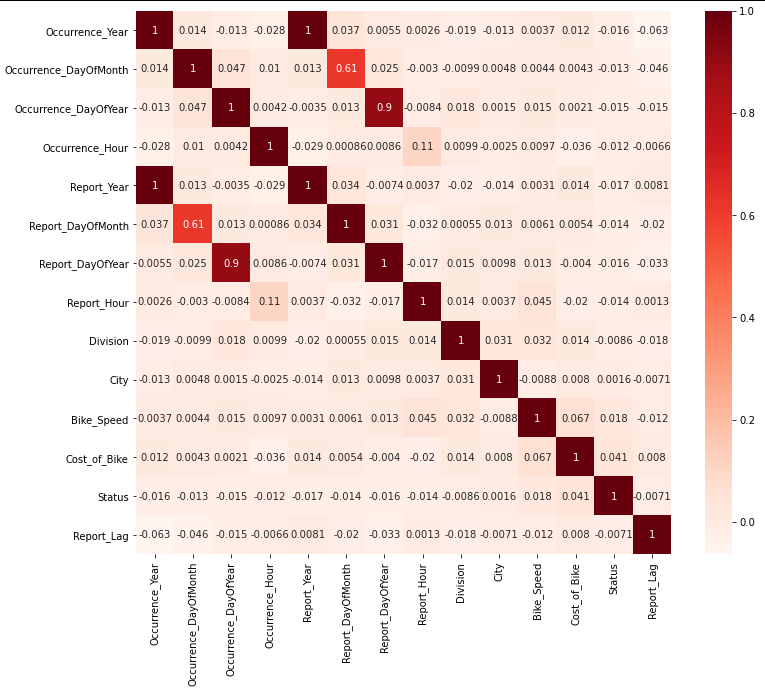


# **Data modeling**

As for our data cleaning, we did the following. For numerical data, we removed any outliers that existed as noted above. For categorical data, we ran into a different set of circumstances. Due to the high cardinality of the data alongside human error that went into the data set, we did our best to logically group values in a column together to prepare them for one hot encoding. For instance:



We proceed our way down each categorical data and determine if a column can be converted into a binary column. If it cannot, we begin a group to reduce the dimensionality that will occur when we apply the one hot encoding.



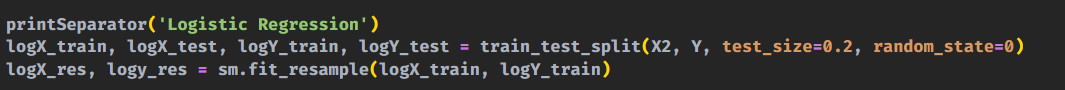
We also perform a heat map to determine if there are any values that are dependent on our target variable Status. If there was a dependency we would remove that column we would remove that column as the models expect independent variables from the target.

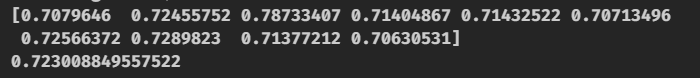
We proceed to then remove redundant columns such as occurrence hour as the important data that can be gleaned from these columns has already been consolidated in the Report Lag column.

# **Model building:**

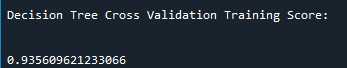
In regards to model building, we tested 3 different classification models: KNN nearest neighbour, Decision Tree , and logistic regression. Due to the largely imbalanced data set, we used smote to over sample the returned bikes to make up for the fact that only 1% of the bikes were returned.

We created a logistic regression model and decision tree. The training data was also split in 80-20 split.The logistic regression had a training score of 72% using cross validation.

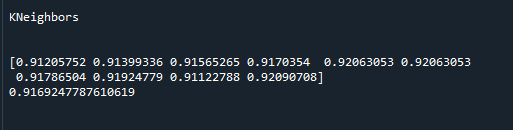




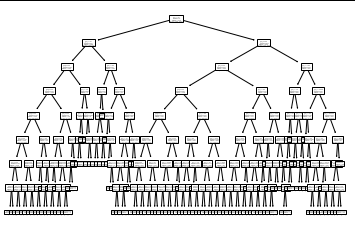
The decision tree’s score was 93% during cross validation.



Our KNN nearest neighbour had a 91% accuracy.



Due to the highly imbalanced data to begin with, we also did a confusion matrix to determine other statistical evaluation of the decision tree model.

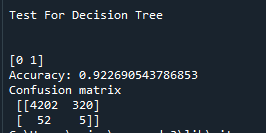


Using the confusion matrix below we can calculate the following:

Precision= 0.929

Recall = 0.988

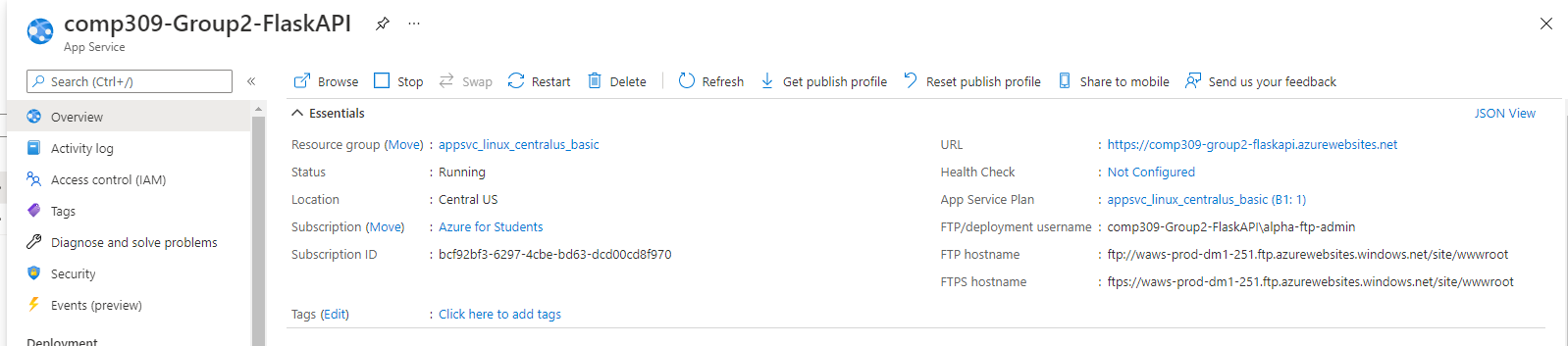
F-Measure = 0.958



Based on these calculations, our decision tree does very well given the imbalanced data set.

# **Deploying API**

We have deployed the API as an App Service on Azure that is running using Python 3.8 on a Linux OS Instance.



The Flask API uses the dependencies listed in “requirements.txt” file to install them while deployment and also has an additional flask\_cors package to overcome the Cross SIte Validation Errors for calling it from Frontend. The flask API then runs from the app.py file in our project.