CS4830 - Final Project Report

Group Name: Veracity

Team Members	
Roll Number	**Name**
J Mahesh	CH17B049
Jain Devansh Rakesh	CH17B050
RA Keerthan	CH17B078

Objective

The dataset provided for the project was the **NYC Parking Tickets dataset** which is based on the NYC Parking Tickets public dataset on Kaggle. The total number of rows in the provided dataset was 18,509,419. Data exploration was performed on a subset of the data (300,000 rows). The aim of the project is to predict the location in which a parking violation takes place given several other attributes.

Dataset Description:
The dataset is based on the Parking Tickets given due to parking violations in and around the region of New York City, New York, United States. Most of the columns of the dataset pertain to the conditions and time at which the parking violation occurred. On a quick glance, the majority of the columns in this dataset do not contain numeric or ordinal type data. Thus, numeric statistics of the dataset such as mean and standard deviation are not of importance.
The prediction column was 'Violation_County', which is assumed to be the location at which the parking violation occurred. It was first verified that this column has no null or missing entries. The rest of the columns were then analysed for feature engineering.

Data exploration, preprocessing and feature engineering

Presence of duplicate rows:
There exist 833,073 duplicate rows in the entire dataset. A sample of the duplicate rows is shown below

Summons Number	Plate ID	Registration State	Plate Type	Issue Date	Violation Code	Vehicle Body Type	Vehicle Make	Issuing Agency	Street Code1	Street Code2	Street Code3	Vehicle Expiration Date	Issuer Code	Issuer Command	Issuer Squad	Violat T
1367826652	5210319	IL	PAS	01/12/2015	21	SDN	TOYOT	S	16930	15780	30630	01/05/0001 12:00:...	587674	KS11	0000	08
1367826652	5210319	IL	PAS	01/12/2015	21	SDN	TOYOT	S	16930	15780	30630	01/05/0001 12:00:...	587674	KS11	0000	08
1367826652	5210319	IL	PAS	01/12/2015	21	SDN	TOYOT	S	16930	15780	30630	01/05/0001 12:00:...	587674	KS11	0000	08

After removing duplicate rows, we will end up with 17,676,346 distinct rows. To find the primary key, we look for that attribute which has as many distinct values as the number of distinct rows. 'Summons Number' attribute satisfies the condition for the primary key and is therefore chosen as the primary key of the dataset.

Presence of null columns:

Some of the columns in the dataset were populated, largely with or completely, with null (NaN) values. These columns had to be removed, so every column with more than 50% null value count was considered for deletion.
Columns dropped:
'Intersecting Street', 'Time First Observed', 'Time First Observed', 'Violation Legal Code', 'Unregistered Vehicle?', 'Meter Number', 'No Standing or Stopping Violation', 'Hydrant Violation', 'Double Parking Violation', 'Latitude', 'Longitude', 'Community Board', 'Community Council', 'Census Tract', 'BIN', 'BBL', 'NTA'

Dropping irrelevant columns:

As the Violation_County target feature is assumed to be related to location of violation occurrence, some of the columns in the dataset were considered as irrelevant for determining the same:
Summons Number: Unique identifying key, a numeric value with no relation to the data itself. Hence ‘Summons Number’ was dropped.
Plate ID: The plate id is generally arbitrary and is not relevant to the task, however the serial formatting of the plate id itself could give an idea about the location, as different states use different formats. The column ‘Vehicle Registration’ is assumed to be a better representation of this information. Hence ‘Plate ID’ was dropped.
Street Name: Street Name and other such name-based columns were removed as they had many unique values (each name is a unique entry) and no common patterns within the names that could be exploited to uncover a feature.
Vehicle Expiration Date: Some entries in this column had erroneous or malformed data entries, and the Expiration Date of a vehicle is not assumed to be relevant to the task. Hence, this column, and other such time-based features which were considered irrelevant were dropped.
House Number: House number is a code consisting of one or more values combined together. As such, a specific pattern or relevant categorization of this column was not possible. Hence it was considered irrelevant and dropped. Other such numeric-based columns which weren’t relevant to the task (uncovering location of violation) were dropped.
Features such as 'Sub Division', 'House Number', 'Street Name', 'Date First Observed', 'From Hours In Effect', 'To Hours In Effect', 'Vehicle Color', 'Days Parking In Effect', 'Violation Post Code', 'Violation Description', 'Vehicle Year', 'Feet From Curb', 'Issue Date' has no relevance with the location of violation and hence is dropped.
'Violation Time' is binned into 6 time intervals and a new column ‘Time Bin' is created, after which 'Violation Time' is dropped.
Thus, the complete list of columns removed based on above considerations:
'Summons Number', 'Plate ID', 'Violation Code', 'Vehicle Expiration Date', 'Sub Division', 'House Number', 'Street Name', 'Date First Observed', 'From Hours In Effect', 'To Hours In Effect', 'Vehicle Color', 'Days Parking In Effect', 'Violation Post Code', 'Violation Description', 'Vehicle Year', 'Feet From Curb', 'Issue Date', 'Violation Time'.

Data exploration on retained features

In total there are 13 retained features:

Registration State,	Plate Type,	Issue Date,	Violation Time,	Vehicle Body Type,
Vehicle Make,	Issuing Agency,	Street Code1,	Street Code2,	Street Code3,
Issuer Command,	Issuer Squad,	Violation In Front Of Or Opposite		

It is observed that all these columns represent some sort of categorical data. Even the ‘Street Code’ columns, which are technically integer values, are in effect a categorical attribute used to distinguish Street locations in and around the NYC area. In other words, the exact numeric value of the Street Code has no significance, nor does the ordinal position of the Street Code hold any importance. Therefore, they can be considered as categorical variables in this case.

Exploration of Registration State:

Registration State	count
NY	229706
NJ	29300
PA	7626
CT	4307
FL	3746
MA	2890
IN	2648
VA	2080
MD	1759
NC	1406

only showing top 10 rows

Inference: As expected, the majority of the violations were caused by vehicles registered in the State of New York. However, there is a significant portion of registrations in other states (primarily NJ). This may potentially hold some information about the Violation County (for example, New York City would more likely attract people from other states, than a less metropolitan region. Hence presence of an out-of-state registration may indicate a violation at NYC).

Exploration of Plate Type:

Plate Type	count
PAS	210526
COM	67632
OMT	7514
SRF	2614
OMS	2234
IRP	1995
999	1933
TRC	973
MOT	764
APP	640

only showing top 10 rows

Inference: It may be worthwhile to consider the different Plate Types as some plate types may be more common in certain regions.

Exploration of Issue Date:

Issue Date	count
01/15/2015	1156
06/30/2015	1148
01/20/2015	1125
01/13/2015	1097
01/23/2015	1085
01/16/2015	1075
01/22/2015	1073
06/26/2015	1027
06/29/2015	1007
01/21/2015	994

only showing top 10 rows

Inference: The Issue Dates are more uniformly distributed than other categories, which is why the top 10 frequencies are accounting for only around 3.5% of the total data used for exploration (300,000 rows). This would indicate that some feature engineering must be performed. The month and year of Issue Date may be separated into different columns.

Exploration of Violation Time:

Violation Time	count
0836A	959
1136A	919
1140A	862
0936A	856
0840A	786
1139A	786
1138A	773
0906A	765
0940A	758
0806A	755

only showing top 10 rows

Inference: The Violation Time is a pseudo-numeric feature, in the sense that although it is represented as string entries, it may be converted into a numeric distribution after accounting for the AM-PM time-shifts. Although the exact time of violation may not be very important, a rough idea of the time of violation would allow many entries to be grouped together. This would be useful for classification, e.g. all violations occurring in the early morning could be grouped together.

Exploration of Vehicle Body Type:

Vehicle Body Type	count
SUBN	93644
4DSD	80185
VAN	49686
DELV	26203
SDN	13785
PICK	7948
2DSD	7936
REFG	2781
TRAC	2482
UTIL	2442

only showing top 10 rows

Inference: This feature is not expected to be especially important, however it will still aid in distinction as certain vehicle types occur more frequently in certain regions.

Exploration of Vehicle Make:

Vehicle Make	count
FORD	39362
TOYOT	29222
HONDA	26834
CHEVR	23193
NISSA	21363
FRUEH	13831
DODGE	10130
INTER	10037
ME/BE	9832
BMW	9521

only showing top 10 rows

Inference: Like Vehicle Body Type, this feature is also not expected to be of high importance. However, an argument may be made that luxury type vehicles are more frequently made by certain manufacturers, and these luxury vehicles are more prevalent in metropolitan regions (such as inner NYC). Thus, this feature may indirectly provide information regarding the region of parking violation.

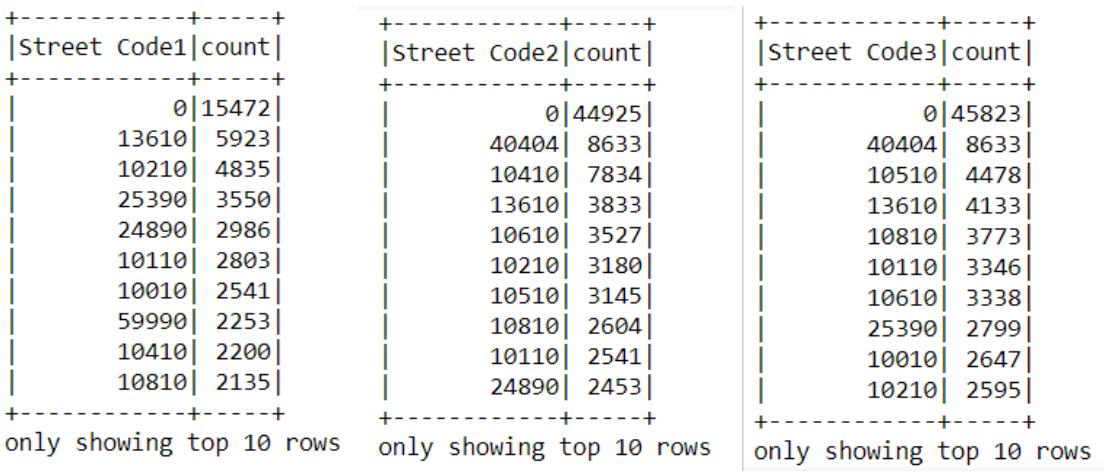
Exploration of Issuing Agency:

Issuing Agency	count
T	261639
P	27473
S	7227
X	3146
V	419
K	245
R	24
A	17
H	16
F	15

only showing top 10 rows

Inference: A clear idea of the information present in the Issuing Agency column in context of Parking Tickets was not obtained. However, it is assumed that the Issuing Agency may relate to the issuing authority or jurisdiction under which the parking ticket falls. It is speculated that this will indirectly help in identifying the region in which the violation occurred.

Exploration of Street Codes:



Street Code1	count
0	15472
13610	5923
10210	4835
25390	3550
24890	2986
10110	2803
10010	2541
59990	2253
10410	2200
10810	2135

only showing top 10 rows

Street Code2	count
0	44925
40404	8633
10410	7834
13610	3833
10610	3527
10210	3180
10510	3145
10810	2604
10110	2541
24890	2453

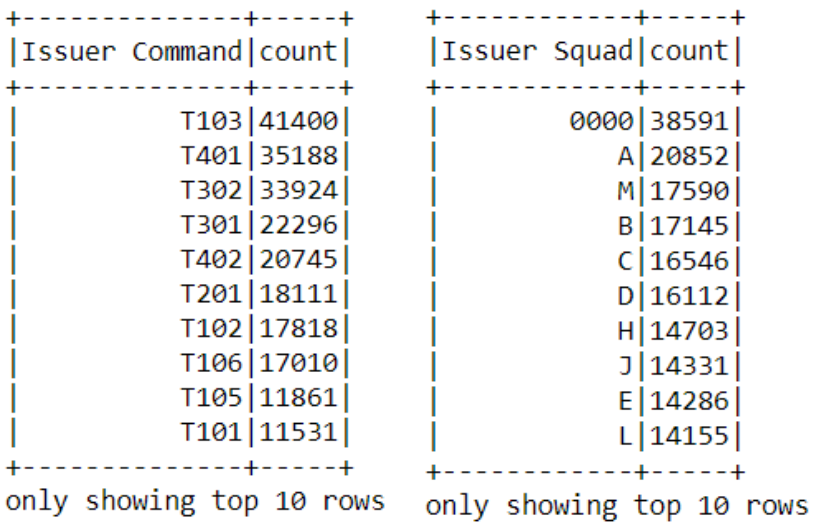
only showing top 10 rows

Street Code3	count
0	45823
40404	8633
10510	4478
13610	4133
10810	3773
10110	3346
10610	3338
25390	2799
10010	2647
10210	2595

only showing top 10 rows

Inference: The 0 value entry in these Street Codes may signify missing values. Moreover, the Street Code 2 and Street Code 3 entries have more missing values. This is expected as some street codes may be of shorter length. A key observation is that despite the Street Codes being a distribution of 5 digit numbers, many of the street codes fall under the same value (such as the most frequently encountered '13610'). This indicates that grouping of street codes into categories by most frequent occurrence could provide insights as to the location of violation.

Exploration of Issuer Command:



Issuer Command	count
T103	41400
T401	35188
T302	33924
T301	22296
T402	20745
T201	18111
T102	17818
T106	17010
T105	11861
T101	11531

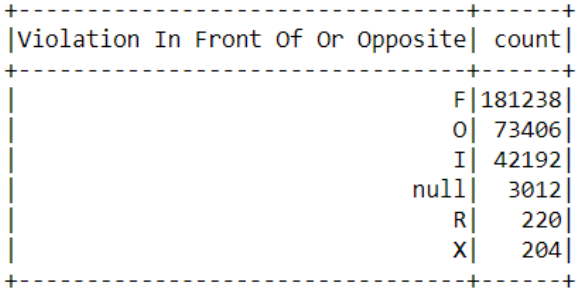
only showing top 10 rows

Issuer Squad	count
0000	38591
A	20852
M	17590
B	17145
C	16546
D	16112
H	14703
J	14331
E	14286
L	14155

only showing top 10 rows

Inference: It is likely that the Issuer Command and Squads operate in designated regions in and around NYC. Therefore, the ticket issuer command and squad will be a good indicator of the region where the violation occurred.

Exploration of 'Violation In Front of Or Opposite'



Violation In Front Of Or Opposite	count
F	181238
O	73406
I	42192
null	3012
R	220
X	204

Inference: It is assumed that this column describes some information regarding the position of the vehicle which caused the violation. This may not be very relevant to the task of determining the region of violation, nevertheless this column is kept in the feature columns list as it may be possible that some labels such as 'I' instead of 'F' (both of which probably denote 'In Front Of') may be more commonly used by issuing authorities in particular regions.

Feature Transformation and handling of missing values

Pyspark UDF was applied on the 'Issue Date' feature to extract the day, month and year following which new features called 'Issue Day', 'Issue Month' and 'Issue Year' were created.

Distribution of these new features on Exploration Subset:

Issue Month	count
1	29814
3	26495
10	26425
8	26077
9	25547
6	25351
7	25270
5	25231
4	24903
11	23305

only showing top 10 rows

Issue Day	count
3	54145
5	53465
6	51033
4	49809
2	47000
7	33256
1	11564

Issue Year	count
2015	158264
2014	76774
2016	65180
2013	14
2017	14
2018	6
2000	6
2019	3
2012	3
2010	2

only showing top 10 rows

The Issue Month is uniformly distributed across all months. In this case, the most frequently occurring month is not of relevance as every month occurs with near equal frequency. Upon observing the issue year, the majority of the tickets seem to have been issued in the years 2014,2015 and 2016, with 2015 being the most frequent. The Issue Day suggests that the majority of the tickets are issued in the first 7 days of the month. However, no other conclusion may be derived from the Issue Day, so this column is not retained. The other two columns, Issue Month and Issue Year are retained for the training.

Pyspark UDF was also applied to bin 'Violation Time' into 6 bins. The bins are grouped as follows:

Violation Time	Bin
00:00 AM (or 12:00 AM) to 04:00 AM	1
04:00 AM to 08:00 AM	2
08:00 AM to 12:00 PM	3
12:00 PM to 04:00 PM	4
04:00 PM to 08:00 PM	5
08:00 PM to 00:00 AM (or 12:00 AM)	6

A new feature named 'Time bin' is thus formed after which 'Violation Time' is dropped.

The resultant distribution in the exploration dataset:

```
+------------------+------+
|Violation Time bin| count|
+------------------+------+
|                 3|119859|
|                 4| 97056|
|                 5| 35758|
|                 2| 25227|
|                 6| 11763|
|                 1| 10609|
+------------------+------+
```

The majority of the violations seem to occur in Bin 3, that is between 8 AM to 12 PM. In comparison, violations in the early morning (bin 1) or late at night (bin 6) are least frequent. Thus, this binning strategy allows for better interpretation than the original time entries themselves.

In the presence of missing value in any of the features that are retained, we use the most frequently occurring value [that is found from data exploration] of that corresponding feature as a proxy for the default value of that feature. An exception is the “Issue Month”, which had evenly distributed values and so the mean was taken. The default values are compiled in the table below:

Retained Feature Name	Data Type	Default Value
Registration State	String	‘NY’
Plate Type	String	‘PAS’
Issue Month	Integer	6
Issue Year	Integer	2015
Vehicle Body Type	String	‘SUBN’
Vehicle Make	String	‘FORD’
Issuing Agency	String	‘T’
Street Code1	Integer (Categorical)	0
Street Code2	Integer (Categorical)	0
Street Code3	Integer (Categorical)	0
Issuer Command	String	‘T103’
Issuer Squad	String	‘A’
Violation In Front Of Or Opposite	String	‘F’
Time bin	Integer	3

Thus, the resultant dataset was used for training. The Categorical and String data retained above were indexed using the StringIndexer.

Model Training

Model Pipeline:

Input ➔ String Indexers ➔ Vector Assembler ➔ Model ➔ Inverse String Indexer

Random Forest:

```
In [33]: print(f"Train F1: {trainF1:.5f}\nTest F1: {testF1:.5f}")
         print(f"Train accuracy: {trainAccuracy:.5f}\nTest Accuracy: {testAccuracy:.5f}")
```

```
Train F1: 0.94311
Test F1: 0.94349
Train accuracy: 0.94563
Test Accuracy: 0.94561
```

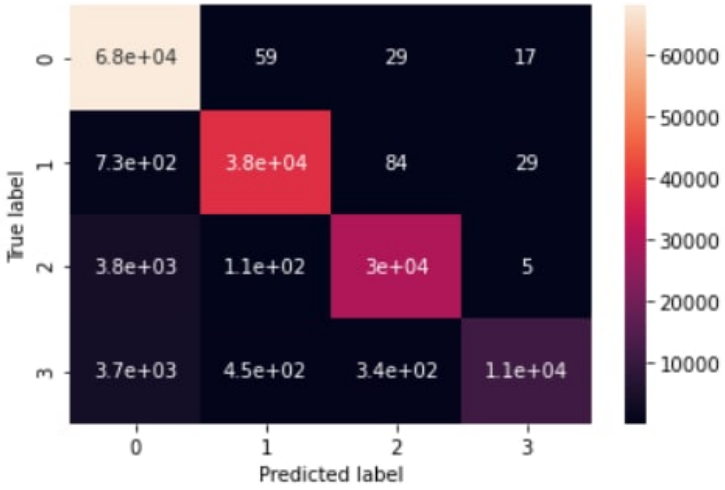


Fig. Confusion Matrix of RF model on a test data of size 300k.

Logistic Regression:

```
In [*]: print(f"Train F1: {trainF1:.5f}\nTest F1: {testF1:.5f}")
        print(f"Train accuracy: {trainAccuracy:.5f}\nTest Accuracy: {testAccuracy:.5f}")
```

```
Train F1: 0.26676
Test F1: 0.26527
Train accuracy: 0.43622
Test Accuracy: 0.43668
```

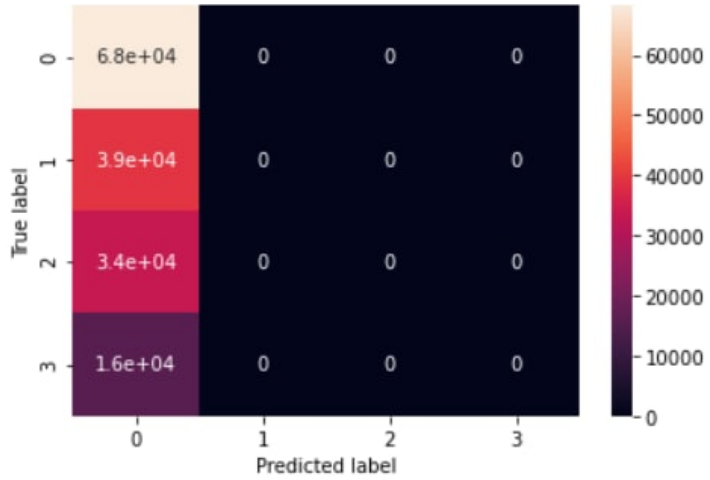
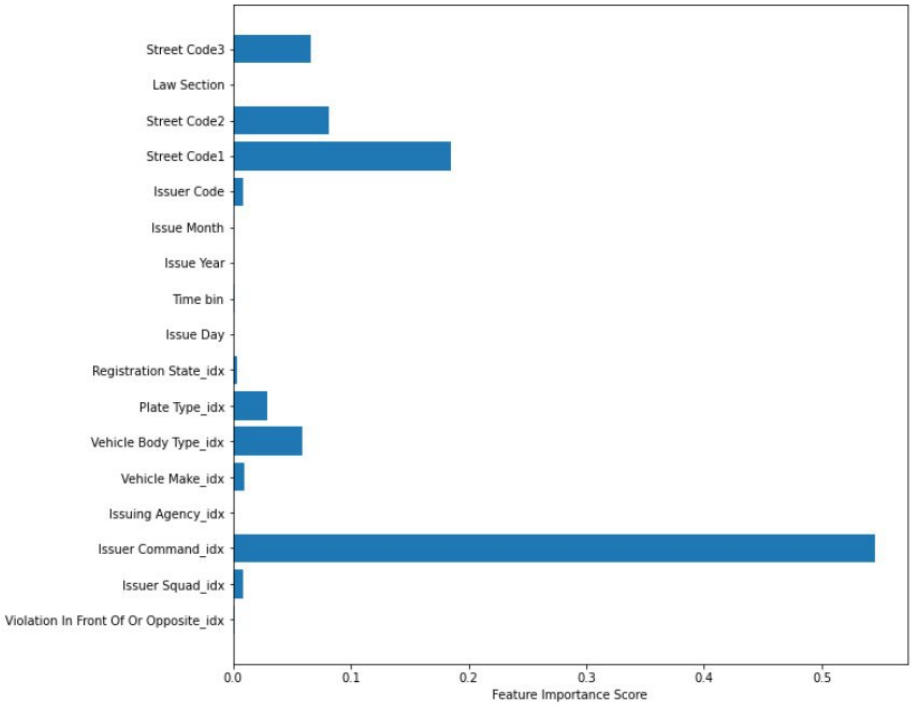


Fig. Confusion Matrix of LR model on a test data of size 300k.

Clearly random forest performs much better than logistic regression and this can be attributed to the following three reasons.

1. Logistic regression performs poorly because the data is not linearly separable. On the other hand, random forest, by being a non linear classifier, is suited for this data.
2. As we have many categorical features, random forests would perform better than logistic regression in general.
3. From the confusion matrix plots, we can infer that the Logistic Regression model predicts 0 all the time whereas Random Forest manages to correctly predict fairly across all classes. This indeed implies that since Logistic Regression is a linear classifier, training on non-linear data will make the model predict the most frequent class, which is 0 (mapped to 'NY').

Feature importance:



Location related features such as Street Code 1, Street Code 2 and Street Code 3 have more importance whereas features such as registration state, issue month and issue year have least importance. The above plot also suggests that Issuer Command feature is the most important feature to predict Violation County. A possible explanation for this is that each county has its own set of issuer commands, and so each entry of Issuer Command very likely indicates that the violation occurred in their designated county.

Real time processing and latency of each window

Since random forest was observed to perform better, we used this model for real time processing. The trained model was saved in google cloud storage. Kafka was integrated with Spark Structured Streaming for real time processing. The code for real time production and subscription can be found in *producer.py* and *subscriber.py*. A Dataproc job for subscription was submitted and this listens to the topic *test*. The producer is run in Kafka Virtual Machine and streams the data to be processed row by row to *test*. The screenshots of successful job execution is found below

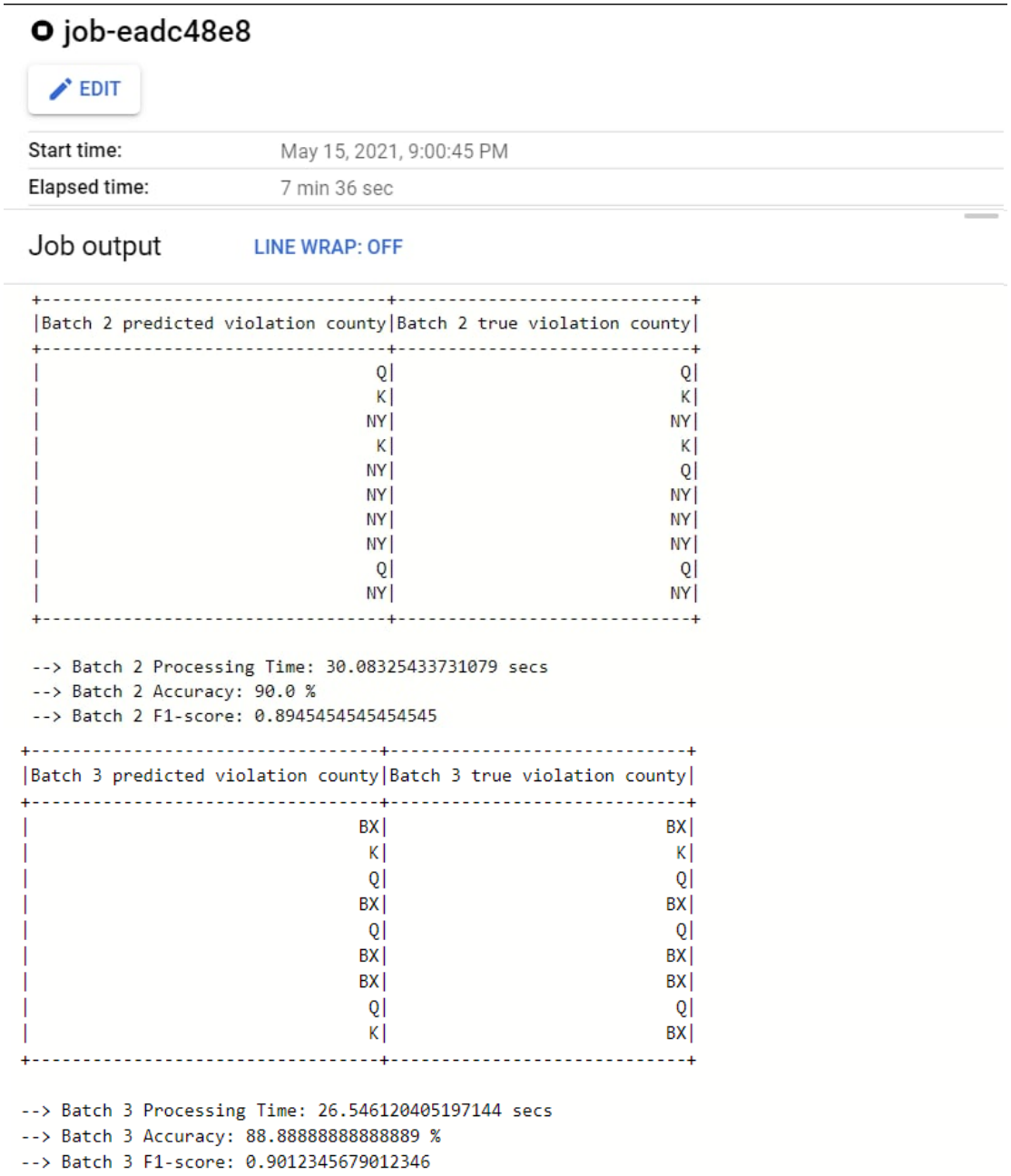


Fig. Sample output of real time prediction using trained random forest model.

Prediction accuracy and F1-score is calculated for every batch along with the processing time (latency) of each batch. We might also be interested in knowing the average processing time per batch. For a sample containing 200 rows, we run real time streaming and get the average latency per window of about **31 seconds**.

The problem of predicting the region of violation is not assumed to be a performance-critical task, and hence the latency of around 31 seconds is considered to be acceptable for any potential use-cases. It is also noted that this latency is in part due to the overheads caused by scheduling and distributing PySpark tasks, and also real-time analysis of accuracy and f1-score. These components may not contribute to the complexity as much when the problem is scaled to larger batches, where the latency is mainly dependent on the speed of classification.

Conclusion

Through the course of the project, the following objectives were satisfied:

- Data retrieval and analysis of the NYC Parking Tickets Dataset on the Google Cloud Platform using the PySpark library, employing a collaborative workflow engineered towards reliable and efficient handling of big data on Distributed File Systems.
- Training and Predictions on Violation County of the dataset implemented through PySpark ML Libraries, on a Dataproc Cluster utilising different ML models.
- Provision for real-time data processing and predictions through Structured Streaming in a fault-tolerant, highly scalable environment through the use of Kafka.

The problem of Big Data Analysis lies not only in the sheer amount of data but also on the structure, complexity, completeness and storage. Using Spark for data processing and HDFS for storage, the accessibility and interoperability of the datasets available are maximised. Through diligent data exploration and careful feature engineering, even a dataset as large as 10 million entries may be utilised for analysis and real-time predictions.