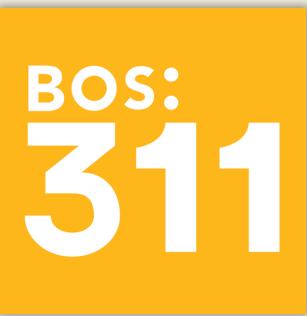


# A COMPARATIVE STUDY OF MACHINE LEARNING ALGORITHMS FOR PREDICTING BOSTON 311 RESPONSE TIME

DS504 FINAL PROJECT  
YANG FU  
2019-10-24

# PROJECT DESCRIPTION



- BOS:311 provides Bostonians with access to report non-emergency issues and request services.
- The estimation of the processing time for a service request is barely found in the BOS:311 system but would be useful and preferred by users.
- In this project, I would like to explore the possibility of predicting the BOS:311 service request processing time using multiple machine learning algorithms.

# GOALS

- Data processing.
- Exploratory data analysis.
- Modeling.
- Model evaluations and comparison.

# DATA PROCESSING

- Dropped missing data.
- Extracted/Created potentially useful features/targets.
- Merged with the Boston weather dataset.

# DATA PROCESSING - FEATURES

```
1 raw.shape  
(1611147, 29)  
  
1 raw.dtypes  
  
case_enquiry_id          int64  
open_dt                  object  
target_dt                object  
closed_dt                object  
ontime                  object  
case_status              object  
closure_reason           object  
case_title               object  
subject                 object  
reason                  object  
type                     object  
queue                   object  
department              object  
submittedphoto           object  
closedphoto              object  
location                 object  
fire_district            object  
pwd_district             object  
city_council_district    object  
police_district          object  
neighborhood              object  
neighborhood_services_district object  
ward                     object  
precinct                 object  
location_street_name     object  
location_zipcode         float64  
latitude                 float64  
longitude                float64  
source                   object  
dtype: object
```



```
In [8]: 1 w = pd.read_csv('weather.csv')  
2 w.head()  
  
Out[8]:  
  
   Year Month Day Avg Temp (F) Precip (in)  
0  2013     1   1       29      0.0  
1  2013     1   2       21      0.0  
2  2013     1   3       16      0.0  
3  2013     1   4       30      0.0  
4  2013     1   5       35      0.0
```



```
1 new_df.shape  
(936602, 13)  
  
1 new_df.dtypes  
  
reason                  object  
department              object  
neighborhood             object  
latitude                 float64  
longitude                float64  
source                  object  
duration                float64  
open_y                  int64  
open_m                  int64  
open_day_of_week          int64  
Avg Temp (F)             int64  
Precip (in)               float64  
class                   int64  
dtype: object
```

# DATA PROCESSING - TARGET

- The target is the duration (processing time in hours) of a service request from open to close.
- The target is numeric by nature.

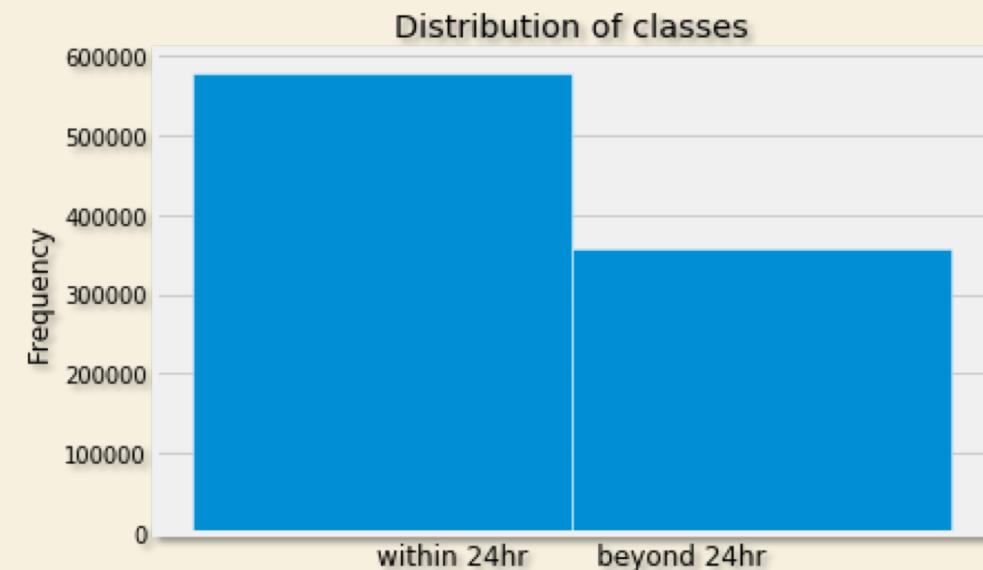
1	new_df.duration.describe()
count	937454.000000
mean	226.835018
std	1092.388273
min	0.000000
25%	1.150000
50%	14.166944
75%	74.523333
max	54630.305556
Name:	duration, dtype: float64

# DATA PROCESSING - TARGET

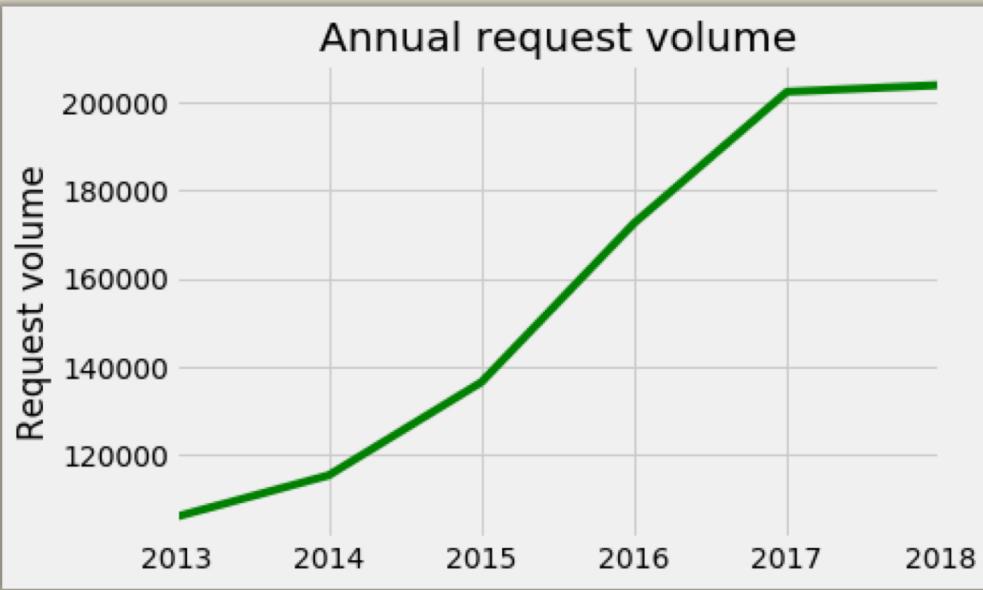
- The distribution of the target values is extremely skewed.
- The target were grouped into within/beyond 24hr classes.

```
1 new_df.duration.value_counts(bins=10)

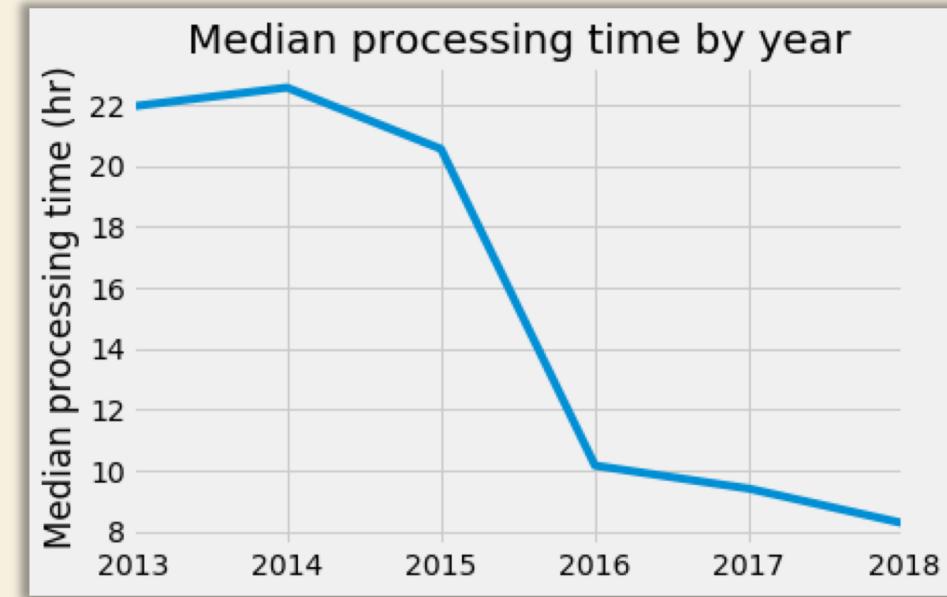
(-2.276999999999997, 227.626]      927623
(227.626, 455.253]                  8360
(455.253, 682.879]                  970
(682.879, 910.505]                  275
(910.505, 1138.131]                 110
(1138.131, 1365.758]                 52
(1365.758, 1593.384]                 34
(1593.384, 1821.01]                  24
(2048.636, 2276.263]                 3
(1821.01, 2048.636]                 3
Name: duration, dtype: int64
```



# EXPLORATORY DATA ANALYSIS

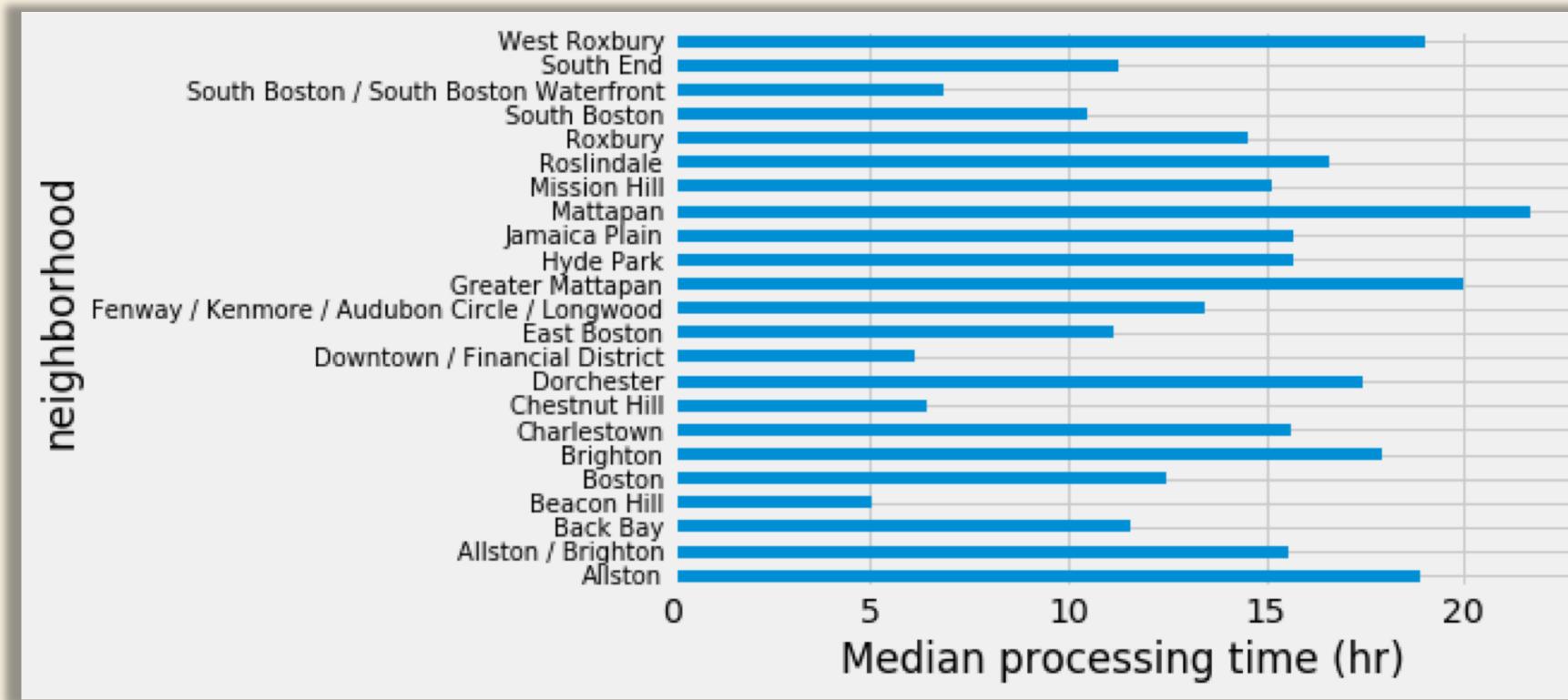


311 usage has been rising as the city introduced various 311 request channels.



311 request processing time has been decreasing probably due to improved city services.

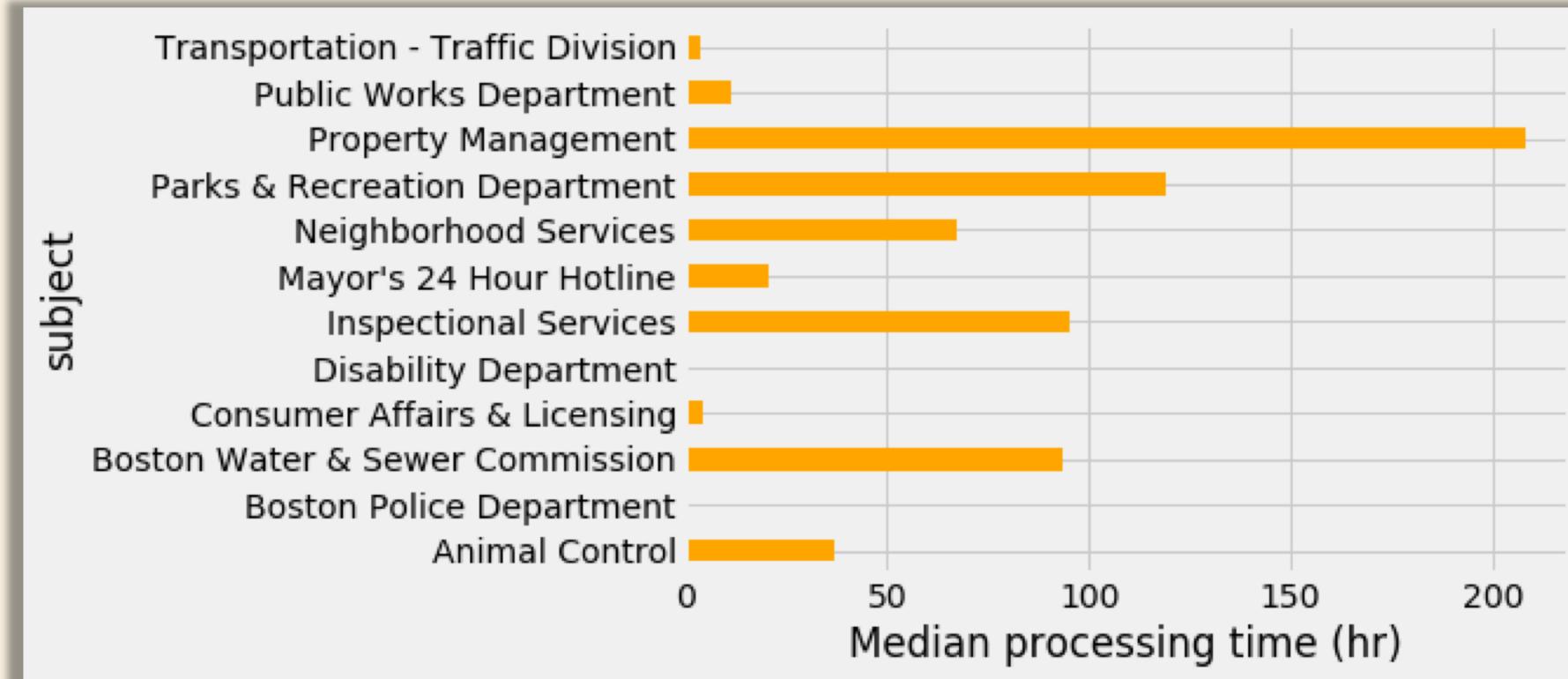
# EXPLORATORY DATA ANALYSIS



Processing times vary by neighborhood.

Beacon Hill is the fastest while Mattapan is the slowest.

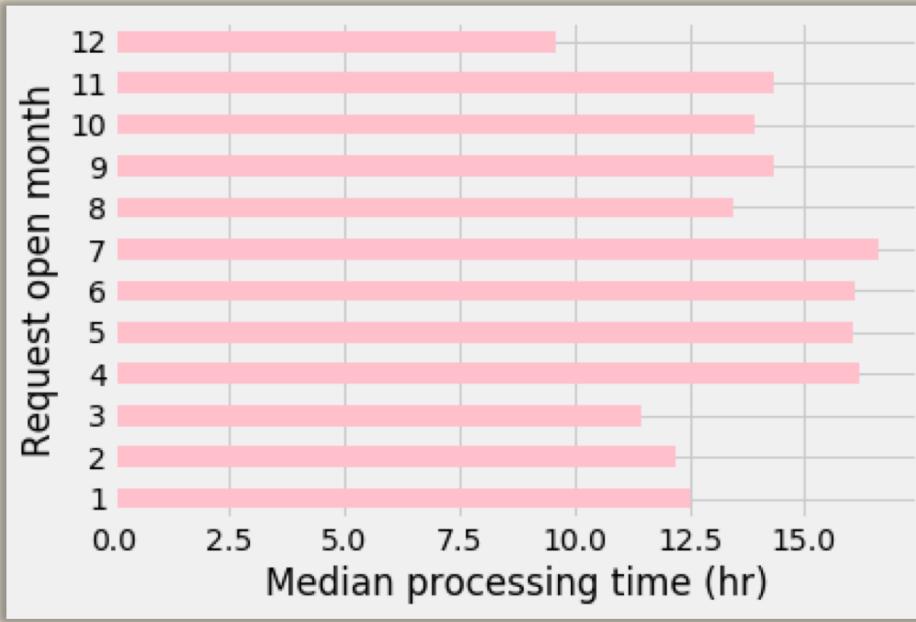
# EXPLORATORY DATA ANALYSIS



Processing times vary by subject.

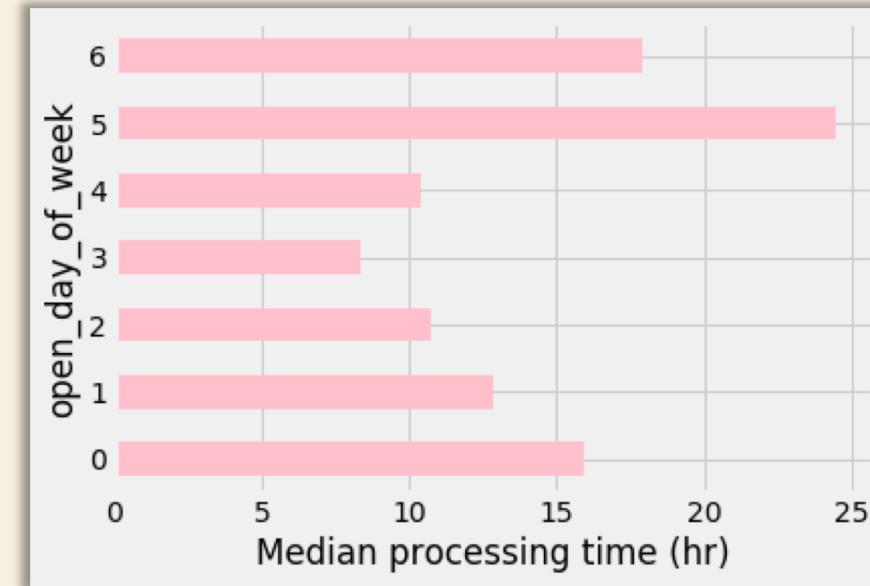
Property Management related services were the most time-consuming.

# EXPLORATORY DATA ANALYSIS



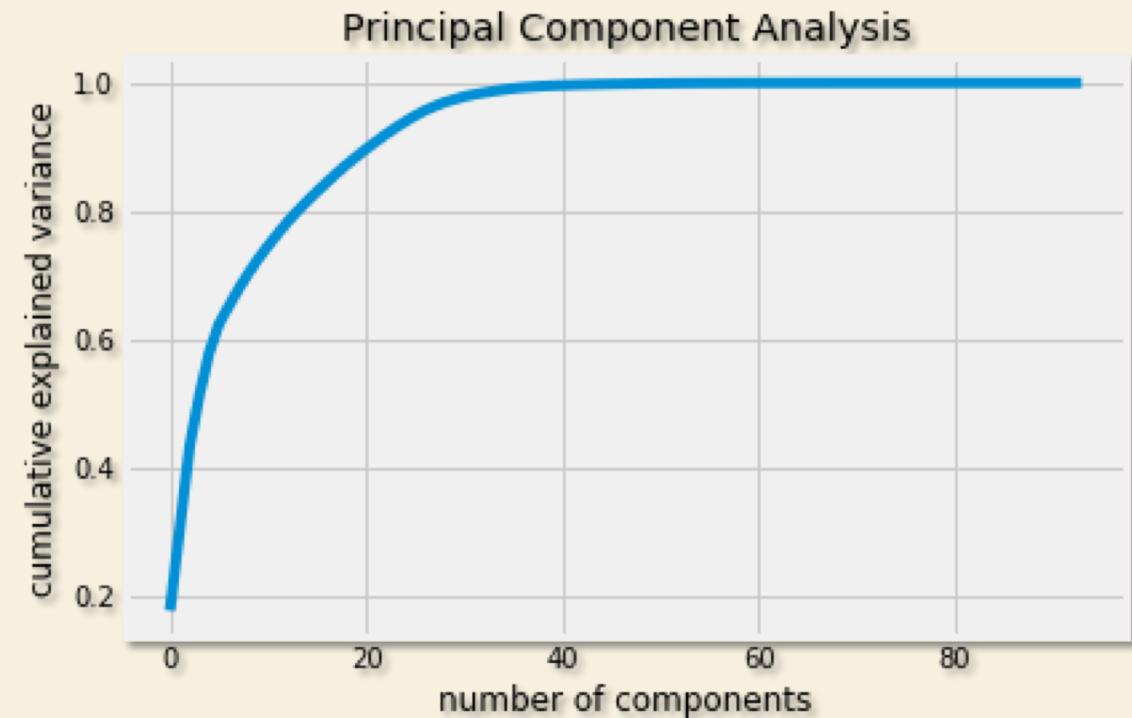
Processing time varies by request open month.

Processing time varies by open day of week.  
Weekend requests are processed slower.



# PRINCIPAL COMPONENT ANALYSIS

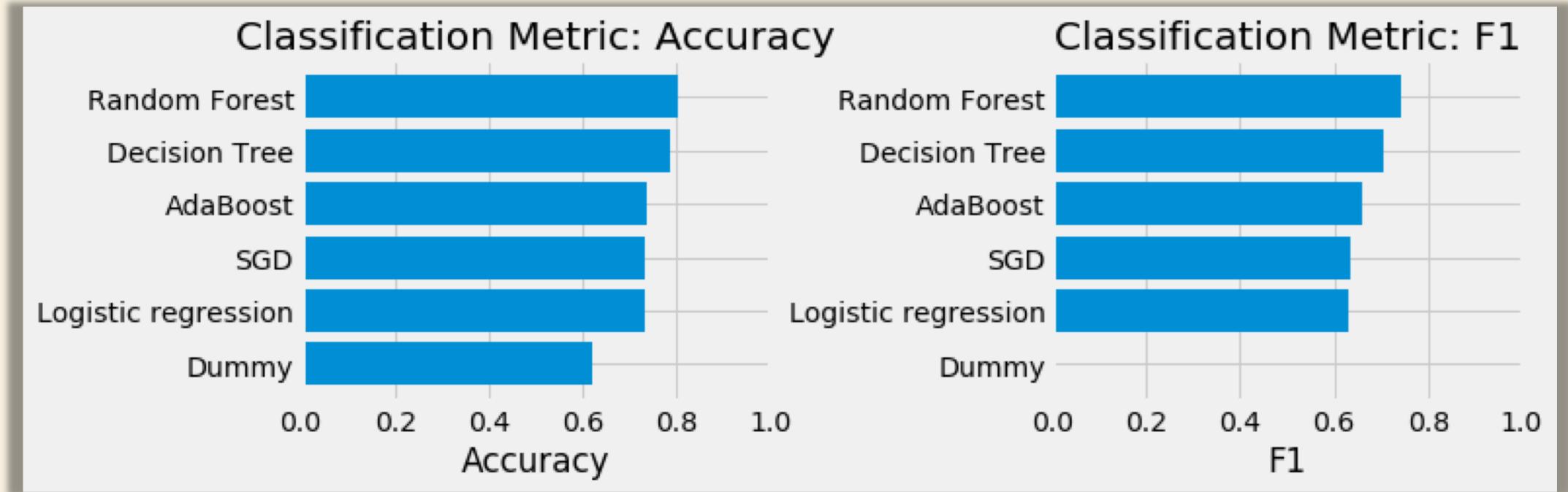
- The encoded dataset contains 107 features.
- The first 20 PCs can explain more than 90% variance.
- The first 20 PCs were applied to the classification modeling.



# MODELING – CLASSIFICATION

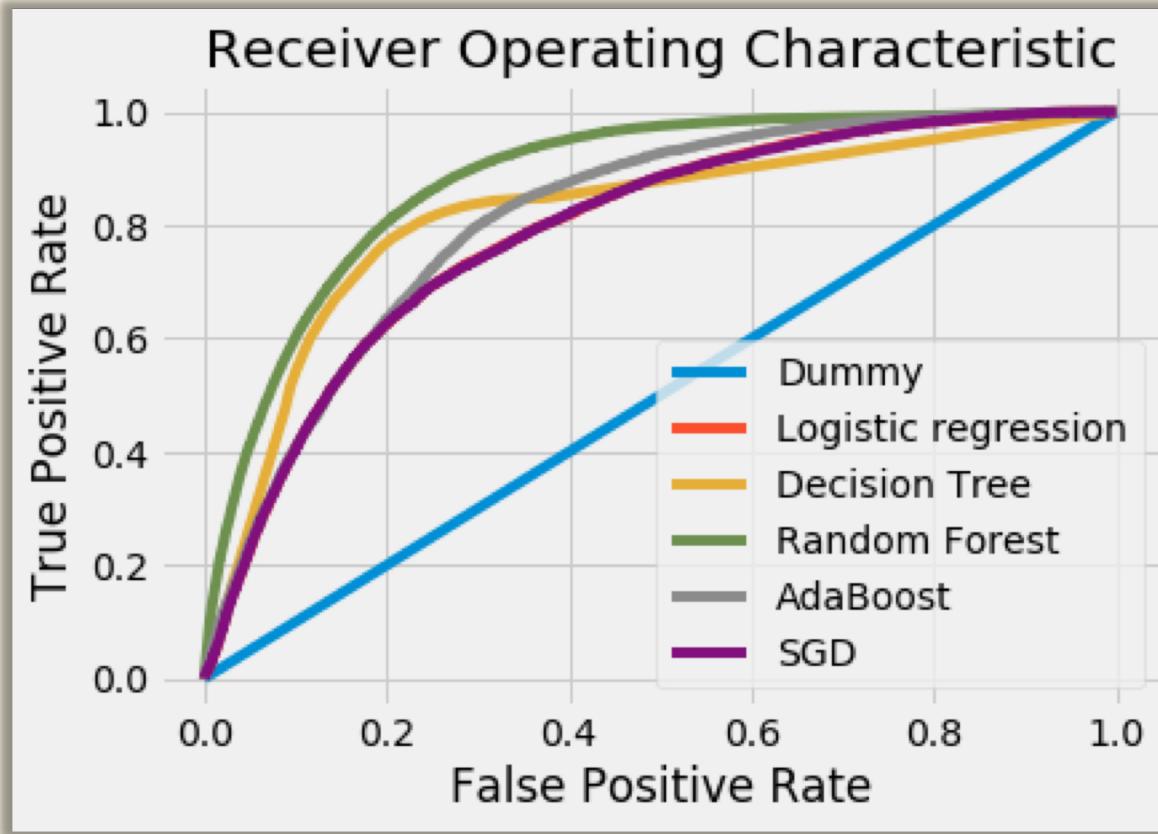
- Predicted whether a request could be resolved within 24hr.
- A dummy classifier was used as baseline control.
- All classifiers were briefly tuned by cross validation.
- Multiple error metrics were included to evaluate model performance:
  - Accuracy.
  - F1 score.
  - AUC.

# MODELING – CLASSIFICATION



- The Random Forest classifier has the best performance, with an accuracy of 0.8, and a F1 score of 0.74.

# MODELING – CLASSIFICATION



- The Random Forest classifier has the best performance, with the max area under ROC curve 0.79.

# MODELING – REGRESSION

- Converted duration from hours to days.
- Focus on the most favorite service request ‘reason’ – Sanitation.
- **Predicted how many DAYS are needed to resolve a Sanitation service request.**

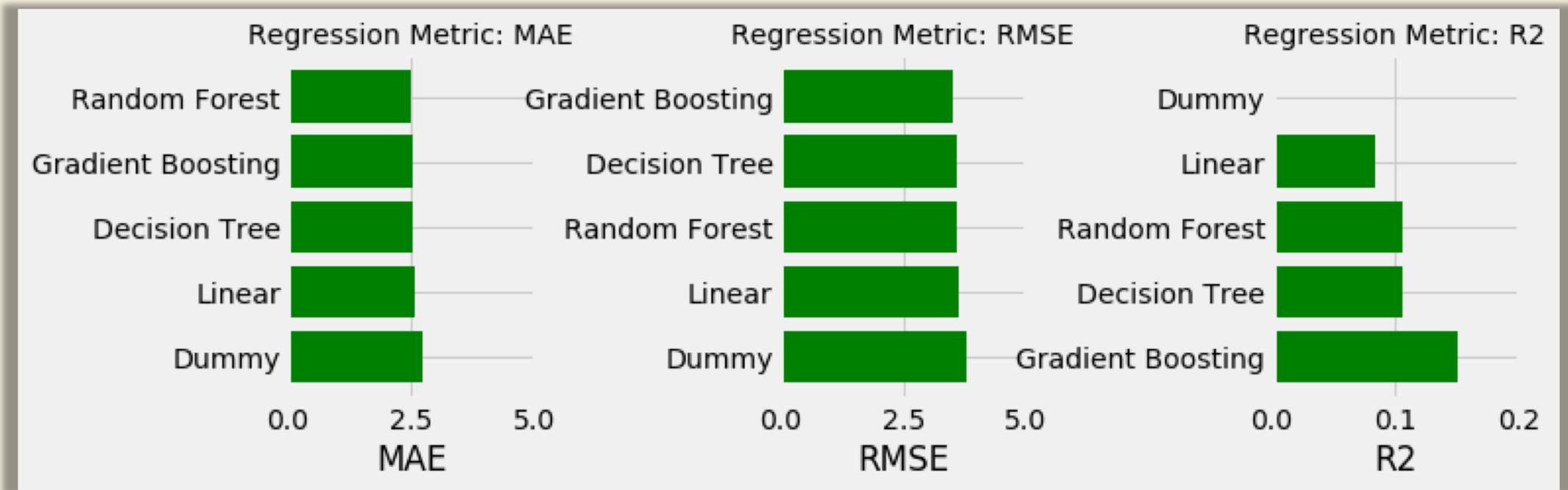
```
1 # convert duration from hours to days.  
2 new_df[ 'duration' ] = new_df[ 'duration' ]/24  
3 new_df[ (new_df[ 'reason' ]=='Sanitation') ][ 'duration' ].describe()  
  
count      171069.000000  
mean        3.386809  
std         4.126719  
min         0.000093  
25%        0.651840  
50%        2.287998  
75%        5.327789  
max        633.062025  
Name: duration, dtype: float64
```

# MODELING – REGRESSION

- A dummy regressor was used as baseline control.
- All regressors were briefly tuned by cross validation.
- Multiple error metrics were included to evaluate model performance.
  - Mean Absolute Error.
  - Root Mean Square Error.
  - R Square.

```
Dummy
Mean absolute error: 2.74
Mean squared error: 3.79
R2: -0.00
=====
Decision Tree
Mean absolute error: 2.53
Mean squared error: 3.59
R2: 0.11
=====
Random Forest
Mean absolute error: 2.49
Mean squared error: 3.59
R2: 0.11
=====
Gradient Boosting
Mean absolute error: 2.52
Mean squared error: 3.49
R2: 0.15
=====
Linear
Mean absolute error: 2.58
Mean squared error: 3.63
R2: 0.08
=====
```

# MODELING – REGRESSION



- In general, regressors failed this task.
- The min MAE is as large as 2.5, the min RMSE is as large as 3.5, and the max R2 is as tiny as 15%.

# PYSPARK ON DATABRICKS

- The same project was rewritten in PySpark and run on Databricks, without feature scaling and PCA.
- Very similar results were observed.

## Classifiers

Logistic regression

Accuracy: 0.77

=====

Random forest

Accuracy: 0.75

=====

GBT

Accuracy: 0.80

=====

MPC

Accuracy: 0.77

=====

## Regressors

Linear regression

R2: 0.11

=====

Decision tree

R2: 0.08

=====

Random forest

R2: 0.09

=====

GBT

R2: 0.14

=====

# CONCLUSIONS

- Among all the **classifiers** tested, the Random Forest classifier performs the best in deciding whether a request can be resolved within 24 hrs with an accuracy of 0.8 and an F1 score of 0.74.
- All **regressors** tested fail to explain the variance in this dataset. However, the MAE and RMSE could still be used as references when predicting the processing time of a service request under certain ‘reason’.

# FUTURE DIRECTIONS

- Predicting BOS:311 service request processing time using various machine learning algorithms is doable but the results are not so great in this project. (And I now understood why such information is missing in the system.)
- Future work could focus on better feature engineering, as the current dataset contains too little numeric features.

# REFERENCES

- Data source:
  - <https://data.boston.gov/dataset/311-service-requests>
  - <https://www.kaggle.com/jqpeng/boston-weather-data-jan-2013-apr-2018>
- Code: [https://github.com/ilovemanu/ds504\\_big\\_data\\_analytics](https://github.com/ilovemanu/ds504_big_data_analytics)
- Tutorials:
  - <https://spark.apache.org/docs/latest/ml-classification-regression.html>
  - <https://docs.databricks.com/getting-started/index.html>

**THANKS**