# Big Data Mining for Smart Cities: Predicting Traffic Congestion using Classification

Aristeidis Mystakidis
The Data Mining and Analytics Research Group
School of Science & Technology
International Hellenic University
Thessaloniki, Greece
a.mystakidis@ihu.edu.gr

Christos Tjortjis
The Data Mining and Analytics Research Group
School of Science & Technology
International Hellenic University
Thessaloniki, Greece
c.tjortjis@ihu.edu.gr

***Abstract*—**This paper provides an analysis and proposes a methodology for predicting traffic congestion. Several machine learning algorithms and approaches are compared to select the most appropriate one. The methodology was implemented using Data Mining and Big Data techniques along with Python, SQL, and GIS technologies and was tested on data originating from one of the most problematic, regarding traffic congestion, streets in Thessaloniki, the 2nd most populated city in Greece. Evaluation and results have shown that data quality and size were the most critical factors towards algorithmic accuracy. Result comparison showed that Decision Trees were more accurate than Logistic Regression.

***Keywords* —** *Data Mining, Big Data, Machine learning, Smart Cities, Prediction, Classification, Traffic Congestion*

## I. Introduction

These days, more and more data regarding cities and urban areas are produced. This information is critical for automating several procedures, such as road traffic control. With urban living increased exponentially in the last century, road traffic congestion has become a significant problem of our era. Moreover, it is common sense, that high traffic congestion may affect other areas such as the environment, health, the economy etc. [1]. There is no panacea, but as far as the solution for this problem is concerned, analyzing congestion data for future traffic prediction could make a significant difference.

It is predicted that by 2050 more than 67 percent of the total global population will live in urban areas [2]. This report also indicates that from 1950 to 2018, urban living exponentially increased from 751 million to 4.2 billion people.

This overpopulation, with the large amount of data which record smart cities procedures and functions broaden the horizon of data science for smart cities. These data can be analyzed for the sake of each city's procedure optimization. One of these procedures that needs to be optimized is traffic flow, which causes many issues related to several social and economic sectors [1].

Our paper researches and develops a prototype model to forecast traffic congestion. We evaluated this model in Thessaloniki, Greece, a highly populated city with average at best public transport infrastructures and no metro subway, with Tsimiski Street (one of the busiest main roads in the city) as a case study. We present a step by step methodology, describing the data engineering part, the data transformation part, and the data mining part.

An important aspect of this work relates to results and evaluation, including a detailed comparison of different classifiers, used for traffic congestion prediction. Furthermore, the most accurate classifier scored over 73% on accuracy, utilizing however a low-cost methodology for data extraction.

The remaining of the paper reviews related literature in section II, describes the methodology in section III, presents results and evaluation in section IV and concludes with directions for future work in section V.

## II. Background

### A. Smart City applications

Big data mining and machine learning for smart cities utilization assists in solving production, transport, and traffic management problems in real-time approaches using frameworks and systems that are incorporated and provide data transfer efficiently through apps and stakeholders [3].

There are several cases of smart cities supporting big data mining technologies like Smart education [4], Smart grid [5] and electricity consumption prediction [6], [7]. However, an important aspect relates to Smart traffic lights [8] and traffic congestion management and prediction [9], [10], [11].

A key smart city feature is efficient traffic flow management, that could boost transportation networks flow and optimize traffic conditions for people and cities in general [8]. As population grows, there are traffic issues, increased emissions, and environmental and economic issues. Because of the above, the utilization of relevant strategies (for example traffic congestion prediction or smart traffic lights) used by smart cities to deal with increasing traffic congestion problems is very important.

The cost of many of these techniques, however, as well as the costs of the necessary technology related to then appears to be very high [9]. For example, one strategy in order to provide enough data on traffic patterns, smart traffic lights and signals must be integrated throughout traffic grids. Every sensor could measure a particular traffic flow variable like vehicle velocity, traffic density, lights waiting time, distances, etc. The number of sensors needed for this would be many as well as the multiplied cost for all of them.

### B. Traffic congestion prediction and related work

Real- or future-time traffic jam knowledge would be very important for congested and overpopulated locations.

978-1-7281-4959-2/19/$31.00 ©2019 IEEE

smart or Intelligent Transportation Systems (ITS) could help develop certain congestion data reports via sophisticated prediction algorithms.

Many programs were developed and introduced both by corporate and state agencies to collect traffic information to supply ITS systems. According to [9], many actions are based on selective setup of static detectors like loop coils and smart video processing recorders. Considering the cost of hardware, plus setup and maintenance costs, the expenses of such technologies seem to be quite heavy [9].

In addition, such static detectors in most locations are exposed to severe weather events. What is more, it is hardly cost effective or technically feasible to deploy fixed detectors to cover all streets in major urban areas. Therefore, an alternative method for collecting traffic information and predicting traffic congestion with greater coverage at a reduced cost is required.

Estimation methodologies for the congestion rate can differ according on the characteristics of the data gathered. There are two different kinds of detectors that can indeed automatically collect traffic data: a fixed sensor and a mobile detector. Street cameras, speedometers etc., considered to be fixed sensors/detectors while mobile phones, vehicle GPS considered to be mobile sensors/detectors.

The research in [12] utilized the algorithm of the neural network from the mobile phone data gathered. this study utilized Cell Dwell Time (CDT), the moment a mobile phone connects to a mobile phone service antenna, offering a rough travel velocity, combined with video recording of traffic from the driver's perspective. The accuracy of the model was 79%. In [9] the authors used another machine learning methodology that was better suited to the data morphology getting very high accuracy (91%).

GPS measurements could offer further reliable traffic information rather than the CDT results. This study utilised a decision tree (J48/C4.5) Classification approach on mobile sensors to identify road traffic congestion rates from GPS data, but also capturing and utilizing images of road traffic conditions by a video camera. Getting data via mobile sensors could monitor much broader areas of traffic. The algorithm would indeed learn about vehicle's motion patterns. The fixed window sliding methodology was also utilized. The studies of [13] and [14] predicted the rate of traffic jams by implementing fuzzy logic and hidden Markov methods, respectively, utilizing traffic camera information.

In addition, the studies reported in [15], [16] and [17] explored numerous alternative approaches related to traffic congestion research. For instance, in several countries, as shown in the surveys of [18] and [19], the key criteria utilized to describe traffic congestion rates are duration, velocity, size, quality of service, and the traffic signal periods that drivers need to stop for.

As it can been seen, there are several highly efficient traffic congestion prediction models, using however costly detectors like cameras to capture video recordings and images in order to score high accuracy. With that being mentioned, it is clear that the challenge is to develop a cost-effective way to deal with traffic congestion management and manage to utilize accurate traffic congestion prediction methods using machine learning algorithms without using costly detectors and cameras.

## III. METHODOLOGY

The aim of this work was to deal with traffic congestion management problems. Our main strategy was to develop data mining/machine learning techniques for traffic congestion prediction in order to predict the traffic depending the street, the specific time etc.

This would help drivers to avoid high traffic and choose routes with better flow. Another advantage of utilizing this strategy could be that in a city, ridden with transportation problems, the use of ITS based on the traffic prediction algorithms could make a difference. What is more, based on these algorithms, smart traffic lights mechanisms could be utilized on future studies.

The idea was to extract data from a specific API, get data from json array, use several technologies and techniques for data pre-processing and mining to predict traffic. For this project three main tasks have been combined.

1 Data engineering phase
2 Data pre-processing phase
3 Classification for traffic congestion forecasting.

### A. *Data engineering*

There were 3 different sources regarding data extraction.

Traffic congestion data: Traffic congestion data originated from http://opendata.imet.gr/dataset/network-congestion. This is a website that contains traffic data regarding speed, congestion, road links, travel times, historical data etc. about the city of Thessaloniki. Data were gathered from various sources [20], [21], [22]. In the json file for congestion we collected data including Link_id, Link_Direction, Timestamp, Congestion using the urllib library.

Emisia Database: This database, offered by Emisia S.A., was created for an application called Wiseride. This App used to provide real time data from osmosis database (a database that contains data from taxi drivers and other sources). Although these real time data were no longer accessible, the database contains data about the Link_id's of part 1 and it could be very useful to have an idea about the street names of the max speed of each road.

Openstreetmaps and PostGIS database: The data of part two can be visualized by QGIS 2.18 app in cooperation with open source opensteetmap.org. OpenStreetMap (OSM) is a community project involving a world map that is publicly available. The data generated by the project are regarded to be its main output instead of the map itself. OSM's creation and growth has been driven by restrictions to the use or accessibility of map data throughout much of the world, as well as the advent of affordable portable satellite navigation devices. OSM is regarded to be a notable example of voluntary information on geography.

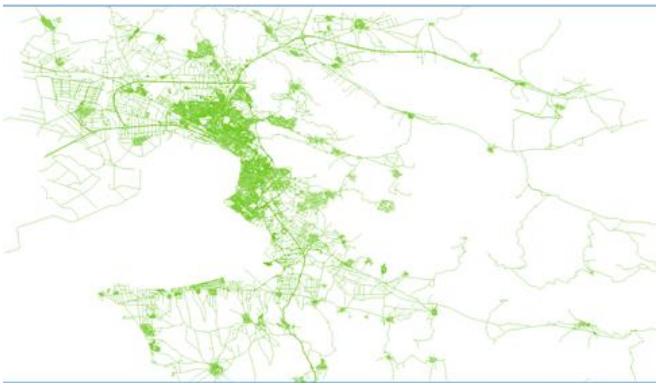

Figure 1: Thessaloniki via QGIS

For this work, we developed a database to store data. Initially, a Thessaloniki table has been created containing data about streets, such as street name, the maximum allowed speed, the road type (main, secondary etc.) as shown in TABLE 1. This table's data originated from the Emisia Wiseride application and contained several information, some of it regarding unknown parameters.

TABLE 1: THESSALONIKI'S IMPORTANT PARAMETERS DETAILED DESCRIPTION.

| Variable | Type | Description |
|---|---|---|
| id | NUMBER(15) UNIQUE NOT NULL | Id of the part of the road of the wiseride app |
| osm_id | NUMBER(15) UNIQUE NOT NULL | Id of the open street map part of the road. This matches with the traffic data, therefore it is the part of the road for traffic congestion prediction. |
| osmname | VARCHAR2(100) | Name of the road |
| osm_source _id | NUMBER(15) | The first osm_id of the road |
| osm_target _id | NUMBER(15) | The last osm_id of the road |
| clazz | NUMBER(5) | Unknown |
| flags | NUMBER(1) | Unknown |
| source | NUMBER(15) | Unknown |
| Target | NUMBER(15) | Unknown |
| Length | FLOAT | The length of each Id |
| Kmh | NUMBER(3) | Top speed |
| Cost | FLOAT | Unknown |
| reverse_cost | FLOAT | Unknown |
| x1 | FLOAT | Map coordinates in X axis, where the Id starts |
| y1 | FLOAT | Map coordinates in Y axis, where the Id starts |
| x2 | FLOAT | Map coordinates in X axis, where the Id ends |
| y2 | FLOAT | Map coordinates in Y axis, where the Id ends |
| category | VARCHAR2(10) | Type of the road |

There were several important notes to be mentioned.

- An osm_id included several id's
- In most of the cases an osm_id was the part of the road that describes the distance from one traffic light to the next traffic light
- An Id described the road for one block.
- There were several street categories as presented in TABLE 2.

TABLE 2:: STREET CATEGORIES

| | | |
|---|---|---|
| KS_2K | Main collector road | 2-way without median strip |
| DA_2K | secondary arterial road | 2-way without median strip |
| KS_1K | Main collector road | One-way or two-way with median strip |
| KT600 | unknown | unknown |
| KT100 | unknown | unknown |
| DS_XX | Secondary collector road | unknown |
| DA_1K | secondary arterial road | One-way or two-way with median strip |
| KA_1K | main arterial road | One-way or two-way with median strip |
| KA_2K | main arterial road | 2-way without median strip |

Also, a very important table was the traffic congestion table (TABLE 3). This table has several variables. In order for this table to be filled, two data extracting methods were developed.

- In early stages of the project, data were extracted from http://opendata.imet.gr/dataset/network-congestion into a csv file or copied to a txt file as a Json array. Later, all these files were parsed using python 3.6 with Spyder IDE. As mentioned, each record of the json array contains information about Link_id, Link_Direction, Timestamp and Congestion. The Link_id, Link_Direction and Timestamp provided the unique LINK_TIMESTAMP_ ID of this table, while the other values were extracted as they were.
- In order to have the best possible algorithm for traffic prediction, the amount of data required was vast. To achieve this, an algorithm has been developed in order to extract traffic data from the aforementioned API and insert them to the table avoiding duplicate records. The link was renewed almost every 15 minutes, so the algorithm could extract data every quarter of an hour. In total, 828897 records have been inserted.

The overall extraction phase was completed during August–October 2019. Our SQL and Python code can be found here: https://github.com/ArisMyst/ParseTrafficData

TABLE 3: TRAFFIC CONGESTION TABLE

| Variable | Type | Description |
|---|---|---|
| LINK_ TIMESTAMP _ID | VARCHAR2( 100) UNIQUE NOT NULL | Unique Id of the open street map part of the road and timestamp |
| LINK_ID | NUMBER(10) NOT NULL | Id of the open street map part of the road. This matches with the osm_id from Thessaloniki table |
| LINK_ DIRECTION | NUMBER(1) NOT NULL | If it has 2 directions, it is a two-way road |
| TIME_STAMP | TIMESTAMP | Exact time of the response data. The provided data renewed every 15 minutes |
| CONGESTION | VARCHAR2( 10) | Low, Medium, or High |

*B. Data preprocessing*

The data preprocessing phase was one of the most challenging parts of this project. In order to use classification for traffic congestion forecasting all data required preprocessing.

A problem is related with the max allowed speed for each road. New parameters had to be created for non-continuous values providing details about the road, such as road length and sequence.

The idea was to combine information from Tables 1, 2 and 3 and discretize continuous values. For example, the timestamp was to be divided into different time slots, like per 15 minutes or per day of the week or per shopping hours, as explained in the following paragraphs.

Timestamp. An empirical evaluation about shopping - office hours and traffic congestion involved interviewing several employees and freelancers motivated us to divide the 24-hour daily cycle into 4 segments. This segmentation is variable and depends on the type of day and store/office opening hours.

- From 09:00 to 20:59 on Tuesdays, Thursdays, and Fridays the stores are 'open'.
- From 09:00 to 17:59 on Mondays, Wednesdays, and Saturdays the stores are 'open'.
- From 07:30 to 08:59 every day except Sundays the stores are 'opening'.
- From 21:00 to 22:00 on Tuesdays, Thursdays, and Fridays the stores are 'closing'.
- From 18:00 to 19:00 on Mondays, Wednesdays, and Saturdays the stores are 'closing'.
- In all other cases the stores are 'closed'.

Another important segmentation we performed, was based on the timestamp providing the day of the week. The prediction was modeled based on a weekly cycle model.

Road length and sequence. As already mentioned, an osm_id or link_id (they were the same object) contained several simple ids (TABLE 1 - TABLE 3). It was easy to identify what ids contained a link_id using the database. However, the difficulty occurred when the sequence of osm_id was needed. To achieve that, the starting and the end point of each osm_id needed to be identified.

Also, the sequence and the location of simple ids was known via x1, x2, y1 and y2 (Table 1). So, the end of each link_id was like where the spot of the end of id was while the link_id was not the same. A similar procedure was applied to find the overall road length of link_id.

Also, as far as the road length is concerned, the average link_id road length was almost 200 meters. For this reason, a new categorical value was created to characterize if a link_id is over 200 meters or not.

*C. Data Mining / Machine Learning*

*1) Selected data*

The data selected were:

- Time
- Day
- Stores
- Congestion
- Osm_id
- Road larger than 200 meters
- Max kilometers allowed on the road
- Road category

Traffic prediction was attempted for one of the busiest roads in Thessaloniki, Tsimiski St., the part with osm_id corresponding to Tsimiski intersections with Venizelou St. and Dragoumi St. This part is not very long and only few traffic records for this osm_id were extracted. This was a common issue regarding data extracted for the whole city of Thessaloniki.

In Fig. 2**Error! Reference source not found.**, green color describes all the osm_ids that Thessaloniki has, while brown color describes the osm_ids that contained more than 10 traffic congestion timestamp records. As it can been seen, there were several roads that did not contain enough data. Similarly, the traffic congestion data about the 13769164 osm_id were not enough. For this reason, the selected data would cover exactly two preceding and one following osm_id, both having sufficient data. These 3 extra osm_id's were 197107696,176665188 and 174019380.

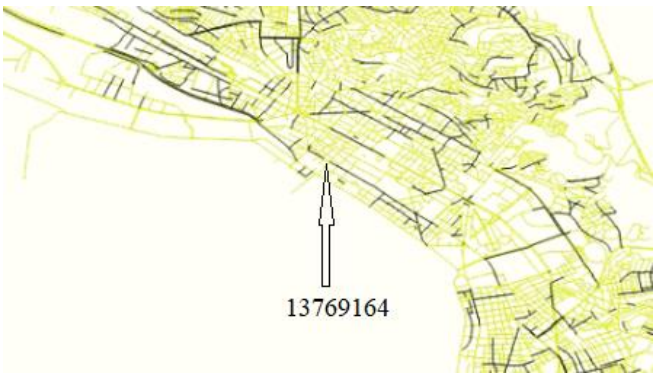


Figure 2: All OMS_ID's vs OMS_ID's with traffic data vs 13769164 ID

As far as the selection is concerned, low congestion output was 0, medium output was 1 and high congestion output was 2. Also, there was a chance that each osm_id contained several different street categories (TABLE 2). This may occurred because each osm_id described several ids and some of them could be in different road categories. An osm_id is assigned the road category with the most occurrences for this specific osm_id.

*2) Initial Prediction Algorithm*

In general, categorical data work well with Decision Trees [23], [24], [25] so all the available parameters in the traffic dataset were discretized. As a result, a decision tree classifier was developed for traffic congestion forecasting using Python's 3.6 Scikit learn library with Jupyter Notebook. In a later step, the same decision tree algorithm was developed utilizing Big Data technologies with Pyspark and Mllib library.

Initially, the exported data in csv format were parsed using Pandas library. The extracted records were 2155 in total. As a first step, removal and targeting of congestion column was done in order for the model to identify the prediction parameter.

The next step was to use label encoder in order to properly categorize the values time, day, open stores, osm_id, road length greater than 200 meters and road category, and pass the information to the model as inputs.

After this, the text variables were removed so the dataset contained only categorical values, as it can been seen in

Figure 3, while the targeted congestion was transformed like the previous categorical values.

| | TIME_n | DAY_n | STORES_n | OSM_ID_n | LARGER_THAN_200M_n | KMH_n | CATEGORY_n |
|---|---|---|---|---|---|---|---|
| 0 | 69 | 1 | 2 | 2 | 1 | 0 | 0 |
| 1 | 88 | 1 | 0 | 2 | 1 | 0 | 0 |
| 2 | 74 | 5 | 2 | 2 | 1 | 0 | 0 |
| 3 | 58 | 6 | 2 | 2 | 1 | 0 | 0 |
| 4 | 83 | 6 | 0 | 2 | 1 | 0 | 0 |

Figure 3: Text variables removed

The next step of this process was to train the algorithm, using 80% of the dataset for training and 20% for testing. Then the tree algorithm, from the sklearn library was used and a decision tree classifier model was created. The model's parameters such as criterion = 'gini index', splitter = 'best' etc. can been seen in Figure 4. This model was trained utilizing the data generated by the training set. Moreover, the model score was very high, reaching 93%

```
from sklearn import tree
model = tree.DecisionTreeClassifier()
model.fit(X_train, y_train)

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

Figure 4: Creating a decision tree classifier

## IV. RESULTS AND EVALUATION

### A. *Decision tree algorithm results*

After the creation of the model, it was time to check the prediction results. For example, it is almost sure that at 04.00 in the morning during a weekday like Wednesday that traffic congestion is low. This case will be mentioned from now on as case 1. The categorical value for 04.00 was 16, while for Wednesday was 6. Also, the categorical osm_id for 13769164 was 0, the kmh parameter 1 and the road category 1 as it can been seen in Fig. 5.

| TIM | DAY | STORES | OSM_ID | LAR | KM | CATEGO | TIME |
|---|---|---|---|---|---|---|---|
| 4:00 | WEDNESDAY | CLOSED | 13769164 | NO | 70 | KS_2K | 16 |

| DAY | STORES | OSM_ID | LARG | KMH | CATEGORY |
|---|---|---|---|---|---|
| 6 | 0 | 0 | 0 | 1 | 1 |

Figure 5: Example for osm_id 13769164

After inserting these labeled parameters in the prediction function, the model's response was 0. That means that the traffic congestion at the aforementioned case 1 was low, which was an expected outcome.

After this prediction, it was time to check the model for a more uncertain case. The model was tested for 20.30 in the afternoon on a Wednesday (case 2). The categorical value for 20.30 was 82, while for Wednesday it was 6. Also, the categorical osm_id for 13769164 was 0, the kmh parameter 1 and the road category 1.

For case 2 the model's response was 1. That means that the traffic congestion was medium, which was expected. As far as the accuracy is concerned, results showed that after importing the necessary metrics, generalization accuracy was 67.8%.

### B. *Improving Prediction Algorithms*

In this stage, several efforts were made for accuracy improvement. First, as far as the splitting criterion is concerned, the default was "gini". This feature provides the user the ability to use different attribute selection measures. One possible attempt to improve accuracy was to choose a different attribute selection measure. Two supported criteria are "gini" (gini index) and "entropy" (information gain).

```
from sklearn import tree
model = tree.DecisionTreeClassifier(criterion='entropy')
model.fit(X_train, y_train)

DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

Figure 6: Choosing information gain

After this, the model score remained comparable. As for the predictions, the results remained the same, 0 (low) for case 1 and 1 (medium) for case 2. However, generalization accuracy dropped to 67.5% from 67.8%. As it seems, utilizing entropy did not increase performance.

For optimizing decision tree performance, another attempt was to change the split strategy of the algorithm. Two supported strategies were "best" to select the best split and "random" to select a random split. The split strategy was changed from "best" to "random" (Figure 7).

```
from sklearn import tree
model = tree.DecisionTreeClassifier(criterion='entropy' , splitter='random')
model.fit(X_train, y_train)

DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='random')
```

Figure 7: Changing the split strategy

After this change accuracy remained roughly the same, however, the results did not remain the same. For case 1 the traffic congestion prediction was 0 (low) and 0 (low) for case 2. However, generalization accuracy dropped to 67.1%. As it can been assumed, utilizing the random split did not improve the algorithm's performance. All in all, implementing the decision tree algorithm with Gini index and best split was the best performed algorithm.

An alternative means to improving accuracy would involve extracting more data. For these reasons, an extra extraction stage was implemented gathering a total number of 4355 records, almost twice as the original data set. The initial decision tree algorithm with Gini index and best split was the best performing algorithm and selected for this attempt (Figure 8).

```
from sklearn import tree
model = tree.DecisionTreeClassifier()
model.fit(X_train, y_train)

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

Figure 8: Decision tree algorithm with Gini index and best split

After this change, the model score dropped to 86.7% and the prediction results remained the same with the 1st algorithm, providing 0 (low) for case 1 and 1 (medium) for case 2.

However, generalization accuracy increased almost to 70%. As it seems, gathering, extracting, and selecting more data and utilizing more information increased the algorithm's performance.

### *C. Decision Tree Vs Logistic Regression Algorithms*

In this phase, we compared Decision trees and Logistic regression in terms of accuracy. As far as the selection was concerned, the only difference was to road length that was set to numerical instead of categorical (Figure 9). After the data were imported via csv parsing, the label encoder was utilized so the data to be categorized.

| | TIME | DAY | STORES | CONGESTION | OSM_ID | ROAD_LENGTH_M | KMH | CATEGORY |
|---|---|---|---|---|---|---|---|---|
| 0 | 17:15 | MONDAY | OPEN | 0 | 176665188 | 985 | 50 | KA_1K |
| 1 | 22:00 | MONDAY | CLOSED | 0 | 176665188 | 985 | 50 | KA_1K |
| 2 | 18:30 | TUESDAY | OPEN | 0 | 176665188 | 985 | 50 | KA_1K |
| 3 | 14:30 | WEDNESDAY | OPEN | 0 | 176665188 | 985 | 50 | KA_1K |
| 4 | 20:45 | WEDNESDAY | CLOSED | 0 | 176665188 | 985 | 50 | KA_1K |

Figure 9: Data for Logistic Regression

Before splitting the algorithm, Logistic Regression from sklearn library was used and a Logistic regression classifier model was created. This model was trained from the training set, as it can been seen in Figure 10.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs_n, target, test_size=0.2,

model.fit(X_train, y_train)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

Figure 10: Creating a Logistic regression classifier and splitting the data

As a result of these changes, the model score was low, close to 66%. Regarding the predictions, the results did not remain the same. The logistic regression algorithm provided 1 (medium) for case 1 and 1 (medium) for case 2.
What is more, the generalization accuracy dropped to 67.4% compared to 70% for Decision Trees. As it was realized, utilizing a Logistic regression algorithm was not the best option when using the specific dataset (Figure 11).

```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.6739380022962113
```

Figure 11: Accuracy while utilizing Logistic regression

### *D. Big Data Vs Data Mining*

So far it seems that the size and quality of the input information is very significant to the model's accuracy. The data collection process only lasted for about 2-3 months. So, what happens if the data collection process lasts for years or if we have significantly more data?
In this step, the so far developed and optimized sklearn Decision tree algorithm was recreated utilizing Big Data technologies like spark and Mllib library and compared to sklearn's Decision tree algorithm based on their accuracy.
By default, pyspark's Mllib maximum different inputs of a parameter (maxBins) are 32. The 'time' parameter used in the original sklearn model had 96 different inputs (24 hours x 4 quarters of an hour). In order for the big data model to work, the 'time' parameter was split into 2 different parameters: 'Hour of the day' and 'Quarter of the hour'.
Besides this difference the overall process of developing the model was quite similar, targeting the value for prediction, properly categorizing the string data and developing the model. The data columns were combined as features array and the target columns as label 'CONGESTION' (Figure 12).

```
+----------+--------------------+
|CONGESTION|            features|
+----------+--------------------+
|         0|[3.0,0.0,1.0,1.0,...|
|         0|[6.0,1.0,1.0,1.0,...|
|         0|[13.0,2.0,1.0,1.0...|
+----------+--------------------+
```

Figure 12: Big Data Decision Tree Input

The data were split into training and test set and the Decision tree model was used utilizing both 'gini' and 'entropy' criterion. (Figure 13).

```
dt = DecisionTreeClassifier(
    labelCol="labelIndex", featuresCol="features", impurity='entropy')
model = dt.fit(trainingData)
```

Figure 13: Big Data Decision Tree with entropy criterion

As far the evaluation was concerned the generalization accuracy increased to 71% with 'gini' index and 73% with 'entropy'. As it seems, the accuracy was almost equal to the previous model, however splitting the 'Time' parameter into 'Hour of the day' and 'Quarter of the hour' could have increased the algorithm's performance.

```
evaluator = MulticlassClassificationEvaluator(
    labelCol="labelIndex", predictionCol="prediction",metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(accuracy)

0.7326968973747017
```

Figure 14: Big Data Decision Tree with entropy criterion accuracy

TABLE 4 summarizes generalization accuracy results for various algorithms, parameter settings and python libraries used.

TABLE 4: CLASSIFIER'S RESULTS

| Classifier | Criterion / Split | No of parameters | Data Records | Python Library | Accuracy % |
|---|---|---|---|---|---|
| Decision Tree | Gini / Best | 7 | 2155 | Scikit learn | 67.8 |
| Decision Tree | Entropy / Best | 7 | 2155 | Scikit learn | 67.5 |
| Decision Tree | Gini / Random | 7 | 2155 | Scikit learn | 67.1 |
| Decision Tree | Gini / Best | 7 | 4355 | Scikit learn | 70 |
| Logistic regression | Best | 7 | 4355 | Scikit learn | 67.4 |
| Decision Tree | Gini / Random | 8 | 4355 | PySpark Mllib | 71 |

| | | | | | |
|---|---|---|---|---|---|
| Decision Tree | Entropy / Random | 8 | 4355 | PySpark Mllib | 73 |

## V. CONCLUSIONS AND FUTURE WORK

### A. Conlusions

This final section summarizes our conclusions. This research was conducted to manage and predict in a low costly but accurate way the traffic congestion in a not very organized, from a public transport infrastructures point of view, city using Thessaloniki as a case study. The example presented was for one of the busiest city streets: Tsimiski St.

After collecting, storing, and pre-processing data from an API for 3 months (August, September, October 2019) using SQL, Python, Geo information system, Machine learning and Big data technologies, a decision tree classification model for predicting the traffic congestion was developed and was compared to other classification algorithms, like logistic regression.

The model was tested to predict the traffic congestion, providing results for 2 different timestamps (04.00 and 20.30 in Wednesday) in a very busy road. While the outcomes were the expected (low congestion in 04.00 and medium in 20.30), extra testing adjusting split strategies, criterions were made to increase accuracy. For this reason, six different tests were conducted with different strategies proposed and applied for traffic congestion prediction. When the algorithm compiled with more extracted data (a second phase of data gathering was done and used for training the model), the model provided a better accuracy. Moreover, when the same data were further parsed to be adjusted to big data technologies generalization accuracy was further increased (71.1%).

The result of these attempts shown that was the quality and the size of the input data have a significant role in the model's accuracy and results. Besides the importance of the data, these experiments showed that the Decision Tree Classifier was more accurate than Logistic Regression.

Finally, this analysis highlighted that there could be still an accurate model with over 71% accuracy without the costly data gathering procedure using cameras and other detectors presented in section II.

### B. Challenges

Although a specific data set was used, the data extraction and pre-processing phase was challenging. The first challenge was to find the data needed to complete the analysis. The goal was to retrieve sufficient information to reflect traffic conditions.

After that important decision we needed to select the most appropriate algorithm for traffic prediction. Generally, categorical data work well with Decision Tree algorithms, while continuous data work well with Logistic Regression algorithms [23], [24], [25]. All the parameters provided were discretised and the Decision Tree algorithm was selected.

### C. Future work

Such research requires time-consuming analysis and different approaches. It is hard to explore every aspect. However, there are several ideas that could be implemented in the future.

Regarding data, one idea would be to alter the pre-processing phase by adding extra normalization, such as hot encoding. This feature changes all the categorical data to Boolean adding columns to all possible parameters. Moreover, another aspect of the data part would be to use the label encoder, but within specific ranges (for example 0 to 1 or Z scores) to avoid the weight in cases like multiple categorical values for the same parameter (this would work better with timestamp categorical values). Furthermore, regarding the numerical values (road length in our case), there could be a standardization pre-processing phase. We could also try different classifiers to see if accuracy improves, as for instance in benchmark work reported in [23, 24, 25]. For example, a different approach would be a combination of one hot encoded and labelled encoded within 0-1 range categorical values, followed by standardized numerical values utilizing different machine learning classifier (e.g. Naïve Bayes, Support Vector Machine, Random Forest, Neural Network etc.)

What is more, it is already mentioned, a key feature of a smart city is efficient traffic flow management throughout the city, that could boost transportation networks flow and optimize traffic conditions for people and cities in general. As the population grows, there are traffic issues, increased emissions, and environmental and economic issues. Because of the above, the utilization of smart traffic lights is among the most relevant strategies used by smart cities to come face to face regarding increasing traffic congestion problems. An idea would be to use the prediction results utilizing a smart traffic lights framework that would operate dynamically based on the predicted traffic congestion.


## ACKNOWLEDGMENT

We would like to thank EMISIA SA (https://www.emisia.com/) for providing access to significant data.


## REFERENCES


[1] Zhanga, K. & Batterman, S., (2013). *Air pollution and health risks due to vehicle traffic*. Science of The Total Environment.

[2] United Nations. World Urbanization Prospects: The 2018 Revision. 2018.

[3] Bertot, J.C. & Choi, H., (2013). *Big data and e-government: issues, policies, and recommendations.*

[4] West, D.M. (2012). Big Data for Education: Data Mining, Data Analytics, and Web Dashboards.

[5] U.S. Department of Energy, Smart Grid / Department of Energy, Retrieved Sep. 29, 2019

[6] Christantonis, K. & Tjortjis C. *Data Mining for Smart Cities: Predicting Electricity Consumption by Classification*, Proc. 10th IEEE Int'l Conf. on Information, Intelligence, Systems and Applications (IISA 2019), pp. 67-73, 2019.

[7] Christantonis, K., Tjortjis C., Manos A., Filippidou D. & Christelis E. (2020). *Smart Cities Data Classification for Electricity Consumption*

*& Traffic Prediction*, Automatics & Software Enginery, 31(1), 49-69

[8] Aguilera, G., Galan, J.L., Campos, J.C. & Rodríguez, P. (2013). *An Accelerated-Time Simulation for Traffic Flow in a Smart City*.

[9] Thianniwet, T., Phosaard, S. & Pattara-Atikom, W. (2009). *Classification of Road Traffic Congestion Levels from GPS Data using a Decision Tree Algorithm and Sliding Windows*.

[10] Christantonis, K., Tjortjis C., Manos A., Filippidou D., Mougiakou E. & Christelis E. (2020). *Using Classification for Traffic Prediction in Smart Cities*, Proc. 16th IFIP Int'l Conf. on Artificial Intelligence Applications and Innovations (AIAI 20), pp. 52-61.

[11] Theodorou, T.I., Salamanis, A., Kehagias, D., Tzovaras, D. & Tjortjis C. (2017). *Short-Term Traffic Prediction Under both Typical and Atypical Traffic Conditions using a Pattern Transition Model*, Proc. 3rd Int'l Conf. Vehicle Technology and Intelligent Transport Systems (VEHITS 17), pp. 79-89.

[12] Pattara-Atikom, W. & Peachavanish, R. (2007). *Estimating Road Traffic Congestion from Cell Dwell Time using Neural Network.*

*[13]* Pongpaibool, P., Tangamchit, P. & Noodwong, K. (2007). *Evaluation of Road Traffic Congestion Using Fuzzy Techniques.*

*[14]* Porikli, F. & Li, X. (2004). *Traffic congestion estimation using hmm models without vehicle tracking.*

[15] Lu, J. & Cao, L. (2003) *Congestion evaluation from traffic flow information based on fuzzy logic*.

[16] Krause, B. & von Altrock, C. (1996). *Intelligent highway by fuzzy logic: Congestion detection and traffic control on multi-lane roads with variable road signs.*

*[17]* Alessandri, R. B. A. & M. Repetto. (2003). *Estimating of freeway traffic variables using information from mobile phones.*

[18] Lomax, T., Turner, S., Shunk, G., Levinson, H.S., Pratt, R. H., Bay, P. N. & Douglas B. B. (1997). *Quantifying Congestion. Volume 1: Final Report.*

[19] Bertini, R. L. (2005). *Congestion and Its Extent.*

[20] Mitsakis, E., Salanova, J. M., Chrysohoou, E. & Aifadopoulou G. (2015). *A robust method for real time estimation of travel times for dense urban road networks using point-to-point detectors.* Transport 30(3) 2015, pp. 264-272. Special Issue on Smart and Sustainable Transport. [DOI:10.3846/16484142.2015.1078845]

[21] Mitsakis, E., Stamos, I., Salanova, J.M. G., Chrysohoou, E. & Aifadopoulou, G. (2013). *Urban Mobility Indicators for Thessaloniki.* Journal of Traffic and Logistics Engineering. (JTLE) (ISSN: 2301-3680), Vol. 1 No. 2, June 2013. pp. 148 – 152. [DOI:10.12720/jtle.1.2.148-152]

[22] Salanova, J. M., Chaniotakis, E., Mitsakis, E., Aifandopoulou, G. & Bischoff J. (2016). *Mobile data for transportation*. Mobile Data, Geography, LBS, 29/06 - 01/07 Tartu, Estonia

[23] Tjortjis C., Saraee M., Theodoulidis B., Keane J.A, 'Using T3, an Improved Decision Tree Classifier, for Mining Stroke Related Medical Data', Methods of Information in Medicine, Vol. 46, No. 5, pp. 523-529, 2007,

[24] Zhang S., Tjortjis C., Zeng X., Qiao H., Buchan I., and Keane J., 'Comparing Data Mining Methods with Logistic Regression in Childhood Obesity Prediction', Information Systems Frontiers Journal, Vol. 11, No. 4, pp. 449-460, 2009, (Springer),

[25] Tatsis V.A., Tjortjis C., Tzirakis P., 'Evaluating data mining algorithms using molecular dynamics trajectories', Int'l Journal of Data Mining and Bioinformatics (IJDMB), Vol. 8, No. 2, pp. 169-187, 2013, (Inderscience),