Completing the Open Street Map Database

Romain PATRY, Gabriel LAFONT--MILLET, Alexis BALESTRA

Introduction :

We aim to focus on the open street map data. This dataset is composed of a lot of position in the earth (represented by latitude and longitude) called nodes, that each have tags and values representing what is in this position. For exemple we can have those tags for a crosswalk on a highway :

```
<node id="368008" lat="48.8493607" lon="2.3953046" version="1">
    <tag k="bicycle" v="yes"/>
    <tag k="button_operated" v="no"/>
    <tag k="crossing" v="traffic_signals"/>
    <tag k="highway" v="traffic_signals"/>
    <tag k="segregated" v="yes"/>
    <tag k="tactile_paving" v="yes"/>
    <tag k="traffic_signals:sound" v="no"/>
</node>
```

In this dataset filled collaboratively, there are many gaps, nodes who don't have every tag listed (for instance the tactile paving or if it is button operated could be missing). We aim to use machine learning to predict what should be in those gaps. This would be a practical application, since we could actively complete the database.

This would have several use cases in real life, indeed not having gaps in the data would help train new machine learning model on it for other applications. Accurate, comprehensive data on crosswalks and their accessibility would enable city planners to identify areas needing infrastructure improvements, such as adding wheelchair-friendly crosswalks or improving accessibility for people with disabilities. This could lead to the creation of more inclusive cities where mobility is optimised for everyone. Yet another application would be to help GPS and navigation systems. Sometimes these systems have outdated or incomplete data, which can lead to suboptimal performance. With more complete and accurate datasets, navigation apps could offer better route planning, avoiding obstacles or inaccessible routes, and providing more efficient travel options for users with specific needs.

We are first trying to fill in the boolean tags such as whether a pedestrian crossing has wheelchair accessibility or if it has touchpad pedestrian crossing or not when such things are not documented. Eventually, we would like to use this technique to fill most or all missing tags in the Open Street Map Database even if those are not boolean, but for now it is better to start with something simple but potentially very useful for disabled people. We will be using primarily the latitude and longitude, and in some models we will also use the other models that we have accessed.

Dataset and Features

We extracted our dataset from the open street map dataset. We extracted only the crosswalk informations from Paris, this is because it will be easier to make our model generalise from one part of Paris to another and Paris is big enough to give us a large enough dataset. Indeed, there are 140 417 crosswalk nodes in our dataset. We extract XML data using the Python library `xml.etree.ElementTree` (documentation: https://docs.python.org/3/library/xml.etree.elementtree.html). This involves creating a tree structure and traversing it using a depth-first search approach to extract the required information.

However, loading all the nodes into memory at once, such as those representing data for Paris, exceeded the available RAM. To address this limitation, an iterative tree traversal method was implemented. This approach avoids loading the entire dataset at once and instead processes the nodes incrementally.

In some cases, multiple passes through the data are required to determine which nodes are useful, but the method remains effective despite these additional steps. Additionally, to manage memory usage efficiently, nodes are not directly destroyed during traversal, as doing so would interfere with Python's iteration mechanism. Instead, the elements are cleared to prevent excessive memory consumption while maintaining the traversal process.

We will be using the following features :

- **longitude and latitude** : the position of the crossing is necessary for applying some models such as k-nearest neighbours
- the tag **traffic_signals:sound** which indicates whether the traffic signals have an audible component
- the tag **crossing:island** which specifies if the crossing include a refuge island halfway across the road
- the tag **crossing:marking** which indicate if there are road marking the pedestrian crossing
- the tag **highway** which represent the type of road
- the tag **crossing** which describe the type of crossing (pedestrian or cyclist)
- the tag **button_operated** which indicated which indicate if there is a button for the traffic signal
- the tag **crossing_ref** which provide additional information about the crossing type
- the tag **kerb** which refers to the design or presence of a curb at the crossing

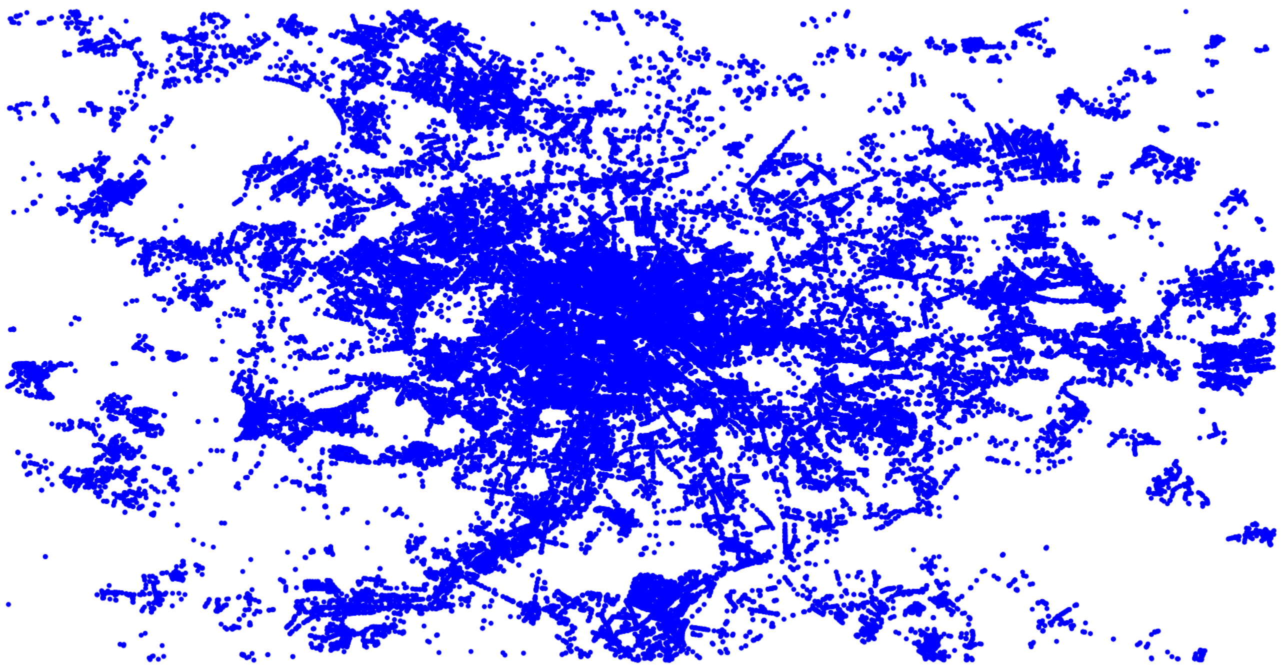
Methods

We are trying to see whether a crossing has tactile paving or not, as such, our problem is a classification problem with boolean values.

We are testing several algorithm, such as the k-nearest neighbour, logistic regression and decision tree classifier.

In order to make analysis of our model easier, we have coded an algorithm that show all crosswalk as nodes which are coloured depending on whether they were properly identified by the model or not. The visualisation of results allow us to more easily understand and use the results of theoretical tools such as the accuracy and F1-score. While it took some time, it makes it easier to identify the potential problem with our models.

Here we can see all the nodes in our dataset



Experiments

We tried a logistic regression algorithm, the model achieved an accuracy of 73%. However, this performance is misleading, as the model always predicted the same class, indicating an over-reliance on class imbalance rather than meaningful learning.

K-Nearest Neighbors (KNN)

We experimented with a k-nearest neighbour algorithm. We tested the algorithm with different elements, we used the longitude and latitude for every model bu we also used different tags to see the results. We put the tags as position (0 or 1) and normalized them.

Here is the accuracy of the different tests :

- the position with the tag **crossing** has an accuracy of **0.8256142506142506**
- the position with the tag **highway** has an accuracy of **0.8253071253071254**
- the position with the tag **button_operated** has an accuracy of **0.8252457002457002**
- the position with the tag **traffic_signals:sound** has an accuracy of **0.8238943488943489**
- the position with the tag **crossing:island** has an accuracy of **0.8216830466830467**
- the position with the tag **crossing_ref** has an accuracy of **0.8214373464373464**
- the position with the tag **kerb** has an accuracy of **0.8209459459459459**
- the position with the tag **crossing:marking** has an accuracy of **0.8193488943488944**
- only the **position** has an accuracy of **0.8256142506142506**
- the position with all the tags has an accuracy of **0.809029484029484**

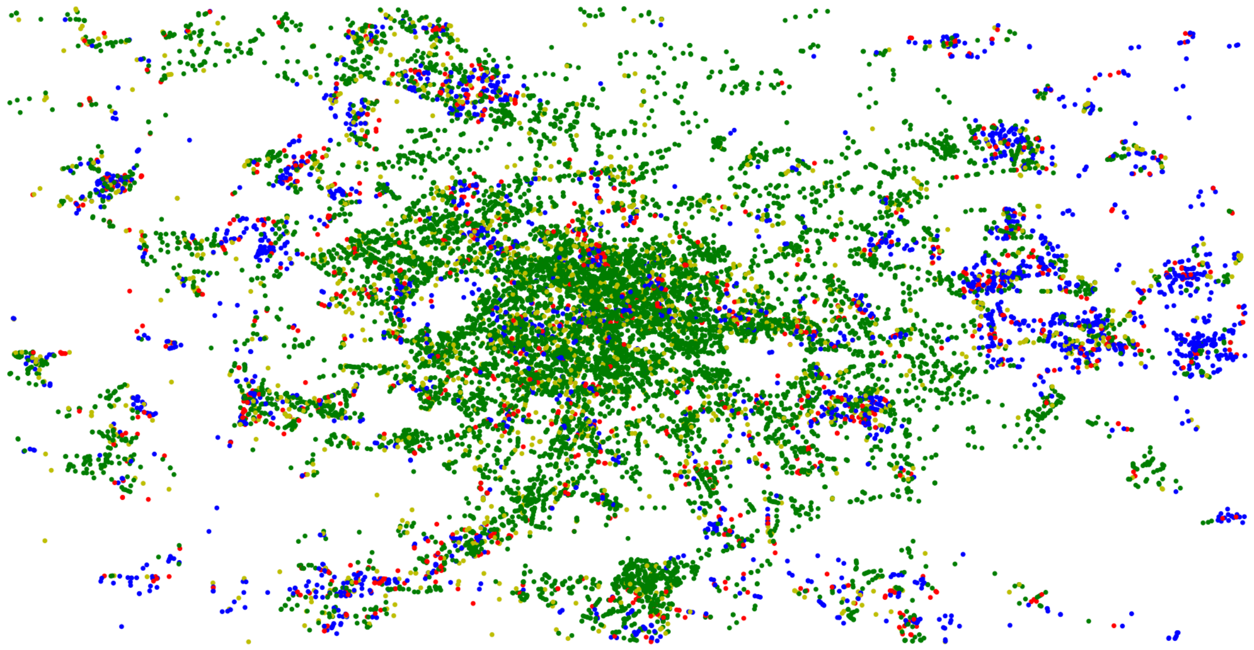
As we can see the addition of tags alongside the position is superfluous and doesn't help to have a better result.

We can have a more in depth look in the results os using position and all tags as well as only position :

1. **Using Position and All Tags**:
   - Precision: 0.68 for class 0 and 0.85 for class 1.
   - Recall: 0.57 for class 0 and 0.90 for class 1.
   - F1-score: 0.62 for class 0 and 0.87 for class 1.
   - Overall accuracy: **81%**.
   - Macro average (average of precision, recall, and F1-score across classes): Precision = 0.76, Recall = 0.73, F1-score = 0.74.
   - Weighted average (weighted by class support): Precision = 0.80, Recall = 0.81, F1-score = 0.80.
2. **Using only position**:
   - Precision: 0.72 for class 0 and 0.86 for class 1.
   - Recall: 0.59 for class 0 and 0.91 for class 1.
   - F1-score: 0.65 for class 0 and 0.88 for class 1.
   - Overall accuracy: **83%**.
   - Macro average: Precision = 0.79, Recall = 0.75, F1-score = 0.77.
   - Weighted average: Precision = 0.82, Recall = 0.83, F1-score = 0.82.

All the measurements indicates that using only the a bit position is better than using the position with all the tags.

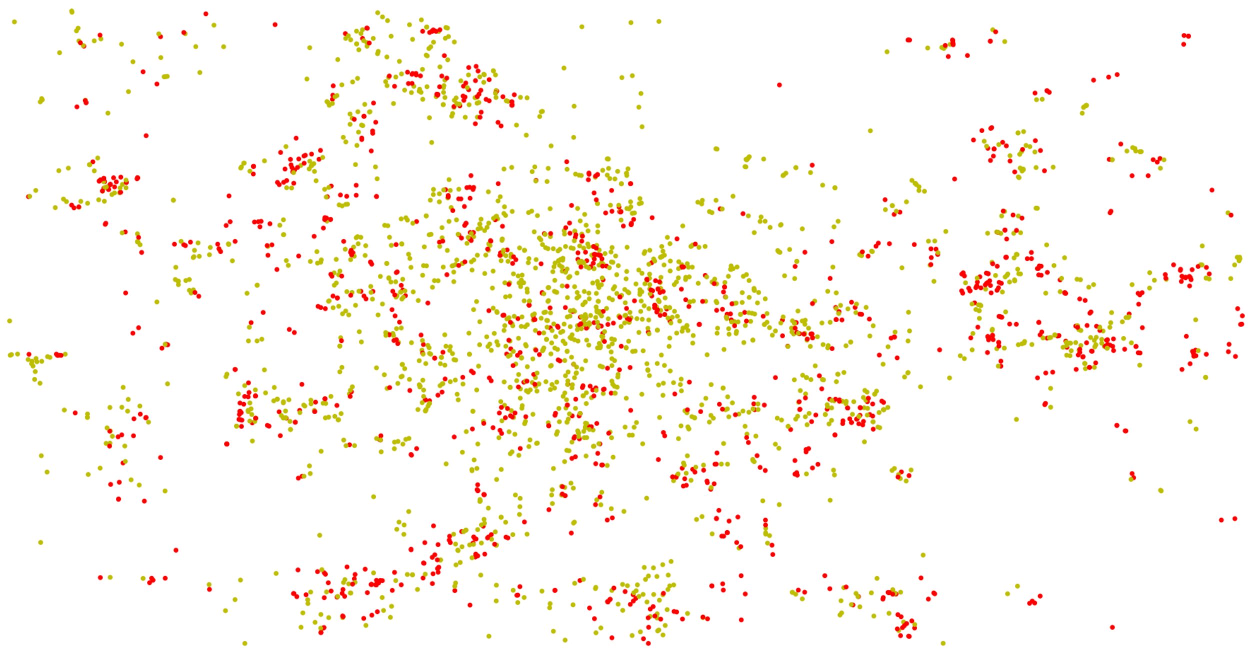
We can visualise the results :



This is the visualisation of the decision tree test nodes with the position and all the tags.

- The green nodes are equipped with a tactile paving and have been identified as such.
- The blue nodes don't have a tactile paving and have been identified as such.
- The yellow nodes have been identified has having a tactile paving without having one.
- The red nodes have a tactile paving but have not been identified as such.

Here we can see only the misclassified nodes :



Decision tree

We did the same experimentation with a decision tree.
Here is the accuracy of the different tests:

- The position with the tag **crossing** has an accuracy of **0.8113022113022113**
- The position with the tag **highway** has an accuracy of **0.8117936117936118**
- The position with the tag **button_operated** has an accuracy of **0.8107493857493857**

- The position with the tag **traffic_signals:sound** has an accuracy of **0.8132063882063882**
- The position with the tag **crossing:island** has an accuracy of **0.8125307125307125**
- The position with the tag **crossing_ref** has an accuracy of **0.8103808353808354**
- The position with the tag **kerb** has an accuracy of **0.8102579852579852**
- The position with the tag **crossing:marking** has an accuracy of **0.811977886977887**
- Only the **position** has an accuracy of **0.812100737100737**
- The position with all the tags has an accuracy of **0.8121621621621622**

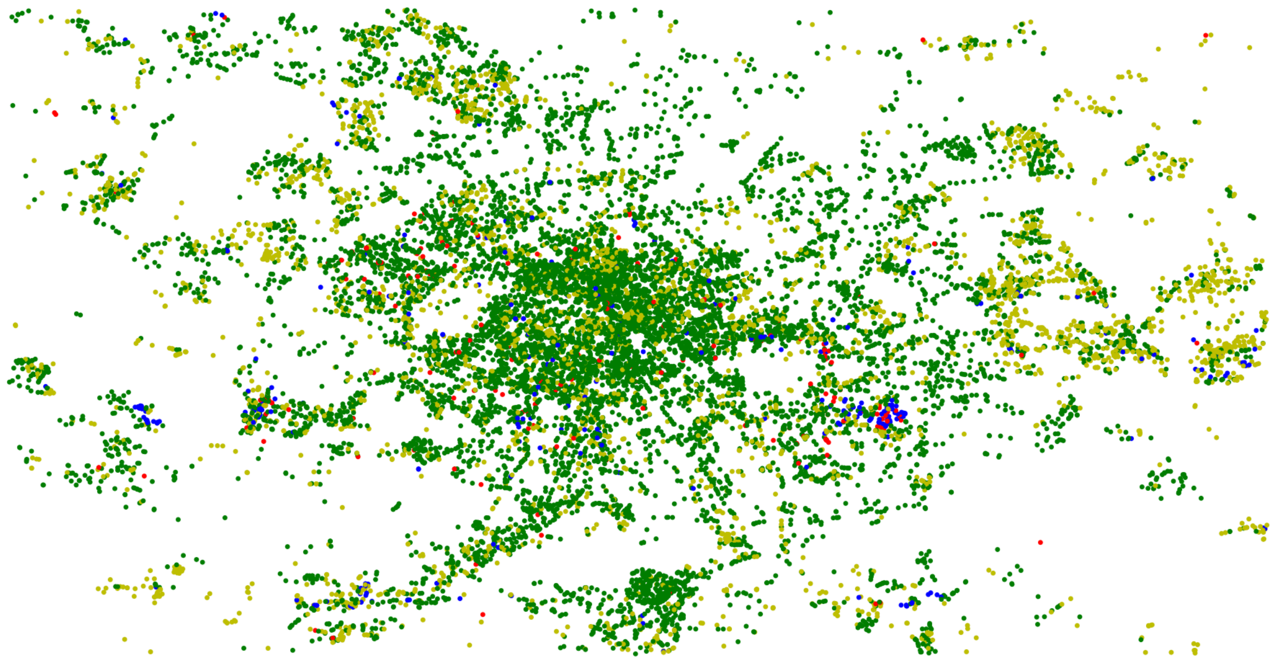
The addition of tags only had a negligible influence on the result of the different tests.

We can have a more in depth look in the results os using position and all tags as well as only position :

1. **Using Position and All Tags:**
    - Precision: 0.66 for class 0 and 0.87 for class 1.
    - Recall: 0.65 for class 0 and 0.87 for class 1.
    - F1-score: 0.65 for class 0 and 0.87 for class 1.
    - Overall accuracy: **81%**.
    - Macro average: Precision = 0.76, Recall = 0.76, F1-score = 0.76.
    - Weighted average: Precision = 0.81, Recall = 0.81, F1-score = 0.81.
2. **Using Only Position:**
    - Precision: 0.66 for class 0 and 0.87 for class 1.
    - Recall: 0.64 for class 0 and 0.87 for class 1.
    - F1-score: 0.65 for class 0 and 0.87 for class 1.
    - Overall accuracy: **81%**.
    - Macro average: Precision = 0.76, Recall = 0.76, F1-score = 0.76.
    - Weighted average: Precision = 0.81, Recall = 0.81, F1-score = 0.81.

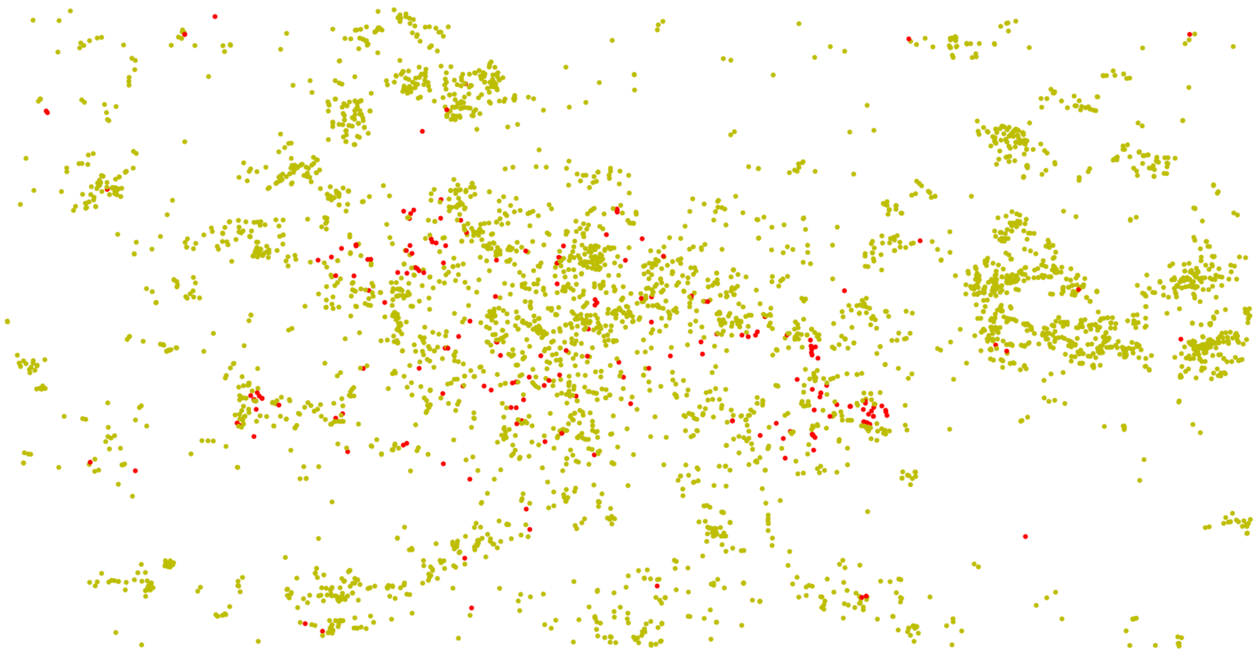
All the measurement indicates that using only the position is as good as using position and all tags.

We can visualize the results :



This is the visualisation of the decision tree test nodes with the position and all the tags.

Here we can see only the misclassified nodes :



Conclusion of the experiments

The K-Nearest Neighbors (KNN) classifier outperformed the Decision Tree Classifier, achieving the highest accuracy (83%) and a strong weighted F1-score (0.82). While Logistic Regression showed poor learning, relying on class imbalance and yielding a lower accuracy of 73%, the Decision Tree Classifier demonstrated a competitive accuracy of 81% with balanced predictions and a weighted F1-score of 0.81. Overall, KNN provided the best performance, especially in handling class `1`, though all models had limitations in addressing class `0`. While the results aren't bad, we have failed to find models that can account for all the data associated with the crosswalks instead of just their position in a meaningful manner. Instead the tag information on the crosswalk have either made no difference or have been detrimental to the result. It is probable that our algorithms detect the places that have the most crosswalk and thus the crosswalks that are the best equipped in Paris.

Next steps

Filling missing values in a dataset is called imputation, we decided to look at the approach others have taken in order to try to take inspiration or improve on the existing methods.

There are two main ways of doing imputation, the first is univariate imputation which uses only the feature of the missing value, for instance replacing all missing values with the mean or median. This approach might be interesting when your goal is to train a machine learning model on the dataset and the correctness of the filled in values won't have a big effect on the model, but it really will not help in our case. Here our goal is a bit different, we want to fill in the missing values in the open street map dataset, but using this dataset for training machine learning algorithms is only one possible use case for our endeavour and having accurate data is our main objective.

The second main way to do imputation, multivariate imputation is more interesting. A regression algorithm is trained to predict missing values from the features that are present, using the relationships between all the features. This process is repeated iteratively, where each feature with missing values is predicted in turn, and the imputation is refined across several rounds to ensure more accurate results. We have tried to apply multivariate imputation to our dataset but we have yet achieved a result that allow us to estimate the accuracy of it and we have more work to do regarding this.

It is unlikely that in the future, new categories for tags are created, instead new tags will be made, however, as cities may change in infrastructure with the arrival of new technologies or new ideas arise in urban development our model will need to be trained again and updated. We are also limiting ourselves to Paris' data for now. We are assuming that what our model will learn will depend on the location of the data, as such focusing on one city for now will make it easier to evaluate our progress, it would be also computationally too expensive to include many cities in our dataset. The potential limits of our model spatially and temporally is a problem that would be interesting to quantify.

Contribution

- **Gabriel LAFONT-MILLET**: Worked on the KNN experiment and the decision tree experiment.
- **Romain PATRY**: Selected the dataset relevant to our problem and developed the algorithm to visualize the model's effectiveness on crosswalks.
- **Alexis BALESTRA**: Authored the project report and worked on the decision tree experiment.